Теория и техника электронной музыки

Миллер Пакетт
University of California, San Diego

участники перевода:

Григорий Амосов
Игорь Кефалиди
Сергей Полтавский

корректура:
Наталья Плотникова

ЦЭАМ (Центр Электроакустической Музыки)
Московская Государственная Консерватория им. П. И. Чайковского

ноябрь 2024 г.

Оглавление

Предисловие **x**

1 Синусоиды, амплитуды и частоты **1**

1.1 Меры амплитуды 3

1.2 Единицы амплитуды 4

1.3 Управление амплитудой 6

1.4 Частота 6

1.5 Синтез синусоиды 8

1.6 Сложение сигналов 11

1.7 Периодические сигналы 13

1.8 О программных примерах 15

1.8.1 Краткое введение в Pd 15

1.8.2 Как найти и запустить примеры 17

1.9 Примеры 18

1.9.1 Умножение амплитуды на константу 18

1.9.2 Управление амплитудой в децибелах 20

1.9.3 Гладкое управление амплитудой с использованием генератора огибающей 21

1.9.4 Мажорное трезвучие 22

1.9.5 Перевод частоты в звуковысотность и наоборот 22

1.9.6 Дополнительные примеры аддитивного синтеза 24

Упражнения 25

2 Волновые таблицы и семплеры **26**

2.1 Таблично-волновой осциллятор 28
2.2 Семплирование 32
2.3 Семплеры с огибающими 35
2.4 Растяжение тембра 39
2.5 Интерполяция 43
2.6 Примеры 47
2.6.1 Осциллятор волновой таблицы 47
2.6.2 Считывание волновой таблицы в общем случае 48
2.6.3 Использование волновой таблицы как семплера 51
2.6.4 Циклические семплеры 51
2.6.5 Циклический семплер с перекрытием 55
2.6.6 Автоматическое изменение точки считывания 55
Упражнения 58

3 Аудиовычисления и управляющие вычисления 59

3.1 Теорема семплирования 59
3.2 Управление 61
3.3 Управляющие потоки 63
3.4 Преобразование аудиосигналов в потоки числового управления . . . 67
3.5 Управляющие потоки в виде блок-схем 69
3.6 Детекция события 69
3.7 Использование аудиосигналов для управления 70
3.8 Операции над управляющими потоками 72
3.9 Управляющие операции в Pd 76
3.10 Примеры 77
3.10.1 Семплирование и наложение спектра 77
3.10.2 Преобразование управляющего сигнала в аудиосигналы . . . 78
3.10.3 Воспроизведение волновой таблицы без зацикливания 80
3.10.4 Сигналы в управления 81
3.10.5 Секвенсер в аналоговом стиле 82

3.10.6 Синтезатор в MIDI стиле 82
Упражнения 83

4 Автоматизация и управление многоголосием 86

4.1 Генераторы огибающей 86
4.2 Линейные и криволинейные амплитудные формы 89
4.3 Непрерывные и прерывистые изменения контрольного управления . 91
4.3.1 Приглушение звука 91
4.3.2 Техника “включить и линейно снизить” 93
4.4 Полифония 95
4.5 Распределение голосов 95
4.6 Метки голосов 96
4.7 Инкапсуляция в Pd 99
4.8 Примеры 100
4.8.1 Генератор огибающей ADSR 100
4.8.2 Функции преобразования для управления амплитудой 102
4.8.3 Аддитивный синтез: колокол Риссе 103
4.8.4 Аддитивный синтез: управление спектральной огибающей . . 106
4.8.5 Полифонический синтез: семплер 107
Упражнения 111

5 Модуляция 112

5.1 Классификация спектров 112
5.2 Умножение аудиосигналов 115
5.3 Вейвшейпинг 120
5.4 Частотная и фазовая модуляция 126
5.5 Примеры 128
5.5.1 Кольцевая модуляция и спектры 128
5.5.2 Деление октавы и добавление формант 129
5.5.3 Вейвшейпинг и разностные тоны 131

5.5.4 Вейвшейпинг, использующий полиномы Чебышёва 132
5.5.5 Вейвшейпинг с помощью экспоненциальной функции 134
5.5.6 Вейвшейпинг синусоидой: чётность и нечётность 136
5.5.7 Фазовая модуляция и FM 136
Упражнения 138

6 Проектирование спектра 141

6.1 Модель несущая/модулятор 142
6.2 Последовательности импульсов 145
6.2.1 Последовательности импульсов через вейвшейпинг 145
6.2.2 Последовательности импульсов через растягивание волновой таблицы 146
6.2.3 Результирующие спектры 148
6.3 Подвижная кольцевая модуляция 151
6.4 Генератор фазово-выровненной форманты PAF 153
6.5 Примеры 158
6.5.1 Импульсная последовательность волновой таблицы 158
6.5.2 Простой генератор форманты 161
6.5.3 Двухкосинусный несущий сигнал 162
6.5.4 Генератор PAF 163
6.5.5 Растянутые волновые таблицы 167
Упражнения 167

7 Временные сдвиги и задержки 168

7.1 Комплексные числа 169
7.1.1 Комплексные синусоиды 171
7.2 Временные сдвиги и изменения фазы 173
7.3 Сети с задержкой 174
7.4 Сети с задержкой и обратной связью 178
7.5 Сохранение мощности и сети с задержкой со сложной структурой . 183
7.6 Искусственная реверберация 188

7.6.1 Управление ревербераторами 190
7.7 Изменяемые и дробные временные задержки 191
7.8 Точность воспроизведения линий задержки с интерполяцией 196
7.9 Сдвиг высоты звука 197
7.10 Примеры 203
7.10.1 Фиксированная линия задержки без интерполяции 203
7.10.2 Гребенчатый фильтр с рециркуляцией 204
7.10.3 Переменная линия задержки 205
7.10.4 Порядок выполнения и нижние пределы времён задержки . 206
7.10.5 Порядок выполнения в линиях задержки без рециркуляции . 209
7.10.6 Гребенчатый фильтр без рециркуляции в качестве октавного удвоителя 210
7.10.7 Изменяющийся во времени сложный гребенчатый фильтр: генератор регулируемых вибраций (шейкер) 212
7.10.8 Ревербератор 213
7.10.9 Питч-шифтер 215
Упражнения 216

8 Фильтры 217

8.1 Классификация фильтров 218
8.1.1 Фильтры, пропускающие низкие и высокие частоты 218
8.1.2 Полосовые и режекторные фильтры 220
8.1.3 Эквалайзеры 220
8.2 Элементарные фильтры 223
8.2.1 Элементарный фильтр без рециркуляции 223
8.2.2 Вторая форма фильтра без рециркуляции 225
8.2.3 Элементарный фильтр с рециркуляцией 226
8.2.4 Составные фильтры 226
8.2.5 Вывод вещественных значений из комплексных фильтров . . 227
8.2.6 Два фильтра с рециркуляцией по цене одного 228

8.3 Проектирование фильтров 229
8.3.1 Однополюсный фильтр низких частот 230
8.3.2 Однополюсный фильтр высоких частот с одним нулём 231
8.3.3 Полочный фильтр 232
8.3.4 Полосовой фильтр 233
8.3.5 Пиковый и режекторный фильтры 234
8.3.6 Фильтры Баттерворта 234
8.3.7 Растяжение единичной окружности рациональными функциями 236
8.3.8 Полосовой фильтр Баттерворта 238
8.3.9 Изменяющиеся во времени коэффициенты 240
8.3.10 Импульсные отклики фильтров с рециркуляцией 241
8.3.11 Всепропускающие фильтры 243
8.4 Применения 244
8.4.1 Субтрактивный синтез 245
8.4.2 Детектор огибающей 246
8.4.3 Однополосная модуляция 248
8.5 Примеры 250
8.5.1 Встроенные в Pd фильтры: полосовой, ФНЧ и ФВЧ 250
8.5.2 Встроенный полосовой фильтр, изменяющийся во времени . 250
8.5.3 Детекторы огибающей 251
8.5.4 Однополосная модуляция 252
8.5.5 Использование элементарных фильтров напрямую: полочный и пиковый фильтры 253
8.5.6 Создание и использование всепропускающих фильтров . . . 255
Упражнения 255

9 Анализ и синтез Фурье 257

9.1 Анализ Фурье периодических сигналов 257
9.1.1 Периодичность преобразования Фурье 259
9.1.2 Преобразование Фурье как аддитивный синтез 259

9.2 Свойства преобразований Фурье 260
9.2.1 Преобразование Фурье DC 260
9.2.2 Сдвиги и изменения фазы 262
9.2.3 Преобразование Фурье синусоиды 263
9.3 Анализ Фурье непериодических сигналов 265
9.4 Анализ Фурье и реконструкция аудиосигналов 268
9.4.1 Узкополосный компандер 270
9.4.2 Наложение тембра (классический вокодер) 272
9.5 Фаза 273
9.5.1 Фазовые соотношения между каналами 276
9.6 Фазовые биения 279
9.7 Примеры 280
9.7.1 Анализ и ресинтез Фурье в Pd 280
9.7.2 Узкополосный компандер: подавление шума 282
9.7.3 Наложение тембра (“вокодер”) 285
9.7.4 Управление временем с помощью фазового вокодера 285
Упражнения 288

10 Классические формы волн 289

10.1 Симметрии и ряды Фурье 290
10.1.1 Пилообразные волны и симметрия 292
10.2 Препарирование классических форм волн 293
10.3 Ряды Фурье элементарных форм волн 295
10.3.1 Пилообразная волна 296
10.3.2 Параболическая волна 297
10.3.3 Квадратная и симметричная треугольная волны 298
10.3.4 Общая (несимметричная) треугольная волна 298
10.4 Предсказание и управление сверткой спектра 301
10.4.1 Пересемплирование 301
10.4.2 “Коварные” треугольные волны 302

10.4.3 Склейка переходов 302
10.5 Примеры 304
10.5.1 Комбинирование пилообразных волн 304
10.5.2 Получение пилообразных волн с ограниченным спектром . . 307
Упражнения 308

Словарь терминов **309**

Предметный указатель **311**

Литература **312**

Предисловие

Эта книга об использовании электронных методов для записи, синтеза, обработки и анализа музыкальных звуков — практики, которая приняла современный вид в 1948–1952 годах, хотя её технологическая составляющая и художественное использование с тех пор претерпели несколько революционных изменений. В настоящее время большинство электронной музыки создаётся с использованием компьютеров, и предлагаемая книга будет сфокусирована исключительно на том, что обыкновенно называют «компьютерной музыкой», но что в действительности должно называться «электронная музыка, использующая компьютер».

Большинство доступных сегодня устройств компьютерной музыки имеют прототипы в предыдущих поколениях оборудования. Компьютер, однако, относительно дешёв и результаты его использования легко документируются и воссоздаются. Как минимум эти преимущества делают компьютер идеальным электронным музыкальным инструментом и трудно представить, чем будущие технологии смогут его заменить.

Методы и практика электронной музыки могут изучаться (по крайней мере, теоретически) без точной отсылки к текущему состоянию технологии. Тем не менее, важно предоставлять работающие примеры. Так что каждая глава начинается с теории (избегая любой ссылки на реализацию) и заканчивается рядом примеров, реализуемых на доступном в настоящий момент программном обеспечении.

Идеальный читатель этой книги тот, кто знает и любит электронную музыку любых жанров, имеет достаточно общих навыков работы с компьютером, и тот, кто хочет научиться создавать электронную музыку, пройдя путь от простого осциллятора до семплирования, частотной модуляции, фильтрации, вейвшейпинга, линий задержки и т. д. Это займёт достаточно много времени.

Мы не пойдем простым путем, рекомендуя уже готовое программное обеспечение для этих целей; вместо этого, упор сделан на изучении того, как использовать среды компьютерной музыки общего назначения для реализации таких методов самостоятельно. Из всех доступных программ мы будем использовать Pd[1], но это не должно остановить вас от использования тех же методов в других программах,

[1]Pure Data — визуальный язык программирования, работающий в парадигме программирования потоков данных, разработанный Миллером Пакетом в 1990-е годы для создания интерактивных компьютерных музыкальных и мультимедийных произведений. Является свободным аналогом Max/MSP.

таких как Csound или Max/MSP.

Для чтения этой книги вы должны понимать математику на среднем уровне алгебры и тригонометрии; начиная с Главы 7, также появляются комплексные числа, хотя и без комплексного анализа. (Например, комплексные числа складываются, умножаются, берётся сопряжённое, но нет комплексных экспонент.) Обзор математики, используемой в компьютерной музыке, Ф. Ричарда Мура можно найти в [Str85, pp. 1-68].

Хотя требуемый уровень математики не высок, математика сама по себе иногда является чем-то чрезвычайно стимулирующим. Все виды «сложной» математики в этой книге вполне доступны для любого студента, изучавшего алгебру или геометрию. Например, в книге встретятся функции Бесселя, полиномы Чебышёва, Центральная предельная теорема и, конечно, анализ Фурье.

Вам не потребуется серьёзная подготовка в области музыки в таком виде; в частности, музыкальная нотная нотация не используется. Используются некоторые элементарные сведения европейской музыкальной теории, такие как равномерно темперированный строй, термины «нота» или «аккорд», обозначение нот латинскими символами. Также вы должны быть знакомы с такими терминами музыкальной акустики, как синусоида, амплитуда, частота, обертонный ряд.

Каждая глава начинается с теоретического обсуждения некоторого набора методов или теоретических подходов, затем для иллюстрации следуют примеры, реализованные в Pd. Примеры включены в справочный пакет, распространяемый вместе с Pd, так что вы можете запустить их и/или редактировать их для своих собственных целей. Кроме того, все рисунки были созданы с использованием патчей Pd, которые имеются в электронном приложении. Для тех кто интересуется, это может служить примером графических возможностей Pd, хотя не все эти возможности еще аккуратно задокументированы.

Я хочу поблагодарить некоторых людей, сделавших возможным для меня написание этой книги. Барри Верко почти полностью ответственен за моё музыкальное образование. В то же время, математику я изучал у Вэйна Холмана, Самуэля Грейтцера, Мюррея Кламкина, Жан-Карло Рота, Франка Моргана, Микаэля Артина, Эндрю Глисона и многих других. Фил Вайт учил меня английскому языку и Рози Песчелл визуальной композиции. Наконец, мои родители (один из которых покойный ныне) были очень терпеливы; теперь мне 47. Спасибо.

Глава 1

Синусоиды, амплитуды и частоты

Электронная музыка обычно создаётся с помощью компьютера путём синтеза или обработки *цифровых аудиосигналов*, которые представляют из себя числовые последовательности

$$\ldots, x[n-1], x[n], x[n+1], \ldots,$$

где коэффициент n (номер семпла) может принимать значение как всех целых чисел, так и отдельной их области[1]. Отдельное число в последовательности называется *семплом*. Примером цифрового аудиосигнала может служить *синусоида*:

$$x[n] = a\cos(\omega n + \phi),$$

где a — это амплитуда[2], ω — угловая (круговая) частота[3] и ϕ - это начальная фаза[4]. Фазой называется функция семпла с номером n, равная $\omega n + \phi$[5]. Начальная фаза — это фаза у нулевого семпла ($n = 0$).

На рисунке 1.1 *a* дано графическое изображение синусоиды. На оси абсцисс показаны последующие значения n, а на оси ординат — соответствующие им значения $x[n]$. График нарисован так, чтобы подчеркнуть семпловую природу сигнала. Как вариант, можно изобразить его как непрерывную кривую *b*. Верхний

[1]Формула читается как «значение x в момент времени $n-1$, значение x в момент времени n, значение x в момент времени $n+1$». Всюду в этой книге используется дискретное время $t = n$, где n принимает целые значения. В формуле приведены три значения, соответствующие трём последовательным значениям времени $t = n-1$, n и $n+1$. Многоточия заменяют значения сигнала в моменты времени, предшествующие $n-1$ и следующие после $n+1$. Предполагается, что момент времени $n = 0$ является первоначальным.

[2]Амплитуда равна максимальной силе, с которой сигнал «синусоида» может действовать в некоторый момент времени.

[3]Частота является мерой скорости изменения величины сигнала во времени. Используется два вида частот: обычная ν и угловая ω (то же, что круговая). Обычная частота ν равна числу колебаний в секунду. Угловая частота $\omega = 2\pi\nu$, то есть число раз, которое аргумент синусоиды пробегает значения 0 до 2π за секунду. Поскольку аргумент синусоиды имеет смысл угла, получаем все возможные значения угла от 0 до $360°$ (что соответствует 2π в радианах).

[4]Начальная фаза устанавливает, насколько величина нулевого семпла $x[0] = a\cos(\phi)$ меньше максимально возможной, равной амплитуде синусоиды a. Так при $\phi = 0$ имеем $x[0] = a$.

[5]Попросту говоря, фазой называется аргумент синусоиды.

рисунок — наиболее верное представление цифровой синусоиды, нижний — скорее идеальный вариант.

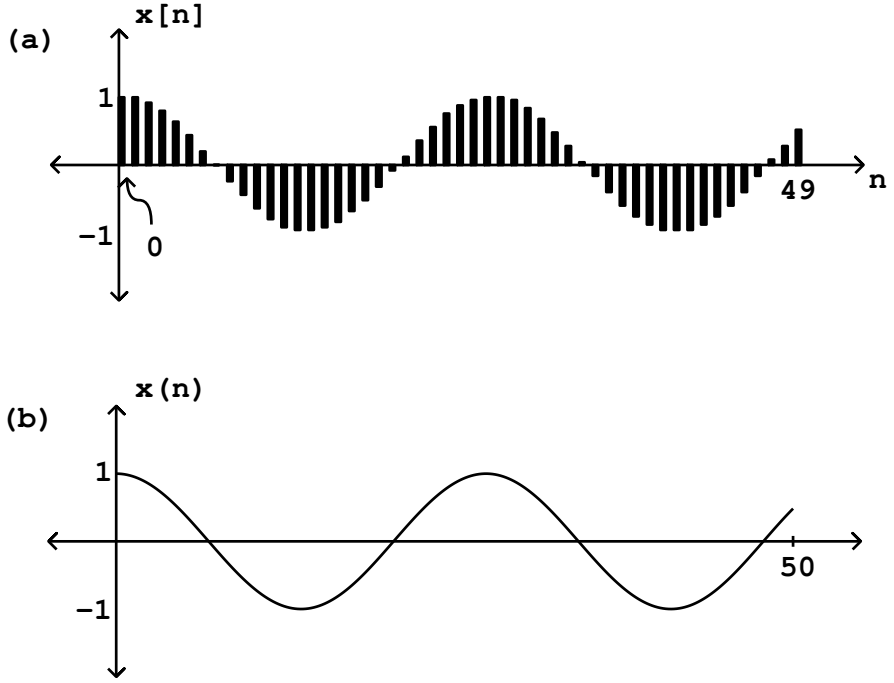


Рис. 1.1: Цифровой аудиосигнал, показанный в его дискретной сущности *a* и идеальный вариант в виде непрерывной функции *b*. Этот сигнал представляет собой вещественнозначную синусоиду, длиною 50 точек, с амплитудой 1, угловой частотой 0.24 и нулевой начальной фазой.

Синусоиды играют ключевую роль в аудиообработке, потому что если сдвинуть одну из них на любое количество семплов влево или вправо, получится другая синусоида. Это даёт возможность быстро рассчитывать эффект любой операции над синусоидами. Наши уши используют ту же самую возможность для анализа поступающих звуков, и поэтому синусоиды и комбинации синусоид могут быть использованы для достижения многих музыкальных эффектов.

Цифровые аудиосигналы сами по себе никак не связаны со временем и для их прослушивания мы должны выбрать *частоту семплирования* (дискретизации), обычно обозначаемую переменной R. Эта величина равна числу семплов, звучащему за секунду. Время t связано с числом семплов по формуле $Rt = n$, или $t = n/R$. Частота (в реальном времени) синусоидального сигнала с угловой частотой ω равна

$$f = \frac{\omega R}{2\pi}$$

в герцах (т. е. колебаниях в секунду), поскольку одному колебанию соответствует 2π радиан, так что в секунде укладывается R семплов.

Амплитуда реального сигнала может быть выражена как электрическое напряжение или давление воздуха, изменяющееся во времени, но семплы цифрового

аудиосигнала — безразмерные величины. Мы будем обычно считать, что численная точность очень высока, и мы можем игнорировать ошибки округления, а также, что цифровой формат имеет неограниченный диапазон, и, следовательно, семплы могут принимать любые значения, которые мы хотим. На самом деле большая часть аудиооборудования работает только с ограниченным диапазоном значений, обычно между -1 и $+1$. Современное программное обеспечение, работающее с цифровым аудио, обычно использует для сигналов представление с плавающей точкой. Это даёт возможность использовать единицы, наиболее удобные для любой поставленной задачи, при том, что выходной сигнал будет в допустимых для оборудования пределах.

1.1 Меры амплитуды

Наиболее фундаментальное свойство цифрового аудиосигнала — амплитуда. К сожалению, амплитуда сигнала не имеет единого канонического определения. Строго говоря, все семплы цифрового аудиосигнала — сами являются амплитудами, но мы также говорим об амплитуде синусоиды как целого. Но также полезно иметь меры амплитуды для цифровых аудиосигналов в целом. Амплитуду лучше всего рассматривать как применение к «окну» фиксированного диапазона выборок сигнала. Например, окно, которое начинается от семпла M, длиной N аудиосигналов $x[n]$, состоит из семплов

$$x[M], x[M+1], \ldots, x[M+N-1].$$

Два наиболее часто используемых варианта измерения амплитуды — *пиковая (максимальная) амплитуда*, то есть просто наибольшее абсолютное значение в пределах окна

$$A_{peak}\{x[n]\} = \max |x[n]|, \qquad n = M, \ldots, M+N-1$$

и *среднеквадратичная амплитуда* (RMS):

$$A_{RMS}\{x[n]\} = \sqrt{P\{x[n]\}},$$

где $P\{x[n]\}$ — это средняя величина мощности, определяемой как

$$P\{x[n]\} = \frac{1}{N}\Big(|x[M]|^2 + \ldots + |x[M+N-1]|^2\Big).$$

(В последней формуле взятие абсолютного значения необязательно, так как мы пока работаем с вещественнозначными сигналами, абсолютное значение величин будет важным позже, когда мы будем рассматривать комплексные сигналы). Ни максимальная, ни среднеквадратичная амплитуда не могут быть отрицательными, и могут быть равны нулю только если весь сигнал равен нулю для всех n в окне.

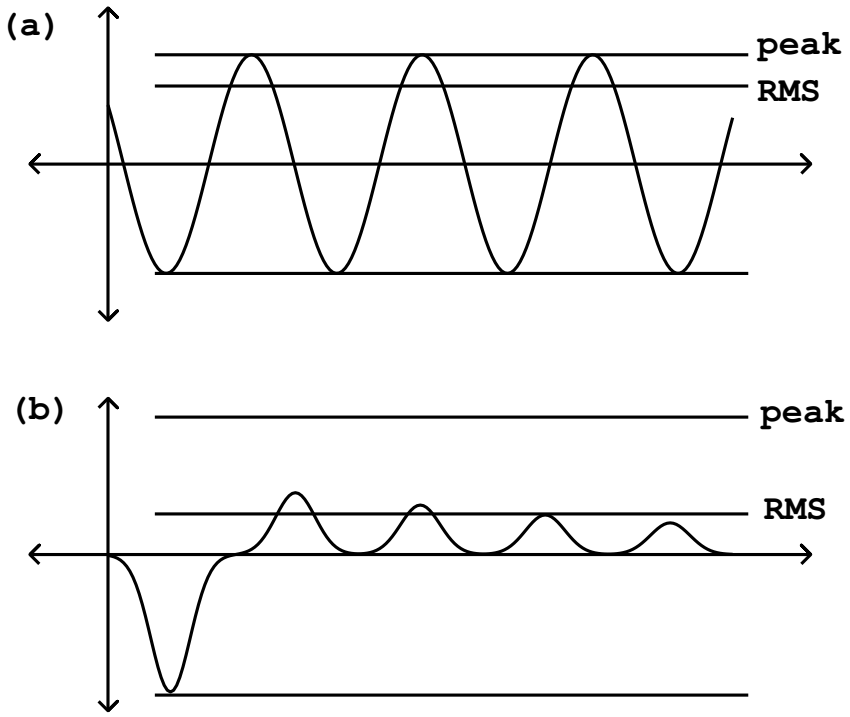


Рис. 1.2: Сравнение среднеквадратичной и максимальной амплитуды сигналов. Для синусоиды a максимальная амплитуда выше среднеквадратичной в $\sqrt{2}$ раз.

Среднеквадратичная амплитуда может быть равна пиковой амплитуде, но не может превосходить ее; и она может быть произведением числа $1/\sqrt{N}$ на максимальную амплитуду, но никогда меньше.

При подходящих условиях (если окно содержит как минимум несколько периодов волны, и если угловая частота существенно меньше одного радиана на семпл), максимальная амплитуда синусоиды равна приблизительно a и ее среднеквадратичная амплитуда будет около $a/\sqrt{2}$. Рисунок 1.2 показывает максимальную и среднеквадратичную амплитуду двух цифровых аудиосигналов.

1.2 Единицы амплитуды

Две амплитуды часто лучше сравнивать по их отношению, а не по их разности. Фраза «амплитуда одного сигнала больше амплитуды другого в два раза» информативнее, чем «амплитуда одного сигнала больше другого на 30 милливольт». Это верно и для максимума, и для среднеквадратичного варианта измерения амплитуды. Для упрощения расчётов амплитуда часто выражается в логарифмических единицах — в децибелах[6]. Если a — это амплитуда сигнала (максимальная

[6] Децибел — это не абсолютная величина, как, например, ватт или вольт, а такая же относительная, как кратность («трёхкратное отличие») или проценты, предназначенная для измерения

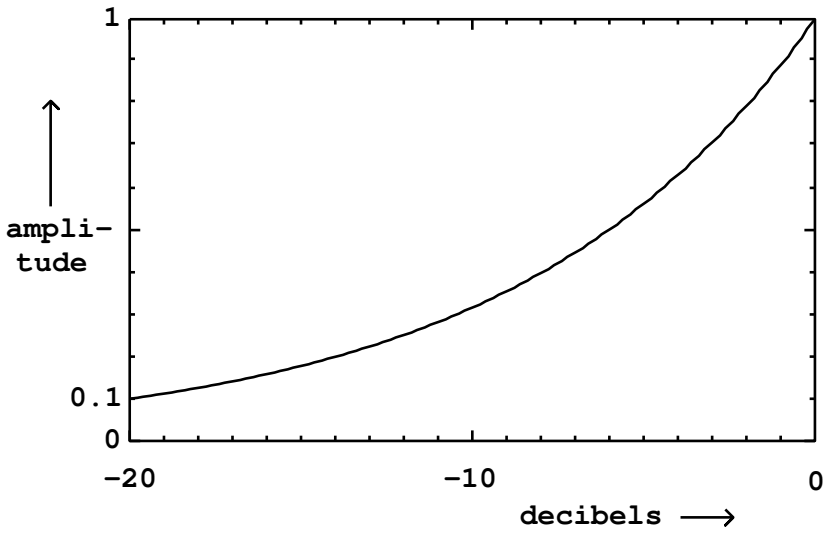


Рис. 1.3: Соотношение между децибелами и линейной шкалой амплитуды. Линейной амплитуде 1 соответствует 0 дб.

или среднеквадратичная), то мы можем определить уровень d в децибелах (dB) следующим образом:

$$d = 20 \cdot \log_{10}(a/a_0),$$

где a_0 — опорная амплитуда, соответствующая началу координат. Это определение устроено так, чтобы если мы увеличиваем мощность сигнала в десять раз (и амплитуда, соответственно, увеличивается в $\sqrt{10}$ раз), логарифм увеличится на 1/2, и, таким образом, значение в децибелах возрастёт на десять[7]. Увеличение амплитуды в два раза соответствует увеличению приблизительно на 6.02 децибела,[8] так что удвоению мощности соответствует 3.01 децибела[9]. Соотношение между линейной амплитудой и амплитудой в децибелах показано на рис. 1.3.

При использовании a_0 для обозначения опорного уровня амплитуды, сигнал с амплитудой, меньшей, чем a_0, получит отрицательное значение в децибелах: $a_0/10$ даёт -20 дб, $a_0/100$ даёт -40 дб, и так далее. Линейная амплитуда нуля меньше, чем любое значение, выраженное в децибелах, поэтому мы считаем его уровень равным $-\infty$.

В цифровом аудио принят следующий опорный уровень амплитуды (учитывая, что максимальная амплитуда для аудиооборудования — единица):

$$a_0 = 10^{-5} = 0.00001,$$

поэтому максимально возможная амплитуда — 100 дБ, а 0 дБ будет неслышимо при любом приемлемом уровне громкости. Довольно удобно, что динамический

отношения двух других величин, причём к полученному отношению применяется логарифмический масштаб.

[7] $20 \cdot \log_{10}(\sqrt[2]{10}) = 20 \cdot \frac{1}{2} = 10$ (*прим. перев.*)

[8] $20 \log_{10} 2 \approx 20 * 0.301 = 6.02$

[9] $20 \log_{10}(\sqrt[2]{2}) \approx 20 * 0.301 * 0.5 = 3.01$

диапазон человеческого слуха, то есть соотношение между самым громким звуком (на грани разрушения барабанных перепонок) и едва слышимым, тоже составляет около 100 дБ.

Амплитуда связана (неточно) с воспринимаемой громкостью звука. В общем случае, два сигнала с одинаковой максимальной или среднеквадратичной амплитудой не будут обязательно иметь одну и ту же громкость. Но усиление сигнала, например, на 3 дБ в любом случае сделает его звучащим «на одну ступень» громче. Все это опирается на логарифмическую природу человеческого слуха (и прочих чувств), что частично объясняет, почему децибелы являются настолько полезной шкалой для амплитуды [RMW+02, с. 99].

Амплитуда также неточно связана с музыкальной динамикой. Динамика — это, скорее, мера усилия, но не громкости или мощности. Она включает в себя 9 значений: **тишина**, ***ppp***, ***pp***, ***p***, ***mp***, ***mf***, ***f***, ***ff***, ***fff***. Это соотносится с амплитудой сигнала ещё более свободно, чем громкость [RMW+02, сс. 110–111].

1.3 Управление амплитудой

Возможно, наиболее часто встречающаяся операция над электронными звуками это изменение их амплитуды. Например, есть простой способ синтезирования звуков путём комбинирования синусоид, определённых формулой, приведённой на стр. 1, семпл за семплом. Но у синусоиды есть постоянная номинальная амплитуда a, и хочется иметь возможность изменять ее с течением времени.

В общем случае, чтобы умножить амплитуду сигнала $x[n]$ на коэффициент $y \geqslant 0$, вы можете просто умножить каждый семпл на y, получая новый сигнал $y \cdot x[n]$. Любое измерение максимальной или среднеквадратичной амплитуды $y \cdot x[n]$ будет больше или меньше в y раз. В ещё более общем случае вы можете изменить амплитуду на величину $y[n]$, меняющую семпл за семплом. Если $y[n]$ — неотрицательное число и изменяется довольно медленно, то амплитуда произведения $y[n] \cdot x[n]$ (в фиксированном окне от M до $M + N - 1$) будет амплитудой от $x[n]$, умноженной на значение $y[n]$ в окне (условимся, что в нем не более чем N семплов).

В более общем случае, когда и $x[n]$, и $y[n]$ могут принимать отрицательные и положительные значения и/или быстро меняться, эффект их умножения не может быть описан просто как изменение амплитуды одного из них; об этом будет идти речь позже, в Главе 5.

1.4 Частота

Частоты, как амплитуды, часто измеряются логарифмически, чтобы подчеркнуть отношения между ними, которые, чаще всего, адекватнее описывают различие частот, по сравнению с линейной разностью. Частотное соотношение между двумя

музыкальными звуками определяет и музыкальный интервал между ними.

Равномерно-темперированный европейский музыкальный строй делит октаву (музыкальный интервал с соотношением частот 2:1) на 12 равных подинтервалов с отношением $2^{1/12}(\approx 1.059463)$. По историческим причинам этот подинтервал называется полутоном. Удобная логарифмическая высотная шкала представляет из себя простой подсчёт числа полутонов от опорной высоты, что позволяет использовать вещественные числа для высот, не попадающих в диапазон нот равномерно-темперированного строя. Наиболее широко используемая логарифмическая звуковысотная шкала, называемая высотами MIDI, где высота 69 соответствует частоте 440[10] колебаний в секунду (ля первой октавы). Для перевода высоты MIDI m в частоту в герцах f и обратно, применяются следующие формулы:

$$m = 69 + 12 \cdot \log_2(f/440) \tag{1.1}$$

$$f = 440 \cdot 2^{(m-69)/12} \tag{1.2}$$

До первой октавы, соответствующее высоте MIDI $m = 60$, имеет частоту $f = 261.626$ колебаний в секунду.

Сам по себе MIDI — это старый аппаратный протокол, который, к сожалению, получил широкое распространение в дизайне компьютерных программ[11]. В «железном» исполнении MIDI допускает для высот только целые значения от 0 до 127. Однако лежащая в основе шкала может определена для любых MIDI-чисел, даже отрицательных, например: «высота MIDI» -4 можно считать определённой частотой вибрато. Тем не менее, эта звуковысотная шкала не может описывать частоты, равные или меньшие нуля герц. (Для наглядного описания MIDI, его возможностей и ограничений см. [Bal03, гл. 6–8])

Полутон соответствует отношению 1.059:1 или приблизительно шестипроцентному увеличению частоты. Полутона далее делятся на центы, каждый цент — одна сотая полутона. Порогом для ощущаемого изменения высоты музыкального звука являются 3 цента. Для до первой октавы это составляет около половины герца. График частоты как функции высоты MIDI (на протяжении двух октав), показан на рис. 1.4.

[10] A440 — международный стандарт настройки музыкальных инструментов, устанавливающий частоту ноты ля первой октавы равной 440 герц. Впервые повсеместно принять частоту ля первой октавы равной 440 герц было рекомендовано международной конференцией в 1939 году. В 1955 году это соответствие было закреплено международным стандартом ISO 16. В настоящее время в концертной музыкальной практике также широко используются частоты 442 и 443 герца.

[11] MIDI является выраженным клавишно-ориентированным протоколом. Приёмы звукоизвлечения, нехарактерные для клавишного инструмента, могут быть лишь сымитированы средствами MIDI с той или иной степенью достоверности. (*прим. перев.*)

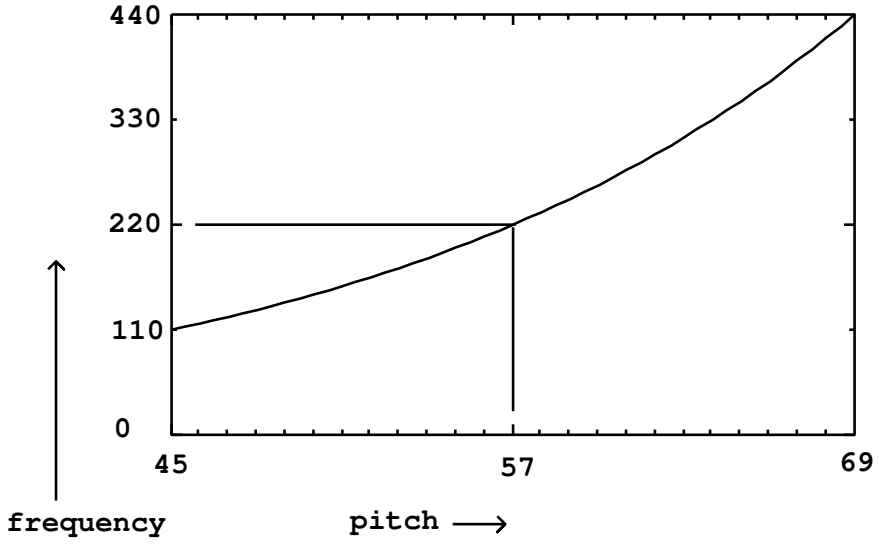


Рис. 1.4: Соотношение между высотой MIDI и частотой в колебаниях в секунду (герцах). Промежуток из 24 значений MIDI на оси абсцисс составляет две октавы, на протяжении которых частота увеличивается в четыре раза.

1.5 Синтез синусоиды

В наиболее часто используемых программных пакетах синтеза и обработки аудио (например, Csound, Max/MSP и Pd), операции над аудио определяются соединением в сеть *юнит-генераторов*[12], обменивающихся пропускаемыми через них аудиосигналами. Пользователь программного пакета создает некую схему, иногда называемую *патчем*, которая соответствует используемому алгоритму синтеза и обеспечивает управление отдельными генераторами во времени. В этом параграфе мы будем использовать абстрактные блок-схемы для описания патчей, в то время как в параграфе «Примеры» (стр. 18) мы выберем конкретную программную оболочку и покажем некоторые программно-зависимые детали.

Чтобы показать как генерировать синусоиду с меняющейся во времени амплитудой, нам потребуется два отдельных генератора. Сначала нам нужна чистая синусоида, которая создаётся с помощью осциллятора. На рис. 1.5 *a* показано графическое представление синусоидального осциллятора в виде иконки. На вход подается частота (в герцах), на выходе получаем синусоиду с пиковой амплитудой, равной единице.

Рисунок 1.5 *b* показывает, как умножить выходной сигнал синусоидального осциллятора на необходимый масштабный коэффициент $y[n]$ для управления его амплитудой. Поскольку пиковая амплитуда осциллятора равна 1, то пиковая

[12]Юнит-генераторы (или *ugens*) являются основными формальными единицами во многих языках программирования компьютерной музыки семейства MUSIC-N. Иногда их называют опкодами (особенно в Csound), хотя это выражение не является точным, поскольку это не инструкции машинного уровня. (*прим. перев.*)

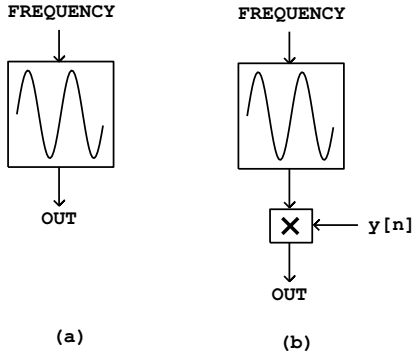


Рис. 1.5: Блок-схема для: *a* синусоидального осциллятора; *b* управления амплитудой с использованием умножителя и амплитудного сигнала $y[n]$.

амплитуда результата будет приблизительно $y[n]$, предполагая, что $y[n]$ изменяется довольно медленно и не принимает отрицательных значений.

На рис. 1.6 показано, как на синусоиду из рис. 1.1 действуют изменения амплитуды, произведённые с помощью двух управляющих сигналов $y[n]$. Управляющий сигнал, показанный в части *a*, имеет разрыв, также как и результирующая синусоида с регулируемой амплитудой, показанная на рисунке *b*. В частях *c* и *d* показан вариант более осторожного изменения $y[n]$ и его результат. Интуиция подсказывает, что результат *b* не будет звучать как синусоида с изменяющейся амплитудой, вместо этого мы услышим синусоиду, прерванную слышимым «щелчком», после чего следует тихое продолжение. В общем, по причинам, которые выходят за рамки этой главы, управляющие амплитудой сигналы $y[n]$, в случае плавного перехода от одного значения к другому, менее склонны к паразитным эффектам (вроде «щелчков»), по сравнению со случаями резких изменений.

Пока же можно сформулировать два общих правила без приведения доказательств. Во-первых, чистые синусоиды — это сигналы, наиболее чувствительные к паразитным эффектам быстрого изменения амплитуды. Поэтому, если вы хотите протестировать амплитудный переход, в случае, когда он работает для синусоид — он, скорее всего, будет прекрасно работать и в случае других сигналов. Во-вторых, в зависимости от сигнала, чью амплитуду мы меняем, амплитудное управление требует от 0 до 30 миллисекунд переходного времени: 0 для сигналов с наибольшей к этому снисходительностью (вроде белого шума) и 30 для с наименьшей (вроде синусоиды). Все это находится в сложной взаимосвязи с уровнями прослушивания и акустической средой.

Подходящие функции для управления амплитудой $y[n]$ могут быть созданы с использованием генератора огибающей. На рис. 1.7 показана схема, в которой генератор огибающей используется для управления амплитудой осциллятора. Генераторы огибающей серьёзно различаются по устройству, но мы сосредоточимся на простейшем варианте, который создает линейные сегменты, как на рисунке 1.6 *c*. Если линейный сегмент определен как переход между двумя выходными значения-

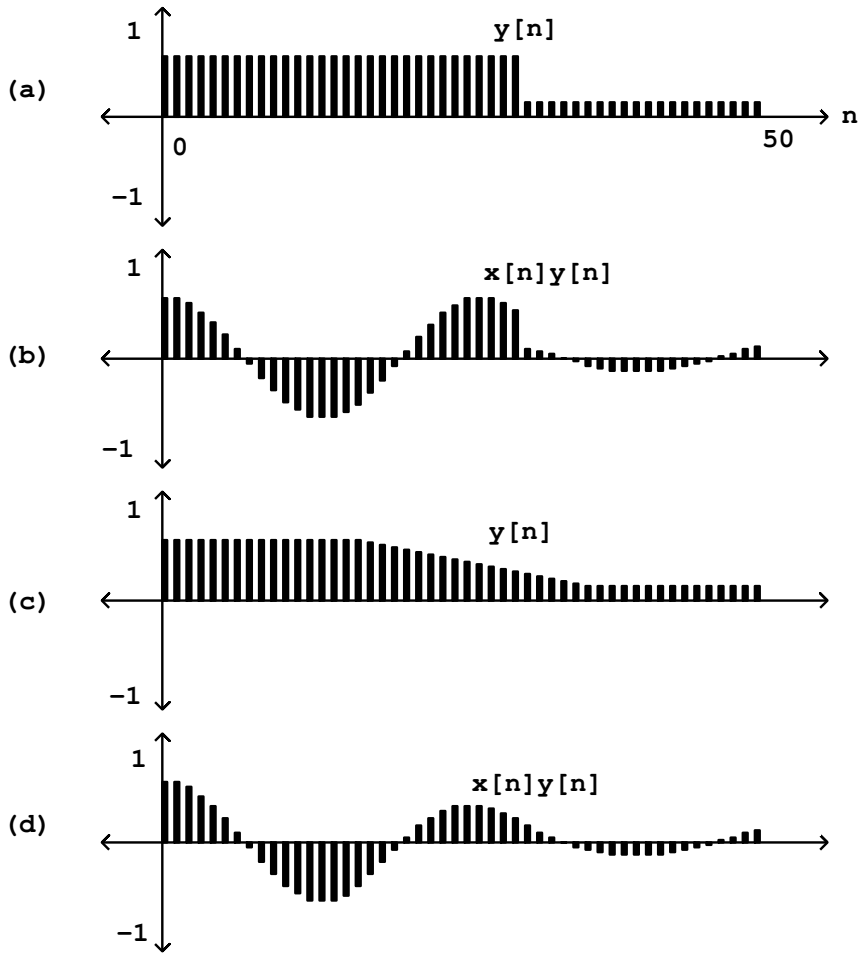


Рис. 1.6: Две амплитудные функции (части a, c) и (части b, d), результат умножения их на чистую синусоиду из рис. 1.1.

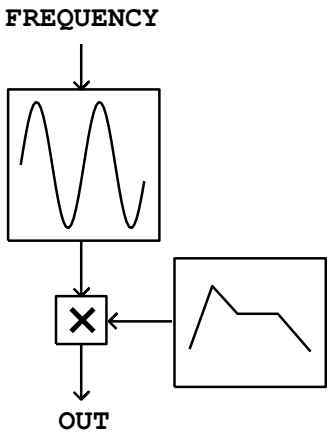


Рис. 1.7: Использование генератора огибающей для управления амплитудой.

ми a и b за N семплов с началом на семпле номер M, тогда на выходе мы получаем

$$y[n] = a + (b - a)\frac{n - M}{N}, M \leqslant n \leqslant M + N - 1.$$

Выходной сигнал может включать любое количество подобных сегментов, примыкающих друг к другу и покрывающих весь диапазон номеров семплов n; плоские горизонтальные сегменты получаются при установке $a = b$.

Кроме перечисленного, управление амплитудой часто используется, особенно в приложениях реального времени, просто для включения и выключения звуков: для выключения звука с плавным уменьшением амплитуды до нуля. Большинство программных пакетов также предлагает и специальные возможности для действительно полной остановки вычислений семплов в модулях, но вместо этого здесь мы будем использовать управление амплитудой.

Генератор огибающей известен еще со времен аналоговой эры [Str95, с. 64] [Cha80, с. 90] (см. пример использования на рис. 1.7); осцилляторы с управляемой частотой назывались VCO, а умножение происходило при помощи VCA [Str95, сс. 34-35] [Cha80, сс. 84-89]. Генераторы огибающей подробнее рассмотрены в §4.1.

1.6 Сложение сигналов

Если у сигнала $x[n]$ есть пиковая или среднеквадратичная амплитуда A (в некоем фиксированном окне), то амплитуда масштабированного сигнала $kx[n]$ (где $k \geqslant 0$) будет равна kA. Средняя мощность масштабированного сигнала изменяется на множитель k^2. В случае если два разных сигнала складываются, ситуация становится более сложной: простое знание их амплитуд не позволяет узнать амплитуду суммы. Величины этих двух амплитуд, как минимум, подчиняются неравенству

треугольника[13]; для любых двух сигналов $x[n]$ и $y[n]$,

$$A_{peak}\{x[n]\} + A_{peak}\{y[n]\} \geqslant A_{peak}\{x[n] + y[n]\}$$

$$A_{RMS}\{x[n]\} + A_{RMS}\{y[n]\} \geqslant A_{RMS}\{x[n] + y[n]\}.$$

Если мы, как обычно, фиксируем окно от M до $M + N - 1$, то сможем записать среднюю мощность суммы двух сигналов:

$$P\{x[n] + y[n]\} = P\{x[n]\} + P\{y[n]\} + 2 \cdot COV\{x[n], y[n]\}.$$

В этой формуле мы ввели *ковариацию*[14] двух сигналов:

$$COV\{x[n], y[n]\} = \frac{x[M]y[M] + \cdots + x[M + N - 1]y[M + N - 1]}{n}.$$

Ковариация может быть положительной, отрицательной или равной нулю. При достаточно большом окне ковариация двух отличающихся по частоте синусоид будет почти равна средней мощности. Если для некоторых двух сигналов ковариация равна нулю — они называются *некоррелированными* (корреляция — это ковариация, нормированная так, чтобы укладываться в диапазон от -1 до $+1$). В общем случае, для двух некоррелированных сигналов, мощность суммы — это сумма мощностей:

$$P\{x[n] + y[n]\} = P\{x[n]\} + P\{y[n]\}, \text{если только } COV\{x[n], y[n]\} = 0.$$

Что применительно к амплитуде дает:

$$(A_{RMS}\{x[n] + y[n]\})^2 = (A_{RMS}\{x[n]\})^2 + (A_{RMS}\{y[n]\})^2.$$

Это обычное соотношение Пифагора. Таким образом, некоррелированные сигналы могут рассматриваться как векторы под прямым углом друг к другу, положительно коррелированные — как векторы под острым углом друг к другу, и отрицательно коррелированные — как векторы под тупым углом друг к другу.

Например, если у двух некоррелированных сигналов среднеквадратичная амплитуда a, то среднеквадратичная амплитуда суммы — $\sqrt{2}a$. С другой стороны, если это два равных сигнала (максимально возможная корреляция), то амплитуда суммы будет равна $2a$ — максимуму, допускаемому неравенством треугольника.

[13]Неравенство треугольника утверждает, что длина любой стороны треугольника всегда не превосходит сумму длин двух его других сторон.

[14]Ковариация — в теории вероятностей и математической статистике мера линейной зависимости двух случайных величин.

1.7 Периодические сигналы

Говорится, что сигнал $x[n]$ повторяется с периодом τ, если

$$x[n+\tau] = x[n]$$

для всех n. Такой сигнал будет повторяться и через промежуток 2τ и так далее; наименьший промежуток τ, через который повторяется сигнал, называется *периодом сигнала*. При обсуждении периодов цифровых аудиосигналов мы быстро придем к сложности описания сигнала, чей период не выражается целым числом, так что уравнение выше не имеет смысла. Сейчас мы просто проигнорируем эту сложность, предположив, что сигнал $x[n]$ можно как-либо интерполировать между семплами, так что он корректно определен независимо от того, является n целым или нет.

Период синусоиды (в семплах) равен $2\pi/\omega$, где ω — это угловая частота. В более общем случае, любая сумма синусоид с частотами $k\omega$, для целых чисел k, будет повторяться через $2\pi/\omega$ семплов. Такая сумма называется *рядом Фурье*:

$$x[n] = a_0 + a_1 cos(\omega n + \phi_1) + a_2 cos(2\omega n + \phi_2) + \cdots + a_p cos(p\omega n + \phi_p). \tag{1.3}$$

Кроме того, если сделать определенные технические допущения (чтобы сигнал содержал конечное количество частот), мы можем представить любой периодический сигнал как такую сумму. Это дискретно-временной вариант анализа Фурье, к которому мы вернемся в Главе 9.

Угловые частоты синусоид выше — это все целочисленные кратные ω. Они называются *гармониками* ω, а ω в этом случае называется *основным тоном*. В звуковысотных терминах, гармоники ω, 2ω, ...— это интервалы величиной в 0, 1200, 1902, 2400, 2786, 3102, 3369, 3600, ... центов вверх от основного тона; эта последовательность высот иногда называется *гармоническим (обертоновым) рядом*. Первые шесть гармоник довольно близки к кратным 100; другими словами, первые шесть гармоник близки к высотам равномерно-темперированной шкалы (но не совпадают с ней точно); так, 3-я и 6-я отличаются лишь на 2 цента, а 5-я – на 14.

Иными словами, соотношение частот 3:2 (чистая квинта) – это практически точно семь полутонов, 4:3 (чистая кварта) – почти 5 полутонов, а соотношения 5:4 и 6:5 (пифагорейские большая и малая терции) довольно близки к интервалам из 4 и 3 полутонов соответственно.

Ряд Фурье (только с тремя слагаемыми) показан на рис. 1.8. Первые три графика – это синусоиды, частоты которых находятся в соотношении 1:2:3. Общий период обозначен на абсциссе. У каждой синусоиды своя амплитуда и начальная фаза. Сумма трех сигналов внизу – не синусоида, но тем не менее сохраняет периодичность, разделяемую всеми тремя синусоидами, входящими в её состав.

Это пример *аддитивного синтеза*, в более общем случае этот термин может быть применен к сетям, в которых частоты осцилляторов управляются независимо

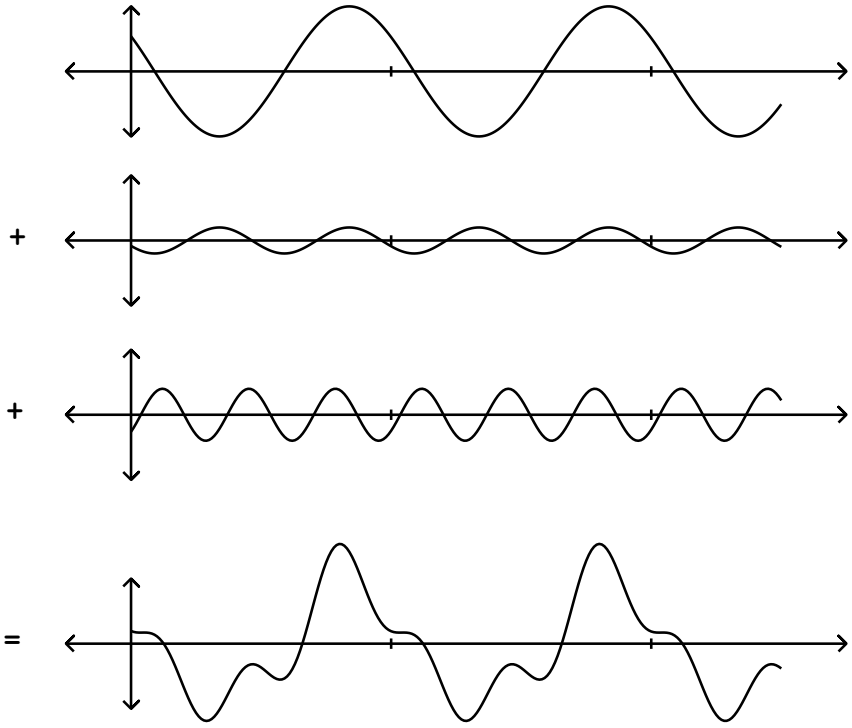


Рис. 1.8: Ряд Фурье, показывающий три синусоиды и их сумму. Три компоненты синусоид имеют частоты в отношении 1:2:3.

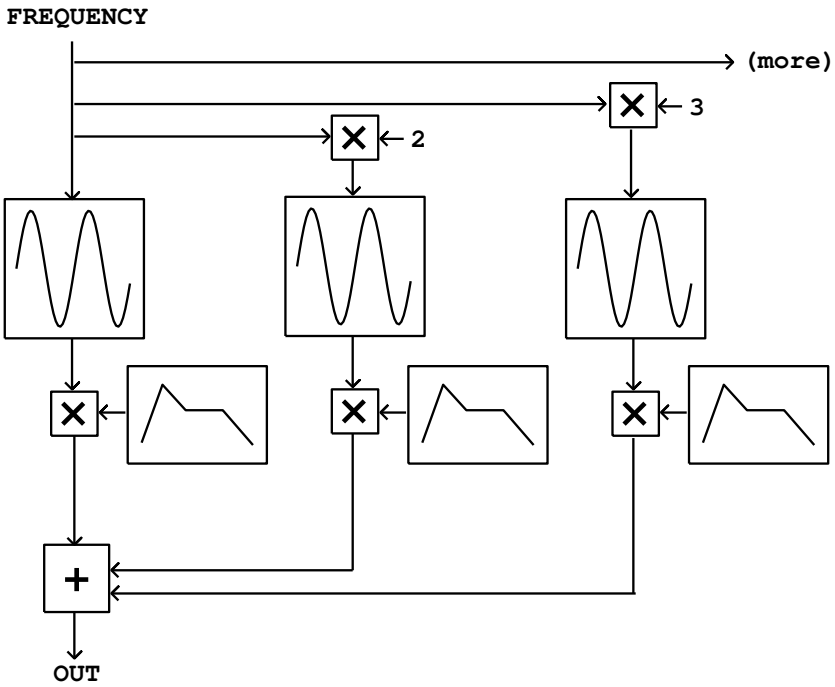


Рис. 1.9: Использование нескольких осцилляторов для синтеза волны с требуемыми гармоническими амплитудами.

друг от друга. Именно звуками, получающимися за счёт аддитивного синтеза, были отмечены первые дни компьютерной музыки.

1.8 О программных примерах

Примеры в этой книге используют язык программирования Pure Data (Pd), и для их понимания вам стоит выучить хотя бы что-нибудь о Pd. Pd – это среда для быстрого создания приложений для компьютерной музыки, в первую очередь, рассчитанных на живые выступления. Pd может быть использован также для других задач, но мы не будем углубляться в эти возможности.

Помимо Pd, есть еще несколько подобных DSP-сред. Наиболее широко используется, конечно, Csound (создатель — Барри Верко (Barry Vercoe) [Bou00]), главное отличие которого от Pd — текстовый интерфейс (нет графической оболочки). Это является преимуществом в одних случаях и недостатком в других. С одной стороны, Csound лучше, чем Pd адаптирован для пакетной обработки и лучше работает с полифонией. С другой стороны, у Pd лучше развито управление в реальном времени. Генеалогически Csound принадлежит к так называемым языкам Music N [Mat69, сс.115–172].

Другая альтернативная широко используемая программа с открытым доступом — SuperCollider (создатель — Джеймс Маккартни (James McCartney)), тоже текстово-ориентирована, но, как и Pd, создана для использования в реальном времени. Мощные лингвистические возможности SuperCollider'а делают её более удобной, чем Csound, для задач вроде написания циклов или работы со сложными наборами данных.

Наконец, существует широко известный близнец Pd, коммерческая программа компании Cycling74 — Max/MSP (все остальные упомянутые программы — с открытым исходным кодом). И для новичков, и для умудренных опытом профессионалов Max/MSP окажется более документированным и поддерживаемым, чем Pd. Представляется возможным применять знания из Pd в Max/MSP и обратно, даже перебрасывать патчи в обоих направлениях, но эти среды не являются абсолютно совместимыми.

1.8.1 Краткое введение в Pd

Документы Pd называются патчами. Грубо говоря, они соответствуют окнам на абстрактных блок-схемах, показанных в предыдущих пунктах этой главы, но на самом деле они имеют различный характер, поскольку Pd является программной средой, а не описательным языком.

Патч Pd, как это показано на рисунке 1.10, состоит из набора соединенных в сеть *блоков*. Форма блока говорит о том, как этот блок функционирует и интерпретируется внутренний текст. В части *a* рисунка мы видим три типа визуальных

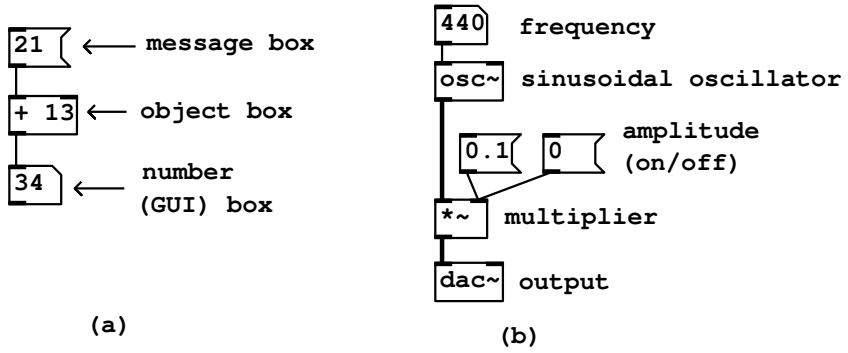


Рис. 1.10: *a* три типа визуальных блоков в Pd (message, object, GUI); *b* Простой патч, создающий синусоиду на выходе.

блоков. Сверху вниз это:

- `message box` Блоки сообщений с рамкой в форме флага интерпретируют свой текст как сообщение, отправляемое всякий раз, когда окно активируется (другим входящим сообщением или при клике указателем мыши). Блок сообщения на картинке состоит просто из числа “21”.

- `object box` Блоки объектов имеют прямоугольную форму; они интерпретируют свой текст для создания объектов при загрузке патча. Существуют сотни различных классов объектов (включая осцилляторы, генераторы огибающей и другие модули, обрабатывающие сигналы, которые будут представлены далее) в зависимости от текста, содержащегося внутри блока. В нашем примере блок отображает объект “сложение”. В большинстве патчей Pd основная часть блоков имеет тип “объект”. Первое слово, набранное внутри объекта, определяет его *класс*, который в данном случае всего лишь “+”. Любые дополнительные (разделенные пробелами) слова в блоке называются *аргументами*. Они определяют начальное состояние объекта в момент его создания.[15].

- `number box` Числовые боксы являются частным случаем GUI[16]. Другие типы блоков GUI — кнопки и тумблеры — появятся в примерах позднее. Числовой бокс имеет границу в форме перфокарты с отсеченным правым верхним краем. Во время выполнения патча, объектный блок и блок сообщений неизменны, в то время как числовое бокс меняет свое содержание (текст), чтобы отобразить текущее значение. Вы можете также использовать числовой бокс для управления, кликнув по нему и меняя его значения путем перетаскивания курсора вверх-вниз или прямым вводом значения внутри него.[17]

[15]В блоке `+ 13` — классом объекта является +, а аргументом число 13.

[16]GUI (graphical user interface) — графический интерфейс пользователя)

[17]С современных версиях Pd комбинация ctrl+click (в MacOS ⌘+click) переключает между текущим значением и 0, а комбинация ⇧+перетаскивание изменяет значение с мелким шагом (0.1).

На рис. 1.10 *a* клик по [21] посылает “21” объекту [+ 13], который прибавляет к нему 13. Соединительные линии передают данные от одного блока к другому; выходы расположены внизу, а входы – наверху блока.

На рис. 1.10 *b* показан патч Pd, который производит синусоиду с управляемой частотой и амплитудой. Соединительные линии бывают двух типов: тонкие предназначены для передачи отдельных *сообщений*, а толстые (соединяющие осциллятор, умножитель и выходной объект [dac~]) передают цифровые аудиосигналы. Поскольку Pd является программой, работающей в режиме реального времени, поток аудиосигналов является непрерывным. С другой стороны, отдельные сообщения появляются в разные и, возможно, непредсказуемые моменты времени.

Передаются ли по соединению сигналы или сообщения — зависит от объекта источника. Так, например, объект [+] выдает сообщения, а объект [*~] выдает сигнал. Входы данного объекта могут принимать или не принимать сигналы (но они всегда принимают сообщения, даже если только для преобразования их в сигналы). По соглашению, объекты с сигнальными входами или выходами имеют название с тильдой в конце (~), например [*~] или [osc~][18].

1.8.2 Как найти и запустить примеры

Для запуска патчей вы должны скачать, установить и запустить Pd. Инструкции для установки находятся в разделе HTML документации Pd по адресу http://crca.ucsd.edu/~msp/software.htm[19].

Эта книга должна быть доступна по адресу http://crca.ucsd.edu/~msp/techniques.htm, возможно в различных вариантах.

Выберите вариант, который соответствует тексту, который вы читаете (или, возможно, самый последний) и скачайте архив, содержащий соответствующую версию примеров (вы можете также скачать архив HTML версии этой книги для более простого доступа на вашем компьютере). Все примеры должны находиться в одной директории, так как некоторые из них зависят от других файлов в той же директории и могут не загрузиться корректно, если вы переместили их в другие места.

Если вы хотите скопировать один из примеров в другую директорию с целью каких-то экспериментов (что можно только поощрить), вы должны или включить директорию с примерами в пути поиска Pd (см. документацию Pd), или выяснить, какие файлы необходимы, и скопировать также и их. Лучший способ проделать это — запустить Pd на перемещенном файле и посмотреть в окне ошибок Pd какие файлы не удается найти.

В папке с примерами находится множество файлов: как примеры сами по себе, так и дополнительные файлы. Все имена файлов примеров начинаются с буквы

[18] Данное соглашение принято, потому что знак тильды напоминает синусоиду. (*прим. перев.*)

[19] Доступна также инструкция на сайте FLOSS Manual: http://write.flossmanuals.net/pure-data/introduction2/

(A — для главы 1, B — для главы 2 и т. д.) и номера, например ‘‘`A01.sinewave.pd`’’.

Патчи примеров также распространяются вместе с Pd, но помните, что вы можете найти различные версии примеров, которые могут не соответствовать текущему тексту.

1.9 Примеры

1.9.1 Умножение амплитуды на константу

Пример A01.sinewave.pd на рис. 1.11 представляет простейший возможный патч, который генерирует звук при помощи всего трех объектов. (Также в патче находятся комментарии и два блока сообщений для включения и выключения аудиопроцесса “DSP” Pd).

Три объектных блока это:

osc~ — *синусоидальный осциллятор*. Левые вход и выход передают цифровые аудиосигналы. Предполагается, что на вход подается частота в герцах (возможно, изменяемая во времени). Выходом является синусоида указанной частоты. Если ничего не присоединено ко задающему частоту входу, то первый аргумент (в данном случае 440) используется как частота. Выход имеет единичную пиковую амплитуду. Вы можете задать начальную фазу, посылая сообщения (не аудиосигналы) на правый вход; на левый вход (отвечающий за частоту) также могут быть отправлены сообщения, которые автоматически преобразуются в требуемый аудиосигнал.

*~ — *умножитель*. Существует в двух формах. Если определён начальный аргумент (в нашем примере — 0.05), то объект умножает цифровой аудиосигнал (на левом входе) на это число; сообщения на правый вход могут изменить число, на которое умножается сигнал. Если аргумент не задан — объект перемножает два входных аудиосигнала.

dac~ — *устройство вывода аудиосигнала*. В зависимости от вашего компьютера, это может не быть настоящим конвертером цифрового сигнала в аналоговый (Digital/Analog Converter — отсюда название dac); но в общем, устройство позволяет вам посылать любой аудиосигнал на аудиовыходы вашего компьютера. При отсутствии аргументов выводит сигнал на первый и второй каналы аудиооборудования; вы можете установить другие номера каналов (один или несколько), используя начальные аргументы. Pd может быть настроен для использования двух или более выходных каналов или может не иметь открытого выходного аудиоустройства вообще; подробности можно найти в документации.

Два блока сообщений показывают особенность в анализе сообщений блоками сообщений. Ранее, на рис. 1.10 (часть *a*), сообщение состояло только из числа 21. При

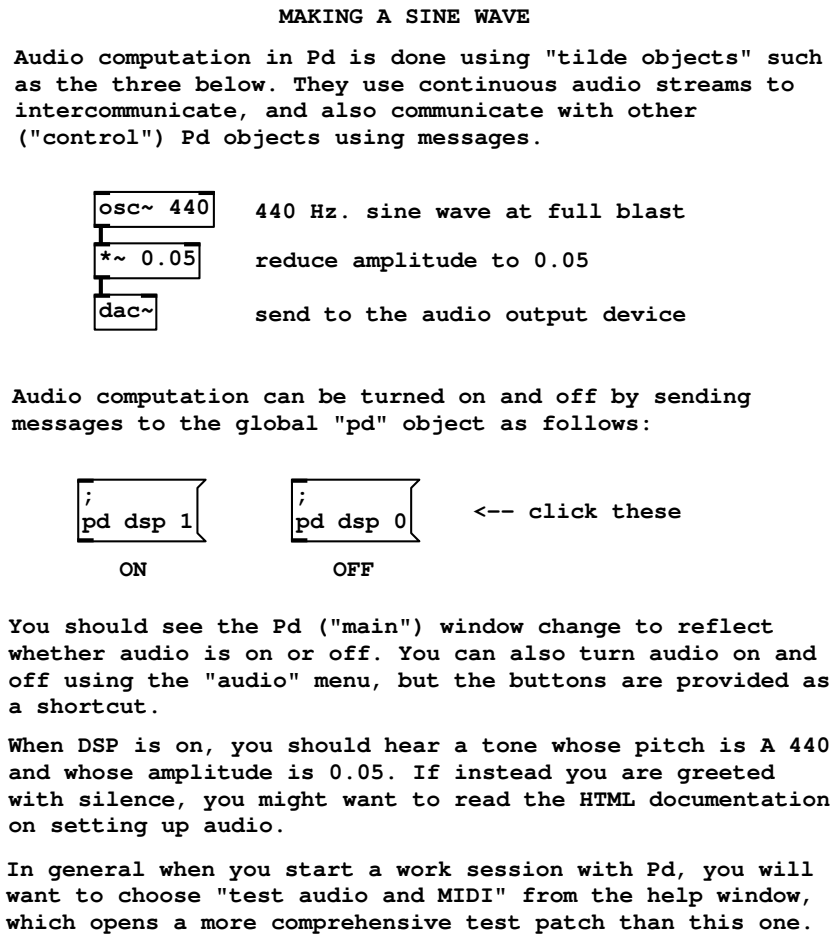


Рис. 1.11: Содержание первого примера патча Pd: A01.sinewave.pd

клике по нему этот блок отправлял сообщение “21” на свой выход и, следовательно, ко всем объектам, соединенным с ним. В текущем примере текст блоков сообщений начинается с точки с запятой. Это разграничитель между сообщениями (таким образом, первое сообщение пусто), после которого следующее слово воспринимается как имя получателя следующего сообщения. Итак, сообщение здесь “dsp 1” (или “dsp 0”), и оно будет отправлено не какому-либо присоединенному объекту (таких все равно нет), а объекту по имени “pd”. Этот специфический объект неявно предоставляется программой Pd, и вы можете отправлять ему различные сообщения для управления глобальным состоянием Pd, в данном случае, включению аудиопроцессинга отвечает “1”, а отключению “0”.

Многочисленные подробности об аспектах управления Pd, таких как сказано выше, разъяснены в файлах примеров (“control examples”) Pd, но они будут затронуты здесь только по мере необходимости, чтобы продемонстрировать методы обработки аудиосигнала, которые являются предметом этой книги.

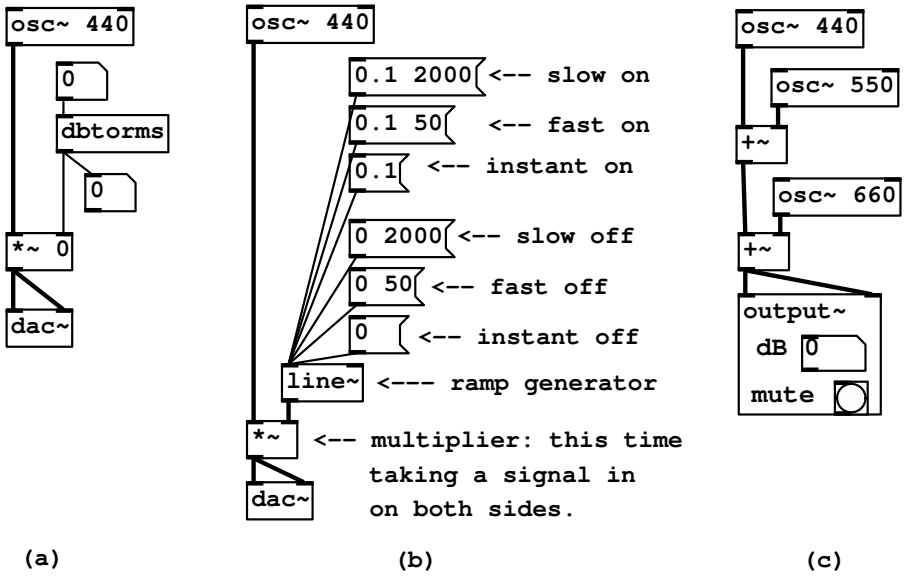


Рис. 1.12: Основные действующие элементы патчей: *a*) A02.amplitude.pd, *b*) A03.line.pd, *c*) A05.output.subpatch.pd.

1.9.2 Управление амплитудой в децибелах

Пример A02.amplitude.pd показывает простое управление амплитудой; действующие элементы показаны на рис. 1.12 (часть *a*). На нем присутствует новый класс объектов:

> dbtorms : *конвертер децибелов в линейную амплитуду.* Окончание названия “rms” (среднеквадратичная амплитуда) этого класса объектов является, строго говоря, не совсем верным; этот объект должен был бы быть названа dbtoam [20], так как он в действительности переводит децибелы в любые единицы линейной амплитуды, такие как RMS, пиковая или другие. Входное значение 100 дб дает единицу на выходе. Значения больше 100 — корректны (например, 120 дает 10), но величины меньшие или равные нулю будут давать ноль (в противном случае — ноль на входе выдавал бы небольшое положительное число). Это управляющий объект, то есть входящие и выходящие значения являются сообщениями, а не сигналами. (Соответствующий ему объект dbtorms~ работает с сигналами. При этом, он расходует больше ресурсов ЦПУ, и очень часто мы будем находить тот или иной путь для сокращения этой нагрузки).

Два числовых бокса соединены с входом и выходом объекта dbtorms. Первый функционирует как элемент управления: используйте указатель курсора (кликните и перетаскивайте вверх или вниз) для изменения амплитуды. Диапазон возможных значений задан от 0 до 80 для защиты вашего слуха и было бы разумно использовать такую защиту при построении ваших собственных патчей.

[20] «неправильное» название существует, видимо, для сохранения совместимости со старыми патчами. (*прим. перев.*)

Другой числовой бокс отображает вывод объекта dbtorms. Бесполезно кликать по нему указателем курсора, поскольку его выход ни с чем не связан; он здесь только для того, чтобы показывать, что поступает на его вход. Числовые боксы могут быть полезны для управления, отображения или для того и другого вместе, хотя если вы используете эти возможности одновременно, то может потребоваться некоторая дополнительная работа[21].

1.9.3 Гладкое управление амплитудой с использованием генератора огибающей

Как показано на рис. 1.6, один из способов сделать гладкое изменение амплитуды сигнала без щелчков — умножить его на вывод генератора огибающей, как это показано в форме блок-схемы на рис. 1.7. Эта операция может быть проделана в Pd с использованием объекта line~:

> line~: *генератор огибающей.* Выходной сигнал объекта изменяется линейно от одного до другого значения за время, определяемое полученными сообщениями. Входы получают сообщения, определяющие целевые значения (левый вход) и время их достижения (правый вход). В силу основного правила обработки сообщений в Pd, пара чисел[22], отправленная на левый вход, достаточна для установки целевого значения и времени одновременно[23]. Время указывается в миллисекундах (принимая во внимание частоту семплирования), а целевое значение — безразмерно или, другими словами, выходной диапазон должен быть согласован с входами, к которым может быть присоединен данный объект.

Пример A03.line.pd демонстрирует использование объекта line~ для управления амплитудой синусоиды. Действующая часть показана на рисунке 1.12 (часть b). Все шесть сообщений подсоединены к объекту line~ и активируются кликом по ним; например, верхнее определяет, что line~ изменяет свой сигнал (начиная с величины, которая была на его выходе перед получением сообщения) до величины 0.1 за две секунды. После того, как две секунды прошли, пока другие сообщения не пришли, вывод остается постоянно равным 0.1. Сообщения могут получены до того, как эти две секунды истекли, в этом случае объект line~ прекращает свою старую траекторию и начинает новую.

Два сообщения на line~ могут быть получены одновременно или настолько близко друг к другу по времени, что между ними не происходит вычисления DSP; в этом случае, более раннее сообщение не имеет эффекта, так как line~ не успевает изменить свой вывод для такого и его текущий выход, без изменений

[21] иначе возможно переполнение стека вызовов из-за возникновения бесконечной рекурсии при “наивной” реализации. (*прим. перев.*)

[22] список из двух чисел, например: 0.5 100

[23] значения списка отправленного на первый вход распределяются по одному между остальными входами.

далее используется как начальная точка для второго сегмента. Исключением из этого правила будет, если line~ получит нулевое значение времени: выходное значение немедленно устанавливается на новую величину и другие сегменты будут начинаться от этой новой величины; таким образом, отправляя две пары, первую с нулевым значением времени и вторую с ненулевым значением времени — можно независимо определить начальные и конечные значения сегмента на выходе line~. Трактовка правого входа line~ необычна среди объектов Pd, поскольку он забывает старые значения; сообщение с единственным числом 0.1 всегда эквивалентно паре 0.1 0 . Почти все остальные объекты удерживают предыдущее значение на правом входе вместо его переустановки на ноль.

Пример A04.line2.pd показывает объект line~ графически. Используя различные блоки сообщений, вы можете воссоздать эффекты, показанные на рис. 1.6.

1.9.4 Мажорное трезвучие

В примере A05.output.subpatch.pd, основные действующие элементы которого показаны на рис. 1.12 (часть *c*), представлены три синусоиды с частотами в соотношении 4:5:6, так что нижние две составляют мажорную терцию, верхние две — минорную терцию, а верхняя и нижняя — квинту. Нижняя частота равна 440, то есть ля первой октавы или MIDI 69. Другие — приблизительно на 4 и 7 полутонов выше, соответственно. Все три имеют равные амплитуды.

Управление амплитудой в этом примере выполняется новым объектом под названием output~. Он не является встроенным объектом Pd, но сам есть патч Pd, лежащий в файле “output.pd”. (Вы можете посмотреть внутреннее устройство output~, открыв контекстное меню “Properties” для этого объекта и выбирая “Open”.) Вы получаете две возможности управления, одну для амплитуды в децибелах (100 означает “единичный коэффициент усиления”) и клавишу “mute”[24]. Обработка звука в Pd автоматически включается, когда вы устанавливаете выходной уровень больше нуля — возможно, это не лучшее поведение в целом, но подходит для этих патчей-примеров. Механизм вложения одного патча Pd в качестве объектного окна[25] внутри другого обсуждается в §4.7.

1.9.5 Перевод частоты в звуковысотность и наоборот

Пример A06.frequency.pd (рисунок 1.13) показывает объект Pd, преобразующий звуковысотность MIDI в частоту в герцах (mtof, что означает “MIDI to frequency”) и обратный ему — ftom. Мы также представим два других класса объектов — send и receive.

> mtof, ftom: *конвертация высоты звука MIDI в частоту в герцах* согласно формуле 1.1. Входы и выходы принимают сообщения (“тильда”-эквиваленты

[24] приглушение звука

[25] часто используется термин «абстракция». (*прим. перев.*)

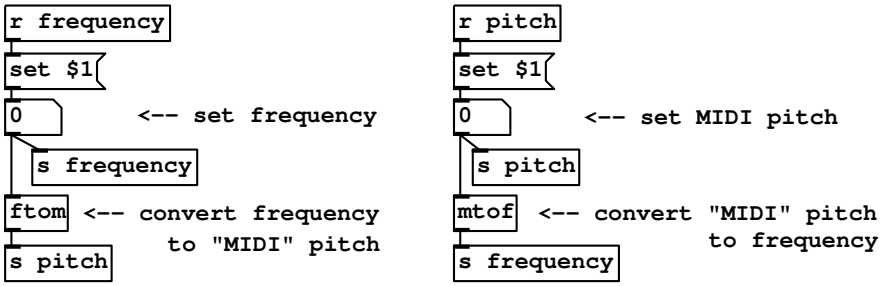


Рис. 1.13: Перевод частоты в высоту звука и наоборот в A06.frequency.pd

этих двух объектов существуют, хотя, как и dbtorms~, они создают большую нагрузку на ЦПУ). Вывод объекта ftom будет равен −1500, если на входе ноль или отрицательное число; и наоборот, если вы подаёте на вход mtof −1500 или более низкое значение, то на выходе получаете ноль.

receive, r: *получение нелокального сообщения.* Объект receive, название которого может быть сокращено до “r”, получает нелокальные сообщения, отправленные объектом send (описан ниже) или окном сообщений с использованием перенаправления (специальный символ “;” обсужденный ранее в примере A01.sinewave.pd). Начальный аргумент (такой как “frequency” и “pitch” в примере) является адресом, на который сообщение отправляется. Несколько объектов receive могут “слушать” один адрес. В таком случае, любое отправленное на этот адрес сообщение будет получено ими всеми.

send, s: Объект send, название которого может быть сокращено до “s”, отправляет сообщения на объекты receive.

Здесь используются два новых свойства числовых боксов. Ранее мы использовали их для управления или отображения информации; в этом примере в двух числовых боксах задействованы обе эти функции. Если числовой бокс получает число на вход, он не только отображает это число, но также повторяет его на выходе. Однако на числовой бокс может быть послано устанавливающее сообщение, например set 55. Оно установит значение числового бокса на 55 (и отобразит это), но без отправки этого значения на выход, что произошло бы от простого сообщения 55. В данном примере числа, приходящие от двух объектов receive, конвертируются (с использованием окон сообщений) в сообщения set 55 и так далее. (Специальный аргумент “$1” заменяется входящим числом.) Это делается потому, что иначе у нас был бы бесконечный цикл: частота меняла бы высоту тона, которая меняла бы частоту и так до бесконечности, или, по крайней мере, до какого-либо прерывания.

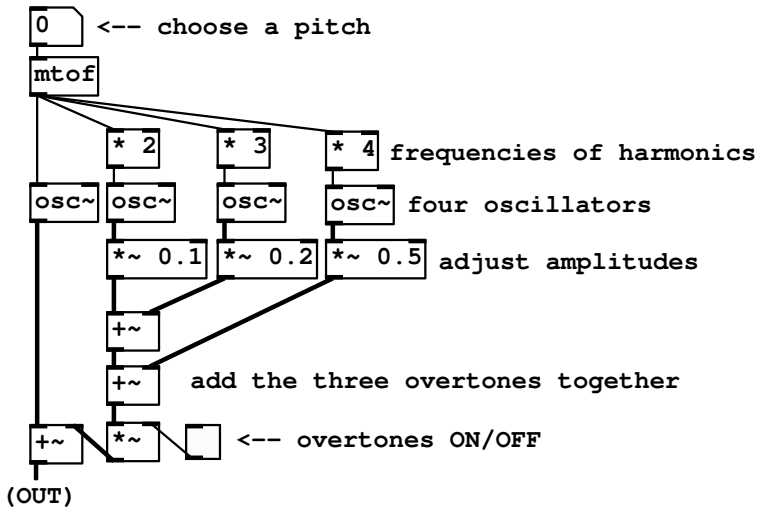


Рис. 1.14: Аддитивный синтез, использующий гармонически настроенные осцилляторы

1.9.6 Дополнительные примеры аддитивного синтеза

Мажорное трезвучие (Пример A06.frequency.pd) показывает один способ комбинирования нескольких синусоид путем суммирования. Имеется много других возможных способов создать объединения синусоид, из которых мы покажем два. Пример A07.fusion.pd (Рисунок 1.14) показывает четыре осциллятора, частоты которых настроены в отношении 1:2:3:4 с относительными амплитудами 1, 0.1, 0.2 и 0.5. Амплитуды установлены путём умножения выходов осцилляторов (объекты *~ расположенные ниже осцилляторов).

Второй, третий и четвертый осциллятор включаются и выключаются с использованием *тумблерного переключателя*. Это графический объект управления, такой же как числовой бокс, описанный выше. Тумблерный переключатель устанавливается на 1 или 0 при клике указателем курсора. Это значение умножается на сумму второго, третьего и четвертого осцилляторов, включая и выключая их.

Даже когда все четыре осциллятора комбинированы (с тумблерным переключателем в положении “1”), результат объединяется в единый тон, слышимый на высоте самого левого осциллятора. В действительности этот патч суммирует ряд Фурье для генерирования сложной периодичной формы волны.

Пример A08.beating.pd (рис. 1.14) показывает другую возможность, в которой шесть осцилляторов настроены в три близкие по частотам пары, например, 330 и 330.2 герц. Эти пары медленно попадают и не попадают в фазу одна с другой, так что амплитуда суммы меняется во времени. Называемый *биением*, этот феномен часто используется для музыкальных эффектов.

Осцилляторы можно комбинировать и другими способами, помимо простого суммирования их выходного сигнала, так что в результате доступен широкий диапазон звуков. Пример A09.frequency.mod.pd (не показанный здесь) демонстрирует

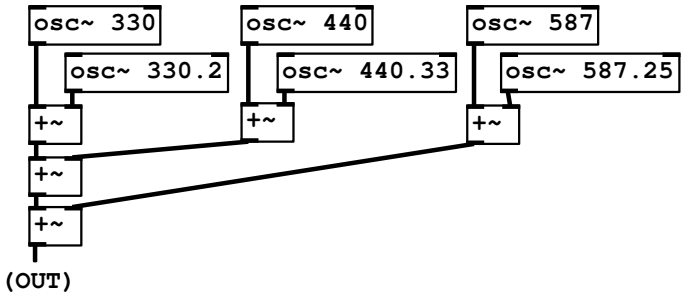


Рис. 1.15: Аддитивный синтез: шесть осцилляторов, разбитых на три пары с биениями

частотно-модуляционный синтез, в котором один осциллятор управляет частотой другого. Эта возможность будет более полно описана в Главе 5.

Упражнения

1. Синусоида (стр. 1) имеет начальную фазу $\phi = 0$ и угловую частоту $\omega = \pi/10$. Какой её период в семплах? Какая фаза будет у семпла $n = 10$?

2. Две синусоиды имеют периоды в 20 и 30 семплов соответственно. Какой период будет у их суммы?

3. Если 0 дб соответствует амплитуде 1, какое количество дб соответствует амплитудам 1.5, 2, 3 и 5?

4. Два некоррелированных сигнала с амплитудой RMS 3 и 4 складывают. Какова амплитуда RMS суммы?

5. Сколько некоррелированных сигналов равной амплитуды нужно сложить для получения сигнала, который на 9 дб больше по амплитуде?

6. Какова угловая частота “до” первой октавы в 44100 семплах за секунду?

7. Две синусоиды представляют из себя “до” первой октавы (MIDI 60) и близкого соседа “до” первой октавы (MIDI 61). Какая разница в герцах между этими двумя частотами?

8. Сколько центов в интервале между седьмой и восьмой гармониками периодического сигнала?

9. Если аудиосигнал $x[n], n = 0, \ldots, N-1$ имеет пиковую амплитуду 1, какая минимально и максимально возможные амплитуды RMS?

Глава 2

Волновые таблицы и семплеры

В Главе 1 мы рассматривали звуковые сигналы так, как будто они всегда идут непрерывным потоком с некоторой частотой дискретизации. На самом деле частота семплирования (дискретизации) не является свойством самого аудиосигнала, а скорее определяет, насколько быстро отдельные семплы должны поступать или покидать компьютер. Но аудиосигналы, по существу, лишь последовательности чисел, и на практике нет требования, чтобы они воспроизводились только последовательно. Другой, дополнительный взгляд заключается в том, что они могут храниться в памяти и, в последствии, могут быть считаны в любом порядке — прямом, обратном, вперёд-назад[1] или вообще случайно. Существует неисчерпаемый диапазон новых возможностей.

В течение многих лет (приблизительно 1950–1990), магнитная лента была основным носителем для хранения звука. Ленты прокручивались вперёд и назад через магнитные головки для воспроизведения сигнала в реальном времени. Приблизительно с 1995 года наиболее распространённым способом хранения звука стало сохранение его в виде цифровых аудиосигналов, считывание которых даёт больше свободы и возможностей, чем использование магнитной плёнки. С ними применяются не только приемы плёночной эры (вырезание, копирование, изменение скорости и обращение времени использования и др.), но и другие технические возможности, например *вейвшейпинг*[2], ставшие доступными в цифровую эру.

Предположим, мы имеем сохраненный цифровой аудиосигнал, то есть просто последовательность семплов (чисел) $x[n]$ для $n = 0,1,\ldots,N-1$, где N — это длина последовательности. Тогда, если мы имеем входной сигнал $y[n]$ (который мы можем представить как сигнал реального времени) — мы можем использовать его значения в качестве индексов для считывания значений сохраненного сигнала $x[n]$. Эта операция, называемая считыванием волновой таблицы, даёт нам новый сигнал $z[n]$, вычисляемый по формуле

$$z[n] = x[y[n]].$$

[1] иногда такой способ воспроизведения называется «пинг-понг»

[2] изменение формы волны

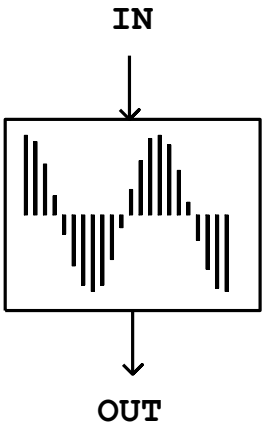


Рис. 2.1: Схема операции считывания волновой таблицы. Вход в семплах имеет диапазон приблизительно от 0 до размера волновой таблицы N, в зависимости от схемы интерполяции.

Этот процесс показан схематически на рис. 2.1.

Возникает две сложности. Во-первых, входные данные $y[n]$ могут лежать вне диапазона $0, 1, \ldots, N-1$. В этом случае, волновая таблица $x[n]$ не имеет значения и выражение $z[n]$ не определено. При такой ситуации мы могли бы обрезать входные данные, подставив 0 для всех отрицательных n и $N-1$ для всех $n \geqslant N$. Как вариант, мы можем предложить завернуть входные значения по кругу. Здесь мы примем соглашение, что семплы вне диапазона всегда обрезаются; когда нам нужно циклическое закручивание, мы будем вводить для этого другую операцию обработки сигнала.

Вторая сложность заключается в том, что входные значения могут быть необязательно целыми числами; иными словами, они могут попасть между точек волновой таблицы. В общем, с этим можно справиться выбором некоторой схемы для интерполяции между точками волновой таблицы. Сейчас мы, однако, будем лишь округлять вход до ближайшего снизу целого числа. Это называется считыванием волновой таблицы без интерполяции и его точное определение таково:

$$z[n] = \begin{cases} x[\lfloor y[n] \rfloor], & \text{если } 0 \leqslant y[n] < N-1 \\ x[0], & \text{если } y[n] < 0 \\ x[N-1], & \text{если } y[n] \geqslant N-1 \end{cases}$$

(где $\lfloor y[n] \rfloor$ означает «максимальное целое значение не превосходящее $y[n]$»).

Образно говоря, мы используем $y[0]$ (число) как положение на горизонтальной оси волновой таблицы, показанной как на рис. 2.1, и выход $z[0]$, получаемый на вертикальной оси; так же для $y[1]$, $z[1]$ и так далее. Диапазон значений $y[n]$: $0 \leqslant y[n] < N$. Он отличается от обычного диапазона значений аудиосигнала от -1 до $+1$ в наших единицах. Позднее мы увидим, что используемый диапазон входных

значений от 0 до N для считывания без интерполяции, слегка уменьшается при использовании считывания с интерполяцией.

Рисунок 2.2 (часть a) показывает волновую таблицу и результат использования двух различных входных сигналов для преобразования индексов в ней. Волновая таблица содержит 40 отсчетов, которые пронумерованы от 0 до 39. В части b в качестве входного сигнала $y[n]$ используется пилообразная волна. Пилообразная волна есть ничто иное, как многократно повторяемая функция линейного изменения. В предлагаемом примере диапазон изменения пилообразной волны будет от 0 до 40 (как показано на вертикальной оси). Таким образом, пилообразная волна сканирует волновую таблицу слева направо (от начальной точки 0 до конечной точки 39) и производит это при каждом повторении. Над пятьюдесятью точками, показанными на рис. 2.2 (часть b), пилообразная волна делает два с половиной цикла. Её период — двадцать семплов или, другими словами, частота (в циклах в секунду) есть $R/20$.

Часть (c) на рис. 2.2 показывает результат использования считывания волновой таблицы $x[n]$ с преобразующим индекс сигналом $y[n]$. Так как пилообразный вход просто циклически считывает содержание волновой таблицы слева направо с постоянной скоростью прецессии[3], результатом будет новый периодический сигнал, форма волны которого определяется $x[n]$, а частота — пилообразной волной $y[n]$.

Части (d) и (e) показывают пример в котором волновая таблица считывается неравномерно; поскольку входной сигнал растёт от 0 до N и далее возвращается к 0 — мы видим сначала последовательное появление волновой таблицы, далее остановку в её конечной точке и возвращение назад. Таблица сканируется слева направо и, далее, уже с большей скоростью, справа налево. Как в предыдущем примере, входящий сигнал управляет скоростью процесса, в то время как выходные амплитуды такие же, как в волновой таблице.

2.1 Таблично-волновой осциллятор

Рисунок 2.2 предлагает простой способ синтеза любой желаемой формы волны с нужной частотой, что показано на рис. 2.3 в виде блок-схемы. Верхний блок — осциллятор (не синусоидальный осциллятор, который мы видели ранее, а производящий пилообразную волну). Его выходные значения, что отмечено слева от блока, должны иметь диапазон от 0 до размера волновой таблицы N. Эти значения используются в качестве индексов в блоке считывания волновой таблицы (определённом на рис. 2.1), приводя к периодической форме волны. На рис. 2.3 (часть b) добавлен генератор огибающей и умножитель для управления выходной амплитудой тем же способом, как для синусоидального осциллятора, показанного на рис. 1.7 (стр. 11). Часто используется волновая таблица с амплитудой 1 (RMS или пиковой), так что амплитуда выхода определяется лишь величиной выхода

[3]термином *Прецессия* автор называет скорость изменения точки считывания сигнала. (*прим. перев.*)

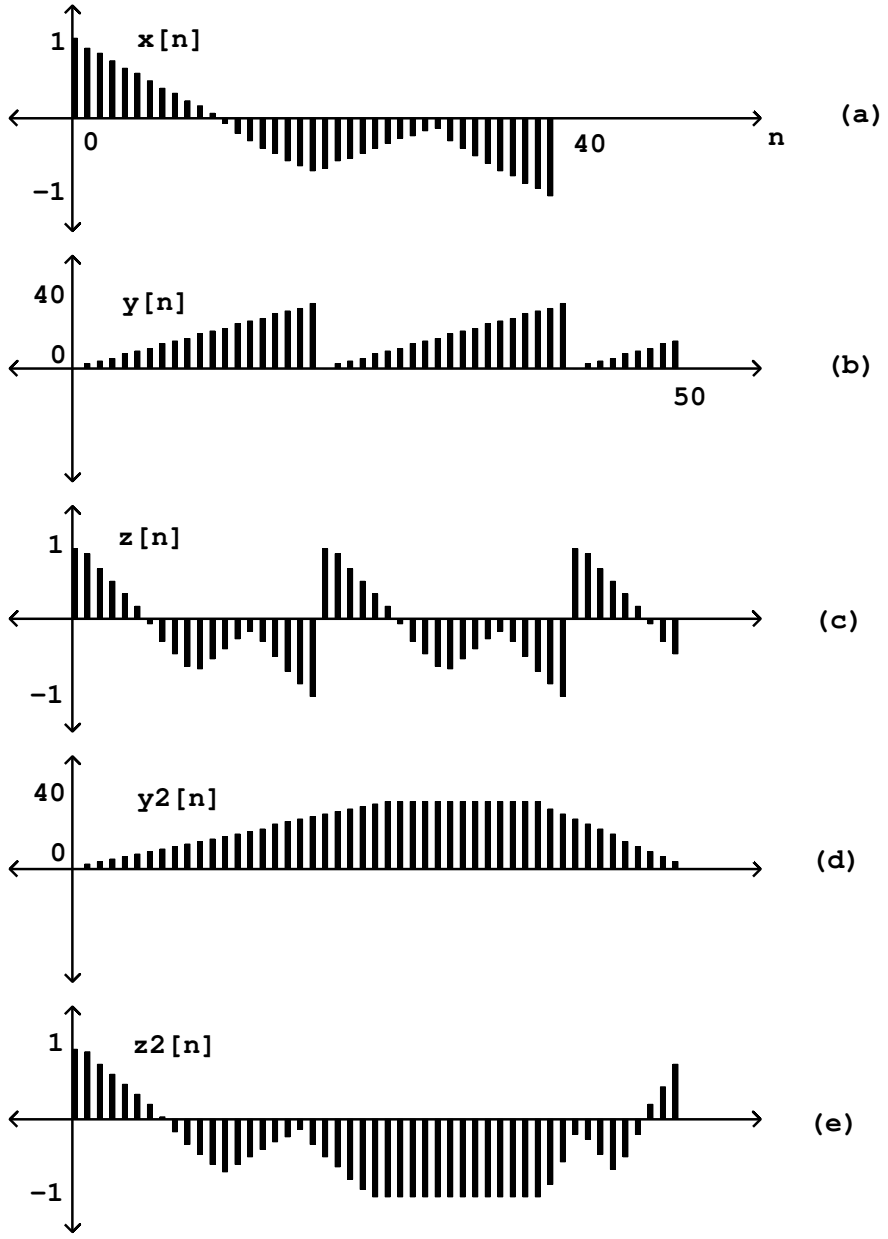


Рис. 2.2: Считывание волновой таблицы: *a* волновая таблица; *b* и *d* входные сигналы для преобразования индексов; *c* и *e* соответствующие выходы.

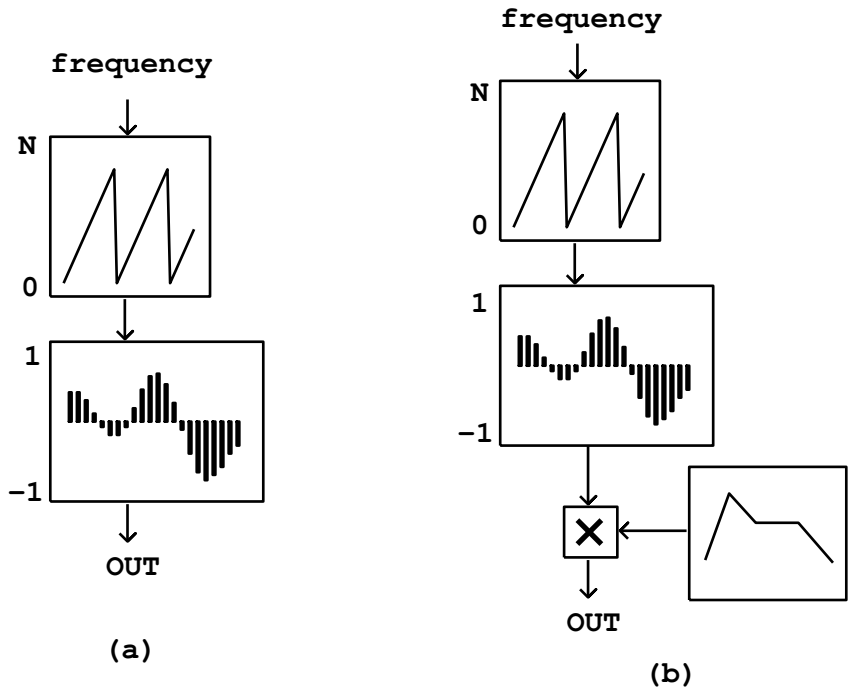


Рис. 2.3: Блок-схема: *a* для осциллятора со считыванием волновой таблицы; *b* с амплитудным управлением генератора огибающей.

генератора огибающей.

Осцилляторы[4] волновой таблицы часто используются для синтеза звука с постоянным спектром. Вы можете предварительно вычислить N семплов любой формы волны и периода N (угловой частотой $2\pi/N$), складывая компоненты ряда Фурье (1.3) (стр. 13). Вычисления в процессе заполнения таблицы могут быть довольно сложными, но их можно сделать до процесса синтеза, который тогда мог бы происходить в реальном времени.

В то время как аддитивный синтез сложных форм волны, как показано в Главе 1, в принципе, бесконечно гибок в качестве техники создания изменяемых тембров, синтезу, использующему волновые таблицы, нужно гораздо меньше вычислений, хотя требуются переключения таблиц для изменения тембра. Компромиссный вариант, более гибкий и требующий больше вычислений, чем простой синтез волновых таблиц, но менее гибкий и менее трудоёмкий для компьютера, чем аддитивный синтез, заключается в изменяемом во времени смешивании[5] небольшого числа фиксированных волновых таблиц. Если число волновых таблиц равно только двум — это приводит к взаимному переходу одной таблицы в другую, как показано в виде схемы на рис. 2.4. Предположим, что мы хотим использовать сигнал $0 \leqslant x[n] \leqslant 1$ для управления относительной интенсивностью двух форм волны, так что если

[4]Под осциллятором понимается устройство, выход которого даёт периодически повторяющийся сигнал определённой формы. Поскольку все рассматриваемые сигналы дискретны, вместо термина “сигнал” можно использовать термин “волновая таблица”.

[5]часто используется термин кроссфейдинг (*cross-fading*)

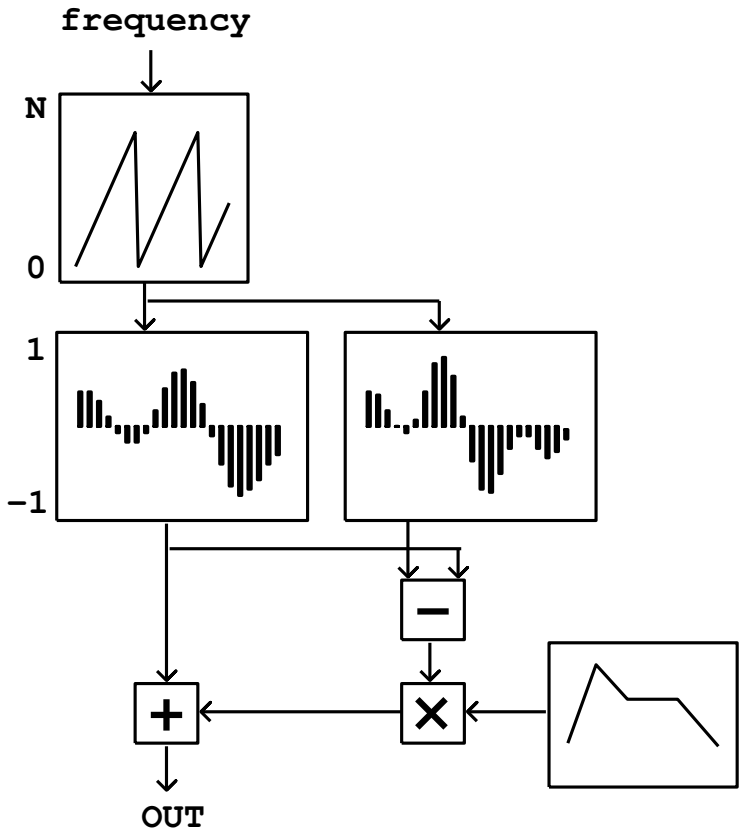


Рис. 2.4: Блок-схема смешивания двух волновых таблиц

$x[n] = 0$, то мы получаем первую, а если $x[n] = 1$, то вторую форму. Обозначая два сигнала, между которыми мы производим микширование, $y[n]$ и $z[n]$, мы вычисляем сигнал

$$(1 - x[n])y[n] + x[n]z[n]$$

или в эквивалентном, но обычно более эффективном[6] для вычисления виде:

$$y[n] + x[n](z[n] - y[n]).$$

Это вычисление показано в виде схемы на рис. 2.4.

При использовании такой техники плавного перехода между табличными осцилляторами может потребоваться синхронизация фаз у соответствующих гармоник в волновых таблицах, так чтобы их амплитуды складывались при их объединении. С другой стороны, если используются произвольные волновые таблицы (взятые, например, из записанного звука), появится фазовый эффект, поскольку смешиваются разные формы волн.

Эта схема может быть расширена по принципу “гирлянды” — цепочки таблиц для непрерывного изменения между последовательными тембрами. В качестве

[6]во втором выражении используется всего одна операция умножения. (*прим. перев.*)

альтернативы или в комбинации с цепочкой таблиц, такое смешивание может быть использовано для интерполяции между тембрами и музыкальной динамикой. Чтобы сделать это, нужно подготовить две или даже несколько волновых таблиц одного синтезируемого звука с различной динамикой и последовательно интерполировать между ними для получения нужной динамики.

2.2 Семплирование

"Семплирование" есть ничто иное, как запись реального сигнала в виде волновой таблицы с дальнейшим его воспроизведением. (В коммерческих семплерах вся волновая таблица обыкновенно называется "семплом", но для избежания недоразумений мы будем здесь использовать слово "семпл" только для одного значения в аудиосигнале).

В простейшем случае, семплер[7] — это просто осциллятор волновой таблицы, как показано на рис. 2.3. Ранее мы предполагали частоту воспроизводящего осциллятора достаточно высокой для различения звуковысотности, то есть не менее 30 Гц или около того. В случае семплирования частота обыкновенно ниже 30 Гц и её период равен как минимум 1/30 секунды или, возможно, гораздо большему значению. Таким образом, он достаточно велик, чтобы можно было слышать индивидуальные циклы как отдельные события.

Возвращаясь к рис. 2.2, предположим, что вместо 40 точек волновая таблица $x[n]$ является односекундной записью с первоначальной скоростью семплов 44100 (семп./сек.), так что она имеет 44100 точек; пусть также $y[n]$ в части b рисунка имеет период в 22050 семплов. Это соответствует частоте в 2 Гц. Но то, что мы слышим, не является звуком высотой в 2 колебания в секунду (что слишком низко, чтобы быть услышанным как высота звука), но, скорее, мы слышим первоначальную запись $x[n]$, воспроизведённую с повторением и удвоенной скоростью. Таким образом, мы только что переизобрели семплер.

В общем, если мы положим частоту семплирования R записи такой же, как частоту подачи семплов на выходе, в случае, когда волновая таблица содержит N семплов, и если мы используем в качестве индексов пилообразную волну периода M семплов, то каждый семпл ускоряется или замедляется на коэффициент N/M[8], равный Nf/R, если f есть частота пилообразной волны в герцах. Если мы обозначим коэффициент транспозиции t (например, $t = 3/2$ означает транспозицию на чистую квинту вверх) и величину транспозиции в полутонах как h, тогда мы получим следующие *формулы транспозиции для циклических волновых*

[7]Под семплером понимается устройство, выходом которого является цифровой аудиосигнал с требуемыми свойствами.

[8]Имеется в виду, что вся первоначальная волновая таблица растягивается или сжимается на этот коэффициент.

таблиц:

$$t = N/M = Nf/R \tag{2.1}$$

$$h = 12\log_2\left(\frac{N}{M}\right) = 12\log_2\left(\frac{Nf}{R}\right) \tag{2.2}$$

Часто известна транспозиция в полутонах h и формула должна быть решена относительно f или N:

$$f = \frac{2^{h/12}R}{N}$$

$$N = \frac{2^{h/12}R}{f}.$$

До сих пор мы использовали пилообразную волну как вход $y[n]$, но, как предложено в частях d и e рис. 2.2, мы могли бы использовать какой угодно входной сигнал в качестве индекса. В общем случае транспозиция может изменяться во времени и управляется скоростью изменения входного сигнала.

В этом случае коэффициенты транспозиции t и транспозиция в полутонах h задаются *формулой моментальной транспозиции для волновой таблицы*:[9]

$$t[n] = |y[n] - y[n-1]| \tag{2.3}$$

$$h[n] = 12\log_2|y[n] - y[n-1]|. \tag{2.4}$$

Например, если $y[n] = n$, тогда $z[n] = x[n]$, так что мы слышим волновую таблицу с первоначальной частотой и это как раз то, что предсказывает формула, так как в этом случае:

$$y[n] - y[n-1] = 1.$$

С другой стороны, если $y[n] = 2n$, тогда волновая таблица транспонируется на октаву, поскольку

$$y[n] - y[n-1] = 2.$$

Если значения $y[n]$ убывают с ростом n, то мы услышим семплы в обратном порядке, но транспозиционная формула по-прежнему даёт положительный коэффициент. Всё это согласуется с предыдущей формулой 2.1; если пилообразная волна пробегает значения от 0 до N (f раз в секунду) — разность между соседними семплами есть лишь Nf/R, за исключением семплов в начале каждого нового цикла.

Хорошо известно, что при транспозиции записи транспонируется также её тембр (так называемый “эффект мультяшного голоса”[10]). Потому что транспонируется

[9] вертикальные черты “|” означают взятие абсолютной величины, модуль числа.

[10] в оригинале “эффект бурундука”. (*прим. перев.*)

не только основная частота (такая, которая может задать высоту тона), но также и все частоты обертонов. Некоторые тембры, особенно вокальные, имеют характеристическую частоту в диапазоне[11], в котором обертоны сильнее, чем в соседних диапазонах. Такие частотные диапазоны также транспонируются, и это воспринимается как изменение тембра. На языке, который будет более точно описан в §5.1, мы говорим, что спектральная огибающая транспонируется вместе с изменением высоты.

В этом и предыдущем параграфах мы рассмотрели периодическое воспроизведение волновых таблиц. В параграфе §2.1 воспроизведение повторялось достаточно быстро, так чтобы повторение приводило к определённой звуковысотности, то есть между 30 и 4000 раз в секунду (грубо говоря, диапазон фортепиано). В текущем параграфе мы предположили, что волновая таблица имеет длину в одну секунду и, в этом случае, "разумный" коэффициент транспозиции (менее четырех октав вверх) привёл бы к скорости повторений меньше 30 (обыкновенно, более низкой), которая могла бы снижаться насколько мы захотим.

Число 30 важно по другой причине: это приблизительно максимальное число отдельных событий, которые ухо может различить за секунду; например, 30 вокальных фонем, или мелодических нот, или ударов барабана являются примерным максимумом, который мы можем вместить в секунду, прежде чем наша возможность различать их не разрушится.

Не существует четкой границы между семплерами и осцилляторами волновых таблиц, например, патч рис. 2.3 может рассматриваться как семплер (если частота повторений меньше 20 Гц) или как осциллятор волновой таблицы (если частота больше 40 Гц). Можно непрерывно переходить между этими двумя режимами. Более того, необязательно циклически воспроизводить всю волновую таблицу; с небольшим применением арифметики мы можем выбрать отдельные сегменты волновой таблицы, которые могут непрерывно менять свою длину и положение при воспроизведении.

Практику воспроизведения многих небольших сегментов волновой таблицы в быстрой последовательности часто называют *гранулярным синтезом*. Подробности можно найти в [Roa01].

На рисунке 2.5 показано как построить очень простой циклический семплер. В схеме на этом рисунке, при частоте f и размере сегмента s, коэффициент транспозиции выхода задаётся формулой $t = fs/R$, где R обозначена частота семплирования, с которой волновая таблица была записана (которая не обязана быть равной текущей скорости обработки семплов в патче). На практике это уравнение должно обыкновенно быть решено относительно f или s для достижения требуемой транспозиции.

На данной схеме осциллятор пилообразной волны управляет позицией считывания сегмента из волновой таблицы, но нижнее и верхнее значения пилы не определены статически, как это было сделано на рис. 2.3; скорее осциллятор пилообразной

[11] речь идет о формантах. (*прим. перев.*)

волны просто пробегает значения от 0 до 1 и этот диапазон приспособлен для выбора требуемого сегмента семплов в волновой таблице.

Положение сегмента l удобно определять либо по его левому краю (по нижней границе), либо как его среднюю точку; в любом случае мы определяем длину s как отдельный параметр. В первом случае мы начинаем с умножения пилообразной волны на s, так что она дальше пробегает значения от 0 до s; далее мы прибавляем l, так что она пробегает теперь значения от l до $l + s$. Для того чтобы определить положение как среднюю точку сегмента, мы вначале отнимаем $1/2$ от пилообразной волны (так что её диапазон будет от $-1/2$ до $+1/2$) и далее умножаем, как выше, на s (так что её диапазон будет теперь от $-s/2$ до $+s/2$), после чего прибавляем l для приведения к диапазону от $l - s/2$ до $l + s/2$.

В семплере с циклическим воспроизведением нам придется побеспокоиться о непрерывности между первым и последним семплом сегмента волновой таблицы, о чем будет сказано в следующем параграфе.

Другая деталь состоит в том, что если размер сегмента и его положение изменяются во времени (например они могут быть любыми цифровыми аудиосигналами) — это будет влиять на коэффициент транспозиции. В результате высота или тембр выходного сигнала могут колебаться вверх и вниз. Простейший способ избежать эту проблему — синхронизировать изменения значений s и l с регулярными разрывами пилообразной волны; поскольку сигнал разрывается скачкообразно, то транспозиция при этом не определена. Если же вы применяете огибающую для того, чтобы убрать разрыв, то эффект изменения s и l также исчезает.

2.3 Семплеры с огибающими

В предыдущем параграфе мы рассмотрели воспроизведение волновой таблицы (однократное или в цикле) для создания семплера. В большинстве реальных приложений мы должны аккуратно воспроизводить семплы, так чтобы выходной сигнал не совершал резких скачков в начале и конце семплирования. В зависимости от волновой таблицы такой разрыв может звучать как щелчок или звуковые помехи низких тонов.

Простейший путь сделать это (предполагая, что мы будем всегда воспроизводить волновую таблицу полностью, от начала до конца) — просто подготовить её заранее так, чтобы величина сигнала постепенно увеличивалась в начале и уменьшалась в конце. Это может быть сделано даже когда волновая таблица семплируется в реальном времени из живого звука, если умножить входной сигнал на линейную огибающую, подобранную по времени так, чтобы соответствовать длине записи.

Во многих ситуациях, однако, представляется неудобным или невозможным построить огибающую для волновой таблицы заранее: например, мы захотим воспроизвести только её часть или нужно динамически изменять форму огибающей. В §1.5 мы уже видели как управлять амплитудой синусоидальных осцилляторов, используя умножение на линейно изменяющуюся функцию (известную как

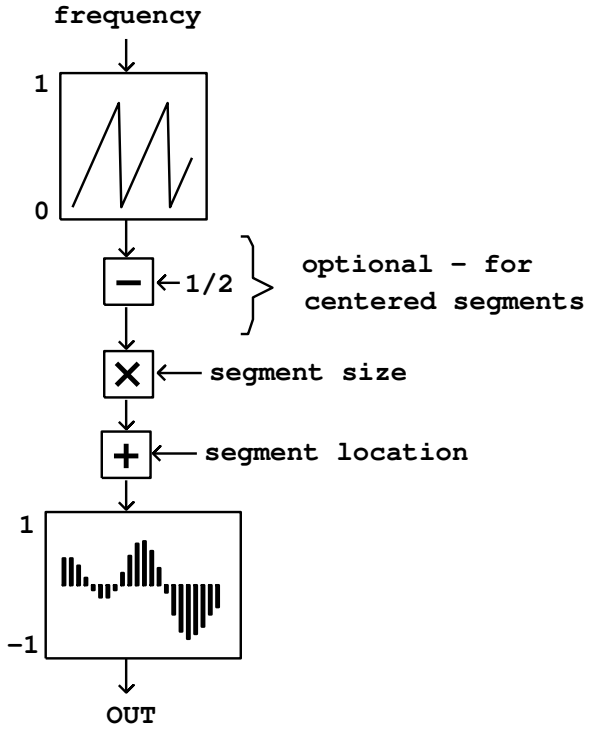


Рис. 2.5: Простой циклический семплер без управления амплитудой. Имеются входы для управления частотой, а также размером и локализацией сегмента. Операция "–" включается, если мы хотим, чтобы локализация сегмента была определена как средняя точка сегмента; в противном случае, мы определяем локализацию левого конца сегмента.

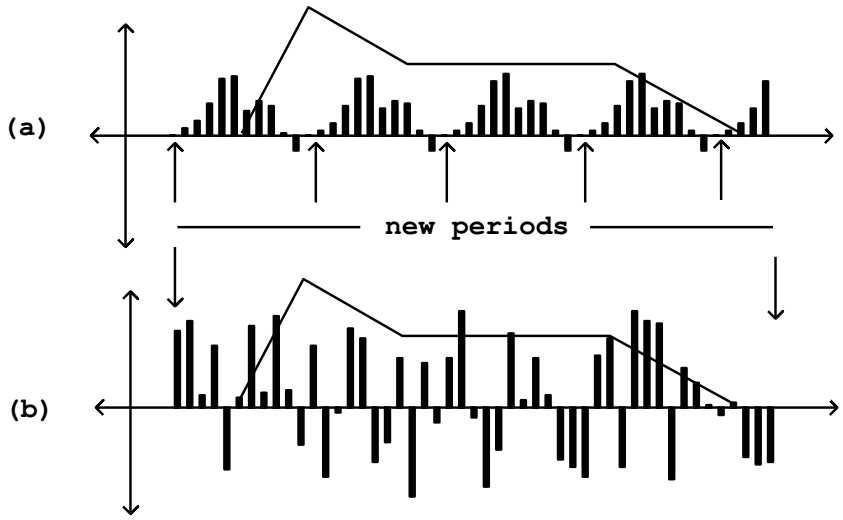


Рис. 2.6: Различные требования к огибающей для осцилляторов и семплеров: *a* в осцилляторе огибающая может быть выбрана так, чтобы соответствовать любой требуемой временной шкале; *b* когда волновая таблица является записанным звуком, вы можете установить огибающую на ноль перед тем как достигнете конца волновой таблицы в первый раз.

генератор огибающей), и мы внесли это обозначение в осцилляторы волновой таблицы на рис. 2.3 и 2.4. Такая техника хорошо работает, чтобы избегать разрывов при включении и выключении семплеров, но с одним принципиальным отличием: в то время как при синтезировании волновой таблицы мы были свободны предположить, что форма волны выстраивается циклически, так что мы можем выбрать любую временную организацию огибающей, которую мы хотим, в случае семплирования, использующего неподготовленные специально для этого формы волн, мы должны установить выход генератора огибающей на ноль в момент времени, когда мы достигаем конца волновой таблицы в первый раз. Эта ситуация изображена на рис. 2.6.

Если произвольная волновая таблица должна повторяться, то простейший путь сделать непрерывное воспроизведение — подобрать синхронизированное с длиной цикла изменение амплитуды, используя для этого отдельную волновую таблицу (огибающую). Как это может быть реализовано показано на рис. 2.7. Единственный осциллятор пилообразной волны используется для вычисления индексов для двух таблиц: одной с записанным звуком и другой — с формой огибающей. Единственная вещь о которой нужно позаботиться — установить для каждого из входов двух волновых таблиц собственный подходящий диапазон.

Во многих ситуациях необходимо комбинировать несколько циклических семплеров волновой таблицы с одной частотой, но с разными фазами. В том случае когда один семплер находится в конце сегмента, другой или несколько других — в середине того же сегмента, при этом вся комбинация непрерывно воспроизводит

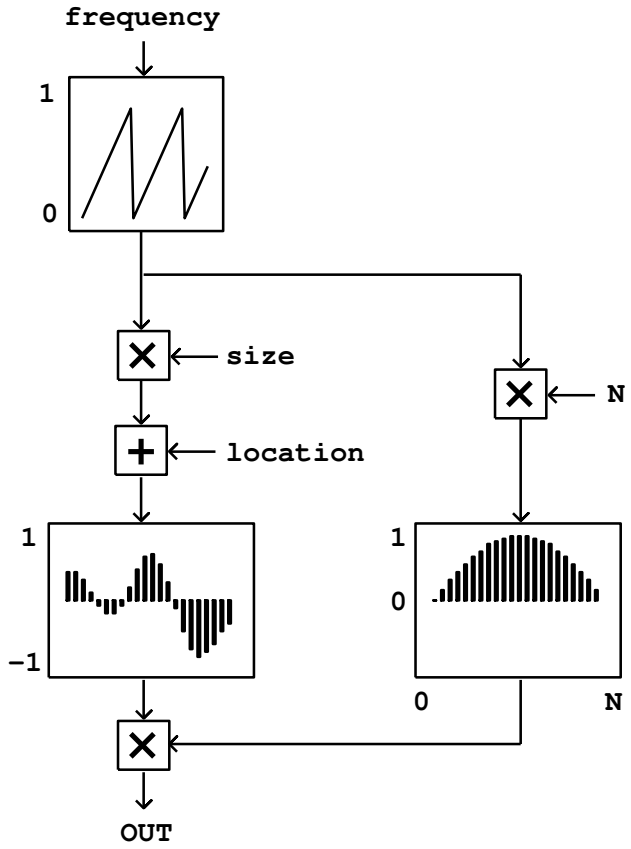


Рис. 2.7: Такой же семплер как на рис. 2.5, но с дополнительным считыванием волновой таблицы для построения огибающей

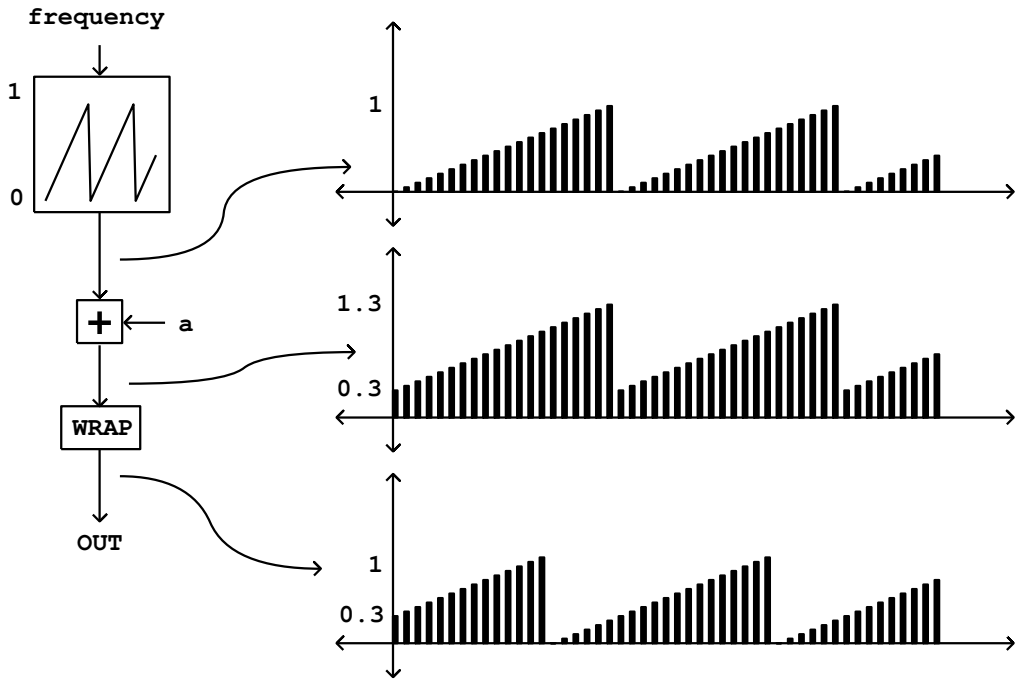


Рис. 2.8: Техника генерирования двух или более пилообразных волн с фиксированным соотношением фаз между ними. Относительная фаза управляется параметром **a** (который имеет значение 0.3 для изображенных сигналов). Операция **wrap** вычисляет дробную часть.

звук. Чтобы сделать это, нам нужен способ генерирования двух или более пилообразных волн с требуемым соотношением между фазами, который мы сможем использовать вместо осциллятора, изображённого на рис. 2.7 сверху. Мы можем начать с одной пилообразной волны и далее произвести остальные с фиксированным соотношением фаз с первой. Если мы хотим получить пилообразный сигнал, который, например, на a цикла впереди первого, мы просто прибавляем параметр a и далее берём дробную часть, которая является требуемой новой пилообразной волной, как показано на рис. 2.8.

2.4 Растяжение тембра

Осциллятор волновой таблицы §2.1, который мы дополнили в §2.2 для воспроизведения произвольных волновых таблиц, таких как записанные живые звуки, может быть расширен способом, который мы будем называть *растяжением тембра*, по причинам, которые мы рассмотрим в этом параграфе. Имеется также много других возможных способов расширения синтеза волновых таблиц, использующих модуляцию несущей частоты и вейвшейпинг, но мы оставим их до дальнейших глав.

В основе идеи растяжения тембра лежит пересмотр непрерывного циклическо-

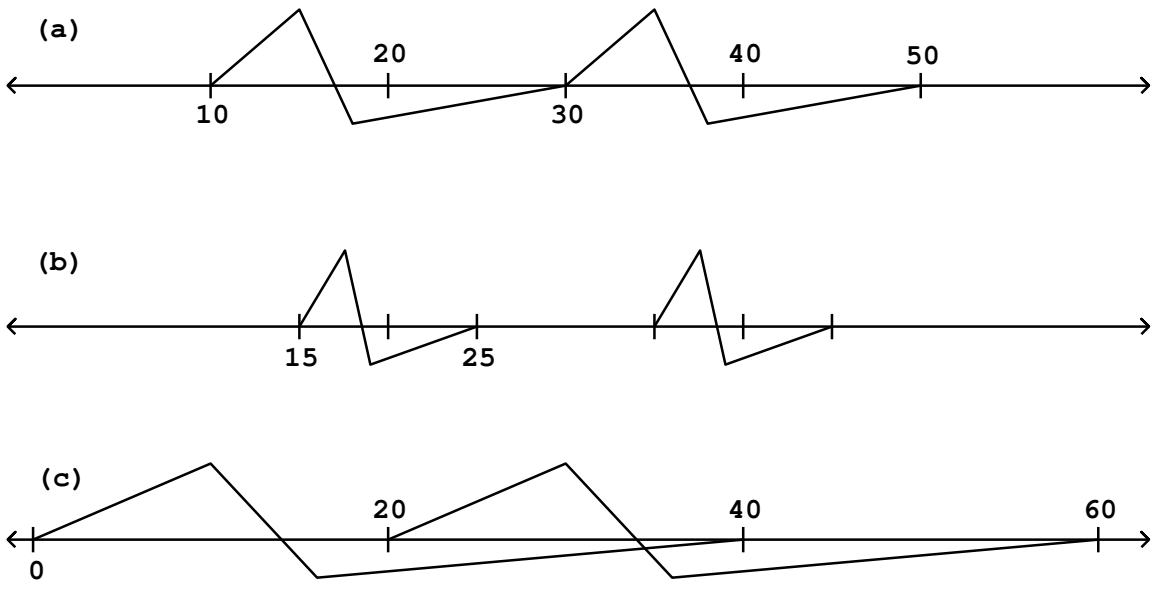


Рис. 2.9: Форма волны воспроизводится с периодом в 20 семплов: *a* со 100 процентным рабочим циклом; *b* с 50 процентным; (c) с 200 процентным.

го характера воспроизведения волновых таблиц (или их частей). Нет причины, по которой конец одного цикла воспроизведения обязательно должен совпадать с началом следующего. Вместо этого мы можем воспроизводить таблицу, чередуя её с моментами тишины или, наоборот, волновые формы могут быть так близко друг к другу, что начинают перекрываться. Единственный доступный параметр в §2.1 — частота — использовался наши раньше для управления двумя отдельными аспектами выхода: периодом формы волны и длиной каждой ее индивидуальной копии. Идея растяжения тембра состоит в независимом управлении обоими аспектами.

Рисунок 2.9 показывает результат воспроизведения волновой таблицы тремя способами. В каждом случае выходная форма волны имеет период 20; другими словами, выходная частота есть $R/20$, если R — скорость семплов на выходе.

В части *a* рисунка каждая форма волны воспроизводится на 20 семплах, так что волны склеивается циклически точно, без пауз и перекрытий. В части *b* период повтора всё так же равен 20, но форма волны сжата на половину периода (10 семплов); или, другими словами, рабочий цикл (относительная величина времени, за которое форма волны заполняет цикл) составляет 50 процентов. Оставшиеся 50 процентов времени сигнал равен нулю.

В части *c* форма волны растягивается на 40 семплов, и, так как она всё еще повторяется каждые 20 семплов, волновые формы перекрывают одна другую. Рабочий цикл, таким образом, составляет 200 процентов.

Предположим теперь, что форма волны 100% рабочего цикла раскладывается

в ряд Фурье (§1.7) равный

$$x_{100}[n] = a_0 + a_1 \cos(\omega n + \phi_1) + a_2 \cos(2\omega n + \phi_2) + \cdots ,$$

где ω есть угловая частота (равная $\pi/10$ в нашем примере, так как период равен 20). Для упрощения мы не будем беспокоиться о том, когда ряд должен закончиться и дадим ему продолжаться до бесконечности.

Мы бы хотели соотнести это выражение с рядами Фурье других двух форм волн в примере, для того чтобы показать, как изменение рабочего цикла изменяет тембр результата. Для случая 50% рабочего цикла (называемого сигналом $x_{50}[n]$), мы заметим, что форма волны, если мы её воспроизведём без фазы с половинным периодом и добавим такую же вторую, даст, в точности, оригинальную форму волны с удвоенной частотой:

$$x_{100}[2n] = x_{50}[n] + x_{50}[n + \frac{\pi}{\omega}],$$

где ω есть угловая частота (и π/ω - половина периода) обоих сигналов. Так, если мы обозначим ряд Фурье x_{50} как

$$x_{50}[n] = b_0 + b_1 \cos(\omega n + \theta_1) + b_2 \cos(2\omega n + \theta_2) + \cdots$$

и подставим ряды Фурье для всех трёх членов в формулу, приведённую выше, мы получим:

$$a_0 + a_1 \cos(2\omega n + \phi_1) + a_2 \cos(4\omega n + \phi_2) + \cdots$$

$$= b_0 + b_1 \cos(\omega n + \theta_1) + b_2 \cos(2\omega n + \theta_2) + \cdots$$
$$+ b_0 + b_1 \cos(\omega n + \pi + \theta_1) + b_2 \cos(2\omega n + 2\pi + \theta_2) + \cdots$$

$$= 2b_0 + 2b_2 \cos(2\omega n + \theta_2) + b_4 \cos(4\omega n + \theta_4) + \cdots$$

так что

$$a_0 = 2b_0,\ a_1 = 2b_2,\ a_2 = 2b_4$$

и так далее, то есть, по крайней мере чётные гармоники x_{50} получаются уменьшением гармоник x_{100} в два раза. (Мы ещё не знаем о нечётных гармониках x_{50} — для них может быть всё в согласии с чётными или нет, в зависимости от коэффициентов, которыми мы ещё не можем управлять. На настоящий момент достаточно сказать, что если форма волны с обоих концов гладко соединяется с горизонтальной осью, то нечётные гармоники будут действовать, в целом, подобно чётным. Чтобы показать это точнее, нам потребуется анализ Фурье, подробнее о котором будет сказано в Главе 9).

Аналогично, x_{100} и x_{200} связаны в точности тем же образом:

$$x_{200}[2n] = x_{100}[n] + x_{100}[n + \frac{\pi}{\omega}]$$

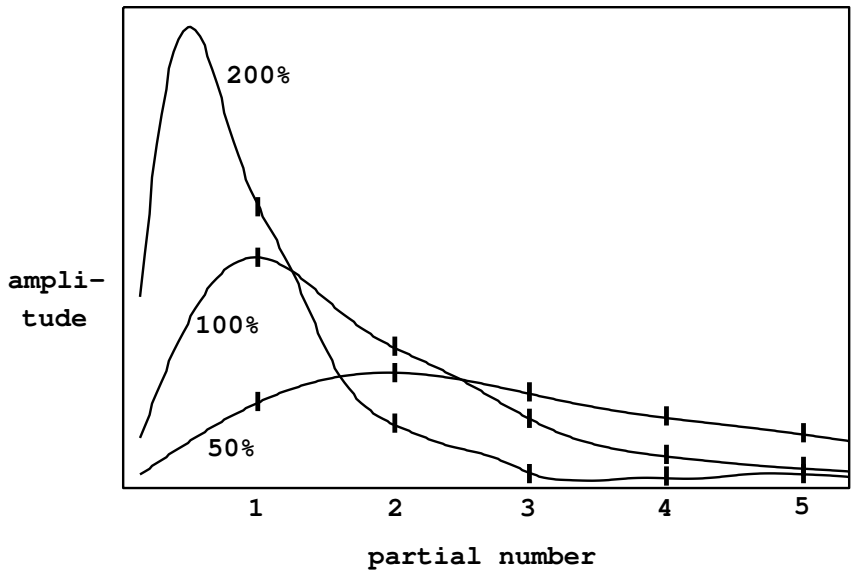


Рис. 2.10: Величины гармоник рядов Фурье для форм волн, показанных на рис. 2.9. Горизонтальная ось даёт номер гармоники. Мы "слышим" лишь коэффициенты для целых номеров гармоник; непрерывные кривые представляют из себя "идеализированные" контуры.

так что, если амплитуды ряда Фурье x_{200} обозначить $c_0, c_1, \ldots$, мы получим

$$c_0 = 2a_0, c_1 = 2a_2, c_2 = 2a_4, \ldots,$$

в следствие чего гармоники x_{200} равны гармоникам x_{100}, если их уменьшить на половину в левой части равенств.

Мы видим, что сжатие формы волны с коэффициентом 2 дает эффект растяжения ряда Фурье в два раза и, наоборот, растяжение формы волны на коэффициент 2 сжимает ряд Фурье в два раза. В общем, получается, что растяжение формы волны с любым положительным коэффициентом f сжимает обертоны по частоте на величину обратную $1/f$ — по крайней мере, приблизительно, и эта аппроксимация достаточно хороша, во всяком случае, если форма волны "ведёт себя хорошо" на своих краях. (Как мы увидим позднее, можно всегда заставить форму волны "вести себя достаточно хорошо" при помощи построения огибающей как на рис. 2.7).

Рисунок 2.10 показывает спектры трёх форм волн (или, другими словами, одной формы волны с тремя рабочими циклами) рис. 2.9. На нем подчёркнута связь между тремя спектрами с помощью проведенных через каждый спектр кривых, которые на поверку оказываются одной и той же кривой, только по-разному растянутой; когда рабочий цикл увеличивается — кривая одновременно сжимается влево (все частоты понижаются) и растет (растягивается вверх).

Эти непрерывные кривые имеют очень простую интерпретацию. Представим сжатие формы волны на очень короткий рабочий цикл, скажем 1-процентный. Контур будет растянут с коэффициентом 100. Этот процесс, осуществлённый

в обратном направлении, позволил бы нам произвести интерполяцию между каждой парой последовательных точек контура 100% (первоначального) рабочего цикла с 99-ю новыми. При этом на рисунке контур уже 50% рабочего цикла определяет кривую с двойным разрешением относительно первоначальной. В пределе, когда рабочий цикл станет произвольно малым, спектр заполняется всё более и более плотно, так что предельный спектр формы волны является “истинным”.

Тем не менее, этот “истинный” спектр слышим лишь на достаточно низких рабочих циклах. В примере 200-процентного рабочего цикла в действительности теряется пик в идеальном (непрерывном) спектре, поскольку пик выпадает ниже первой гармоники. В общем, высокие рабочие циклы семплируют идеальную кривую с низким разрешением.

Растяжение тембра является чрезвычайно мощной техникой для генерации звуков с систематически изменяющимися спектрами. В комбинации с возможностями микширования форм волн (§2.1) и захватом постоянно меняющихся волновых фрагментов из записанных семплов (§2.2), это даёт возможность генерировать все виды звуков. Например, блок-схема рис. 2.7 даёт нам способ захватывания и растяжения тембров из предварительно записанной волновой таблицы. Когда “частотный” параметр f достаточно велик, чтобы быть слышимым с определённой высотой, “размер” параметра s может рассматриваться как управляющий растяжением тембра, через формулу $t = fs/R$ из §2.2, где мы интерпретируем t теперь уже как коэффициент, на который тембр должен быть растянут.

2.5 Интерполяция

Как отмечалось выше, интерполяционные схемы часто используются для увеличения точности считывания из таблицы. Здесь мы дадим некоторый упрощённый обзор влияния размеров таблицы и способов интерполяции на результаты её считывания.

Для того чтобы говорить об ошибке при считывании таблицы, мы должны рассматривать волновую таблицу как семплированную версию первоначальной функции. Когда мы ищем значение первоначальной функции, которое лежит между точек волновой таблицы, ошибка есть разность между результатом считывания волновой таблицы и “идеальным” значением функции в той же точке. Наиболее показательное исследование ошибки табличного поиска предполагает, что основной функцией является синусоида (стр. 1). После этого мы можем понять, что происходит с другими волновыми таблицами, рассматривая их как суперпозиции (суммы) синусоид.

Точность считывания волновой таблицы, содержащей синусоиду, зависит от двух факторов: качества схемы интерполяции и периода синусоиды. В целом, чем больше период синусоиды, тем точнее результат.

В случае синтетической волновой таблицы, мы можем заранее знать её синусоидальные составляющие, и в этом случае проблема становится вопросом выбора

размера таблицы при её расчете, чтобы она соответствовала алгоритму интерполяции и удовлетворяла желаемому стандарту точности. В случае записанных звуков анализ точности может привести к корректировке частоты дискретизации записи, либо в самом начале, либо путем повторному пересемплированию позже.

Ошибка интерполяции для синусоидальной волновой таблицы может иметь две составляющие: во-первых, непрерывный сигнал (теоретический результат непрерывного чтения волновой таблицы во времени, как если бы выходная частота дискретизации была бесконечной) может не быть чистой синусоидой; во-вторых, амплитуда может быть неверной. (Можно также получить ошибки и в фазе, но только из-за невнимательности.)

В нашем подходе мы будем рассматривать полиномиальные интерполяционные схемы, такие как округление, линейная и кубическая интерполяции. Эти схемы предполагают использование значений полиномов (нулевого, первого и третьего порядка, соответственно) в промежутках между точками волновой таблицы. Идея состоит в том, что для любого индекса x мы выбираем близлежащую точку отсчёта x_0 и устанавливаем на выходе значение, вычисленное с помощью некоторого полинома:

$$y_{\mathrm{INT}}(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \cdots + a_n(x - x_0)^n.$$

Обыкновенно, мы выбираем полином, проходящий через $n + 1$ ближайших точек волновой таблицы. Для одноточечной интерполяции (полином нулевого порядка) это означает установку a_0 равным ближайшей точке волновой таблицы. При двухточечной интерполяции, мы чертим прямую линию, соединяющую две точки волновой таблицы с какой-либо стороны от требуемой точки x. Мы можем положить x_0 равным ближайшему целому числу слева от x (которое мы обозначаем $\lceil x \rceil$) и формула для линейной интерполяции будет тогда:

$$y_{\mathrm{INT}}(x) = y[x_0] + (y[x_0 + 1] - y[x_0]) \cdot (x - x_0),$$

что является полиномом, как в предыдущей формуле, с

$$a_0 = y[x_0]$$

$$a_1 = y[x_0 + 1] - y[x_0].$$

В общем, можно подобрать ровно один полином порядка $n - 1$, проходящий через любые n точек, если только их величины x различны.

Рисунок 2.11 показывает эффект использования линейной (двухточечной) интерполяции для замещения синусоиды с периодом 6. Сверху помещены три кривых: первоначальная синусоида, результат линейной интерполяции, использующей 6 точек на период для воспроизведения синусоиды и, наконец, другая синусоида немного меньшей амплитуды, которая лучше соответствует шестисегментной форме волны. Ошибка, введённая заменой первоначальной синусоиды на линейно интерполированный вариант имеет две составляющие: во-первых, (едва заметное) изменение амплитуды и, во-вторых, (сильно заметное) искажение формы волны.

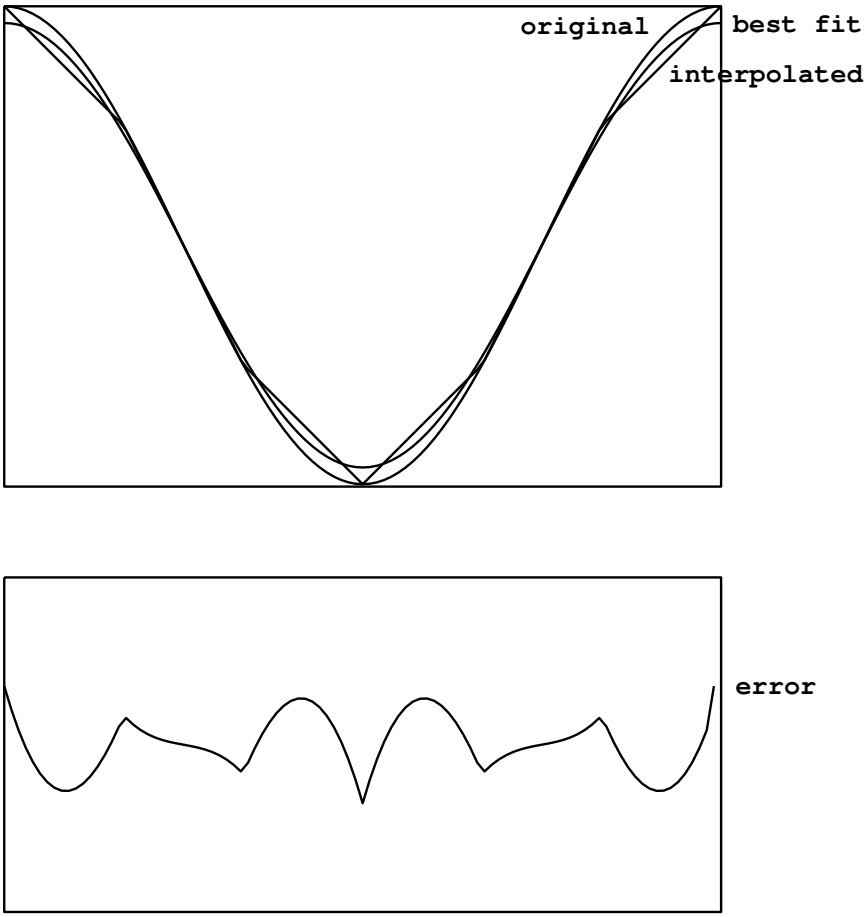


Рис. 2.11: Линейная интерполяция синусоиды: (график вверху) первоначальная синусоида, интерполированная синусоида, наилучшая синусоида, соответствующая интерполированной версии; (график внизу) ошибка, отмасштабированная вертикально.

Нижний график на рисунке показывает разницу между интерполированной формой сигнала и наилучшей интерполированной синусоидой. Это остаточный сигнал и вся его энергия принадлежит обертонам первоначальной синусоиды. Когда число точек возрастает, величина ошибки убывает. Поскольку ошибка равна разности между синусоидой и последовательностью аппроксимирующих линейных сегментов, величина ошибки, грубо говоря, пропорциональна квадрату фазовой разности между каждой парой точек или, другими словами, обратно пропорциональна квадрату числа точек в волновой таблице. Иными словами, погрешность волновой таблицы уменьшается на 12 дБ каждый раз, когда таблица удваивается в размере. (Это эмпирическое правило подходит только для таблиц с четырьмя и более точками).

Четырёхточечная (кубическая) интерполяция работает аналогично. Интерполяционная формула будет

$$y_{\mathrm{INT}}(x) =$$
$$-f(f-1)(f-2)/6 \cdot y[x_0-1] + (f+1)(f-1)(f-2)/2 \cdot y[x_0]$$

$$-(f+1)f(f-2)/2 \cdot y[x_0+1] + (f+1)f(f-1)/6 \cdot y[x_0+2],$$

где $f = x - x_0$ равно дробной части индекса. Для таблиц с четырьмя и более точками, удвоение числа точек на таблицу ведёт к улучшению в среднеквадратичной (RMS) ошибке на 24 дб. Таблица 2.1 показывает вычисленную среднеквадратичную ошибку для синусоид с различными периодами для 1, 2 и 4-х точечной интерполяции. (Немного другое значение вычислено в [Moo90, с. 164]. Там включены ошибки в амплитуде и фазе, дающие несколько более пессимистические результаты. Смотрите также [Har87].)

Допустимая область значений на входе для считывания волновой таблицы зависит от числа точек интерполяции. В общем, при использовании k-точечной интерполяции в таблице из N точек, вход может иметь диапазон в интервале $N+1-k$ точек. Если $k = 1$ (то есть интерполяции нет вообще), область допустимых значений от 0 до N (включая крайнюю точку в 0, но исключая точку в N), предполагая, что входные значения округляются до своей целой части (как это делается при считывании таблицы без интерполяции в Pd). Область допустимых значений составляет промежуток от $-1/2$ до $N-1/2$, если, наоборот, мы округляем вход до ближайшего целого значения вместо интерполяции. В любом случае, область допустимых значений растягивается в длину на N точек.

Для двухточечной интерполяции вход должен лежать между первой и последней точками, то есть между 0 и $N-1$. То есть N точек достаточно для определения функции на области определения длины $N-1$. Для четырёхточечной интерполяции мы не можем получить значения для входов между 0 и 1 (не имея требуемых двух точек слева от входа) и также не можем получить значения для последних двух точек ($N-2$ и $N-1$). Таким образом, в этом случае, область допустимых значений пролегает от 1 до $N-2$ и имеет длину $N-3$.

Периодические формы волн, сохраняемые в волновых таблицах, требуют специальной обработки на краях таблицы. Например, предположим, что мы хотим сохранить чистую синусоиду длины N. Для считывания волновой таблицы без

period	interpolation points		
	1	2	4
2	-1.2	-17.1	-20.2
3	-2.0	-11.9	-15.5
4	-4.2	-17.1	-24.8
8	-10.0	-29.6	-48.4
16	-15.9	-41.8	-72.5
32	-21.9	-53.8	-96.5
64	-27.9	-65.9	-120.6
128	-34.0	-77.9	-144.7

Таблица 2.1: Среднеквадратичная (RMS) ошибка для считывания волновой таблицы, использующего 1, 2 и 4-х точечную интерполяцию для различных размеров таблицы.

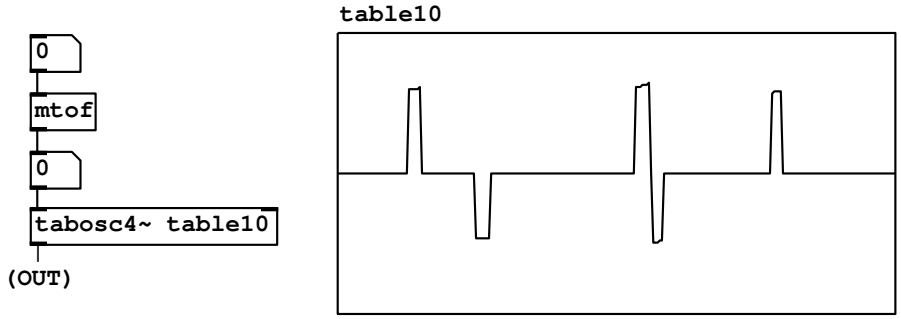


Рис. 2.12: Осциллятор волновой таблицы: B01.wavetables.pd

интерполяции, достаточно установить, например,

$$x[n] = \cos(2\pi n/N), n = 0, \ldots, N-1.$$

Для двухточечной интерполяции нам нужна $N+1$ точка:

$$x[n] = \cos(2\pi n/N), n = 0, \ldots, N.$$

Другими словами, мы должны повторить первую (n = 0) точку в конце, так что последний сегмент от $N-1$ до N возвращается назад к первоначальному значению.

Для четырехточечной интерполяции цикл должен начинаться с точки $n = 1$, так как мы не можем правильно интерполировать значения для входных данных меньше единицы. Если тогда один цикл волновой таблицы соответствует значениям от 1 до N, мы должны ввести дополнительные точки для 0 (копируя значение из N), а также для $N+1$ и $N+2$, копируя значения из 1 и 2, для того чтобы сделать таблицу длины $N+3$. Для такой же синусоиды, как выше, таблица должна содержать

$$x[n] = \cos(2\pi(n-1)/N), n = 0, \ldots, N+2.$$

2.6 Примеры

2.6.1 Осциллятор волновой таблицы

Пример B01.wavetables.pd, показанный на рис. 2.12, реализует осциллятор волновой таблицы, который воспроизводит таблицу, названную “table10”. Здесь используются два новых элемента Pd. Во-первых, волновая таблица сама по себе, показанная на рисунке справа. Вы можете воздействовать указателем курсора на волновую таблицу для изменения её формы или прослушивания появившегося в результате изменения звука. В патче также демонстрируется возможность Pd автоматического вычисления волновых таблиц с определёнными амплитудами

обертонов, что часто предпочтительнее рисования волн вручную. Вы можете также считывать и записывать таблицы в (текстовые или звуковые) файлы для обмена данными с другими программами. Другим нововведением является класс объектов:

> tabosc4~ : *осциллятор волновой таблицы.* Цифра “4” означает, что этот класс использует 4-точечную (кубическую) интерполяцию. В нашем примере, название таблицы “table10” введено как начальный аргумент в объект tabosc4~ . (Вы можете также переключать между волновыми таблицами динамически, посылая соответствующие сообщения на объект).

Волновые таблицы, используемые tabosc4~ должны всегда иметь период, равный степени двойки; но как показано выше, волновая таблица должна иметь еще три дополнительных точки, окружающих границы. Допустимые длины таблицы, тем самым, имеют форму $2^m + 3$, как 131, 259, 515 и т.п.

Осцилляторы волновых таблиц не сводятся только к использованию в качестве источника аудиосигнала осцилляторов. Патч B02.wavetable.FM.pd (не изображённый здесь) использует пару осцилляторов волновых таблиц последовательно. Выход первого используется в качестве входа второго, и таким образом контролирует его частоту, которая периодически меняется во времени.

2.6.2 Считывание волновой таблицы в общем случае

Объект tabosc4~ , хотя и является удобным и эффективным, достаточно специфичен, и для многих применений, описанных в этой главе, нам нужно нечто более общее. Пример B03.tabread4.pd (рис. 2.13) демонстрирует технику растяжения тембра, обсуждавшуюся в §2.4. Это простой пример ситуации, в которой недостаточно tabosc4~ .

Здесь описаны следующие новые объекты:

> tabread4~ : *считывание волновой таблицы.* Как и в tabosc4~ таблица читается с использованием 4-точечной интерполяции. Но, в то время как tabosc4~ принимает частоту на входе и автоматически считывает форму волны по повторяющемуся образцу, семплер tabread4~ ожидает индекс для считывания волновой таблицы. Если вы хотите использовать его для того, чтобы сделать что-нибудь повторяющееся, как в этом примере, сигнал на входе сам по себе должен быть повторяющейся формой волны. Как и для tabosc4~ (и всех других объектов, читающих и записывающих таблицы), вы можете посылать сообщения для выбора используемой таблицы.
>
> tabwrite~ : *запись аудиосигнала в волновую таблицу.* В этом примере tabwrite~ используется для отображения выходного сигнала (хотя позднее он будет использоваться и в других целей). Всегда, когда он получает сообщение

"bang"[12] от кнопки выше него, tabwrite~ начинает последовательно записывать входные семплы в указанную таблицу.

Пример B03.tabread4.pd показывает как скомбинировать объекты phasor~ и tabread4~ для создания осциллятора волновой таблицы. Вывод phasor~ имеет диапазон от 0 до 1 по величине. В этом случае входная волновая таблица, названная "waveform12", имеет 131 элемент в длину. Область определения для объекта tabread4~ будет, таким образом, от 1 до 129. Для согласования диапазона phasor~, мы умножаем его на длину области значений (128), так что он принимает диапазон значений между 0 и 128, и, далее, добавляем 1, смещая интервал вправо на одну точку. Это изменение масштаба выполняется объектами *~ и +~, помещёнными между phasor~ и tabread4~.

Только с этими четырьмя объектами мы бы в основном воссоздали tabosc4~. В этом примере, однако, умножение производится не на константу 128, а на переменное значение, управляемое "сжимающим" параметром. Задача четырёх объектов с правой стороны патча состоит в обеспечении объекта *~ значениями для масштабирования phasor~. Это достигается использованием ещё одного нового класса объектов:

pack: составляет список их двух или более элементов. Аргументы устанавливают число элементов, их типы (обыкновенно, числа) и их начальные значения. Входы (их будет столько, сколько вы определили начальными аргументами) обновляют значения аргументов сообщения и, только если крайний левый вход изменяется (или отправлено сообщение "bang"), отправляется сообщение.

В этом патче аргументами первоначально являются 0 и 50, но числовой бокс будет обновлять значения только первого аргумента, так что самое последнее сообщение, выходящее из объекта pack будет "206 50". В результате этого сообщения объект line~ линейно изменяет текущее значение до 206 за 50 миллисекунд; в общем, выход объекта line~ есть звуковой сигнал, который плавно следует спорадически изменяющимся значениям числового бокса, обозначенного как "squeeze" ("сжатие").

Наконец, 128 добавляется к значению "squeeze"; если "squeeze" принимает неотрицательные значения (как ограничивает числовой бокс в этом патче), то объект *~ задаёт диапазон пилообразного сигнала на 128 или больше. Если значение больше 128, эффект состоит в том, что изменивший свой масштаб пилообразный сигнал проводит некоторую часть своего цикла "застрявшим" в конце волновой таблицы (которая ограничивает вход на значении 129). В результате форма волны сканируется на некоторой части цикла. Как видите, форма волны сжимается на $128/(128 + 206)$ за цикл, так что спектр растягивается на коэффициент около 1/2.

Для простоты, этот патч незначительно отличается от примера в §2.4, в котором формы волны сжимаются по направлению начала каждого цикла и не сжимаются по направлению к середине. Это имеет эффект небольшого изменения фазы

[12] Стандартное сообщение, посылаемое для запуска некоторого процесса обработки сигналов.

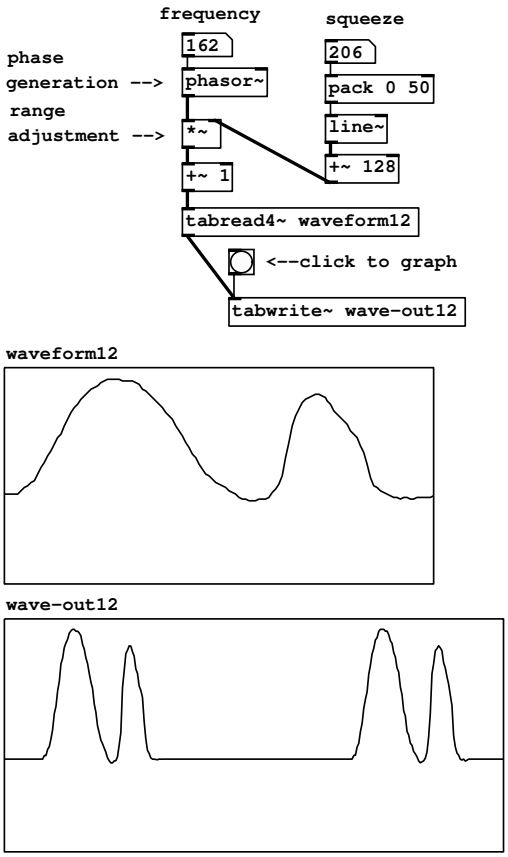


Рис. 2.13: Осциллятор волновой таблицы с переменным рабочим циклом (пример B03.tabread4.pd

)

различных гармоник формы волны, поскольку она растягивается и сжимается; если коэффициент сжатия изменяется быстро, соответствующий фазовый сдвиг будет звучать как слабое дрожание высоты. Этого можно избежать используя немного более сложную компоновку: отнять 1/2 от phasor~, умножить его на 128 или больше и, далее, добавить 65 вместо единицы.

2.6.3 Использование волновой таблицы как семплера

Пример B04.sampler.pd (рис. 2.14) показывает как использовать волновую таблицу как семплер. В этом примере индекс в семпле (волновой таблице) управляется кликом указателя курсора по числовому боксу сверху. Удобная шкала для числового бокса — сотые доли секунды; для преобразования семплов (как требует вход tabread4~) мы умножаем 44100 семплов/сек на 0.01 секунду для получения 441 семпла на единицу перед применением pack или line~, в целом, тем же способом, как они использовались в предыдущем примере. Транспозиция, которую вы слышите, зависит от того, как быстро вы двигаете указателем курсора вверх и вниз. Этот пример вводит один новый класс объектов:

> hip~: простой фильтр, пропускающий верхние частоты (срезающий нижние). Первый аргумент задаёт граничную частоту в герцах. Мы используем его, чтобы убрать постоянное смещение (с нулевой частотой), когда воспроизводится какой-либо одиночный семпл (то есть в тот момент, когда вы не изменяете активно местоположение считывания волновой таблицы указателем курсора). Фильтры обсуждаются в Главе 8.

Объекты pack и line~ в этом примере включены не просто для того, чтобы сделать звук более сглаженным, но и вообще необходимы для создания нормально различимого звука.

Если бы индекс при считывании волновой таблицы просто изменялся каждый раз, когда указатель курсора передвигалась на пиксель (например, от двадцати до пятидесяти раз за секунду) подавляющее большинство семплов получили бы тот же индекс как предыдущий семпл (другие 44000+ семплов, не считая те, где указатель курсора двигался). Таким образом, скорость воспроизведения была бы почти всегда нулевой. Вместо того, чтобы менять высоту транспозиции, вы бы слышали “звуковой песок” от 30 до 50 циклов в секунду. (Попробуйте проверить, на что такие звуки похожи!)

2.6.4 Циклические семплеры

В большинстве ситуаций вам потребуется более автоматизированный способ, чем перемещение мыши для указания места чтения волновой таблицы; например, может быть нужна возможность воспроизводить семпл с постоянной транспозицией; вы могли бы иметь несколько семплов, воспроизводимых одновременно (или другие

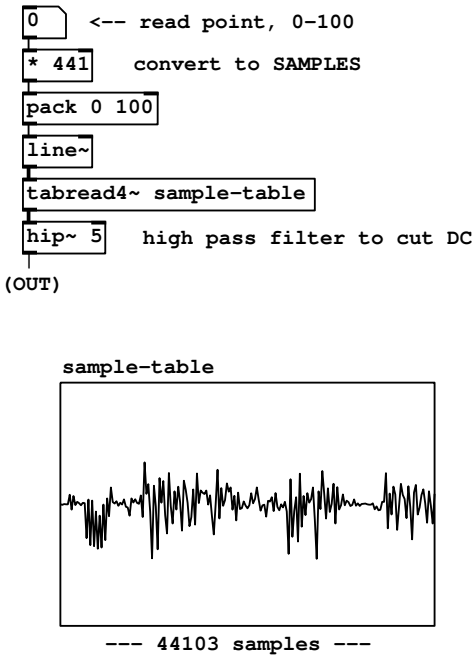


Рис. 2.14: семплер с индексом, контролируемым указателем курсора: B04.sampler.pd

вещи, заслуживающие внимания) или вы можете захотеть быстро переключать между семплами, или перемещаться на предварительно подобранные местоположения. В следующих нескольких примерах мы создадим автоматический циклический считыватель семплов, который, хотя показывает только один из многих возможных подходов, является мощным и часто используется.

Патчи B05.sampler.loop.pd и B06.sampler.loop.smooth.pd показывают, как это сделать: первый — простейшим возможным образом и второй (изображённый на рис. 2.15, часть *a*), включая вторую форму волны для огибания звука, как было описано в §2.3. Здесь используется один новый класс объектов:

> cos~ : вычисление косинуса угла, представляющего из себя произведение 2π на входной сигнал (так что значения от 0 до 1 образуют полный цикл). В отличие от объектов, читающих таблицы в Pd, cos~ справляется со значениями вне указанного диапазона, циклически заворачивая их, так что нет ограничений на входной сигнал.

На рис. 2.15 (часть *a*) объект phasor~ посылает одновременно индексы для волновой таблицы (справа) и фазы для огибающей функции в форме полукосинуса слева. Далее они перемножаются и результат подаётся на фильтр, пропускающий высокие частоты, и, далее, на выход. Считывание волновой таблицы прямое; пилообразный сигнал умножается на параметр “chunk size” (“размер образца”), складывается с 1, и используется как индекс в tabread4~. Параметр “chunk size” умножается на 441 для преобразования его из миллисекунд в семплы. Это в точности соответствует блок-схеме, показанной на рис. 2.5, с локализацией сегмента в 1. (Локализация

сегмента не может быть 0, так как 1 является минимальным индексом с которым работает tabread4~.)

Путь сигнала слева в примере соответствует технике считывания таблицы с огибающей, показанной на рис. 2.7. Здесь пилообразная волна, подогнанная к диапазону $(-1/4, 1/4)$ (вычитанием и умножением на 0.5), посылается на cos~. Это определяет функцию косинус в диапазоне $(-\pi/2, \pi/2)$, давая таким образом только положительную половину формы волны.

Часть *b* рис. 2.15 вводит третий параметр "read point" ("точка чтения"), который определяет, номер семпла, с которого начинается цикл. (В части *a* мы всегда начинали в начале). Необходимое изменение достаточно простое: добавляем управляющее значение "read point" к индексу волновой таблицы и продолжаем также как перед этим. Для избежания разрывов в индексе, мы сглаживаем значения "read point", используя объекты pack и line~, точно как мы делали в первом примере семплера (рис. 2.14).

Это приводит к важному, хотя едва различимому, результату. Формула моментальной транспозиции (стр. 33) предсказывает, что пока "chunk size" и "read point" не меняются во времени, транспозиция представляет из себя лишь произведение частоты на размер образца (как всегда, используя подходящие единицы; герцы или, например, секунды, так что произведение безразмерно). Однако изменение "chunk size" и "read point" по времени будет приводить к моментальной транспозиции, часто очень ощутимыми образами, как можно услышать в Примере B07.sampler.scratch.pd. Пример B08.sampler.nodoppler.pd (показанный на рисунке) демонстрирует один возможный способ управления этим эффектом, в то же время используя новый класс объектов:

> samphold~: выборка и хранение. (Рядовая процедура для пользователей обычных синтезаторов, но с некоторым цифровыми особенностями; детали смотрите в §3.7). Этот объект сохраняет одиночный семпл с левого входа и постоянно подаёт его на выход, пока это обеспечивается правым входом (также цифровым аудиосигналом, называемым триггером) для переписывания сохранённого семпла на новый — опять с левого входа. Объект получает новый семпл, по падению фронта сигнала триггера. Устройство создано для простого соединения объектов phasor~ при переключения во время свёртывания фазы.

Пример B08.sampler.nodoppler.pd использует два объекта samphold~ для обновления значений "chunk size" и "read point", в точности когда phasor~ свёртывается, то есть в моменты, когда косинусная огибающая обращается в ноль, так что эффект немедленных изменений не может быть услышан. В этой ситуации мы можем применить более простую *формулу транспозиции для зацикленных волновых таблиц* для соотнесения частоты, размера образца и транспозиции. Это продемонстрировано в примере B09.sampler.transpose.pd (не показан).

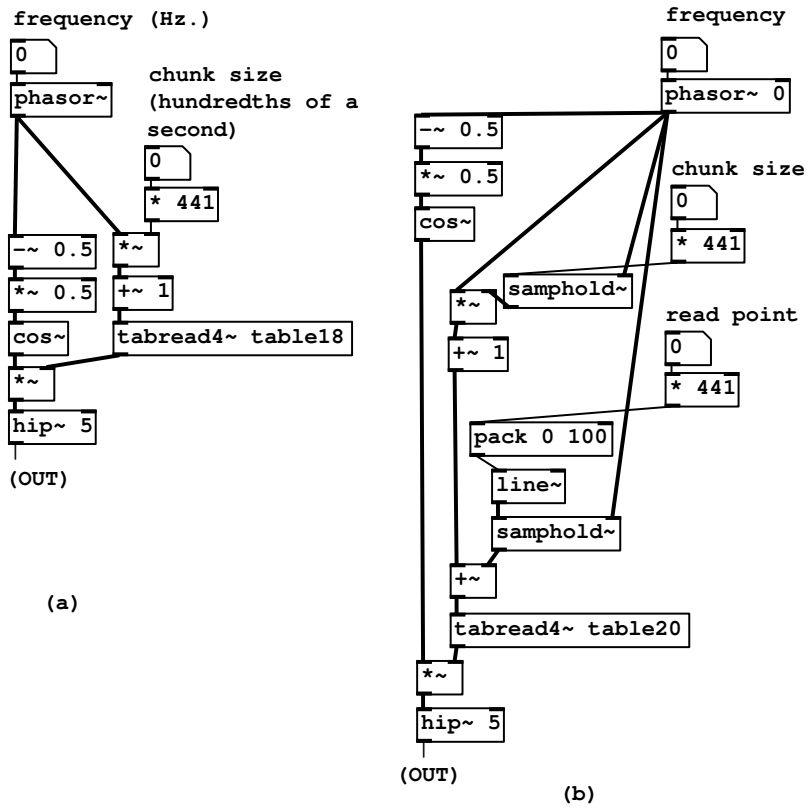


Рис. 2.15: *a*) циклический семплер с синхронизированной огибающей (пример B06.sampler.loop.smooth.pd); *b*) тоже самое, но с управлением местоположением считывания (пример B08.sampler.nodoppler.pd).

2.6.5 Циклический семплер с перекрытием

Как описано в §2.3, иногда желательно использовать два или более перекрывающихся циклических семплеров для воспроизведения достаточно непрерывного звука без необходимости слишком резкого обрыва на краях цикла. Это особенно хорошо в ситуациях, где зацикленный образец короткий, десятая часть секунды или меньше. Пример B10.sampler.overlap.pd, показанный на рис. 2.16 (часть *a*), реализует два циклических семплера сдвинутых на половину цикла по фазе один от другого. Новые классы объектов следующие:

loadbang: выводит сообщения "bang" при загрузке. Используется в этом патче для того, чтобы гарантировать, что деление транспозиции на длину семпла будет иметь корректный коэффициент транспозиции в случае, если "длина семпла" задан указателем курсора вначале.

expr: вычисляет математические выражения. Переменные появляются как **$f1**, **$f2** и так далее, соответственно с входами объекта. Допускаются арифметические операции, группировка скобками, доступны многие математические функции, такие как вычисление экспоненты, которое показано в этом примере как "pow" (the power function — показательная функция).

wrap~: свёртывание в интервал от 0 до 1. Так, например, 1.2 становится 0.2; 0.5 остаётся без изменения; и −0.6 переходит в 0.4[13].

send~, s~, receive~, r~: сигнальные версии send и receive. Аудиосигнал, посланный на объект send~ появляется на выходах всех объектов receive~ такого же имени. В отличие от send и receive вы не можете иметь больше одного объекта send~ с тем же именем (в этой связи, посмотрите объекты throw~ и catch~).

2.6.6 Автоматическое изменение точки считывания

Термином **Прецессия** будет называть скорость изменения точки считывания сигнала.

Пример B11.sampler.rockafella.pd, показанный в части *b* рис. 2.16, адаптирует показанные выше идеи к ситуации, когда считываемая точка вычисляется автоматически. Здесь мы циклически изменяем положение считываемой точки в семплах, что позволяет нам ускорять или замедлять считывание независимо от транспозиции.

Этот пример указывает на недостаток предыдущего, который состоит в том, что если относительная скорость прецессии близка к единице (то есть естественной скорости прослушивания записанной волновой таблицы) или если нет большой транспозиции, становится предпочтительным использование больших гранул[14]

[13] Операция состоит в добавлении к входу такого целого числа, чтобы результат лежал от 0 до 1.
[14] Фрагментов волновой таблицы.

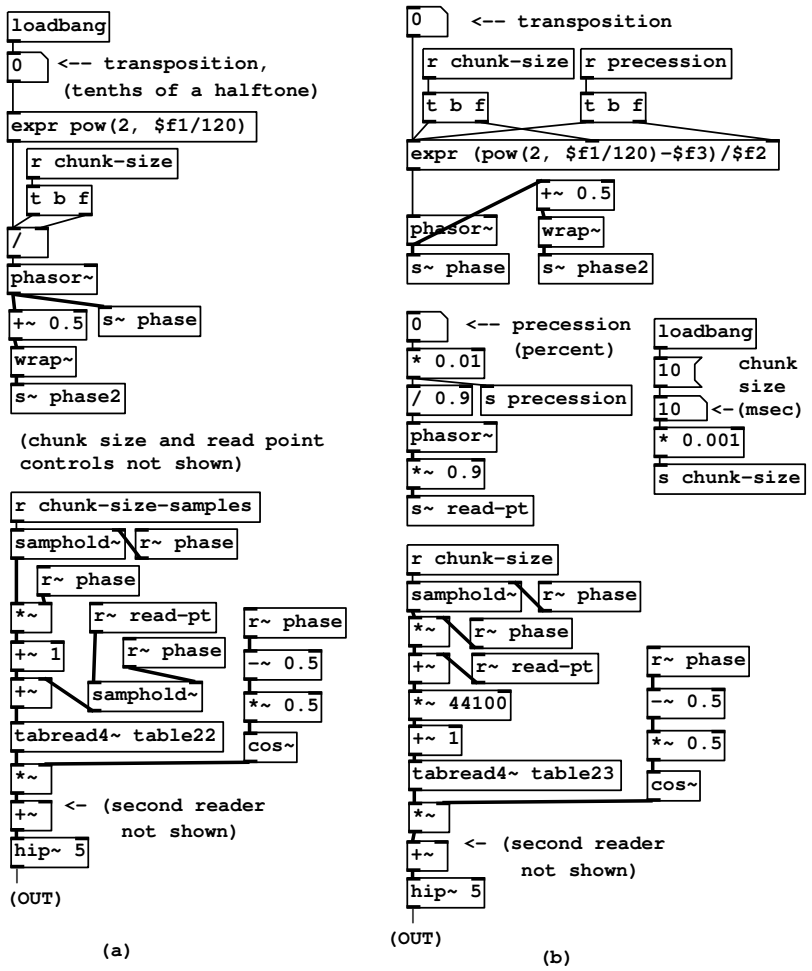


Рис. 2.16: *a* два перекрывающихся циклических семплера B10.sampler.overlap.pd; *b* то же, но со местоположением считывания, управляемым пилообразным сигналом B11.sampler.rockafella.pd.

и низкой частоты повторов, соответственно (сохраняя произведение постоянным для достижения нужной транспозиции). Однако если размер гранулы может быть большим, становится неудобно дробить изменения управления на моменты свёртывания фазы (обращения сигнала в ноль), поскольку они могут быть слишком далеко разделены, чтобы позволить управлять изменениями для разумного времени отклика.

В этом патче мы удаляем объект samphold~, который управлял считываемой точкой (но мы оставляем в нём размер образца, который гораздо труднее изменить в середине цикла). Вместо этого, мы используем (известную) скорость прецессии считываемой точки[15] для коррекции частоты пилообразной волны, так что мы поддерживаем желаемую транспозицию. Получается, что, когда коэффициент транспозиции и прецессия близки друг к другу (и мы практически делаем тоже, что простое изменение скорости), частота будет падать к значению, близкому к нулю, так что мы будем увеличивать естественность результата в тоже самое время.

В этом патче мы уходим от управления считыванием точек, размеров образца и т. п. в семплах и используем вместо этого секунды, преобразовывая их в семплы (и сдвигая на один) только лишь перед объектом taberead4~. Волновая таблица содержит одну секунду звука и мы будем предполагать здесь, что номинальный размер образца не будет превышать 0.1 секунду, так что мы можем безопасно задать диапазон считываемых точек от 0 до 0.9; "реальный" размер образца будет меняться и может стать очень большим из-за подвижного считывания точек.

Параметр прецессии устанавливает частоту пилообразной волны с амплитудой равной 0.9 и, поэтому, этот параметр должен быть умножен на 0.9 для установки частоты пилообразной волны (так что, для единичной прецессии, например, и амплитуда, и частота считывания точки есть 0.9, а наклон, равный частному амплитуды и частоты, есть единица). Выход этого патча называется "read-pt" (от "точка считывания") как раньше и используется обоими вариантами чтения волновой таблицы.

Для известной прецессии p и размера образца c, если мы обозначим частоту верхнего (первоначального) phasor~ как f, коэффициент транспозиции будет

$$t = p + cf$$

и решая относительно f получаем

$$f = \frac{t-p}{c} = \frac{2^{h/12} - p}{c},$$

где h есть требуемая транспозиция в полутонах. Это формула, используемая в объекте expr.

[15] Как задаваемый параметр.

Упражнения

1. Если волновая таблица с 1000 семплами воспроизводится с единичной транспозицией с частотой семплирования 44100 герц, сколько будет длиться получающийся звук?

2. Односекундная волновая таблица воспроизводится за 0.5 секунды. На какой интервал будет транспонирован звук?

3. Если мы проигрываем односекундную таблицу периодически (с зацикливанием), с какой частотой в герцах мы должны ее воспроизводить, чтобы произошла транспозиция оригинального звука вверх на полтона?

4. Мы хотим 10 раз в секунду циклически воспроизводить волновую таблицу (записанную с $R = 44100$), так что оригинальный звук, хранящийся в волновой таблице, транспонируется на чистую квинту (смотри стр. 14). Насколько большой сегмент волновой таблицы (в семплах) должен воспроизводиться?

5. Предположим, вы хотите использовать растяжение формы волны таблицы, которая содержит периодическую форму волны с периодом 100. Вы хотите слышать нетранспонированный спектр с периодом 200 семплов. На какой рабочий коэффициент вы должны сжать форму волны?

6. Первая половина волновой таблицы содержит цикл синусоиды с единичным пиком амплитуды. Вторая половина содержит нули. Какова сила второго обертона волновой таблицы?

7. Синусоида хранится в волновой таблице с периодом 4, так что первые четыре элемента есть 0, 1, 0 и -1 соответствуют индексам 0, 1, 2 и 3. Какое значение мы получим при входном значении 1.5: *(a)* используя 2-точечную интерполяцию? *(b)* используя 4-точечную интерполяцию? *(c)* каково значение там первоначальной синусоиды?

8. Если всё содержание волновой таблицы лежит между -1 и 1 по величине, каков диапазон возможных выходных значений при считывании волновой таблицы, использующем 4-точечную интерполяцию?

Глава 3

Аудиовычисления и управляющие вычисления

3.1 Теорема семплирования

До сих пор мы обсуждали цифровые аудиосигналы, как если бы они допускали описание любой функции времени в том смысле, что знание значений, которые функция принимает на целых числах, должно определить значения, которые она принимает между ними. На самом деле это не так. Например, предположим, что некоторая функция f (определённая для действительных чисел) принимает значение 1 для всех целых чисел:

$$f(n) = 1\,, \quad n = \ \ldots, -1, 0, 1, \ldots$$

Мы могли бы считать, что $f(t) = 1$ для всех действительных t. Но возможно f будет единицей для целых чисел и нулем для всех остальных — это в полной мере также корректно определённая функция, но её значения в целых числах не дают ничего, что отличает эту функцию от просто $f(t) = 1$. Но интуиция говорит нам, что постоянная функция *относится по духу* к цифровым аудиосигналам, тогда как та, которая обращается в ноль между семплами — нет. Функция, которая "возможна для семплирования" должна быть такой, что мы можем использовать для неё

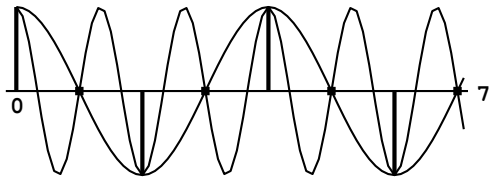


Рис. 3.1: Две действительные синусоиды с угловыми частотами $\pi/2$ и $3\pi/2$ (показано, что они совпадают в целых точках). Цифровой аудиосигнал не позволяет различить две такие синусоиды.

разумную интерполяционную схему для восстановления её значений на нецелых числах из значений на целых.

На этом моменте обсуждения компьютерной музыки является естественным привлечение знаменитой теоремы Найквиста. Она утверждает (грубо говоря), что если функция является конечной или даже бесконечной комбинацией синусоид, угловые частоты которых не превышают π, тогда, по крайней мере теоретически, она полностью определяется своими значениями на целых числах. Один из возможных путей восстановления функции состоит в нахождении предела интерполяций полиномами всё более и более высокого порядка.

Угловая частота π, называемая частотой Найквиста, отвечает $R/2$ циклам в секунду, если R есть частота семплирования. Соответствующий период равен двум семплам[1]. Частота Найквиста является неулучшаемой границей в том смысле, что любая действительная синусоида большей частоты равна для всех целых чисел некоторой другой синусоиде с частотой меньше частоты Найквиста, и эта более низкая частота будет реконструирована с помощью идеального интерполяционного процесса. Например, синусоида с угловой частотой между π и 2π, скажем $\pi + \omega$, может быть записана как

$$\begin{aligned} cos((\pi - \omega)n + \phi) &= cos((\pi - \omega)n + \phi - 2\pi n) \\ &= cos((\omega - \pi)n + \phi) \\ &= cos((\pi + \omega)n + \phi) \end{aligned} \tag{3.1}$$

для всех целых n. (Если n не было бы целым, то первое преобразование не было бы справедливым). Таким образом, синусоида с частотой между π и 2π равна, как минимум для целых чисел, какой-нибудь синусоиде с частотой между 0 и π; вы просто не сможете отличить одну от другой. И так как любое преобразующее оборудование должно делать “правильный” выбор и реконструировать синусоиду с наиболее низкой частотой, любая синусоида с большей частотой, которую вы пытаетесь синтезировать, будет приходит из ваших динамиков с неверной частотой (точнее, вы будете слышать единственную частоту между 0 и π, в которую высокая частота свёртывается при обращении, описанном выше). Этот феномен называется *свёртыванием* (foldover), поскольку полуось частот от 0 до ∞ свёртывается назад и вперёд на длины, кратные π, вплоть до интервала от 0 до π. Термин *наложение спектра* означает то же самое. рис. 3.1 показывает, что синусоиды с угловыми частотами $\pi/2$ и $3\pi/2$, например, не могут быть различимы как цифровые аудиосигналы.

Мы заключаем, что когда, например, мы вычисляем значения ряда Фурье (стр. 13) волновой таблицы или сигнала в режиме реального времени, лучше было бы убрать из суммы любую синусоиду, частота которой превосходит π. Но картина в целом не столь проста, какой она является для конечно-полосных сигналов (таких, чьи компоненты лежат в некоторой ограниченной полосе частот), так что большинство методов, отличных от аддитивного синтеза, приводят к неясным

[1]Угловая частота ω связана с периодом T в семплах формулой $T = 2\pi/\omega$.

последствиям. Например, пилообразная волна частоты ω, выводимая объектом Pd phasor~, но рассматриваемая как непрерывная функция $f(t)$, раскладывается в ряд

$$f(t) = \frac{1}{2} - \frac{1}{\pi}\left(sin(\omega t) + \frac{sin(2\omega t)}{2} + \frac{sin(3\omega t)}{3} + \dots\right),$$

который включает произвольно большие частоты; и, более того, сотая гармоника только на 40 дб слабее, чем первая. Для любых, но очень низких значений частот ω, гармоники, превышающие π, будут слышимо представлены и, вследствие свёртывания, будут прослушиваться с неправильными частотами. Это не означает, что нельзя использовать пилообразные волны в качестве фазовых генераторов (структура процедуры считывания волновой таблицы чудесным образом исправляет свёртывание для пилы), но необходимо подумать дважды перед тем, как использовать пилообразную волну саму по себе, как цифровой источник звука.

Многие методы синтеза, даже не строго конечно-полосные, дают гармоники, которые могут спадать быстрее, чем в $1/n$, как в примере пилообразного сигнала, и, таким образом, являются более приемлемыми для работы с цифровыми сигналами. В любом случае, всегда хорошо иметь в виду возможность возникновения свертки спектра и тренировать свой слух для ее распознавания.

Первая линия защиты от свертки спектров — просто использовать высокую частоту семплирования; хорошей практикой является систематическое использование максимальной частоты дискретизации, с которой ваш компьютер может легко справиться. Наиболее практичная частота будет варьироваться в зависимости от того, работаете ли вы в режиме реального времени или нет, времени ЦПУ и ограничений памяти, устройств входа и выхода и а иногда даже от ограничений, наложенных программным обеспечением.

Достаточно нетехническая трактовка теории семплирования дана в [Bal03]. Более подробную информацию можно найти в [Mat69, стр. 1–30].

3.2 Управление

Ранее мы имели дело с аудиосигналами, которые были просто последовательностями $x[n]$, определёнными для целых n, что отвечает регулярно размещённым точкам во времени. Часто этого достаточо для описания методов синтеза, но реальные приложения электронной музыки обычно также влекут за собой другие вычисления, которые должны быть выполнены в нерегулярные моменты времени. В этом параграфе мы рассмотрим подход для описания того, что мы будем называть *управляющими вычислениями.* Мы будем всегда требовать, чтобы любое вычисление соответствовало специфическому *логическому времени.* Логическое время управляет тем, какой аудиосемпл будет первым на выходе при отображении результата вычисления.

В системе, работающей не в режиме реальном времени (такой как Csound в его классической форме), это означает, что логическое время простирается от

нуля до длины выходного звукового файла. Каждая “нотная карта” имеет соответствующее ей логическое время (время в партитуре) и задействуется, когда аудиовычисление достигнет этого времени. Таким образом, аудиовычисления и вычисления управляющих сигналов (отшлифовка звучания семплов и управление нотными картами) выполняются поочередно в порядке возрастания логического времени.

В системе реального времени, логическое время, которое отвечает времени следующего задействованного семпла аудиовыхода, всегда немного опережает *реальное время*, которое измеряется семплом действительно покидающим компьютер. Управляющие и аудиовычисления, тем не менее, выполняются поочерёдно в соответствии с логическим временем.

Причина использования логического времени и времени, не соответствующего реальному, в вычислениях компьютерной музыки состоит в сохранении вычислений независимыми от реального времени выполнения компьютером, которое может варьироваться по множеству причин, даже для двух кажущихся идентичными вычислений. Когда мы вычисляем новое значение аудиосигнала или задействуем некоторый управляющий вход, реальное время может продолжать идти, но мы требуем, чтобы логическое время оставалось неизменным в течение всего вычисления, как если бы оно производилось моментально. В результате этого электронные музыкальные вычисления (если они произведены корректно) являются детерминированными: два выполнения одного аудиовычисления в реальном или не в реальном времени, входные данные которых одинаковы, должны иметь идентичные результаты.

Рисунок 3.2 (часть *a*) схематично показывает, как логическое время связано с вычислением семпла. Аудиосемплы вычисляются с регулярными периодами (отмеченными волнистыми линиями), но перед вычислением каждого семпла мы совершаем все управляющие вычисления, которые могут его затрагивать (отмечены как прямолинейные отрезки). Во-первых, мы совершаем управляющие вычисления, ассоциированные с логическим временем начинающимся в нуле, и вплоть до, но не включая, единицы; далее мы вычисляем первый аудиосемпл (с индексом ноль) в логическое время один. Далее мы делаем все управляющие вычисления вплоть до логического времени 2, не включая его, потом семпл с индексом один и т. д. (Здесь мы принимаем определенные соглашения обозначений, которые могут быть выбраны по-другому. Например, нет никаких фундаментальных причин, по которым управление должно быть представлено как приходящее “перед” аудиовычислением, но так проще думать).

Часть *b* рисунка показывает ситуацию, когда мы хотим вычислить аудиовыход в блоках с размером более одного семпла по времени. Используя переменную B для обозначения числа элементов в блоке (так $B = 4$ на рисунке), первое аудиовычисление будет выдавать семплы $0, 1, \ldots, B - 1$ одновременно в блоке, вычисленном в логическое время B. Мы должны сделать значимые управляющие вычисления для всех B периодов времени заранее. Имеется задержка в B семплов между логическим временем и появлением аудиовыхода.

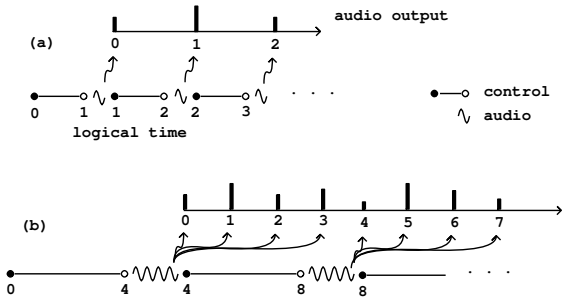


Рис. 3.2: Ось времени для цифрового аудио- и управляющего вычисления: *a* с размером блока в один семпл; *b* с размером блока в четыре семпла.

Большинство программ компьютерной музыки вычисляет аудиоблоками. Это делается для увеличения эффективности индивидуальных аудиоопераций (таких как единичные генераторы в Csound и тильда-объекты Max/MSP и Pd). Каждый единичный генератор или тильда-объект отнимает дополнительное время, равное, в среднем, двадцатикратному вычислению одного семпла, каждый раз когда он вызывается. Если размер блока — единица, это означает 2000% дополнительного времени; если же размер — 64 (что принято по умолчанию в Pd), дополнительное время составляет всего лишь около 30%.

3.3 Управляющие потоки

Управляющие вычисления могут приходить из множества источников, как внутренних, так и внешних по отношению к общему процессу вычисления. Примеры внутренне вызванных управляющих вычислений включают установление последовательности (когда управляющие вычисления должны иметь место в предопределённые моменты) или определение структуры аудиовыхода (например, поиск пересечений нуля в сигнале). Внешне вызванные вычисления могут приходить из входных устройств, таких как контроллер MIDI, компьютерная мышь или клавиатура, сетевые пакеты и т. п. В любом случае управляющие вычисления могут происходить с нерегулярными интервалами, в отличие от аудиосемплов, которые соответствуют стабильному тактовому сигналу.

Нам понадобится способ описания того, как информация переходит между управляющими и аудиовычислениями, который мы будем основывать на понятии *управляющих потоков*. Это просто набор чисел (возможно пустой), которые появляются как результат управляющих вычислений, регулярно или нерегулярно размещённых в логическом времени. Простейший возможный управляющий поток не имеет никакой другой информации кроме временной последовательности:

$$\ldots, t[0], t[1], t[2], \ldots$$

Хотя значения времени лучше представляются в единицах семплов, здесь их значения не дискретизированы, они могут быть произвольными действительными

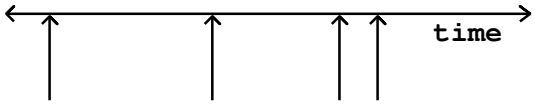


Рис. 3.3: Графическое представление управляющего потока как последовательности точек по времени.

числами. Мы должны потребовать, чтобы они были отсортированы в неубывающем порядке:

$$\ldots \leqslant t[0] \leqslant t[1] \leqslant t[2] \leqslant \ldots$$

Каждое слагаемое в последовательности называется событием.

Управляющие потоки могут быть изображены графически, как показано на рис. 3.3. Числовая ось показывает время, а последовательность стрелок указывает моменты времени, соответствующие каждому событию. Показанный управляющий поток не имеет данных (это последовательность моментов времени). Если мы хотим показать данные в управляющем потоке, мы будем записывать их в основании каждой стрелки.

Численный управляющий поток содержит одно число на одну единицу времени, так что он представляет из себя последовательность упорядоченных пар:

$$\ldots, (t[0], x[0]), (t[1], x[1]), (t[2], x[2]), \ldots,$$

где $t[n]$ — моменты времени и $x[n]$ — величины сигнала в эти моменты.

Численный управляющий поток аналогичен, грубо говоря, “контроллеру MIDI”, значения которого меняются нерегулярно, например, когда физическое управление заменяется исполнителем. Другие источники управляющего потока могут иметь большие возможные скорости изменения и/или больше точности. С другой стороны, временная последовательность могла бы быть последовательностью нажатий педали, которая (несмотря на осуществление MIDI) не должна рассматриваться как имеющая величины, но лишь как моменты времени.

Численные управляющие потоки аналогичны аудиосигналам в том, что они имеют численные значения, меняющиеся во времени. Но, в то время как аудиосигнал поступает с постоянной скоростью (и, таким образом, нет необходимости определять моменты времени для семплов), управляющий поток поступает непредсказуемо: возможно равномерно, возможно неравномерно, возможно никогда.

Давайте сейчас посмотрим, что происходит, когда мы пытаемся перевести численный управляющий поток в аудиосигнал. Как и раньше мы выберем размер блока $B = 4$. Мы будем рассматривать в качестве управляющего потока квадратную волну периода 5.5:

$$(2, 1), (4.75, 0), (7.5, 1), (10.25, 0), (13, 1), \ldots$$

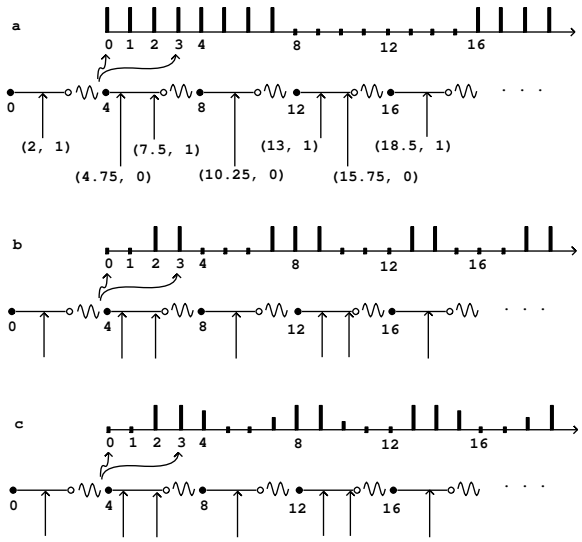


Рис. 3.4: Три способа преобразования управляющего потока в аудиосигнал: (a) наиболее быстрый; (b) задерживающий до ближайшего семпла; (c) с двухточечной интерполяцией для более высокой точности задержки.

и продемонстрируем три способа, которыми она может быть преобразована в аудиосигнал. Рисунок 3.4 (часть a) показывает простейшее, “быстрое насколько возможно”, преобразование. Каждый аудиосемпл выхода просто отражает наиболее близкое значение управляющего сигнала. Так что семплы от 0 до 3 (которые вычисляются в логическое время 4 в силу размера блока) равны 1 по величине в следствие точки (2,1). Следующие четыре семпла также равны единице, поскольку из двух точек, $(4.75, 0)$ и $(7.5, 1)$ самое последнее значение имеет величину 1.

“Быстрое насколько возможно” преобразование больше всего подходит для управляющих потоков, которые меняются редко по сравнению с размером блока. Его главные преимущества состоят в простоте вычисления и наиболее быстром возможном отклике на изменения. Как показывает рисунок, когда обновления управляющего потока слишком быстры (порядка размера блока), аудиосигнал не может быть хорошим отображением спорадического. (Если, как в этом случае, управляющий поток поступает через регулярные интервалы времени, мы можем использовать теорему семплирования для анализа результата. Здесь частота Найквиста, ассоциированная со скоростью блоков R/B, ниже, чем у входной частоты квадратной волны, и, таким образом, выход заменяется на новую частоту, ниже частоты Найквиста.)

Часть (b) показывает результат преобразования “в ближайший семпл”. Каждое новое значение управляющего потока в момент времени t затрагивает выходные семплы, начиная с индекса $\lceil t \rceil$ (наибольшее целое значение не превосходящее t). Это эквивалентно использованию “быстрого насколько возможно” преобразования с блоками размера 1; другими словами, преобразование в ближайший семпл убирает влияние большого размера блоков. Это лучше, чем “быстрое насколько возможно” преобразование в случаях, если управляющий поток может меняться быстро.

Часть (c) показывает преобразование “спорадический сигнал в аудио”, опять в ближайший семпл, но теперь используя при этом двухточечную интерполяцию для большего увеличения временной точности. Концептуально мы можем описать это следующим образом. Предположим, что предыдущее значение управляющего потока было x и следующая точка есть $(n+f, y)$, где n — целое и f — дробная часть значения времени (то есть $0 \leqslant f < 1$). Первой точкой, попадающей в аудиосигнал на выходе, будет семпл с индексом n. Но вместо того, чтобы устанавливать выход равный y, как раньше, мы установим его равным

$$fx + (1-f)y \ ,$$

другими словами, взвешенному среднему предыдущего и нового значений, чьи веса поддерживают другое новое значение, если спорадический сигнал приходит раньше, ближе к n. В показанном примере переход от 0 к 1 в момент времени 2 даёт

$$0x + 1y = 1 \ ,$$

в то время как переход от 1 к 0 в момент времени 4.75 даёт:

$$0.75x + 0.25y = 0.75 \ .$$

Этот метод даёт ещё более близкое представление управляющего потока (по крайней мере, его части, лежащей ниже частоты Найквиста) с бóльшим количеством вычислений и чуть более высокой задержкой.

Численные управляющие потоки могут также преобразовываться в аудиосигналы с использованием линейных функций для сглаживания разрывов. Это часто используется, когда управляющий поток предназначен для управления амплитудой, как описано в §1.5. В общем, имеется три значения для определения такой линейной функции при движении: начальное время, конечное значение (определённое управляющим потоком) и конечное время, часто выраженное в виде задержки после начального времени.

В таких ситуациях выбор сценария “быстро насколько возможно”, описанного выше, почти всегда даёт достаточную точность для подгонки начального и конечного времени, чтобы соответствовать первому аудиосемплу, вычисленному в последующее логическое время. Рисунок 1.3 (часть а) показывает эффект линейного перехода от 0, начинающийся в момент времени 3, к значению 1 в момент времени 9, тут же начинающий возвращение назад в 0 к моменту времени 15, с размером блока $B = 4$. Моменты времени 3, 9 и 15 усечены до 0, 8 и 12 соответственно.

В реальных ситуациях размер блока может быть порядка миллисекунд и подгонка конечных точек линейной функции к границам блока работает хорошо при управлении амплитудами; достижения целевого значения, отличающиеся на дробную часть миллисекунды, редко дают различимую на слух разницу. Однако в других случаях использование линейных функций более чувствительно к времени фиксации конечных точек. Например, если мы хотим делать что-либо с повторением каждые несколько миллисекунд, то вариация длин отрезков будет приводить к слышимому нарушению периодичности.

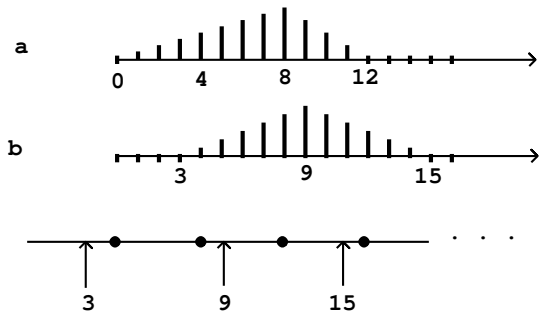


Рис. 3.5: Линейный отрезок, сглаживающий численные управляющие потоки: *a* выравненный по границам блока; *b* выравненный относительно ближайшего семпла.

Для ситуаций, таких как эти, мы можем улучшить алгоритм создания линейных переходов, чтобы иметь возможность начинать и останавливаться в произвольных семплах, как показано, например, на рис. 3.5 (часть *b*). Здесь конечные точки линейных отрезков выстраиваются в линию точно вровень с запрашиваемыми семплами 3, 9 и 15. Мы можем идти даже дальше и устраивать подгонку для дробных семплов, заставляя линейные отрезки достигать значений 0 и 1 в точно определённых точках числовой оси.

Например, предположим, что мы хотим повторить записанный звук из волновой таблицы 100 раз в секунду каждые 441 семплов при обыкновенной частоте семплирования. При приближении, ошибки, произведённые составлением блоков с границами в 64 семпла, могли бы расстроить воспроизведение на целый тон по высоте; даже границы в один семпл могут давать изменение в 0.2% или три цента. Эта ситуация могла бы потребовать точность меньше чем в один семпл при преобразовании “спорадический сигнал в аудио”.

3.4 Преобразование аудиосигналов в потоки числового управления

Нам иногда нужно осуществлять преобразование в обратном направлении: аудиосигнал в спорадический сигнал. Чтобы двигаться в этом направлении мы, так или иначе, должны предоставить ряд логических моментов времени (временную последовательность), а также сам звуковой сигнал. На выходе нам нужен управляющий поток, образованный из временной последовательности с величинами, взятыми из звукового сигнала. Это делается, когда мы хотим включить величины сигнала в качестве части управляющего вычисления.

Например, мы могли бы управлять амплитудой сигнала, используя объект line~ как в примере A03.line.pd. Предположим, что мы хотим выключить звук с некоторой фиксированной скоростью выключения, а не через фиксированный момент времени. Например, нам может понадобиться повторно использовать нашу звуковую систему уже для другого звука, и мы захотим выключить текущий звук

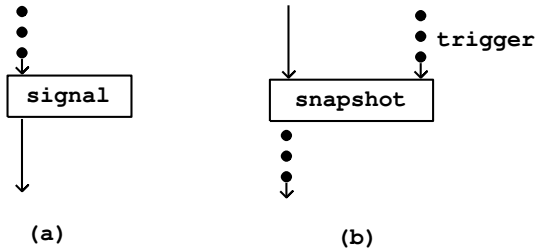


Рис. 3.6: Преобразование между управляющим и аудиосигналами: (a) управление в сигнал; (b) сигнал в управление “моментальными снимками”.

как можно быстрее без слышимых артефактов; вероятно, мы сможем выключить его за меньшее время при низкой амплитуде звука, со сравнению с тем, когда она высока. Для того чтобы это сделать, мы должны послать на объект line~ сообщение для перехода его в ноль в течение времени, вычисленного на основе текущего выходного значения. Для этого потребуется, прежде всего, “семплировать” вывод объекта line~ (аудиосигнал) в управляющий поток.

Результирующая временная задержка и точность, появляющиеся при этом, будут такими же, как при преобразовании спорадического сигнала в аудио. Опять будет появляться альтернатива между моментальностью и точностью. Предположим, как раньше, что мы вычисляем аудиоблоками по 4 семпла, и предположим, что в логическое время 6 мы хотим посмотреть значение аудиосигнала и использовать его для изменения значения другого аудиосигнала. Как показано на рис. 3.2, (часть b) последнее вычисленное значение сигнала будет для индекса 3 и самый ранний индекс, на сигнал с которым наше вычисление может воздействовать, равен 4. Поэтому мы можем выполнить задачу полностью с задержкой всего в один семпл. Однако мы не можем выбрать точно *на какой* — обновление может появиться только на границе блока.

Как и раньше, мы можем пожертвовать моментальностью для возросшей точности времени. Если имеет значение, на каком семпле в точности мы выполняем вычисление “аудио в управление” и “управление в аудио”, то мы считываем семпл с индексом 2 и обновляем семпл с индексом 6. Далее, если мы хотим сделать то же самое опять в логическое время 7, мы считываем семпл с индексом 3 и обновляем семпл с индексом 7 и т. д. В общем, если размер блока есть B, и для любого индекса n мы можем всегда считать семпл с индексом $n - B$, а так же воздействовать на семпл с индексом n, то, таким образом, имеется постоянная задержка в B семплов при преобразованиях в вычислении аудио $\rightarrow$ управление $\rightarrow$ аудио, что является платой, вытекающей из возможности определения индекса n точно.

Если мы хотим продвинуться дальше, так чтобы иметь возможность определить дробную часть семпла, тогда (как раньше) мы можем использовать интерполяцию (с небольшой дополнительной задержкой). В общем, как и в случае преобразования спорадического сигнала в аудио, в большинстве ситуаций простейшее решение — наилучшее, но иногда мы должны делать дополнительную работу.

3.5 Управляющие потоки в виде блок-схем

Рисунок 3.6 показывает, как управляющие потоки выражаются с помощью блок-схем с использованием преобразований “управление в сигнал” и “сигнал в управление” в качестве примеров. Управляющие потоки представлены точками (в противоположность аудиосигналам, которые появляются в виде сплошных стрелок).

Блок **signal** преобразовывает численный управляющий поток в аудиосигнал. Точный вид преобразования не определён на настоящем уровне детализации; в примерах Pd выбор оператора преобразования задаст его.

Блок **snapshot** (моментальный снимок) преобразует аудиосигналы обратно в численные управляющие потоки. В дополнение к аудиосигналу, отдельный управляющий вход нужен для определения временной последовательности, по которой семплируется аудиосигнал.

3.6 Детекция события

Кроме моментальных снимков, второй способ переведения информации из аудиосигнала в управляющие вычисления состоит в *детекции события*. Здесь мы выводим информацию о времени из аудиосигнала. Примером является *детекция порога*, при котором входом является аудиосигнал, а выходом — временная последовательность. Здесь мы рассмотрим пример детекции порога более подробно.

Типичная причина использования детекции порога состоит в нахождении того, когда начинается и прекращается какая-либо активность, например, исполнитель играет на музыкальном инструменте. Предположим, что у нас уже есть непрерывная мера этой активности в виде звукового сигнала. (Это может быть сделано, например, с использованием *повторителя огибающей*). Нам нужна пара временных последовательностей: одна — отмечающая те моменты времени, в которые активность начинается, и вторая — для тех, в которых она заканчивается.

Рисунок 3.7 (часть a) показывает простейшую реализацию этой идеи. Мы предполагаем, что входной сигнал такой, как показано на непрерывном графике. Горизонтальная линия показывает постоянное значение порога. Временная последовательность, обозначенная “onsets”[2], содержит одно событие для каждого момента времени, когда сигнал пересекает порог снизу вверх; последовательность “turnoffs”[3] отмечает пересечения в обратном направлении.

Во многих ситуациях мы будем получать ненужные атаки и выключения, вызванные небольшими пульсациями сигнала около порогового значения. Этого удаётся избежать с помощью *устранения пульсаций*, которое можно сделать, по крайней мере, двумя простыми способами. Во-первых, как показано в части (b) рисунка,

[2]onset — атака
[3]turnoff — выключение

мы можем установить два порога: высокий — для детекции атак, и более низкий — для выключений. В этой схеме правило состоит в том, что мы сообщаем только о первой атаке после каждого выключения и, *наоборот*, мы сообщаем только о единственном выключении после каждой атаки. Таким образом, когда третий раз сигнал пересечёт высокий порог, это не будет расценено как атака, поскольку не было выключения с момента предыдущей атаки. (При запуске мы действуем так, как если бы самое последнее событие было выключением, чтобы сообщить о первой атаке)

Второй подход к фильтрации множества атак и выключений, показанный в части (*c*) рисунка, состоит в сопоставлении *мёртвого периода* каждой атаке. Этот период — постоянный интервал времени после каждой отмеченной атаки, в течение которого мы отказываемся от регистрации других атак или выключений. После окончания периода, если сигнал упал ниже порога в прошедшее время, мы отмечаем с задержкой выключение. Мёртвые периоды могут также быть сопоставлены выключениям, и эти два периода времени могут иметь различную протяжённость.

Две стратегии фильтрации могут быть использованы по отдельности или одновременно. Как правило, необходимо вручную адаптировать пороговые значения и/или мертвые периоды к каждой конкретной ситуации, в которой используется пороговое значение.

Использование пороговых значений часто используется в качестве первого шага при создании высокоуровневых стратегий для организации реакции компьютера на слышимые сигналы, подаваемые исполнителями. Простым примером может быть запуск последовательности заранее запланированных действий, каждое из которых должно быть запущено при атаке звука после определенного периода относительной тишины, например, если бы музыкант играл последовательность фраз, разделенных паузами.

Более сложные детекторы (построенные на основе пороговой детекции) могут различать непрерывный звук или тишину внутри ожидаемого диапазона длительностей, или последовательности быстрых чередований между игрой и её отсутствием, или периоды времени, в течение которых процент времени игры по отношению к паузам выше или ниже некоторого порога, или много других возможных особенностей. Они могут вызвать заранее определенные реакции или использоваться в импровизации.

3.7 Использование аудиосигналов для управления

Из традиции аналогового синтеза происходит элегантный подход старой формации к задачам управления, который может быть использован как альтернативный управляющим потокам, затронутым так глубоко в этой главе. Вместо них или в дополнение к использованию управляющих потоков, мы можем использовать аудиосигналы сами по себе для управления производством других аудиосигналов. Два специфических метода успешно заимствуются у аналогового синтеза при таком

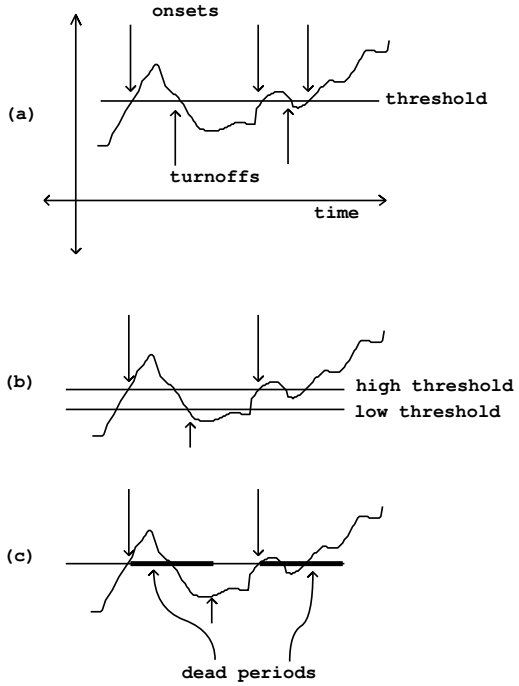


Рис. 3.7: Детекция порога: *a* без устранения пульсаций; *b* с устранением пульсаций двумя пороговыми уровнями; *c* с устранением пульсаций с использованием мёртвых периодов.

подходе: аналоговое секвенцирование и техника “захвата и удерживания”[4].

Аналоговый секвенсер [Str95, сс. 70–79] [Cha80, сс. 93, 304–308] часто использовался для воспроизведения регулярно или полурегулярно повторяющейся последовательности звуков. Секвенсер сам по себе обыкновенно производил повторяющуюся последовательность напряжений вместе с сигналами триггера, которые периодически возникали в каждом переходе между напряжениями. Напряжения использовались как параметры высоты или тембра, а триггер для управления одним или несколькими генераторами огибающей. Получение циклических последовательностей предопределённых значений в практике цифрового аудио столь же просто, как помещение объекта phasor~ при считывании волновой таблицы без интерполяции. Если вы хотите, скажем, четыре значения в последовательности, масштабируйте выход phasor~, чтобы иметь значения от 0 до 3.999..., так что в первую четверть цикла считывается точка 0 таблицы и так далее.

Для получения периодических значений триггера нужно, во-первых, синтезировать другую пилообразную волну, которая двигается синхронно с выходом phasor~, но в четыре раза быстрее. Это делается с использованием варианта метода рис. 2.8, на котором мы использовали операции сложения и циклического возврата для достижения желаемого сдвига фазы. Рисунок 3.8 показывает эффект умножения пилообразной волны на целое число с последующим круговым зацикливанием для получения пилы с частотой, кратной первоначальной.

[4]sample-and-hold — устройство выборки и хранения

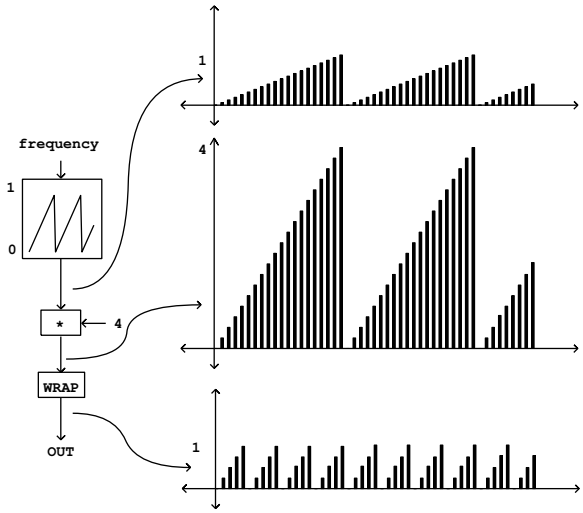


Рис. 3.8: Умножение и зацикливание пилообразной волны для генерации волны большей частоты.

Таким образом, легко получить повторяющуюся форму огибающей для, например, считывания волновой таблицы (используя считывание таблицы с интерполяцией на этот раз, в отличие от случая с последовательностью напряжений). Различные способы генерирования волн и другие методы, используемые для создания высот звуков, могут быть также привнесены для использования здесь.

Другой стандартный метод управления, восходящий к аналоговому синтезу, выполняется устройством выборки и хранения [Str95, сс. 80–83] [Cha80, стр. 92]. Оно принимает приходящий сигнал, захватывает определённые его мгновенные значения и “замораживает” эти значения на своём выходе. Определённые значения для такого захвата отбираются вторичным, “триггерным” входом. В моменты времени, определённые появлением нового сигнала на триггерном входе, одно значение берётся с первоначального входа и сохраняется непрерывно на выходе до следующего момента времени, когда оно заменяется на новое значение с первоначального входа.

В цифровом аудио часто полезно брать образец для нового значения на падающих краях сигнала триггера, то есть когда текущая величина сигнала триггера меньше, чем его предыдущая величина, как это показано на рис. 3.9. Это особенно удобно для использования с пилообразным триггером, когда мы хотим семплировать сигналы синхронно с процессом, управляемым осциллятором. Объект выборки и хранения Pd был ранее введён в контексте семплирования (пример B08.sampler.nodoppler.pd, стр. 53).

3.8 Операции над управляющими потоками

До сих пор мы обсуждали как осуществлять преобразования между управляющими потоками и аудиосигналами. В дополнение к этой возможности имеется

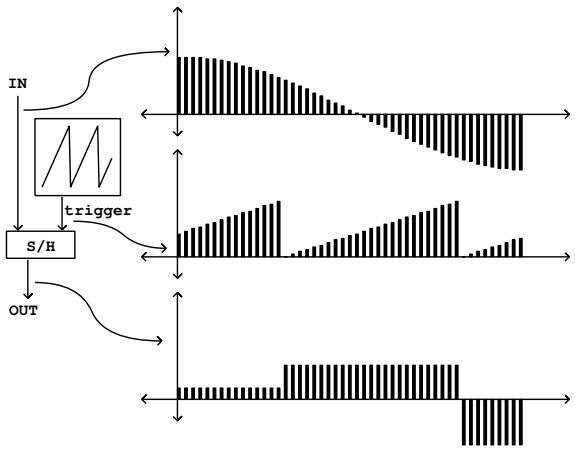


Рис. 3.9: “Выборка и хранение” (“S/H”) с использованием спадающих краёв сигнала триггера.

четыре типа операций, которые вы можете сделать над управляющими потоками, для того чтобы получить другие управляющие потоки. Эти операции над управляющими потоками не имеют соответствий среди операций над аудиосигналами. Их существование объясняет в большей степени, почему полезно ввести целую структуру управления параллельно с аудиосигналами.

Первый тип состоит из операций задержки, которые компенсируют значения времени, связанные с управляющим потоком. В системах реального времени задержки не могут быть отрицательными по величине. Управляющий поток может быть задержан на постоянную величину или, наоборот, вы можете задержать каждое событие отдельно на различные величины.

Два разных типа задержки используются на практике: простая и сложная. Примеры каждой из них показаны на рис. 3.10. Простая задержка, действующая на управляющий поток, назначает каждому событию, когда оно приходит, определённое время в будущем. Однако если другое событие приходит на вход перед тем, как первое событие послано на выход, то первое событие забывается в интересах второго. При сложной задержке каждое событие на входе производит выход, даже если другие входы приходят перед тем, как выход появился.

Вторая операция над управляющими потоками представляет из себя слияние: взятие двух управляющих потоков и комбинирование всех их событий в один. Рисунок 3.11 (часть a) показывает, как эта и остающиеся операции представляются блок-схемами.

Часть (b) рисунка показывает эффект слияния двух потоков. Потоки могут содержать более, чем одно событие в один момент времени. Если два сливающихся потока содержат относящиеся к одному моменту времени события — объединенный поток содержит оба этих события в корректно определённом порядке.

Третий тип операции над управляющими потоками есть обрезание. Обрезание управляющего потока означает считывание ассоциированных с ним данных и про-

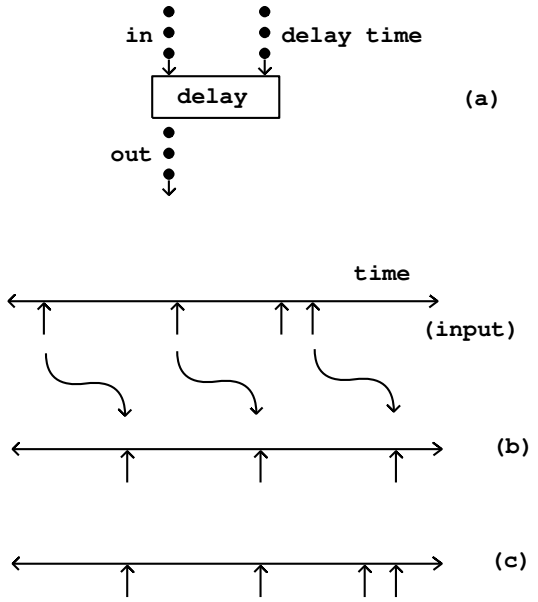


Рис. 3.10: Задержка как операция над управляющим потоком: (a) блок-схема; (b) эффект простой задержки на управляющий поток; (c) эффект сложной задержки.

пускание только определённых элементов. Часть (c) показывает пример, в котором события (каждое из которых имеет ассоциированный с ним номер) пропускаются только если этот номер положительный.

Наконец, имеется концепция пересинхронизации одного управляющего потока в другой, как показано в части (d). Здесь один управляющий поток (источник) представляет значения, которые помещаются на временную последовательность второго (синхронизирующего). Значение, дающееся на выходе, всегда наиболее раннее из потока источника. Заметим, что каждое событие из источника может появиться больше одного раза (как отмечено на рисунке) или, с другой стороны, оно может не появиться вообще.

И снова мы должны рассмотреть, что случается, когда каждый из двух потоков содержит событие в один и тот же момент времени. Должно ли синхронизирующее событие рассматриваться как случившееся перед событием источника (так что выход принимает значение предыдущего события источника)? Или событие источника должно рассматриваться как бывшее первым, так что его значение идёт на выход в то же самое время? Как данная неоднозначность должна быть решена — это вопрос проектирования, который в различных программных средах имеет разные подходы. (В Pd это задаётся непосредственно пользователем.)

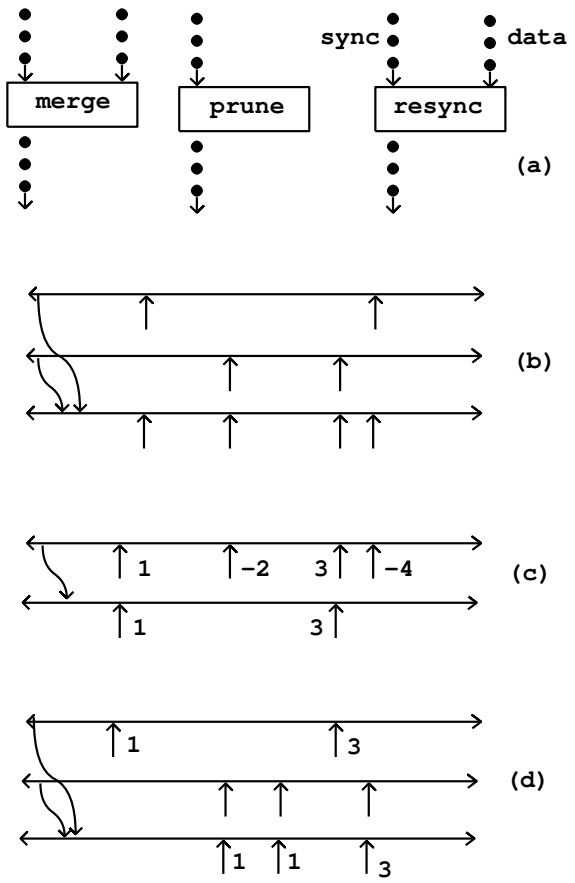


Рис. 3.11: Операции над управляющими потоками (кроме задержки): (a) блок-схемы; (b) слияние; (c) обрезание; (d) пересинхронизация.

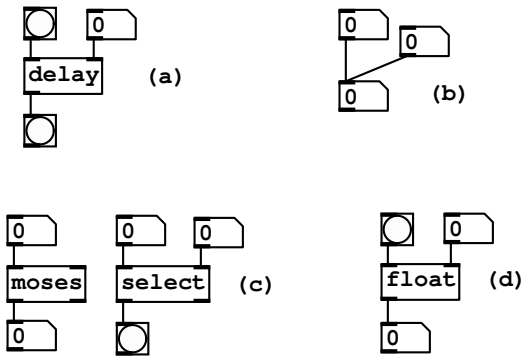


Рис. 3.12: Четыре управляющих операции в Pd: (*a*) задержка; (*b*) слияние; (*c*) обрезание; (*d*) пересинхронизация.

3.9 Управляющие операции в Pd

До сих пор мы использовали Pd в основном для обработки звуковых сигналов, хотя уже на рис. 1.10 нам пришлось провести различие между понятием Pd аудиосигналов и управляющих потоков: тонкие соединения передают управляющие потоки, а толстые — аудио. Управляющие потоки в Pd являются последовательностями *сообщений*. Эти сообщения могут содержать какие-то данные (чаще всего одно или несколько чисел) или не содержать их вообще. *Числовой управляющий поток* (§3.3) отображается как (тонкое) соединение, которое передаёт числа в виде сообщений.

Сообщения, не содержащие данные, образуют *временные последовательности*. Таким образом, вы можете видеть сообщения без данных, в Pd они обозначаются (произвольным) символом “bang”[5].

Четыре типа управляющих операций, описанных в предыдущем, параграфе могут быть выражены в Pd, как показано на рис. 3.12. Задержки выполняются с использованием двух конкретных объектов задержки:

[del], [delay]: *простая задержка*. Вы можете определять время задержки в аргументе или отправляя сообщения на правый вход. “bang” на левом входе устанавливает задержку, которая далее выводит “bang” после определённой в миллисекундах задержки. Задержка простая в том смысле, что отправка “bang” на уже установленную задержку переустанавливает устройство на новое выходное время, отменяя определённое ранее.

[pipe]: *сложная задержка*. Сообщения, приходящие на левый вход, появляются на выходе после определённой задержки, которая устанавливается первым начальным аргументом. Если указано несколько начальных аргументов, они определяют один или несколько входов для числовых или символьных данных, которые сообщения будут содержать. Любое число сообщений может

[5]удар (*прим. перев.*)

храниться в pipe одновременно, и сообщения могут быть переупорядочены при отправке, в зависимости от различных заданных для них времён задержки.

Слияние управляющих потоков в Pd выполняется не конкретными объектами, а механизмом связей Pd самим по себе. Это показано в части (b) рисунка с числовыми окнами в качестве примера. В общем, когда с управляющим входом связано более одного соединения, управляющие потоки объединяются.

Pd предлагает несколько объектов для обрезания управляющих потоков, из которых два показаны в части (c) рисунка:

moses: *разделение числового потока*[6]. Численные сообщения, приходящие на левый вход, появляются на левом выходе, если они меньше пороговой величины (установленной начальным аргументом или правым входом), и, в противном случае, выводится правый вход.

select, sel: *выбор определённых чисел.* Численные сообщения, приходящие на левый вход, производят "bang" на выходе только в том случае, когда они в точности соответствуют тестовой величине. Тестовая величина устанавливается или начальным аргументом, или правым входом.

Наконец, Pd неявно обеспечивает пересинхронизацию управляющих потоков через механизм их связей, как показано в части (d) рисунка. Большинство объектов с более чем одним входом синхронизуют все входы по самому левому из них. Так, объект float, показанный на рисунке, пересинхронизует правый вход (который получает числа) левым. При получении "bang" на левый вход выводит самое последнее число, которое float получил перед этим.

3.10 Примеры

3.10.1 Семплирование и наложение спектра

Пример C01.nyquist.pd (рис. 3.13, часть a) показывает осциллятор, воспроизводящий волновую таблицу, у которого несущая частота изменяется в диапазоне от 500 до 1423. Волновая таблица состоит только из 46-й гармоники, которая, следовательно, варьируется от 23000 до 65458 герц. С частотой семплирования 44100 эти две частоты звучат теоретически как 22100 и 21358 герц, но переход от одной к другой производит свёртывание до нуля и возврат назад.

Две другие формы волны приведены для того, чтобы показать интересные эффекты биений между гармониками, которые, хотя и "должны" быть разделены

[6] название этого объекта в Pd имеет элемент своеобразного юмора — этот объект, как библейский Моисей, разделяет "море" чисел (*прим. перев.*)

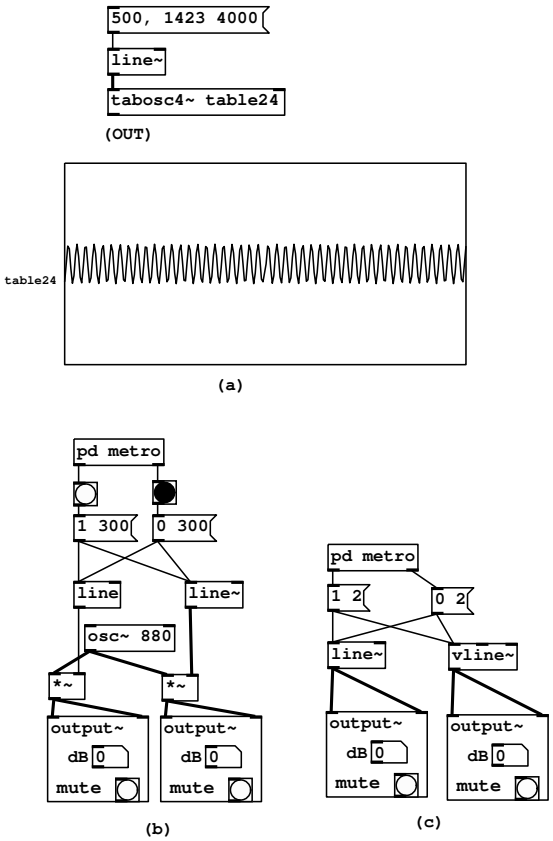


Рис. 3.13: *a* задание осциллятора с превышением частоты Найквиста; *b* шум “расстёгивающейся молнии” от управляющего объекта line; *c* совмещение объектов line~ и vline~.

друг от друга, оказываются рядом из-за свёртывания. Например, для 1423 герц вторая гармоника 2846 Гц, в то время как 33-я гармоника звучит на $1423 * 33 - 44100 = 2859$ герц — грубый диссонанс.

Другие, менее экстремальные примеры могут дать слышимое наложение спектра в менее сильных формах. Обыкновенно это также нежелательно, и лучше научиться слышать этот эффект. Пример C02.sawtooth-foldover.pd (не изображённый здесь) демонстрирует свёртывание для пилы (объект phasor~). Для волновых таблиц, хранящих аудиозаписи, ошибка интерполяции может создать дополнительное свёртывание. Эффекты этого могут варьироваться широко; звук иногда описывается как “хрустящий” или “брызгающий”, в зависимости от записи, транспозиции и алгоритма интерполяции.

3.10.2 Преобразование управляющего сигнала в аудиосигналы

Пример C03.zipper.noise.pd (рис. 3.13, часть *b*) демонстрирует эффект преобразования медленно обновляющегося управляющего потока в аудиосигнал. Здесь

введён новый объект:

> line: *генератор линейного сигнала с управляющим выходом.* Как line~, line берёт пары чисел (конечное значение, время) и строит прямую линию до конечного значения за данный промежуток времени; однако в отличие от line~, выходом является численный управляющий поток, появляющийся, по умолчанию, с 20–миллисекундными временными интервалами.

В примере вы можете сравнить звук возрастающей и падающей амплитуды, управляемой выходом line, с тем, который управляется аудиосигналом, произведённым line~.

Выход line преобразуется в аудиосигнал на входе объекта *~. Преобразование здесь выполняется присоединением численного управляющего потока к сигнальному входу. В Pd неявные преобразования численных управляющих потоков в аудиопотоки выполняются методом “быстро насколько можно”, показанном на рис. 3.4 (часть *a*). Выход line становится ступенчатым сигналом с 50 ступеньками в секунду. Такой результат называют обыкновенно “шумом расстёгивающейся молнии”.

В то время как ограничения объекта line для генерации аудиосигналов были ясно слышимы даже за такие длинные периоды как 300 миллисекунд, вариант сигнала line~ не выдаёт слышимых на слух проблем, пока включённые периоды времени не станут более короткими. Пример C04.control.to.signal.pd (рис. 3.13, часть *c*) демонстрирует эффект использования line~ для генерирования треугольной волны с частотой 250 герц. Здесь при воспроизведении появляются эффекты, показанные на рис. 3.5. Поскольку line~ всегда выстраивает в линию прямолинейные отрезки на границах блока, точные продолжительности этих отрезков меняются и, в примере, изменение (порядка миллисекунд) является существенной долей их длины.

Для таких ситуаций предназначается более точный объект (и более дорогостоящий в смысле затрат на вычисления):

> vline~: *точный генератор линейного отрезка.* Этот третий член семейства “line” выводит аудиосигнал (как line~), но выстраивает конечные точки сигнала в требуемые моменты времени с точностью до доли семпла. (Точность ограничена лишь числовым форматом с плавающей точкой, принятым в Pd.) Далее много линейных отрезков могут быть определены внутри одного аудиоблока; vline~ может генерировать формы волн с длинами периодов с ограничением снизу вплоть до двух семплов (при меньшем периоде вы получите лишь свёртывание).

Объект vline~ может также быть использован для преобразования численных управляющих потоков в аудиопотоки методами ближайшего семпла и двухточечной интерполяции, как показано на рис. 3.4 (части *b* и *c*). Для получения преобразования ближайшего семпла просто задайте vline~ нулевое время преобразования. Для линейной интерполяции задайте время преобразования в один семпл (0.0227 миллисекунд при частоте семплирования 44100 герц).

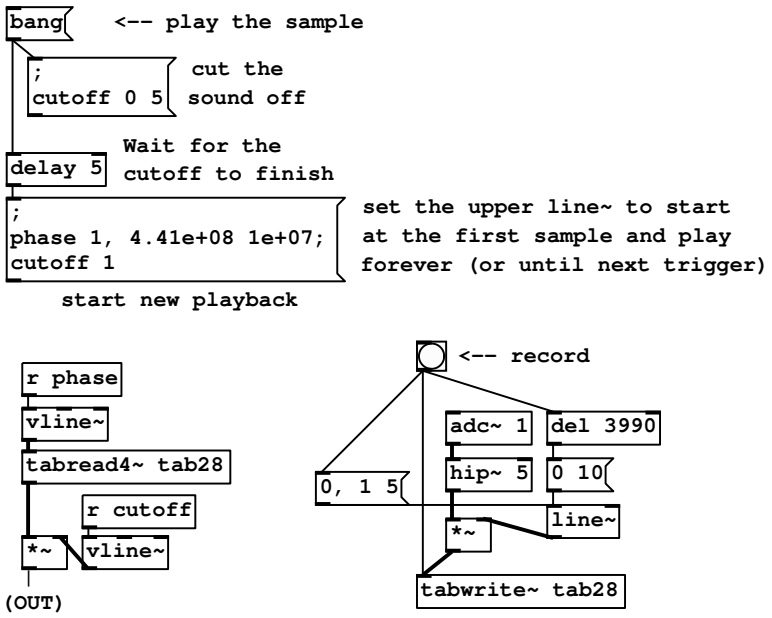


Рис. 3.14: семплер без зацикливания.

3.10.3 Воспроизведение волновой таблицы без зацикливания

Одной из областей приложения, требующей внимательного отношения к границе сигнала управляющего потока или аудио, является семплирование. Вплоть до этого момента наши семплеры воспроизводили по кругу на выходе в силу постоянного зацикливания. Это может выдавать богатое множество звуков, к которому может быть получен доступ с помощью непрерывных изменений параметров, таких как размер цикла и форма огибающей. Тем не менее, во многих случаях использования семплирования, требуется предоставление внутреннего содержания волновой таблицы в предсказуемые, синхронизованные моменты времени. Например, записанные перкуссионные звуки обыкновенно воспроизводятся с начала, редко с зацикливанием и обыкновенно в определённом временном соответствии с остальной музыкой.

В этой ситуации управляющие потоки лучше подходят, чем аудиосигналы в качестве триггеров. Пример C05.sampler.oneshot.pd (рис. 3.14) показывает один из возможных способов сделать это. Четыре звуковых объекта слева внизу образуют граф для воспроизведения сигнала. Первый объект vline~ производит фазовый сигнал (в действительности просто индекс для считывания таблицы) для объекта tabread4~; это заменяет phasor~ в примере B03.tabread4.pd (стр. 50) и его производные.

Амплитуда выхода tabread4~ управляется вторым объектом vline~ для того, чтобы предотвратить разрывы на выходе в случае начала нового события, в то время как предыдущее событие ещё воспроизводится. “Срезающий” объект vline~ линейно уводит выход на ноль (во время воспроизведения или нет), так что, когда выход нулевой, индекс волновой таблицы может быть скачкообразно изменён.

Для того чтобы начать новую “ноту”, во-первых, “срезающий” объект vline~ сводит сигнал линейно на ноль; далее, с задержкой в 5 миллисекунд (за которые vline~ достигнет нуля) фаза переустанавливается. Это делается двумя сообщениями: во-первых, фаза устанавливается на 1 (без назначения времени, так что она прыгает на 1 без линейного сглаживания). Значение “1” определяет первую читаемую точку волновой таблицы, так как мы используем 4-точечную интерполяцию. Во-вторых, в том же окне сообщения фаза линейно меняется до 441 000 000 за временной период в 10 000 000 миллисекунд. (В Pd большие числа показываются при помощи обозначений в экспоненциальной форме; два числа, приведённые выше, появляются как $4.41e + 08$ и $1e + 07$.) Отношение равно 44.1 (в единицах на миллисекунду), что даёт транспозицию. Объект vline~ сверху (который генерирует фазу) получает эти сообщения с помощью объекта r phase выше.

Пример предполагает, что волновая таблица линейно сглажена к нулю в одном из концов, и нижняя правая часть патча показывает, как записать такую волновую таблицу (четырёхсекундной длины в этом случае). Здесь достаточно регулярного (и более дешёвого в смысле вычислений) объекта line~. Хотя волновая таблица должна иметь, по крайней мере, 4 секунды в длину, чтобы всё сработало, вы можете записывать более короткие волновые таблицы, просто срезая их объектом line~ раньше. Единственное предостережение состоит в том, что если вы одновременно читаете и записываете в одну таблицу, нужно избегать ситуаций, где операции чтения и записи обращаются к одной и той же части таблицы одновременно.

Объекты vline~, окружающие tabread4~, были выбраны вместо line~, поскольку округление последними точек разрыва до ближайшей границы блока (обыкновенно 1.45 миллисекунд) может произвести слышимые непериодичности в звуке, если волновая таблица повторяется чаще, чем 10–20 раз в секунду, и может препятствовать получению хорошего периодичного звука при высоких скоростях повторения.

Мы вернёмся к семплированию на основе vline~ в следующей главе, при обращении к транспозиции, огибающим и полифонии.

3.10.4 Сигналы в управления

Пример C06.signal.to.control.pd (не показан) демонстрирует преобразование аудиосигналов в численные управляющие потоки с помощью нового тильда-объекта, определённого здесь.

> snapshot~: *преобразование аудиосигнала в управляющие сообщения.* Этот объект всегда выдаёт самый последний вычисленный аудиосемпл (преобразование “быстрое насколько можно”), так что точное время семплирования варьируется вплоть до одного аудиоблока.

Часто требуется узнать среднюю амплитуду аудиосигнала, а не её пик на одном семпле; пример C07.envelope.follower.pd (также не показанный) определяет другой объект, который осуществляет это:

env~ : *вычисление среднеквадратичной (RMS) огибающей.* Выводит управляющие сообщения, дающие кратковременную среднеквадратичную (RMS) амплитуду (в децибелах) аудиосигнала на входе. Аргумент инициализации позволяет вам указать число семплов, используемых при вычислении амплитуды; меньшие числа дают более быстрый (и, возможно, менее постоянный) вывод.

3.10.5 Секвенсер в аналоговом стиле

Пример C08.analog.sequencer.pd (рис. 3.15) реализует аналоговый секвенсер и генератор огибающей, описанные в §3.7. "Таблица последовательности" с девятью элементами хранит последовательность частот. Объект phasor~ сверху осуществляет циклическое прохождение таблицы последовательности с частотой 0.6 герц. Считывание таблицы без интерполяции (tabread~ вместо tabread4~) используется для считывания частот с дискретным шагом. (Такие ситуации, в которых мы предпочитаем считывание таблицы без интерполяции, редки.)

Объект wrap~ преобразовывает пилообразный сигнал с амплитудой 9 в сигнал с единичной амплитудой, как описано раньше на рис. 3.8, эта пила далее используется для получения огибающей функции из второй волновой таблицы. Это используется для управления размером гранулы в циклическом семплере (из §2.6). Здесь волновая таблица состоит из шести периодов синусоиды. Гранулы сглаживаются умножением на поднятую функцию косинус (cos~ и + 1).

Пример C09.sample.hold.pd (не показан здесь) показывает устройство выборки и хранения — еще одно полезное устройство для выполнения задач управления в области звукового сигнала.

3.10.6 Синтезатор в MIDI стиле

Пример C10.monophonic.synth.pd (рис. 3.16) также осуществляет монофонический нотно-ориентированный синтезатор, но в этом случае ориентированный на управление по MIDI. Здесь задачи генерации огибающей и секвенсирования высот осуществляются с использованием управляющих потоков вместо аудиосигналов. Новые управляющие объекты необходимы для этого примера:

notein : *Ввод нот из MIDI устройства.* Три выхода выводят номер ноты, силу нажатия и канал, по которому приходят MIDI события "нажатие ноты"[7] и "снятие ноты"[8] (с событиями "снятие ноты", появляющимися в виде события "нажатие ноты" с нулевой силой нажатия). Выходы, по обычаю Pd, упорядочены справа налево.

[7] Note On
[8] Note Off

stripnote: *Фильтрация сообщений "снятие ноты"*. Пропускает пары (номер ноты, сила нажатия), только если сила нажатия ненулевая, отбрасывая остальные. В отличие от notein, stripnote не использует напрямую устройства входа и выхода MIDI.

trigger, t: *Копирование сообщения на выходы в порядке справа налево, с преобразованием типа.* Начальные аргументы ("b" и "f" в этом примере) определяют два выхода: один — выводящий сообщения "bang", а другой — "float" (то есть числа). Для каждого аргумента создаётся один выход. Выходы появляются в стандартном для Pd порядке справа налево.

Управляющие объекты патча направляют частоты на объекты phasor~, когда сообщение "нажатие ноты" MIDI получено. Управление амплитудой (через объект line~) более сложно. Когда сообщение "нажатие ноты" получено, объект sel 0 выводит силу нажатия в правый выход (поскольку вход не может быть 0); она делится на максимальную силу нажатия MIDI 127 и упаковывается в сообщение для line~ с временем 100 миллисекунд.

Однако когда получено событие "снятие ноты", остановить звук возможно, только если высота "снятия ноты" действительно соответствует ноте, которую инструмент играет. Например, предположим, что полученные сообщения есть "60 127", "72 127", "60 0" и "72 0". Когда "нажатие ноты" с высотой 72 приходит, нота должна быть изменена на 72 и, далее, сообщение "60 0" должно быть проигнорировано для ноты, воспроизводимой вплоть до получения сообщения "72 0".

Для осуществления этого, во-первых, мы сохраняем силу нажатия в объекте float сверху. Во-вторых, когда высота приходит, она также сохраняется (нижним объектом float) и, далее, сила нажатия проверяется на равенство нулю (выход "bang" t b f возвращает силу нажатия, которая отправляется на sel 0). Если она нулевая, второй шаг состоит в возвращении высоты и сравнении (объектом select) её с самой последней высотой сообщения "нажатие ноты". Только если они равны (так что "bang" появляется на левом выходе select) сообщение "0 1000" идёт на объект line~.

Упражнения

1. Сколько обертонов тона A 440 могут быть представлены в цифровом формате с частотой семплирования 44100 герц ?

2. Какую частоту вы бы услышали, если синтезируете синусоиду в 88000 герц с частотой семплирования 44100?

3. Предположим, что вы синтезируете звук в 44100 герц и вычисляете 64-семпловые аудиоблоки. Назначено, что управляющее событие случается по прошествии ровно одной секунды с использованием схемы обновления "быстро насколько можно". На каком семпле обновление реально случится?

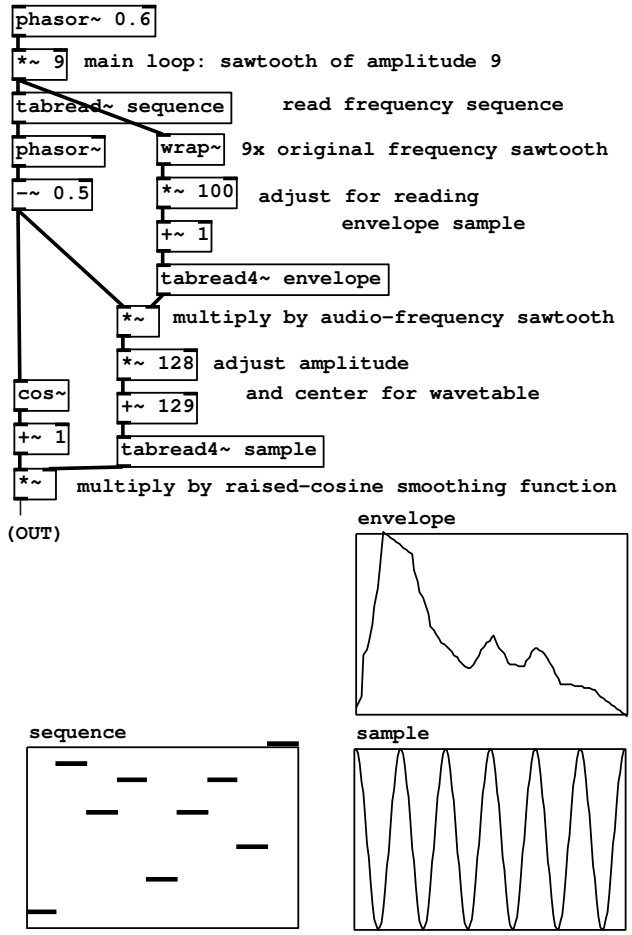


Рис. 3.15: Секвенсер синтезатора аналогового стиля.

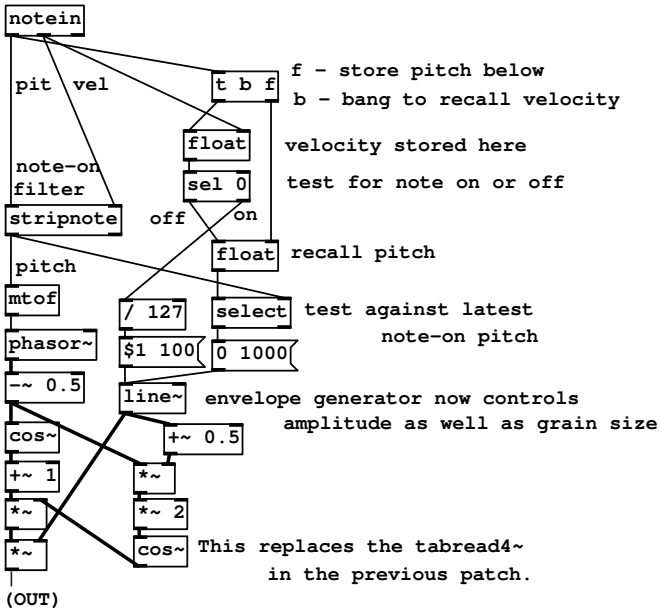


Рис. 3.16: Монофонический синтезатор стиля MIDI.

4. Семплируя на 44100 Гц, мы хотим приблизительно воспроизвести тон До первой октавы, повторяя фиксированную форму волны каждые N семплов. Какое значение N мы должны выбрать и на сколько центов (стр. 7) получится отличие от “реального” До первой октавы?

5. Две пилообразных волны единичной амплитуды имеют частоты 200 и 300 герц соответственно. Какова периодичность их суммы? Что если вы, далее, свернёте сумму в диапазон от 0 до 1? Будет ли результат меняться в зависимости от относительной фазы этих двух волн?

6. Две пилообразных волны равной частоты и амплитуды, отличающихся на половину цикла по фазе, просуммированы. Какова форма волны суммы, и чему равны амплитуда и частота?

7. Какой относительный уровень (в децибелах) третьей гармоники (в три раза выше основной частоты) у пилообразной волны по сравнению с основной частотой?

8. Предположим, что вы синтезируете пилообразную волну в 44000 герц при частоте семплирования 44100 герц. Какова получающаяся форма волны?

9. Используя методы §3.7, начертите блок-схему для генерации двух фазово-сцепленных синусоид в 500 и 700 герц.

10. Нарисуйте блок-схему, показывающую как использовать пороги для детекции превышения одним аудиосигналом другого по величине. (Вам может это понадобиться для детекции и фильтрации обратной связи от говорящих в микрофоны.)

Глава 4

Автоматизация и управление многоголосием

Часто желательно управлять музыкальными объектами или событиями в комплексе, а не по отдельности. Такие комплексы могут иметь форму ряда событий во времени, в котором детали событий образуют общую “арку” (например, как ноты в мелодии). Или же отдельные части могут звучать одновременно, как голоса в аккорде или гармоники в сложном тоне. Часто некоторые или все свойства отдельных элементов лучше выводятся из свойств целого.

В репертуаре электронной музыки возникла богатая коллекция инструментов и идей для описания индивидуального поведения из совокупного. В этой главе мы рассмотрим два основных класса таких инструментов: генераторы огибающих и голосовые банки. Генератор огибающей автоматизирует поведение с течением времени, а голосовой банк — совокупность одновременных процессов (например, генераторы сигналов).

4.1 Генераторы огибающей

Генератор огибающей (иногда, и более правильно, называемый *генератором переходных процессов*) создает аудиосигнал, который гладко растёт и падает, как будто контролируя громкость музыкальной ноты. Генераторы огибающей были затронуты ранее в §1.5. Управление амплитудой путём умножения (рис. 1.4) является наиболее прямым, обычным способом использования, но существуют и многие другие возможные способы.

На протяжении многих лет появилось много форм генераторов огибающей, но самым простым и неизменным фаворитом является генератор огибающей ADSR. “ADSR” является сокращением от “Attack, Decay, Sustain, Release” (атака, спад, поддержка, завершение) — четыре отрезка выхода генератора ADSR. Генератор ADSR включается и выключается управляющим потоком, называемым “триггером”. При включении генератора ADSR запускаются сегменты атаки, затухания и

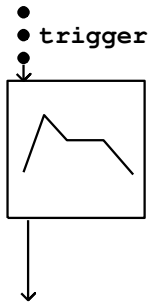


Рис. 4.1: Огибающая ADSR в виде блок-схемы, показывающей триггерный вход (управляющий поток) и аудиовыход.

поддержки. При выключении начинается сегмент отрезок завершения. Рисунок 4.1 показывает представление генератора огибающей ADSR в виде блок-схемы.

Имеется пять параметров, управляющих генератором ADSR. Во-первых, параметр уровень задаёт значение выхода в конце отрезка атаки (обыкновенно, наибольшее значение, выдаваемое генератором ADSR). Второй и третий параметры — атака и спад — дают временную продолжительность отрезков атаки и спада. Четвёртый параметр поддержки даёт уровень отрезка поддержки по отношению к уровню конца атаки. Наконец, параметр завершения даёт продолжительность отрезка завершения. Эти пять величин вместе с временными триггерными включениями и выключениями полностью определяют выход генератора ADSR. Например, продолжительность части поддержки равна времени между триггерным включением и выключением, минус продолжительность отрезков атаки и спада.

Рисунок 4.2 графически представляет некоторые возможные выходы генератора огибающей ADSR. В части (a) мы предполагаем, что триггерное включение и отключение достаточно далеко отделены, так что отрезок поддержки достигается перед тем, как получен сигнал отключения. Части (b) и (c) рис. 4.2 показывают результат следования сигнала выключения сразу за сигналом на включение: (b) в течение отрезка спада и (c) даже раньше, во время атаки. Генератор ADSR реагирует на эти ситуации отменой остающихся отрезков атаки и спада и продолжением прямо с отрезка завершения. Также генератор ADSR может быть включен командой триггера перед тем, как отрезок завершения закончен или даже во время атаки, спада или поддержки. Часть (d) рисунка показывает новый выход на атаку во время отрезка поддержки, а часть (e) во время отрезка спада.

Классическое приложение огибающей ADSR состоит в использовании клавиатуры или секвенсера, управляющего напряжением, для извлечения музыкальных нот на синтезаторе. Нажатие и отпускание клавиши (например) производило бы сигналы включения и отключения триггера. Генератор ADSR мог, далее, управлять амплитудой синтезатора, так что “ноты” начинались бы и останавливались клавишами. В дополнение к амплитуде генератор ADSR может (и часто) использоваться для управления тембром, который может далее быть использован для естественного развития течения каждой ноты.

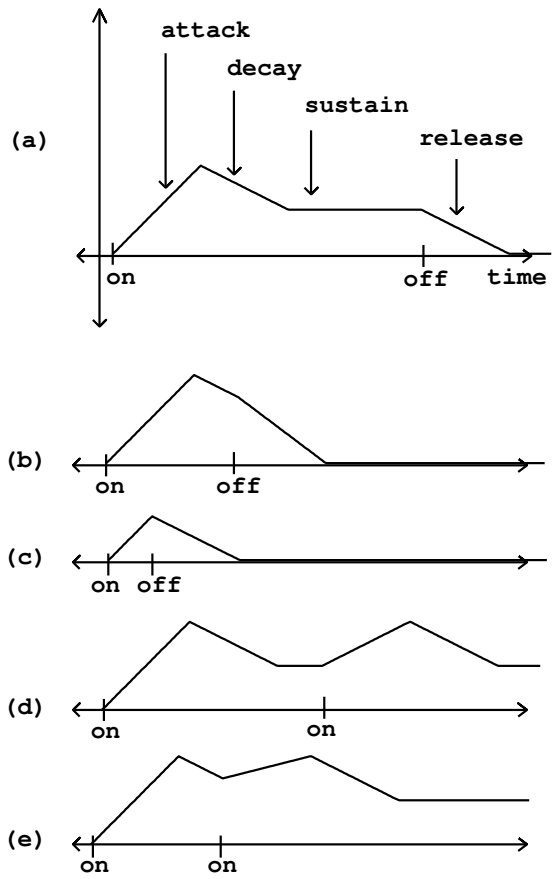


Рис. 4.2: Выход огибающей ADSR: *a* с разделёнными сигналами включения (on) выключения (off) триггера; *b*, *c* с ранним сигналом выключения триггера; *d*, *e* с повторной атакой.

4.2 Линейные и криволинейные амплитудные формы

Предположим, вы хотите постепенно увеличить уровень сигнала в течение десяти секунд, то есть вы хотите умножить его на сигнал, управляющий амплитудой $y[n]$, который растёт от 0 до 1 по величине за $10R$ семплов, где R есть частота семплирования. Самым очевидным выбором был бы линейный рост: $y[n] = n/(10R)$. Но это не приведёт к гладкому возрастанию воспринимаемой громкости. В первую секунду $y[n]$ растёт от $-\infty$ дб до -20 дб, в течение следующих четырёх — на 14 дб, а за последние пять — только на оставшиеся 6 дб. Большую часть десятисекундного периода увеличение в амплитуде будет мало заметно.

Существует другая возможность — экспоненциальное увеличение $y[n]$, так что амплитуда меняется с постоянной скоростью в децибелах. Для этого нужно зафиксировать начальную амплитуду как неслышимую, скажем, 0 дб (если мы принимаем единицу равной 100 дб). Теперь мы имеем противоположную проблему: за первые пять секунд управление амплитудой возрастёт от 0 дб (неслышимая) до 50 дб (пианиссимо); при постепенном возрастании эта часть должна лишь занять первую секунду или около того.

Более естественный переход возможно состоял бы в рассмотрении постепенного увеличения как динамической последовательности **0–*ppp*–*pp*–*p*–*mp*–*mf*–*f*–*ff*–*fff***, упорядоченной во времени, с каждым циклом, занимающим, примерно, одну секунду.

В идеале постепенное увеличение должно подчиняться некоторой шкале, лежащей между логарифмической и линейной. В некотором смысле произвольный, но полезный на практике выбор состоит в использовании кривой четвёртой степени:

$$y[n] = \left(\frac{n}{N}\right)^4,$$

где N есть число семплов за которые постепенное увеличение должно произойти (в примере выше, это 10R). Так, после первой секунды из десяти мы бы дошли до -80 дб, после пяти до -24 дб и после девяти до -4 дб.

Рисунок 4.3 показывает три функции, преобразующие амплитуду:

$$f_1(x) = x \quad \text{(линейная)},$$

$$f_2(x) = 10^{2(x-1)} \quad \text{(dB в линейную)},$$

$$f_3(x) = x^4 \quad \text{(четвертой степени)}.$$

Вторая функция преобразует децибелы в линейные амплитуды, упорядоченные так, что входной диапазон, от 0 до 1, соответствует 40 дб. (Этот входной диапазон в 40 дб отвечает разумному динамическому диапазону по 5 дб на каждый из 8 шагов.) Кривая четвёртой степени (quatric) очень хорошо приближает экспоненциальную кривую (децибельную) для больших амплитуд, но спадает быстрее для малых

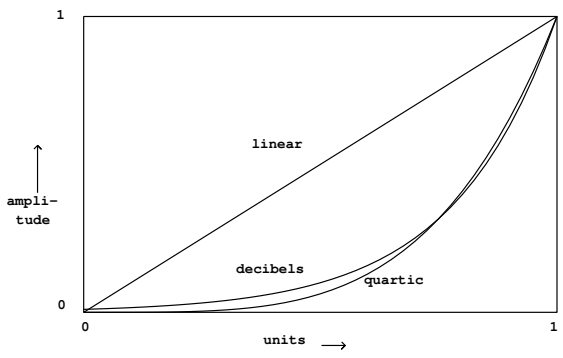


Рис. 4.3: Три функции преобразования амплитуды. Горизонтальная ось в линейных, логарифмических или биквадратных единицах, в зависимости от кривой.

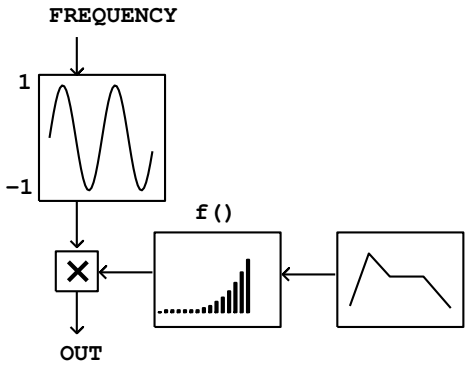


Рис. 4.4: Использование преобразующей функции для изменения формы амплитудных кривых.

амплитуд, достигая реальный ноль слева (в то время как экспоненциальная кривая лишь спадает до 1/100).

Мы можем рассматривать эти три кривых как функции преобразования из абстрактного управления (имеющего диапазон от 0 до 1) в линейную амплитуду. После того, как мы выбрали подходящую переходную функцию f, мы можем вычислить соответствующий управляющий амплитудой сигнал; если мы хотим подняться за N семплов от тишины до единичного усиления, управляющий сигнал должен быть:

$$y[n] = f(n/N).$$

Блок-схема этого преобразования показана на рис. 4.4. Здесь мы вводим новый тип блоков для представления действия преобразующей функции. Здесь мы не будем беспокоиться об осуществлении преобразования; в зависимости от требуемой функции, это лучше сделать арифметически или с использованием операции считывания таблицы.

4.3 Непрерывные и прерывистые изменения контрольного управления

Алгоритмы синтеза широко варьируются по их способности иметь дело с скачкообразно меняющимся управлением. До сих пор в этой главе мы предполагали, что управления должны осуществлять непрерывные изменения и генератор огибающей ADSR кажется идеально предназначенным для таких управлений. Может даже случиться так, что максимальная амплитуда новой ноты меньше текущей амплитуды её предшественницы (с использованием того же генератора) и огибающая ADSR будет просто спускаться вниз (вместо того, чтобы двигаться вверх) к новому значению для атаки.

Однако это не обязательно требуется в ситуациях, когда генератор огибающей отвечает за какой-то аспект тембра: возможно, например, мы не хотим, чтобы резкость ноты уменьшалась во время атаки, а скорее чтобы быстро перескочила на гораздо более низкое значение, чтобы всегда иметь возможность повышаться во время новой атаки.

Такая же ситуация может возникнуть и с огибающими высоты тона: может быть желательно делать глиссандо[1] от одной ноты к другой, или наоборот, чтобы траектория высоты тона каждой ноты начиналась заново в точке, независимой от предыдущего звука.

Когда мы хотим сделать прерывистые изменения параметров синтеза, возникают две ситуации: или всё происходит чисто (например, при скачкообразном изменении высоты ноты); или наоборот, как при изменении индекса волновой таблицы (которое приводит к прерывистому изменению в выходных данных). Существуют даже такие параметры, для которых *возможно* не существует непрерывного изменения; например, при переключении между набором волновых таблиц. В общем, скачкообразное изменение фазы осциллятора или амплитуды сигнала будет производить слышимый артефакт, но некоторые приращения фазы (такие как у высоты) могут совершать прыжки, не приводящие к плохим результатам.

В тех случаях, когда изменение параметра не может быть произведено непрерывно по той или иной причине, имеется, по крайней мере, две стратегии для чистого внесения изменений: *приглушение звука* и техника “включить и линейно снизить”.

4.3.1 Приглушение звука

Техника *приглушения* звука состоит в применении огибающей к выходной амплитуде, которая быстро линейно снижается до нуля перед тем, как параметр меняется и, далее, восстанавливается после этого. Может случиться или не случиться так,

[1]Этот прием иногда называется *портаменто* — способ исполнения, при котором следующая нота не сразу берётся точно (в звуко-высотном отношении), а используется плавный переход к нужной высоте от предыдущей ноты. (*прим. перев.*)

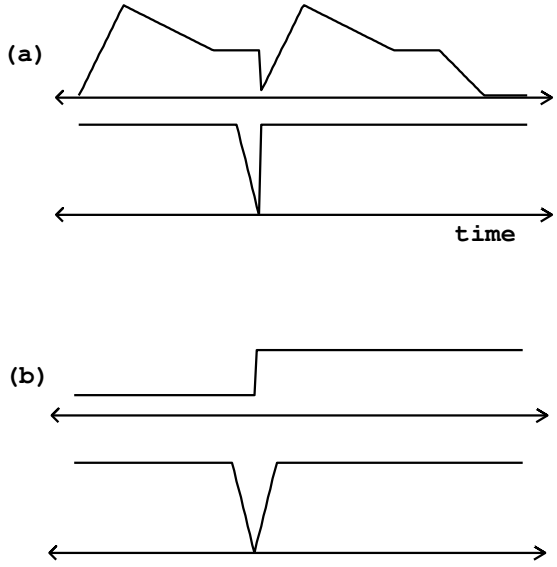


Рис. 4.5: Техника приглушения звука для избавления от скачкообразных изменений: (a) огибающая (график сверху) скачкообразно уходит в ноль и приглушающий сигнал (нижний график) снижается линейно вниз заранее для приготовления к изменению и, далее, восстанавливается (скачкообразно) к своему прежнему значению;
(b) огибающая скачкообразно изменяется от одного значения до другого; приглушающий сигнал должен линейно спадать перед этим, а также линейно возвращаться назад после этого.

что разрывные изменения приведут к тому, что сигнал будет расти гладко из нуля после этого. На рис. 4.5 (часть a) мы приводим пример амплитудной огибающей (выходной сигнал генератора ADSR) и предполагаем, что разрывное изменение происходит в начале новой ноты с нулевой амплитудой.

Для скачкообразного изменения выхода генератора ADSR мы *переустанавливаем* его. Это другая операция по отношению к отправке сигнала триггера; результат состоит в принуждении его к скачку на новое значение, после которого мы можем или просто оставить его там или переключить триггером на новое значение. рис. 4.5 (часть a) показывает эффект переустановки и отправки нового сигнала триггера на генератор ADSR.

Ниже выхода генератора ADSR мы видим подходящий приглушающий сигнал, который линейно преобразовывает сигнал в ноль для приготовления к разрыву. Промежуток времени, в который мы разрешаем приглушение, должен быть мал (так чтобы оборвать предыдущий звук настолько мало, насколько возможно), но не слишком мал, чтобы не вызвать слышимые артефакты на выходе. Рабочий пример этого типа приглушения был уже показан на стр. 80; там мы разрешили 5 миллисекунд для линейного спада. Приглушающий сигнал умножается на выход процесса, который должен быть переключён.

Рисунок 4.5 (часть b) показывает ситуацию, в которой мы предполагаем, что

скачкообразное изменение осуществляется между двумя ненулевыми величинами. В этом случае сигнал глушения должен не только снижаться, как раньше (до разрыва), но и снова увеличиваться после него. Время спада не обязано быть равным времени нарастания; они должны выбираться, как и всегда, прослушиванием звукового результата.

Как правило, сложность техники приглушения заключается в том, что необходимо запустить операцию приглушения до внесения желаемого изменения в элемент управления. В режиме реального времени это часто требует, чтобы мы намеренно вносили задержку в изменение управления. Это еще одна причина, по которой время отключения звука должно быть как можно ниже. (Более того, не стоит пытаться свести задержку к минимуму, условно опуская период постепенного снижения, когда он не нужен; постоянная задержка намного лучше, чем переменная, даже если она и меньше в среднем.)

4.3.2 Техника “включить и линейно снизить”

Метод *“включить и линейно снизить”* также предназначен для избавления от скачков, происходящих от резких изменений управления, но делает это другим способом: синтезируя противоположный скачок, который мы добавляем для того, чтобы убрать первоначальный. На рис 4.6 показан пример, в котором синтетический перкуссионный звук (с огибающей синусоидой) начинает новую ноту в середине предыдущей. Атака звука происходит не от амплитудной огибающей, а от начальной фазы синусоиды, что часто хорошо подходит для перкуссионных звуков. График внизу на рисунке показывает компенсирующий аудиосигнал с противоположным скачком, который может быть добавлен к верхнему для избавления от разрыва. Достоинство этого метода по сравнению с приглушением звука состоит в том, что, во-первых, не нужна задержка между решением произвести атаку и звуком атаки; и, во-вторых, любые артефакты, возникающие при использовании этого метода, вероятнее всего будут маскированы новой звуковой атакой.

Рисунок 4.7 показывает, как метод “включить и линейно снизить” может быть представлен в виде блок-схемы. Отмеченный многоточием блок может содержать любой алгоритм синтеза, который мы хотим прервать скачкообразно, так чтобы он опять начинался с нуля (как, например, в части (a) предыдущего рисунка). В то же самое время, когда мы устанавливаем триггером, какие изменения управления необходимы (как проиллюстрировано сверху на основном генераторе ADSR), мы также переустанавливаем и триггер другого компенсирующего генератора ADSR (правее, в середине) для устранения скачка. Скачок равен в точности противоположному выходному значению синтезатора перед тем, как он устанавливается в ноль.

Для того чтобы сделать это, мы измеряем уровень, на который основной генератор ADSR должен теперь перескочить. Это его собственный текущий уровень (который может и не быть нулём) минус скачек (или равносильно *плюс* последнее значение выхода синтезатора). Два эти сигнала складываются (в блоке $\boxed{+}$ ниже)

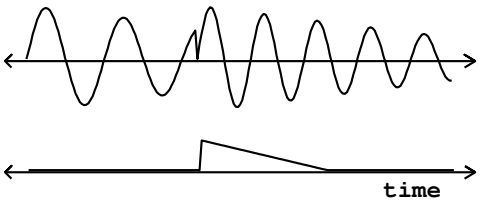


Рис. 4.6: Техника “включить и линейно снизить” для убирания разрывных изменений. Разрыв (рисунок сверху) измеряется и убирается сигналом, имеющим противоположный разрыв (график снизу), который далее плавно убывает.

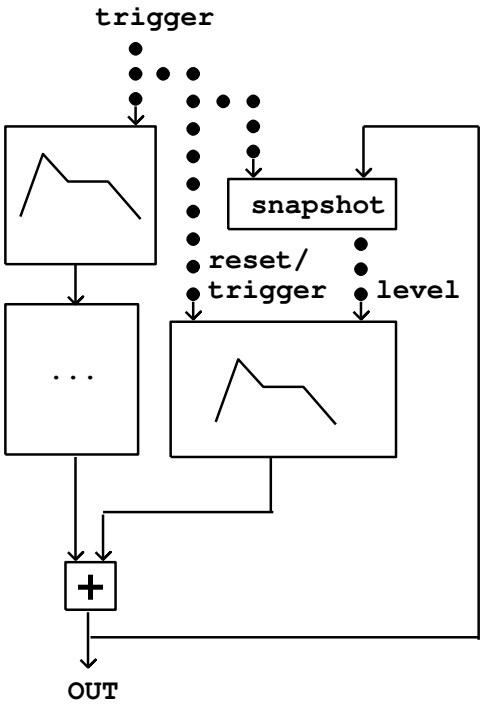


Рис. 4.7: Блок-схема для метода “включить и линейно снизить”.

и, далее, делается снимок мгновенного значения (snapshot). Компенсирующий генератор огибающей (справа) переустанавливается на это новое значение и, далее, управляется триггером для линейного снижения до нуля. Выход блока [+] (сумма выхода синтезатора и сигнала с убранным скачком) есть “переключённый” сигнал.

4.4 Полифония

В музыке термин *полифония* обыкновенно используется в значении “несколько отдельных голосов, поющихся или играющихся на различных высотах по отношению друг к другу”. Когда мы говорим об электронных музыкальных инструментах, мы используем этот термин в значении “поддержание нескольких копий некоторого процесса одновременно”. Мы обыкновенно называем каждую копию “голосом” для поддержания аналогии, хотя голоса не обязаны производить отдельно различимые звуки.

На этом языке фортепиано является полифоническим инструментом с 88 “голосами”. Каждый голос фортепиано обыкновенно способен воспроизводить в точности одну высоту. При этом не возникает вопроса, какой голос использовать для воспроизведения ноты данной высоты, а также не ставится задача воспроизведения нескольких нот одновременно на одной и той же высоте.

Многие полифонические электронные музыкальные инструменты дают более гибкий подход к управлению голосом. Многочисленные программы синтеза (такие как Csound) используют схему динамического назначения голоса, так что в результате новый звук создаётся для каждой ноты в партитуре. В системах Max/MSP или Pd, которые ориентированы на интерактивное использование в режиме реального времени, *голосовой банк* назначается заранее, и необходимая работа (проигрывание нот или что-либо другое) распределяется между голосами банка.

Это изображено в виде схемы на рис. 4.8, где каждый голос производит один выходной сигнал, а далее все сигналы складываются для получения общего выхода голосового банка. Часто индивидуальные голоса нуждаются в нескольких отдельных выходах: например, они могут иметь несколько выходных каналов, так что они могут быть панорамированы индивидуально; или они могут иметь индивидуальные уровни посылов на эффекты.

4.5 Распределение голосов

Часто требуется автоматизировать выбор голосов для ассоциирования с индивидуальными *задачами* (такими как ноты для воспроизведения). Например, музыкант, играющий на клавиатуре, на практике не может выбрать, какой голос должен идти с каждой играемой нотой. Для автоматизации выбора голоса нам нужен алгоритм распределения голосов, показанный на рис. 4.9.

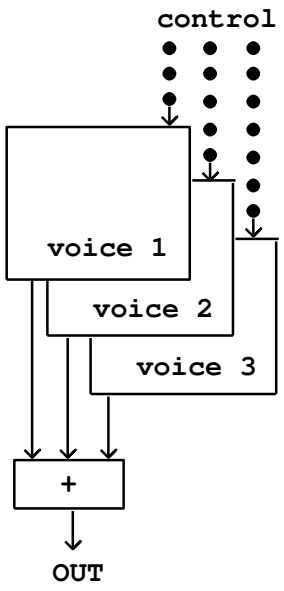


Рис. 4.8: Голосовой банк для полифонического синтеза.

Вооружившись подходящим алгоритмом распределения голосов, управляющему источнику не требуется знать детали о том, какой конкретный голос выполняет ту или иную задачу; на это часто рассчитывают алгоритмические нотные генераторы и секвенсеры. С другой стороны, музыкальная запись для ансамблей часто определяет точно, какой инструмент играет какую ноту, так что ноты будут связаны одна с другой требуемым образом.

Как работает один простой алгоритм распределения голосов показано на рис. 4.10. Здесь мы предполагаем, что голосовой банк имеет всего два голоса, и мы пытаемся назначить голоса для задач **a**, **b**, **c** и **d**. Всё идет гладко, до тех пор пока не приходит задача **d**, и мы не видим свободных голосов (они заняты задачами **b** и **c**). Мы могли бы теперь или снять задачу **d**, или забрать голос у **b** или **c**. На практике лучший выбор состоит в том, чтобы забрать голос. В этом конкретном примере мы принимаем решение забирать голос у наиболее старой задачи **b**.

Если у нас есть возможность узнать продолжительность задач **b** и **c** в момент начала задачи **d**, у нас будет возможность сделать выбор, какой голос забрать лучше. В данном случае возможно лучше забрать голос у **c**, чтобы в конце воспроизводились вместе **d** и **b**, а не только **d**. В некоторых ситуациях эта информация будет доступна в момент, когда выбор должен быть сделан, а, в некоторых (ввод с клавиатуры при живом исполнении, например) будет недоступна.

4.6 Метки голосов

Теперь предположим, что мы используем голосовой банк для воспроизведения нот, как в примере выше, но предположим, что ноты **a**, **b**, **c** и **d** имеют одинаковую

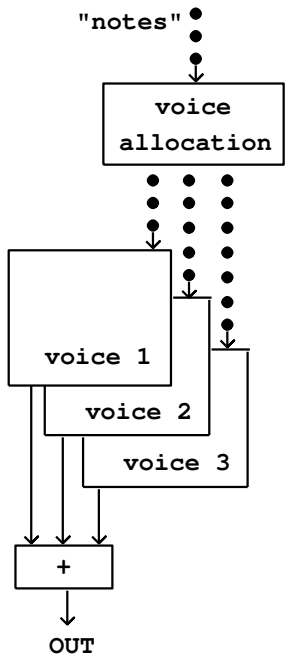


Рис. 4.9: Полифоническое назначение голосов.

высоту и, более того, что все остальные их параметры идентичны. Как мы можем построить управляющий поток так, что когда любая из этих нот выключается, мы бы знали, какая именно?

Этот вопрос не возникает, если источником управления являются фортепианные клавиши, поскольку невозможно воспроизводить больше, чем одну ноту одновременно от одной клавиши. Но он может легко возникнуть алгоритмически или просто как результат слияния двух клавиатурных потоков вместе. Более того, выключение нот является лишь простейшим примером более общей проблемы: как, выделив задачу в голосовом банке, мы можем управлять её изменениями как функцией от входного сигнала реального времени или другого непредсказуемого

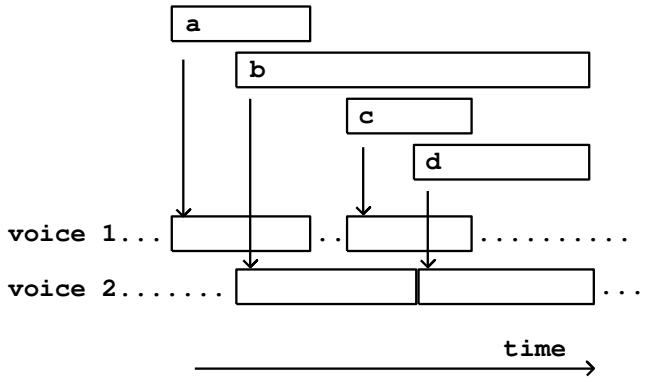


Рис. 4.10: Алгоритм полифонического распределения голосов, показывающий захват голосов.

фактора.

Для того чтобы иметь дело с такими ситуациями, мы можем добавить одну или несколько *меток* к сообщению, запускающему ноту (или, в общем случае, задачу). Метка — это любой набор данных, с которыми может быть позднее идентифицирована задача, так что мы можем в дальнейшем использовать её для поиска назначенного голоса.

Вновь рассматривая пример на рис. 4.10, приведем способ, которым мы могли бы записать эти четыре задачи в виде управляющего потока:

начальное время	конечное время	высота	...
1	3	60	...
2	8	62	...
4	6	64	...
5	8	65	...

В этом представлении нам не нужны метки, поскольку каждое сообщение (каждая строка таблицы) содержит всю необходимую информацию, чтобы определить задачу в целом. (Здесь мы предположили, что задачи **a**, ..., **d** являются в действительности музыкальными нотами с высотами 60, 62, 64 и 65.) В результате мы представляем каждую задачу, как одно событие в управляющем потоке (§3.3).

С другой стороны, если предположить, что мы не знаем заранее длину каждой ноты, то лучше представить управляющий поток следующим образом:

время	метка	действие	параметры
1	a	запуск	60 ...
2	b	запуск	62 ...
3	a	окончание	
4	c	запуск	64 ...
5	d	запуск	65 ...
6	c	окончание	
8	b	окончание	
8	d	окончание	

Здесь каждая нота разделена на два отдельных события для её запуска и окончания. Индексы **a**, ..., **d** используются как метки; мы знаем, какой запуск идёт с каким окончанием, поскольку они имеют одинаковые метки. В целом, заметим, что метка необязательно связана с голосом, который будет использован для воспроизведения каждой ноты.

Стандарт MIDI не поддерживает метки; при нормальном использовании, высота ноты служит также как её метка (так что метки постоянно переиспользуются). Если две ноты, имеющие одинаковую высоту должны быть запрошены отдельно (для сдвига их высот в различных направлениях, например), её MIDI–канал может быть использован в дополнение к ноте в качестве метки.

В музыкальном программном обеспечении, работающем в режиме реального времени, часто необходимо переходить в обе стороны между представлениями “одно событие на задачу” и использованием меток, описанном выше, так как первое представление больше подходит для хранения и графического редактирования, в то время как второе часто лучше для операций в режиме реального времени.

4.7 Инкапсуляция в Pd

Примеры для этой главы будут использовать механизм *инкапсуляции* в Pd, который позволяет создавать патчи, которые могут быть повторно использованы любое число раз. Объектные боксы в патче Pd могут быть *сабпатчами*, которые представляют из себя отдельные патчи, размещённые внутри окон. Они бывают двух типов: *одноразовые сабпатчи* и *абстракции*. В любом случае сабпатч появляется как объектный бокс в другом патче, называемом *родительским*.

Если вы наберёте “pd” или “pd my-name” в боксе при создании объекта, это создаст одноразовый сабпатч. Содержание сабпатча сохраняется как часть родительского патча в том же самом файле. Если вы делаете несколько копий сабпатча, вы можете изменять их индивидуально. С другой стороны, вы можете создать абстракцию набирая внутри бокса имя патча Pd, сохранённого в виде файла (без расширения “.pd”). В этой ситуации Pd будет читать этот файл в сабпатче, а изменения файла происходят везде, где задействована абстракция.

Сабпатч (одноразовый или абстракция) может иметь входы и выходы, которые появляются в окне родительского патча. Это определяется с использованием следующих объектов:

> inlet, inlet~: создание входов для объектного окна, содержащего сабпатч. Версия inlet~ создаёт вход для аудиосигналов, в то время как inlet — для управляющих потоков. В любом случае, всё что приходит на вход окна в родительском патче, приходит из объекта inlet или inlet~ в сабпатче.
>
> outlet, outlet~: Соответствующие объекты для выхода из сабпатчей.

Pd обеспечивает механизм сквозного прохождения аргумента так, что вы можете параметризовать различные задействования абстракции. Если в объекте вы вводите “$1”, это раскрывается до значения “первый начальный аргумент в родительском патче”, сходным образом для “$2” и так далее. Текст в объекте интерпретируется в момент создания объекта, в отличие от текста в боксе сообщений. В боксах сообщений этот же “$1” означает “первый аргумент сообщения, которое только что получил” и интерпретируется всегда, когда приходит новое сообщение.

Пример абстракции, использующей входы, выходы и параметризацию, показан на рис. 4.11. В части (a) патч задействует plusminus в объектном окне с начальным аргументом, равным 5. Число 8 направляется на вход объекта plusminus, а на выход приходят сумма и разность 8 и 5.

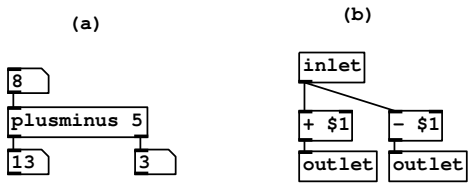


Рис. 4.11: Механизм абстракции в Pd: (*a*) задействование абстракции, plusminus с 5 как начальным аргументом; (*b*) содержание файла “plusminus.pd”.

Объект plusminus не определяется Pd, а определён патчем, помещаемым в файле с названием “plusminus.pd”. Этот патч показывается в части (*b*) рисунка. Один объект inlet и два объекта outlet соответствуют входам и выходам объекта plusminus. Два аргумента “$1” (в объектах + и -) заменяются на 5 (начальный аргумент объекта plusminus).

Мы уже видели одну абстракцию в примерах: объект output~, введённый на рис. 1.10 (стр. 16). Этот пример также показывает, что абстракция может отражать управления как часть своего окна в родительском патче; смотрите документацию Pd, в которой описывается это свойство.

4.8 Примеры

4.8.1 Генератор огибающей ADSR

Пример D01.envelope.gen.pd (рис. 4.12) показывает как объект line~ может быть использован для генерации огибающей ADSR, управляющей синтезирующим патчем (только огибающая ADSR показана на этом рисунке). Нажатие кнопки “attack” (“атака”) имеет два эффекта. Первый (самый левый на рисунке) — это установка объекта line~ на отрезок атаки с целевым значением 10 (пиковая амплитуда) за 200 миллисекунд (время атаки). Второй устанавливает объект del 200, так что, когда отрезок атаки завершён, может начаться отрезок спада[2]. На отрезке спада значение падает до 1 (уровень поддержки[3]) за другие 2500 миллисекунд (время спада).

Кнопка “release” (“завершение”) меняет целевое значение того же самого объекта назад на ноль за 500 или более миллисекунд (время завершения). Также, в случае, если объект del 200 оказался установленным в тот момент, когда кнопка “release” нажата, на него посылается сообщение “stop”. Это препятствует началу отрезка спада генератора ADSR после отрезка завершения.

[2] *decay* (*прим. перев.*)
[3] *sustain* (*прим. перев.*)

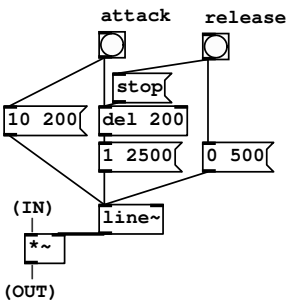


Рис. 4.12: Использование объекта line~ для генерации огибающей ADSR.

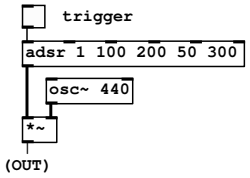


Рис. 4.13: Задействование абстракции adsr

В примере D02.adsr.pd (рис. 4.13) мы инкапсулируем генератор ADSR в виде абстракции Pd (с именем adsr), так что его просто использовать. Дизайн абстракции adsr делает возможным управление пятью параметрами ADSR или установкой начальных аргументов, или присоединением управляющих потоков к его входам.

В этом примере пять начальных аргументов (1, 100, 200, 50 и 300) определяют уровень пика, время атаки, время спада, время поддержки (в процентах пикового уровня) и время завершения. Имеется шесть управляющих входов: первый для триггерного управления генератором ADSR и другие для обновления значений пяти параметров. Выходом абстракции является аудиосигнал.

Эта абстракция реализована как показано на рис. 4.14. (Вы можете открыть этот сабпатч, кликнув по объекту adsr в патче.) Единственные сигнальные объекты это line~ и outlet~. Три объекта pack соответствуют трём объектам сообщений рис. 4.12, рассмотренного ранее. Слева направо они отвечают за отрезки атаки, спада и завершения.

Отрезок атаки идёт к целевому значению, определяемому “$1” (первый начальный аргумент абстракции) за “$2” миллисекунд; эти величины могут быть переписаны отправкой чисел на входы “peak level” и “attack”. Отрезок завершения сходен, но проще, поскольку целевое значение всегда ноль. Трудную часть представляет из себя отрезок спада, который опять должен быть вызван после задержки, равной времени атаки (объект del $2). Уровень поддержки вычисляется из пикового уровня и значения поддержки в процентах (их перемножением и делением на 100).

Триггерный вход, если послано число, отличное от нуля, задаёт включение с начала (отрезки атаки и спада) и, если послан ноль, включает отрезок заверше-

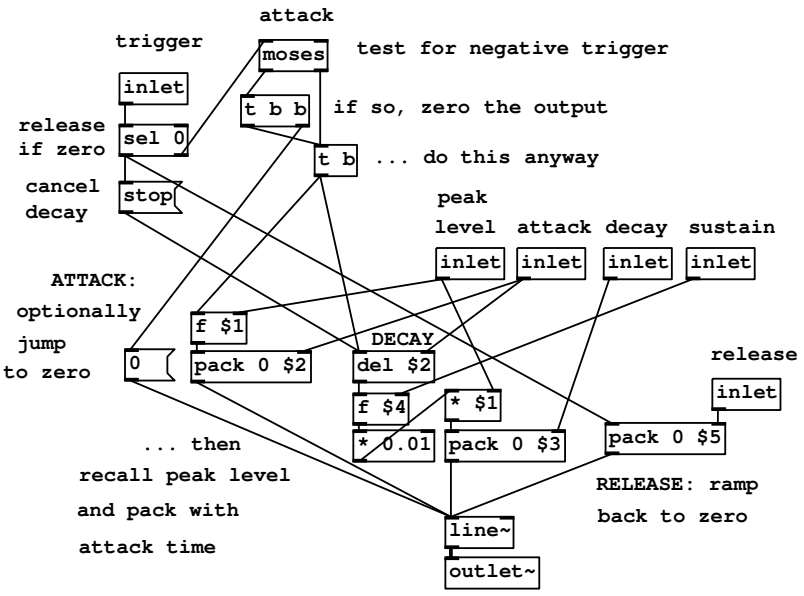


Рис. 4.14: Внутреннее содержание абстракции adsr

ния. Более того, генератор ADSR может быть переустановлен на ноль отправкой отрицательного триггерного сигнала (который также производит включение с начала).

4.8.2 Функции преобразования для управления амплитудой

В §4.2 описывалось использование огибающих ADSR для управления амплитудой, при этом экспоненциальные или биквадратные отрезки[4] часто звучат естественнее линейных. Патчи D03.envelope.dB.pd и D04.envelope.quartic.pd (последний показан на рис. 4.15) демонстрируют использование отрезков в децибелах и биквадратных функциях. В дополнение к амплитуде в примере D04.envelope.quartic.pd частота звука также управляется с использованием линейной и (для сравнения) биквадратной форм.

Поскольку конвертация децибелов в единицы линейной амплитуды — дорогостоящая операция (по крайней мере, по сравнению с осциллятором или генератором линейной огибающей), пример D03.envelope.dB.pd использует считывание таблицы для осуществления необходимой функции преобразования. Этот способ эффективнее, но недостаток в том, что мы заранее должны принять решение по поводу диапазона допустимых значений (здесь, от 0 до 120 дб).

Для отрезка функции четвёртой степени, как в примере D04.envelope.quatric.pd, считывание волновой таблицы не требуется; мы просто дважды последовательно возводим в квадрат выходной сигнал объекта line~. Для компенсации возведения в четвёртую степень, целевые значения, посылаемые на объект line~, должны

[4]четвертой степени (*прим. перев.*)

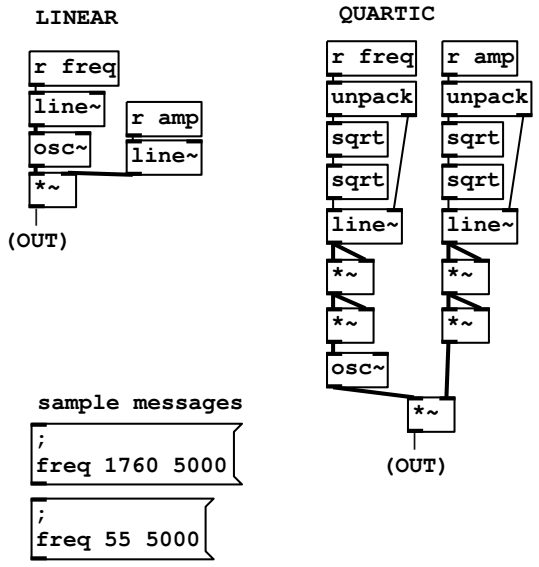


Рис. 4.15: Функции линейного и биквадратного (четвертой степени) преобразований для изменения амплитуды и высоты.

представлять из себя корни четвёртой степени из требуемых. Так, сообщения о линейном изменении частоты или амплитуды вначале распаковываются на целевое значение и временной интервал, берётся корень четвёртой степени из целевого значения (двумя последовательными функциями квадратного корня), и далее оба отправляются далее на объект line~. Здесь мы использовали новый объект Pd:

> unpack: распаковывание списка чисел (и/или символов) и распределение их по различным выходам. Обыкновенно, выходы появляются в порядке справа налево. Число выходов и их тип определяется начальными аргументами. (См. также pack, стр. 49.)

Следующие два патча, D05.envelope.pitch.pd и D06.envelope.portamento.pd, используют генератор огибающей ADSR для создания огибающей высоты тона и простого объекта line~, также контролирующего высоту для создания эффекта портаменто. В обоих случаях требуются экспоненциальные отрезки, которые вычисляются с использованием считывания волновой таблицы.

4.8.3 Аддитивный синтез: колокол Риссе

Механизм абстракций в Pd, которым мы воспользовались выше для создания многократно используемого генератора ADSR, также полезен для создания банков голосов. Здесь мы будем использовать абстракции для организации банков осцилляторов для аддитивного синтеза. Существует множество способов создания банков осцилляторов, кроме демонстрируемых здесь.

Простейшая и наиболее прямая организация синусоид состоит в формировании

ноты как суммы гармоник. Результат является монофоническим, в том смысле, что патч будет воспроизводить только одну ноту в единицу времени, которая, тем не менее, будет состоять из нескольких синусоид, чьи индивидуальные частоты и амплитуды могут зависеть и от ноты, которую мы воспроизводим, и также от их индивидуального места в гармоническом (или ангармоническом) обертонном ряду.

Пример D07.additive.pd (рис. 4.16) использует банк из 11 копий абстракции, называемой partial (рис. 4.17), которая имитирует хорошо известный колокольный инструмент Жана Клода Риссе. Как описано в [DJ85, стр. 94], звук колокола имеет 11 гармоник, каждая со своей собственной относительной амплитудой, частотой и длительностью.

Для каждой ноты абстракция partial вычисляет простую (с использованием функции четвёртой степени) амплитудную огибающую, состоящую только из отрезков атаки и спада; отрезки удержания и завершения отсутствуют. Огибающая умножается на синусоиду, и их произведение добавляется в суммирующую шину. Для реализации суммирующей шины введены два новых класса объектов:

catch~ : чтение сигнала из суммирующую шины. Начальный аргумент ("sumbus" в этом примере) определяет имя суммирующей шины, так что в дальнейшем объекты throw~ могут ссылаться на неё. Вы можете иметь неограниченное число суммирующих шин (и, следовательно, объектов catch~), но при этом у всех должны быть уникальные имена.

throw~ : добавить к суммирующей шине. Начальный аргумент определяет выбор суммирующей шины.

Управляющий интерфейс незамысловат: числовые окна управляют "основной" частотой колокола и продолжительностью. Отправка сообщения "bang" объекту s trigger начинает воспроизведение ноты. (После чего нота спадает за период времени, управляемый параметром длительности; отдельного триггера для остановки ноты нет). Управление амплитудой возможно только через объект output~ .

Четыре начальных аргумента абстракции partial определяют следующие параметры:

1. *Амплитуда*. Амплитуда гармоники на её пике, в конце атаки и в начале спада ноты.

2. *Относительная длительность*. Умножается на полную продолжительность ноты (управляемую в основном патче) для определения длительности части спада синусоиды. Индивидуальные гармоники могут, таким образом, иметь различные времена спада, так что некоторые гармоники затухают быстрее других при общем управлении основного патча.

3. *Относительная частота*. Как и с относительной длительностью, она управляет частотой каждой гармоники, которая является произведением общей частоты, управляемой основным патчем.

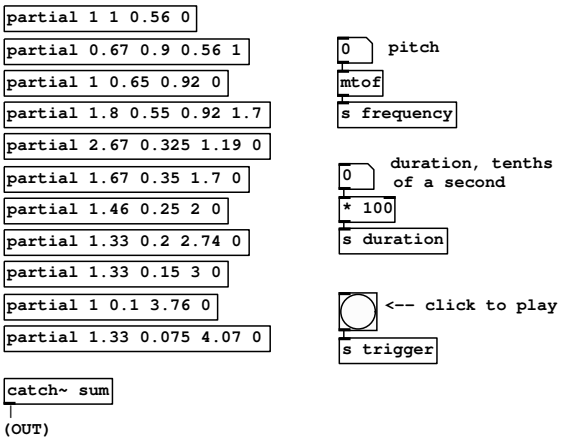


Рис. 4.16: Реализация Pd колокольного инструмента Жана-Клода Риссе. Звук колокола создан 11 синусоидами, каждая из которых является копией абстракции partial.

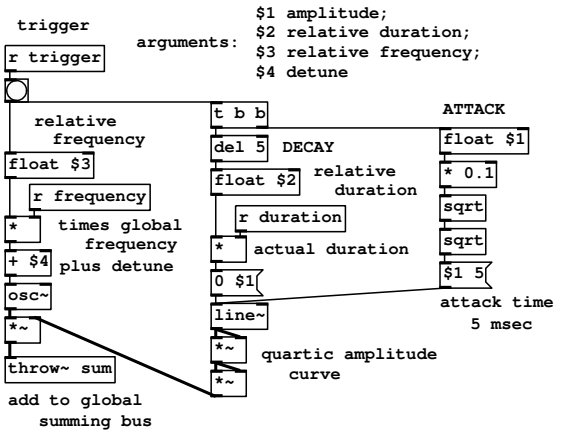


Рис. 4.17: Абстракция partial, использованная колокольным инструментом Риссе с рис. 4.16.

4. *Расстройка*. Частота в герцах, добавляемая к произведению глобальной частоты и относительной частоты.

Внутри абстракции partial амплитуда просто берётся напрямую от аргумента “$1” (с умножением на 0.1 для коррекции отдельных высоких амплитуд); длительность вычисляется объектом r duration, умножением его на аргумент “$2”. Частота вычисляется как $fp + d$, где f есть глобальная частота (от объекта r frequency), p — относительная частота гармоники и d — частота расстройки.

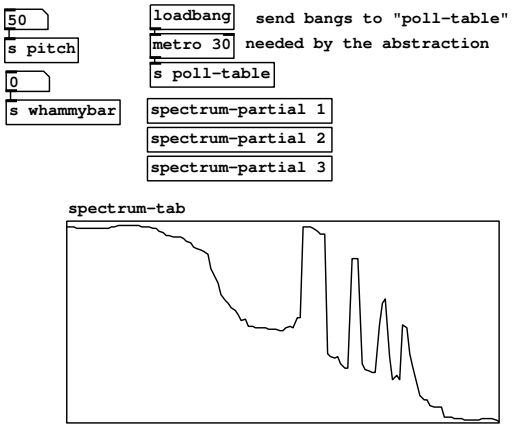


Рис. 4.18: Аддитивный синтез для определённой спектральной огибающей, извлечённой из таблицы.

4.8.4 Аддитивный синтез: управление спектральной огибающей

Следующий пример патча D08.table.spectrum.pd (рис. 4.18) показывает совершенно другое, чем в предыдущем параграфе, приложение аддитивного синтеза. Здесь синусоиды управляются абстракцией spectrum-partial, показанной на рис. 4.19. Каждая гармоника вычисляет собственную частоту, как в предыдущем патче. Также каждая гармоника вычисляет периодически её собственную амплитуду (при включении триггером объекта poll-table) с использованием объекта tabread4. Содержание таблицы (которая имеет номинальный диапазон 50 дб) преобразуется в линейные единицы и используется в качестве управления амплитудой обычным образом.

Похожий пример B09.shepard.tone.pd (не показан) воспроизводит тон Шепарда с использованием той же техники. В нем частоты синусоид циклически проходят через фиксированный диапазон, в самом конце снова перескакивая на начало. Спектральная огибающая имеет пик в середине диапазона высот и спадает до нуля на краях диапазона, так что мы слышим лишь непрерывное изменение высоты тона без скачков. Результатом является знаменитая слуховая загадка — бесконечно поднимающийся и опускающийся тон.

Метод синтеза с использованием спектральной огибающей может быть обобщён многими способами. Огибающая может изменяться во времени либо в результате анализа другого сигнала реального времени или вычисления, заданного множеством композиционных правил, либо попеременным выбором из коллекции заранее созданных спектральных огибающих, либо деформированием частотных огибающих — и это всего лишь несколько возможностей.

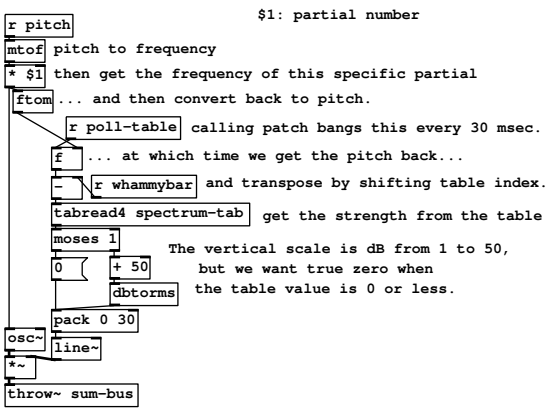


Рис. 4.19: Абстракция spectrum-partial, использованная на рис. 4.18.

4.8.5 Полифонический синтез: семплер

Теперь мы переходим к примеру использования динамического распределения голоса, описанного в §4.5. В примерах аддитивного синтеза, показанных ранее, каждый голос использовался для фиксированной цели. В настоящем примере мы выделяем голоса из банка, необходимые для воспроизведения нот в управляющем потоке.

Пример D11.sampler.poly.pd (рис. 4.20) показывает полифонический семплер, который использует абстракцию sampvoice (чьё содержание показано на рис. 4.21). Техника изменения высоты и других параметров в *семплере однократного воспроизведения* продемонстрирована в примере D01.sampler.notes.pd (не показан), который, в свою очередь, построен на основе первоначального *семплера однократного воспроизведения* предыдущей главы C05.sampler.oneshot.pd (рис. 3.14).

На рисунке 4.20 показан другой способ реализации суммирующей шины объектами sampvoice, в отличие от приведённого ранее: здесь каждый из них складывает свой собственный сигнал с сигналом на его входе и выводит всю сумму. Выходной сигнал самого нижнего объекта, таким образом, содержит сумму всех восьми предыдущих. Преимущество этого способа — явная форма по сравнению с использованием шин throw~ / catch~, его лучше использовать, когда визуальный беспорядок не является проблемой[5].

Основная работа патча, тем не менее, состоит в распределении “нотных” сообщений по объектам sampvoice. Для того чтобы проделать это, мы должны ввести некоторые новые объекты Pd:

> mod: отношение по модулю. Например, 17 *mod* 10 даёт 7 и −2 *mod* 10 даёт 8. Имеется также целочисленное деление div; деление 17 на 10 с помощью div даёт 1 и −2 на 10 даёт −1.

[5] “визуальный беспорядок” может возникнуть при большом количестве соединений (*прим. перев.*)

poly: полифоническое распределения голосов. Начальные аргументы определяют число голосов в банке и флаг ("1", если используется заимствование голосов, и "0", если нет). На первый вход отправляется числовая метка, на второй — флаг, определяющий, должен ли начаться или остановиться голос с данной меткой (ненулевые числа означают "начать", и ноль — "остановить"). Выходы слева направо: номер голоса, в центре — метка, а затем флаг начать/остановить. В приложениях MIDI, метка может быть высотой ноты[6], и флаг начать/остановить может быть силой нажатия ноты[7].

makenote: обеспечить отложенные сообщения "снятие ноты", соответствующие сообщениям "нажатие нота". Первый вход — метка, второй — флаг начать/остановить (номер ноты и сила нажатия при использовании MIDI) и третий — требуемая продолжительность в миллисекундах. Пара метка/флаг при получении сразу отправляется на выходы; далее, после задержки, метка повторяется с нулевым флагом, чтобы остановить ноту после требуемой длительности.

Сообщения "нота" содержат поля высоты, амплитуды, продолжительности, номера семпла, позиции в семпле, времени атаки и времени спада. Например, сообщение

60 90 1000 2 500 10 20

(если получено объектом r note) означает воспроизведение ноты с высотой 60 (в единицах MIDI), амплитудой 90 дб, длиной одну секунду из волновой таблицы "sample2" в точке 500 миллисекунд от начала таблицы, с временем атаки и спада в 10 и 20 миллисекунд соответственно.

После распаковки сообщения на семь частей, патч создаёт метку для ноты. Для этого первый объект t b f выводит "bang" после того, как последний из семи параметров появился в разделённом виде. Комбинация объектов +, f и mod действует как счётчик, который повторяется после миллиона шагов, по существу дела генерируя единственное число, соответствующее ноте.

Следующий шаг состоит в использовании объекта poly для определения, какой голос какую ноту воспроизводит. Объект poly ожидает отдельные сообщения для задач начала и остановки (то есть, нот). Так, метка и длительность отправляются сначала объекту makenote, который выводит флаг ("velocity") на правый выход и метку на левый. Для каждой метки, которую получает makenote, выводятся две пары чисел: одна — для запуска ноты, и другая, после задержки, равной времени продолжительности — для остановки.

После отправки объекту poly этих отдельных сообщений, нам нужно убрать сообщения на выходе, соответствующие концам нот, поскольку нам в действительности нужны только комбинации сообщений "нота" с полями длительности. Объект stripnote делает эту работу. Наконец, номер голоса, который мы вычислили,

[6] *pitch*
[7] *velocity*

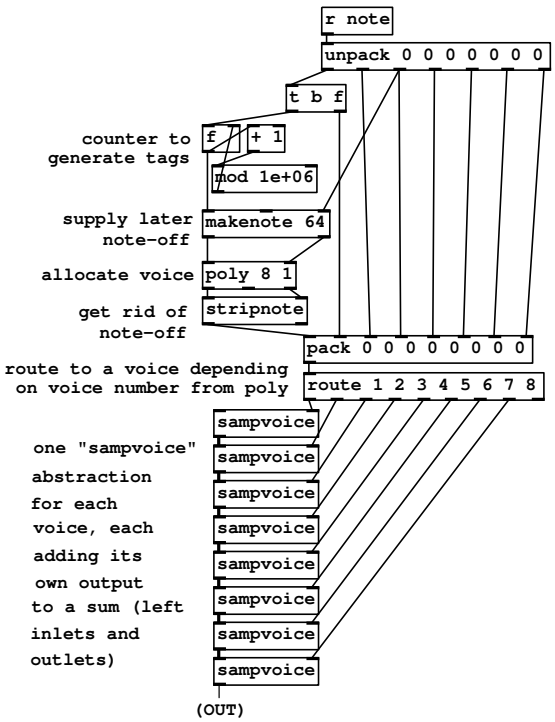


Рис. 4.20: Полифонический семплер, демонстрирующий назначение голоса и использование меток.

присоединяется спереди к семи параметрам, с которых мы начали (объект pack), так что выход объекта pack выглядит как

4 60 90 1000 2 500 10 20

где “4” есть номер голоса, выведенный объектом poly. Этот номер голоса используется для отправки сообщения требуемому голосу с использованием объекта route. Далее соответствующий объект sampvoice получает первоначальный список, начинающийся с “60”.

Внутри объекта sampvoice (рис. 4.21) сообщение используется для управления объектами tabread4~ и окружающими объектами line~ и vline~. Это управление происходит с задержкой в 5 миллисекунд, как в примере семплера, приведённом ранее. Здесь, однако, нам необходимо хранить семь параметров ноты (в предыдущем примере параметров не было). Это сделано с использованием шести объектов f и правого входа объекта delay, крайнего справа. Эти значения используются после задержки в 5 миллисекунд. Это делается вместе с механизмом приглушения звука, описанным на стр. 91, который использует другой объект vline~.

По прошествии 5 миллисекунд, объект vline~, ответственный за генерацию индекса волновой таблицы, получает приказ выступать (и, одновременно, номер волновой таблицы устанавливается для объекта tabread4~, так что амплитудный генератор огибающей начинает атаку[8]). Индекс волновой таблицы должен быть установлен

[8]по-видимому, автор проводит шутливую параллель между атакой звука и военным термином

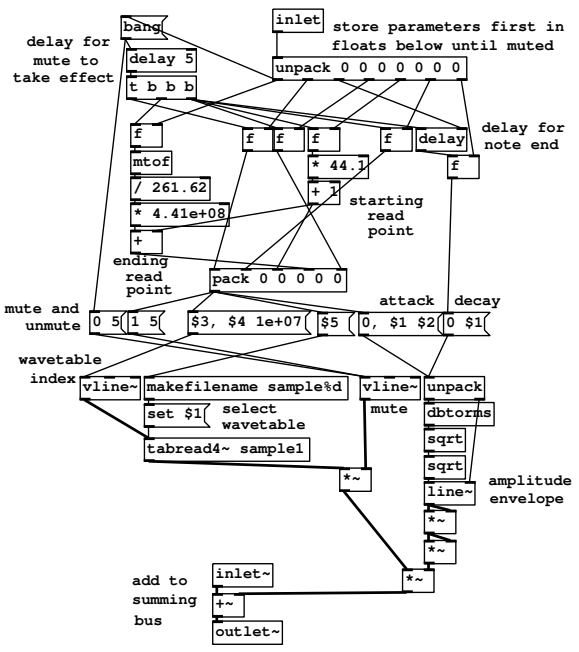


Рис. 4.21: Абстракция sampvoice, использованная в полифоническом семплере рис. 4.20

сразу на начальную позицию, а далее линейно изменён до конечной позиции за подходящее для необходимой транспозиции время. Начальный индекс в семплах равен в точности 44.1, умноженному на локализацию начала в миллисекундах, с добавлением единицы, чтобы сделать возможной четырёхточечную интерполяцию. Он становится третьим числом в запакованном списке, сгенерированном объектом pack в центре голосового патча.

Мы произвольным образом предполагаем, что линейное изменение будет длится десять тысяч секунд (отсюда число “1e+07” в окне сообщения, отправленного на генератор индекса волновой таблицы), надеясь, что это, по крайней мере, достаточно для воспроизведения любой ноты. Конечный индекс — это начальный индекс плюс число семплов, через которые нужно пройти. При единичном коэффициенте транспозиции мы должны пройти 441’000’000 семплов в течение 10’000’000 миллисекунд или пропорционально, в зависимости от коэффициента транспозиции. Этот коэффициент транспозиции вычисляется объектом mtof и последующим делением полученного значения на 261.62 (частота в Гц, соответствующая ноте MIDI 60), так что заданная “высота” MIDI 60 приводит к единичному коэффициенту транспозиции.

Эти и другие параметры объединяются в одно сообщение объектом pack, так что следующие боксы сообщений могут создавать необходимые управляющие сообщения. Единственным новшеством является объект makefilename, который преобразовывает числа, такие как “2”, в символы, такие как “sample2”, для установки имени волновой таблицы объекта tabread4~.

Внизу голосового патча мы видим, как суммирующая шина реализована внутри сабпатча; объект inlet~ берёт сумму предыдущих голосов, добавляется выход

текущего голоса, и результат отправляется на следующий голос с помощью объекта outlet~.

Упражнения

1. Какое входное значение преобразующей функции четвёртой степени даёт на выходе -12 дб, если входное значение “1” даёт 0 дб?

2. Генератор огибающей растёт от нуля до пикового значения в течение отрезка атаки. На сколько децибел меньше пика выходное значение на половине времени атаки, предполагая линейную зависимость? При функции четвёртой степени?

3. Какую степенную преобразующую функцию (то есть какой выбор n для функции $f(x) = x^n$) вы бы использовали, если нужно получить на половине пути значение -12 децибел?

4. Предположим, что вы хотите плавно заменить один сигнал на другой, то есть постепенно увеличить один сигнал, одновременно уменьшая другой. Если они имеют одинаковую мощность и некоррелированы, линейный плавный переход дал бы падение в 3 децибела на полпути перехода. Какой степенной закон вы бы использовали для поддержания постоянной мощности сигнала в течение всего перехода?

5. Трёхнотный аккорд, длительностью 1.5 секунды, воспроизводится, периодически начинаясь, каждую секунду. Сколько голосов необходимо для такого синтеза без прерывания нот?

6. Предположим, что синтезирующий патч выводит сигнал между -1 и $+1$. В момент воспроизведения ноты начинается воспроизведение новой ноты с использованием техники “линейно снизить” с заимствованием голоса. Каков максимальный выходной сигнал?

Глава 5

Модуляция

Пройдя тур из двух глав, касавшийся аспектов управления и организации в электронной музыке, мы возвращаемся к описанию синтеза и технике обработки аудио. Ранее мы встретились с методами аддитивного синтеза и синтеза на основе волновых таблиц. В этой главе мы введём три метода так называемой техники *модуляции*: *амплитудную модуляцию*, *частотную модуляцию* и *вейвшейпинг*. Термин "модуляция" относится, в широком смысле, к любому методу, который систематически меняет форму волны, делая изгибы её графика более выпуклыми или более плоскими. Модуляция широко используется для построения синтетических звуков с различными семействами *спектров*, для чего мы должны разработать некоторую терминологию перед тем, как перейти к технике.

5.1 Классификация спектров

На рис. 5.1 показан метод визуализации *спектра* аудиосигнала. В общих чертах, спектр описывает то, каким образом мощность сигнала распределена по частотам. (Конечно, можно дать гораздо более точные определения, но они потребовали бы более серьезной математической подготовки.)

Часть (a) рисунка показывает спектр *гармонического сигнала*, который является периодическим сигналом, чья фундаментальная частота лежит в диапазоне воспринимаемых высот, примерно от 50 до 4000 герц. Ряд Фурье (стр. 13) даёт описание периодического сигнала в виде суммы синусоид. Частоты синусоид имеют отношения $0 : 1 : 2 : \cdots$. (Константа, являющаяся первым слагаемым в ряде Фурье, может рассматриваться как синусоида

$$a_0 = a_0 \cos(0 \cdot \omega n)$$

с нулевой частотой.)

Для гармонического сигнала мощность спектра сконцентрирована на дискретном подмножестве оси частот (дискретное множество состоит из изолированных

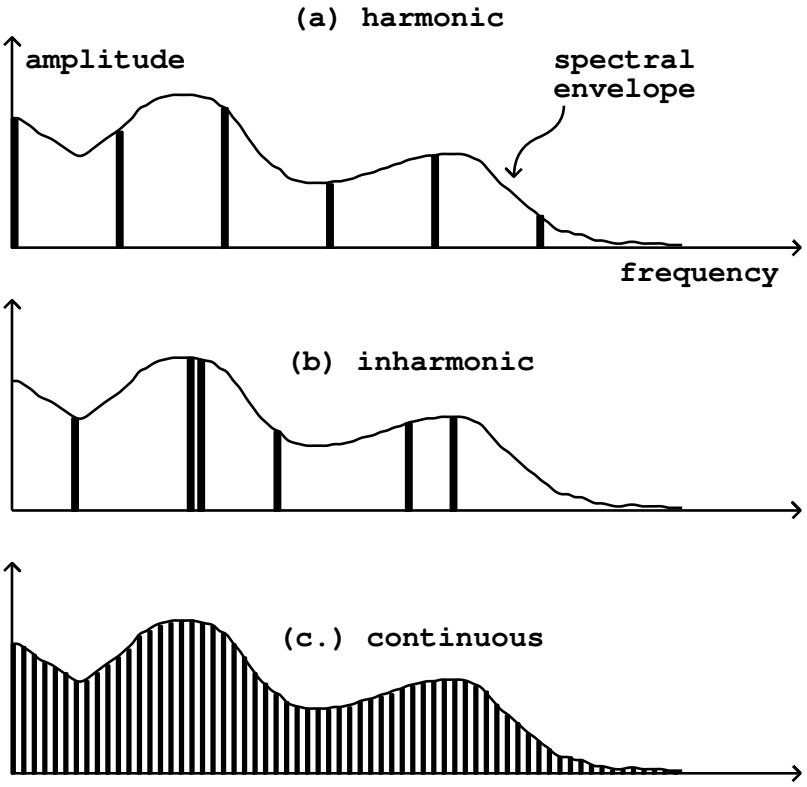


Рис. 5.1: Классификация тембров. Спектральная огибающая описывает форму спектра. Звук может быть дискретно или непрерывно распределённым по частоте; в дискретном случае он может быть гармоническим или ангармоническим.

точек, которых может быть лишь конечное число на любом ограниченном интервале). Мы называем это *дискретным* спектром. Более того, частоты, по которым распределена мощность сигнала, находятся в отношении $0 : 1 : 2 : \cdots$, которое возникает в силу периодичности сигнала. (Присутствие *всех* гармонических частот не является необходимым; некоторые гармоники могут иметь нулевую амплитуду.) Для гармонического сигнала график спектра показывает амплитуды обертонов сигналов. Знание амплитуды и фазы обертонов полностью определяет первоначальный сигнал.

Часть (b) рисунка показывает также дискретный спектр, так что сигнал может опять рассматриваться как сумма ряда гармоник. В этом случае, однако, нет основной частоты, то есть частоты гармоник некратны какой-то общей частоте. Такой сигнал называется *ангармоническим*[1]. (Термины *гармонический* и *ангармонический* могут быть использованы для описания как сигналов, так и их спектров.)

Имея дело с дискретными спектрами, мы описываем амплитуды гармоник несколько неинтуитивным способом. Каждая синусоидальная компонента

$$a\cos(\omega n + \phi)$$

имеет амплитуду $a/2$, если угловая частота ω ненулевая. Но для компоненты нулевой частоты, для которой $\omega = \phi = 0$, амплитуда считается равной a — без деления на два. (Компоненты нулевой частоты часто называются компонентами DC; “DC” исторически является аббревиатурой от “direct current”[2].) Такое соглашение для амплитуд в спектре упростит математику, приведённую далее в этой главе; более глубокая причина для такого соглашения будет выявлена отдельно в Главе 7.

Часть (c) рисунка показывает третью возможность: спектр мог бы и не быть сконцентрированным на дискретном множестве частот, а распределяться по всем возможным частотам. Это может быть названо *непрерывным* или *шумовым* спектром. Спектры необязательно подпадают под категорию дискретного или непрерывного; реальные звуки, на практике, обыкновенно где-то посередине.

Каждая из трёх частей рисунка содержит непрерывную кривую, называемую *спектральной огибающей*. В общем, у звука нет единственной корректно определённой спектральной огибающей; может быть много способов провести гладкую кривую через спектр. С другой стороны, спектральная огибающая может быть определена намеренно; в этом случае обыкновенно ясно, как произвести соответствующий по форме спектр. Например, для дискретного спектра мы можем просто считать со спектральной огибающей нужную амплитуду каждой гармоники и задать её таким образом.

По спектру звука иногда можно сделать вывод о его высоте. Для дискретного спектра высота просто закодирована в частотах гармоник. Гармонические сигналы имеют высоту, определяемую основной частотой; для ангармонических сигналов высота может быть ясной, неопределённой или вообще отсутствовать, согласно сложным и не до конца понятным законам. Шумовой спектр может также иметь

[1] также используется термин *ингармонический*

[2] постоянный ток

воспринимаемую высоту, если спектральная огибающая содержит один или более узких пиков. В общем, звуковая громкость и тембр зависят больше от спектральной огибающей в целом, чем от конкретных частот в спектре, хотя различие между непрерывным и дискретным спектрами может также быть слышимо как разница в тембре.

Тембр, так же как и высота, может эволюционировать в течение жизни звука. Мы говорили здесь о спектрах как статичных сущностях, не рассматривая, изменяются ли они во времени. Если высота и тембр сигнала меняются во времени, мы можем думать о спектре как об изменчивом с течением времени описании моментального поведения сигнала.

Эта точка зрения на звуки является сильно упрощённой. Реальные свойства воспринимаемой высоты звука и тембра имеют много аспектов, которые не могут быть объяснены в терминах этой модели. Например, характеристика тембра, называемая “неровность”, иногда мыслится как отражающее быстрые изменения спектральной огибающей во времени. Приведенное здесь упрощённое описание тем не менее полезно при обсуждении того, как создавать дискретные или непрерывные спектры для широкого множества музыкальных целей, что будет показано в оставшейся части этой главы.

5.2 Умножение аудиосигналов

Начиная с Главы 1, мы регулярно складывали аудиосигналы вместе и умножали их на медленно меняющиеся сигналы (используемые, например, как амплитудные огибающие). Для полного понимания алгебры аудиосигналов мы должны также рассмотреть ситуацию, в которой умножаются два аудиосигнала, ни один из которых не меняется медленно. Ключ к пониманию того, что произойдёт даёт *формула произведения косинусов*:

$$\cos(a)\cos(b) = \frac{1}{2}\Big[\cos(a+b) + \cos(a-b)\Big] \tag{5.1}$$

Для того чтобы убедиться, что эта формула верна, мы можем использовать формулу косинуса суммы двух углов

$$\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$$

для вычисления правой части формулы произведения косинусов; после упрощения мы получим левую часть.

Мы можем использовать эту формулу для того, чтобы увидеть, что происходит, когда мы умножаем две синусоиды (стр. 1):

$$\begin{gathered}\cos(\alpha n + \phi)\cos(\beta n + \xi) = \\ = \tfrac{1}{2}\Big[\cos\left((\alpha+\beta)n + (\phi+\xi)\right) + \cos\left((\alpha-\beta)n + (\phi-\xi)\right)\Big].\end{gathered}$$

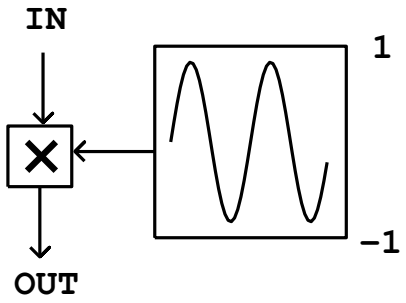


Рис. 5.2: Блок-схема кольцевой модуляции входного сигнала синусоидой.

Умножение двух синусоид даёт в результате две гармоники, одна из которых определяется суммой первоначальных частот, а вторая — разностью. (Если разность $\alpha - \beta$ оказалась отрицательной, просто поменяйте местами две первоначальные синусоиды и разность станет положительной.) Эти две новые компоненты называются *боковыми полосами частот*.

Это даёт нам метод для сдвига частот составляющих звука, называемый *кольцевой модуляцией*, который, в его простейшей форме, показан на рис. 5.2. Осциллятор производит *несущий сигнал*, который просто умножается на входной. В этом контексте входной сигнал называется *модулирующим*. Термин "кольцевая модуляция" часто используется в более общем смысле для обозначения перемножения любых двух сигналов, но здесь мы будем лишь рассматривать использование синусоидального несущего сигнала. (Техника кольцевой модуляции восходит к аналоговой эре [Str95]; цифровые умножители заменили теперь как VCA (§1.5) так и кольцевой модулятор.)

Рисунок 5.3 показывает множество результатов, которые могут быть получены при умножении (модулирующей) синусоиды с угловой частотой α и пиковой амплитудой $2a$ на (несущую) синусоиду с угловой частотой β и пиковой амплитудой 1:

$$[2a\cos(\alpha n)] \cdot [\cos(\beta n)]$$

(для простоты фазовая часть опущена). Каждая часть рисунка показывает одновременно как модулирующий сигнал, так и результат модуляции на том же спектре. Модулирующий сигнал появляется с единственной частотой α и амплитудой a. Произведение, в общем, имеет две частотных компоненты, каждая с амплитудой $a/2$.

Части (a) и (b) рисунка показывают "общие" случаи, где α и β ненулевые и отличны друг от друга. Составляющие частоты выхода есть $\alpha + \beta$ и $\alpha - \beta$. В части (b), так как $\alpha - \beta < 0$, мы получаем отрицательную частотную компоненту. Поскольку косинус является чётной функцией, мы имеем

$$\cos((\alpha - \beta)n) = \cos((\beta - \alpha)n)$$

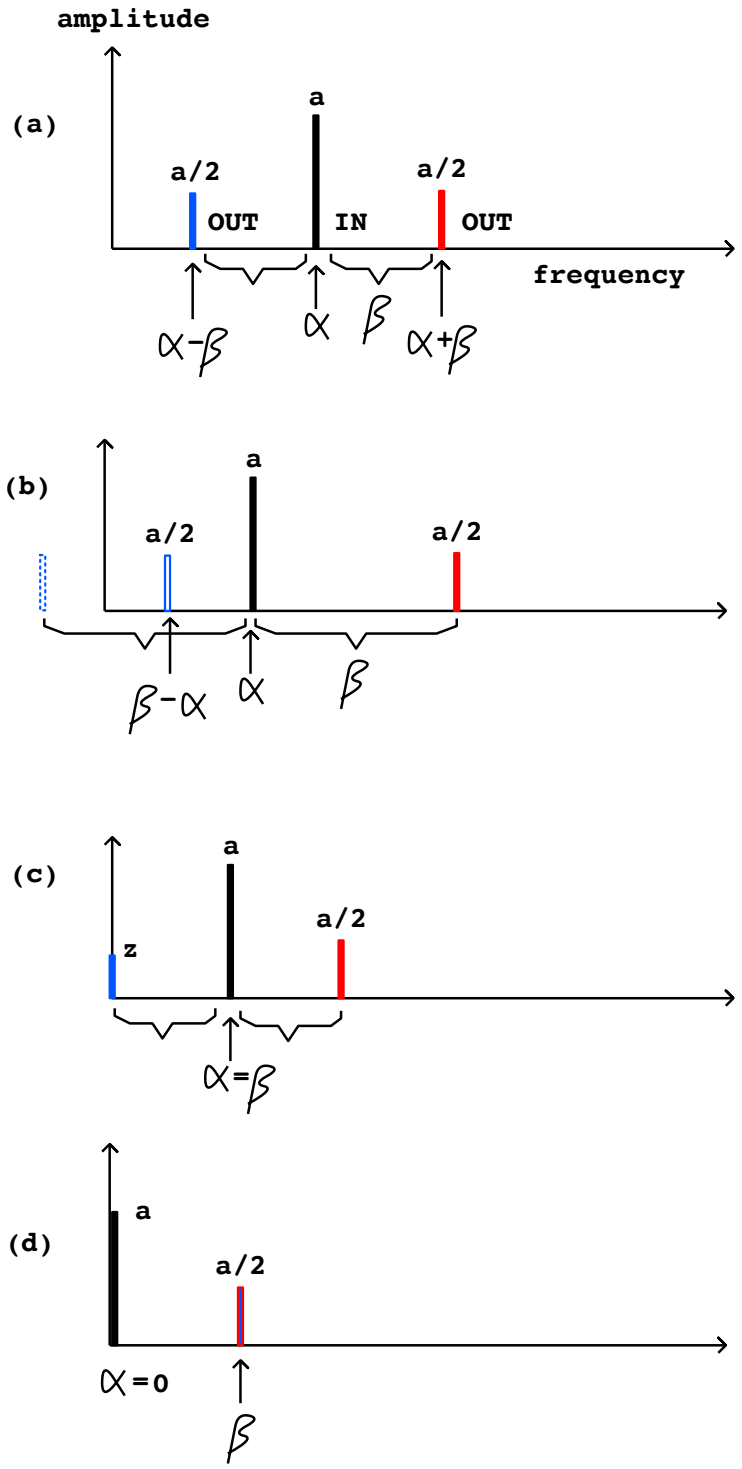


Рис. 5.3: Боковые полосы частот, возникающие при умножении двух синусоид с частотами α и β:
(a) при $\alpha > \beta > 0$;
(b) при $\beta > \alpha$, так что нижняя боковая полоса отражается относительно оси $f = 0$;
(c) при $\alpha = \beta$, так что амплитуда боковой полосы с нулевой частотой зависит от фаз двух синусоид;
(d) при $\alpha = 0$.

так что отрицательная составляющая в точности эквивалентна составляющей с положительной частотой $\beta - \alpha$ и той же амплитудой.

В специальном случае, когда $\alpha = \beta$, вторая (разностная) боковая полоса имеет нулевую частоту. В этом случае фаза будет важна, так что, переписывая произведение с точными фазами и заменяя β на α, получаем:

$$2a\cos(\alpha n + \phi)\cos(\alpha n + \xi) =$$

$$= a\cos\left(2\alpha n + (\phi + \xi)\right) + a\cos\left(\phi - \xi\right).$$

Второе слагаемое имеет нулевую частоту; его амплитуда зависит от относительной фазы двух синусоид и может принимать значения от $+a$ до $-a$, когда разность фаз меняется от 0 до π радиан. Эта ситуация показана в части (c) рис. 5.3.

Наконец, часть (d) показывает несущий сигнал, чья частота равна нулю. Его значение равно константе a (не $2a$; нулевая частота — особый случай). Здесь мы получаем только одну боковую полосу, обыкновенно, с амплитудой $a/2$.

Мы можем использовать свойство дистрибутивности умножения для того, чтобы найти, что случится при перемножении сигналов, состоящих из более чем одной гармоники. Например, в ситуации выше мы можем заменить сигнал с частотой α на сумму нескольких синусоид, такую как

$$a_1\cos(\alpha_1 n) + \cdots + a_k\cos(\alpha_k n).$$

Умножение на сигнал с частотой β даёт гармоники с частотами, равными

$$\alpha_1 + \beta, \alpha_1 - \beta, \ldots, \alpha_k + \beta, \alpha_k - \beta.$$

Рисунок 5.4 показывает результат умножения сложного периодического сигнала (с несколькими компонентами, находящимися в соотношениях $0 : 1 : 2 : \cdots$) на синусоиду. Как спектральная огибающая, так и составляющие частоты в результате меняются согласно относительно простым правилам.

Получаемый спектр, в сущности, представляет из себя первоначальный спектр, комбинированный со своим отражением относительно вертикальной оси. Этот комбинированный спектр, далее, сдвигается вправо несущей частотой. Наконец, если какая-либо компонента сдвинутого спектра всё ещё находится левее вертикальной оси, она отражается относительно неё, чтобы сделать частоты опять положительными.

В части (b) рисунка несущая частота (частота синусоиды) ниже основной частоты сложного сигнала. В этом случае сдвиг происходит на относительно малое расстояние, так что свертка спектра в конце практически совмещает обе половинки. В результате мы получаем спектральную огибающую, в общих чертах напоминающую оригинал (хотя и с амплитудой в два разе меньше и в два раза большей плотностью спектра).

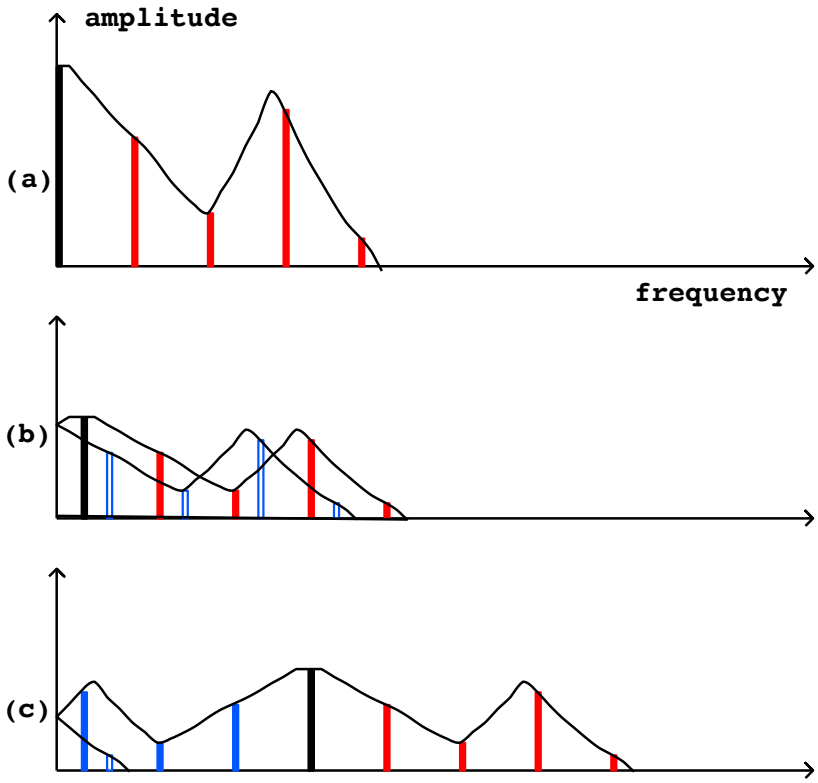


Рис. 5.4: Результат кольцевой модуляции сложного сигнала чистой синусоидой:
(a) спектр первоначального сигнала и спектральная огибающая;
(b) модуляция с относительно низкой модулирующей частотой (1/3 основной);
(c) модуляция с высокой частотой, 10/3 основной.

Специальный случай, не показанный здесь, состоит в использовании несущей частоты равной половине основной. В этом случае пары гармоник будут попадать одна на другую, и мы будем иметь отношения $1/2 : 3/2 : 5/2 : \cdots$, что даёт сигнал только с нечётными обертонами, на октаву ниже первоначального. Это очень простой и эффективный делитель октавы для гармонического сигнала, в предположении, что вы знаете или можете найти его основную частоту. Если вам нужны как чётные обертоны, так и нечётные (для сигнала сдвинутого вниз на октаву), просто смешайте первоначальный сигнал с модулированным.

Часть (c) рисунка показывает эффект использования модулирующей частоты, более высокой, чем основная частота сложного сигнала. Здесь эффект развёртывания виден гораздо яснее (только одна гармоника, самая левая, должна была быть отражена, чтобы сделать частоту положительной). Спектральная огибающая теперь сильно смещена от первоначальной; это смещение часто дает более слышимый эффект, чем перемещение составляющих гармоник.

В другом специальном случае несущая частота может быть кратна основной частоте сложного периодического сигнала; тогда все гармоники попадают на другие гармоники той же основной частоты, и единственным эффектом станет сдвиг спектральной огибающей.

5.3 Вейвшейпинг

Другой подход к модуляции сигнала, называемый *вейвшейпингом*, состоит в простом пропускании его через выбранную подходящим образом нелинейную функцию. Блок-схема для этой операции показана на рис. 5.5. Функция $f()$, называемая *преобразующей функцией*, деформирует входящую форму волны в другую. Новая форма зависит от формы входящей волны, от преобразующей функции и, существенным образом, от амплитуды входного сигнала. Поскольку амплитуда входящей волны влияет на форму выходной волны (и, следовательно, тембр) это даёт нам простой способ создать непрерывно меняющееся семейство тембров, просто варьируя входной уровень при преобразовании. По этой причине управление входной амплитудой часто включают как часть операции вейвшейпинга, как показано на блок-схеме.

Амплитуда входящего сигнала называется *индексом* вейвшейпинга. Во многих ситуациях небольшой индекс приводит к относительно малой деформации (так что выход очень похож на вход), а больший индекс даёт более искаженный, более богатый тембр.

Рисунок 5.6 показывает пример распространенного вида вейвшейпинга, в котором $f()$ равносильна *ограничительной функции*. Этот пример ясно показывает, как входная амплитуда (индекс) может воздействовать на выходную форму волны. Ограничительная функция пропускает свой входной сигнал без изменений, пока он остаётся в интервале между -0.3 и $+0.3$. Таким образом, когда входной сигнал не превышает 0.3 по абсолютной величине, выход совпадает со входом. Но когда

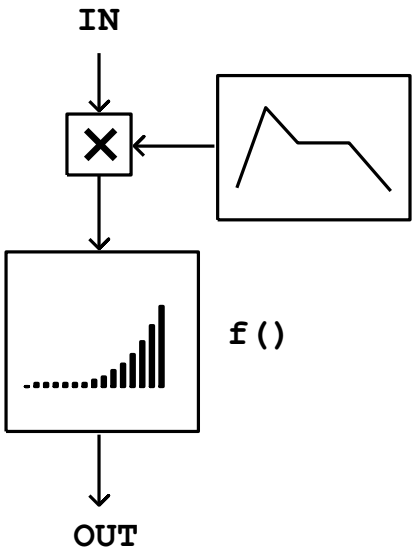


Рис. 5.5: Блок-схема вейвшейпинга входного сигнала, использующая нелинейную функцию $f()$. Этап корректировки амплитуды предшествует действию функции для достижения различного эффекта просмотра волновой таблицы при различных амплитудах.

входной сигнал выходит за эти пределы, выходной сигнал остаётся внутри них; и с ростом амплитуды сигнала эффект от действия этого ограничения становится все сильнее. На рисунке входным сигналом является затухающая синусоида. Выходной сигнал изменяется от почти квадратной формы волны в начале до чистой синусоиды в конце. Этот эффект хорошо известен каждому, кто воспроизводит звук с помощью усилителя, поставленного на слишком большое усиление. Чем громче входной сигнал, тем более искажённым будет выходной. По этой причине вейвшейпинг также иногда называют эффектом *дисторшн*[3].

Рисунок 5.7 показывает более простую и легкую для анализа ситуацию, в которой преобразующая функция просто возводит в квадрат свой входной сигнал:

$$f(x) = x^2.$$

Для синусоидального входа

$$x[n] = a\cos(\omega n + \phi)$$

мы получаем

$$f(x[n]) = \frac{a^2}{2}\left(1 + \cos(2\omega n + 2\phi)\right).$$

Если амплитуда a равна единице, это в точности совпадает с кольцевой модуляцией синусоиды другой синусоидой той же частоты, чей результат мы описали

[3] “искажение” (*прим. перев.*)

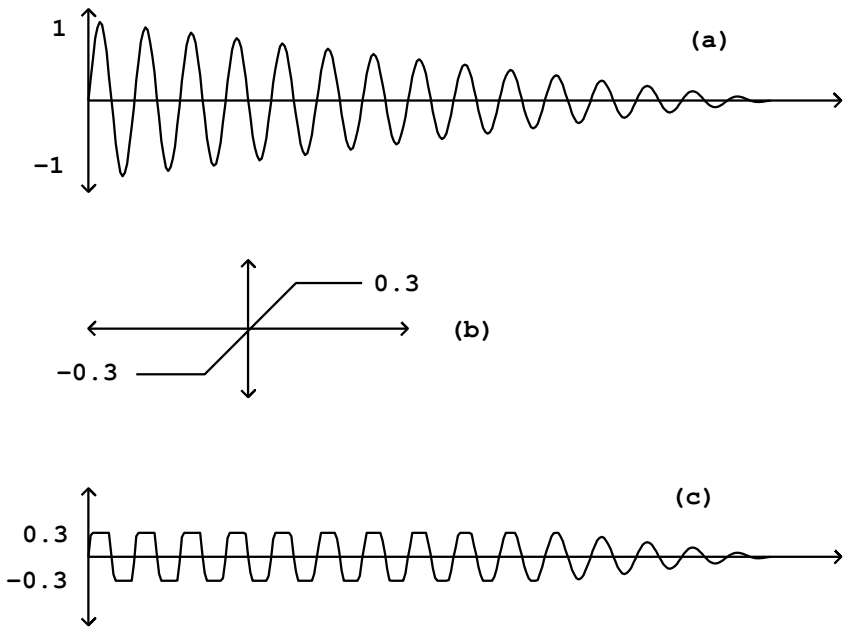


Рис. 5.6: Ограничение сигнала, как пример вейвшейпинга: (a) вход — затухающая синусоида; (b) функция вейвшейпинга, которая ограничивает входной сигнал до интервала между -0.3 и $+0.3$; (c) результат.

в предыдущем параграфе: выходом является синусоида DC (с нулевой частотой) плюс синусоида с удвоенной первоначальной частотой. Однако в этом примере вейвшейпинга, в отличие от кольцевой модуляции, выходная амплитуда растёт как квадрат входа.

Сохраняя ту же функцию преобразования, мы теперь рассмотрим эффект отправки входного сигнала, состоящего из комбинации двух синусоид с амплитудами a и b и угловыми частотами α и β. Для простоты мы опустим начальные фазы. Мы задаём

$$x[n] = a\cos(\alpha n) + b\cos(\beta n)$$

и подстановка этого выражения в $f()$ даёт

$$\begin{gathered} f(x[n]) = \tfrac{a^2}{2}\left(1+\cos(2\alpha n)\right) + \\ +\tfrac{b^2}{2}\left(1+\cos(2\beta n)\right) \\ +ab\left[\cos((\alpha+\beta)n) + \cos((\alpha-\beta)n)\right]. \end{gathered}$$

Первые два слагаемых — это лишь то, что мы бы получили, отправляя две синусоиды по отдельности. Третье слагаемое представляет из себя удвоенное произведение двух слагаемых входа, которое получается из среднего перекрёстного члена в выражении

$$f(x+y) = x^2 + 2xy + y^2.$$

Этот эффект, называемый *интермодуляцией*, становится всё более и более доминирующим, когда число слагаемых на входе растёт; если имеется k синусоид

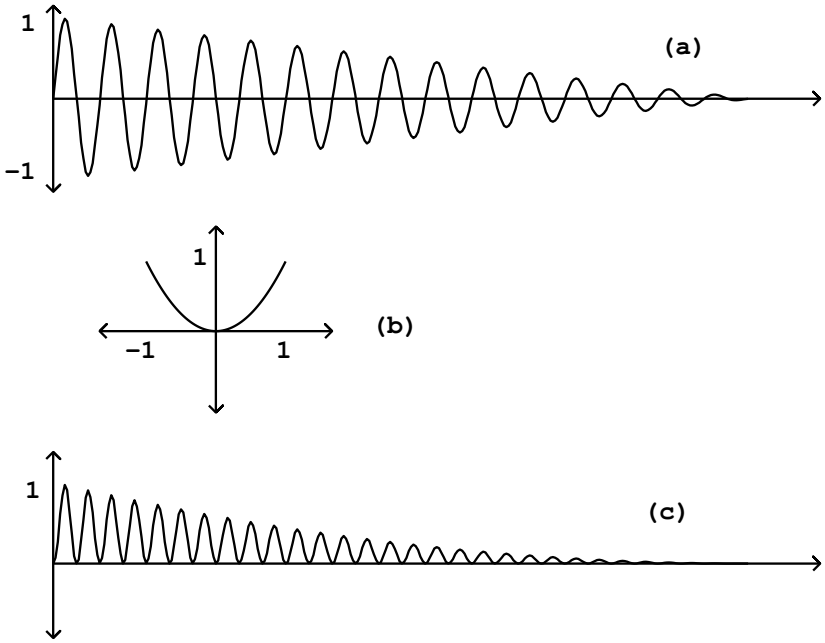


Рис. 5.7: Вейвшейпинг, использующий возводящую в квадрат функцию преобразования $f(x) = x^2$: (a) вход;
(b) преобразующая функция;
(c) результат, звучащий с удвоенной первоначальной частотой.

на входе, то существует только k “прямых” слагаемых в результате, в то время как интермодуляционных слагаемых $k(k-1)/2$.

В противоположность кольцевой модуляции, представляющей из себя линейную функцию входного сигнала, вейвшейпинг нелинеен. В то время как для линейных процессов мы были в состоянии их анализировать, рассматривая их действие на все компоненты входа по отдельности, в нелинейном случае мы также должны рассматривать взаимодействия между компонентами. Результаты гораздо сложнее: иногда богаче по звуку, но, с другой стороны, более трудны для понимания и предсказания.

В общем, мы можем показать, что периодический вход, вне зависимости от своей сложности, будет повторяться с тем же периодом после вейвшейпинга: если период есть τ, так что

$$x[n+\tau] = x[n]$$

и временно устанавливая индекс $a = 1$, то

$$f(x[n+\tau]) = f(x[n]).$$

(В некоторых специальных случаях выходной сигнал может повторяться с периодом, кратным τ, так что мы получаем гармонику входа в результате; это происходит в примере на рис. 5.4.)

Составление комбинаций периодических тонов с консонансными интервалами может привести к искажению результатов на субгармониках. Например, если два периодических сигнала x и y составляют музыкальную кварту (периоды в отношении 4 : 3), тогда период повтора их суммы происходит с более низкой скоростью, задаваемой общей субгармоникой. В виде уравнений мы бы имели:

$$x[t+\tau/3] = x[t]$$
$$y[t+\tau/4] = y[t],$$

следовательно,

$$x[t+\tau] + y[t+\tau] = x[t] + y[t]$$

и, таким образом, обработанная вейшейпером сумма $f(x+y)$ повторялась бы через период τ:

$$f(x+y)[n+\tau] = f(x+y)[n].$$

Каждый электрогитарист, который устанавливает усилитель на "перегруз" и играет верхние открытые струны *си* и *ми* вместе, заметит: искажённый сигнал иногда звучит на высоте 6-й струны *ми*, на две октавы ниже 1-й струны *ми*.

Для получения некоторого более точного анализа влияния вейвшейпинга на входящий сигнал, иногда удобно записывать функцию как конечный или бесконечный *степенной ряд*:

$$f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots$$

Если входной сигнал $x[n]$ — синусоида с единичной амплитудой, $cos(\omega n)$, мы можем рассматривать действие слагаемых, приведённых выше, по отдельности:

$$f(a \cdot x[n]) = f_0 + a f_1 \cos(\omega n) + a^2 f_2 \cos^2(\omega n) + a^3 f_3 \cos^3(\omega n) + \cdots$$

Поскольку слагаемые ряда последовательно умножаются на возрастающие степени индекса a, меньшее значение a будет выделять более ранние слагаемые в большей степени, а большее значение a будет выделять более поздние.

Спектры отдельных слагаемых могут быть найдены с помощью многократного применения формулы умножения косинусов:

$$1 = \cos(0)$$

$$x[n] = \cos(\omega n)$$

$$x^2[n] = \tfrac{1}{2} + \tfrac{1}{2}\cos(2\omega n)$$

$$x^3[n] = \tfrac{1}{4}\cos(-\omega n) + \tfrac{2}{4}\cos(\omega n) + \tfrac{1}{4}\cos(3\omega n)$$

$$x^4[n] = \tfrac{1}{8}\cos(-2\omega n) + \tfrac{3}{8}\cos(0) + \tfrac{3}{8}\cos(2\omega n) + \tfrac{1}{8}\cos(4\omega n)$$

$$x^5[n] = \tfrac{1}{16}\cos(-3\omega n) + \tfrac{4}{16}\cos(-\omega n) + \tfrac{6}{16}\cos(\omega n) + \tfrac{4}{16}\cos(3\omega n) + \tfrac{1}{16}\cos(5\omega n)$$

и так далее. Можно заметить, что числители дробей образуют треугольник Паскаля. Центральная предельная теорема теории вероятностей указывает, что каждый k-ый ряд может быть аппроксимирован гауссовой кривой, чьё стандартное отклонение (мера ширины) пропорционально квадратному корню из k.

Слагаемые с отрицательной частотой (которые показаны здесь отдельно для ясности) должны быть скомбинированы с теми, у которых она положительна; спектральная огибающая сворачивается так же, как при кольцевой модуляции в примере рис. 5.4.

Пока коэффициенты f_k все положительны или равны нулю, то же будет для всех амплитуд синусоид в разложениях выше. В этом случае все фазы остаются согласованными при изменении a и, таким образом, мы получаем расширение спектра (и, возможно, радикально возрастающие амплитуды) при увеличении величин a. С другой стороны, если некоторые f_k положительны, а другие отрицательны, различные разложения выше, будут взаимодействовать разрушающим структуру образом; это будет давать более сложные спектральные изменения звука.

Заметим также, что последовательные разложения содержат только чётные или только нечётные гармоники. Если преобразующая функция (в форме ряда) оказалась содержащей только чётные степени:

$$f(x) = f_0 + f_2x^2 + f_4x^4 + \cdots$$

тогда результат, имея только чётные гармоники, будет звучать на октаву выше, чем входящая синусоида. Если только нечётные степени появляются в разложении $f(x)$, тогда выходной сигнал будет содержать лишь нечётные обертоны. Даже если f не может быть выражена точно, как степенной ряд (например, для ограничительной функции на рис. 5.3), всё ещё остаётся справедливым, что если f — чётная функция, то есть, если

$$f(-x) = f(x),$$

вы будете получать только чётные гармоники, и если f — нечётная функция

$$f(-x) = -f(x),$$

вы будете получать нечётные гармоники.

Существует множество математических приёмов для использования при вейвшейпинге с целью генерации определённых спектров. Например, вы можете генерировать чистые синусоиды с любыми гармониками основной частоты с использованием полиномов Чебышёва в качестве преобразующей функции [Bru79] [DJ85], и отсюда вы можете продвинуться к построению любого требуемого статического спектра (пример E05.chebychev.pd демонстрирует это). Генерирование *семейства спектров* при помощи вейвшейпинга синусоиды с изменяемой амплитудой оказывается более сложным, хотя было найдено несколько интересных специальных случаев. Некоторые из них подробно разбираются в Главе 6.

5.4 Частотная и фазовая модуляция

Если частота синусоиды изменяется медленно, мы воспринимаем это, как изменение высоты тона. Но если высота меняется так быстро, что наши уши не могут отслеживать изменение (например, если изменение само по себе случается с основной частотой синусоиды или близко к ней), мы слышим изменение тембра. Тембры, генерированные таким образом, богаты и широко варьируются. Открытие Джоном Чоунингом (John Chowning) этой возможности [Cho73] создало революцию в области компьютерной музыки. Здесь мы введём *частотную модуляцию*, обыкновенно называемую FM (frequency modulation), как особый случай вейвшейпинга [Bru79] [DJ85, стр.155–158]; анализ, данный в работе [Puc01], несколько другой.

Техника FM, в простейшей форме, показана на рис. 5.8 (часть a). Частота частотно-моделированной синусоиды меняется синусоидально, с некоторой угловой частотой ω_m, около центральной частоты ω_c, так что мгновенные частоты изменяются между $(1-r)\omega_c$ и $(1+r)\omega_c$ с параметрами: ω_m, управляющим частотой изменения, и r, управляющим глубиной изменения. Параметры ω_c, ω_m и r называются *несущей частотой*, *модулирующей частотой* и *индексом модуляции* соответственно.

Обыкновенно используется более простая, эквивалентная по сути формулировка, в которой синусоидально модулируется фаза вместо несущей частоты. (Это даёт эквивалентный результат, поскольку мгновенная частота представляет из себя скорость изменения фазы, и поскольку скорость изменения синусоиды представляет из себя другую синусоиду.) Формулировка фазовой модуляции демонстрируется в части (b) рисунка.

Мы можем проанализировать результат фазовой модуляции следующим образом, предполагая, что модулирующий осциллятор и волновая таблица имеют синусоидальный вид, и что несущая и модулирующая частота не меняются во времени сами по себе. Результирующий сигнал может тогда быть записан как

$$x[n] = \cos(a\cos(\omega_m n) + \omega_c n).$$

Параметр a, который играет роль параметра r, использованного ранее, по аналогии называется индексом модуляции; он также управляет шириной изменения частоты по отношению к несущей частоте ω_c. Если $a = 0$, то изменения частоты не происходит, и выражение редуцируется к неменяющейся несущей синусоиде; по мере роста a, форма волны становится более сложной.

Для анализа получающегося спектра мы можем переписать сигнал как

$$\begin{aligned} x[n] = {} & \cos(\omega_c n) * \cos(a\cos(\omega_m n)) \\ & - \sin(\omega_c n) * \sin(a\cos(\omega_m n)). \end{aligned}$$

Мы можем рассматривать результат как сумму двух вейвшейперов, каждый из которых воздействует на синусоиду ω_m индексом вейвшейпинга a, одновременно

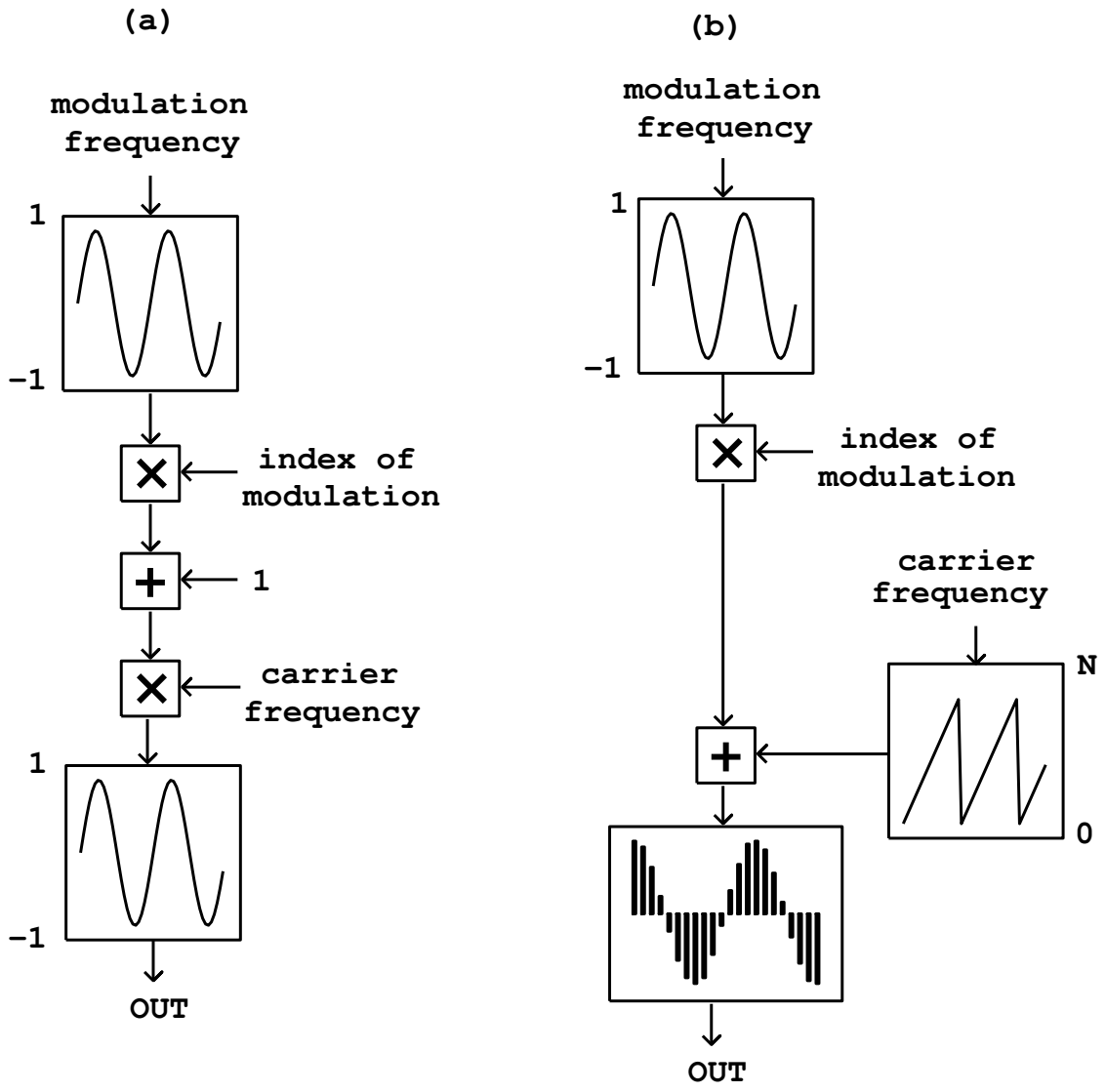


Рис. 5.8: Блок-схема синтеза частотной модуляции (FM): (a) в классическом виде; (b) реализованной как фазовая модуляция.

модулируя ее синусоидой с частотой ω_c. Функция f вейвшейпинга определяется как $f(x) = cos(x)$ для первого слагаемого и $f(x) = sin(x)$ для второго.

Возвращаясь к рис. 5.4, мы можем предсказать, как будет выглядеть спектр. Два гармонических спектра с выхода вейвшейпера

$$\cos(a\cos(\omega_m n))$$

и

$$\sin(a\cos(\omega_m n))$$

имеют гармоники, настроенные соответственно как

$$0, 2\omega_m, 4\omega_m, \ldots$$

и

$$\omega_m, 3\omega_m, 5\omega_m, \ldots,$$

причём каждая умножается на синусоиду с несущей частотой. Таким образом, получаем спектр, сконцентрированный на несущей частоте ω_c, с боковыми полосами, являющимися чётными и нечётными кратными модулирующей частоты, которые получаются благодаря вейвшейпингу синусом и косинусом выше соответственно. Изменение индекса модуляции a управляет относительной силой различных гармоник. Гармоники сами по себе размещены на частотах

$$\omega_c + m\omega_m$$

где

$$m = \ldots - 2, -1, 0, 1, 2, \ldots$$

Как в любой ситуации, где два периодических сигнала перемножаются, если есть некоторое общее кратное двух периодов, получающееся произведение будет повторяться с наиболее длинным периодом. Так что, если два периода есть $k\theta$ и $m\theta$, где k и m взаимно просты, они оба повторяются с интервалом времени $km\theta$. С другой стороны, если нет общего кратного или оно меньше, чем любая различимая высота, тогда результат будет ангармоническим.

Гораздо больше о FM можно найти в учебниках [Moo90, с. 316] [DJ85, стр. 115-139] [Bou00] и исследовательской литературе. Некоторые из возможностей показаны в примерах ниже.

5.5 Примеры

5.5.1 Кольцевая модуляция и спектры

Пример E01.spectrum.pd служит для отображения спектра, который мы будем использовать; сейчас мы перейдем ко второму примеру, E02.ring.modulation.pd,

который показывает эффект кольцевой модуляции на гармонический спектр (выполненной теоретически в §5.2 и показанной на рис. 5.4). В рассматриваемом примере, сигнал имеет гармоники (от 0 до 5) с единичной амплитудой. Гармоники могут быть включены и выключены по отдельности с использованием выключателей. Когда все они включены, спектральная огибающая имеет пик на гармонике DC с нулевой частотой (поскольку постоянный сигнал даёт в два раза более сильный вклад, чем остальные гармоники), плоскую структуру с 1 по 5 гармонику и, далее, спадает к нулю.

В части патча, генерирующей сигнал (часть *a* рисунка), мы суммируем шесть гармоник и умножаем сумму на один несущий осциллятор. (Шесть сигналов суммируются неявно путём подсоединения их всех к одному входу объекта *~.) Сверху вычисляется величина основной частоты для корректного построения спектрального анализа, чей результат показан в части (*b*) рисунка.

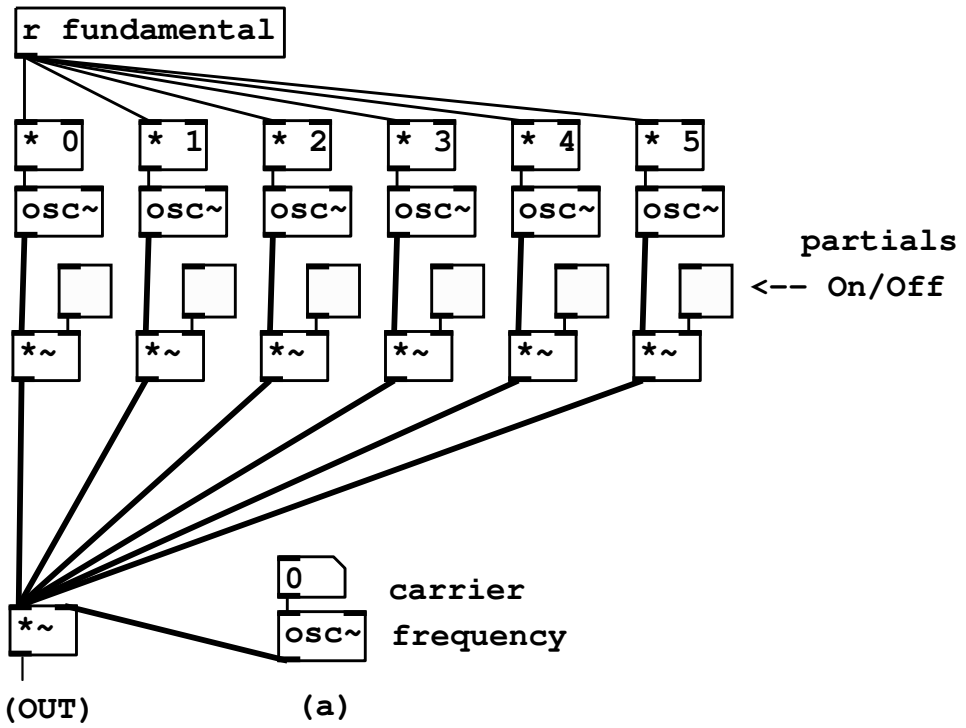
Спектральный анализ (использующий технику, которую мы не будем описывать вплоть до Главы 9) показывает положение синусоид (предполагая дискретный спектр) на горизонтальной оси и их мощность на вертикальной. Таким образом, положение пика мощности входного сигнала в спектре на DC предсказывает, как на рис. 5.3, что должен быть пик в выходном спектре на несущей частоте с высотой 1/2. Сходным образом две другие синусоиды во входном сигнале, которые имеют высоту 1/2 в спектре, приводят каждая к двум пикам на выходе высотой 1/4. Один из этих четырёх пиков получается при отражении на левом крае рисунка (при взятии абсолютной величины отрицательной частоты).

5.5.2 Деление октавы и добавление формант

Как показано в §5.2, при рассматривании результата модуляции сложного гармонического (то есть периодического) сигнала синусоидой, интересным специальным случаем является установка несущего осциллятора на 1/2 основной частоты, что даёт падение выходного звука на октаву с относительно небольшим изменением спектральной огибающей. Другим случаем будет модуляция синусоидой с частотой пропорциональной основной, что в результате приводит к замене спектральной огибающей без изменения основной частоты. Это продемонстрировано в примере E03.octave.divider.pd (рис. 5.10).

Сигнал, который мы обрабатываем здесь — записанный голос говорящего. Сабпатчи pd looper и pd delay скрывают детали. Первый — циклический семплер, определённый в Главе 2. Второй — задержка в 1024 семпла, которая использует объекты определённые позднее, в Главе 7. Мы определим здесь один новый класс объектов:

fiddle~ : *анализатор высоты.* Его вход принимает сигнал для анализа и сообщения для изменения настроек. В зависимости от его аргументов, fiddle~ может иметь различное число выходов, выводящих разную информацию о входном сигнале. Как показано здесь, с одним аргументом (размера окна)



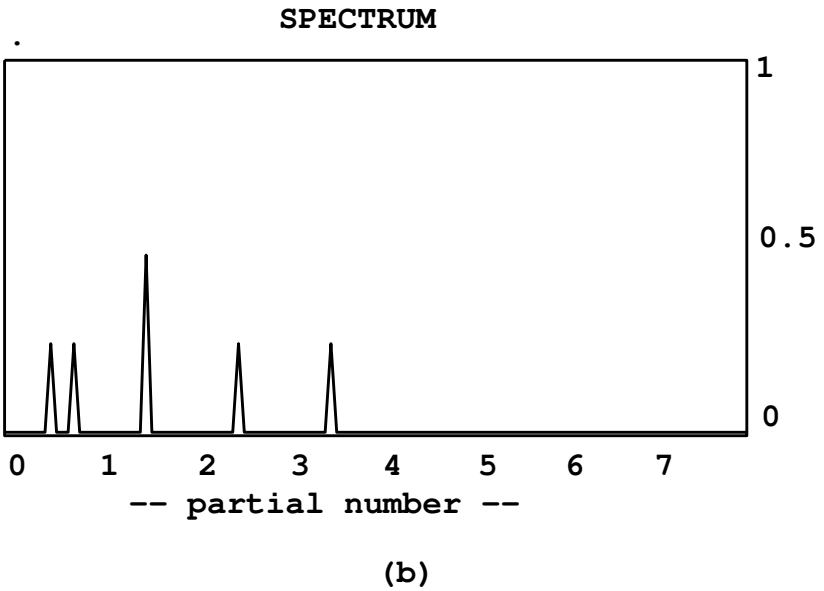


Рис. 5.9: Кольцевая модуляция сложного тона синусоидой: (a) её реализация; (b) измеренный спектр.

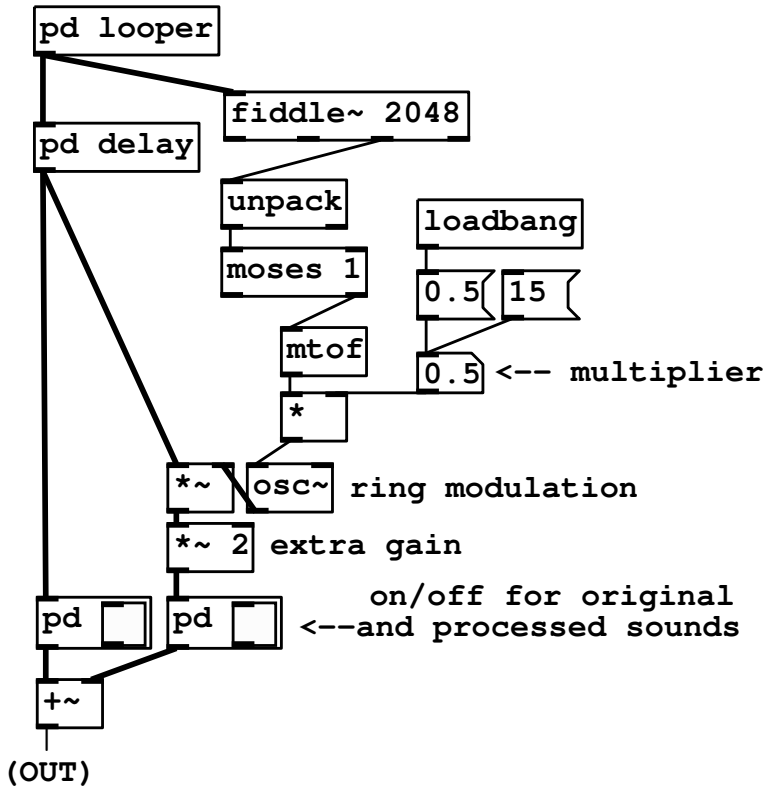


Рис. 5.10: Понижение высоты звука на октаву с помощью определения его высоты и последующей модуляцией с половиной основной частоты.

третий выход выводит высоту (в единицах MIDI) и амплитуду входного сигнала. Эта информация выводится в виде списка из двух чисел. Если высоту невозможно идентифицировать — выводится ноль.

В этом патче третий выход распаковывается на его высотную и амплитудную компоненты, и высотная компонента фильтруется объектом moses, так что используются только успешные оценки (ненулевые). Они обращаются в единицы частоты объектом mtof. Окончательно оценки частоты или уменьшаются на 1/2, или, наоборот, умножаются на 15 (в зависимости от выбранного множителя) для создания модулирующей частоты. В первом случае мы получаем делитель октавы, а во втором — дополнительные высокие гармоники, которые преобразуют гласные звуки.

5.5.3 Вейвшейпинг и разностные тоны

Пример E04.difference.tone.pd (рис. 5.11) — вейвшейпер, демонстрирующий нелинейность этого процесса. Две синусоиды (300 и 225 герц в отношении 4 : 3) суммируются и затем обрезаются с использованием нового класса объектов:

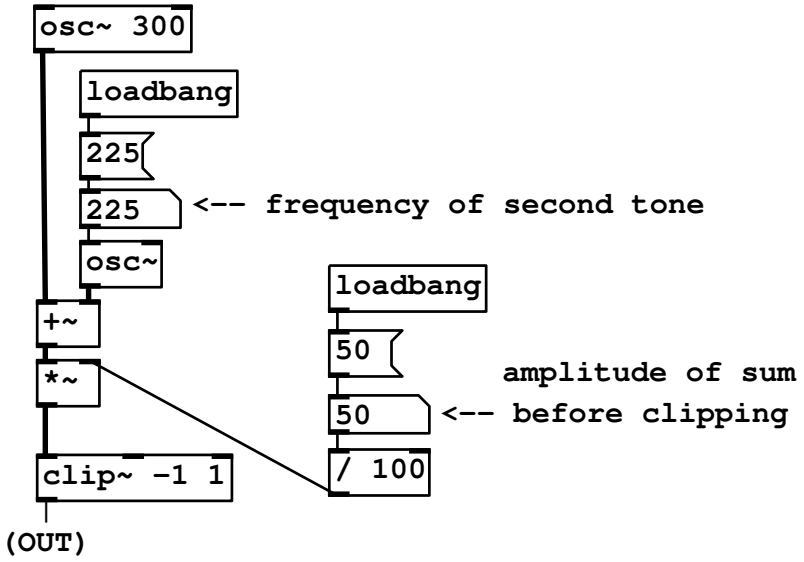


Рис. 5.11: Нелинейное искажение суммы двух синусоид для создания разностного тона.

clip~ : *ограничение сигнала.* Когда сигнал лежит в пределах, определённых аргументами объекта clip~, он пропускается без изменения; но когда он попадает ниже нижнего предела или возрастает выше верхнего предела, он меняется на предельное значение. Эффект обрезания синусоидального сигнала показан графически на рис. 5.6.

Пока амплитуда суммы синусоид меньше 50% сумма не может превысить единицу по абсолютной величине, и объект clip~ пропускает пару синусоид без изменения на свой выход. С другой стороны, когда амплитуда превышает 50%, нелинейность объекта clip~ даёт искажённые результаты (с частотами $300m+225n$ для целых m и n), все являющиеся кратными 75, что является основной частотой результирующего тона. Другими словами, наиболее короткий общий период двух синусоид составляет 1/75 секунды (что является четырьмя периодами для тона в 300 герц и тремя периодами для 225 герц), так что результат повторяется 75 раз в секунду.

Частота тона в 225 герц в патче может быть изменена. Если она немного варьируется около 225, то в результате появляются биения в звуке. Другие значения находят другие общие субгармоники и по-прежнему приводят к богатым ангармоническим тонам.

5.5.4 Вейвшейпинг, использующий полиномы Чебышёва

Пример E05.chebychev.pd (рис. 5.12) демонстрирует, как вы можете использовать вейвшейпинг для генерирования чистых гармоник. Здесь мы ограничимся специфическим примером, в котором мы хотим генерировать пятую гармонику

$$\cos(5\omega n),$$

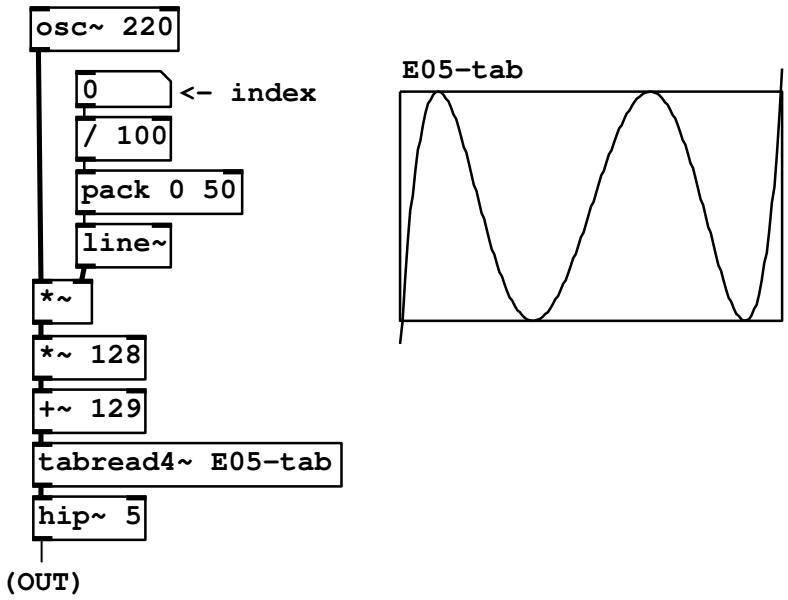


Рис. 5.12: Использование полиномов Чебышёва в качестве преобразующей функции для вейвшейпинга.

задавая форму волны синусоиды

$$x[n] = \cos(\omega n).$$

Нам нужно найти подходящую функцию преобразования $f(x)$. Во-первых, мы используем формулу для вейвшейпинга $f(x) = x^5$ (стр. 124), что даёт первую, третью и пятую гармоники:

$$16x^5 = \cos(5\omega n) + 5\cos(3\omega n) + 10\cos(\omega n).$$

Далее мы добавляем подходящее кратное x^3 для того, чтобы убрать третью гармонику:

$$16x^5 - 20x^3 = \cos(5\omega n) - 5\cos(\omega n)$$

и далее кратное x, для того чтобы убрать первую гармонику:

$$16x^5 - 20x^3 + 5x = \cos(5\omega n).$$

Таким образом, для нашей функции вейвшейпинга мы выбираем

$$f(x) = 16x^5 - 20x^3 + 5x.$$

Эта процедура позволяет нам выделить любую требуемую гармонику; получающиеся функции f известны как полиномы Чебышёва [Bru79].

Для включения этого в инструмент вейвшейпинга мы просто строим патч, который работает как на рис. 5.5, вычисляя выражение

$$x[n] = f(a[n]\cos(\omega n)),$$

где $a[n]$ — подходящий индекс, который может изменяться как функция номера семпла n. Когда а равняется единице по величине, наружу приходит пятая гармоника. Другие значения a дают различные спектры, которые, в общем, имеют первую и третью гармоники, так же как пятую.

Комбинируя подходящим образом полиномы Чебышёва, мы можем фиксировать любую требуемую суперпозицию компонент в выходной форме волны (опять же, пока индекс вейвшейпинга равен единице). Но к настоящей возможности метода вейвшейпинга, состоящему в том, что просто меняя индекс мы можем создавать спектры, которые меняются интересным, но управляемым образом, представление Чебышёва, по крайней мере напрямую, не относится.

5.5.5 Вейвшейпинг с помощью экспоненциальной функции

Вернемся опять к спектрам, вычисленным на стр. 124, отвечающим функциям вейвшейпинга вида $f(x) = x^k$. С радостью отметим, что все они не только синхронны по фазе (так что можно взять их суперпозицию с легко предсказуемыми результатами), но и расширяются при возрастании k. Также в ряде вида

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots ,$$

больший индекс модуляции будет давать больший соответствующий вес для членов более высокой степени в разложении; как мы видели раньше, если индекс модуляции есть a, то различные члены x^k умножаются на f_0, af_1, $a^2 f_2$ и так далее.

Теперь предположим, что мы хотим подобрать различные члены разложения выше так, чтобы предсказуемым образом управлять результатом как функцией индекса a. Для выбора наиболее простого возможного примера, предположим, мы хотим, чтобы f_0 было наибольшим членом для $0 < a < 1$, и далее, чтобы его догнал быстрее растущий член af_1 для $1 < a < 2$, который далее превосходится $a^2 f_2$ при $2 < a < 3$ и так далее, так что каждый n-ый член становится доминирующим при индексе n. Для того чтобы это случилось, мы лишь потребуем, чтобы

$$f_1 = f_0, 2f_2 = f_1, 3f_3 = f_2, \ldots$$

и, таким образом, выбирая $f_0 = 0$, получаем $f_1 = 1$, $f_2 = 1/2$, $f_3 = 1/6$ и, в общем,

$$f_k = \frac{1}{1 \cdot 2 \cdot 3 \cdot \ldots \cdot k}$$

Оказывается, что это коэффициенты разложения по возрастающим степеням функции

$$f_k = \frac{1}{1 \cdot 2 \cdot 3 \cdot \ldots \cdot k}$$

где $e \approx 2.7$ называется константой Эйлера.

Перед использованием e^x как функции преобразования лучше запланировать, что мы будем делать с амплитудой сигнала, поскольку e^x возрастает быстро, когда

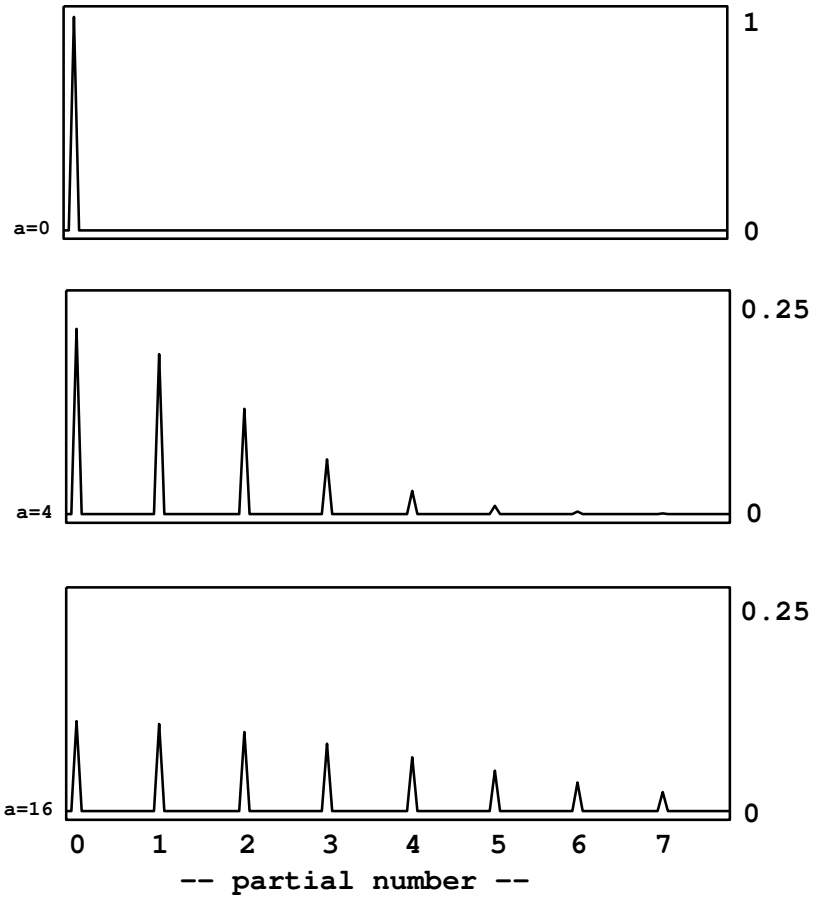


Рис. 5.13: Спектры вейвшейпера, использующего экспоненциальную функцию преобразования. Показаны индексы модуляции 0, 4 и 16; обратите внимание на различные масштабы по вертикали.

x растёт. Если мы собираемся послать на вход синусоиду амплитуды a, максимальный выход будет e^a, возникающий всегда, когда фаза равна нулю. Простой и естественный выбор состоит просто в делении на e^a для уменьшения пика до единичного, так что

$$f(a\cos(\omega n)) = \frac{e^{a\cos(\omega n)}}{e^a} = e^{a(\cos(\omega n)-1)}$$

Это реализуется в примере E06.exponential.pd. Результирующие спектры для $a = 0$, 4 и 16 показаны на рис. 5.13. Когда индекс вейвшейпинга растёт, всё меньшая энергия соответствует основной частоте; она перераспределяется по гармоникам.

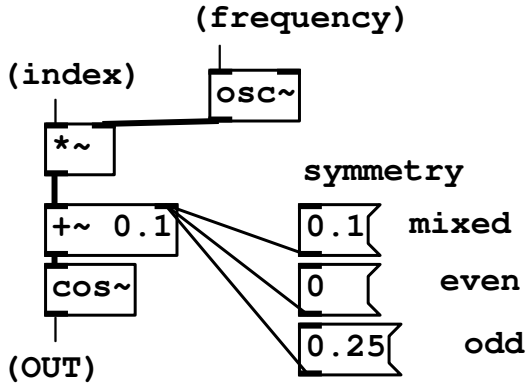


Рис. 5.14: Использование начального смещения для преобразующей функции косинус для изменения симметрии между чётной и нечётной составляющими. Без смещения симметрия чётная. Для нечётной симметрии к фазе добавляется четверть цикла. Меньшие смещения дают смесь чётной и нечётной симметрии.

5.5.6 Вейвшейпинг синусоидой: чётность и нечётность

Другой интересный класс преобразующих функций для вейвшейпинга состоит из синусоид

$$f(x) = \cos(x + \phi),$$

которые включают функции косинус и синус (выбором $\phi = 0$ и $\phi = -pi/2$, соответственно). Эти функции, одна чётная, а вторая нечётная, приводят к чётным и нечётным гармоническим спектрам, которые оказываются следующими:

$$\cos(a\cos(\omega n)) = J_0(a) - 2J_2(a)\cos(2\omega n) + 2J_4(a)\cos(4\omega n) - 2J_6(a)\cos(6\omega n) \pm \cdots$$

$$\sin(a\cos(\omega n)) = 2J_1(a)\cos(\omega n) - 2J_3(a)\cos(3\omega n) + 2J_5(a)\cos(5\omega n) \mp \cdots$$

Функции $J_k(a)$ называются *функциями Бесселя* первого рода; их используют иногда инженеры для решения проблем с вибрацией или тепловыми потоками на дисках. Для других значений ϕ мы можем разложить выражение для f:

$$f(x) = \cos(x)\cos(\phi) - \sin(x)\sin(\phi),$$

так что результат представляет из себя смесь чётных и нечётных гармоник, с ϕ, управляющей их относительной амплитудой. Это демонстрируется в Патче E07.evenodd.pd, показанном на рис. 5.14.

5.5.7 Фазовая модуляция и FM

Пример E08.phase.mod.pd, показанный на рис. 5.14, демонстрирует как использовать Pd для реализации правильной частоты модуляции (часть a) и фазовой

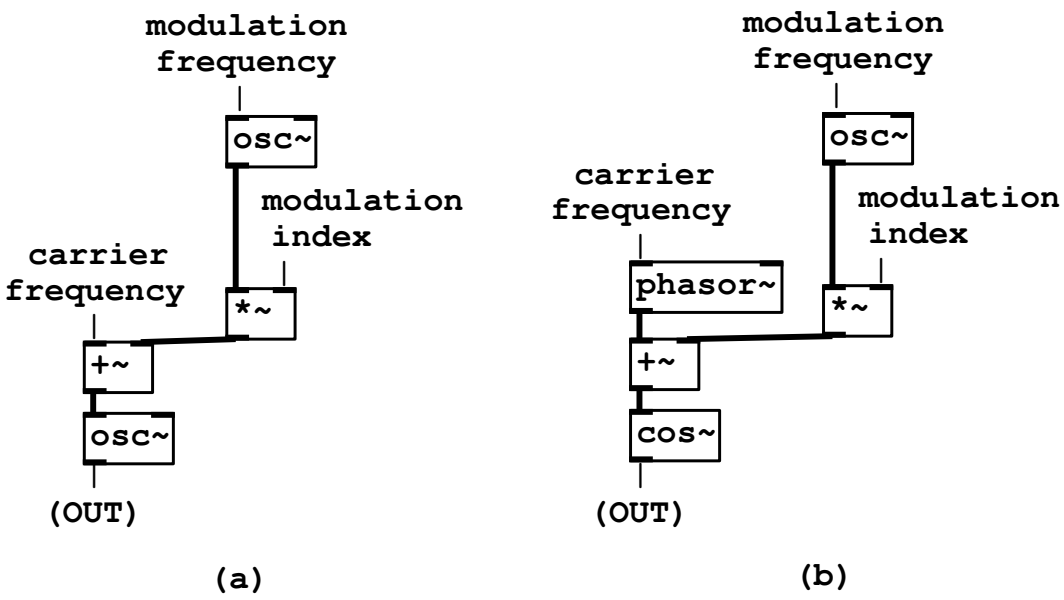


Рис. 5.15: Патчи Pd для: (a) частотной модуляции; (b) фазовой модуляции.

модуляции (часть b). Это отвечает блок-схемам на рис. 5.8. Для осуществления фазовой модуляции несущий осциллятор состоит из фазового компонента и функции сканирования косинуса. Сигнал имеет форму:

$$x[t] = \cos(\omega_c n + a\cos(\omega_m n)),$$

где ω_c есть несущая частота, ω_m — модулирующая частота, и a — индекс модуляции (все в угловых единицах).

Мы можем предсказать спектр, раскладывая внешний косинус:

$$x[t] = \cos(\omega_c n)\cos(a\cos(\omega_m n)) - \sin(\omega_c n)\sin(a\cos(\omega_m n)).$$

Подстановка в выражение на стр. 136 и упрощение выражения даёт:

$$x[t] = J_0(a)\cos(\omega_c n)$$

$$+J_1(a)\cos((\omega_c + \omega_m)n + \tfrac{\pi}{2}) + J_1(a)\cos((\omega_c - \omega_m)n + \tfrac{\pi}{2})$$

$$+J_2(a)\cos((\omega_c + 2\omega_m)n + \pi) + J_2(a)\cos((\omega_c - 2\omega_m)n + \pi)$$

$$+J_3(a)\cos((\omega_c + 3\omega_m)n + \tfrac{3\pi}{2}) + J_3(a)\cos((\omega_c - 3\omega_m)n + \tfrac{3\pi}{2}) + \cdots$$

Так что компоненты центрированы вокруг несущей частоты ω_c с боковыми полосами, расширяющимися в каждом из направлений и отделёнными на ω_m каждая от следующей. Амплитуды являются функциями индекса модуляции и не зависят от частот. Рисунок 5.16 показывает некоторые двухоператорные спектры фазовой модуляции, измеренные с использованием примера E09.FM.spectrum.pd.

Фазовая модуляция может рассматриваться просто как вид вейвшейпинга кольцевой модуляцией. Так мы можем использовать стратегии, описанные в §5.2, для генерации определённой комбинации частот. Например, если несущая частота является половиной модулирующей частоты, вы получаете звук с нечётными гармониками, в точности как в примере с делением октавы (рис. 5.10).

Частотная модуляция не нуждается в ограничении на чисто синусоидальные несущий и модулирующий осцилляторы. Один хорошо изученный путь состоит в воздействии фазовой модуляции на сам фазово-модулированный спектр. Тогда имеется два индекса модуляции (назовём их a и b) и две частоты модуляции (ω_m и ω_p), а формой волны будет:

$$x[n] = \cos(\omega_c n + a\cos(\omega_m n) + b\cos(\omega_p n)).$$

Для анализа результата просто перепишите первоначальный ряд FM выше, везде заменяя $\omega_c n$ на $\omega_c n + b\cos(\omega_p)$. Третья положительная боковая полоса становится, например:

$$J_3(a)\cos((\omega_c + 3\omega_m)n + \frac{3\pi}{2} + b\cos(\omega_p n)).$$

Это само по себе в точности другой спектр FM, со своими собственными боковыми полосами частот

$$\omega_c + 3\omega_m + k\omega_p, k = 0, \pm 1, \pm 2, \ldots,$$

имеющими амплитуды $J_3(a)J_k(a)$ и фазы $(3+k)\pi/2$ [Bru77]. Пример E10.complex.FM.pd (не показан здесь) иллюстрирует это с помощью построения спектров двухмодулирующего инструмента FM.

Начиная с ранних времен [Sch77], исследователи искали комбинации фаз, частот и индексов модуляции для простых и компактных инструментов фазовой модуляции, которые управляли бы имитацией привычных инструментальных звуков. Это стало основной индустрией с введением коммерческих синтезаторов FM.

Упражнения

1. Звук имеет основную частоту 440. Как осуществить кольцевую модуляцию, чтобы дать тон в 110 герц, в котором присутствуют только нечётные гармоники? Как бы вы далее могли получить чётные гармоники, если бы захотели?

2. Синусоида с частотой 400 и единичной пиковой амплитудой возводится в квадрат. Каковы амплитуды и частоты новых компонент сигнала?

3. Какую несущую и модулирующую частоты вы бы дали двухоператорному инструменту FM, чтобы выдать частоты 618, 1000 и 2618 герц? (Это основа пьесы «Stria» Джона Чоунинга [DJ85].)

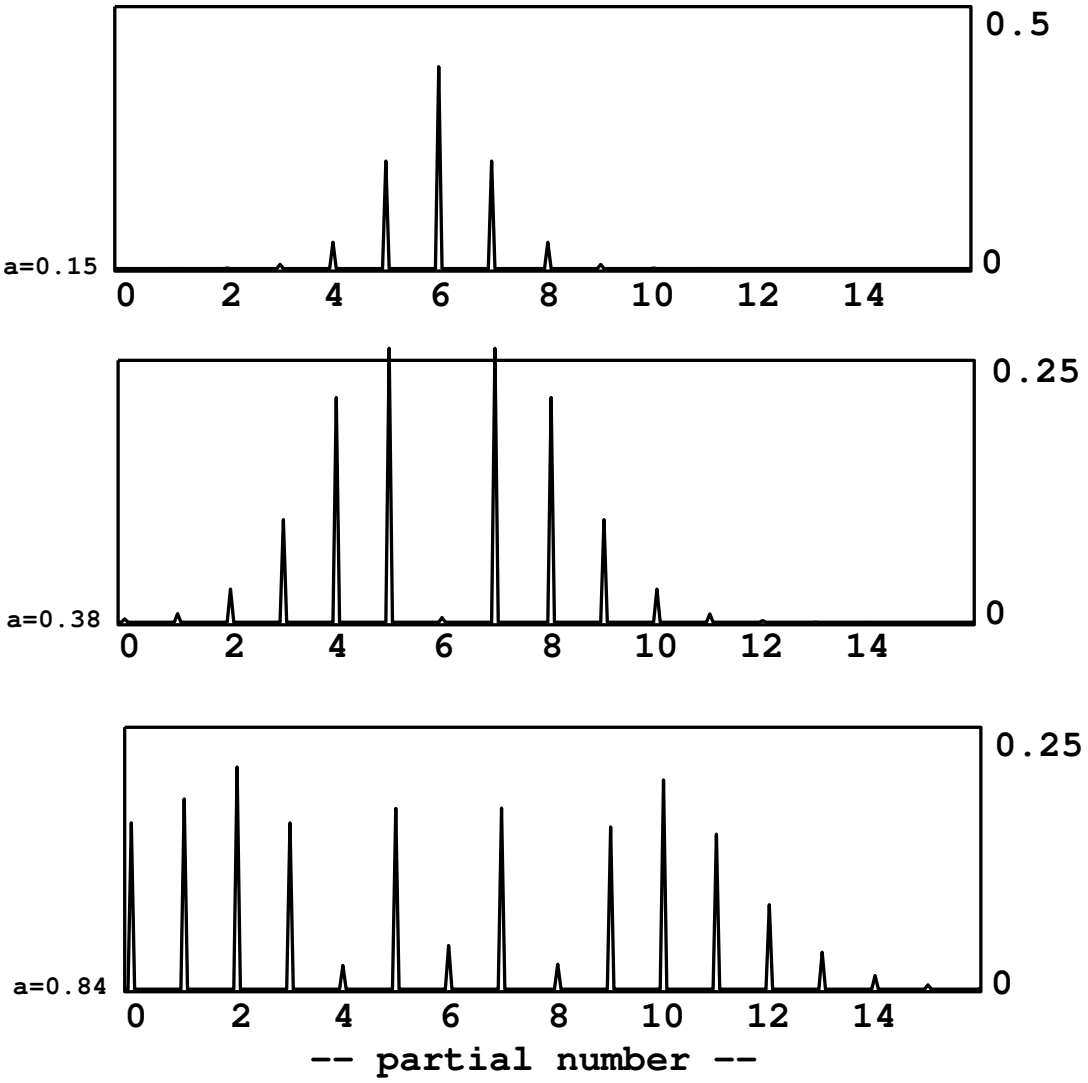


Рис. 5.16: Спектры фазовой модуляции с тремя различными индексами. Индексы даны в виде кратных 2π радианам.

4. Две синусоиды с частотами 300 и 400 герц и единичной пиковой амплитудой (так что среднеквадратичная амплитуда RMS ≈ 0.707) перемножаются. Какова амплитуда RMS произведения?

5. Предположим, что вы захотели создать FM еще более сложным образом, модулируя модулирующий осциллятор, так что

$$\cos(\omega_c n + a\cos(\omega_m n + b\cos(\omega_p n)))$$

   Как, качественно говоря, спектр отличался бы от того, который имеет простой двухмодуляторный пример (§5.5)?

6. Синусоида с частотой ω модулируется кольцевым образом другой синусоидой с точно такой же частотой. При каких фазовых разностях постоянная компонента DC результата исчезает?

Глава 6

Проектирование спектра

Как говорится в начале предыдущей главы, мощным методом синтеза музыкальных звуков является определение (и реализация, далее) специфических траекторий высоты (или, в более общем, частот гармоник), вместе с траекториями спектральной огибающей [Puc01]. Спектральная огибающая используется для определения амплитуды индивидуальных гармоник, как функции частот, и воспринимается как управление тембром звука (возможно, в зависимости от времени).

Простым примером такого подхода была бы имитация щипка струны с помощью построения звука с гармонически разделёнными обертонами, на которых спектральная огибающая насыщенна в начале, но затухает экспоненциально на высоких частотах, убывая быстрее, чем на нижних, так что тембр становится всё более устоявшимся со временем. Модели спектральной эволюции для различных акустических инструментов были предложены в [GM77] [RM69]. Более сложный пример даёт говорящий или поющий голос, в котором гласные появляются в виде спектральных огибающих, дифтонги и многие согласные появляются как временные вариации спектральных огибающих, а другие согласные появляются как спектрально оформленный шум.

Спектральные огибающие могут быть получены из анализа записанных звуков (который развит в Главе 9) или из чисто синтетических параметров. Определение спектральной огибающей с помощью отслеживания каждой возможной частоты — утомительно, и, в большинстве случаев, вам нужно её описать с помощью наиболее ярких особенностей. Наиболее популярный способ сделать это состоит в определении размера и формы пиков спектральной огибающей, которые называются формантами. Рисунок 6.1 показывает спектральную огибающую с двумя формантами. Хотя формы двух пиков спектральной огибающей различны, они оба могут быть грубо описаны заданием координаты каждой вершины (что даёт центральную частоту форманты и амплитуду) и ширины полосы каждой форманты. Типичной мерой ширины полосы была бы ширина пика на уровне в 3 децибела ниже его вершины. Заметим, что если пик расположен (или находится рядом) с осью $f = 0$, мы препятствуем ему спадать слева с той же скоростью, как (в действительности) он спадает справа.

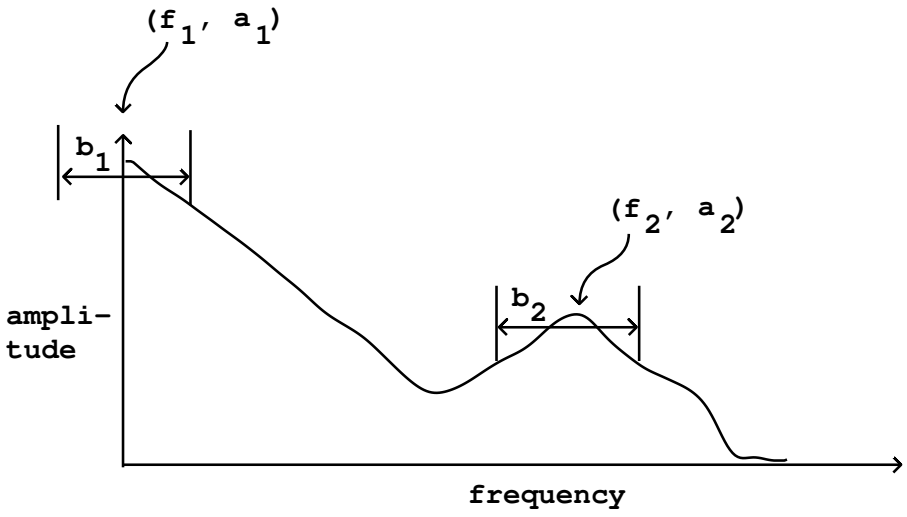


Рис. 6.1: Спектральная огибающая, на которой показаны частоты, амплитуды и ширина полос двух формант.

Предположим, что мы хотим генерировать гармонический звук с определённым набором формант. Независимо от желаемой основной частоты, мы хотим, чтобы спектр имел пики с предписанными центральными частотами, амплитудами и шириной полос. Возвращаясь к спектрам, полученным с помощью фазовой модуляции, которые показаны на рис. 5.16, мы видим, что по крайней мере при небольших индексах модуляции, в результате получается единственный хорошо определённый спектральный пик. Мы можем представить добавление нескольких таких пиков, все из которых повышают основную (модулирующую) частоту, но с несущими, настроенными на различные гармоники, для выбора различных желаемых центральных частот, и с индексами модуляции, выбранными так, чтобы дать результат с требуемой шириной полос. Это было впервые реализовано Чоунингом [Cho89], который организовал форманты, генерированные с помощью фазовой модуляции, для синтезирования поющих голосов. В этой главе мы установим общий подход к построению гармонических спектров с требуемыми, возможно зависящими от времени, формантами.

6.1 Модель несущая/модулятор

Ранее мы видели, как использовать кольцевую модуляцию для модифицирования спектра периодического сигнала, с помещением спектральных пиков в определённых местах (см. рисунок 5.4, стр. 119). Для того чтобы сделать это, нам нужно быть в состоянии генерировать периодические сигналы, чьи спектры имеют максимумы на DC и спадают монотонно с ростом частоты. Если мы можем сделать сигнал с формантой на нулевой частоте (и без других формант, кроме неё) мы можем использовать кольцевую модуляцию для перемещения форманты на любую желае-

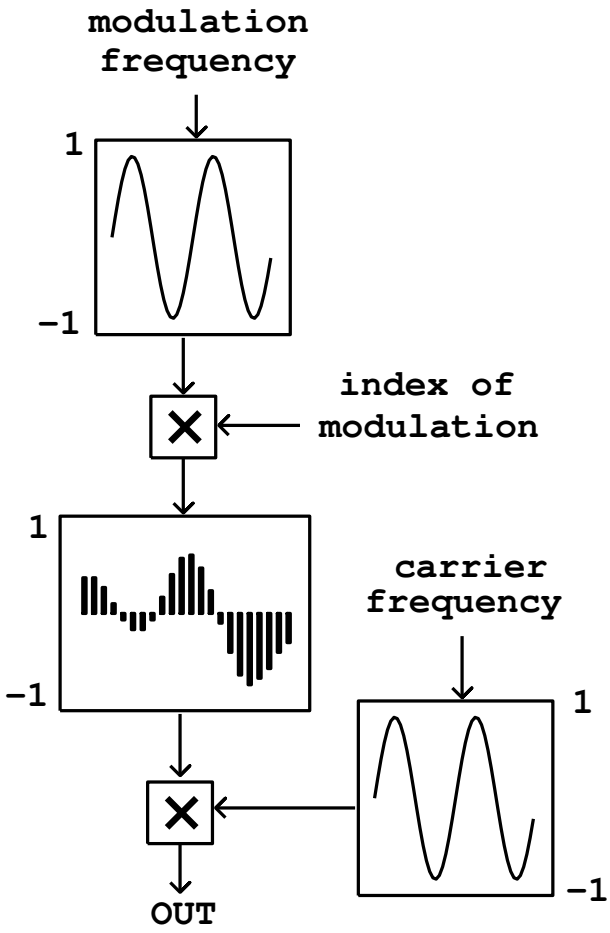


Рис. 6.2: Вейвшейпинг с помощью кольцевой модуляции для генерации форманты.

мую гармонику. Если мы используем вейвшейпинг для генерации первоначальной форманты, результат кольцевой модуляции будет иметь вид

$$x[n] = \cos(\omega_c n) f(a \cos(\omega_m n))$$

где ω_c (несущая частота) установлена на центральную частоту форманты и $f(a \cdot \cos(\omega_m n))$ есть сигнал с основной частотой, определённой ω_m, созданный с использованием функции вейвшейпинга f и индекса a. Этот второй член является сигналом, вид которого мы хотим придать форманте на DC с управляемой шириной полосы. Блок-схема синтеза этого сигнала показана на рис. 6.2.

Значительно раньше, в §2.4, при обсуждении синтеза волновых таблиц, мы определили технику *растягивания тембра*. Эта техника, которая даёт возможность генерации сложных, меняющихся тембров, может быть вложена в тот же контекст. Выход волновой таблицы с огибающей за один цикл есть:

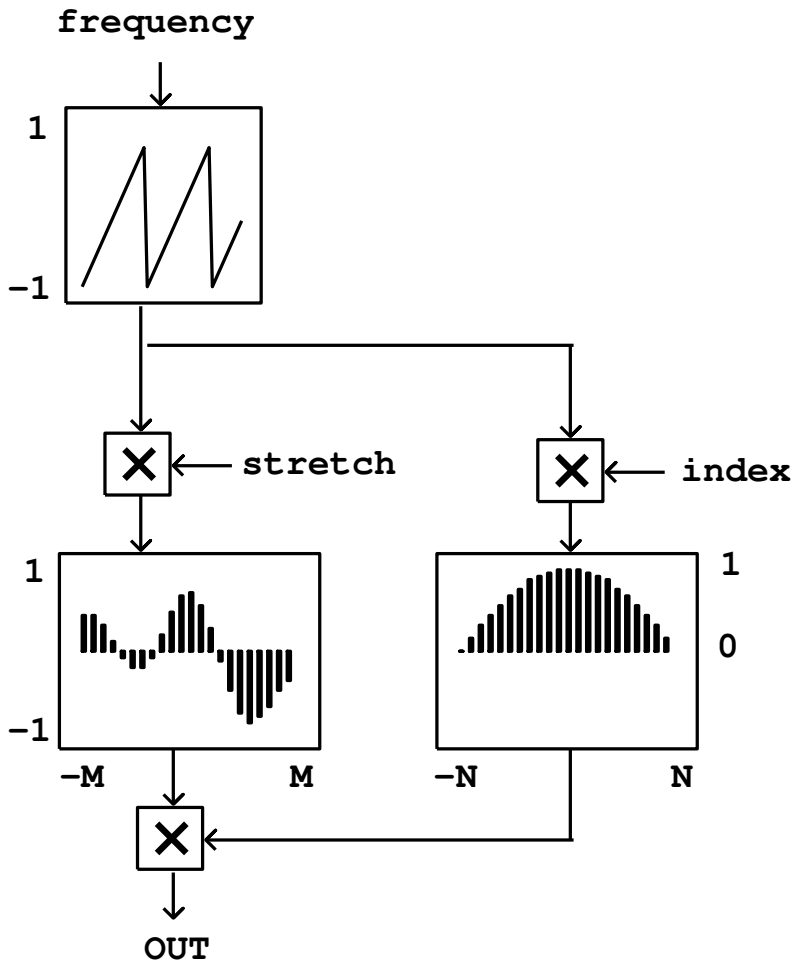


Рис. 6.3: Синтез волновой таблицы, обобщённый до генератора переменного спектра.

$$x(\phi) = T(c\phi) * W(a\phi),$$

где ϕ, фаза, такова, что $-\pi \leqslant \phi \leqslant \pi$. Здесь T есть функция, сохраняемая в волновой таблице, W — функция, организующая окно, а c и a — индексы растягивания волновой таблицы и модуляции для функции, организующей окно. рис. 6.3 показывает, как реализовать это в форме блок-схемы. Сравнивая его с рис. 2.7, мы видим, что единственная важная новая черта — добавление индекса a.

В этой установке, как и в предыдущей, первый член определяет размещение энергии в спектре (в этом случае, с параметром c, действующем для растягивания спектра волновой таблицы). Ранее эту роль выполнял выбор несущей частоты кольцевой модуляции ω_c.

И то и другое (вейвшейпинг с помощью кольцевой модуляции и синтез растянутой волновой таблицы) может рассматриваться как частные случаи более общего подхода, заключающегося в вычислении функций вида

$$x[n] = c(\omega n) m_a(\omega n)$$

где c есть периодическая функция, описывающая несущий сигнал и m_a — функция периодического модулятора, которая зависит от индекса a. Функции модуляции, которые нам интересны, будут обыкновенно иметь вид последовательности импульсов, а индекс a будет управлять шириной импульса; большие значения a дадут более узкие импульсы. В случае волновой таблицы, модулирующая функция должна достигать нуля в точках свёртывания фазы для избежания любых разрывов несущей функции в моменты времени, когда фаза свёртывается. Несущий сигнал будет приводить к единственному спектральному пику (форманте) в случае вейвшейпинга кольцевой модуляцией; для волновых таблиц, спектр может быть более сложным.

В следующем параграфе мы будем дальше развивать две формы модуляции сигнала, которые мы определили здесь, и, далее, рассмотрим более близко несущий сигнал.

6.2 Последовательности импульсов

Последовательности импульсов могут быть генерированы или с использованием формулы вейвшейпинга или растяжением волновой таблицы. Формулы вейвшейпинга проще для анализа и управления и мы рассмотрим её в начале.

6.2.1 Последовательности импульсов через вейвшейпинг

Когда мы используем вейвшейпинг, вид форманты определяется модуляционным членом

$$m_a[n] = f(a\cos(\omega n)).$$

Для маленьких значений индекса a, модуляционный член лишь немного варьируется около постоянного значения $f(0)$, так что большая часть энергии сконцентрирована на DC. Когда a растёт, энергия распределяется среди всё более высоких гармоник основной частоты ω. В зависимости от функции f, это распределение может быть упорядочено или не упорядочено. Упорядоченное распределение может быть востребовано, а, далее, опять не востребовано, в зависимости от того, является ли нашей целью предсказуемый спектр или широкий диапазон различных (и, возможно, трудно предсказуемых) спектров.

Преобразующая функция $f(x) = e^x$, проанализированная на стр. 134, выдаёт стабильные, простые и предсказуемые результаты. После подходящей нормализации мы получаем спектры, показанные на рис. 5.13. Небольшое видоизменение модулятора вейвшейпинга для такого выбора f (принимая во внимание перенормировку)

даёт:

$$m_a[n] = e^{a\cdot(\cos(\omega n)-1))}$$

$$= e^{-\left[b\sin\frac{\omega}{2}\right]^2},$$

где $b^2 = 2a$, так что b пропорционально ширине полосы. Это может быть переписано как

$$m_a[n] = g(b\sin\frac{\omega}{2}n)$$

с

$$m_a[n] = g(b\sin\frac{\omega}{2}n).$$

За исключением опущенного нормировочного множителя, это гауссово распределение, иногда называемое “колоколообразной кривой”. Амплитуды гармоник даются функциями Бесселя первого рода.

Другой прекрасный выбор это (опять ненормированное) распределение Коши:

$$h(x) = \frac{1}{1+x^2},$$

что приводит к спектру с экспоненциально спадающими гармониками:

$$h(b\sin(\frac{\omega}{2}n)) = G\cdot\left(\frac{1}{2} + H\cos(\omega n) + H^2\cos(2\omega n) + \cdots\right),$$

где G и H — функции индекса b (точные формулы даны в [Puc95a]).

В этом и гауссовом случае выше, ширина полосы (рассчитанная в пиках, то есть единицах ω) грубо говоря пропорциональна $1/(1+b)$. Для любой функции вейвшейпинга (g или h), если b больше, чем примерно 2, форма волны $m_a(\omega n)$ является приблизительным (вперёд или назад) сканом преобразующей функции, так что получающаяся форма волны выглядит, как импульсы, чья ширина убывает, когда заданная ширина полосы растёт.

6.2.2 Последовательности импульсов через растягивание волновой таблицы

В формуле волновой таблицы, последовательность импульсов может быть сделана растяжением волновой таблицы:

$$M_a(\phi) = W(a\phi),$$

где $-\pi \leqslant \phi \leqslant \pi$ есть фаза, то есть величина ωn свёртывается на промежуток между $-\pi$ и π. Функция W должна быть нулём в граничных точках и вне отрезка от $-\pi$ до π, а также расти до максимума в 0. Возможным выбором функции W будет

$$W(\phi) = \frac{1}{2}\left(\cos(\phi)+1\right).$$

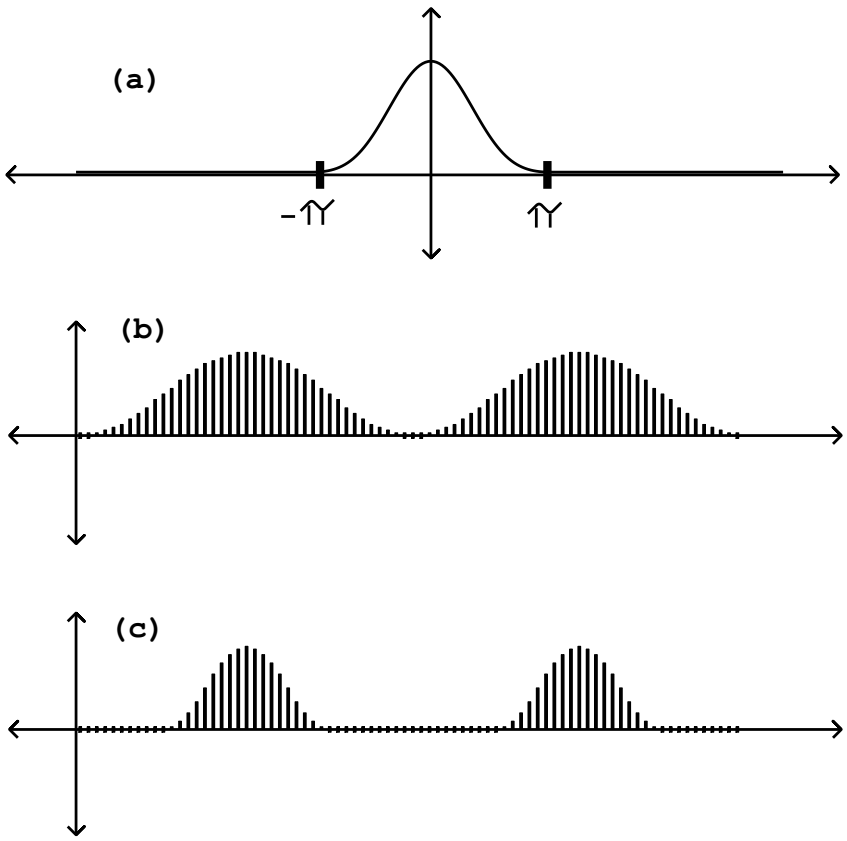


Рис. 6.4: Модуляция ширины импульса, использующая функцию окна фон Ханна: (a) функция $W(\phi) = (1+\cos(\phi))/2$; (b) функция как форма волны, повторяющейся с рабочим циклом 100% (индекс модуляции $a = 1$); (c) форма волны с 50% рабочим циклом ($a = 2$).

Эта функция графически изображена в части (a) рис. 6.4. Она известна как функция *окна Ханна*; она появится опять в Главе 9.

Реализуя её в виде повторяющейся формы волны, мы получаем последовательность (подходящим образом семплированных) копий функции W, чей рабочий цикл есть $1/a$ (части b и c рисунка). Если вы не хотите, чтобы копии перекрывались, индекс a должен быть, по крайней мере, 1. Если вы хотите позволить перекрывание, простейшей стратегией будет дублирование блок-схемы (рис. 6.3) без фазы, как описано в §2.4 и реализовано в §2.6.

6.2.3 Результирующие спектры

Перед рассмотрением более сложных несущих сигналов и продолжением работы с модуляторами, которые мы рассматривали до сих пор, полезно заметить, что умножение на чистую синусоиду задаёт нам как формы волн, так и спектры. рис. 6.5 показывает результат умножения двух разных последовательностей импульсов на синусоиду шестой гармоники:

$$\cos(6\omega n)M_a(\omega n),$$

где индекс модуляции a равен двум в обоих случаях. В части (a) M_a — растянутая функция окна Ханна; часть (b) показывает вейвшейпинг через ненормированное распределение Коши. Показан один период для каждой формы волны.

В обоих случаях мы видим, в результате, шестую гармонику (несущую), обёрнутую в *волновой пакет*, центрированную по середине цикла, где фаза синусоиды равна нулю. Изменение частоты синусоиды меняет центральную частоту форманты; изменение ширины пакета (часть формы волны, в течение которой синусоида велика) меняет ширину полосы. Заметим, что растянутая функция окна Ханна равна нулю в начале и конце периода, в отличие от задающего форму волны пакета.

Рисунок 6.6 показывает, как форма форманты зависит от метода её получения. Форма растянутой волновой таблицы (часть (a) рисунка) ведёт себя хорошо в окрестности пика, но несколько странно, начиная с расстояния четвёртой гармоники от пика, после чего мы видим то, что называется боковыми лепестками: ложные дополнительные пики с более низкой амплитудой, чем центральный пик. Как предсказывает анализ §2.4, целая форманта, боковые лепестки и прочее, растягивается или наоборот сжимается, когда последовательность импульсов сжимается или растягивается по времени.

Первые, самые сильные боковые лепестки на каждой из сторон приблизительно на 32 дб ниже по амплитуде, чем основной пик. Другие боковые лепестки спадают медленно, будучи выраженными в децибелах; амплитуды спадают как квадрат расстояния от центрального пика, так что шестой боковой лепесток справа, в три раза дальше чем первый от центральной частоты, ниже на около 20 дб. Эффект этих боковых лепестков часто слышим как лёгкое жужжание в звуке.

Форма такой форманты может быть сделана произвольно плоской (то есть большой ширины), но имеется предел, на сколько узкой она может быть сделана,

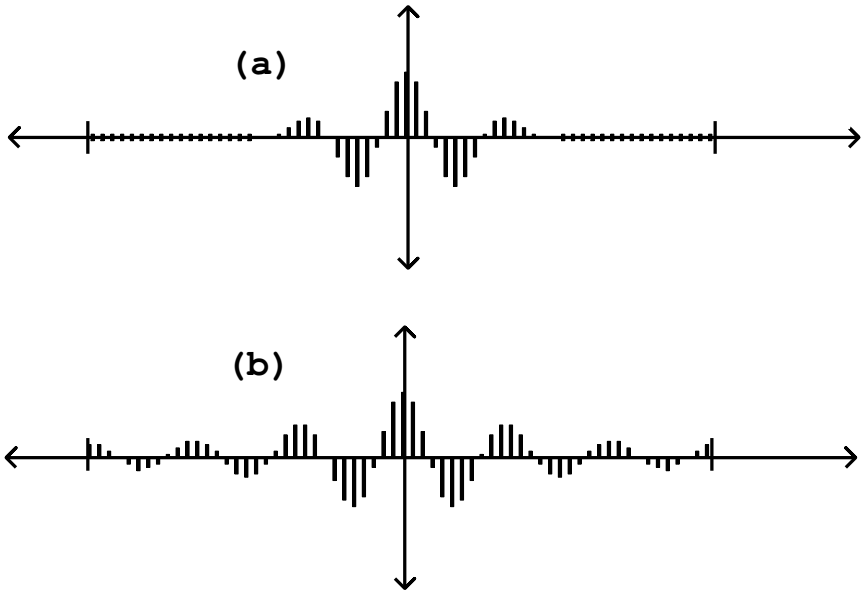


Рис. 6.5: Аудиосигналы, получающиеся от умножения косинуса (номер гармоники 6) на последовательность импульсов: (a) с функциями окна в формализации с помощью волновой таблицы; (b) выход вейвшейпинга, использующего функцию просмотра волновой таблицы Коши.

поскольку рабочий цикл формы волны не может превышать 100%. С максимальным рабочим циклом интенсивность форманты падает до нуля на расстоянии двух гармоник от центрального пика. Если всё ещё требуется более узкая ширина полосы, формы волн могут быть сделаны с перекрыванием, как описано в §2.6.

Части (b) и (c) рисунка показывают форманты, генерированные с использованием вейвшейпинга кольцевой модуляцией, с функциями преобразования Гаусса или Коши. Индекс модуляции равен двум в обоих случаях (так же как для окна Ханна части a) и ширина полосы сравнима с примером Ханна. В этих примерах нет боковых лепестков и, более того, индекс модуляции может вообще упасть до нуля, давая чистую синусоиду; нет нижнего предела на ширину полосы. С другой стороны, поскольку форма волны не достигает нуля в концах цикла, этот тип пульсовой последовательности не может быть использован как окно для произвольной волновой таблицы, как в случае импульсной последовательности Ханна.

Пример Коши особенно удобен для конструирования спектров, поскольку форма форманты является правильным равнобедренным треугольником, при изображении в децибелах. С другой стороны, в гауссовом примере в форманте скапливается больше энергии и на хвостах спад происходит быстрее, так что получается более чистый звук и обеспечивается лучшая защита от свёртывания.

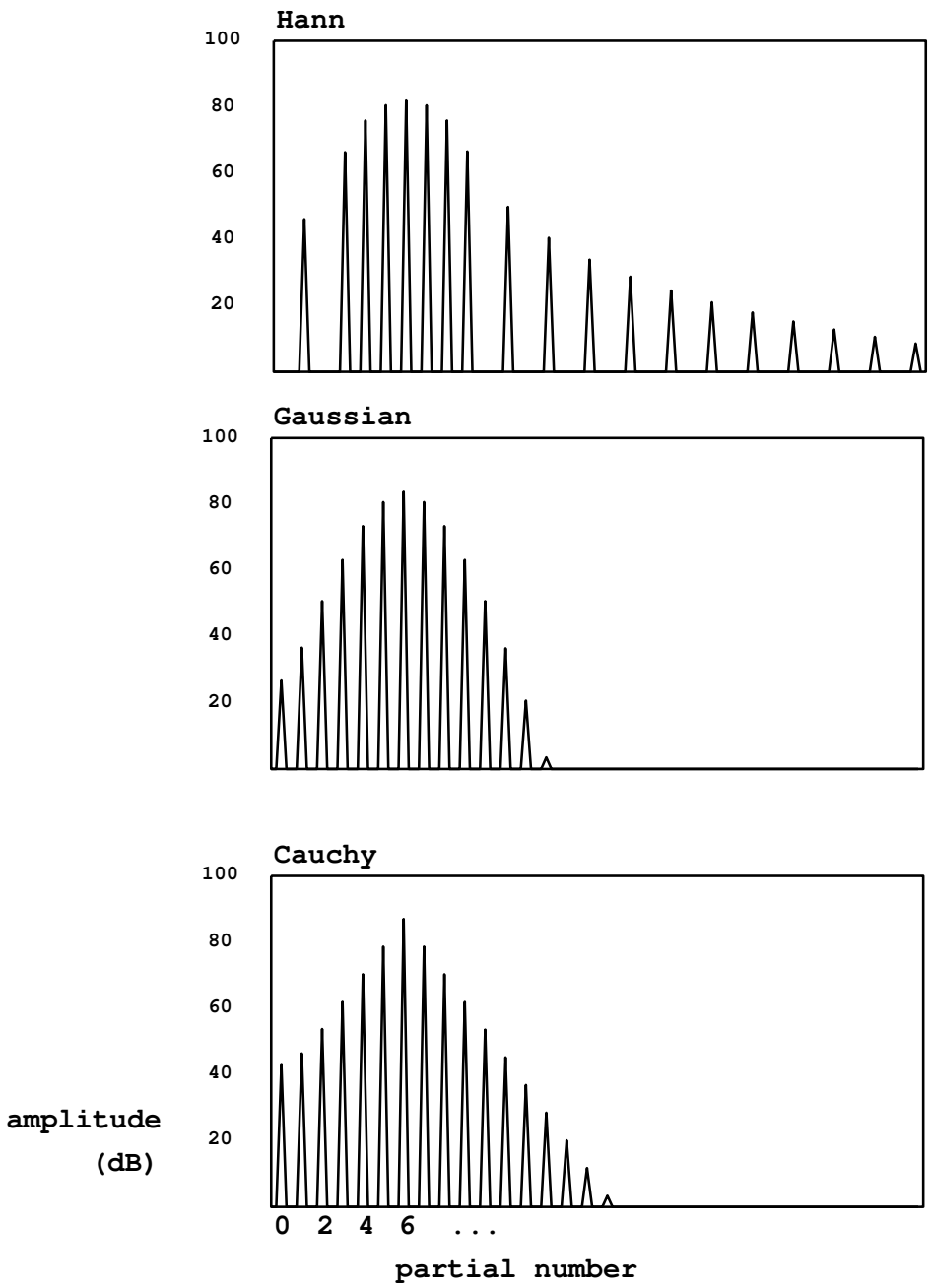


Рис. 6.6: Спектры трёх последовательностей импульсов, полученных с помощью кольцевой модуляции: (a) функция окна фон Ханна, 50% рабочий цикл (соответствующий индексу 2); (b) последовательность импульсов с вейвшейпингом гауссовской преобразующей функцией; (c) то же, но с преобразующей функцией Коши. Амплитуды даны в децибелах.

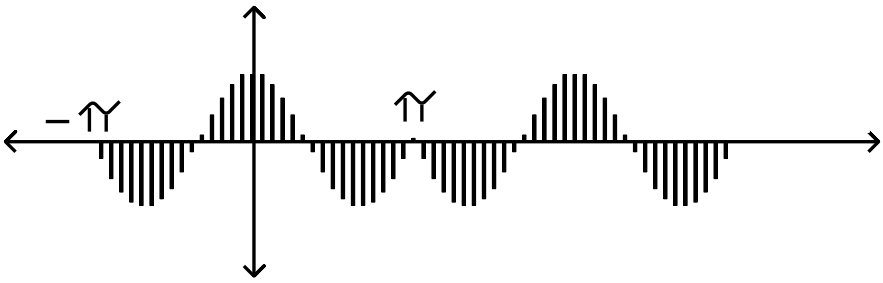


Рис. 6.7: Форма волны для несущего сигнала, построенного на основе волновой таблицы, для получения форманты в полтора раза большей основной частоты. Показаны два периода.

6.3 Подвижная кольцевая модуляция

Мы переходим теперь к несущему сигналу, в поисках путей, позволяющих сделать его более управляемым. Мы бы, в частности, хотели иметь возможность непрерывно перемещать спектральную энергию вверх и вниз по частотам. Простое изменение частоты несущего осциллятора не будет осуществлять это, поскольку спектры не будут гармоническими, за исключением случая, когда несущая частота является целым кратным основной частоты.

В подходе с растяжением волновой таблицы, мы можем осуществить это просто семплируя синусоиду и транспонируя её до требуемой "высоты". Транспонированная высота не слышна как периодичная, поскольку волновая таблица сама по себе считывается периодически с основной частотой. Вместо этого, синусоида транспонируется как спектральная огибающая.

Рисунок 6.7 показывает несущий сигнал, произведённый этим путём, настроенный для получения форманты, центрированной на величине, в полтора раза большей основной частоты. Сигнал не имеет чёткого разрыва на частоте свёртывания фазы, но он имеет разрыв на скате, который, если он не убран умножением на подходящий модулирующий сигнал, будет иметь очень слышимые высокочастотные компоненты.

Используя эту идею, мы можем сделать полное описание того, как использовать блок-схему рис. 6.3 для создания желаемой форманты. Просмотр волновой таблицы слева поддерживал бы синусоиду (помещённую симметрично, так что фаза равна нулю в центре волновой таблицы). Волновая таблица справа поддерживала бы функцию окна Ханна или другую подходящую. Если мы хотим, чтобы основная частота была ω, центральная частота форманты ω_c, и ширина полосы ω_b, мы устанавливаем параметр "растяжения" на *относительную центральную частоту*, определённую как ω_c/ω, а индекс модуляции на *относительную ширину полосы*, ω_b/ω.

Выходной сигнал есть просто семпл волны косинуса с требуемой центральной частотой, повторяемый с (независимым, вообще говоря) требуемым периодом и обработанный окном, чтобы убрать разрывы на границах периода.

Хотя мы не можем ещё вывести этот результат (нам будет нужен анализ Фурье), окажется, что в главном лепестке форманты фазы все равны нулю в центре формы волны (то есть, компоненты все являются косинусами, если мы рассматриваем нулевую фазу в центре волновой формы). Это означает, что мы можем взять суперпозицию любого числа этих формант для построения более сложного спектра и амплитуды гармоник будут объединяться путём сложения. (Боковые лепестки не ведут себя так хорошо: они имеют противоположные знаки и будут производить взаимно уничтожающиеся куски; но мы можем часто просто не принимать их во внимание, как малый, неуправляемый остаточный сигнал.)

Этот метод приводит к интересному обобщению, состоящему в том, чтобы взять последовательность записанных волновых таблиц, разместить все их составляющие фазы в одну линию с фазами косинусов и использовать их вместо функции косинуса как несущий сигнал. Выстраивание фаз в линию необходимо для того, чтобы позволить связный плавный переход между семплами, так чтобы спектральная огибающая менялась плавно. Если, например, мы используем последовательные фрагменты вокального семпла на входе, мы получаем замечательно эффективный вокодер; см. §9.6.

Другой метод создания несущих сигналов, которые могут непрерывно скользить вверх и вниз по частоте, при поддерживании основной частоты постоянной, состоит просто в плавном микшировании гармоник. Несущий сигнал тогда:

$$c(\phi) = c(\omega n) = p\cos(k\omega n) + q\cos((k+1)\omega n)$$

где $p + q = 1$ и k целое, все три числа выбраны так, что

$$(k + q) * \omega = \omega_c$$

и, таким образом, спектральный центр масс двух косинусов попадает на ω_c. (Заметим, что мы делаем амплитуду суммы двух складываемых косинусов равной единице, а не устанавливаем единичную общую энергию; мы делаем это, поскольку модулятор будет действовать на них с согласованными фазами.) Для осуществления этого мы просто устанавливаем k и q равными целой и дробной части, соответственно, относительной центральной частоты ω_c/ω.

Простейший путь создания управляющего интерфейса для техники синтеза состоял бы в использовании линейных функций для обновления ω и ω_c и, далее, вычисления q и k как аудиосигналов от линейно изменённых, плавно варьирующихся ω и ω_c. Достаточно странно, кроме того факта, что k, p и q — прерывистые функции от ω_c/ω, что несущая $c(\phi)$ оказывается варьирующейся непрерывно относительно ω_c/ω и, таким образом, если требуемая центральная частота ω_c линейно меняется от значения к значению, результатом является непрерывное смещение центральной частоты. Однако необходима большая работа, если требуются прерывистые

изменения центральной частоты. Такое желание не является неразумным, являясь аналогом прерывистого изменения частоты осциллятора.

Оказывается, имеется хороший способ осуществления этого. Уловка при обновлении k и q состоит в том, чтобы заметить, что $c(\phi) = 1$ тогда, когда ϕ кратно 2π, безотносительно к выбору k, p и q, при $p + q = 1$. Следовательно, мы можем делать прерывистые изменения k, p и q один раз за период (справедливо, когда фаза кратна 2π), без создания разрывов в несущем сигнале.

В специфическом случае FM, при желании, мы можем теперь вернуться и модифицировать первоначальную формулу на

$$p\cos(n\omega_2 t + r\cos(\omega_1 t)) +$$

$$+q\cos((n+1)\omega_2 t + r\cos(\omega_1 t))$$

Это позволяет нам добавить глиссандо (которое прослушивается как дифтонги) к оригинальной технике вокального синтеза Чоунинга, основанной на фазовой модуляции.

6.4 Генератор фазово-выровненной форманты PAF

Объединение двух-косинусного несущего сигнала с генератором последовательности импульсов, использующим процедуру вейвшейпинга, даёт *генератор фазово-выровненной форманты*, обыкновенно называемый PAF[1]. Общая формула даётся выражением:

$$x[n] = \underbrace{g\left(a\sin(\omega n/2)\right)}_{\text{модулятор}} \underbrace{\left[p\cos(k\omega n) + q\cos((k+1)\omega n)\right]}_{\text{несущая}}$$

Здесь функция вейвшейпинга g может быть или гауссовой (колоколообразной) или функцией Коши, ω — основная частота, a — индекс модуляции, управляющий шириной полосы, k, p и q управляют центральной частотой форманты.

Рисунок 6.8 показывает PAF в виде блок-схемы, разделённой на этапы генерации фазы, несущей и модулятора. Этап генерации фазы выводит пилообразный сигнал с основной частотой. Модулятор реализован при помощи стандартного вейвшейпинга с небольшой хитростью. Для формулы сигналов модулятора в PAF нужна входная синусоида с половиной основной частоты, то есть $\sin(\omega/2)$, и это номинально потребовало бы от нас использование пилообразного сигнала, настроенного на половину основной частоты. Однако поскольку функция вейвшейпинга волны чётная, мы можем подставить абсолютное значение синусоиды

$$\left|\sin(\frac{\omega}{2})\right|,$$

[1]PAF является предметом патента 1994 года, собственником которого является IRCAM.

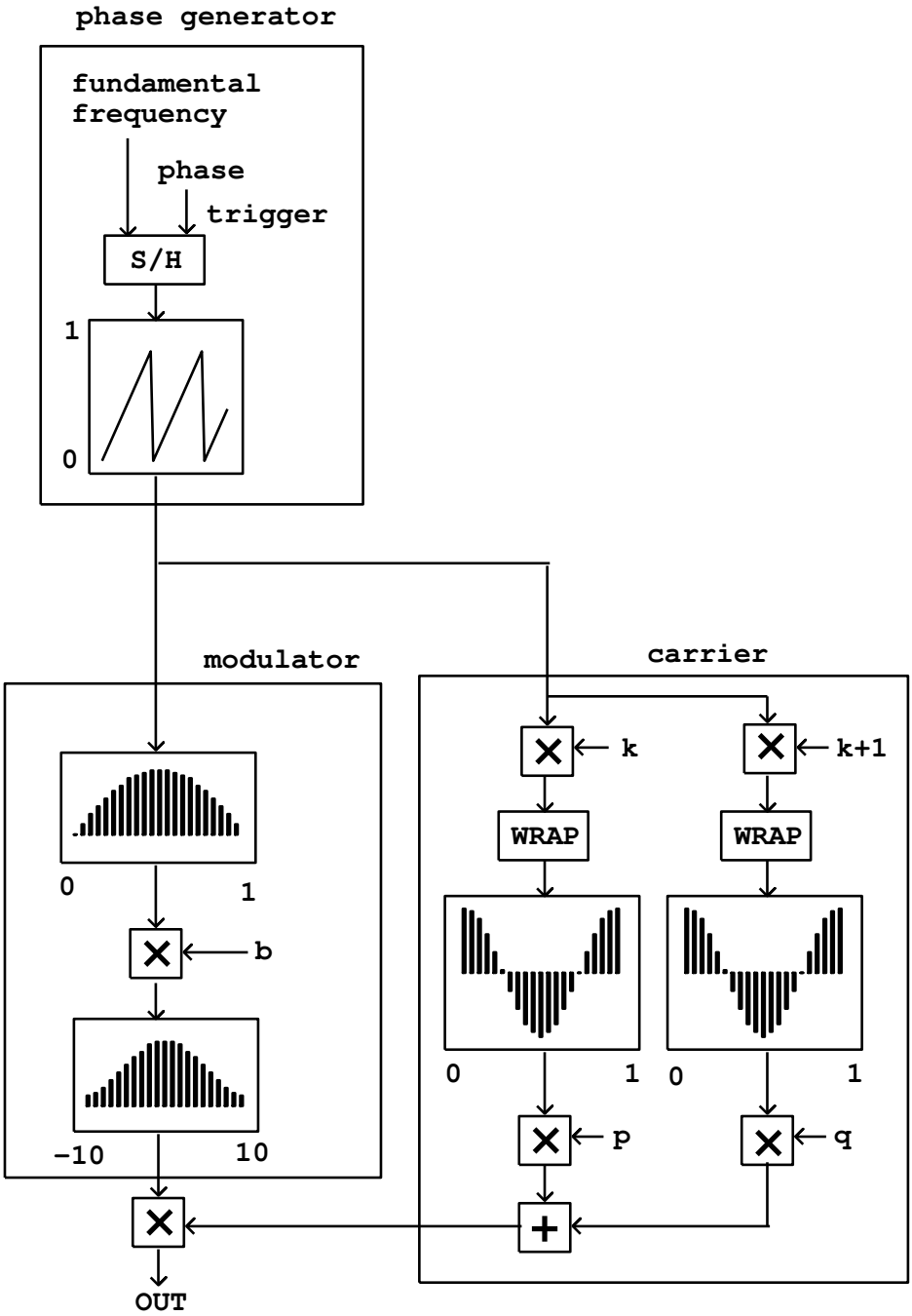


Рис. 6.8: Генератор PAF в виде блок-схемы.

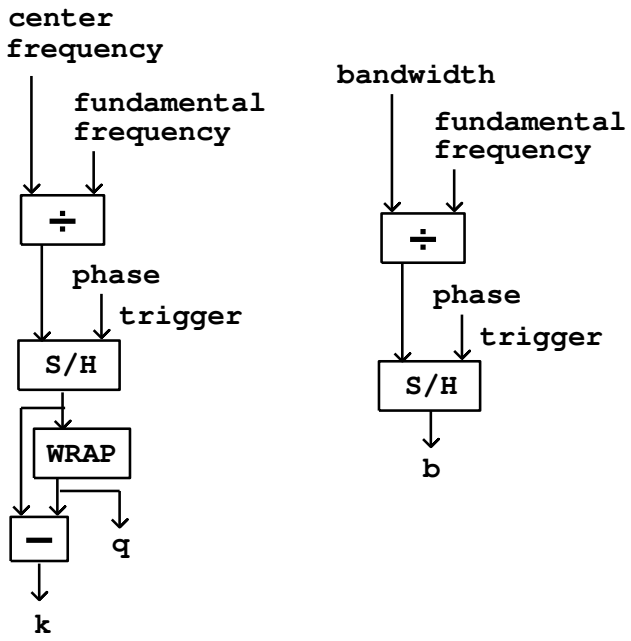


Рис. 6.9: Вычисление зависящих от времени параметров a (индекс вейвшейпинга), k и q для использования в блок схеме рис. 6.8.

которое повторяется с частотой ω (первая половина цикла совпадает со второй). Мы можем вычислить это значение просто используя половину цикла синусоиды как функцию просмотра волновой таблицы (с фазой, пробегающей от $-\pi/2$ до $\pi/2$), что даст необходимую синусоиду, которую мы направляем на функцию вейвшейпинга.

Хотя функция волновой таблицы отображает как отрицательные, так и положительные значения (пробегающие от -10 до 10), в действительности мы лишь используем положительную часть для просмотра, пробегаемую от 0 до b, индекса модуляции. Если индекс модуляции превышает входной диапазон таблицы (здесь, например, установленный на 10), индекс при просмотре волновой таблицы должен быть ограничен. Таблица должна простираться вплоть до окончания функции вейвшейпинга, так чтобы эффект ограничения не был слышен.

Несущий сигнал представляет из себя взвешенную сумму двух косинусов, чьи частоты увеличиваются умножением (на k и $k+1$, соответственно) и свёртываются. Тем самым, все фазы при просмотре управляются одним и тем же пилообразным осциллятором.

Величины k, q и индекс волновой таблицы b вычисляются, как показано на рис. 6.9. Они являются функциями определённой основной частоты, центральной частоты форманты и ширины полосы, то есть первоначальных параметров алгоритма. Величина p, не показанная на рисунке, равна, в точности, $1-q$.

Как описано в предыдущем параграфе, величины k, p и q должны только меняться в точках со свёрнутой фазой, то есть, с периодами $2\pi/\omega$. Поскольку вычисление k и др. зависит от величины параметра ω, следует, что ω сама по себе

должна меняться только, когда фаза кратна 2π; в противном случае, изменение ω могло бы установить центральную частоту $(k+q)\omega$ на некорректное значение для (очень слышимой) дробной части периода. В действительности все вычисления параметров должны быть синхронизованы с фазой первоначального осциллятора.

Изменение собственной частоты осциллятора с использованием управления его фазой является примером обратной связи, которая, в общем, означает использование какого-либо выхода процесса в качестве одного из его входов. При обработке цифровых аудиосигналов с фиксированной частотой семплирования (как мы делаем), никогда нет возможности получить текущий выход процесса в качестве входного сигнала, поскольку мы ещё не вычислим его в нужный момент времени. Лучшее, на что мы можем надеяться, это использование предыдущего семпла выхода (в действительности добавляя один семпл задержки). В средах где используются аудиовычисления блоками (таких как Max/MSP, Pd, Csound) ситуация становится более сложной, но мы отложим обсуждение этого до следующей главы (и просто хотим опустить эту проблему в примерах в конце главы).

Амплитуда центрального пика в спектре генератора PAF приблизительно равна $1/(1+b)$; другими словами, близка к единице, когда индекс b меньше единицы, и наоборот падает при больших значениях b. Для величин b, меньших чем десять, громкость выхода не варьируется сильно, поскольку ввод других гармоник, чётных или нечётных, начинает уменьшение амплитуды центральной гармоники. Однако если использовать генератор PAF для генерации формант с определёнными пиковыми амплитудами, выход должен умножаться на $1+b$ (или даже, если необходимо, на более хорошую аппроксимацию корректирующего множителя, чьё точное значение зависит от функции вейвшейпинга). Эта коррекция амплитуды должна осуществляться линейным изменением, а не методом “выборки и хранения”.

Поскольку разложение сигнала вейвшейпера (модулятора) в сумму состоит только из косинусов (то есть все они имеют нулевую начальную фазу), также как у двух компонент несущей, из формулы произведения косинусов следует, что компоненты результата также все являются косинусами. Это означает, что любое число генераторов PAF, если они сделаны разделяющими один и тот же осциллятор для генерации фаз, будет находится в фазе и их объединение даёт сумму индивидуальных спектров. Таким образом, мы можем сделать мульти-формантную версию, как показано на рис. 6.10.

Рисунок 6.11 показывает возможный результат для пары формант, генерированных таким способом; первая форманта центрирована посередине между 3 и 4 гармониками, вторая на 12 гармонике, с более низкой амплитудой и шириной полосы. Использована функция вейвшейпинга Коши, что производит линейно спадающие спектры (показанные в децибелах). Суперпозиция двух формант представляет из себя простое сложение, так что спектральная огибающая гладко огибает форманты от одной к другой. Наиболее низкая форманта также складывается со своим отражением относительно вертикальной оси, так что оказывается слегка искривлённой вверх в этом месте.

Генератор PAF может быть изменён, если, для создания ангармонических спек-

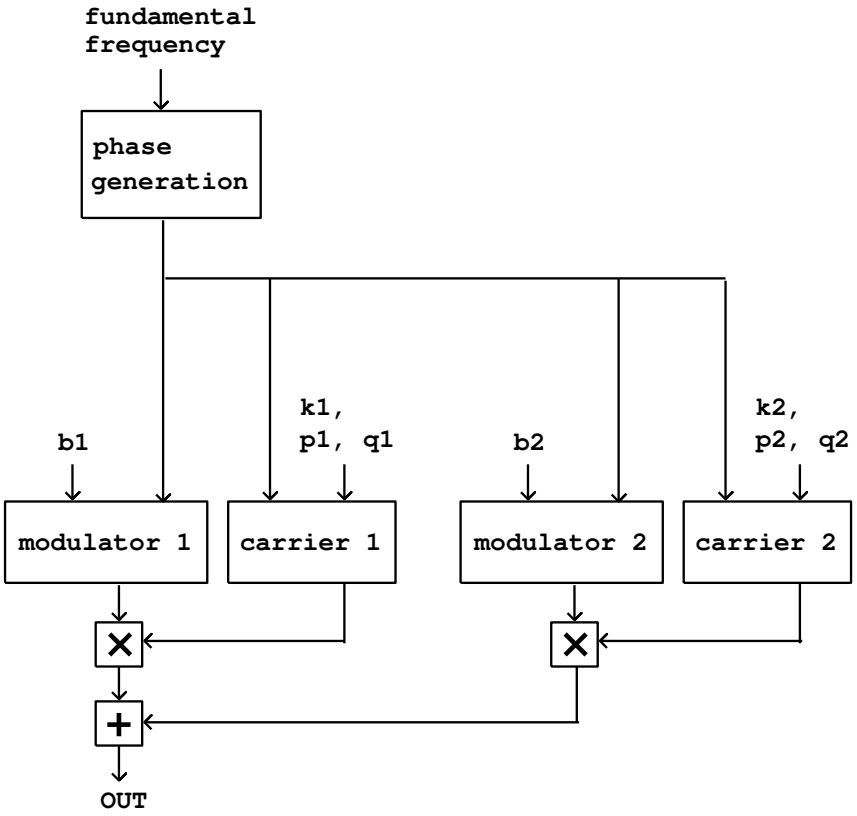


Рис. 6.10: Блок-схема для создания спектра с двумя формантами при использовании генератора PAF.

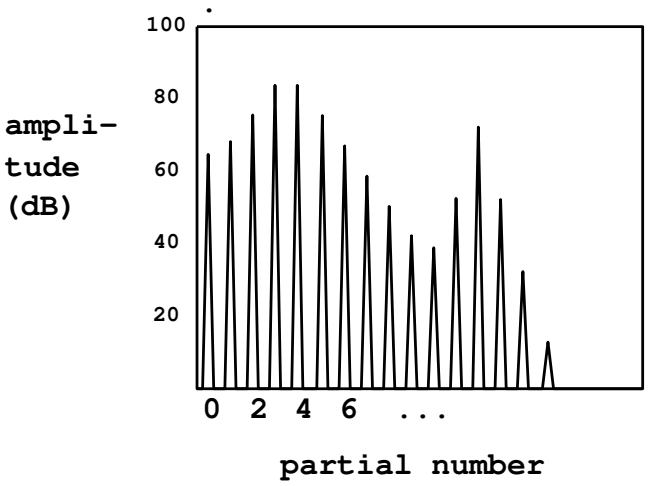


Рис. 6.11: Спектр двух-формантного генератора PAF.

тров, требуется сдвиг гармоник вверх или вниз по частоте. Для того чтобы сделать это, добавим второй осциллятор к фазе обоих несущих косинусов, но не к фазе модуляционной части диаграммы, и не к управляющей фазе единиц "выборки и хранения". Оказывается, что стратегия "выборки и хранения" для гладкого изменения параметров по-прежнему работает; более того, многократные генераторы PAF, разделяющие один и тот же сигнал фазового генератора, будут, по-прежнему, в фазе один с другим.

Этот метод суперпозиции спектров не работает так же предсказуемо для фазовой модуляции, как для генератора PAF; гармоники выхода фазовой модуляции имеют сложные взаимоотношения и их согласованное объединение оказывается трудно осуществить. В общем, фазовая модуляция будет давать более сложные образцы спектральной эволюции, в то время как PAF проще для предсказания и предназначен для специфических требуемых эффектов.

6.5 Примеры

6.5.1 Импульсная последовательность волновой таблицы

Пример F01.pulse.pd (рис. 6.12) генерирует импульсную последовательность с изменяемой шириной при помощи просмотра волновой таблицы с растяжением. Рисунок 6.13 показывает два промежуточных результата патча и его вывод. Патч реализован самым простым образом, помещая импульс на фазу π, вместо фазы ноль; в примерах далее это будет фиксировано добавлением 0.5 к фазе и зацикливанием.

Начальная фаза настраивается для вывода значений от -0.5 до $+0.5$ и, далее, масштабируется множителем, равным, как минимум, единице, выводя сигнал на рис. 6.13 (часть a); это соответствует выходу объекта [*~], пятого снизу на показанном патче. График части (b) показывает результат ограничения значений пилообразной волны в диапазоне от -0.5 до $+0.5$ с использованием объекта [clip~]. Если бы масштабирующий множитель достиг своего минимума (единицу), пилообразная волна имела бы диапазон от -0.5 до $+0.5$ и ограничение не производилось. Для любого значения масштабирующего множителя, большего, чем единица, ограниченное выходное значение остается равным -0.5, далее изменяется линейно до $+0.5$ и, далее, остается уже на $+0.5$. Больший множитель даёт более быстрое линейное изменение формы волны и большее время волна проводит после ограничения снизу и сверху.

Затем объект [cos~] преобразует эту форму волны в импульс. Входные значения как -0.5, так и $+0.5$ становятся -1 (они находятся на границе цикла); в середине входной сигнал равен 0, а выходной сигнал равен 1. После этого выходной описывает полный цикл функции косинус и возвращается назад в -1[2]. Относительное время, которое волна строит функцию косинуса составляет число обратное мас-

[2]на части (c) рис. 6.13 ошибочно указан диапазон выходных значений: должно быть от -1 до $+1$. (*прим. перев.*)

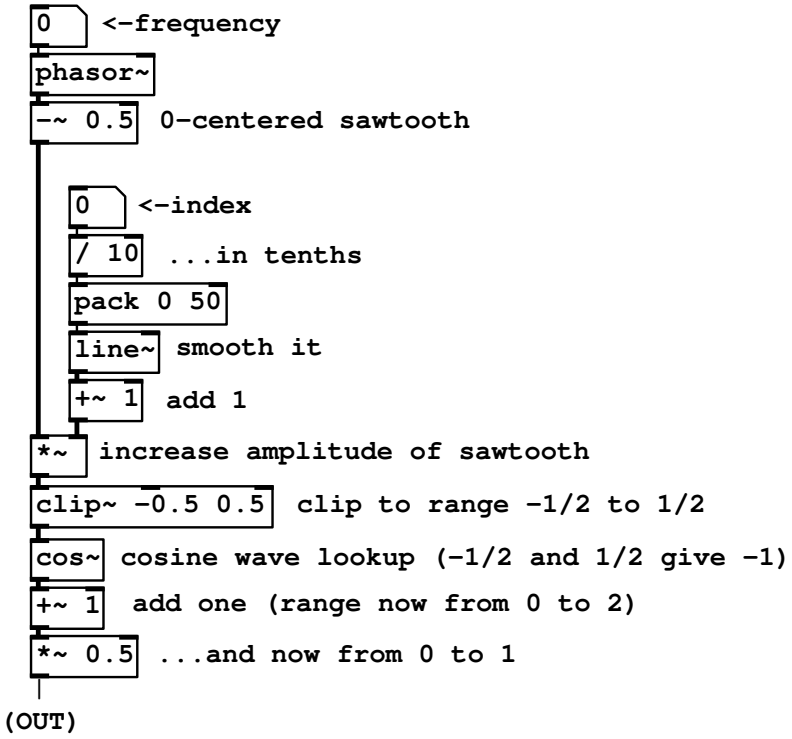


Рис. 6.12: Пример патча F01.pulse.pd, который синтезирует импульсную последовательность с использованием просмотра волновой таблицы с растяжением.

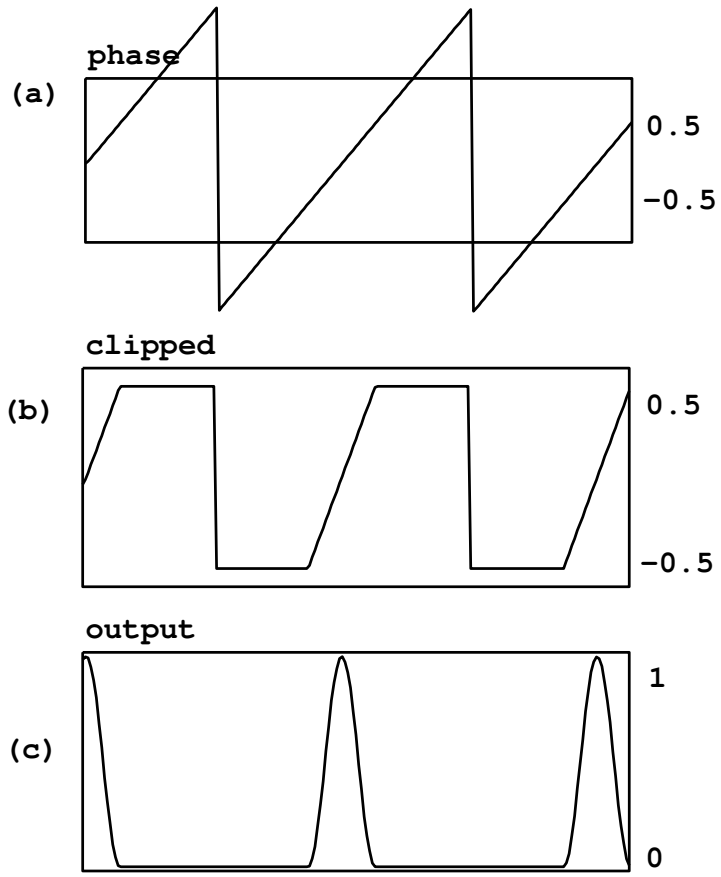


Рис. 6.13: Промежуточные аудиосигналы из рис. 6.12: (a) результат умножения пилообразного сигнала на “индекс”; (b) тоже, ограниченное в диапазоне между 0.5 и 0.5; (c) выход.

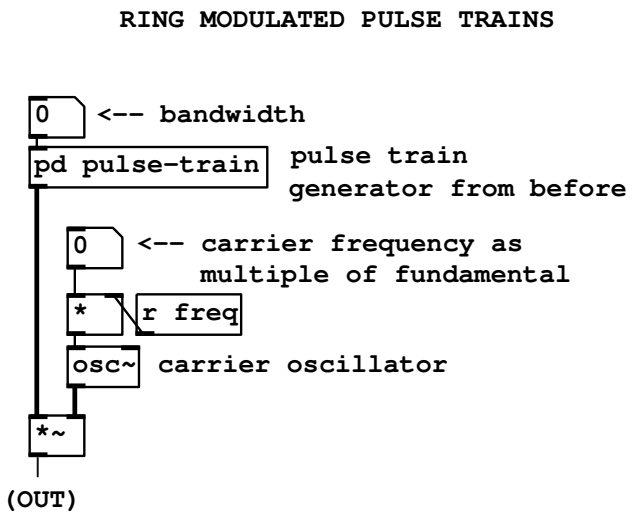


Рис. 6.14: Фрагмент примера F05.ring.modulation.pd, демонстрирующего объединение кольцевой модуляции с генератором импульсов растянутой волновой таблицы.

штабирующий множитель; так что это 100% для множителя 1, 50% для 2 и т.д. Наконец, импульсный выход настраивается на диапазон от 0 до 1 по величине; это изображено в части (c) рисунка.

6.5.2 Простой генератор форманты

В следующих трёх примерах демонстрируется звук с изменяемой шириной импульсов, строится график его спектра и производится сопоставление с импульсным генератором, использующим вейвшейпинг. Перескакивая сразу к примеру F05.ring.modulation.pd (рис. 6.14), мы показываем простейший способ объединения импульсного генератора с осциллятором с кольцевой модуляцией для получения форманты. Импульсная последовательность предыдущего примера содержится в сабпатче Pd pulse-train. Она умножается на осциллятор, чья частота управляется таким образом, чтобы быть кратной основной частоте. Если кратное — целое, мы получаем гармоничный звук. При этом не делается попытки управлять относительными фазами компонент импульсной последовательности и несущей синусоиды.

Следующий пример, F06.packets.pd (рис. 6.15) показывает, как комбинировать импульсную последовательность растянутой волновой таблицы с семплированной синусоидой для реализации подвижных формант, как описывалось в §6.3. Импульсный генератор — такой же как раньше, но теперь несущий сигнал является синусоидой со скачками. Поскольку её фаза равна произведению основной фазы и относительной центральной частоты, приращение фазы семпл в семпл такое же, как для синусоиды с центральной частотой. Однако когда фаза сворачивается, фаза несущего сигнала прыгает на другое место в цикле, как было проиллюстрировано на рис. 6.7. Хотя относительная ширина полосы ω_b/ω должна быть, как минимум, единицей, относительная центральная частота ω_c/ω может быть малой вплоть до

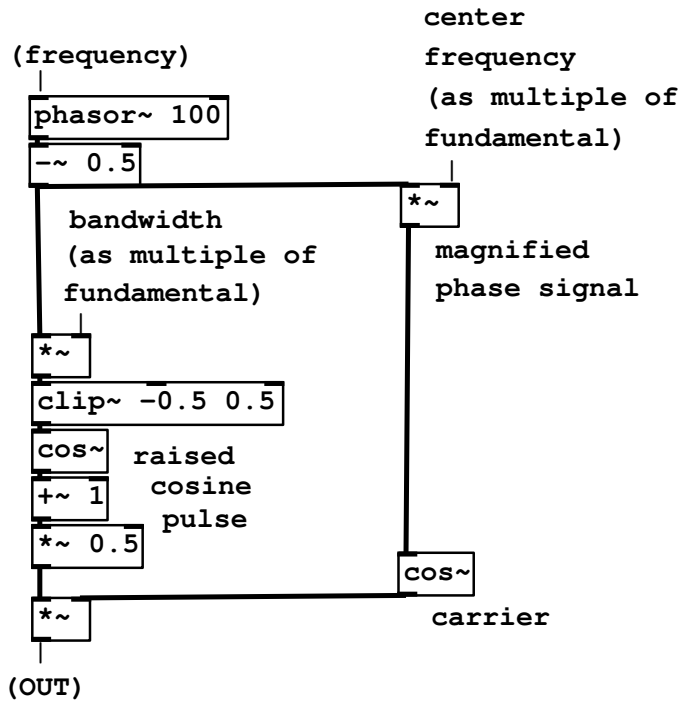


Рис. 6.15: Использование техники растяжения волновых таблиц для получения форманты с изменяемой центральной частотой

нуля, если это требуется.

6.5.3 Двухкосинусный несущий сигнал

Пример F08.two.cosines.pd (рис. 6.16) показывает, как создать несущий сигнал, который медленно меняет гармоники для получения непрерывно меняющихся центральных частот. Относительная центральная частота появляется как выход объекта line~. Из него извлекается его дробная часть (с использованием объекта wrap~) и целая часть (подстановкой дробной части из оригинала). Они обозначаются метками q и k для согласованности с обсуждением в §6.3.

Фаза (пилообразная волна с основной частотой) умножается и на k и на $k+1$ (для последнего добавлением первоначальной пилы к предыдущему вычисленному значению), после чего берутся косинусы обоих получившихся фаз, которые равны k и $k+1$ умноженным на основную частоту и не имеют разрывов в точках сворачивания фазы. Следующие несколько объектов в патче вычисляют взвешенную сумму $pc_1 + qc_2$, где c_1 и c_2 есть две синусоиды и $p = 1 - q$, которая вычисляется в эквивалентном виде $c_1 + q(c_2 - c_1)$. Это даёт нам требуемый несущий сигнал с подвижной частотой.

Пример F09.declickit.pd (не показан здесь) демонстрирует, как добавлением объекта samphold~ после объекта line~, управляющего центральной частотой, можно

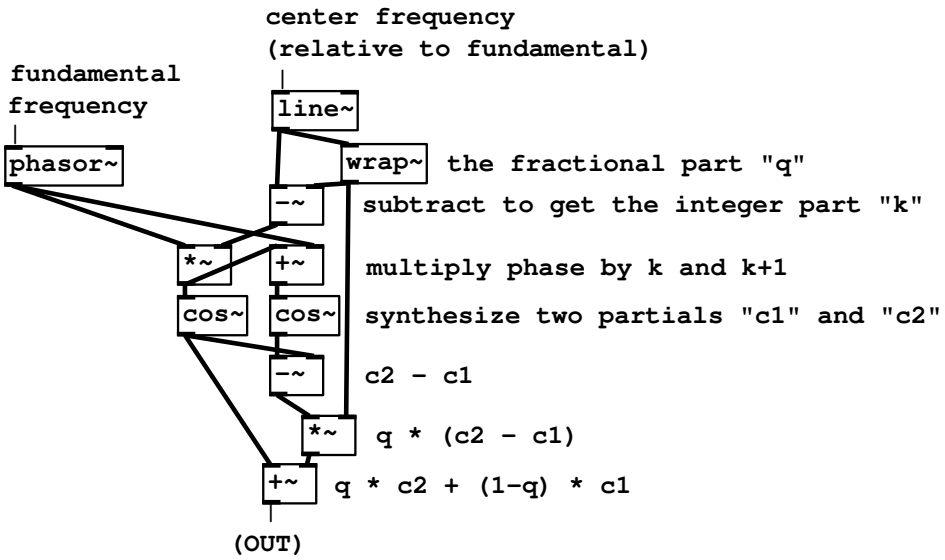


Рис. 6.16: Микширование между синусоидами для получения изменяемых центральных частот

избежать разрывов в выходном сигнале, даже если требуемая центральная частота меняется скачкообразно. В примере, относительная центральная частота изменяется между 4 и 13.5. При временах линейного изменения ниже, чем примерно 20 миллисекунд, имеются слышимые артефакты при использовании одного объекта line~, которые исчезают, если добавить объект samphold~. (Недостаток метода "выборки и хранения" относительной частоты состоит в том, что для очень низких основных частот, изменения могут быть слышимы в виде дискретных шагов. Так что в ситуациях, когда основная частота является низкой и нет необходимости в очень быстро меняющейся центральной частоте, может быть лучшим опустить шаг "выборки и хранения".)

Следующие два примера демонстрируют использование микширования осцилляторов несущего сигнала, как часть классической техники двухоператорной фазовой модуляции. Один и тот же модулирующий осциллятор добавляется отдельно к фазам двух косинусов. Получающиеся спектры могут быть сделаны двигающимися вверх и вниз по частоте, но в силу сложных фазовых взаимоотношений между соседними пиками в спектре фазовой модуляции, неважно, как вы выстроите эти два спектра — всё равно вы никогда не сможете избежать фазовых сокращений там, где они перекрываются.

6.5.4 Генератор PAF

Пример F12.paf.pd (рис. 6.17) является реализацией генератора PAF, описанного в §6.4. Управляющие входы определяют основную частоту, центральную частоту и ширину полосы, все в единицах MIDI. Первые шаги, проделываемые при реализации, состоят в делении центральной частоты на основную (для получения

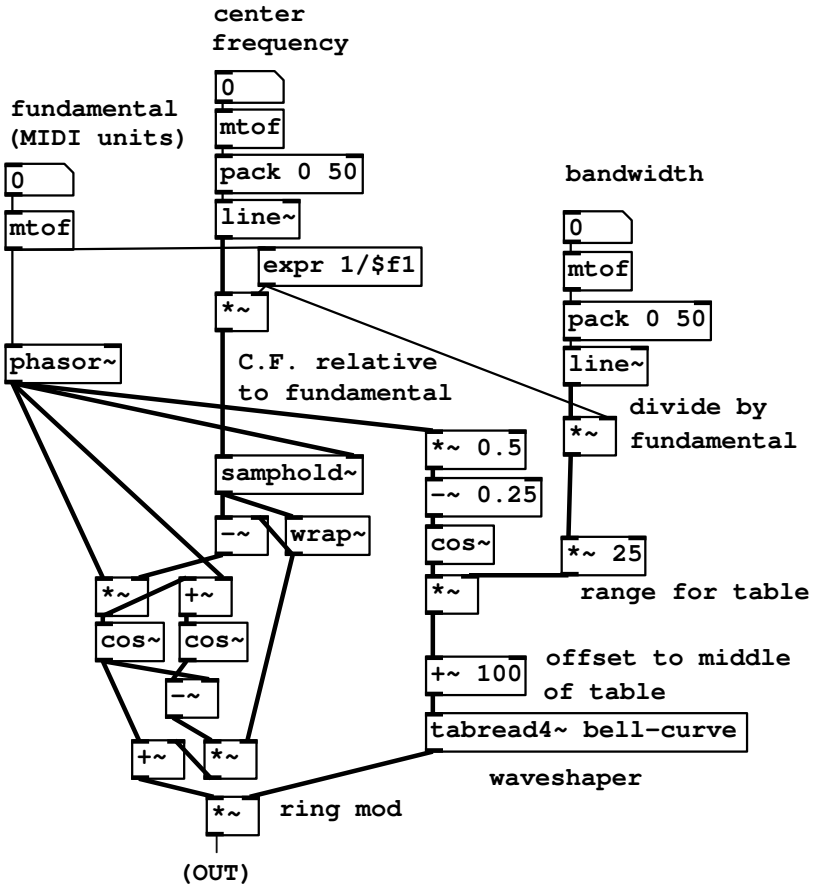


Рис. 6.17: Алгоритм синтеза фазово-выстраиваемой форманты (PAF).

относительной центральной частоты) и ширины полосы на основную частоту, для получения индекса модуляции, задающего форму волны. Относительная центральная частота захватывается и удерживается, так что она обновляется с периодами, задаваемыми основной частотой.

Один осциллятор (объект phasor~) установлен на основную частоту. Это используется и для управления объектом samphold~, который синхронизует обновления с относительной центральной частотой (отмеченной как "C.F. relative to fundamental" на рисунке), и для вычисления фаз для всех объектов cos~, которые действуют, как показано ранее на рис. 6.16.

Часть патча, относящаяся к установке формы волны, использует половину периода синусоиды в качестве функции просмотра (для компенсации удвоения частоты из-за симметричности функции просмотра). Для получения половины цикла функции синус, мы умножаем фазу на 0.5 и вычитаем 0.25, так что подогнанные фазы пробегают значения от -0.25 до $+0.25$ за каждый период. Это считывает положительную половину цикла, определённого объектом cos~.

Амплитуда половины синусоиды настраивается далее с помощью индекса модуляции (который равен, в точности, относительной ширине полосы ω_b/ω). Таблица

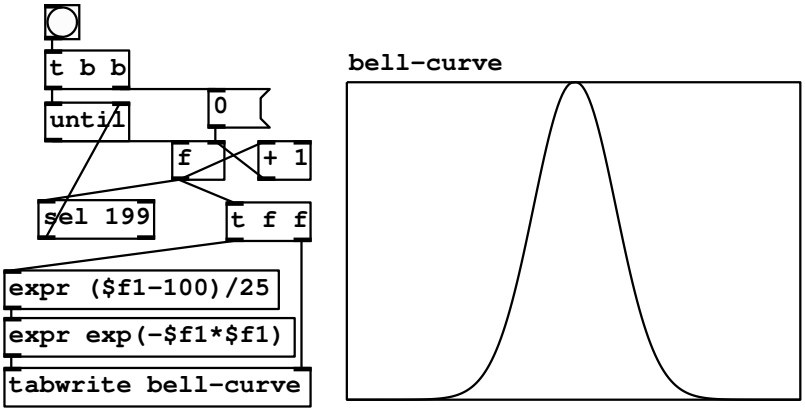


Рис. 6.18: Заполнение волновой таблицы для рис. 6.17.

(колоколообразная кривая) хранит ненормированную гауссову кривую, семплированную от −4 до 4 на 200 точек (25 точек на единицу), так что центр таблицы, в точке 100, соответствует центральному пику колоколообразной кривой. Вне интервала от −4 до 4 гауссовская кривая пренебрежимо мала.

Рисунок 6.18 показывает, как приготавливается гауссова волновая таблица. Необходим один новый управляющий объект:

> until: Когда на левый, “запускающий” вход отправлен bang, объект выводит последовательные послания bang (без временных пробелов между ними) итеративно, пока на правый “останавливающий” вход будет не послан bang. Останавливающее сообщение “bang” должно порождаться каким-либо образом выходом объекта until; в противном случае выход будет посылать сообщения “bang” до бесконечности, замораживая любой другой объект, который мог бы разомкнуть цикл.

Здесь используется циклическое повторение, управляемое объектом until, пересчитывающим значения от 0 до 199 включительно. Подсчёт циклов поддерживается объектами f и + 1, каждый из которых поставляет информацию для другого. Но, когда выход объекта + 1 идёт на правый вход f, его результат (на единицу больший) будет лишь тогда появляться из f, когда на него в очередной раз будет послано сообщение bang от until. Таким образом, каждый “bang” от until увеличивает значение на единицу.[3]

Порядок, в котором цикл запускается, следующий: объект t b b сверху (сокращение от trigger bang bang) должен, во-первых, отправить ноль на f, инициализируя, таким образом, его, и, далее, установить объект until на посылание “bang”, которые увеличивают значение, вплоть до остановки. Для остановки процесса, когда достигается значение 199, объект select проверяет значение и, когда видит соответствие, посылает “bang” на останавливающий вход until.

[3]Это стандартная для Pd идиома счетчика. (*прим. перев.*)

Между тем, для каждого числа от 0 до 199, которое приходит от объекта f, мы создаём упорядоченную пару сообщений для объекта tabwrite. Первое, на правый вход, пробегает индекс сам по себе, от 0 до 199. Сообщение на левый вход формируется так: первый объект expr подгоняет индекс на диапазон от -4 до 4 (первоначальный диапазон был от 0 до 199) и второй вычисляет функцию Гаусса.

В этом патче мы не полностью затронули результат обновления относительной центральной частоты в подходящие моменты времени. Всякий раз когда несущая частота изменяется, шаг "выборки и хранения" успешно задерживает обновление относительной частоты. Но, если (вместо этого или дополнительно) основная частота сама по себе резко меняется, тогда на долю периода частота объекта phasor~ и относительная частота не синхронизированы. Pd не позволяет выходу samphold~ быть соединённым назад с входом phasor~ без включения точной задержки (см. следующую главу) и нет простого способа модифицирования патча для решения этой проблемы.

Предполагая, что мы каким-либо образом достигли синхронизации входа объекта phasor~ с его собственными точками сворачивания фазы, мы бы должны были сделать то же самое для частного (ширина полосы)/(основная частота) с правой стороны патча, также. В текущем сценарии, однако, нет проблемы непрерывного обновления этой величины.

Практическое решение проблемы обновления могло бы просто заключаться в переписывании всего патча на C, как классе Pd; это, к тому же, приведёт к использованию гораздо меньшего времени CPU, чем для изображённого патча, и является более практичным решением (пока вы не хотите экспериментировать с изменениями алгоритма). Такие изменения могут включать: добавление ангармонического сдвига гармоник вверх или вниз; разрешение переключения между гладкими и использующими "выборку и хранение" обновлениями центральной частоты; добавление отдельных управлений увеличением для чётных и нечётных обертонов; введение "зернистости" с помощью иррегулярной модуляции фазы; возможности микширования двух или более функций вейвшейпинга; или создание более резких перкуссионных атак с помощью выстраивания фазы осциллятора по отношению ко времени амплитудного генератора огибающей.

Одна последняя деталь об амплитуде, для полноты: поскольку амплитуда наиболее сильной гармоники убывает приблизительно как $1/1 + b$, где b есть индекс модуляции, иногда (но не всегда) желательно скорректировать амплитуду выхода умножением на $1 + b$. Это является возможным вариантом, если b гладко обновляется (как в этом примере), но не приемлемо при выборке и хранении. Одна из ситуаций, когда это приемлемо — моделирование щипка струн (устанавливая центральную частоты на основную, и начиная с высокого индекса модуляции с уменьшением его экспоненциально); затуханию струны соответствует спад, а не рост амплитуды основной частоты.

6.5.5 Растянутые волновые таблицы

Вместо использования вейвшейпинга, формантный синтез также возможен с использованием растягивания волновых таблиц, как демонстрирует пример F14.wave.packet.pd (не показан здесь). Метод, в существенном, такой же, как в примере B10.sampler.overlap.pd (описан в §2.6), с просмотром косинусной таблицы, вместо более общей, но с добавлением управления для установки рабочего цикла амплитудных огибающих. Единицы подогнаны так, чтобы быть совместимыми с единицами предыдущего примера.

Упражнения

1. Последовательность импульсов содержит окна Ханна (верхние части косинусов), один за другим, без промежутков между ними. Каков будет результирующий спектр?

2. Какие (в терминологии рис. 6.8) должны быть значения k и q для синтеза форманты с центральной частотой 2000 герц и основной частотой 300 герц?

3. Как бы вы изменили блок-схему рис. 6.8 для получения только нечётных гармоник?

Глава 7

Временные сдвиги и задержки

Как-нибудь, в 17:00 поставьте вашу любимую запись пятого струнного квартета Ramones. В следующую субботу, воспроизведите ту же запись в 17:00:01, на одну секунду позже. В идеале два воспроизведения должны звучать одинаково. Сдвиг всей вещи на секунду (или, если хотите, на несколько дней и секунду) не имеет физического воздействия на звук.

А теперь, предположим, что вы проигрываете эту запись в 17:00 и 17:00:01 того же дня (на двух различных воспроизводящих системах, поскольку продолжительность музыки гораздо большая, чем одна секунда). Теперь звук совершенно другой. Разница, какова бы она ни была, явно относится не к одному из двух индивидуальных звуков, но скорее к *интерференции* между ними. Эта интерференция может быть различима по меньшей мере четырьмя различными образами:

Канон: Комбинируя две копии сигнала с временным сдвигом, достаточным для того, чтобы сигнал изменился ощутимо, мы можем слышать два сигнала в виде отдельных музыкальных потоков, в результате чего сравниваем сигнал с его собственной более ранней частью. Если сигнал является мелодией, временной сдвиг может быть сравним с длиной одной или нескольких нот.

Эхо: С временными сдвигами между, примерно, 30 миллисекундами и одной секундой, более поздняя копия сигнала может звучать как эхо более раннего сигнала. Эхо может уменьшать разборчивость сигнала (особенно, если он состоит из речи), но обыкновенно не будет менять всю “форму” мелодий или фраз.

Фильтрация: При временных задержках менее 30 миллисекунд (примерно), копии слишком близки друг к другу по времени, чтобы восприниматься независимо, и основной эффект состоит в том, что некоторые частоты усиливаются, а другие подавляются. Это меняет спектральную огибающую звука.

Изменение акустического пространства: Если вторая копия воспроизводится тише первой и, особенно, если мы добавляем больше копий с задержкой и

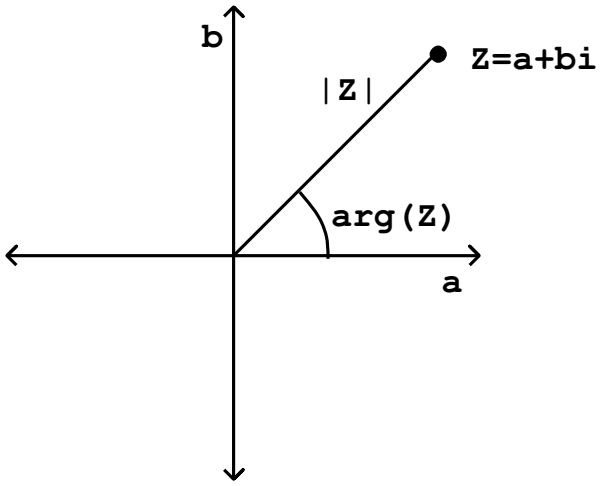


Рис. 7.1: Число Z в комплексной плоскости. Оси a — для действительной части и b — для мнимой части.

меньшими амплитудами, результат может быть похожим на эхо, которые возникает в комнате или другом акустическом пространстве.

В зависимости от того, как организована задержка копий сигнала, полученное звучание может состоять из комбинации вышеуказанных эффектов.

Математически, влияние временн*о*го сдвига на сигнал может быть описано как фазовое изменение каждой из составляющих его синусоидальных компонент. Фазовый сдвиг каждой компоненты различен в зависимости от её частоты (а также от величины временного сдвига). В дальнейшем, в этой главе, мы будем часто рассматривать суперпозиции синусоид с различными фазами. Ранее, в нашем анализе, мы довольствовались использованием действительнозначных синусоид, но в этой и дальнейших главах формулы станут более сложными и нам потребуется более мощный математический аппарат для обращения с ними. В предварительном параграфе этой главы мы рассмотрим дополнительный аппарат, который нам понадобится.

7.1 Комплексные числа

Комплексные числа записываются в виде:

$$Z = a + bi,$$

где a и b есть действительные числа и $i = \sqrt{-1}$. (В этой книге мы будем использовать заглавные латинские буквы, такие как Z, для обозначения комплексных чисел. Действительные числа появляются в виде прописных латинских и греческих букв, за исключением целочисленных значений диапазонов, обыкновенно пишущихся как

M или N.) Поскольку комплексные числа имеют две действительные компоненты, мы используем декартову систему координат для графического представления, как показано на рис. 7.1. Величины a и b называются *действительной* и *мнимой* частями Z, что записывается как

$$a = \mathrm{re}(Z)$$
$$b = \mathrm{im}(Z).$$

Если Z — комплексное число, его *модуль* (или *абсолютное значение*), обозначаемое $|Z|$, есть, в точности, расстояние на плоскости от начала координат до точки (a, b):

$$|Z| = \sqrt{a^2 + b^2},$$

а его *аргумент*, обозначаемый $\angle(Z)$, равен углу между положительной осью a и отрезком, соединяющим начало координат с точкой (a, b):

$$\angle(Z) = \arctan\left(\frac{b}{a}\right).$$

Если мы знаем модуль и аргумент комплексного числа (назовём их r и θ), мы можем восстановить действительную и мнимую части:

$$a = r\cos(\theta)$$

$$b = r\sin(\theta).$$

Комплексное число может быть записано через действительную и мнимую части a и b как $Z = a + ib$ (это называется прямоугольная форма) или, альтернативно, в *полярной* форме, через r и θ:

$$Z = r \cdot [\cos(\theta) + i\sin(\theta)].$$

Прямоугольный и полярный виды взаимозаменяемы; уравнения, приведённые выше, показывают, как вычислять a и b из r и θ и наоборот.

Основная причина, по которой мы используем комплексные числа в электронной музыке состоит в том, что они волшебным образом автоматизируют тригонометрические вычисления. Нам часто нужно складывать углы, для того чтобы говорить об меняющейся фазе аудиосигнала во времени (или сдвигах во времени, как в этой главе). Оказывается, что если вы перемножаете два комплексных числа, аргумент произведения является суммой аргументов двух множителей. Для того чтобы увидеть, как это происходит, мы умножим два числа Z_1 и Z_2, записанных в полярной форме:

$$Z_1 = r_1 \cdot [\cos(\theta_1) + i\sin(\theta_1)]$$
$$Z_2 = r_2 \cdot [\cos(\theta_2) + i\sin(\theta_2)],$$

что даёт

$$Z_1 Z_2 = r_1 r_2 \cdot [\cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2) +$$

$$+i\left(\sin(\theta_1)\cos(\theta_2)+\cos(\theta_1)\sin(\theta_2)\right)].$$

Здесь знак минус перед $sin(\theta_1)sin(\theta_2)$ возникает от умножения i на себя, что даёт -1. Мы можем заметить формулы косинуса и синуса суммы в выражении выше, что упрощает выражение:

$$Z_1Z_2 = r_1r_2 \cdot \left[\cos(\theta_1+\theta_2)+i\sin(\theta_1+\theta_2)\right].$$

Проверка показывает, что произведение Z_1Z_2 имеет модуль r_1r_2 и аргумент $\theta_1+\theta_2$.

Мы можем использовать это свойство комплексных чисел для сложения и вычитания углов (умножая и деля комплексные числа с подходящими аргументами) и, далее, извлекая косинус и синус из результата, взяв действительную и мнимую части.

7.1.1 Комплексные синусоиды

Напомним формулу для (действительнозначной) синусоиды на стр. 1:

$$x[n] = a\cos(\omega n+\phi).$$

Это последовательность косинусов углов (называемых фазами), которые возрастают арифметически с ростом номера семпла n. Амплитуда всех косинусов задается множителем a. Теперь, используя свойства аргументов и модулей, мы можем переписать это выражение как действительную часть более простой и лёгкой для манипулирования последовательности комплексных чисел.

Предположим, что комплексное число Z имеет единичный модуль и аргумент ω, так что оно может быть записано в виде:

$$Z = \cos(\omega)+i\sin(\omega).$$

Тогда для любого целого n, число Z^n должно также иметь единичный модуль (поскольку модули перемножаются) и аргумент $n\omega$ (поскольку аргументы складываются). Таким образом,

$$Z^n = \cos(n\omega)+i\sin(n\omega).$$

Это также верно для отрицательных значений n, так, например,

$$\frac{1}{Z} = Z^{-1} = cos(\omega)-i\sin(\omega).$$

Рисунок 7.2 показывает графически, как степени Z закручиваются вокруг единичной окружности, состоящей из комплексных чисел с единичным модулем. Они образуют геометрическую прогрессию:

$$\ldots, Z^0, Z^1, Z^2, \ldots$$

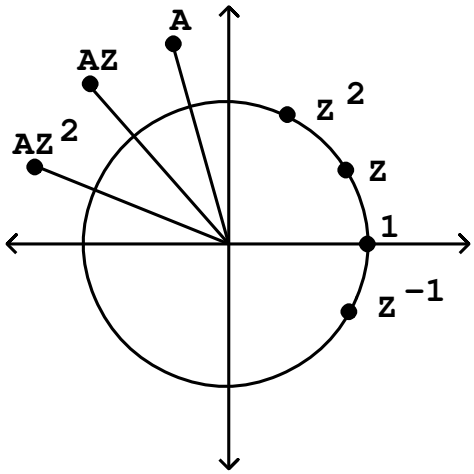


Рис. 7.2: Степени комплексного числа Z с $|Z| = 1$ и та же последовательность, умноженная на константу A.

и взяв действительную часть каждого члена мы получаем действительную синусоиду с нулевой начальной фазой и единичной амплитудой:

$$\ldots, \cos(0), \cos(\omega), \cos(2\omega), \ldots$$

Более того, предположим, что мы умножаем элементы последовательности на какую-нибудь (комплексную) константу A с модулем a и аргументом ϕ. Это даёт

$$\ldots, A, AZ, AZ^2, \ldots$$

Все модули равны a и аргумент n-ого члена есть $n\omega + \phi$, так что последовательность будет

$$AZ^n = a \cdot [\cos(n\omega + \phi) + i \sin(n\omega + \phi)]$$

и действительная часть есть, в точности, действительнозначная синусоида:

$$\mathrm{re}(AZ^n) = a \cdot \cos(n\omega + \phi).$$

Комплексное число A кодирует информацию и о действительной амплитуде a и начальной фазе ϕ; комплексное число с единичным модулем Z управляет частотой, которая есть, в точности, его аргумент ω.

Рисунок 7.2 также показывает последовательность $A, AZ, AZ^2, \ldots$; в действительности это та же самая последовательность как $1, Z, Z^2, \ldots$, но увеличенная по модулю и повернутая, в согласии с амплитудой и начальной фазой. Для комплексной синусоиды такого вида A называется *комплексной амплитудой*.

Использование комплексных чисел для представления амплитуд и фаз синусоид может упростить манипуляции, которые в противном случае были бы неоправданными. Например, предположим, что мы хотим знать амплитуду и фазу суммы

двух синусоид с одной и той же частотой. На языке этой главы, мы записываем эти две синусоиды в виде

$$X[n] = AZ^n,\ Y[n] = BZ^n,$$

где A и B кодирует фазы и амплитуды двух сигналов. Тогда сумма равна

$$X[n] + Y[n] = (A + B)Z^n,$$

что является синусоидой, чья амплитуда равна $|A+B|$, а фаза $\angle(A+B)$. Очевидно, что это гораздо более простой способ манипуляции с амплитудами и фазами, чем использование свойств синусов и косинусов. В ряде случаев, мы будем брать действительную часть результата; обыкновенно это может быть сделано в конце вычислений.

7.2 Временные сдвиги и изменения фазы

Взяв за основу любой (действительный или комплексный) сигнал $X[n]$, мы можем создавать другие сигналы, сдвигая время сигнала X на (положительное или отрицательное) целое d:

$$Y[n] = X[n - d]$$

так что d-ый семпл Y является нулевым семплом X и так далее. Если целое d положительно, тогда Y задержанная копия X. Если d отрицательно, тогда Y предшествует X; такая процедура может быть осуществлена с предварительно записанным звуком, но не осуществима на практике в реальном времени.

Временной сдвиг является линейной операцией (рассматриваемой как функция входного сигнала); если происходит сдвиг по времени суммы $X_1 + X_2$, получается тот же результат, как если бы слагаемые сдвигались по времени отдельно, а потом бралась их сумма.

Временной сдвиг имеет другое свойство — при сдвиге по времени синусоиды с частотой ω получается другая синусоида с той же частотой; временной сдвиг никогда не добавляет частот, которых не было до этого в сигнале. Это свойство, называемое *временная инвариантность*, делает более простым анализ эффектов временного сдвига (и его линейных комбинаций), рассматривая по отдельности воздействие этой операции на составляющие синусоиды.

Более того, эффект временного сдвига на синусоиду прост: он лишь меняет фазу. Если вы используете комплексную синусоиду, эффект даже проще. Если, например,

$$X[n] = AZ^n$$

тогда

$$Y[n] = X[n - d] = AZ^{(n-d)} = Z^{-d}AZ^n = Z^{-d}X[n]$$

так что временной сдвиг комплексной синусоиды на d семплов есть масштабирование на Z^{-d} — т. е. просто изменение амплитуды на определённое комплексное

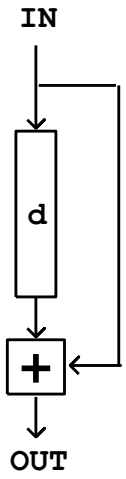


Рис. 7.3: Сеть с задержкой. Здесь мы складываем входящий сигнал с его задержанной копией.

число. Поскольку $|Z| = 1$ для синусоиды, изменение амплитуды не меняет модуль синусоиды, а только фазу.

Изменение фазы равно $-d\omega$, где $\omega = \angle(Z)$ есть угловая частота синусоиды. Это в точности то, что мы должны ожидать, поскольку синусоида продвигается на ω радиан за семпл и начинается (с задержкой) на d-ом семпле.

7.3 Сети с задержкой

Если мы рассматриваем наши цифровые аудиосемплы $X[n]$ как соответствующие последовательным моментам времени, тогда временной сдвиг сигнала на d семплов соответствует задержке на d/R единиц времени, где R есть частота семплирования. Рисунок 7.3 показывает пример *линейной сети с задержкой*: узел задержки (возможно с операциями изменения амплитуды) объединенный с использованием сложения и вычитания. Выход есть линейная функция от входа, в том смысле, что сложение двух сигналов на входе даёт то же самое, что обработка обоих сигналов по отдельности и сложение результатов. Более того, линейные сети с задержкой не создают новых частот на выходе, которые не были представлены на входе, пока сеть остаётся инвариантной по времени, так что времена опережений и задержек не меняются по времени.

В общем, имеется два способа представить себе сети с задержкой. Мы можем представлять их во *временной области*, в которой мы строим формы волн как функции времени (или индекса n) и рассматриваем задержки как сдвиги времени. Также, мы можем создать представление в *частотной области*, в которой мы задаём вход с комплексной синусоидой (так что выход является синусоидой с той же частотой) и описываем амплитудное и/или фазовое изменение, сделанное сетью,

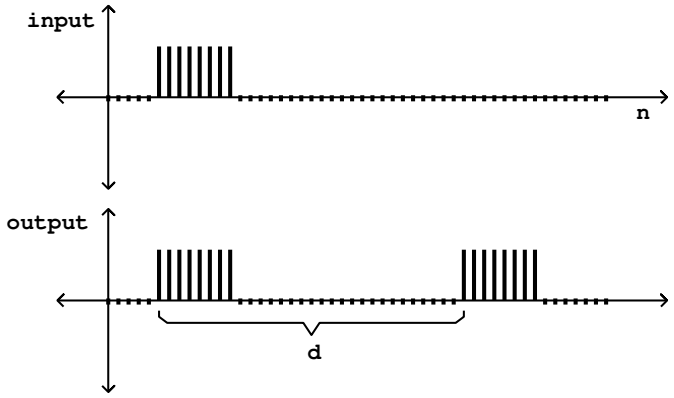


Рис. 7.4: Представление во временной области для сети с задержкой рис. 7.3. Выход является суммой входа и его сдвинутой по времени копии.

как функцию частоты. Теперь рассмотрим сеть с задержкой на рис. 7.3 каждым из способов, по порядку.

Рисунок 4.4 показывает поведение сети во временной области. Мы задаём некоторую тестовую функцию подходящего вида на входе (это прямоугольный импульс шириной в восемь семплов в примере) и изображаем графики входа и выхода как функции номера семпла n. Эта специальная сеть с задержкой складывает вход с его задержанной копией.

Часто используемой тестовой функцией является импульс, который занимает только один семпл. Полезность её заключается в том, что если мы знаем выход сети для такого импульса, мы можем найти выход для любого другого цифрового аудиосигнала — поскольку любой сигнал $x[n]$ является суммой импульсов, первый величины $x[0]$, следующий, появляющийся на семпл позже, имеет величину $x[1]$ и т. д. Позже, когда сети станут более сложными, мы начнём использовать импульсы в качестве входных сигналов, чтобы показать поведение сетей во временной области.

С другой стороны, мы можем проанализировать ту же сеть в частотной области, рассматривая (комплекснозначный) тестовый сигнал

$$X[n] = Z^n,$$

где Z имеет единичный модуль и аргумент ω. Мы уже знаем, что выход является другой комплексной синусоидой с той же частотой, то есть

$$HZ^N$$

для некоторого комплексного числа H (которое мы хотим найти). Так что мы записываем выход напрямую, как сумму входа и его задержанной копии:

$$Z^n + Z^{-d}Z^n = (1 + Z^{-d})Z^n$$

и рассматривая это выражение, находим:

$$H = 1 + Z^{-d}.$$

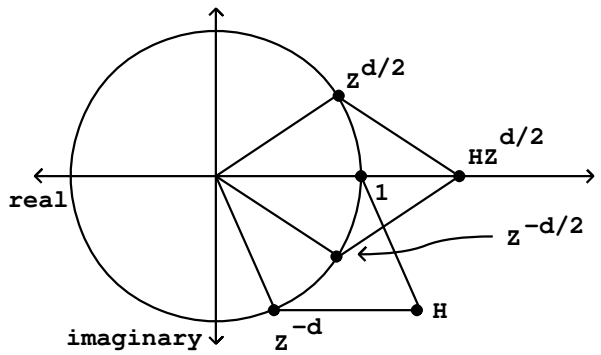


Рис. 7.5: Анализ, в комплексной плоскости, поведения в частотной области сети с задержкой рис. 7.3. Комплексное число Z кодирует частоту входа. Задержанный линейный выход есть вход, умноженный на Z^{-d}. Общий (комплексный) коэффициент усиления есть H. Мы находим модуль и аргумент H с помощью симметризации суммы, поворачивая её на произведение $d/2$ на угловую частоту входа.

Мы можем узнать поведение этой сети с задержкой во временной области изучив, как комплексное число H варьируется как функция угловой частоты ω. Нас особенно интересуют его аргумент и модуль, которые говорят нам относительную фазу и амплитуду выходящей синусоиды. Мы подробно разберём этот пример, чтобы показать, как арифметика комплексных чисел может предсказать, что случится при и сложении синусоид.

Рисунок 7.5 показывает результат сложения величин 1 и Z^{-d} в комплексной плоскости. Для сложения комплексных чисел мы складываем их действительные и мнимые части по отдельности. Так что комплексное число 1 (действительная часть 1, мнимая часть 0) добавляется покоординатно к комплексному числу (действительная часть $\cos(-d\omega)$, мнимая часть $\sin(-d\omega)$). Это показано графически с помощью параллелограмма с вершинами в начале координат и в двух точках, которые должны быть сложены, так что четвёртая вершина даёт сумму H.

Как показывает рисунок, результат может быть понят с помощью симметризации его относительно действительной оси: вместо 1 и Z^{-d}, проще просуммировать величины $Z^{d/2}$ и $Z^{-d/2}$, поскольку они симметричны относительно (горизонтальной) оси. (Строго говоря, мы не определили надлежащим образом величины $Z^{d/2}$ и $Z^{-d/2}$; мы используем эти выражения для обозначения комплексных чисел с единичным модулем, чьи аргументы являются половинами аргументов Z^d и Z^{-d}, так что возведение их в квадрат дало бы Z^d и Z^{-d}.) Мы перепишем результат как

$$H = Z^{-d/2}(Z^{d/2} + Z^{-d/2}).$$

Первый член представляет из себя фазовый сдвиг $-d\omega/2$. Второй член лучше

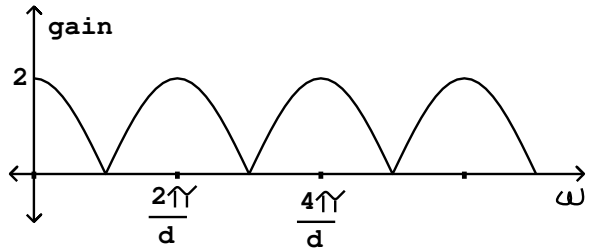


Рис. 7.6: Коэффициент усиления сети с задержкой рис. 7.3, показанный как функция угловой частоты ω.

воспринимается в прямоугольной форме:

$$Z^{d/2} + Z^{-d/2}$$

$$= (\cos(\omega d/2) + i\sin(\omega d/2)) + (\cos(\omega d/2) - i\sin(\omega d/2))$$

$$= 2\cos(\omega d/2).$$

Это вещественная величина может быть или положительной или отрицательной; её абсолютное значение даёт модуль выхода:

$$|H| = 2|\cos(\omega d/2)|.$$

Величина $|H|$ называется коэффициентом усиления сети с задержкой для угловой частоты ω и изображена на рис. 7.6. Коэффициент усиления сети с задержкой в зависимости от частоты (то есть коэффициент усиления как функция частоты), называется *частотным откликом сети.*

Поскольку сеть имеет больший коэффициент усиления на некоторых частотах, по отношению к другим, она может рассматриваться как фильтр, который может быть использован для отделения определённых компонент звука от других. В силу формы зависимости коэффициента усиления как функции от ω, этот вид сети с задержкой называется *гребенчатым фильтром* (без рециркуляции).

Выход сети есть сумма двух синусоид одинаковой амплитуды, чьи фазы отличаются на ωd. Результирующий частотный отклик согласован с интуитивным ощущением: если угловая частота ω установлена так, что целое число периодов подогнано к d семплам, то есть если ω кратно $2pi/d$, выход задержки в точности такой же как первоначальный сигнал и, следовательно, они комбинируются создавая в два раза большую амплитуду на выходе. С другой стороны, если, например, мы берём $\omega = pi/d$, так что задержка составляет половину периода, тогда задержанный выходной сигнал идет в противофазе и полностью сокращает входной.

Эта конкретная сеть с задержкой имеет интересное применение: если у нас есть периодический (или почти периодический) входной сигнал, чья основная

частота составляет ω радиан на семпл, мы можем настроить гребенчатый фильтр так, что пики коэффициента усиления выстраиваются по чётным гармоникам, а на нечётных усиление падает и обращается в ноль. Для осуществления этого мы выбираем $d = \pi/\omega$, то есть устанавливаем время задержки в точности на половину периода входного сигнала. Тем самым, мы получаем новый сигнал, чьи гармоники есть 2ω, 4ω, 6ω, ... и его основная частота, таким образом, теперь в два раза больше первоначальной. Кроме множителя два, амплитуды оставшихся гармоник по-прежнему следуют спектральной огибающей первоначального звука. Следовательно, теперь у нас есть инструмент для изменения высоты входящего звука на октаву выше без изменения спектральной огибающей. Этот удвоитель октавы является обратной операцией к делителю октавы, введённому ранее в Главе 5.

Временн*а*я и частотная области предлагают взаимодополняющие способы рассмотрения одной и той же сети с задержкой. Когда задержки внутри сети меньше возможностей слуха для распознавания событий по времени — примерно менее 20 миллисекунд — представление во временной области становится менее важным для нашего понимания сети с задержкой и мы обращаемся больше к представлению в частотной области. С другой стороны, когда задержки превышают 50 миллисекунд, пики и впадины графиков, показывающих коэффициент усиления по отношению к частоте (так как на рис. 7.6) стиснуты так близко вместе, что представление в частотной области становится менее важным. Оба подхода, тем не менее, пригодны во всём диапазоне возможных времён задержки.

7.4 Сети с задержкой и обратной связью

Иногда требуется вернуть выходы одной или нескольких задержек в сети назад на их собственные или любые другие входы. Вместо того, чтобы получить одно или несколько эхо-сигналов первоначального звука, как в примерах выше, мы, потенциально, можем получить бесконечное число эхо-сигналов, каждый из которых возвращается в сеть для порождения еще других.

Простейший пример рециркулирующей сети есть гребенчатый фильтр с обратной связью, блок-схема которого показана на рис. 7.7. Как и ранее, у простого гребенчатого фильтра, входной сигнал посылается вниз через линию задержки, чья длина составляет d семплов. Но теперь выход линии задержки также возвращается на свой вход; вход задержки является суммой первоначального входа и выхода задержки. Выходной сигнал умножается на число g перед возвращением на вход.

Поведение во временной области гребенчатого фильтра с рециркуляцией показано на рис. 7.8. Здесь мы рассматриваем эффект отправки импульса в такую сеть. Мы получаем назад первоначальный импульс, плюс серию эхо-сигналов, каждый из которых, по очереди, через d семплов после предыдущего, с умножением каждый раз на коэффициент усиления g. В общем, выходной сигнал сети с задержкой, на которую в качестве входа подан импульс, называется *импульсным откликом* сети.

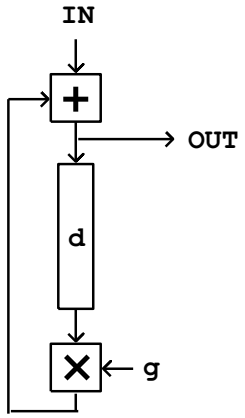


Рис. 7.7: Блок-схема гребенчатого фильтра с обратной связью. Здесь d есть временная задержка в семплах и g — коэффициент обратной связи.

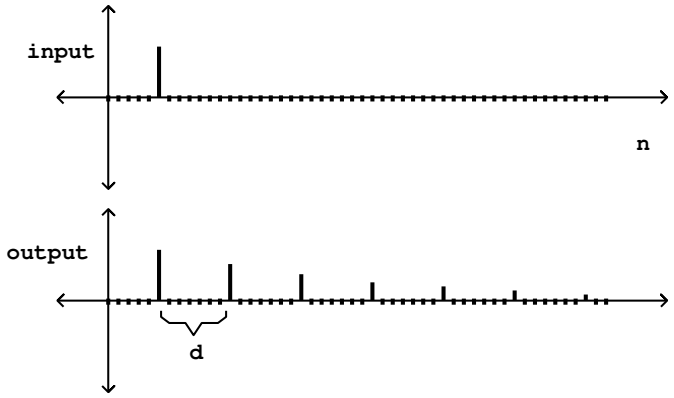


Рис. 7.8: Анализ во временной области гребенчатого фильтра с рециркуляцией, использующий импульс в качестве входа.

Заметим, что мы выбрали коэффициент усиления g, который меньше единицы по абсолютной величине. Если бы мы выбрали коэффициент усиления больше единицы (или меньше -1), каждый следующий эхо-сигнал имел бы большую величину, чем предыдущий. Вместо экспоненциального убывания, происходящего на рисунке, эхо экспоненциально возрастало бы. Сеть с обратной связью, выходной сигнал которой, в конечном счёте, спадает до нуля, после того как входной сигнал заканчивается, называется *стабильной*; а сеть, выход которой неограниченно растёт — *нестабильной.*

Мы можем также проанализировать гребенчатый фильтр с рециркуляцией в частотной области. Теперь ситуация становится чрезвычайно сложна для анализа, использующего действительные синусоиды и, поэтому, мы получаем первое большое вознаграждение от введения комплексных чисел, которые сильно упрощают анализ.

Если, как ранее, мы пошлём на вход сигнал

$$X[n] = Z^n$$

с $|Z| = 1$, мы можем записать выход как

$$Y[n] = (1 + gZ^{-d} + g^2 Z^{-2d} + \cdots)X[n].$$

Здесь слагаемые в сумме соответствуют ряду дискретных эхо-сигналов. Отсюда вытекает, что амплитуда выхода есть

$$H = 1 + gZ^{-d} + \left(gZ^{-d}\right)^2 + \cdots$$

Это сумма геометрической прогрессии; мы можем просуммировать её с помощью стандартного метода. Во-первых, умножим обе стороны на gZ^{-d}:

$$gZ^{-d}H = gZ^{-d} + \left(gZ^{-d}\right)^2 + \left(gZ^{-d}\right)^3 + \cdots$$

и вычтем из первоначального уравнения, что даст:

$$H - gZ^{-d}H = 1.$$

Затем решаем относительно H:

$$H = \frac{1}{1 - gZ^{-d}}.$$

Более быстрый (но несколько менее интуитивно понятный) метод получения того же результата состоит в разборе самой рециркулирующей сети для получения уравнения для H. Он состоит в следующем. Назовём вход $X[n]$ и выход $Y[n]$. Сигнал, поступающий на линию задержки есть выход $Y[n]$, а прошедший через линию задержки и умножитель есть

$$Y[n] \cdot gZ^{-d}.$$

Это выражение, плюс входной сигнал, есть, в точности, опять выходной сигнал, так что

$$Y[n] = X[n] + Y[n] \cdot gZ^{-d}$$

и деля на $X[n]$, принимая во внимание $H = Y[n]/X[n]$, получаем:

$$H = 1 + HgZ^{-d}.$$

Это выражение эквивалентно полученному ранее равенству для H.

Теперь мы бы хотели построить график частотного отклика (коэффициента усиления, как функции частоты), также как мы сделали для гребенчатого фильтра без обратной связи на рис. 7.6. Для этого опять требует, чтобы мы сделали предварительное представление в комплексной плоскости. Мы бы хотели оценить модуль H, равный

$$|H| = \frac{1}{|1 - gZ^{-d}|},$$

где мы использовали свойство мультипликативности модулей для того, чтобы вывести, что модуль обратного от (комплексного) числа равен обратной величине от его (действительного) модуля. Рисунок 7.9 показывает ситуацию графически. Коэффициент усиления $|H|$ равен обратной величине к длине отрезка, соединяющего точки 1 и gZ^{-d}. Рисунок 7.10 показывает график частотного отклика $|H|$ как функции угловой частоты $\omega = \angle(Z)$.

Рисунок 7.9 может быть использован для анализа поведения частотного отклика $|H(\omega)|$ как функции от g. Как высота, так и ширина полосы пиков зависят от g. Максимальное значение $|H|$ достигается, когда

$$Z^{-d} = 1.$$

Это происходит на частотах $\omega = 0$, $2\pi/d$, $4\pi/d$, $\ldots$ как для простейшего гребенчатого фильтра, приведённого выше. На этих частотах коэффициент усиления достигает

$$|H| = \frac{1}{1 - g}.$$

Следующий важный вопрос состоит в ширине полос пиков в частотном отклике. Так что мы бы хотели найти синусоиды W^n, с частотой $\angle(W)$, приводящей к величине $|H|$, которая, скажем, на 3 децибела ниже максимума. Для того чтобы проделать это, мы вернёмся к рис. 7.9 и попытаемся поместить W так, чтобы расстояние от точки 1 до точки gW^{-d} было примерно в $\sqrt{2}$ раз больше, чем расстояние от 1 до g (поскольку $\sqrt{2} : 1$ даёт отношение, приблизительно соответствующее 3 децибелам).

Мы делаем это подбирая мнимую часть gW^{-d} приблизительно равной $1 - g$ или этой же величине со знаком минус, делая приблизительно равнобедренный прямоугольный треугольник между точками 1, $1-g$ и gW^{-d}. (Здесь мы предполагаем, что g есть, как минимум, $2/3$ или около того; в противном случае, эта аппроксимация

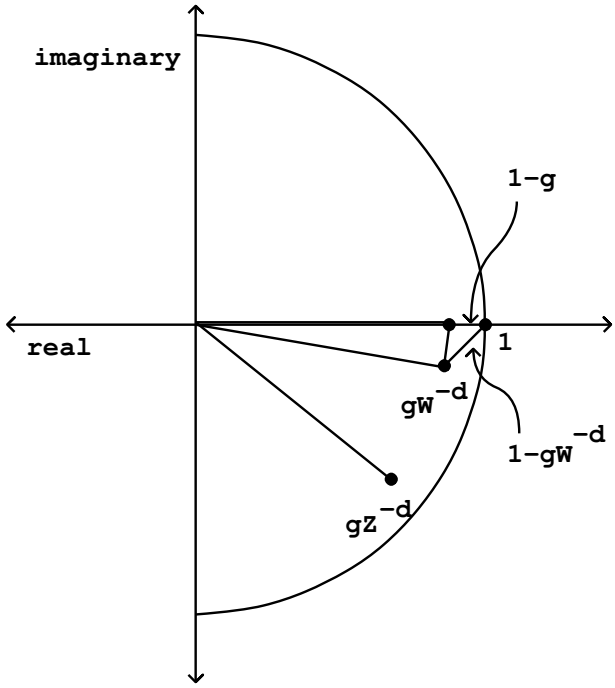


Рис. 7.9: Диаграмма в комплексной плоскости, показывающая аппроксимацию выходного коэффициента усиления $|H|$ гребенчатого фильтра с рециркуляцией для трёх различных частот: нулевой и равной аргументам двух комплексных чисел W и Z с единичными модулями; W выбрано так, чтобы дать коэффициент усиления, приблизительно на 3 децибела ниже пика.

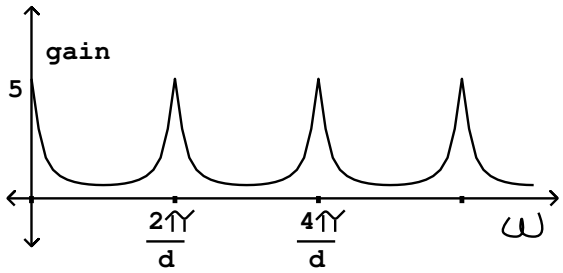


Рис. 7.10: Частотный отклик для гребенчатого фильтра с рециркуляцией при $g = 0.8$. Пиковый коэффициент усиления $1/(1-g) = 5$. Пики гораздо уже, чем для гребенчатого фильтра без рециркуляции.

не является хорошей). Гипотенуза прямоугольного равнобедренного треугольника равна произведению $\sqrt{2}$ на длину катета, так что усиление падает на такой же коэффициент по сравнению со своим максимумом.

Теперь мы сделаем другую аппроксимацию, предполагающую, что мнимая часть gW^{-d} приблизительно равна углу в радианах, на который она отстаёт от действительной оси:

$$\pm(1-g) \approx \mathrm{im}(gW^{-d}) \approx \angle(W^{-d}).$$

Так что окрестность каждого пика, в которой коэффициент усиления отличается не более, чем на 3 децибела от максимального, приблизительно составляет

$$(1-g)/d$$

(в радианах) с каждой стороны пика. Ширина полос сужается (и пики фильтра становятся более резкими), когда g достигает своего максимума, равного 1.

Как и для гребенчатого фильтра без обратной связи §7.3, зубья гребня сближаются вместе для больших значений задержки d. С другой стороны, задержка $d = 1$ (наиболее короткая из возможных) даёт только один зубец (на нулевой частоте) ниже частоты Найквиста π (следующий зубец, на 2π, соответствует опять частоте ноль, после свёртывания). Таким образом, гребенчатый фильтр с рециркуляцией и $d = 1$ представляет из себя всего лишь фильтр, пропускающий низкие частоты. Сети с односемпловыми задержками будут базой для конструкции многих других видов цифровых фильтров в Главе 8.

7.5 Сохранение мощности и сети с задержкой со сложной структурой

Такие же методы будут работать при анализе любой сети с задержкой, хотя для более сложных сетей становится сложнее охарактеризовать результаты или построить сеть, имеющую специфические желаемые свойства. Другая точка зрения может иногда быть с пользой привнесена для анализа ситуации, особенно когда требуются плоские частотные отклики, или сами по себе, или для того, чтобы гарантировать, что сети с рециркуляцией остаются устойчивыми при коэффициентах обратной связи, близких к единице.

Центральный факт, который мы будем использовать, состоит в том, что если любая сеть с задержкой, с одним или несколькими входами и выходами, построена так, что её выходная мощность (усреднённая по времени) всегда равна её входной мощности, такая сеть должна иметь плоский частотный отклик[1]. Это почти тавтология; если вы поместите синусоиду любой частоты на один из входов, вы получите синусоиды той же частоты на выходах, и сумма мощностей на всех выходах будет равна мощности входа, так что коэффициент усиления, определённый подходящим образом, равен в точности единице.

[1] то есть с коэффициентом усиления сети, практически не зависящим от частоты.

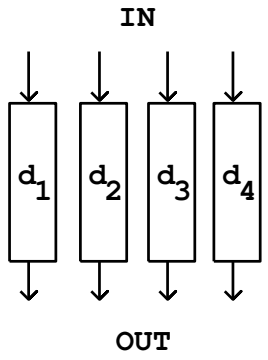


Рис. 7.11: Первый основной строительный блок для унитарных сетей с задержкой: параллельные линии задержки.

Для того чтобы работать с сохраняющими мощность сетями с задержкой, нам понадобится точное определение "полной средней мощности". Если имеется только один сигнал (назовём его $x[n]$), средняя мощность определяется как

$$P(x[n]) = \left[|x[0]|^2 + |x[1]|^2 + \cdots + |x[N-1]|^2\right]/N,$$

где N достаточно большое число, так что любые флуктуации амплитуды убираются усреднением. Это определение работает как для комплексных сигналов, так и для вещественных. Усреднённая полная мощность нескольких цифровых аудиосигналов есть, в точности, сумма мощностей индивидуальных сигналов:

$$P(x_1[n], \ldots, x_r[n]) = P(x_1[n]) + \cdots + P(x_r[n]),$$

где r есть число комбинируемых сигналов.

Оказывается, что широкий диапазон интересных сетей с задержкой имеет то свойство, что полная мощность выхода равна полной мощности входа; они называются *унитарными*. Для начала, мы можем поместить любое число задержек параллельно, как показано на рис. 7.11. Какова бы ни была полная мощность входов, полная мощность выходных сигналов равна ей.

Второе семейство преобразований, сохраняющих мощность образуют повороты и отражения сигналов $x_1[n], \ldots, x_r[n]$ при рассматривании их, в каждый фиксированный момент времени n, как r координат точки в r-мерном пространстве. Поворот или отражение должны оставлять начало координат $(0, \ldots, 0)$ на месте.

Для каждого номера семпла n, полный вклад в усреднённую мощность сигнала пропорционален

$$|x_1|^2 + \cdots + |x_r|^2.$$

Это, в точности, пифагоровское расстояние от начала координат до точки $(x_1, \ldots, x_r)$. Поскольку повороты и отражения являются преобразованиями, сохраняющими расстояние, расстояние до начала координат перед преобразованием должно равняться расстоянию до начала координат после него. Таким образом, полная мощность набора сигналов должна сохраняться при вращении.

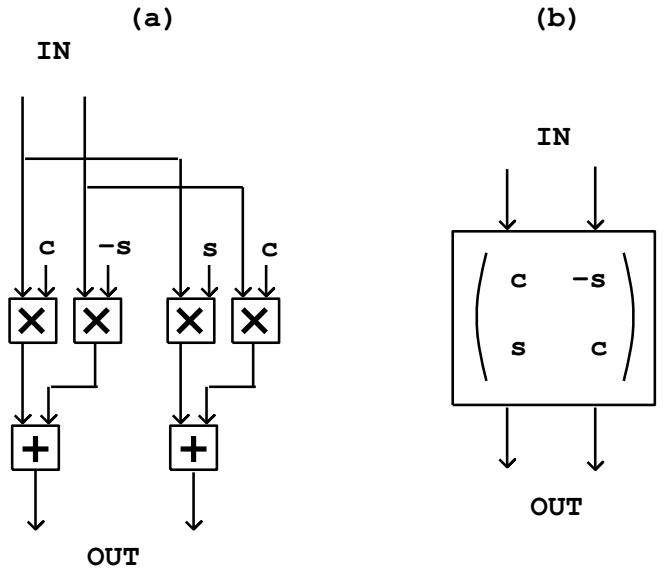


Рис. 7.12: Второй основной строительный блок для унитарных сетей с задержкой: поворот двух цифровых аудиосигналов. Часть (a) показывает точный вид преобразования; (b) показывает его как матричную операцию.

Рисунок 7.12 показывает матрицу вращения, действующую на два сигнала. В части (a) преобразование показано в точности. Если входные сигналы есть $x_1[n]$ и $x_2[n]$, выходами будут:

$$y_1[n] = cx_1[n] - sx_2[n]$$
$$y_1[n] = cx_1[n] - sx_2[n],$$

где c, s даются формулой

$$c = \cos(\theta)$$
$$s = \sin(\theta)$$

для угла поворота θ. Если рассматривать сигналы как точки в декартовой системе координат, точка (y_1, y_2) есть в точности точка (x_1, x_2), повёрнутая против часовой стрелки на угол θ. Две точки имеют, таким образом, одинаковое расстояние до начала координат:

$$|y_1|^2 + |y_2|^2 = |x_1|^2 + |x_2|^2$$

и, тем самым, два выходных сигнала имеют ту же полную мощность, как два входных сигнала.

Для альтернативного описания двумерного поворота, рассмотрим комплексные числа $X = x_1 + x_2 i$ и $Y = y_1 + y_2 i$. Преобразование, рассмотренное выше, эквивалентно требованию

$$Y = XZ,$$

где Z есть комплексное число с единичным модулем и аргументом θ. Поскольку $|Z| = 1$, получаем, что $|X| = |Y|$.

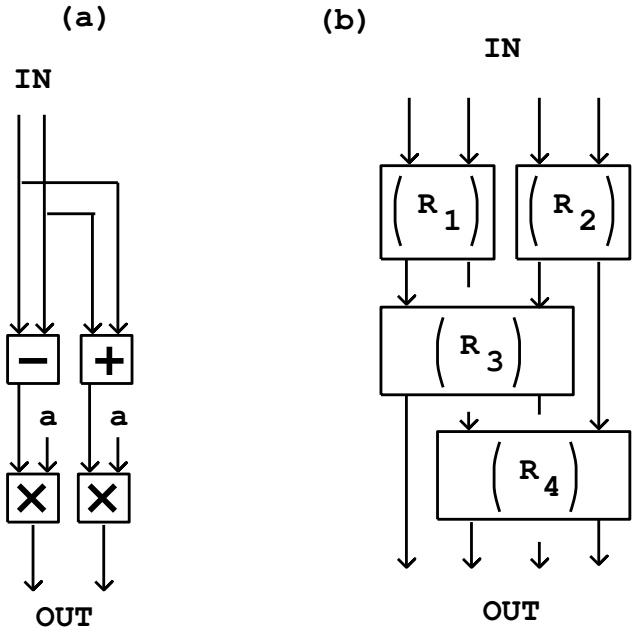


Рис. 7.13: Детали матричных операций поворота (и отражения): (a) поворот на угол $\theta = \pi/4$, такой что $a = \cos(\theta) = \sin(\theta) = 1/\sqrt{2} \approx 0.7071$; (b) комбинация двумерных вращений для осуществления вращений в больших размерностях.

Если мы выполним поворот пары сигналов и далее обратим один из них (но не оба), результатом будет *отражение*. Оно также сохраняет полную мощность сигнала, поскольку мы можем обратить любой или весь набор сигналов без изменения полной мощности. Для размерности два, отражение появляется как преобразование вида

$$\begin{aligned} y_1[n] &= cx_1[n] + sx_2[n] \\ y_2[n] &= sx_1[n] - cx_2[n]. \end{aligned}$$

Специальная полезная матрица поворота получается установкой $\theta = \pi/4$, так что $s = c = 1/\sqrt{2}$. Это позволяет нам упростить вычисления, как показано на рис. 7.13 (часть a), поскольку каждый сигнал должен только быть умножен на одну величину $c = s$.

Более сложные повороты или отражения более чем двух сигналов могут быть проделаны последовательным их попарным вращением и/или отражением. Например, на рис. 7.13 (часть b), четыре сигнала комбинируются в пары, на двух последовательных этапах, так что в конце каждый сигнальный вход направляется на все выходы. Мы могли бы сделать то же самое с восемью сигналами (используя три этапа) и так далее. Более того, если мы используем специальный угол $\pi/4$, все входные сигналы будут давать одинаковый вклад на все выходы.

Любая комбинация задержек и матриц поворотов, последовательно применённая

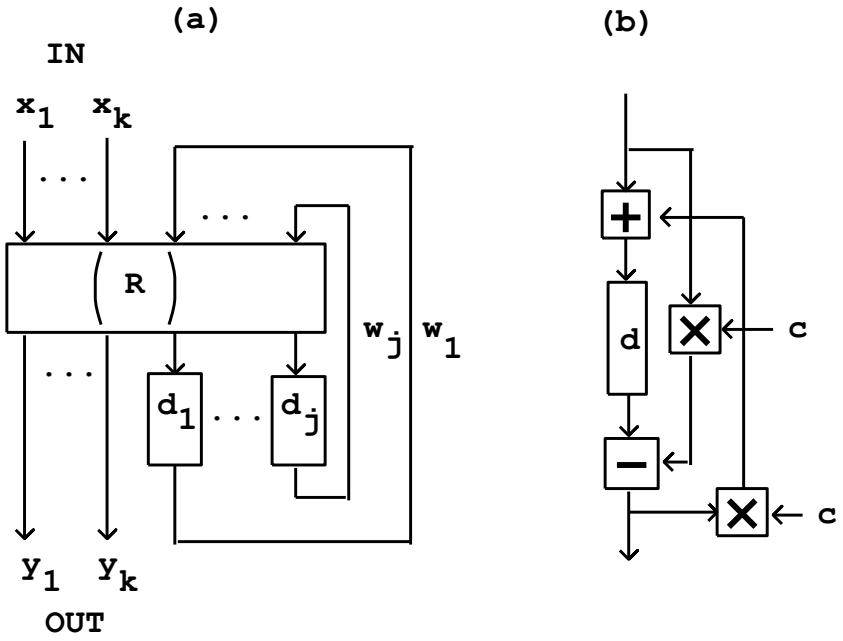


Рис. 7.14: Плоский частотный ответ в сетях с рециркуляцией: (a) в общем, с использованием матрицы поворота R; (b) пропускающая все частоты конфигурация.

к набору аудиосигналов, приведёт к плоскому частотному отклику, к которому приводит каждая отдельная операция. Это уже позволяет нам генерировать бесконечно много сетей с задержкой с плоским частотным откликом, но, до сих пор, ни в одной из них не было рециркуляции. Третья операция, показанная на рис. 7.14, позволяет создать сети с рециркуляцией, которые по-прежнему дают плоские частотные отклики.

Часть (a) рисунка показывает общую схему. Предполагается, что преобразование R является любой комбинацией задержек и матриц преобразований, сохраняющих общую мощность. Сигналы $x_1, \ldots, x_k$ поступают на цепь с единой задержкой и возникают выходные сигналы $y_1, \ldots, y_k$. Некие другие сигналы $w_1, \ldots, w_j$ (где j не обязательно равно k) появляются на выходе преобразования R и направляются обратно на его вход.

Если R действительно сохраняет мощность, полная входная мощность (сумма мощностей сигналов $x_1, \ldots, x_k$ и $w_1, \ldots, w_j$) должна равняться выходной мощности (сумма мощностей сигналов $y_1, \ldots, y_k$ и $w_1, \ldots, w_j$), и вычитая все w из равенства, мы находим, что полные мощности входа и выхода равны.

Если мы положим $j = k = 1$, так что имеется один x, y и w, зададим операцию R как поворот на θ с последующей задержкой на d семплов на выходе W, результатом является хорошо известный *всепропускающий* фильтр. Устанавливая $c = \cos(\theta)$, мы можем показать, что это эквивалентно сети, показанной в части (b) рисунка. Всепропускающие фильтры имеют много приложений, к некоторым из которых мы обратимся позднее в этой книге.

7.6 Искусственная реверберация

Искусственная реверберация широко используется для улучшения звука записей, но также имеет широкий диапазон других музыкальных приложений [DJ85, стр. 289-340]. Реверберация в реальных естественных пространствах возникает из сложных звуковых отражений от стен и других объектов пространства. Огромным упрощением является моделирование этого процесса с использованием рециркулирующих дискретных сетей с задержкой. Несмотря на это, при больших усилиях моделирование реверберации с использованием рециркулирующих линий задержки может давать хорошие результаты.

Главная идея состоит в идеализации любой комнаты (или другого пространства с реверберацией) в виде набора параллельных линий задержки, которые моделируют воздушное пространство комнаты. В каждой точке на стене комнаты заканчиваются многочисленные прямые пути, по которым переносится звук к этой точке; затем звук разбивается на другие многочисленные пути, начинающиеся из этой точки, и оканчивающиеся со временем в какой-нибудь другой точке стены.

Хотя стена (и воздух, через который мы перемещались для ее достижения) поглощает звук, некоторая часть падающей мощности отражается и переносится к другой стене. Если большая часть энергии рециркулирует, реверберация в комнате продолжается долгое время; если рециркулирует вся энергия, реверберация продолжается бесконечно долго. Если на некоторой частоте стены отражают больше энергии в целом, чем они получили, возникнет нестабильная обратная связь, “заводка”; такого никогда не случается в реальных пространствах (этому препятствует закон сохранения энергии), но может случаться при некорректно спроектированной искусственной реверберации.

Для создания искусственной реверберации с использованием сетей с задержкой, мы должны выставить два конкурирующих требования. Во-первых, линии задержки должны быть достаточно длинными, чтобы предотвратить возникновение *окрашивания* выходного сигнала в результате действия гребенчатого фильтра. (Даже если не использовать простой гребенчатый фильтр §7.4, частотный отклик будет стремиться к тому, чтобы иметь пики и впадины, размер которых меняется обратно пропорционально полному времени задержки.) С другой стороны, мы не должны слышать индивидуальные эхо-сигналы; *плотность эхо-сигналов* должна в идеале быть не меньше тысячи раз в секунду.

Преследуя эти цели, мы подбираем некоторое число линий задержки и соединяем их выходы с их входами. Линия обратной связи (соединение выходов обратно с входами задержек) должна иметь совокупное усиление, которое мягко варьируется как функция частоты и никогда не превышает единицу для любой частоты. Хорошей отправной точкой является придание линии обратной связи плоской частотной характеристики и коэффициента усиления чуть меньше единицы; это делается с помощью матриц вращения.

В идеале это всё, что нам нужно сделать, но в реальности мы не всегда хотим использовать тысячи линий задержки, которые бы соответствовали в нашей мо-

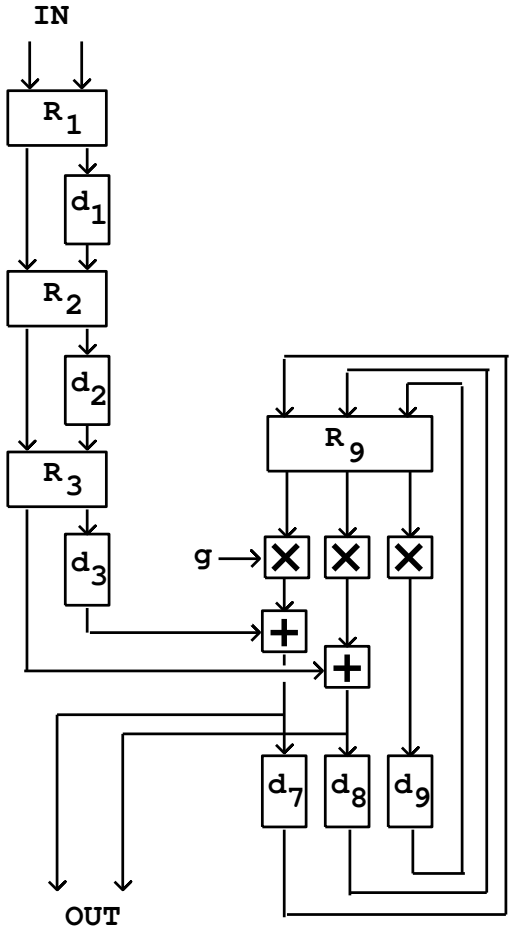


Рис. 7.15: Устройство ревербератора, использующего преобразования, сохраняющие мощность и рециркулирующие задержки.

дели путям между любыми возможными парами точек на стенах. На практике обыкновенно используется от четырёх до шестнадцати линий задержки для моделирования комнаты. Это упрощение иногда уменьшает плотность эхо-сигналов ниже требуемой величины, так что мы можем использовать большее число линий задержки на входе рециркулирующей сети, чтобы увеличить плотность.

Рисунок 7.15 показывает создание простого ревербератора с использованием такого принципа. Входящий звук, показанный в этом примере как два отдельных сигнала, сначала "уплотняется"последовательной задержкой одного из двух сигналов, а затем смешивает их с использованием матрицы поворота. На каждом этапе число эхо-сигналов первоначального сигнала удваивается; обыкновенно мы используем от 6 до 8 этапов для создания от 64 до 256 эхо-сигналов, все с полной задержкой от 30 до 40 миллисекунд. Рисунок показывает три таких этапа.

Далее включается рециркулирующая часть ревербератора. После уплотнения входной сигнал направляется к банку параллельных линий задержки, и их выходы опять смешиваются с помощью матрицы поворота. Смешанные выходы усиливают-

ся с коэффициентом $g \leqslant 1$ и направляются назад на линии задержки для создания сети с рециркуляцией.

Величина g управляет временем реверберации. Если средняя длина рециркулирующих линий задержки есть d, тогда любой входящий звук усиливается с коэффициентом g после времени задержки d. Через время t сигнал рециркулирует t/d раз, теряя $20\log_1 0(g)$ децибел каждый раз, так что итоговое усиление в децибелах составит

$$20\frac{t}{d}\log_{10}(g).$$

Естественной мерой времени реверберации (RT) будет время, за которое усиление упадёт на 60 децибел:

$$20\tfrac{\mathrm{RT}}{d}\log_{10}(g) = -60\mathrm{RT} = \tfrac{-3d}{\log_{10}(g)}.$$

Если g равно единице, эта формула даёт ∞, поскольку логарифм единицы равен нулю.

Приведенная выше схема является основой для построения многих современных ревербераторов. Было предложено много улучшений этого основополагающего дизайна. Следующий наиболее важный шаг состоял бы во введении фильтров на пути рециркуляции, так чтобы на высоких частотах сделать затухание быстрее, чем на низких; это легко осуществляется с помощью очень простого фильтра, пропускающего низкие частоты; но мы не будем использовать здесь этот метод, не рассмотрев прежде необходимую для него теорию фильтров.

В общем, использование такого подхода для создания ревербератора означает множество сложных выборов: времён задержки, коэффициентов усиления и фильтров. Горы литературы были опубликованы по этой теме; у Барри Блессера есть хороший обзор этой темы [Ble01]. Об устройстве и настройке ревербераторов известно гораздо больше, чем опубликовано; точное описание устройства часто держится в секрете по коммерческим причинам. В целом, процесс создания включает кропотливую и длинную настройку путём проб и ошибок, с учётом критических замечаний.

7.6.1 Управление ревербераторами

Искусственная реверберация используется практически повсеместно при записи или усилении звука для украшения общего звучания. Однако, что более интересно, реверберация может использоваться как источник звука сама по себе. Специальный вид бесконечной реверберации полезен для захвата живых звуков и растяжения их во времени.

Чтобы сделать это на практике, необходимо открыть вход ревербератора на довольно короткий промежуток времени, в ходе которого входной звук не изменяется слишком быстро. Если вход бесконечного ревербератора оставлен открытым

слишком долго, звук будет набираться и быстро станет невнятной массой. Для “бесконечной реверберации” ноты живого инструмента лучше подождать конца фрагмента атаки в ноте и далее отправить приблизительно 1/2 секунды устойчивого состояния ноты в ревербератор. Можно построить аккорды от монофонического инструмента, последовательно открывая вход для различных моментов устойчивой высоты.

Рисунок 7.16 показывает, как это может быть проделано на практике. Два наиболее важных инструмента управления: входной уровень ревербератора и коэффициент усиления обратной связи. Для захвата звука мы устанавливаем коэффициент усиления обратной связи на единицу (бесконечное время реверберации) и на мгновение открываем вход в момент времени t_1. Для добавления других звуков к уже захваченным мы просто в нужный момент снова добавляем входное усиление (в момент времени t_2 на рисунке, например). Наконец, мы можем стереть рециркулирующий звук, устанавливая коэффициент усиления обратной связи меньше единицы (в момент времени t_3), таким образом одновременно плавно уменьшая выходной уровень и опустошая ревербератор. Дальнейшее уменьшение коэффициента обратной связи ускорит затухание выхода.

7.7 Изменяемые и дробные временные задержки

Как любой аудиосинтез или техника преобразования, сети с задержкой становятся более мощными и интересными, если их характеристики могут быть изменены во времени. Параметры усиления (такие как g в гребенчатом фильтре с рециркуляцией) могут управляться генераторами огибающей, которые изменяют их без щелчков или других артефактов. Времена задержек (как d ранее) не так просто поменять гладким образом по двум причинам.

Во-первых, мы определили временные сдвиги только для целых значений d, поскольку для дробных величин d такие выражения как $x[n-d]$ не определены, если $x[n]$ определены *только* для целых значений n. Для того чтобы делать дробные задержки, мы должны определить какую-нибудь подходящую интерполяционную схему. И если мы хотим варьировать d гладко по времени, это не даст хороших результатов для простого прыжка от одного целого значения к следующему.

Во-вторых, даже когда мы достигли достаточно гладко меняющихся времён задержки, артефакты, вызванные изменением времени задержки, становятся значительными даже для очень маленьких относительных скоростей изменения; в то время как в большинстве случаев вы можете без проблем линейно изменить параметр управления амплитудой между любыми двумя значениями за 30 миллисекунд, изменение задержки лишь на один семпл из каждых ста создаёт очень весомый сдвиг по высоте. В действительности задержку часто меняют сознательно, для того чтобы услышать артефакты, просто случайным переходом от одного специфического времени задержки к другому.

С первой сложностью (дробные задержки) можно бороться с использованием

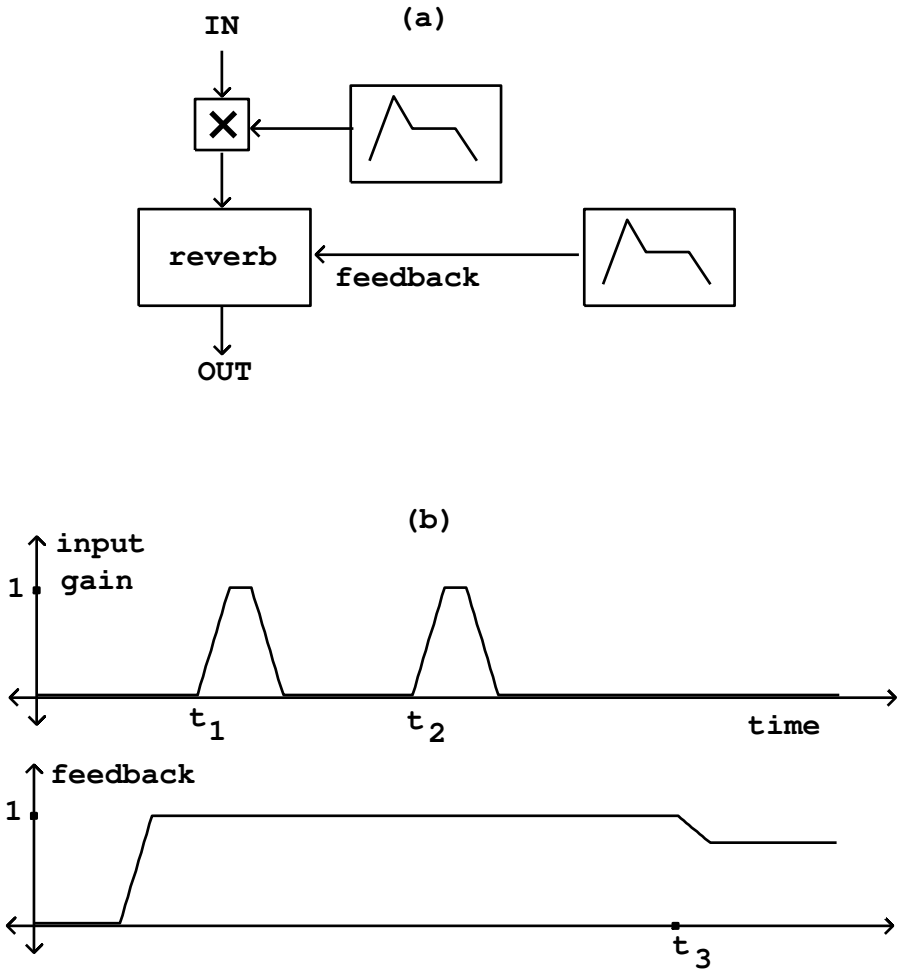


Рис. 7.16: Управление ревербератором для селекционного захвата звуков: (a) схема; (b) примеры того, как управлять входным усилением и обратной связью для захвата звуков в моменты времени t_1 и t_2 и удержания их вплоть до момента времени t_3.

интерполяционной схемы точно таким же образом, как при просмотре волновой таблицы (§2.5). Например, предположим, что мы хотим задержку в $d = 1.5$ семпла. Для каждого n мы должны оценить величину $x[n - 1.5]$. Мы могли бы сделать это, используя стандартную четырёхточечную интерполяцию, проводя кубическую полиномиальную кривую через "известные" точки $(0, x[n])$, $(1, x[n-1])$, $(2, x[n-2])$, $(3, x[n-3])$ и вычисляя после этого значение полинома в точке 1.5. Циклическое повторение этой процедуры для каждого значения n даёт задержанный сигнал.

Такая четырёхточечная интерполяционная схема может быть использована для любой задержки, как минимум, в один семпл. Задержки менее одного семпла не могут быть вычислены этим путём, поскольку нам нужно как минимум две входных точки рядом с требуемой задержкой. Они были доступны в примере выше, но для времени задержки, например, в 0.5 семпла, нам бы потребовалось значение $x[n + 1]$, которое лежит в будущем.

Точность оценки может быть далее усилена путём использования интерполяционных схем более высокого порядка. Однако имеется компромисс между качеством и вычислительной эффективностью. Более того, если мы переходим к схемам интерполяции более высокого порядка, минимально возможное время задержки будет возрастать, что вызывает проблемы в некоторых ситуациях.

Вторая рассматриваемая сложность состоит в артефактах (желаемых или нежелаемых), которые возникают от изменений в линиях задержки. В общем, скачкообразное изменение времени задержки будет приводить к скачкообразному изменению выходного сигнала, поскольку он в результате прерывается в одной точке и делает скачек в другую. Если входной сигнал является синусоидой, результатом будет разрывное изменение фазы.

Если требуется иногда менять время задержки между фиксированными величинами (например, в начале музыкальных нот), тогда мы можем использовать методы, которые мы ввели в §4.3 для управления спорадическими разрывами. В сущности, все эти методы работают с помощью выключения выходного сигнала тем или иным путём. С другой стороны, если требуется, чтобы время задержки менялось непрерывно (пока мы прослушиваем выход), тогда мы должны учесть артефакты, которые появляются от изменений.

Рисунок 7.17 показывает взаимосвязь между входным и выходным временем в переменной линии задержки. Предполагается, что линия задержки имеет фиксированную максимальную длину D. В каждом семпле выхода (соответствующем точке на горизонтальной оси), мы выводим один (возможно, интерполированный) семпл входного сигнала линии задержки. Вертикальная ось показывает, какой семпл (целый или дробный) входного сигнала использовать. Если установить, что n обозначает номер выходного семпла, вертикальная ось показывает величину $n - d[n]$, где $d[n]$ есть (зависящая от времени) задержка в семплах. Если обозначить момент времени входного семпла как

$$y[n] = n - d[n],$$

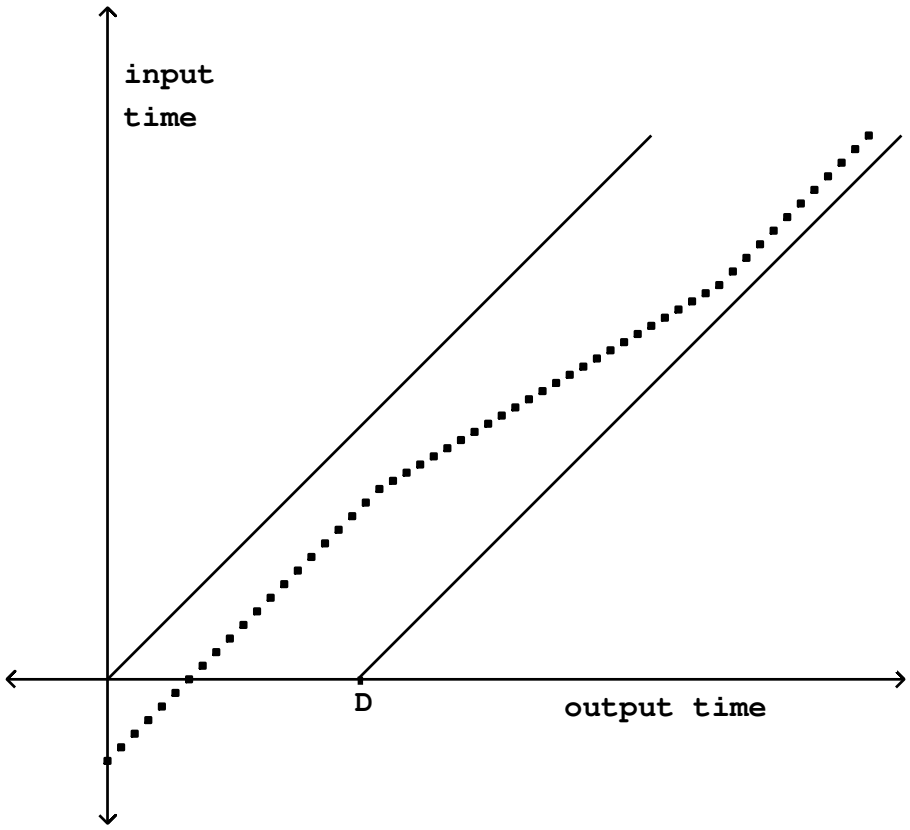


Рис. 7.17: Линия задержки переменной длины, чей выход является входом для некоторого предыдущего момента времени. Выходные семплы не могут появляться раньше входных, а также позже, чем максимальная длина D линии задержки. Наклон входной/выходной кривой управляет моментальной транспозицией выходного сигнала.

тогда выход линии с задержкой есть

$$z[n] = x[y[n]],$$

где сигнал x вычисляется в точке $y[n]$, с подходящей интерполяцией, в случае, когда $y[n]$ не является целым числом. Это в точности формула просмотра волновой таблицы (стр. 26). Мы можем использовать все свойства техники просмотра волновой таблицы для предсказания поведения переменных линий задержки.

Единственная остающаяся разница между линиями задержки и волновыми таблицами — материал в линии задержки постоянно обновляется. Мы не только не можем считывать будущее, но, если линия задержки имеет D семплов в длину, мы не можем считать более, чем за D семплов в прошлое:

$$0 < d[n] < D$$

или, умножая на минус единицу и добавляя n к каждой из сторон,

$$n > y[n] > n - D.$$

Это последнее соотношение определяет область между диагональными линиями на рис. 7.17; функция $y[n]$ должна оставаться внутри этой полосы.

Возвращаясь к §2.2, мы можем использовать *формулы моментального транспонирования для волновых таблиц* (2.3) для вычисления транспозиции $t[n]$ выхода. Это даёт *формулу моментального транспонирования для линий задержки*:

$$t[n] = y[n] - y[n-1] = 1 - (d[n] - d[n-1]). \tag{7.1}$$

Если $d[n]$ не меняется при замене n, коэффициент транспозиции равен 1 и звук возникает из линии задержки с такой же скоростью, с какой пришёл на неё. Но если время задержки растёт как функция n, получающийся звук транспонируется вниз, и если $d[n]$ убывает — вверх.

Это называется *эффектом Допплера* и встречается также и в природе. Воздух, через который проходит звук, может иногда рассматриваться как линия задержки. Изменение длины линии задержки соответствует движению слушателя по направлению к или от стационарного источника звука; эффект Допплера от изменения длины пути работает в точности так же в линии задержки, как в физическом воздухе.

Возвращаясь к рис. 7.17, мы можем предсказать, что нет сдвига высоты в начале, но далее, когда наклон пути уменьшается, высота падает в некотором временном интервале перед тем, как вернуться к первоначальной высоте (когда наклон становится первоначальным). Время задержки может быть подобрано так, чтобы получить требуемое транспонирование, но чем больше величина транспонирования, тем меньше мы можем ее удерживать, перед тем как достигнем низа или верха диагональной области.

7.8 Точность воспроизведения линий задержки с интерполяцией

Поскольку линии задержки в действительности производят просмотр волновой таблицы, переменные линии задержки приводят к искажению обрабатываемых ими сигналов. Помимо этого, может возникнуть более неочевидная проблема даже когда длина линии задержки не меняется: частотный отклик в реальных ситуациях никогда не является идеально плоским для линии задержки, чья длина не является целой.

Если время задержки меняется от семпла к семплу, появляются результаты искажений описанные в §2.5. Для их использования мы предположим, что входной сигнал линии задержки может быть разбит на синусоиды, и рассмотрим отдельно, что случится с каждой индивидуальной синусоидой. Мы можем использовать Таблицу 2.1 для предсказания уровня RMS скомбинированных результатов искажения для переменной линии задержки с интерполяцией.

Если мы, предположим, хотим использовать четырёхточечную интерполяцию, то для синусоид с периодами длиннее 32 семплов (то есть для частот ниже 1/16 частоты Найквиста) искажения будут ниже на 96 дб или еще меньше (едва ли заметным). При частоте семплирования 44.1 кГц эти периоды соответствовали бы частотам до ≈ 1400 герц. На более высоких частотах качество падает, и выше $\approx 1/4$ частоты Найквиста результаты искажения, которые всего на 50 дб ниже, будут, возможно, слышимы.

Ситуация со сложным тоном зависит в основном от амплитуд и частот его верхних гармоник. Предположим, например, что гармоники, которые выше 5000 герц, по крайней мере на 20 дб ниже его самой сильной гармоники, и что выше 10000 герц они падают ещё на 60 дб. Тогда, при грубой оценке результатов, искажения в диапазоне 5000–10000 будут ограничены ≈ -68 дб, а соответствующие величинам выше 10000 герц будут ограничены ≈ -75 дб (поскольку наихудшая картина в таблице даёт около -15 дб, и это должно быть добавлено к силе привлечённых гармоник).

Если высокочастотная составляющая входного сигнала оказывается дающей неприемлемые результаты искажения, в общем, эффективнее увеличивать частоту семплирования, чем число точек интерполяции. Для периодов больших, чем 4 семпла, удваивание периода (с помощью удваивания частоты семплирования, например) уменьшает искажение приблизительно на 24 дб.

Частотный отклик линии задержки с четырёхточечной интерполяцией почти плоский вплоть до половины частоты Найквиста, но после этого он быстро спадает. Предположим (подбирая худший случай), что задержка установлена на полпути между двумя целыми числами, скажем 1.5. Кубическая интерполяция даёт:

$$x[1.5] = \frac{-x[0] + 9x[1] + 9x[2] - x[3]}{8}.$$

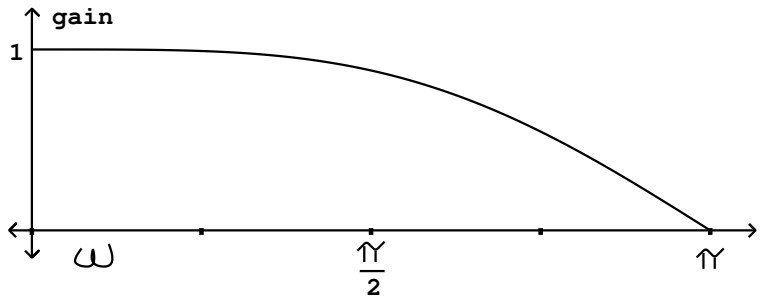


Рис. 7.18: Коэффициент усиления линии задержки с четырёхточечной интерполяцией с задержкой на половине пути между двумя целыми числами. Коэффициент усиления константы (DC) равен единице.

Теперь пусть $x[n]$ есть (действительнозначная) синусоида единичной амплитуды с угловой частотой ω, чья фаза равна нулю в 1.5:

$$x[n] = \cos(\omega \cdot (n - 1.5)).$$

Вычислим $x[1.5]$, используя формулу выше:

$$x[1.5] = \frac{9\cos(\omega/2) - \cos(3\omega/2)}{4}.$$

Это — пиковое значение синусоиды, которое возвращается из линии задержки, и, поскольку входная пиковая амплитуда была равна единице, это показывает частотный отклик линии задержки. Он изображён на рис. 7.18. Для половины частоты Найквиста ($\omega = \pi/2$) коэффициент усиления равен около -1 дб, что просто ощущается как падение амплитуды. На частоте Найквиста самой по себе, однако, коэффициент усиления равен нулю.

Как и с результатами для искажения, частотный отклик улучшается радикально при удваивании частоты семплирования. Если мы запускаем нашу задержку с частотой семплирования 88200 герц, вместо стандартной 44100, мы получим только около 1 дб спада вплоть до 20000 герц.

7.9 Сдвиг высоты звука

Наиболее популярное использование переменных линий с задержкой состоит в изменении высоты входящего звука с использованием эффекта Допплера. Может потребоваться переменное изменение высоты (случайное или периодическое, например) или, иначе, поддержка фиксированного музыкального интервала транспонирования в течение некоторого времени.

Возвращаясь к рис. 7.17, мы видим, что с одной переменной линией задержки мы можем поддерживать любой желаемый сдвиг высоты для ограниченного промежутка времени, и если мы хотим поддерживать фиксированную транспозицию,

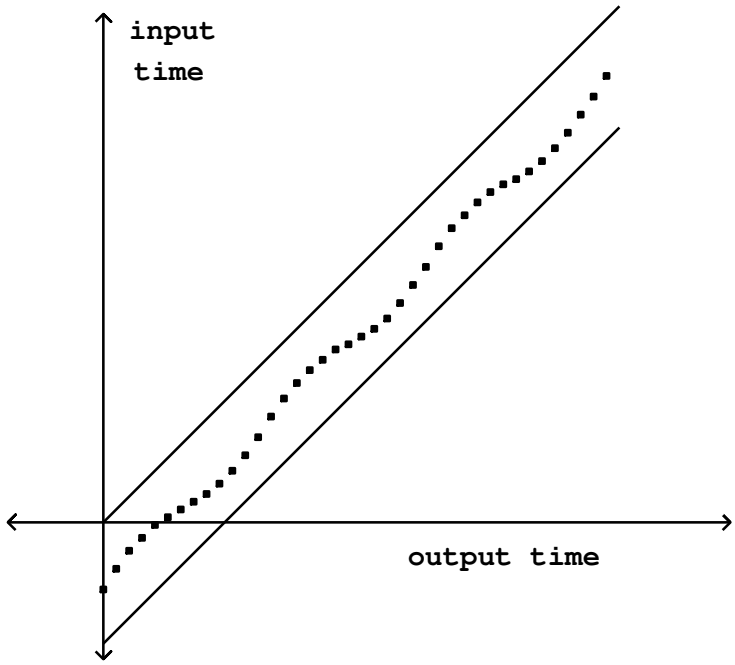


Рис. 7.19: Вибрато, использующее переменную линию задержки. Поскольку высота скользит вверх-вниз, можно поддерживать её без сдвига за полосу допустимых задержек.

мы будем всегда, в итоге, попадать за диагональную полосу допустимых времён задержки. В простейшем сценарии мы просто меняем транспонирование вверх и вниз, так чтобы оставаться внутри полосы.

Это работает, например, если мы хотим осуществить вибрато для звука, как показано на рис. 7.19. Здесь функция задержки есть

$$d[n] = d_0 + a\cos(\omega n),$$

где d_0 есть средняя задержка, a — амплитуда вариации вокруг средней задержки и ω — угловая частота. *Моментальная транспозиция* (стр. 7.1) равна приблизительно

$$t = 1 + a\omega\cos(\omega n - \pi/2),$$

то есть пробегает значения от $1 - a\omega$ до $1 + a\omega$.

С другой стороны, предположим, что мы хотим поддерживать постоянную транспозицию в течение долгого промежутка времени. В этом случае мы не можем поддерживать транспонирование всё время, но всё-таки возможно поддерживать его в течение фиксированных промежутков времени, разбитых разрывными изменениями, как показано на рис. 7.20. Время задержки является выходом подходящим образом нормализованной пилообразной функции, и выход переменной линии задержки осуществлён, как показано на рисунке, с тем чтобы избежать разрывов.

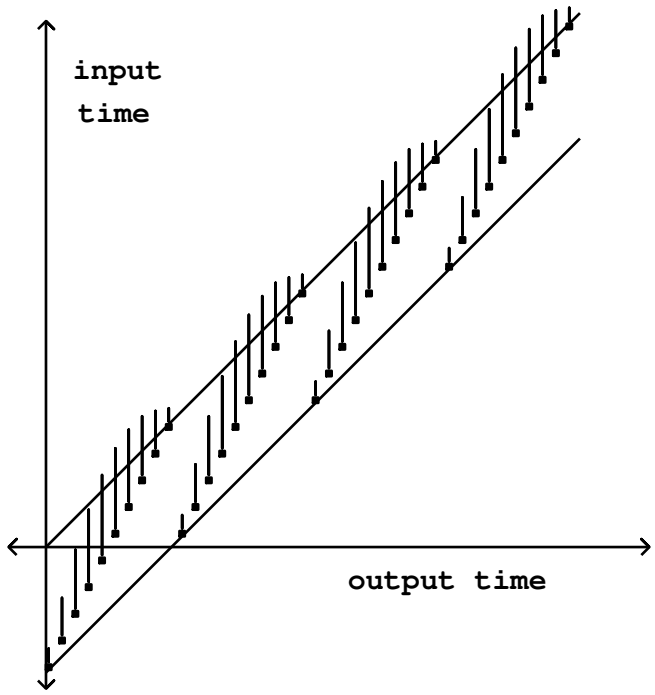


Рис. 7.20: Кусочно-линейные функции задержки для поддерживания постоянной транспозиции (за исключением точек разрыва). К выходному сигналу применена огибающая, согласно длине отрезка выше каждой точки, для сглаживания выхода в точках разрыва времён задержки.

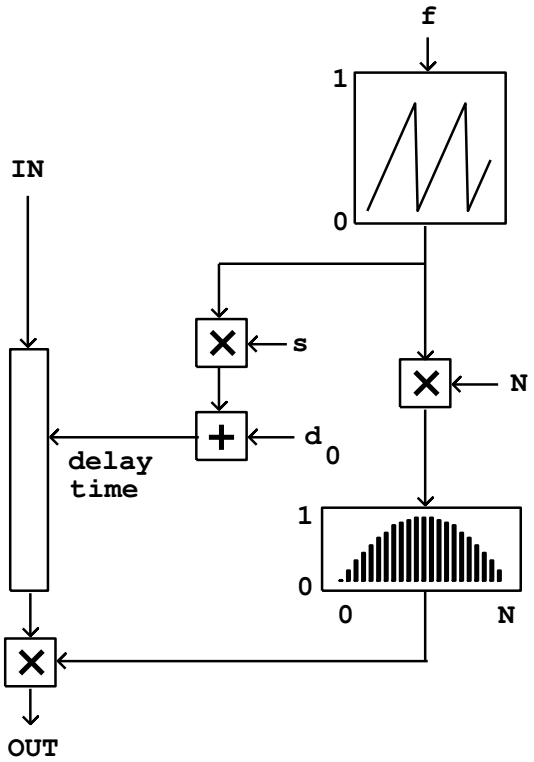


Рис. 7.21: Использование изменяемой линии задержки в качестве питч-шифтера. Пилообразный сигнал создает гладко возрастающее или спадающее время задержки. К выходу линии задержки применена амплитудная огибающая для устранения разрывов сигнала. Другая копия этой же схемы должна быть запущена с отставанием по фазе в 180 градусов (π радиан).

Это делается так, как показано на рис. 7.21. Выход генератора пилообразной волны используется двумя путями. Во-первых, он настроен так, чтобы пробегать между границами d_0 и $d_0 + s$, и это настраивает пилообразную волну так, чтобы управлять временем задержки в семплах. Первоначальная задержка d_0 должна быть, по крайней мере, достаточной, чтобы сделать переменную задержку возможной; для четырёхточечной интерполяции она должна быть, по крайней мере, в один семпл. Большие значения d_0 добавляют константу — дополнительную задержку на выходе; она обыкновенно предлагается в качестве управления при сдвиге высоты, поскольку является существенно свободной. Величину s иногда называют *размером окна*. Она, грубо говоря, соответствует длине семпла в циклическом семплере (§2.2).

Пилообразный сигнал часто используется в качестве выходной огибающей в точности тем же способом, как в семплере волновой таблицы с огибающей на рис. 2.7 (стр. 38). Огибающая равна нулю в точках, где пила сворачивается, и между этими точками растёт гладко до максимального значения 1 (для единичного коэффициента усиления).

Если частота пилообразной волны есть f (в циклах за секунду), тогда её значения пробегают промежуток от 0 до 1 каждые R/f семплов (здесь R есть частота семплирования). Разность между последовательными значениями есть, таким образом, f/R. Если мы обозначим $x[n]$ выход пилообразного осциллятора, тогда

$$x[n+1] - x[n] = \frac{f}{R}$$

(за исключением точек свертки). Если мы подстраиваем выходной диапазон осциллятора волновой таблицы на значение s (как сделано на рисунке), мы получаем новый наклон:

$$s \cdot x[n+1] - s \cdot x[n] = \frac{sf}{R}.$$

Добавление константы d_0 не меняет этот наклон. *Моментальная транспозиция* (стр. 195) тогда равна

$$t = 1 - \frac{sf}{R}.$$

Для завершения конструкции питч-шифтера мы должны добавить другую копию на половину в противофазе. Это приводит к задержке, считывающей массив как показано на рис. 7.22.

Питч-шифтер может транспонировать как вверх (с использованием отрицательных частот пилы, как на рисунке), так и вниз при использовании положительных частот. Сдвиг высоты обыкновенно управляется изменением f при фиксированном s. Для достижения требуемого интервала транспонирования t устанавливаем

$$f = \frac{(1-t)R}{s}.$$

Размер окна s должен быть выбран достаточно малым, настолько насколько это возможно, так чтобы каждая из копий с задержкой (разделённые на $s/2$ семплов) не

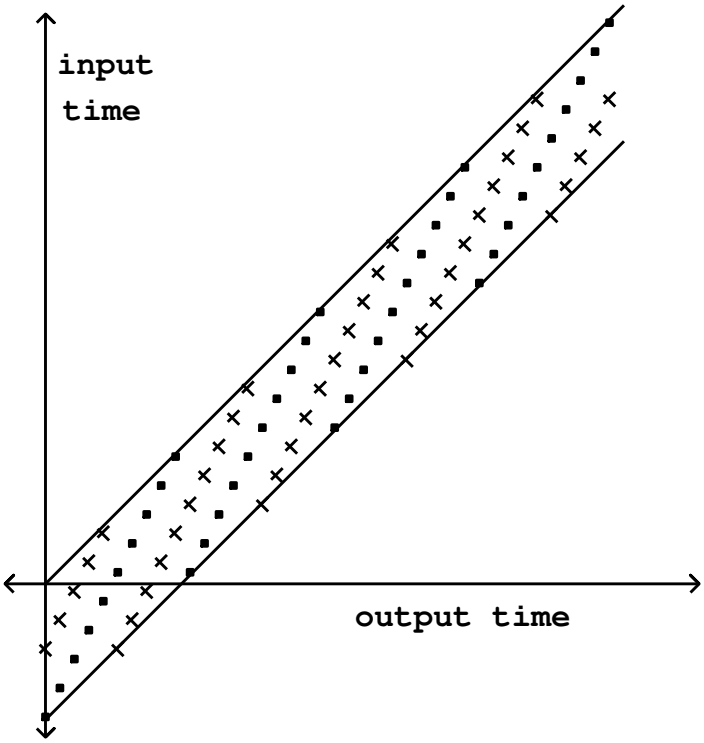


Рис. 7.22: Массив считывания с задержкой для питч-шифтера, использующего две линии задержки, так что одна имеет максимальную амплитуду, когда вторая включается.

звучала как отдельное эхо. Однако очень маленькие величины s будут поднимать f вверх; величины f более ≈ 5 герц, приводят к очень слышимой модуляции. Так что, если требуются очень большие транспозиции, величина s не может быть увеличена. Типичные значения имеют диапазон от 30 до 100 миллисекунд (около $R/30$ до $R/10$ семплов).

Хотя частота может быть изменена по желанию (даже разрывным образом), s необходимо менять более аккуратно. Возможным решением является приглушение выходного сигнала в момент разрывного изменения s; или же s можно непрерывно линейно изменять, но это вызовет сложноуправляемые допплеровские сдвиги.

Хорошим выбором огибающей является половина цикла синусоиды. Если мы предположим, что в среднем два выхода задержки некоррелированны (стр. 12), мощность сигнала от двух линий задержки после огибающей будет равняться константе (поскольку сумма квадратов двух огибающих равна единице).

Существует много вариаций этого алгоритма сдвига высоты. Один из классических вариантов — использовать единственную линию задержки без огибающей вообще. В этой ситуации необходимо выбрать точку, в которой время задержки совершает скачки и точку, куда скачек совершается, так чтобы выход оставался непрерывным. Например, можно найти точку, где выходной сигнал проходит через ноль ("пересечение нуля"), и совершить скачкообразный переход в другую точку. Использование только одной линии задержки имеет то достоинство, что выходной сигнал звучит как более "настоящий". Недостаток заключается в том, что поскольку время задержки является функцией входного значения сигнала, выход уже более не является линейной функцией входного сигнала, так что непериодичные входные сигналы могут приводить к таким артефактам, как разностные тоны.

7.10 Примеры

7.10.1 Фиксированная линия задержки без интерполяции

В примере G01.delay.pd (рис. 7.23) к входному сигналу применяется простая линия задержки. Необходимы два новых объекта:

delwrite~: определяет линию задержки и записи в нее. Первый начальный аргумент даёт имя линии задержки (две линии задержки не могут иметь одно имя). Второй начальный аргумент является максимальной длиной линии задержки в миллисекундах. На вход принимает аудиосигнал и непрерывно записывает его в линию задержки.

delread~: чтение из линии задержки. Первый начальный аргумент даёт имя линии задержки (которое должно быть согласовано с именем соответствующего объекта delwrite~; по этой причине Pd знает, какой объект delwrite~ ассоциировать с объектом delread~). Второй (необязательный) начальный

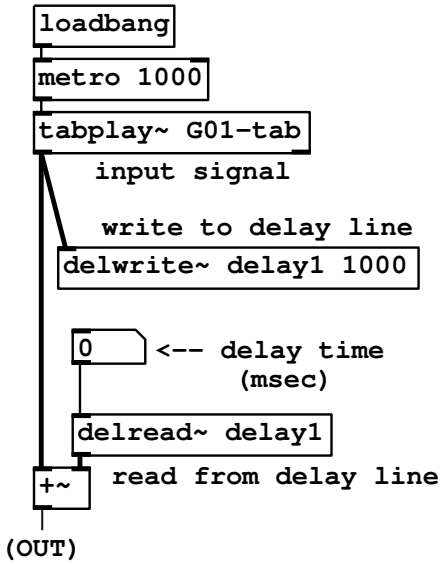


Рис. 7.23: Пример патча G01.delay.pd, показывающий задержку без интерполяции с временем задержки, управляемым в миллисекундах.

аргумент даёт время задержки в миллисекундах. Оно не может быть отрицательным и также не может превышать максимальную длину линии задержки, определённую объектом delwrite~. Входящие числа (сообщения) могут быть использованы для динамического изменения времени задержки. Однако это приведёт к разрывным изменениям на выходе, который должен быть, поэтому отключен, когда время задержки меняется.

Пример просто составляет пару из одного объекта delwrite~ и одного объекта delread~ для создания простой задержки без интерполяции. Входным сигналом является зацикленная запись. Задержанный и незадержанный сигналы складываются для создания гребенчатого фильтра без рециркуляции. С временами задержки ниже $\approx$ 10 миллисекунд, эффект фильтрации является наиболее заметным, а при более высоких значениях результат становится отдельно слышимыми эхо-сигналами. Защиты, приглушающей выход задержки, нет, так что когда время задержки меняется — возможны щелчки.

7.10.2 Гребенчатый фильтр с рециркуляцией

Пример G02.delay.loop.pd (рис. 7.24) показывает, как создать задерживающую сеть с рециркуляцией. Задержка опять выполняется с помощью пары delwrite~ / delread~. Выход объекта delread~ умножается на коэффициент усиления обратной связи 0.7 и направляется на объект delwrite~. Вход (обеспечиваемый phasor~ и связанными с ним объектами) добавляется к входу delwrite~; эта сумма становится выходным сигналом. Это представляет из себя гребенчатый фильтр с рециркуляцией §7.4.

Схема не имеет каких-либо циклов, в том смысле, что объекты соединены друг с

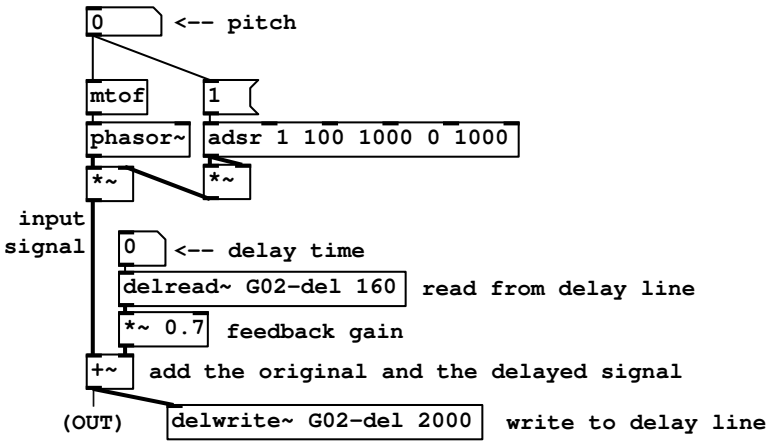


Рис. 7.24: Задержка с рециркуляцией (всё ещё без интерполяции).

другом напрямую или косвенно (через соединения с другими объектами). Обратная связь в схеме возникает неявно между объектами delwrite~ и delread~.

7.10.3 Переменная линия задержки

Следующий пример, G03.delay.variable.pd (рис. 7.25), представляет другой гребенчатый фильтр с рециркуляцией, на этот раз с использованием линии задержки с переменной длиной. Здесь введён один новый объект:

> vd~: изменяемое во времени чтение из линии задержки. Как и с объектом delread~, происходит считывание линии задержки, чьё имя определяется начальным аргументом. Вместо использования второго аргумента и/или управляющих сообщений для определения времени задержки, для объекта vd~, задержка в миллисекундах определяется входным аудиосигналом. Линия задержки определяется с использованием четырёхточечной (кубической) интерполяции; минимально возможная задержка равна одному семплу.

Здесь объекты с левой стороны, сверху вниз до объекта clip~ -0.2 0.2, образуют схему, задающую форму волны; индекс устанавливается управляющим параметром “timbre”, и выход схемы, задающей форму волны, варьируется между близлежащей синусоидой и ярким звонким звуковым шумом. Выход складывается с выходом объекта vd~. Сумма далее проходит через фильтр, пропускающий высокие частоты (объект hip~ слева внизу), умножается на коэффициент усиления обратной связи, подрезается и записывается на линию задержки справа внизу. Справа имеется управляющий параметр для установки коэффициента усиления обратной связи; здесь, в противоположность предыдущему примеру, можно определить коэффициент усиления, больший единицы, для того чтобы получить нестабильную обратную связь. По этой причине в задерживающем цикле вставлен второй объект clip~

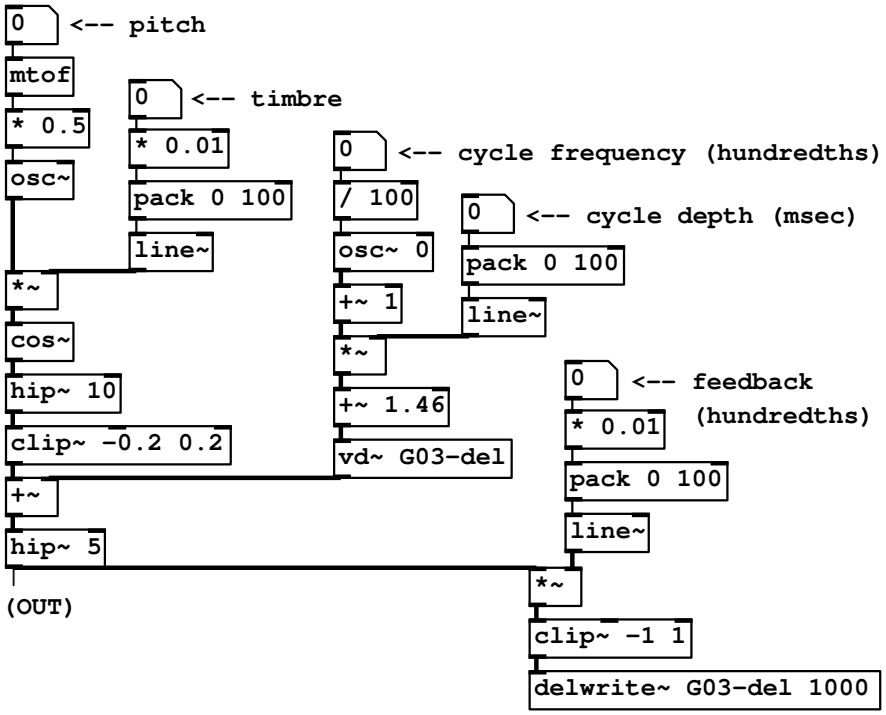


Рис. 7.25: Фленжер: переменная линия задержки с интерполяцией.

(непосредственно над объектом delwrite~), так что сигнал не может превзойти единицу по абсолютной величине.

Длина задержки управляется сигнальным входом объекта vd~. Осциллятор с переменной частотой и коэффициент усиления в центре рисунка задают время задержки. Сигнал осциллятора складывается с единицей, для того чтобы сделать его неотрицательным перед тем, как умножить его на управляющий параметр “cycle depth” (“глубина цикла”), что эффективно устанавливает диапазон для времени задержки. Минимально возможное время задержки 1.46 миллисекунды определяется тем, что реальный диапазон времён задержки лежит между минимальным временем и суммой минимального времени с удвоенной “глубиной”. Смысл минимального времени задержки выявляется при обсуждении следующего примера.

Гребенчатые фильтры с переменными временами задержки иногда называют *фленжерами*. Когда время задержки меняется, пики в частотном отклике сдвигаются вверх и вниз по частоте, так что тембр выхода постоянно меняется характерным образом.

7.10.4 Порядок выполнения и нижние пределы времён задержки

При использовании задержек (также как других тильда-объектов в Pd, разделяющих одно состояние) порядок, в котором проделываются операции записи и

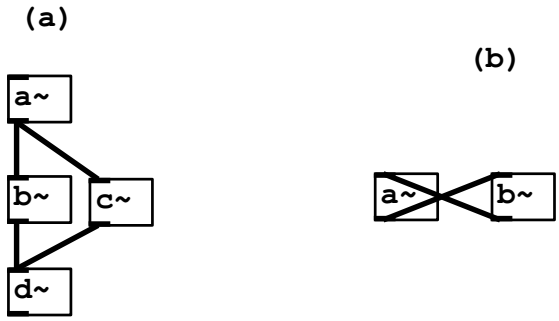


Рис. 7.26: Порядок выполнения тильда-объектов в Pd: (a) сеть без циклов. Объекты могут быть выполнены или в порядке “a-b-c-d”, или “a-c-b-d”. В части (b) имеется цикл и не существует, таким образом, совместимых линейно-упорядоченных объектов, поскольку каждый из них должен был бы быть запущенным раньше другого.

чтения, может влиять на выходное значение вычисления. Хотя тильда-объекты в патче могут иметь сложную топологию аудиосоединений, в действительности Pd все их выполняет в последовательном порядке, один за другим, для вычисления каждого блока аудиовыхода. Этот линейный порядок гарантирован для совместимости с внутренними аудиосоединениями в том смысле, что никакое вычисление тильда-объекта не проделывается, пока все его входы в том же блоке не были вычислены.

Рисунок 7.26 показывает два примера топологий тильда-объектов и их перевод в последовательность вычислений. В части (a) имеется четыре тильда-объекта и, в силу имеющихся соединений, объект [a~] должен произвести свой выходной сигнал перед тем, как [b~] или [c~] могут быть запущены; и оба они в свою очередь используются в вычислении [d~]. Таким образом, возможный порядок для этих четырёх объектов будет “a-b-c-d” и “a-c-b-d”. Эти два порядка будут иметь в точности один и тот же результат, если только вычисления [b~] и [c~] не влияют каким-либо образом одно на выход другого (что может произойти при использовании операций задержки, например).

Часть (b) рисунка показывает цикл тильда-объектов. Эта сеть не может быть сортирована в совместимом последовательном порядке, поскольку каждый из [a~] и [b~] требует, чтобы сначала был вычислен выход другого. В общем случае, последовательное упорядочение тильда-объектов возможно тогда и только тогда, когда в сети этих объектов и взаимосвязей их аудиосигналов нет циклов. Pd объявляет об ошибке, когда такой цикл появляется. (Заметим, что ситуация для *управляющих соединений* между объектами является более сложной и гибкой; смотрите документацию Pd для изучения деталей).

Для того чтобы видеть эффект порядка вычислений для пары [delwrite~]/[delread~], мы можем записать точно входной и выходной сигналы в двух возможных порядках с минимально возможной задержкой. Если сначала происходит запись с блока,

начинающегося на семпле с номером N, операция может быть записана так:

$$x[N], \ldots, x[N+B-1] \longrightarrow \boxed{\texttt{delwrite~}},$$

где B есть размер блока (как в §3.2). Поместив эти специальные семплы в линию задержки, мы даём возможность следующему delread~ считать эти значения:

$$\boxed{\texttt{delread~}} \longrightarrow x[N], \ldots, x[N+B-1].$$

С другой стороны, предположим, что объект delread~ поставлен перед delwrite~. Тогда семплы $x[N], \ldots, x[N+B-1]$ ещё не помещены в линии задержки, так что наиболее близкие семплы, которые могут быть считаны, принадлежат предыдущему блоку:

$$\boxed{\texttt{delread~}} \longrightarrow x[N-B], \ldots, x[N-1]$$
$$x[N], \ldots, x[N+B-1] \longrightarrow \boxed{\texttt{delwrite~}}.$$

Здесь минимальная задержка, которую мы можем получить, равна размеру блока B. Таким образом, минимальная задержка есть или 0, или B, в зависимости от порядка последовательности выполнения, в которой объекты delread~ и dewrite~ размещены.

Возвращаясь назад к патчам рис. 7.24 и 7.25, которые относятся к задержкам с рециркуляцией, видим, что объект delread~ или vd~ должен быть помещен в последовательности раньше, чем объект delwrite~. Это справедливо для любой схемы, в которой выход задержки перенаправляется на её вход. Минимальная возможная задержка содержит B семплов. Для (типичной) частоты семплирования в 44100 герц и размера блока в 64 семпла, это соответствует 1.45 миллисекундам. На первый взгляд это может показаться не очень существенным ограничением. Но если вы пытаетесь настроить гребенчатый фильтр с рециркуляцией на определённую высоту, наибольшее значение, которое можно получить, равно приблизительно 690 герц. Для получения более коротких задержек с рециркуляцией, вы должны увеличить частоту семплирования или уменьшить размер блока.

Пример G04.control.blocksize.pd (рис. 7.27) показывает, как размер блока может управляться в Pd с использованием нового объекта:

> block~, switch~: Установить локальный размер блока окна патча, в котором находится объект. Размеры блока, обыкновенно, являются степенями двойки. Объект switch~, в дополнение, может быть использован для включения и выключения аудиовычислений внутри окна при помощи использования управляющих сообщений. Дополнительные начальные аргументы могут устанавливать локальную частоту семплирования и определять перекрывающиеся вычисления (что демонстрируется в Главе 9).

В части (a) рисунка (основной патч) прямоугольный импульс посылается на сабпатч Pd delay-writer, выход которого затем возвращается в основной патч. Часть

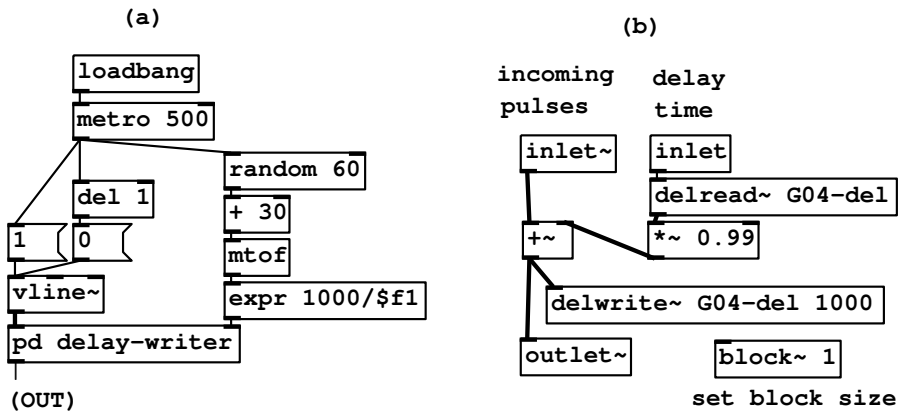


Рис. 7.27: Патч, использующий управление размером блока для уменьшения циклической задержки ниже обычных 64 семплов: (a) основной патч; (b) сабпатч "delay-writer" с объектом block~ и рециркулирующей сетью с задержкой.

(b) показывает содержание сабпатча, который посылает импульсы на задержку с рециркуляцией. Объект block~ определяет, что в этом сабпатче при вычислении сигнала используется размер блока (B) равный всего лишь единице. Так что минимально возможная задержка равна одному семплу, вместо принятых по умолчанию 64.

Помещение импульса (или другого сигнала возбуждения) на гребенчатый фильтр с рециркуляцией для создания высоты тона иногда называют синтезом Карплуса-Стронга, описанным в их статье [KS83], хотя идея представляется более старой. Она представлена, например, в пьесе Пола Лански 1979 года (Paul Lansky, Six Fantasies on a Poem by Thomas Campion).

7.10.5 Порядок выполнения в линиях задержки без рециркуляции

В задерживающих сетях без рециркуляции должна быть возможность записи в линию задержки раньше операции ее чтения. Таким образом, не должно быть предела снизу для длины линии задержки (за исключением наложенного схемой интерполяции; смотри §7.7). В таких языках, как Csound, последовательность единичных генераторов вычисления (почти) явная, так что её легко указать. Однако в графических средах исправлений этот порядок является неявным, и необходимо использовать другой подход, чтобы гарантировать, что, например, объект delwrite~ вычисляется перед соответствующим объектом delread~. Один из способов сделать это показан в примере G05.execution.order.pd (рис. 7.28).

В части (a) рисунка соединения в патче не определяют, в каком порядке две операции задержки будут в отсортированной последовательности вычисления тильда-объектов; объект delwrite~ мог бы вычисляться или перед, или после объекта vd~.

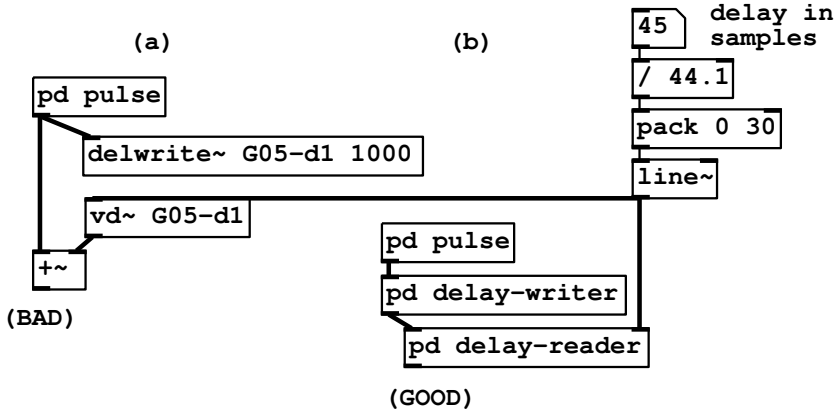


Рис. 7.28: Использование сабпатчей для гарантирования того, что линии задержки записаны перед тем, как они считываются в сетях без рециркуляции: (a) объекты delwrite~ и vd~ могут выполняться в "правильном" или "неправильном" порядке; (b) объект delwrite~ внутри сабпатча pd delay-writer и объект vd~ внутри сабпатча pd delay-reader. По причине аудиосвязей между двумя сабпатчами, порядок выполнения пары read/write насильно задан корректно.

Если мы хотим убедиться, что операция записи происходит до операции чтения, мы можем поступить так, как показано в части (b) рисунка, и поместить две операции в сабпатчи, соединив их с помощью аудиосигналов так, чтобы первый сабпатч должен был быть вычислен до второго. (Вычисление звука в сабпатчах выполняется атомарно, в том смысле, что все содержимое сабпатча рассматривается как аудио вычисление для сабпатча в целом. Таким образом, все, что происходит в одном сабпатче, происходит раньше, чем что-либо во втором.)

В этом примере "правильный" и "неправильный" способы создания гребенчатого фильтра приводят к различимым на слух результатам. Для задержек менее 64 семплов, правая часть патча (использующая сабпатчи) даёт корректный результат, но левая часть не может произвести задержки меньше размера блока 64.

7.10.6 Гребенчатый фильтр без рециркуляции в качестве октавного удвоителя

В примере G06.octave.doubler.pd (рис. 7.29) мы обращаемся опять к идее сдвига октавы, основанного на высоте звука, введённой ранее в E03.octave.divider.pd. Там знание периодичности входного сигнала позволило нам настроить кольцевую модуляцию для введения субгармоник. Здесь мы создадим октавный удвоитель, описанный в §7.3. Используя переменный гребенчатый фильтр без рециркуляции, мы убираем нечётные гармоники, оставляя только чётные, что даёт звучание на октаву выше. Как и прежде, спектральная огибающая звука в грубом виде сохраняется такой операцией, так что мы можем избежать эффекта *мультяшного голоса*, который мы бы получили, используя изменение скорости для транспозиции.

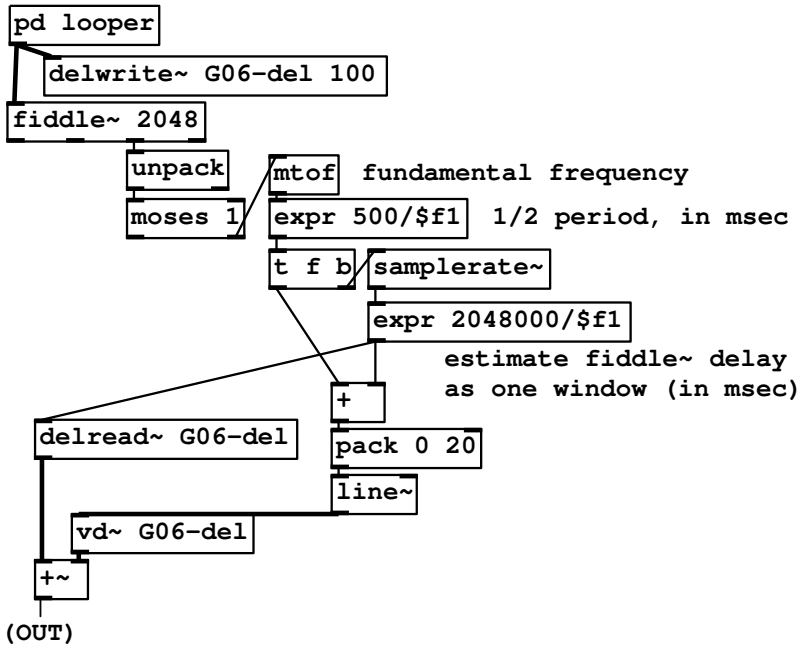


Рис. 7.29: "Удвоитель октавы" использует информацию о высоте (полученную с использованием объекта fiddle~) для настройки гребенчатого фильтра на удаление всех нечётных гармоник из входного звука.

Фильтрация гребенчатым фильтром производится с помощью объединения двух задержанных копий входного сигнала (из сабпатча pd looper сверху). Одна, с фиксированной задержкой (delread~), устанавливается на размер окна алгоритма считывания высоты. Поскольку в примере ранее это было скрыто в другом сабпатче, мы можем показать теперь его в явном виде. Задержка в миллисекундах оценивается равной 2048-семпловому окну анализа, используемому объектом fiddle~; в миллисекундах это приводит к $1000 \cdot 2048/R$, где R есть частота семплирования.

Переменная задержка равна тому же значению плюс 1/2 измеренного периода входного сигнала или $1000/(2f)$ миллисекунд, где f есть частота в циклах в секунду. Сумма этого и фиксированного времени задержки затем сглаживается с использованием объекта line~ для создания входного сигнала для переменной линии задержки.

Поскольку разность между двумя задержками есть $1/(2f)$, резонансные частоты получившегося гребенчатого фильтра есть $2f, 4f, 6f, ldots$; частотный отклик (§7.3) равен нулю на частотах $f, 3f, \ldots$, так что получающийся звук содержит только гармоники, кратные $2f$ — на октаву выше первоначальной. С другой стороны, входящий звук выводится дважды, с интервалом в половину цикла; нечётные гармоники, тем самым, сдвигаются на 180 градусов (π радиан) и уничтожаются; чётные гармоники будут в фазе с их задержанными копиями и останутся в сумме.

Вместе, этот алгоритм и октавный делитель, могут быть изменены, чтобы сделать сдвиг в три или четыре раза по частоте, и они также могут быть объединены для создания составных сдвигов, таких как музыкальная квинта (отношение 3 : 2) с

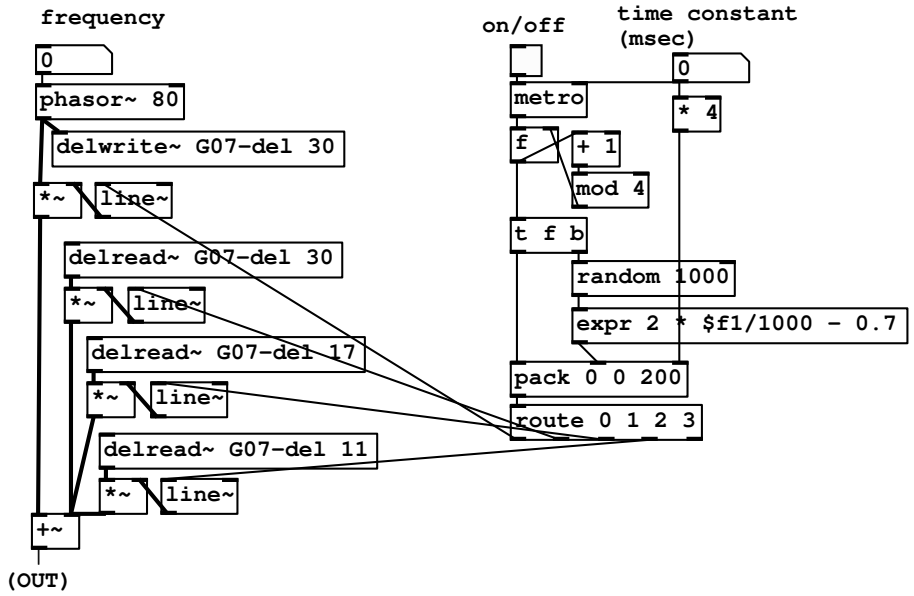


Рис. 7.30: «Шейкер» — четырехотводный гребенчатый фильтр со случайно изменяемыми амплитудами выходов линий задержки.

помощью сдвига вниз на октаву и возвращения наверх с коэффициентом 3. (Для достижения наилучших результатов нужно делать сдвиг вниз перед сдвигом вверх.)

7.10.7 Изменяющийся во времени сложный гребенчатый фильтр: генератор регулируемых вибраций (шейкер)

Пример G07.shaker.pd (рис. 7.30) показывает другой способ расширения идеи гребенчатого фильтра. Здесь мы комбинируем входной сигнал с четырьмя различными временными сдвигами (вместо двух, как в первоначальном гребенчатом фильтре без рециркуляции), каждый с различным положительным или отрицательным коэффициентом усиления. Для того чтобы сделать это, мы помещаем входной сигнал в линию задержки и считываем его в трёх различных точках; четвёртое "считывание" представляет из себя первоначальный сигнал без задержки.

Чтобы представить себе частотный отклик четырёхсигнального гребенчатого фильтра, мы сначала рассмотрим, что происходит, когда два из четырёх коэффициентов усиления близки к нулю. В итоге мы получаем простой нерециркуляционный гребенчатый фильтр с небольшим усложнением, заключающимся в том, что усиление двух отсроченных копий может быть разным. Если оба они имеют один знак, мы получаем такие же пики и спады, какие были предсказаны в §7.3, только спады между пиками возможно более мелкие. Если они различных знаков, спады становятся пиками, а пики — спадами.

В зависимости от того, какие две копии мы предположили ненулевыми, пики и спады размещаются по различным значениям; времена задержек выбираются

так, что 6 различных времён задержки могут возникнуть в диапазоне от 6 до 30 миллисекунд. В общем случае, когда все коэффициенты усиления не равны нулю, мы можем представить, что частотный отклик непрерывно изменяется между этими крайностями, образуя последовательность сложных паттернов. Это приводит к росту и падению амплитуд гармоник входящего сигнала в сложном паттерне, одних независимо от других, что даёт постоянно меняющийся во времени тембр.

Правая часть патча предназначена для изменения коэффициентов усиления входного сигнала и его трёх копий, сдвинутых по времени. Всякий раз, когда активируется объект metro, счётчик получает приращение (объекты f и mod 4), которое управляет изменением амплитуд. Сама амплитуда вычисляется созданием случайной величины и нормализацией её таким образом, чтобы её величина лежала между -0.7 и 1.3. Случайная величина и индекс пакуются с третьей величиной (интервалом времени), и этот триплет отправляется на объект route. Первый элемент триплета (счётчик) выбирает выход, на который нужно отправить остальные два значения; в результате один из четырёх возможных объектов line~ получает сообщение о линейном изменении к новому значению.

Если изменение времени выполняется достаточно быстро, то на исходный сигнал также накладывается эффект модуляции; в этой ситуации прямолинейные отрезки, использованные в этом примере, должны быть заменены модулирующими сигналами с более управляемым частотным содержанием, например, с использованием фильтров (предмет Главы 8).

7.10.8 Ревербератор

Пример G08.reverb.pd (рис. 7.31) показывает простой искусственный ревербератор, в существенном являющийся реализацией конструкции, показанной на рис. 7.15. На четыре линии задержки направляется вход и их собственный рециркулирующий выход. Выходы задержек смешиваются с использованием матриц поворота, построенных из элементарных поворотов на $\pi/4$, как на рис. 7.13 (часть а).

Нормализующее умножение (на $1/\sqrt{2}$ на каждом этапе) поглощается коэффициентом усиления обратной связи, который поэтому не может превышать 1/2. При коэффициенте усиления, равном в точности 1/2, вся энергия, выходящая из линий задержки, возвращается обратно в них, так что реверберация продолжается бесконечно.

Рисунок 7.32 показывает внутренность абстракции reverb-echo, использованной в ревербераторе. Два входа смешиваются (с использованием одной и той же матрицы поворота, откладывая нормализацию на потом). После этого один канал задерживается. Времена задержки выбираются, грубо говоря, растущими экспоненциально; это дает гладкий и распределенный эхо-паттерн.

Существует множество вариантов развития этой идеи, из которых мы назовем лишь несколько. Естественно, во-первых, поставить фильтры низких частот в конце

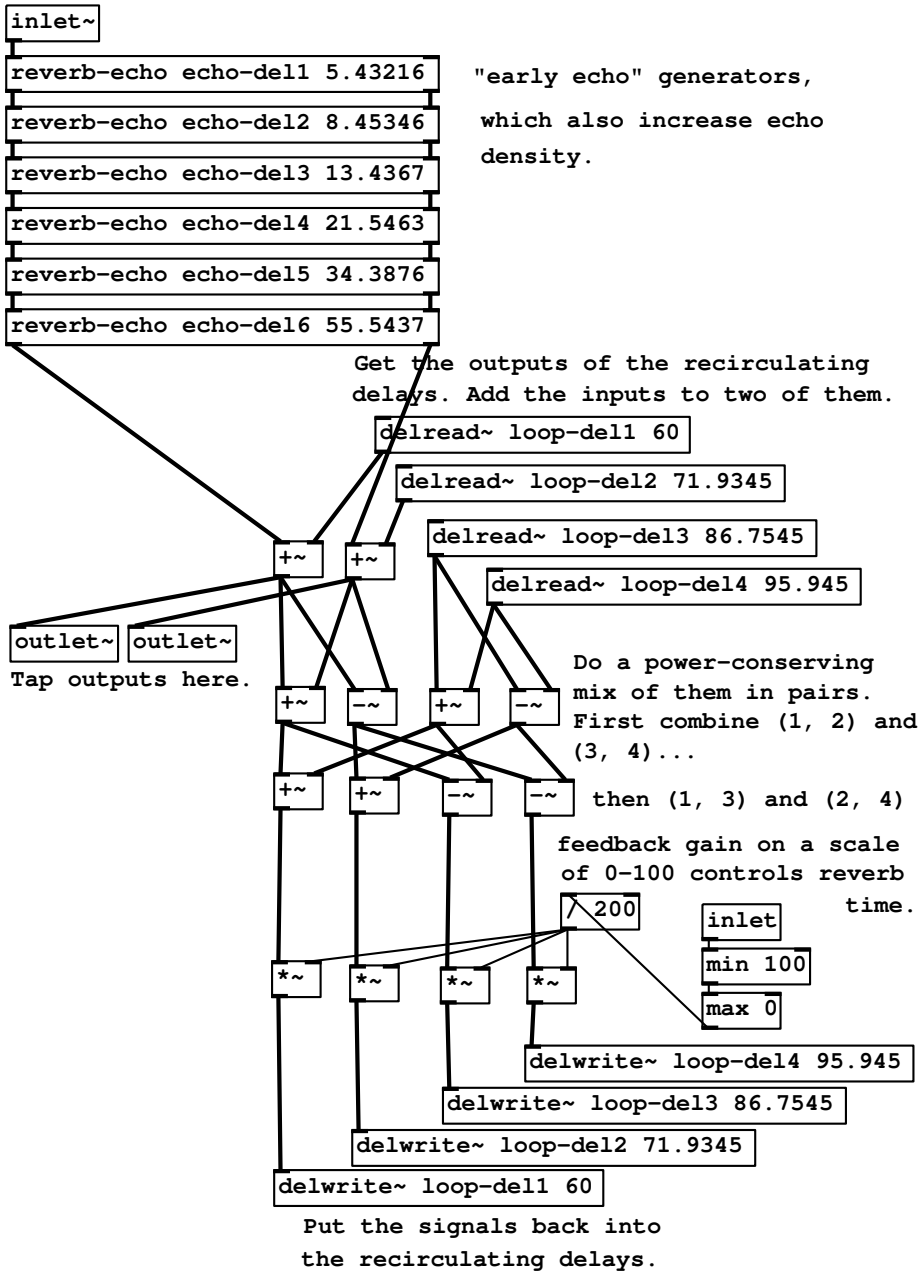


Рис. 7.31: Искусственный ревербератор.

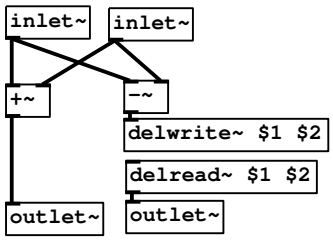


Рис. 7.32: Эхо-генератор, используемый в ревербераторе.

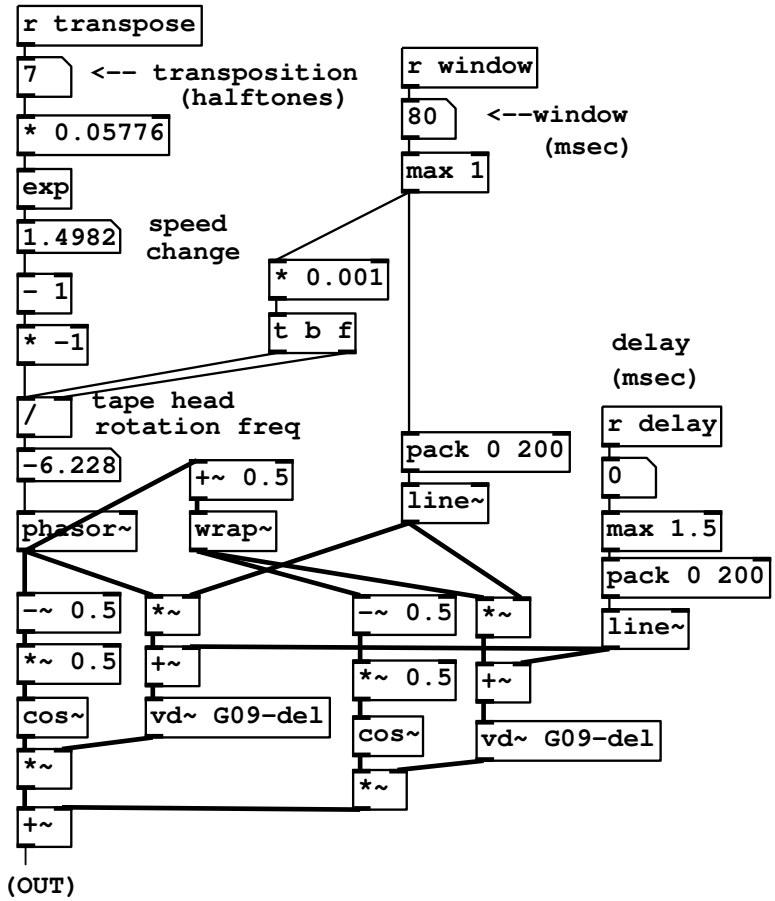


Рис. 7.33: Питч-шифтер, использующий два переменных "выхода"с линии задержки.

линий задержки, чтобы имитировать обычно более быстрое затухание высоких частот, чем низких. Также часто используется более четырех рециркуляционных задержек: например, один из ревербераторов, распространяемых в Pd, использует шестнадцать. Наконец, обычно допускается раздельное управление амплитудами ранних эхо-сигналов (слышимых непосредственно) и амплитудами рециркулирующего сигнала. Считается, что такие параметры контролируют звуковые качества, описываемые как “эффект присутствия”, “теплота”, “ясность” и так далее.

7.10.9 Питч-шифтер

Пример G09.pitchshift.pd (рис. 7.33) показывает реализацию питч-шифтера, описанного в §7.9. Линия задержки (определённая где-то в другом месте патча) задана с использованием двух объектов vd~. Времена задержки варьируются от минимальной задержки (представленной элементом управления “delay”) до минимальной задержки плюс размер окна (элемент управления “window”).

Требуемый сдвиг высоты тона в полутонах (h) сначала преобразуется в коэф-

фициент транспозиции

$$t = 2^{h/12} = e^{\log(2)/12 \cdot h} \approx e^{0.05776h}$$

(называемый “speed change”[2] в патче). Вычисление, названное “tape head rotation speed”[3], то же самое, как в формуле для f на стр. 201. Здесь положительный интервал (семь полутонов) приводит к коэффициенту транспозиции выше, чем единица и, поэтому, к отрицательному значению f.

Когда f вычислено, результат двух пилообразных сигналов в фазе и соответствующих огибающих в точности такой же, как у циклического семплера с перекрытием (пример B10.sampler.overlap.pd на стр. 56).

К каждому из двух пилообразных сигналов добавляется минимальное значение задержки для создания сигналов управления задержкой объектов vd~, выходы которых умножаются на соответствующие огибающие и суммируются.

Упражнения

1. Комплексное число имеет единичный модуль и аргумент $\pi/4$. Каковы его действительная и мнимая части?

2. Комплексное число имеет единичный модуль и действительную часть 1/2. Какова его мнимая часть? (Имеется два возможных значения.)

3. Какое время задержки вы бы дали гребенчатому фильтру, так чтобы его первый пик в частотном отклике был на 440 герцах? Если частота семплирования равна 44100, какая частота будет соответствовать ближайшей целой задержке?

4. Предположим, вы внесли изменения в гребенчатый фильтр без рециркуляции таким образом, что сигнал с задержкой вычитается из исходного сигнала, а не добавляется. Какой будет новая частотная характеристика?

5. Если вы хотите создать вибрато 6 Гц при помощи синусоидально меняющейся линией задержки, и если вы хотите, чтобы вибрато изменяло частоту на 5%, насколько большое изменение задержки вам необходимо? Что стало бы с этим изменением, если бы та же глубина вибрато требовалась на 12 герцах?

6. Комплексная синусоида $X[n]$ имеет частоту 11025 герц, амплитуду 50 и первоначальную фазу 135 градусов. Другая синусоида $Y[n]$ имеет ту же частоту, но амплитуду 120 и первоначальную фазу 45 градусов. Какова амплитуда и первоначальная фаза суммы X и Y?

[2] (*прим. перев.*) изменение скорости
[3] (*прим. перев.*) скорость поворота головки ленты

Глава 8

Фильтры

В предыдущей главе мы видели, что сеть с задержкой может иметь неравномерный частотный отклик — коэффициент усиления, который изменяется как функция частоты. Сети с задержкой также обыкновенно меняют фазу входящих сигналов переменным образом в зависимости от частоты. Когда используемое время задержек очень короткое, наиболее важные свойства сети с задержкой — её частотный и фазовый отклики. Сеть с задержкой, сконструированная специально для определённого частотного или фазового отклика, называется *фильтром*.

На блок-схемах фильтры показываются как на рис. 8.1 (часть a). Кривая, показанная внутри блока, даёт качественное представление о частотном отклике фильтра. Частотный отклик может зависеть от времени и, в зависимости от конструкции фильтра, для его изменения могут быть использованы один или более управляющих параметров (или дополнительных аудиовходов).

Предположим, следуя процедуре §7.3, мы передаём комплекснозначную синусоиду с единичной амплитудой и угловой частотой ω в фильтр. Мы ожидаем получить на выходе синусоиду той же частоты и некоторой амплитуды, которая зависит от ω. Это даёт нам комплекснозначную функцию $H(\omega)$, которая называется *передаточной функцией фильтра*.

Частотным откликом называется коэффициент усиления как функция частоты ω. Он равен модулю передаточной функции. Частотный отклик фильтра принято строить как на рис. 8.1 (часть b). Входная синусоида единичной амплитуды и частоты ω выходит из фильтра с амплитудой, равной модулю $|H(\omega)|$.

Иногда также полезно знать фазовый отклик фильтра, равный $\angle(H(\omega))$. Для фиксированной частоты ω выходная фаза фильтра будет на $\angle(H(\omega))$ радиан больше его входной фазы.

Конструкция фильтров и их использование представляет из себя огромную тему, так как широкий диапазон возможного использования фильтров предполагает множество методов их создания. В некоторых приложениях фильтр должен точно следовать предписанному частотному отклику, в других важно минимизировать время вычисления, в-третьих важен фазовый отклик, а ещё в других фильтр

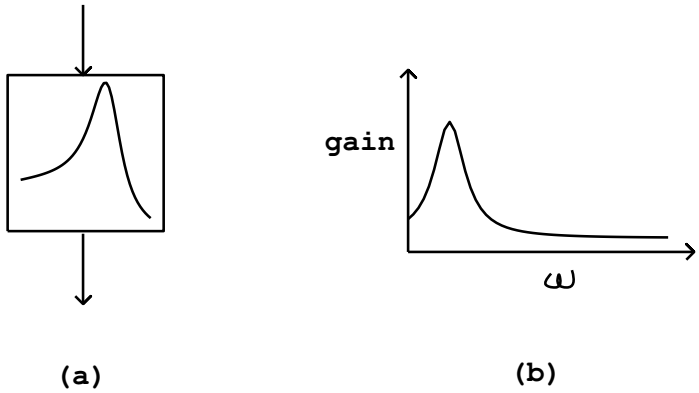


Рис. 8.1: Представление фильтра: (a) на блок-схеме; (b) график его частотного отклика.

должен вести себя предсказуемо, когда его параметры меняются быстро по времени.

8.1 Классификация фильтров

На протяжении истории электронной музыки технология построения фильтров постоянно менялась, но некоторые виды фильтров часто открывались заново. В этом параграфе мы дадим некоторую терминологию для описания фильтров нескольких основных повторяющихся типов. Позднее мы разовьём некоторые основные стратегии создания фильтров с требуемыми характеристиками и, наконец, мы обсудим некоторые общие приложения фильтров в электронной музыке.

8.1.1 Фильтры, пропускающие низкие и высокие частоты

С давних времён наиболее часто встречающейся задачей использования фильтра является извлечение, или низкочастотной, или высокочастотной части аудиосигнала с удалением оставшихся частей. Это осуществляется с использованием фильтров, *пропускающих низкие* или *высокие* частоты, которые мы будем называть *фильтром низких частот* (ФНЧ) и *фильтром высоких частот* (ФВЧ) соответственно.

В идеале фильтры высоких или низких частот должны были бы иметь частотный отклик, равный единице вплоть до (или после) некоторой срезающей частоты и нулю после неё; но такие фильтры не могут быть реализованы на практике. Вместо этого мы попытаемся найти реализуемые приближения этого идеального отклика. Чем больше конструктивных усилий и вычислительного времени мы затратим, тем ближе к идеалу будет то, что мы можем получить.

Рисунок 8.2 показывает частотный отклик фильтра низких частот. Частота делится на три полосы, отмеченных на горизонтальной оси. *Полоса пропускания* (passband) есть область, где фильтр должен пропускать свой входной сигнал с

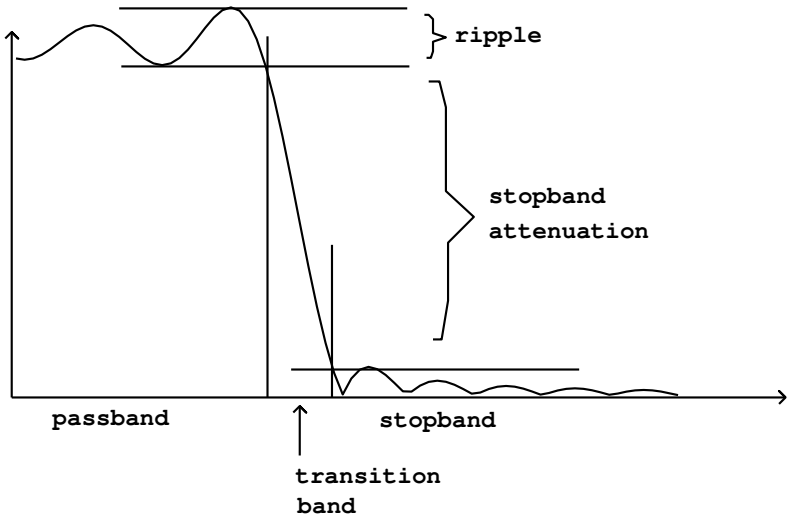


Рис. 8.2: Терминология для описания частотного отклика ФНЧ и ФВЧ. Горизонтальная ось отвечает частоте и вертикальная ось — коэффициенту усиления. Показан фильтр низких частот; фильтр высоких частот имеет те же черты с взаимным отражением правой и левой частей.

единичным коэффициентом усиления. Для фильтра низких частот (как показано) пропускаемая полоса простирается от нулевой частоты до некоторого частотного предела. Для ФВЧ пропускаемая полоса появилась бы в правой части графика и простиралась бы от частотного предела до максимально возможной частоты. Любая реализуемая пропускаемая полоса фильтра будет только приблизительно плоской; отклонение от плоскости называется *пульсациями в полосе пропускания* (ripple) и часто определяется отношением между наиболее высоким и наиболее низким коэффициентами усиления в пропускаемой полосе, выражаемого в децибелах. Идеальный ФНЧ или ФВЧ частот имел бы пульсации в полосе пропускания в 0 дб.

Полоса задерживания[1] ФНЧ или ФВЧ есть полоса частот, которую фильтр не должен пропускать. *Падение на полосе задерживания* (stopband attenuation) есть разность в децибелах между наиболее низким коэффициентом усиления на пропускаемой полосе и наиболее высоким коэффициентом усиления на задерживаемой полосе. В идеале он должен быть равен бесконечности; наибольшее его значение является наилучшим.

Наконец, реализуемый фильтр, чей частотный отклик является всегда непрерывной функцией частоты, должен иметь полосу частот, на которой коэффициент усиления падает от коэффициента усиления пропускаемой полосы до коэффициента усиления задерживаемой полосы; она называется *переходной полосой*[2]. Чем уже может быть сделана эта полоса, тем ближе будет фильтр к идельному.

[1] stopband
[2] transition band

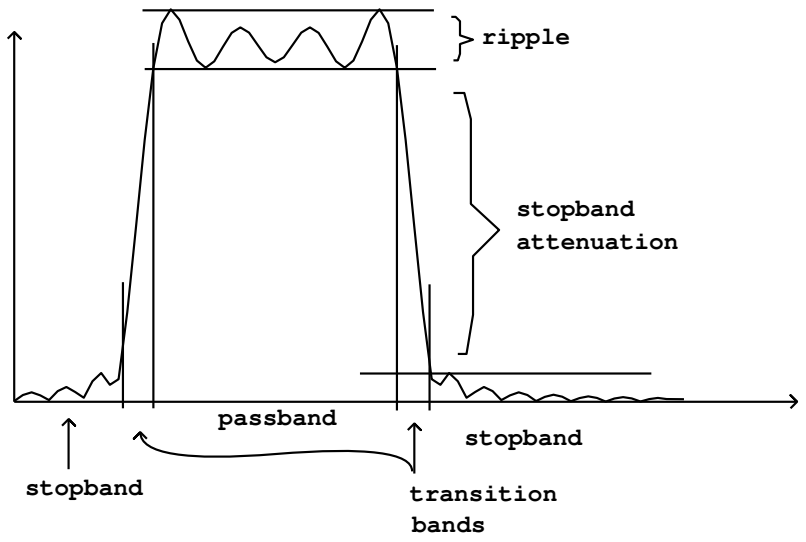


Рис. 8.3: Терминология для описания частотного отклика полосовых и режекторных фильтров. Горизонтальная ось соответствует частоте, и вертикальная — коэффициенту усиления. Показан полосовой фильтр; режекторный фильтр имел бы задерживаемую полосу, окружённую двумя пропускаемыми полосами.

8.1.2 Полосовые и режекторные фильтры

Полосовой фильтр пропускает частоты, внутри данной полосы и не пропускает частоты ниже или выше её. Рисунок 8.3 показывает частотный отклик полосового фильтра с отмеченными основными параметрами. Режекторный фильтр (полосно-заграждающий) делает всё наоборот: не пропускает частоты внутри полосы, пропуская частоты вне неё.

На практике часто используется более простой язык для описания полосовых фильтров, как показано на рис. 8.4. Здесь имеется только два параметра: *центральная частота* (center frequency) и *ширина полосы* (bandwidth). Пропускаемая полоса рассматривается как область, где фильтр имеет как минимум половину мощности усиления на пике (то есть усиление не меньше 3 децибелов от максимального). *Ширина полосы* есть ширина пропускаемой полосы в единицах частоты. Центральная частота есть точка максимального усиления, которая приблизительно равна средней точке полосы.

8.1.3 Эквалайзеры

В некоторых случаях, таких как *эквализация*, цель состоит не в пропускании сигналов определённых частот и задерживанием остальных, а в управляемой настройке, увеличивающей или уменьшающей сигнал в частотном диапазоне с требуемым коэффициентом усиления. Для этого полезны два вида фильтров.

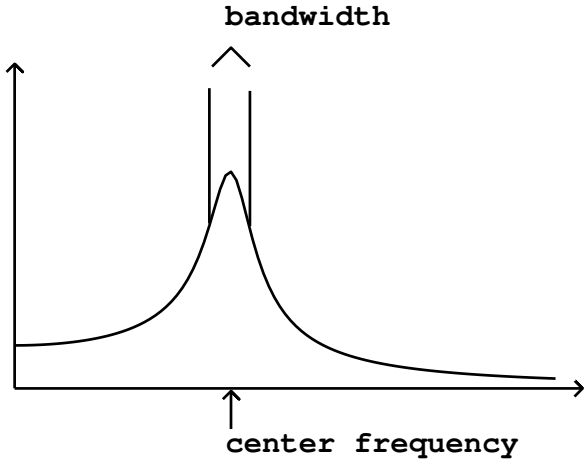


Рис. 8.4: Упрощённый взгляд на полосовой фильтр, показывающий ширину полосы и центральную частоту.

Первый — *полочный фильтр*[3] (рис. 8.5) — используется для выборочного усиления или уменьшения нижнего или верхнего конца частотного диапазона. Ниже выбираемой перекрёстной частоты [4] этот фильтр стремится к коэффициенту усиления на низкой частоте [5] и выше её стремится к другому коэффициенту усиления на высокой частоте [6]. Перекрёстная частота, коэффициенты усиления на высокой и низкой частоте могут настраиваться независимо.

Второй — *пиковый фильтр* (рис. 8.6) — объединяет усиление и уменьшение сигналов внутри нужного диапазона частот. Центральная частота[7] и ширина полосы[8] (которые вместе управляют задействованным диапазоном частот), а также внутриполосный коэффициент усиления[9] и внеполосный коэффициент[10] усиления могут быть настроены отдельно.

Параметрические эквалайзеры часто состоят из двух полочных фильтров (каждый для настройки нижнего и верхнего концов спектра), а также двух или трёх пиковых фильтров для настройки полос между ними.

[3]также используется термин шельфовый. (*прим. перев.*)
[4]crossover frequency
[5]low frequency gain
[6]high frequency gain
[7]center frequency
[8]bandwidth
[9]in-band gain
[10]out-band gain

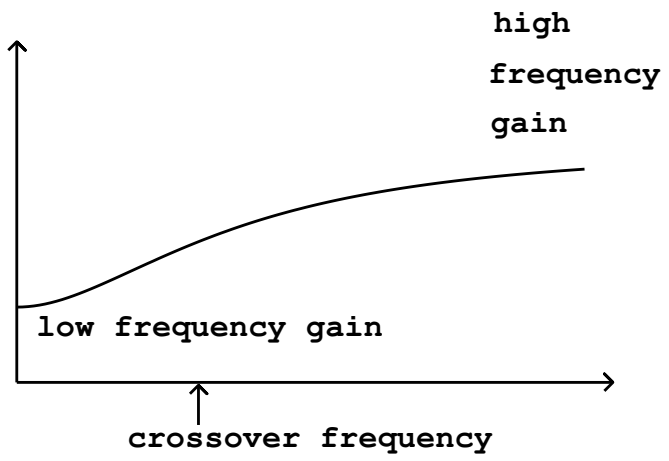


Рис. 8.5: Полочный фильтр: показаны коэффициенты усиления для нижних и верхних частот и перекрёстная частота.

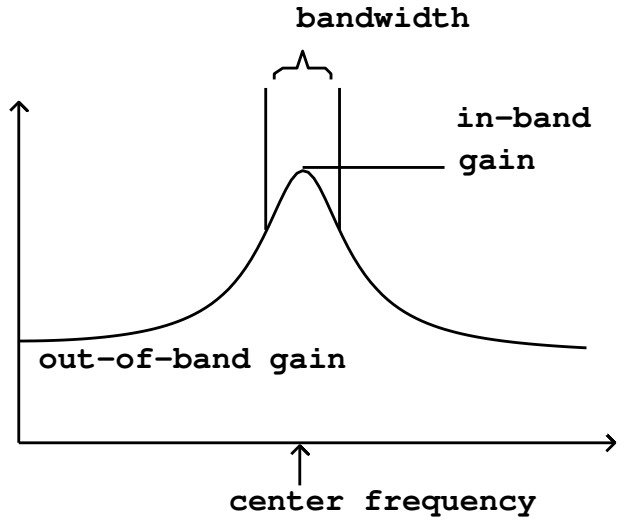


Рис. 8.6: Пиковый фильтр с управляемой центральной частотой, шириной полосы, внутриполосным и внеполосным коэффициентами усиления.

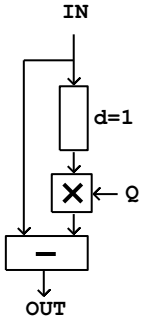


Рис. 8.7: Односемпловая сеть задержки с комплексным усилением Q, то есть элементарный фильтр без рециркуляции первой формы. Сравните с гребенчатым фильтром без рециркуляции, показанным на рис. 7.3, который здесь соответствует выбору $Q = -1$.

8.2 Элементарные фильтры

В Главе 7 мы узнали, как предсказывать частотный и фазовый отклик сетей с задержкой. Искусство конструкции фильтра состоит в создании сети с задержкой, чья передаточная функция (которая управляет частотным и фазовым откликом) имеет требуемую форму. Мы разработаем подход к построению таких сетей с задержкой из двух типов гребенчатых фильтров, построенных в Главе 7: с рециркуляцией и без рециркуляции. Здесь нам будет интересен специальный случай, в котором длина задержки равна лишь одному семплу. В этой ситуации частотные отклики, показанные на рис. 7.6 и 7.10, больше не выглядят как гребни; второй пик отступает практически к частоте семплирования 2π радиан, когда $d = 1$. Поскольку слышимыми являются частоты между 0 и частотой Найквиста (π радиан), в результате при $d = 1$ есть только один пик.

В гребенчатых фильтрах, показанных в Главе 7, пики размещены на DC (нулевая частота), но нам будет часто требоваться сместить их на другие, ненулевые частоты. Это делается с помощью использования сетей с задержкой — гребенчатых фильтров — с комплекснозначными коэффициентами усиления.

8.2.1 Элементарный фильтр без рециркуляции

Гребенчатый фильтр без рециркуляции может быть обобщён до конструкции, показанной на рис. 8.7. Это первая форма *элементарного фильтра без рециркуляции.* Его единственный комплекснозначный параметр Q управляет комплексным коэффициентом усиления задержанного сигнала, вычитаемого из первоначального.

Для того чтобы найти его частотный отклик, мы, как в Главе 7, направим на сеть с задержкой комплексную синусоиду $1, Z, Z^2, \ldots$ с частотой $\omega = \arg(Z)$. Семпл с номером n входа есть Z^n, и выход равен

$$(1 - QZ^{-1})Z^n,$$

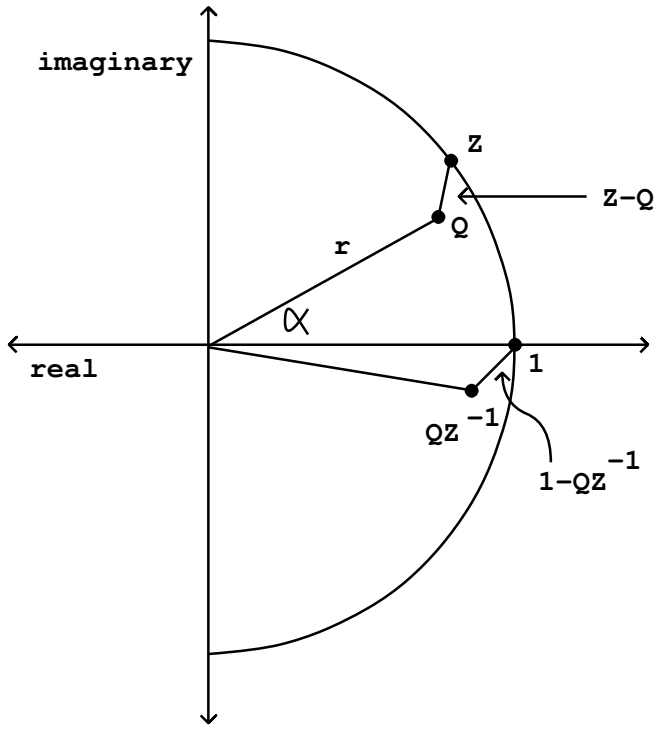


Рис. 8.8: Схема вычисления частотного отклика элементарного фильтра без рециркуляции (рис. 8.7). Частотный отклик даётся длиной отрезка, соединяющего Z и Q в комплексной плоскости.

так что передаточная функция

$$H(Z) = 1 - QZ^{-1}.$$

Это может быть проанализировано графически, как показано на рис. 8.8. Действительные числа r и α являются модулем и аргументом комплексного числа Q:

$$Q = r \cdot (\cos(\alpha) + i \sin(\alpha)).$$

Коэффициент усиления фильтра равен расстоянию между точками Q и Z на комплексной плоскости. Аналитически мы можем видеть это, поскольку

$$|1 - QZ^{-1}| = |Z||1 - QZ^{-1}| = |Q - Z|.$$

Графически число QZ^{-1} есть в точности число Q, повёрнутое назад (по часовой стрелке) на угловую частоту ω входной синусоиды. Величина $|1 - QZ^{-1}|$ равна расстоянию между QZ^{-1} и 1 в комплексной плоскости, которое равно расстоянию между Q и Z.

Если частота входа пробегает от 0 до 2π, точка Z проходит единичный круг против часовой стрелки. В точке, где $\omega = \alpha$, расстояние достигает минимума, равного $1 - r$. Максимум возникает, когда Z находится в противоположной точке круга.

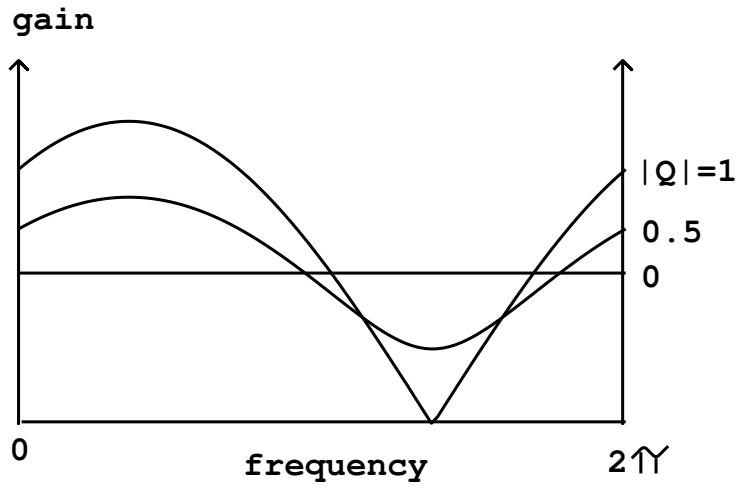


Рис. 8.9: Частотный отклик элементарного фильтра без рециркуляции рис. 8.7. Используются три значения Q, все с одинаковыми аргументами (-2 радиана), но с различными абсолютными величинами (модулями) $r = |Q|$.

Рисунок 8.8 показывает передаточную функцию для трёх различных значений $r = |Q|$.

8.2.2 Вторая форма фильтра без рециркуляции

Иногда нам будет нужен вариант фильтра выше, показанный на рис. 8.10, который называется *элементарным фильтром без рециркуляции второй формы*. Вместо того чтобы умножить задержанный выход на Q, мы умножаем прямой сигнал на комплексно сопряжённое $\overline{Q}$. Если

$$A = a + bi = r \cdot (\cos(\alpha) + i \sin(\alpha))$$

есть любое комплексное число, то его комплексно сопряжённое определяется как

$$\overline{A} = a - bi = r \cdot (\cos(\alpha) - i \sin(\alpha)).$$

Графически это отражает точки комплексной плоскости вверх и вниз относительно действительной оси. Передаточная функция нового фильтра есть

$$H(Z) = \overline{Q} - Z^{-1}.$$

Это приводит к тому же частотному отклику как перед этим, поскольку

$$|\overline{Q} - Z^{-1}| = |Q - \overline{Z^{-1}}| = |Q - Z|.$$

Здесь мы используем тот факт, что $\overline{Z} = Z^{-1}$ для любого комплексного числа Z с единичным модулем, что может быть проверено записью $Z\overline{Z}$ в полярной или прямоугольной форме.

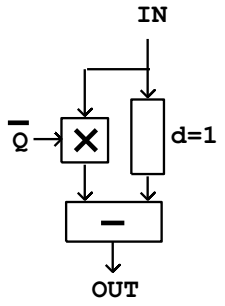


Рис. 8.10: Элементарный фильтр без рециркуляции второй формы.

Хотя две формы элементарного фильтра без рециркуляции имеют одинаковый частотный отклик, их фазовые отклики различны; это иногда приводит нас к предпочтению второй формы.

8.2.3 Элементарный фильтр с рециркуляцией

Элементарный фильтр с рециркуляцией представляет из себя гребенчатый фильтр с рециркуляцией рис. 7.7 с комплекснозначным коэффициентом усиления обратной связи P, как показано на рис. 8.11 (часть а). С помощью такого же анализа как раньше, направление на эту сеть синусоиды с n-ым семплом Z^n даёт выход

$$\frac{1}{1 - PZ^{-1}} Z^n,$$

так что передаточная функция

$$H(Z) = \frac{1}{1 - PZ^{-1}}.$$

Фильтр с рециркуляцией устойчив, когда $|P| < 1$; если, наоборот, $|P| > 1$ то выход растёт экспоненциально при рециркуляции задержанных семплов. Функция рециркуляции есть, таким образом, обращение передаточной функции фильтра без рециркуляции (первой формы). Если вы поместите оба фильтра последовательно с $P = Q$, выход, теоретически, будет равен входу. (Проведённый анализ доказывает это только для синусоидальных входов; то, что это следует для других сигналов, также может быть проверено вычислением импульсного отклика объединённой сети.)

8.2.4 Составные фильтры

Мы можем использовать фильтры с рециркуляцией и без неё, разработанные здесь, для создания составного фильтра, объединённого из ряда нескольких элементарных. Если параметры фильтров без рециркуляции (первой формы) есть

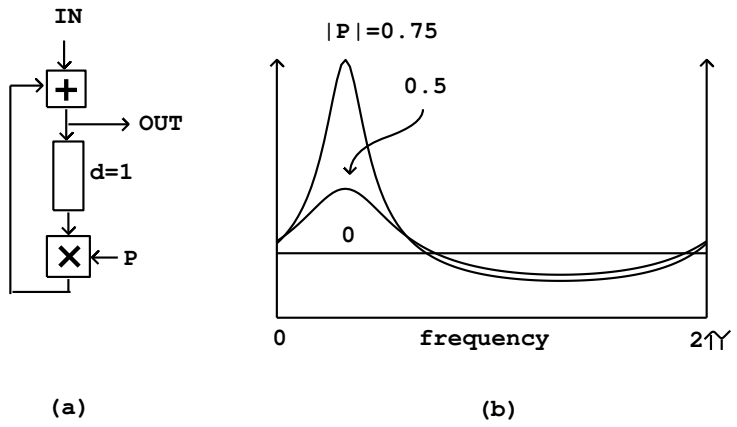


Рис. 8.11: Элементарный фильтр без рециркуляции второй формы.

$Q_1, \ldots, Q_j$ и фильтров с рециркуляцией $P_1, \ldots, P_j$, тогда расставляя их в ряд в любом порядке, мы получим передаточную функцию

$$H(Z) = \frac{(1 - Q_1 Z^{-1}) \cdots (1 - Q_j Z^{-1})}{(1 - P_1 Z^{-1}) \cdots (1 - P_k Z^{-1})}.$$

Частотный отклик получившегося составного фильтра является произведением частотных откликов элементарных составляющих. (Можно также скомбинировать элементарные фильтры, складывая их выходы или создавая более сложные схемы из них; но для большинства целей последовательная конфигурация является простейшей для работы.)

8.2.5 Вывод вещественных значений из комплексных фильтров

В большинстве приложений мы начинаем с вещественного сигнала для фильтрации, и нам нужен вещественный вывод, но, в общем случае, составной фильтр с передаточной функцией, как выше, будет давать комплекснозначный вывод. Тем не менее, мы можем конструировать фильтры с невещественными коэффициентами, которые, однако, выводят вещественные сигналы, так что анализ, который мы выполняем с использованием комплексных чисел может быть использован для предсказания, объяснения и управления вещественными выходными сигналами. Мы сделаем это с помощью соединения каждого элементарного фильтра (с коэффициентом P или Q) с другим, имеющим комплексно сопряжённый коэффициент $\overline{P}$ или $\overline{Q}$.

Например, устанавливая два фильтра без рециркуляции с коэффициентами Q и $\overline{Q}$ последовательно, получаем передаточную функцию, равную

$$H(Z) = (1 - QZ^{-1}) \cdot (1 - \overline{Q}Z^{-1}),$$

которая имеет свойство

$$H(\overline{Z}) = \overline{H(Z)}.$$

Теперь, если мы посылаем любую действительнозначную синусоиду

$$X_n = 2\,\mathrm{re}(AZ^n) = AZ^n + \overline{AZ}^n,$$

мы получаем

$$A \cdot H(Z) \cdot Z^n + \overline{A} \cdot \overline{H(Z)} \cdot \overline{Z}^n;$$

проверка показывает, что это другая действительнозначная синусоида. Здесь мы используем два свойства комплексного сопряжения. Во-первых, вы можете складывать и умножать с ним в любом порядке:

$$\overline{A+B} = \overline{A} + \overline{B}$$
$$\overline{AB} = \overline{A} \cdot \overline{B},$$

и, во-вторых, что-либо плюс его комплексно сопряжённое является действительным числом и на самом деле равно удвоенной действительной части исходного:

$$A + \overline{A} = 2\,\mathrm{re}(A).$$

Этот результат для двух сопряжённых фильтров расширяется на любой составной фильтр; в общем, мы всегда получаем действительнозначный вход, если подбираем коэффициенты так, что каждый Q_i и P_i в составном фильтре или вещественный, или появляется в паре со своим комплексно сопряжённым.

8.2.6 Два фильтра с рециркуляцией по цене одного

При соединении двух элементарных фильтров с рециркуляцией можно избежать вычислений для каждого фильтра из пары, поскольку вход действительнозначный (и выход так же). Предположим, что вход является вещественной синусоидой вида

$$AZ^n + \overline{A}Z^{-n},$$

и мы применяем один фильтр с рециркуляцией с коэффициентом P. Обозначая $a[n]$ действительную часть выхода, получим:

$$a[n] = \mathrm{re}\left[\tfrac{1}{1-PZ^{-1}}AZ^n + \tfrac{1}{1-PZ}\overline{A}Z^{-n}\right]$$

$$= \mathrm{re}\left[\tfrac{1}{1-PZ^{-1}}AZ^n + \tfrac{1}{1-\overline{P}Z^{-1}}AZ^n\right]$$

$$= \mathrm{re}\left[\tfrac{2-2\,\mathrm{re}(P)Z^{-1}}{(1-PZ^{-1})(1-\overline{P}Z^{-1})}AZ^n\right]$$

$$= \mathrm{re}\left[\tfrac{1-\mathrm{re}(P)Z^{-1}}{(1-PZ^{-1})(1-\overline{P}Z^{-1})}AZ^n + \tfrac{1-\mathrm{re}(P)\overline{Z}^{-1}}{(1-\overline{P}\overline{Z}^{-1})(1-P\overline{Z}^{-1})}\overline{AZ}^{-n}\right].$$

(На втором шаге мы использовали тот факт, что можно взять сопряжённое от всего или от части выражения без изменения результата, если вы лишь собираетесь взять действительную часть каким-либо образом. На четвёртом шаге сделано то же самое в обратном направлении.) Сравнивая вход и выход мы видим, что эффект прохождения вещественного сигнала через комплексный однополюсный фильтр с дальнейшим взятием действительной части эквивалентен прохождению сигнала через двухполюсный фильтр с одним нулём и передаточной функцией, равной

$$H_{\mathrm{re}}(Z) = \frac{1 - \mathrm{re}(P)Z^{-1}}{(1 - PZ^{-1})(1 - \overline{P}Z^{-1})}.$$

Сходное вычисление показывает, что взятие мнимой части (рассматриваемой как вещественный сигнал) эквивалентно фильтрации входа с передаточной функцией

$$H_{\mathrm{im}}(Z) = \frac{\mathrm{im}(P)Z^{-1}}{(1 - PZ^{-1})(1 - \overline{P}Z^{-1})}.$$

Так что взятие действительной или мнимой части выхода однополюсного фильтра даёт фильтр с двумя сопряжёнными полюсами. Две части могут быть скомбинированы для создания фильтров с другими возможными числителями; другими словами, с помощью одного комплексного фильтра с рециркуляцией мы можем создать фильтр, который действует на действительные сигналы с двумя (комплексно сопряжёнными) полюсами и одним (действительным) нулём.

Этот метод, известный как *метод элементарных дробей*, может повторяться для любого числа этапов последовательно, пока мы вычисляем подходящую комбинацию действительной и мнимой частей выхода на каждом этапе для формирования (вещественного) входа следующего этапа. Кажется, не существует сходной короткой комбинации для фильтров без рециркуляции; для них необходимо вычислять каждый член каждой комплексно сопряжённой пары точно.

8.3 Проектирование фильтров

Частотный отклик ряда элементарных фильтров с рециркуляцией и без может быть оценён графически с помощью построения всех коэффициентов $Q_1, \ldots, Q_j$ и $P_1, \ldots, P_k$ на комплексной плоскости и такого же рассуждения, как на рис. 8.8. Общий частотный отклик является произведением всех расстояний от точки Z до каждой из точек Q_i, делённым на произведение всех расстояний до каждой из P_i.

Обычно отмечают каждый из Q_i как “o” (называя “нулём”) и каждый P_i как “x” (называя “полюсом”); эти названия заимствованы из области комплексного анализа. График, показывающий полюсы и нули, связанные с фильтром, незамысловато называется *графиком полюсов-нулей*.

Когда Z близко к нулю, частотный отклик спадает, а когда к полюсу — растёт. Эффект полюса или нуля проявляется более значительно и локально одновременно,

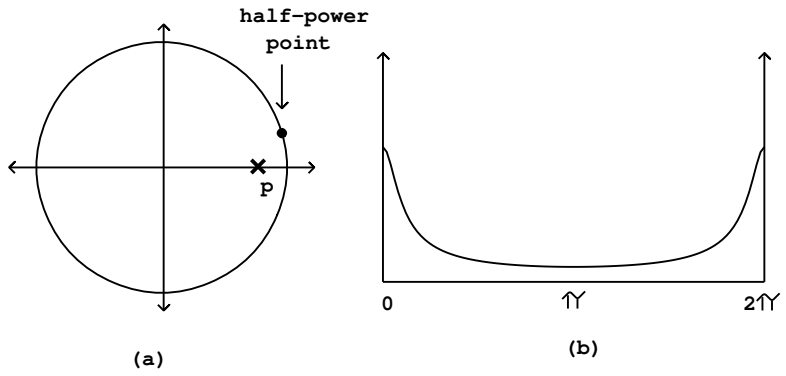


Рис. 8.12: Однополюсный фильтр низких частот: (a) нуль-полюсная диаграмма; (b) частотный отклик.

если Z прближается к единичной окружности, которой оно ограничено. Полюсы должны находиться внутри единичного круга, чтобы фильтр был устойчив. Нули могут лежать внутри или вне его, но любой ноль Q вне единичной окружности может быть заменён нулём внутри него, в точке $1/\sqrt{Q}$, для того чтобы получить постоянное кратное того же частотного отклика. Исключая специальные случаи, мы будем помещать нули, так же как и полюсы, внутри окружности.

В оставшейся части этого параграфа мы будем показывать, как конструировать несколько типов фильтров, наиболее широко используемых в электронной музыке. Теория дизайна цифровых фильтров очень обширна, и здесь мы дадим лишь введение. Глубокое рассмотрение, проведённое Юлиусом Смитом, доступно в интернете по адресу `ccrma.stanford.edu`. Смотрите также [Ste96] для введения в дизайн фильтров с более общей точки зрения обработки цифровых сигналов.

8.3.1 Однополюсный фильтр низких частот

Однополюсный фильтр, пропускающий низкие частоты, имеет единственный полюс, помещённый в положительную действительную точку p, как изображено на рис. 8.12. Это в точности гребенчатый фильтр с рециркуляцией с задержкой длины $d = 1$, так что применяется анализ §7.4. Максимальный коэффициент усиления возникает на нулевой частоте в соответствии с точкой круга, наиболее близкой к точке p. Коэффициент усиления здесь равен $1/(1-p)$. Предполагая, что p близко к единице, если мы перемещаем точку на $1-p$ единиц вверх или вниз от действительной (горизонтальной) оси, расстояние возрастает на коэффициент около $\sqrt{2}$, и, таким образом, мы ожидаем, что точка половинной мощности возникнет для угловой частоты около $1-p$.

Это вычисление часто проделывается в обратном направлении: если мы хотим, чтобы точка половинной мощности лежала на заданной угловой частоте ω, мы

устанавливаем $p = 1-\omega$. Эта аппроксимация хорошо работает, только если величина ω существенно ниже $\pi/2$, что часто бывает на практике. Общепринятой является нормализация однополюсного фильтра низких частот с помощью умножения его на постоянный коэффициент $1 - p$, для того чтобы дать единичный коэффициент усиления на нулевой частоте; ненулевые частоты будут тогда иметь коэффициент усиления меньше единицы.

Частотный отклик изображён графически на рис. 8.12 (часть b). Слышимые частоты доходят только до середины графика; правая часть кривой частотного отклика соответствует частотам выше частоты Найквиста π.

Однополюсный фильтр низких частот часто используется для нахождения динамики изменений в зашумленных сигналах. Например, если вы используете физический контроллер, и вам нужны только изменения порядка 1/10 секунды, вы можете сгладить значения фильтром низких частот, у которого точка половинной мощности составляет 20 или 30 циклов в секунду.

8.3.2 Однополюсный фильтр высоких частот с одним нулём

Иногда аудиосигнал имеет нежелательное постоянное смещение или, другими словами, компоненту нулевой частоты. Например, спектры при вейвшейпинге §5.3 почти всегда содержат постоянную составляющую. Она не воспринимается на слух, но, поскольку она несёт электрическую мощность, которую вы отправляете на ваши громкоговорители, её присутствие уменьшает уровень громкости, который вы могли бы достигнуть без искажения. Другое название постоянной компоненты сигнала — “DC”, что означает “direct current” (“постоянный ток”).

Простой и практичный способ удаления компоненты нулевой частоты из аудиосигнала состоит в использовании однополюсного фильтра низких частот с одним нулём, который выделяет её с дальнейшим вычитанием результата из сигнала. Получающаяся передаточная функция равна разнице единицы и передаточной функции фильтра низких частот:

$$H(Z) = 1 - \frac{1-p}{1-pZ^{-1}} = p\frac{1-Z^{-1}}{1-pZ^{-1}}.$$

Коэффициент $1-p$ в числителе низкочастотной передаточной функции является нормирующим множителем, необходимым, чтобы коэффициент усиления равнялся единице на нулевой частоте.

Исследуя правую часть уравнения (сравнивая её с общей формулой для сложного фильтра), мы видим, что мы по-прежнему имеем полюс в действительном числе p, а также ещё ноль в точке 1. Нуль-полюсная диаграмма показана на рис. 8.13 (часть a), и частотный отклик — в части (b). (С этого момента мы будем только строить частотный отклик до частоты Найквиста π; в предыдущем примере мы построили его везде где можно, вплоть до частоты семплирования 2π.)

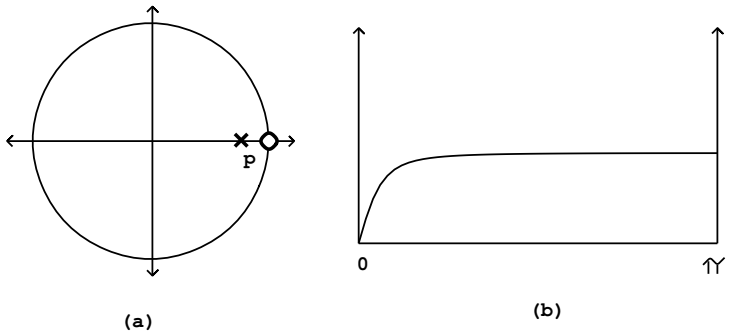


Рис. 8.13: Однополюсный фильтр высоких частот с одним нулём: (a) нуль-полюсная диаграмма; (b) частотный отклик (от нуля до частоты Найквиста).

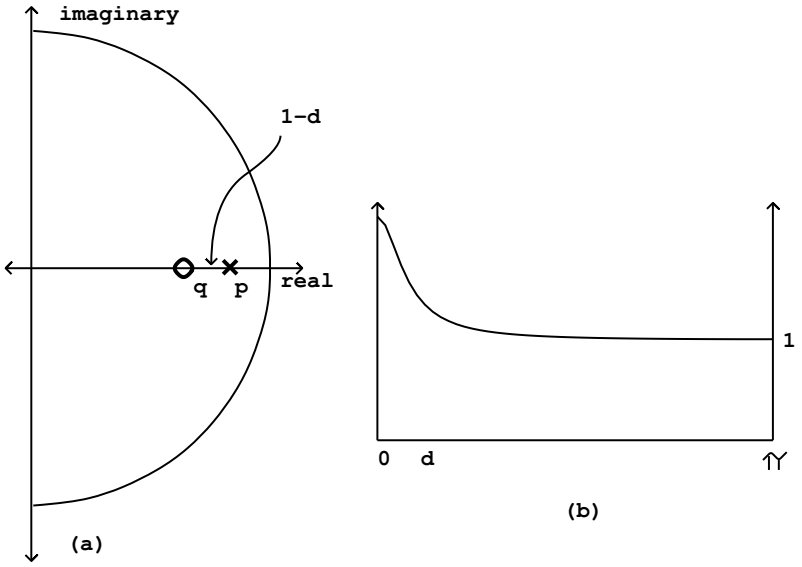


Рис. 8.14: Однополюсный полочный фильтр с одним нулём: (a) нуль-полюсная диаграмма; (b) частотный отклик.

8.3.3 Полочный фильтр

Обобщая фильтр с одним полюсом и одним нулём, введённый выше, предположим, что мы помещаем ноль в точке q, действительном числе — близком к единице, но меньше неё. Полюс в точке p расположен сходным образом и может быть или больше, или меньше q, то есть справа или слева от него, соответственно, но так, чтобы q и p оставались внутри единичного круга. Эта ситуация изображена на рис. 8.14.

В точках круга, далёких от p и q, эффекты наличия полюса и нуля почти взаимообратны (расстояния до них почти равны), так что фильтр пропускает такие частоты почти без изменения. В окрестности p и q, с другой стороны, фильтр будет иметь коэффициент усиления выше или ниже единицы, в зависимости от того, p или q ближе к единичной окружности. Поэтому эта конфигурация действует

как полочный фильтр низких частот. (Для того чтобы сделать полочный фильтр высоких частот, мы делаем то же самое, только помещаем p и q рядом с -1 вместо $+1$.)

Для нахождения параметров полочного фильтра с заданными требуемой переходной частотой ω (в угловых единицах) и низкочастотным коэффициентом усиления g, во-первых, мы выбираем среднее расстояние d, как изображено на рисунке, от полюса и нуля до кромки круга. Для маленьких величин d область влияния составляет приблизительно d радиан, так что просто устанавливаем $d = \omega$ для получения требуемой переходной частоты.

Далее установим полюс в $p = 1 - d/\sqrt{g}$ и ноль в $q = 1 - d\sqrt{g}$. Коэффициент усиления на нулевой частоте есть тогда

$$\frac{1-q}{1-p} = g,$$

как и требовалось. Например, на рисунке 8.14, d есть 0.25 радиан, и g равно 2.

8.3.4 Полосовой фильтр

Начиная с трёх типов фильтров, показанных выше, имеющих вещественные полюсы и нули, мы теперь трансформируем их для оперирования с полосами, так чтобы они лежали вне действительной оси. Тогда ФНЧ, ФВЧ и полочный фильтры становятся полосовыми фильтрами, полосно-заграждающими фильтрами, а также пиковыми фильтрами. Во-первых, мы разработаем полосовой фильтр. Предположим, что мы хотим центральную частоту в ω радиан и ширину полосы в β. Мы берём фильтр низких частот с частотой среза β; для малых значений β его полюс расположен приблизительно в $p = 1 - \beta$. Теперь повернём это значение на ω радиан в комплексной плоскости, то есть умножим на комплексное число $\cos\omega + i\sin\omega$. Новый полюс будет в

$$P_1 = (1-\beta)(\cos\omega + i\sin\omega).$$

Для получения действительного значения на выходе, он должен быть объединён в пару с другим полюсом:

$$P_2 = \overline{P_1} = (1-\beta)(\cos\omega - i\sin\omega).$$

Получающаяся нуль-полюсная диаграмма будет такой, как показано на рис. 8.15.

Пик расположен приблизительно (но не точно) на требуемой частоте ω, и частотный отклик падает на 3 децибела через приблизительно β радиан вверх или вниз от него. Часто требуется нормировать фильтр так, чтобы он имел приблизительно единичный пиковый коэффициент усиления; это делается умножением входа или выхода на произведение расстояний двух полюсов до пика на окружности, или (очень приблизительно) на

$$\beta * (\beta + 2\omega)$$

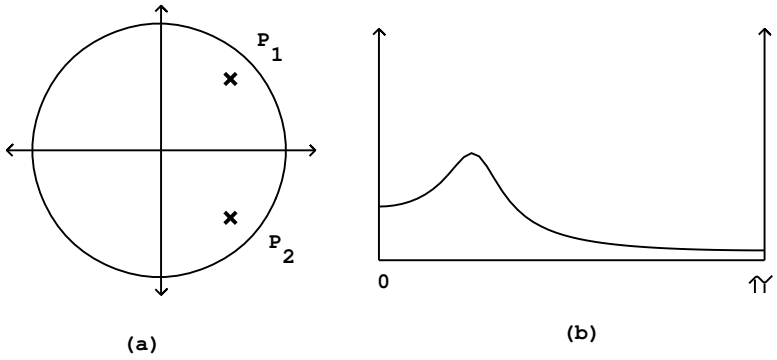


Рис. 8.15: Двуполюсный полосовой фильтр: (a) нуль-полюсная диаграмма; (b) частотный отклик.

Для некоторых случаев требуется добавление нуля в точках 1 и -1, так что коэффициент усиления падает до нуля на угловых частотах 0 и π.

8.3.5 Пиковый и режекторный фильтры

Таким же образом пиковый фильтр получается из полочного фильтра поворотом полюса и нуля и созданием их комплексного сопряжения, как показано на рис. 8.16. Если требуемая центральная частота есть ω и радиусы полюса и нуля (как для полочного фильтра) есть p и q, тогда мы помещаем верхние полюс и нуль в

$$P_1 = p \cdot (\cos\omega + i\sin\omega)$$
$$Q_1 = q \cdot (\cos\omega + i\sin\omega).$$

В качестве специального случая, помещение нуля на единичную окружность даёт задерживающий полосовой фильтр; в этом случае коэффициент усиления на центральной частоте будет нулём. Это аналогично фильтру высоких частот с одним полюсом и одним нулём выше.

8.3.6 Фильтры Баттерворта

Фильтр с одним вещественным полюсом и одним вещественным нулём может быть создан как полочный фильтр, как фильтр высоких частот (устанавливая ноль в точку $+1$) или как фильтр низких частот (устанавливая ноль в точку -1). Частотные отклики этих фильтров достаточно пологие; другими словами, переходная область широка. Часто требуется получить фильтр с более крутым переходом: или полочный, или низко-, или высоко- частотный, чьи две полосы являются более плоскими и разделены узкой переходной областью.

Процедура, заимствованная из мира аналоговых фильтров, преобразует действительные фильтры с одним полюсом и одним нулём в соответствующие фильтры

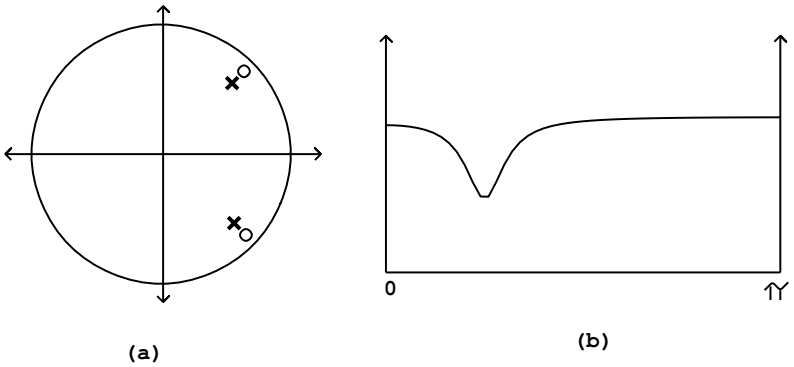


Рис. 8.16: Пиковый фильтр: (a) нуль-полюсная диаграмма; (b) частотный отклик. Здесь фильтр установлен на ослабление в 6 децибел на центральной частоте.

Баттерворта, которые имеют более узкие переходные области. Эта процедура четко и элегантно описана в последней главе [Ste96]. Этот расчет требует больше математических знаний, чем мы здесь рассмотрели, поэтому мы просто представим здесь результат без доказательства.

Для того чтобы сделать фильтр Баттерворта из ФНЧ, ФВЧ или полочного фильтров, предположим, что или полюс, или ноль задаётся выражением

$$\frac{1-r^2}{(1+r)^2},$$

где r есть параметр в диапазоне от 1 до ∞. Если $r=0$, получаем 1, и если $r=\infty$ получаем -1.

Далее, по причинам, в которые не будем углублятся, мы заменяем точку (полюс или ноль) на n точек, заданных как

$$\frac{(1-r^2)-(2r\sin(\alpha))i}{1+r^2+2r\cos(\alpha))}$$

где α принимает значения:

$$\frac{\pi}{2}(\frac{1}{n}-1),\ \frac{\pi}{2}(\frac{3}{n}-1),\ \ldots,\ \frac{\pi}{2}(\frac{2n-1}{n}-1).$$

Иными словами, α принимает n последовательно равноудалённых значений между $-\pi/2$ и $\pi/2$. Точки размещены в комплексной плоскости так, как показано на рис. 8.17. Они лежат на окружности, проведённой через первоначальные действительнозначные точки, которая пересекается с единичной окружностью под прямыми углами.

Хорошая оценка переходной частоты или частоты среза, определённой этими дугообразными наборами полюсов или нулей, есть просто участок, где окружность пересекает единичный круг, что соответствует $\alpha=\pi/2$. Это даёт точку

$$\frac{(1-r^2)-2ri}{1+r^2},$$

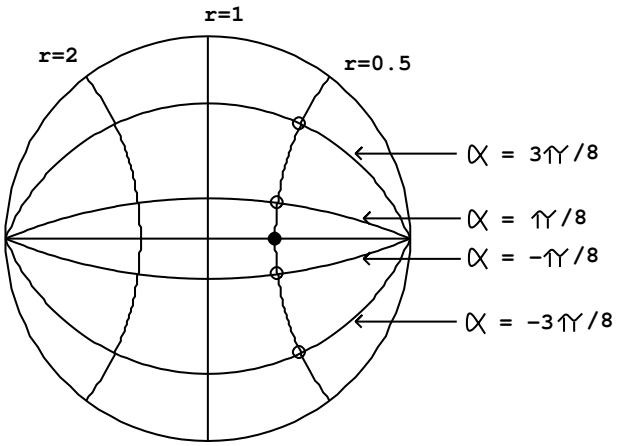


Рис. 8.17: Замена действительного полюса или нуля (показан жирной точкой) на набор из четырёх полюсов или нулей (на окружности) как для фильтра Баттерворта. В этом примере мы получаем четыре новых полюса или нуля, как показано, лежащих на окружности радиуса $r = 0.5$.

которая после некоторых преобразований даёт угловую частоту, равную

$$\beta = 2\arctan(r).$$

Рисунок 8.18 (часть a) показывает нуль-полюсную диаграмму и частотный отклик для фильтра низких частот Баттерворта с тремя полюсами и тремя нулями. Часть (b) показывает частотный отклик фильтра низких частот и трёх других фильтров, полученных выбором различных значений β (и, следовательно, r) для нулей, оставляя, в то же время, полюса стационарными. Когда нули меняются от $\beta = \pi$ до $\beta = 0$, низкочастотный фильтр становится полочным и далее высокочастотным.

8.3.7 Растяжение единичной окружности рациональными функциями

В §8.3.4 мы видели простой способ полочения полосового фильтра из фильтра низких частот. Заманчиво применить тот же метод для переделывания фильтра низких частот Баттерворта в высококачественный полосовой фильтр; но если мы хотим сохранить высокое качество фильтра Баттерворта, мы должны быть более осторожными, чем раньше, при проектировании используемого преобразования. В этом параграфе мы разработаем способ сделать фильтр низких частот Баттерворта с помощью введения класса рациональных преобразований комплексной плоскости, которые сохраняют единичную окружность.

Этот подход мы позаимствовали из [PB87, стр. 201–206] (Я признателен Юлиусу Смиту за указание на него). В нём преобразование выполняется в непрерывном времени, но здесь мы адаптировали этот метод для работы с дискретным временем, для того чтобы сделать рассуждение самодостаточным.

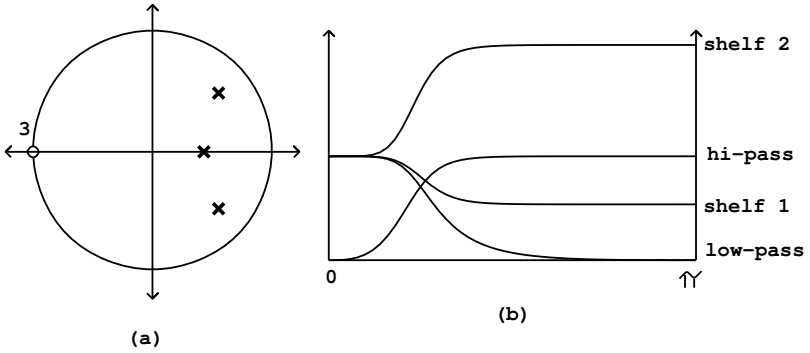


Рис. 8.18: Фильтр низких частот Баттерворта с тремя полюсами и тремя нулями: (a) нуль-полюсная диаграмма. Полюсы выбраны для частоты среза $\beta = \pi/4$; (b) частотные отклики для четырёх фильтров с одинаковой конфигурацией полюсов, но разными размещениями нулей (с остающимися на месте полюсами). Фильтр низких частот получается при установке $\beta = \pi$ для нулей; два полочных фильтра соответствуют $\beta = 3\pi/10$ и $\beta = 2\pi/10$, и наконец фильтр высоких частот получается установкой $\beta = 0$. Фильтр высоких частот нормирован для получения единичного коэффициента усиления на частоте Найквиста, а все остальные для единичного коэффициента усиления на DC.

Идея состоит в том, чтобы начать с любого фильтра с такой же передаточной функцией как раньше:

$$H(Z) = \frac{(1 - Q_1 Z^{-1}) \cdots (1 - Q_j Z^{-1})}{(1 - P_1 Z^{-1}) \cdots (1 - P_k Z^{-1})},$$

чей частотный отклик (коэффициент усиления на частоте ω) даётся выражением

$$|H(\cos(\omega) + i \sin(\omega))|.$$

Теперь предположим, что мы можем найти рациональную функцию $R(Z)$, которая деформирует единичную окружность каким-либо требуемым образом. То что R является рациональной функцией означает, что она может быть представлена в виде отношения двух полиномов (например, передаточная функция H является рациональной функцией). Эта R перемещает точки единичной окружности в другие точки единичной окружности в точности тогда, когда $|R(Z)| = 1$, если $|Z| = 1$. Нетрудно проверить, что любая функция вида

$$R(Z) = U \cdot \frac{A_n Z^n + A_{n-1} Z^{n-1} + \cdots + A_0}{\overline{A_0} Z^n + \overline{A_1} Z^{n-1} + \cdots + \overline{A_n}}$$

(где $|U| = 1$) имеет это свойство. То же рассуждение, как в §8.2.2, подтверждает, что $|R(Z)| = 1$, если $|Z| = 1$.

Если мы имеем подходящую рациональную функцию R, мы можем просто составить её композицию с первоначальной передаточной функцией H для создания новой рациональной функции

$$J(Z) = H(R(Z)).$$

Коэффициент усиления нового фильтра J на частоте ω равен тогда коэффициенту усиления H на другой частоте ϕ, выбранной так, что

$$\cos(\phi) + i\sin(\phi) = R(\cos(\omega) + i\sin(\omega)).$$

Функция R перемещает точки по кругу на единичной окружности; J в любой точке равен H в точке, в которую R её сдвигает.

Например, предположим, что мы начинаем с фильтра низких частот с одним нулём и одним полюсом:

$$H(Z) = \frac{1 + Z^{-1}}{1 - gZ^{-1}}$$

и применяем функцию

$$R(Z) = -Z^2 = -\frac{1 \cdot Z^2 + 0 \cdot Z + 0}{0 \cdot Z^2 + 0 \cdot Z + 1}.$$

Геометрически этот выбор R равномерно растягивает единичную окружность в два раза по длине и закручивает её вокруг себя дважды. Обе точки 1 и -1 перемещаются в точку -1, а точки i и $-i$ перемещаются в точку 1. Получающаяся передаточная функция есть

$$J(Z) = \frac{1 - Z^{-2}}{1 + gZ^{-2}} = \frac{(1 - Z^{-1})(1 + Z^{-1})}{(1 - i\sqrt{g}Z^{-1})(1 + i\sqrt{g}Z^{-1})}.$$

Нуль-полюсные диаграммы H и J показаны на рис. 8.19. Начиная с фильтра низких частот, мы заканчиваем полосовым фильтром. Точки i и $-i$, которые R перемещает в 1 (где первоначальный коэффициент усиления фильтра наибольший), становятся точками наибольшего коэффициента усиления для нового фильтра.

8.3.8 Полосовой фильтр Баттерворта

Мы можем применить преобразование $R(Z) = -Z^2$ для преобразования фильтра Баттерворта в высококачественный полосовой фильтр с центральной частотой $\pi/2$. После этого дальнейшее преобразование может быть применено для сдвига центральной частоты на любое желаемое значение ω между 0 и π. Преобразование будет иметь вид

$$S(Z) = \frac{aZ + b}{bZ + a},$$

где a и b есть действительные числа, одновременно не равные нулю. Это частный случай общего вида введённых выше рациональных функций, сохраняющих единичный окружность. Мы имеем $S(1) = 1$ и $S(-1) = -1$ и верхняя с нижней половинки единичного окружности преобразуются симметрично (если Z переходит в W, тогда $\overline{Z}$ переходит в $\overline{W}$). Качественный эффект преобразования S состоит в прижатии точек единичной окружности к 1 или -1.

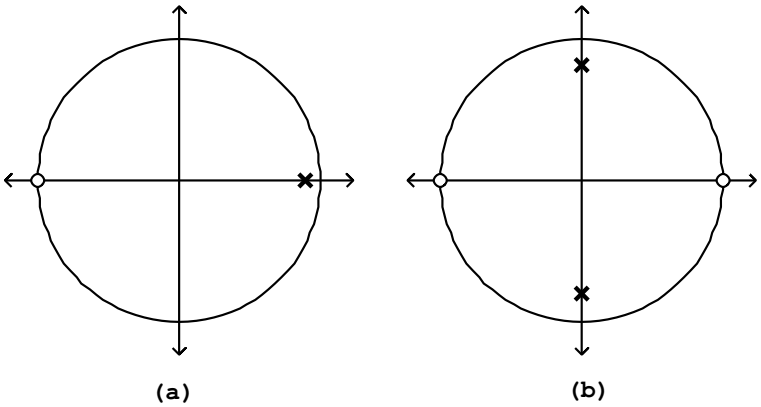


Рис. 8.19: Фильтр низких частот с одним полюсом и одним нулём: (a) нуль-полюсная диаграмма; (b) диаграмма для фильтра, получающегося после преобразования $R(Z) = -Z^2$. Результатом является полосовой фильтр с центральной частотой $\pi/2$.

В частности, для требуемой центральной частоты ω мы хотим выбрать S, так чтобы

$$S(\cos(\omega) + i\sin(\omega)) = i.$$

Если мы оставим $R = -Z^2$, как раньше, и обозначим H передаточную функцию для фильтра низких частот Баттерворта, тогда комбинированный фильтр с передаточной функцией $H(R(S(Z)))$ будет полосовым фильтром с центральной частотой ω. Решение относительно a и b даёт

$$a = \cos(\frac{\pi}{4} - \frac{\omega}{2}),\ b = \sin(\frac{\pi}{4} - \frac{\omega}{2}).$$

Новая передаточная функция, $H(R(S(Z)))$ будет иметь $2n$ полюсов и $2n$ нулей (если n равно степени фильтра Баттерворта H). Знать передаточную функцию — это хорошо, но еще лучше знать расположение всех полюсов и нулей нового фильтра, которые нам нужны для его вычисления с использованием элементарных фильтров. Если Z есть полюс передаточной функции $J(Z) = H(R(S(Z)))$, то есть, если $J(Z) = \infty$, тогда $R(S(Z))$ должно быть полюсом H. То же действует для нулей. Для нахождения полюса или нуля J мы устанавливаем $R(S(Z)) = W$, где W есть полюс или ноль H, и решаем относительно Z. Это даёт

$$-\left[\tfrac{aZ+b}{bZ+a}\right]^2 = W$$

$$\tfrac{aZ+b}{bZ+a} = \pm\sqrt{-W}$$

$$Z = \tfrac{\pm a\sqrt{-W}-b}{\mp b\sqrt{-W}+a}.$$

(Здесь a и b те же, что выше, и мы использовали тот факт, что $a^2 + b^2 = 1$). В качестве образца на рис. 8.20 приведена нуль-полюсная диаграмма и частотный отклик J.

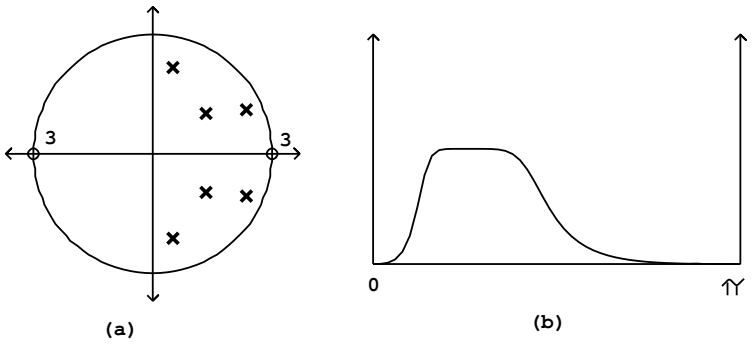


Рис. 8.20: Полосовой фильтр Баттерворта: (a) нуль-полюсная диаграмма; (b) частотный отклик. Центральная частота есть $\pi/4$. Ширина полосы зависит как от центральной частоты, так и от ширины полосы используемого первоначально фильтра низких частот Баттерворта.

8.3.9 Изменяющиеся во времени коэффициенты

В некоторых рекурсивных конструкциях фильтров изменение их коэффициентов может ввести дополнительную энергию в систему. Физическим аналогом является ребёнок в качающейся колыбели. Ребёнок качается на резонансной частоте системы, и, толкая или притягивая колыбель, вы гладко вводите или выводите энергию из системы. Однако если вы решаете укоротить цепочку или передвинуть всю колыбель, вы можете ввести непредсказуемое количество энергии в систему. Та же самая вещь может случиться, когда вы меняете коэффициенты в резонансном фильтре с рециркуляцией.

Для простых фильтров с одним нулём и одним полюсом, используемых здесь, такая проблема не возникает; если коэффициент усиления обратной связи или коэффициент прямого усиления меняется гладко (в смысле амплитудной огибающей), то выходной сигнал фильтра будет также вести себя гладко. Но трудности начинают появляться при попытке нормировать выход рекурсивного фильтра, когда коэффициент обратной связи близок к единице. Например, предположим, что мы имеем фильтр низких частот с одним полюсом и коэффициентом усиления 0.99 (для частоты среза 0.01 радиан или 70 герц при обычной частоте семплирования). Для его нормирования на единичный коэффициент усиления для DC, мы умножаем на 0.01. Предположим, что мы хотим удвоить срезающую частоту, изменяя коэффициент усиления слегка, до 0.98. Это прекрасно за исключением того, что нормировочный множитель неожиданно удваивается. Если мы умножаем выход фильтра на нормирующий множитель, выход будет внезапно, хотя возможно только на какой-то момент, совершать прыжок с коэффициентом два.

Решение заключается в том, чтобы нормировать вход, а не выход фильтра. рис. 8.21 (часть a) показывает комплексный фильтр с рециркуляцией и коэффициентом усиления обратной связи P, нормированный на входе на $1 - |P|$, так что пиковый коэффициент усиления становится равным единице. Часть (b) показывает неправильную реализацию, заключающуюся в умножении выхода.

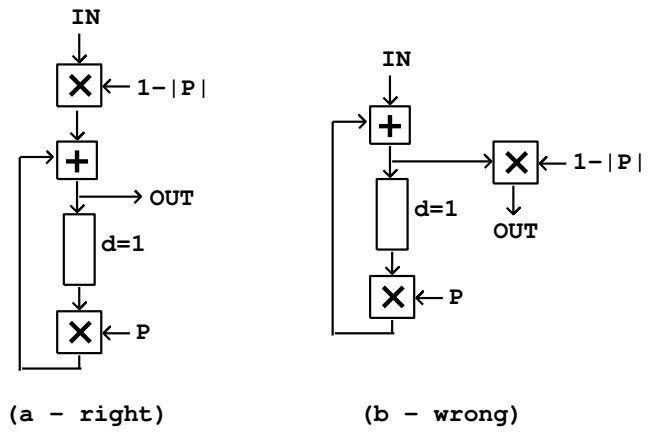


Рис. 8.21: Нормализация элементарного фильтра с рециркуляцией: (a) корректная, с умножением на нормировочный множитель входа; (b) некорректная, с умножением выхода.

Ситуация становится более сложной, когда несколько элементарных фильтров с рециркуляцией помещены последовательно, поскольку правильный нормирующий множитель, в общем, является функцией от всех коэффициентов. Когда требуется быстрое изменение для такого фильтра, один из возможных подходов заключается в нормировании каждого входа по отдельности, как если бы они действовали раздельно, и далее в умножении выхода, в конечном итоге, на какой-нибудь необходимый корректирующий множитель.

8.3.10 Импульсные отклики фильтров с рециркуляцией

В §7.4 мы проанализировали импульсный отклик гребенчатого фильтра с рециркуляцией, являющегося однополюсным фильтром низких частот специального вида. Рисунок 8.22 показывает результат для двух ФНЧ и одного комплексного резонансного фильтра с одним полюсом. Все они являются элементарными фильтрами с рециркуляцией, определёнными в §8.2.3. Каждый нормирован так, чтобы иметь единичный максимальный коэффициент усиления.

В случае фильтра низких частот, импульсный отклик становится длиннее (и ниже), когда полюс помещается ближе к единице. Предположим, что полюс находится в точке $1 - 1/n$ (так что частота среза равна $1/n$ радиан). Нормирующий множитель равен также $1/n$. После n точек, выход уменьшается на коэффициент

$$\left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e},$$

где e — есть константа Эйлера, ≈ 2.718. Можно сказать, что фильтр имеет *время стабилизации* в n семплов. На том же рисунке $n = 5$ для части (a), и $n = 10$ для части (b). В общем, время стабилизации (в семплах) приблизительно равно единице, делённой на частоту среза (в угловых единицах).

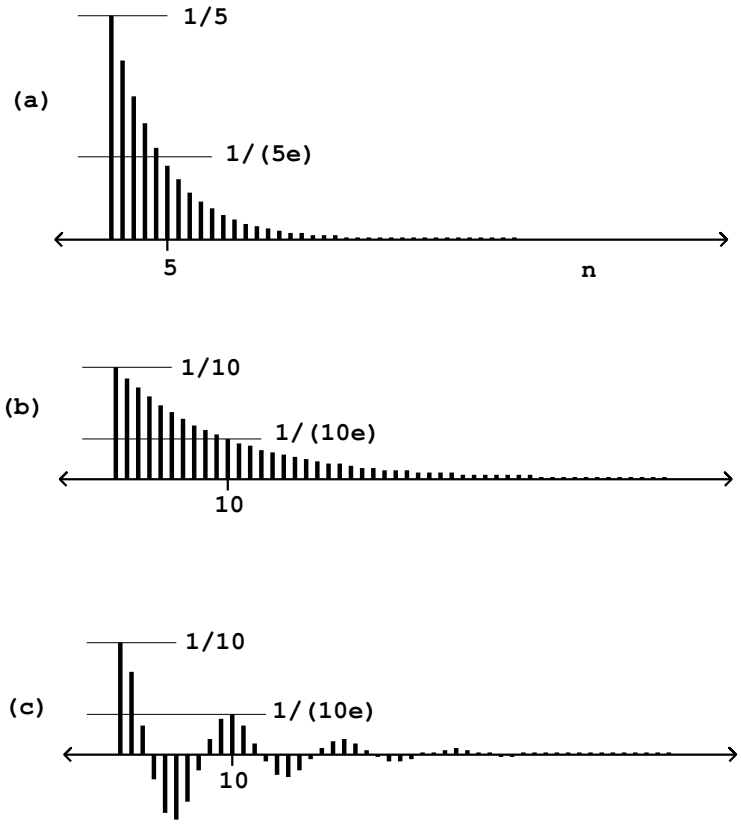


Рис. 8.22: Импульсный отклик для трёх элементарных фильтров (с одним полюсом), нормированных на пиковый коэффициент усиления 1: (a) ФНЧ с $P = 0.8$; (b) ФНЧ с $P = 0.9$; (c) полосовой фильтр (показана только действительная часть) с $|P| = 0.9$ и центральной частотой $2\pi/10$.

Ситуация становится еще интереснее, когда мы рассматриваем резонансный однополюсной фильтр, полюс которого лежит вне действительной оси. В части (c) рисунка полюс P имеет абсолютное значение 0.9 (как в части (b)), но его аргумент устанавливается на $2\pi/10$ радиан. Мы получаем такое же время стабилизации, как в части (b), но при этом выход пульсирует на резонансной частоте (с периодом 10 семплов в этом примере).

Возникает естественный вопрос: сколько периодов пройдет перед тем, как выходной сигнал фильтра спадет на $1/e$? Если полюс резонансного фильтра имеет модуль $1 - 1/n$, как мы видели в §8.2.3, то ширина полосы (назовём её b) приблизительно равна $1/n$, и мы видим, что время стабилизации приблизительно равно n. Резонансная частота (назовём её ω) равна аргументу полюса, а ее период — $2\pi/\omega$. Число периодов, которое составляет время стабилизации есть, таким образом,

$$\frac{n}{2\pi/\omega} = \frac{1}{2\pi}\frac{\omega}{b} = \frac{q}{2\pi},$$

где q есть *добротность* фильтра, определяемая как центральная частота, делённая на ширину полосы. Резонансные фильтры часто определяются в терминах центральной частоты и *"q"*, вместо ширины полосы.

8.3.11 Всепропускающие фильтры

Иногда фильтр применяется для получения требуемого фазового изменения, а не для изменения амплитуд частотных компонент звука. Для того чтобы сделать это нам бы понадобился способ создания фильтра с постоянным единичным частотным откликом, но меняющим фазу входной синусоиды в зависимости от её частоты. Мы уже видели в Главе 7, что задержка длины d определяет фазовый сдвиг $-d\omega$ для угловой частоты ω. Другой класс фильтров, называемых *всепропускающими фильтрами*, может создавать фазовые изменения в виде более интересных функций от ω.

Для построения всепропускающих фильтров, мы начнём с двух фактов: во-первых, элементарный фильтр с рециркуляцией и элементарный фильтр без рециркуляции полностью взаимоуничтожают действие каждого, если они имеют одинаковый коэффициент усиления. Другими словами, если сигнал был помещён на фильтр с одним нулём, действительным или комплексным, эффект может быть отменен с помощью последующего применения фильтра с одним полюсом и наоборот.

Второй факт состоит в том, что элементарный фильтр без рециркуляции второго вида имеет тот же частотный отклик как у первого вида; они отличаются только фазовым откликом. Таким образом, если мы скомбинируем элементарный фильтр с рециркуляцией и элементарный фильтр без рециркуляции второго вида, частотные отклики взаимоуничтожаются (до плоского коэффициента усиления, не зависящего от частоты), а фазовый отклик не является константой.

Для того чтобы найти передаточную функцию, мы выбираем одно комплексное

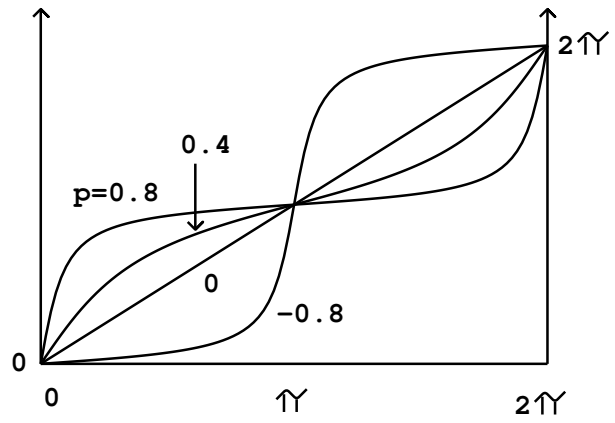


Рис. 8.23: Фазовые отклики всепропускающих фильтров с различными локализациями полюсов p. Когда полюс локализован в нуле, фильтр редуцируется в односемпловую задержку*.

* Исправление ошибки: на картинке в действительности показано обратное фазовому отклику — фазовые задержки. (*прим. перев.*)

число $P < 1$ для обоих элементарных фильтров и перемножаем их передаточные функции:

$$H(Z) = \frac{\overline{P} - Z^{-1}}{1 - PZ^{-1}}.$$

Коэффициент P управляет как локализацией единственного полюса (в самой точке P), так и нуля (в $1/\overline{P}$). Рисунок 8.23 показывает фазовый отклик всепропускающего фильтра для четырёх вещественных вариантов коэффициента P. На частотах 0, π и 2π фазовый отклик такой же, как при односемпловой задержке; но для частот между этими значениями фазовый отклик наклонён вверх или вниз в зависимости от коэффициента.

Комплексные коэффициенты дают сходные кривые фазового отклика, но частоты, на которых они пересекают диагональную линию на рисунке сдвинуты в соответствии с аргументом коэффициента P.

8.4 Применения

Фильтры используются в самых разных областях применения, как в звукорежиссуре, так и в электронной музыке. Последние включают, например, эквалайзеры, разделительные фильтры для акустических систем, ресемплинг, при удалении постоянной составляющей сигнала (которое мы уже использовали в предыдущих главах). Здесь, однако, мы будем рассматривать применения только в музыкальной области .

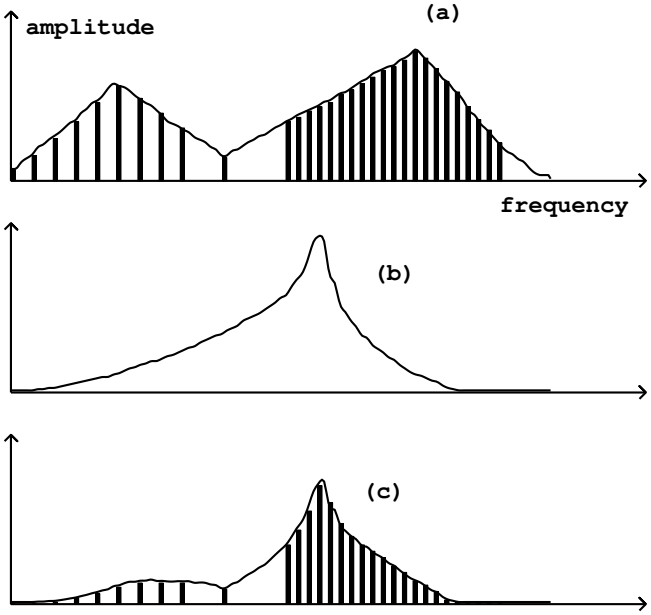


Рис. 8.24: Субтрактивный синтез: (a) спектр входного звука; (b) частотный отклик фильтра; (c) спектр выходного звука.

8.4.1 Субтрактивный синтез

Субтрактивный синтез является методом использования фильтров для формирования спектральной огибающей звука, создавая другой звук, сохраняющий обыкновенно такие качества первоначального звука как высота, неровность, шумность или зернистость. Спектральная огибающая получающегося звука является произведением спектральной огибающей первоначального звука и частотного отклика фильтра. Рисунок 8.24 показывает возможную конфигурацию источника, фильтра и результата.

Фильтр может быть постоянным или изменяющимся во времени. Субтрактивный синтез, широко используемый уже с середины 1950-х годов, произвел бум с появлением фильтра, управляемого напряжением (VCF — voltage controlled filter), который стал широко доступен с середины 1960-х с появлением модулярных синтезаторов. Типичный VCF имеет два входа: один — для звука, поступающего на фильтр, и один — для изменения центральной частоты или частоты среза фильтра.

Популярным применением VCF является управление центральной частотой резонансного фильтра от того же генератора ADSR, который управляет амплитудой; возможная блок-схема показана на рис. 8.25. Также с использованием этого фильтра можно сделать наиболее громкую часть ноты ярче других, более тихих частей (громкостью в целом управляет умножитель внизу); это может имитировать спектральную огибающую струнных или медных духовых инструментов на протяжении ноты.

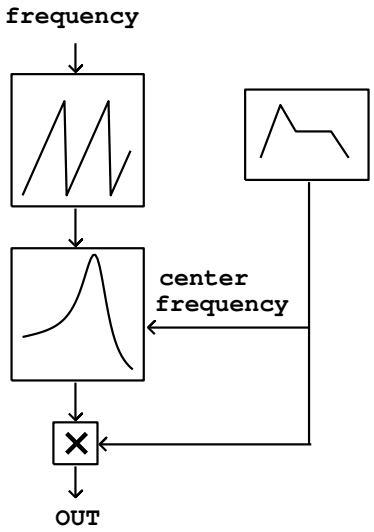


Рис. 8.25: Субтрактивный синтез с управлением ADSR.

8.4.2 Детектор огибающей

Часто бывает желательно использовать изменяющуюся во времени мощность входного сигнала для запуска или управления музыкальным процессом. Для этого нам понадобится процедура измерения мощности аудиосигнала. Поскольку большинство аудиосигналов проходят через ноль много раз в секунду, недостаточно взять мгновенные значения сигнала для измерения его мощности; Вместо этого мы должны рассчитать среднюю мощность за интервал времени, достаточно длинный, чтобы ее изменения не отображались в оценке мощности, но достаточно короткий, чтобы быстро сообщать об изменениях уровня сигнала. Вычисление, которое обеспечивает оценку мощности сигнала, зависящую от времени, называется *детектором огибающей*.

Выход фильтра низких частот может быть рассмотрен как скользящее среднее[11] его входа. Например, предположим, что мы применяем нормированный фильтр с одним полюсом и одним нулём с коэффициентом p, как на рис. 8.21, к входному сигналу $x[n]$. Выход (назовём его $y[n]$) является суммой выхода задержки, умноженного на p, и входа, умноженного на $1 - p$:

$$y[n] = p \cdot y[n-1] + (1-p) \cdot x[n],$$

так что каждый выходной сигнал усредняется с весом $1 - p$ с предыдущим выходным сигналом для создания нового выходного сигнала. Таким образом, мы можем рассчитать скользящее среднее квадрата аудиосигнала, используя диаграмму на рис. 8.26. Выход — это зависящее от времени среднее мгновенной мощности

[11] функция, значения которой в каждой точке определения равны некоторому среднему значению исходной функции за предыдущий период (*прим. перев.*)

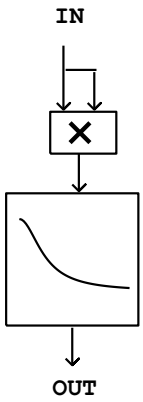


Рис. 8.26: Детектор огибающей. Выход является средней мощностью входного сигнала.

$x[n]^2$, и дизайн фильтра низких частот управляет, среди прочих вещей, временем стабилизации скользящего среднего.

Для большего понимания конструкции подходящего фильтра низких частот для детектора огибающей, мы проанализируем его с точки зрения спектров сигнала. Если, например, мы помещаем на вход действительнозначную синусоиду:

$$x[n] = a \cdot \cos(\alpha n),$$

то результат возведения в квадрат есть:

$$x[n]^2 = \frac{a^2}{2}\left(\cos(2\alpha n) + 1\right),$$

и, таким образом, если фильтр низких частот эффективно задерживает компоненту частоты 2α, мы будем получать приблизительно константу $a^2/2$, которая является, действительно, средней мощностью.

Ситуация для сигнала с несколькими компонентами сходная. Предположим, что входной сигнал теперь

$$x[n] = a \cdot \cos(\alpha n) + b \cdot \cos(\beta n),$$

чей спектр построен на рис. 8.27 (часть a). (Мы опустили два фазовых слагаемых, но они не имеют влияния на выходной сигнал.) Возведение сигнала в квадрат производит спектр, показанный в части (b) (см. §5.2). Мы можем получить требуемое фиксированное значение $(a^2 + b^2/2)$ просто фильтруя все другие компоненты; в идеале результат будет постоянным сигналом (DC). Пока мы отфильтровываем все гармоники, а также разностные тоны, мы получаем стабильный выходной сигнал, который даёт корректную оценку средней мощности.

Детекторы огибающих также могут быть полезны для шумовых сигналов, которые могут рассматриваться как сигналы с плотными спектрами. В этой ситуации будут появляться разностные частоты произвольно близкие к нулю, и их полная

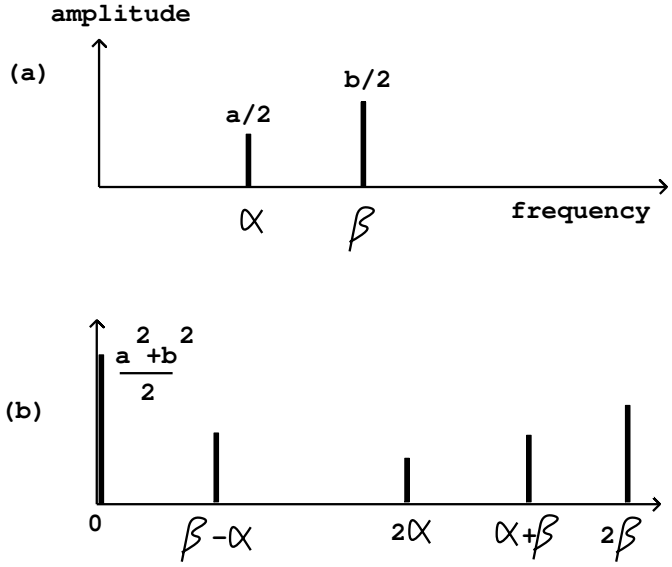


Рис. 8.27: Детектор огибающей со спектральной точки зрения: (a) входной сигнал с двумя компонентами; (b) результат возведения в квадрат.

фильтрация станет невозможной; мы будем всегда получать флуктуации на выходе, но они будут пропорционально уменьшаться при сужении пропускной полосы фильтра.

Хотя более узкая пропускная полоса будет всегда давать более чистый выходной сигнал, как для дискретных, так и для непрерывных спектров, время стабилизации фильтра будет увеличиваться пропорционально сужению пропускной полосы. Таким образом, существует компромисс между быстрым откликом и гладким результатом.

8.4.3 Однополосная модуляция

Как мы видели в Главе 5, перемножение двух вещественных синусоид приводит к сигналу с двумя новыми компонентами на сумме и разности первоначальных частот. Если мы выполним такую же операцию с комплексными синусоидами, мы получим только одну новую выходную частоту; это одно из следствий большей математической простоты комплексных синусоид по сравнению с вещественными. Если мы умножим одну комплексную синусоиду $1, Z, Z^2, \ldots$ на другую $1, W, W^2, \ldots$, то в результате получим $1, WZ, (WZ)^2, \ldots$ — третью комплексную синусоиду, частота которой $\angle(WZ)$ является суммой двух первоначальных частот.

В общем, поскольку комплексные синусоиды имеют более простые свойства, чем вещественные, часто полезно иметь возможность преобразовывать вещественные синусоиды в комплексные. Другими словами, для вещественной синусоиды

$$x[n] = a \cdot \cos(\omega n)$$

(со спектральным пиком амплитуды $a/2$ и частотой ω) мы хотели бы найти способ

вычисления комплексной синусоиды

$$X[n] = a\left(\cos(\omega n) + i\sin(\omega n)\right),$$

так чтобы

$$x[n] = \mathrm{re}(X[n]).$$

Для этого мы хотели бы использовать линейный процесс, чтобы суперпозиции синусоид обрабатывались так, как если бы их компоненты рассматривались отдельно.

Конечно, мы могли бы равным образом выбрать комплексную синусоиду с частотой $-\omega$:

$$X'[n] = a\left(\cos(\omega n) - i\sin(\omega n)\right)$$

и, действительно, $x[n]$ является в точности полусуммой двух таких комплексных синусоид. Главное, нам нужен фильтр, который будет пропускать положительные частоты (в действительности частоты между 0 и π, соответствующие значениям Z на верхней половине единичной окружности) и задерживать отрицательные (от $-\pi$ до 0 или, что эквивалентно от π до 2π — нижняя половина единичной окружности).

Можно сконструировать такой фильтр, создав фильтр низких частот с частотой среза $\pi/2$ радиан с использованием техники §8.3.4. Однако оказывается проще сделать его с использованием двух специально разработанных схем всепропускающих фильтров с действительными коэффициентами.

Обозначая функции перехода двух фильтров H_1 и H_2, мы строим фильтр так, что

$$\angle(H_1(Z)) - \angle(H_2(Z)) \approx \begin{cases} \pi/2 & 0 < \angle(Z) < \pi \\ -\pi/2 & -\pi < \angle(Z) < 0 \end{cases}$$

или, другими словами,

$$\begin{gathered} H_1(Z) \approx iH_2(Z),\ 0 < \angle(Z) < \pi \\ H_1(Z) \approx -iH_2(Z),\ -\pi < \angle(Z) < 0. \end{gathered}$$

Тогда для любого входного вещественного сигнала $x[n]$ мы просто задаём комплексное число $a[n] + ib[n]$, где $a[n]$ есть выход первого фильтра и $b[n]$ есть выход второго. Любая комплексная синусоидальная компонента $x[n]$ (обозначим её Z^n) будет переходить в

$$H_1(Z) + iH_2(Z) \approx \begin{cases} 2H_1(Z), & \text{если } 0 < \angle(Z) < \pi \\ 0, & \text{иначе.} \end{cases}$$

Таким образом, начав с вещественного сигнала, энергия которого делится поровну на положительные и отрицательные частоты, мы получили комплексный сигнал только с положительными частотами.

8.5 Примеры

В этом разделе мы сначала представим некоторые простые в использовании готовые фильтры, доступные в Pd, для работы с примерами, показывающими практические применения предыдущего раздела. Затем мы покажем некоторые более сложные случаи, для которых требуются специально разработанные фильтры.

8.5.1 Встроенные в Pd фильтры: полосовой, ФНЧ и ФВЧ

Примеры H01.low-pass.pd, H02.high-pass.pd и H03.band-pass.pd (рис. 8.28) показывают встроенные фильтры Pd, которые осуществляют конструкции фильтров, описанные в §§ 8.3.1, 8.3.2 и 8.3.4. Два из этих патчей также используют генератор шума, который мы не определили ранее. Нам потребуется четыре новых объекта Pd:

lop~ : *однополюсной фильтр низких частот.* Левый вход принимает сигнал, который должен быть отфильтрован, а правый вход получает управляющие сообщения для установки частоты среза фильтра. Фильтр нормирован так, что коэффициент усиления равен единице на частоте 0.

hip~ : *фильтр высоких частот с одним полюсом и одним нулём*, с такими же входами и выходами как lop~ , нормированный так, чтобы иметь единичный коэффициент усиления на частоте Найквиста.

bp~ : *резонансный фильтр.* Средний вход получает управляющие сообщения для установки центральной частоты, а правый вход для установки “q”[12].

noise~ : *генератор белого шума.* Все семплы представляют из себя независимые псевдослучайные числа, равномерно распределённые от -1 до 1.

Первые три патча примеров демонстрируют эти три фильтра (см. рис. 8.28). Объекты lop~ и bp~ показаны с шумом на входе; hip~ , как показано, используется для удаления компоненты DC (частоты 0) из сигнала.

8.5.2 Встроенный полосовой фильтр, изменяющийся во времени

Изменяемая по времени фильтрация с пропусканием полосы, часто используемая в классическом субтрактивном синтезе (§8.4.1), может быть осуществлена с использованием объекта vcf~ , введённого здесь:

[12] добротности фильтра (*прим. перев.*)

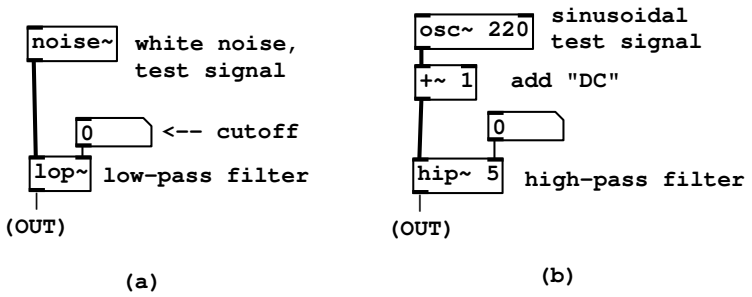


Рис. 8.28: Использование предустановленных фильтров в Pd: (a) ФНЧ с белым шумом в качестве тестового входа; (b) использование ФВЧ для удаления компоненты сигнала с частотой 0.

vcf~ : "управляемый напряжением" полосовой фильтр, сходный с bp~ , но с входом для сигнала, управляющего центральной частотой. Как bp~ , так и vcf~ являются резонансными фильтрами с одним полюсом, описанными в §8.3.4; bp~ выводит только действительную часть получающегося сигнала, а vcf~ выводит как действительную, так и мнимую части отдельно.

Пример H04.filter.sweep.pd (рис. 8.29) демонстрирует использование объекта vcf~ для простой и характерной задачи субтрактивного синтеза. Объект phasor~ (сверху) создаёт пилообразную волну для фильтрации. (Это не особенно хорошая практика, так как мы не контролируем свертку спектра; более совершенный генератор пилообразной волны для этой цели будет разработан в Главе 10.) Второй объект phasor~ (отмеченный как "LFO for sweep") управляет изменяющейся во времени центральной частотой. После настройки, устанавливающей глубину и базовую центральную частоту (заданных в единицах MIDI), результат переводится в герцы (с использованием объекта tabread4~ [13]) и проводится через vcf~ для установки центральной частоты. Другой способ использования объекта vcf~ для субтрактивного синтеза демонстрируется в примере H05.filter.floyd.pd.

8.5.3 Детекторы огибающей

Пример H06.envelope.follower.pd показывает простую и понятную реализацию детектора огибающей, описанного в §8.4.2. Интересное приложение считывания огибающей показано в примере H07.measure.spectrum.pd (рисунок 8.30, часть a). В нём известный сэмпл колокола зацикливается в качестве тестового звука. Вместо того чтобы получить полную среднеквадратичную мощность колокола, мы хотим оценить частоту и мощность каждой из его гармоник. Чтобы сделать это, мы перемещаем частоту полосового фильтра вверх и вниз, слушая результат и/или

[13] видимо, на момент написания книги еще не был доступен объект mtof~ , присутствующий в современных версиях Pd (*прим. перев.*)

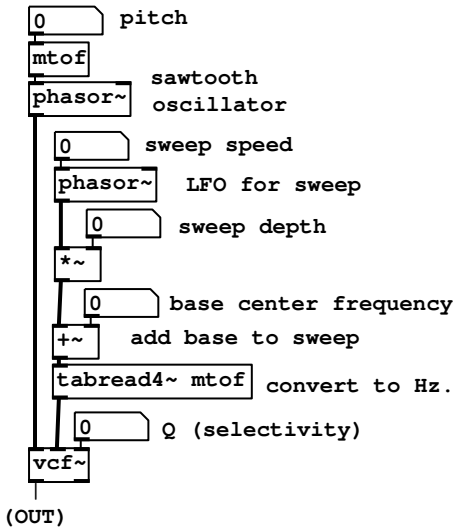


Рис. 8.29: Полосовой фильтр vcf~ с центральной частотой, управляемой аудиосигналом (для сравнения с bp~, который только получает управляющие сообщения для установки его центральной частоты).

наблюдая выходную мощность фильтра с использованием детектора огибающей. (Мы используем два полосовых фильтра последовательно для лучшей изоляции гармоник; это не самая лучшая практика конструкции фильтра, но может использоваться в данном случае). Когда фильтр настроен на гармонику, детектор огибающей выдаёт её силу.

Пример H08.heterodyning.pd (часть b рисунка) показывает альтернативный способ нахождения силы гармоник входного сигнала; его преимущество заключается в том, что он сообщает как о фазе, так и о силе. Во-первых, мы совершаем модуляцию требуемой гармоники вниз, на нулевую частоту. Мы используем комплекснозначную синусоиду как модулятор, так что получается только боковая полоса для каждой входной компоненты. Измеряемая частота есть просто частота, модулированная на DC; прочие сдвигаются на какие-то другие частоты. Далее мы фильтруем получившийся комплексный сигнал фильтром низких частот. (Мы можем использовать действительнозначный ФНЧ для действительной и мнимой частей отдельно.) Это в основном удаляет все гармоники за исключением DC, которая нам нужна. Этот метод является основой анализа Фурье, разбираемого в Главе 9.

8.5.4 Однополосная модуляция

Как описано в §8.4.3, пара всепропускающих фильтров может быть сконструирована так, чтобы давать разность фаз приблизительно равную $\pi/2$ для положительных частот и $-\pi/2$ для отрицательных. Конструкция этих пар остаётся за рамками нашего обсуждения (см., например [Reg93]), тем не менее Pd предоставляет абстракцию hilbert~, которая осуществляет это. Пример H09.ssb.modulation.pd,

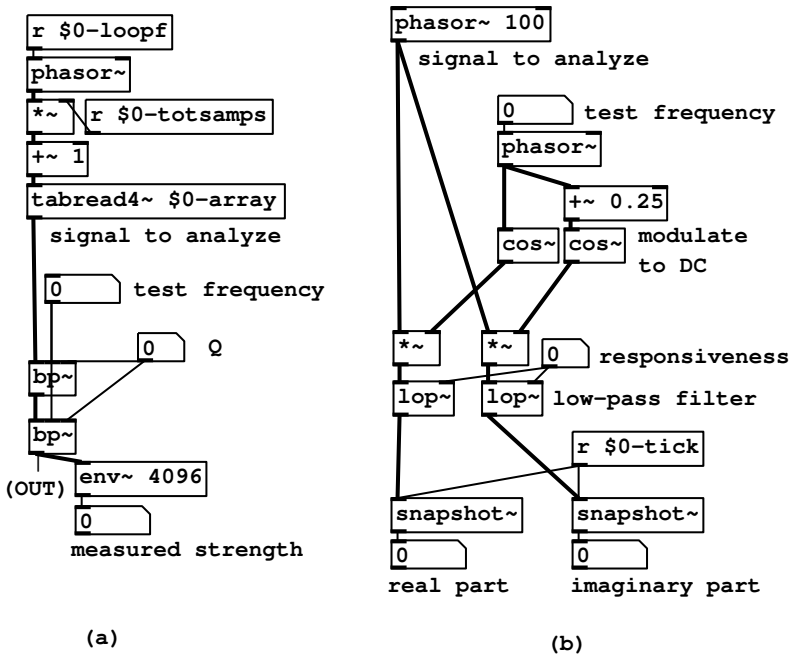


Рис. 8.30: Анализ спектра звука: (a) полосовая фильтрация семплированного звука колокола со детекцией огибающей результата; (b) сдвиг частоты гармоники на DC и считывание её действительной и мнимой части.

показанный на рис. 8.31, демонстрирует как использовать абстракцию hilbert~ для того, чтобы сделать модуляцию на одну боковую полосу. Преобразование Гильберта относится к аналоговой эре [Str95, pp.129-132].

Два выхода hilbert~, рассматриваемые как действительная и мнимая части комплекснозначного сигнала, умножаются на комплексную синусоиду (справа на рисунке), и выводится действительная часть. Компоненты получающегося сигнала представляют из себя компоненты входа, сдвинутые на (положительную или отрицательную) частоту, определённую в числовом боксе.

8.5.5 Использование элементарных фильтров напрямую: полочный и пиковый фильтры

Ни один конечный набор встроенных фильтров не мог бы удовлетворить все возможные потребности и, таким образом, Pd предоставляет элементарные фильтры из разделов §8.2.1–8.2.3 в “сыром” виде, чтобы пользователь мог задавать коэффициенты фильтров в явном виде. В этом параграфе мы будем описывать патчи, которые реализуют полочный и пиковый фильтры §8.3.3 и §8.3.5 из элементарных фильтров. Сначала мы представим шесть объектов Pd, которые реализуют элементарные фильтры:

rzero~, rzero_rev~, rpole~: элементарные фильтры с вещественными коэф-

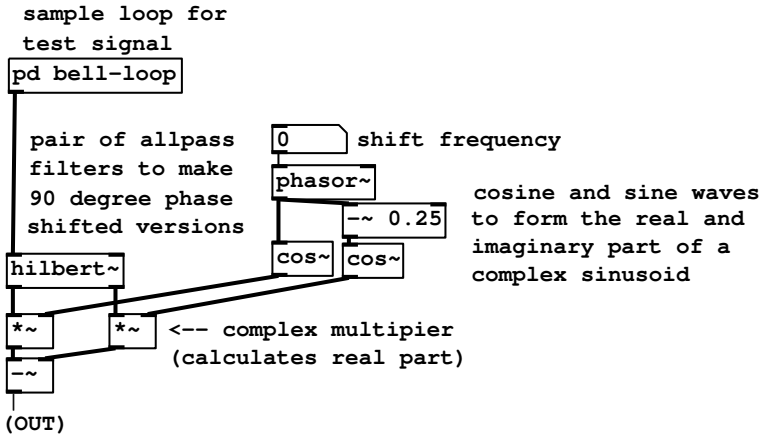


Рис. 8.31: Использование всепропускающего фильтра для создания сдвига частоты

фициентами, оперирующие с вещественными сигналами. Эти три объекта осуществляют фильтры без рециркуляции первого и второго типов и фильтр с рециркуляцией. Они все имеют справа один вход для коэффициента, задающего локализацию нуля или полюса. Этот вход (так же как и левый вход для фильтруемого сигнала) получает аудиосигнал. Проверка устойчивости не осуществляется.

czero~, rzero~, rpole~: элементарные фильтры с комплексными коэффициентами, оперирующие комплекснозначными сигналами, соответствующие объектам для вещественных сигналов выше. Вместо двух входов и одного выхода, каждый из этих фильтров имеет четыре входа (действительная и мнимая части фильтруемого сигнала, а также действительная и мнимая части коэффициента) и два выхода для комплекснозначного выхода.

Патчи примеров используют пару абстракций для построения частотного и фазового откликов фильтров, как описано в примере H10.measurement.pd. Пример H11.shelving.pd (рис. 8.32, часть a) показывает создание полочного фильтра. Элементарный фильтр без рециркуляции (rzero~) и элементарный фильтр с рециркуляцией (rpole~) соединены последовательно. Как и при анализе в разделе §8.3.9, можно предложить, чтобы объект rzero~ стоял первым.

Пример H12.peaking.pd (часть (b) рисунка) реализует пиковый фильтр. Здесь полюс и ноль повёрнуты на угол ω для управления центральной частотой фильтра. Ширина полосы и коэффициент усиления центральной частоты равны переходной частоте и коэффициенту усиления DC соответствующего полочного фильтра.

Пример H13.butterworth.pd демонстрирует полочный фильтр Баттерворта с тремя полюсами и тремя нулями. Сам фильтр представляет собой абстракцию butterworth3~ для удобного повторного использования.

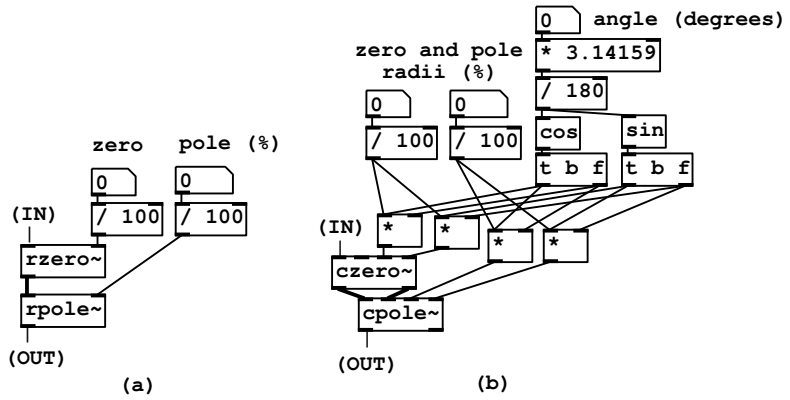


Рис. 8.32: Построение фильтров из элементарных низкоуровневых: (a) полочный; (b) пиковый.

8.5.6 Создание и использование всепропускающих фильтров

Пример H14.all.pass.pd (рис. 8.33, часть a) показывает как создать всепропускающий фильтр из фильтра без рециркуляции второго вида (rzero_rev~) и фильтра с рециркуляцией (rpole~). Коэффициент, изменясь в диапазоне от -1 до 1, регулируется в процентах.

Пример H15.phaser.pd (часть b рисунка) показывает как использовать четыре всепропускающих фильтра для создания классического эффекта фэйзера[14]. Фэйзер работает, суммируя входной сигнал с его версией другой фазы, что создаёт интерференционный эффект. Величина изменения фазы варьируется по времени с помощью изменения (общего) коэффициента всепропускающих фильтров. Общий эффект представляет из себя нечто близкое к фленджеру (варьирующийся по времени гребенчатый фильтр), но фазер не устанавливает высоту, как это делает гребенчатый фильтр.

Упражнения

1. Элементарный фильтр с рециркуляцией имеет полюс в $i/2$. На какой угловой частоте его коэффициент усиления будет наивысшим, и чему он равен? На какой угловой частоте коэффициент усиления минимальный, и чему он равен?

2. Полочный фильтр имеет полюс в 0.9 и ноль в 0.8. Каковы тогда коэффициенты усиления на DC и частоте Найквиста и приблизительная переходная частота?

3. Предположим, что комплексный фильтр с рециркуляцией имеет полюс в P. Предположим далее, что вы хотите объединить его действительный и мни-

[14] Фэйзер (англ. phaser), также часто называемый фазовым вибрато — звуковой эффект, который достигается фильтрацией звукового сигнала с созданием серии максимумов и минимумов в его спектре. Положение этих максимумов и минимумов варьируется на протяжении звучания, что создаёт специфический круговой эффект. (*прим. перев.*)

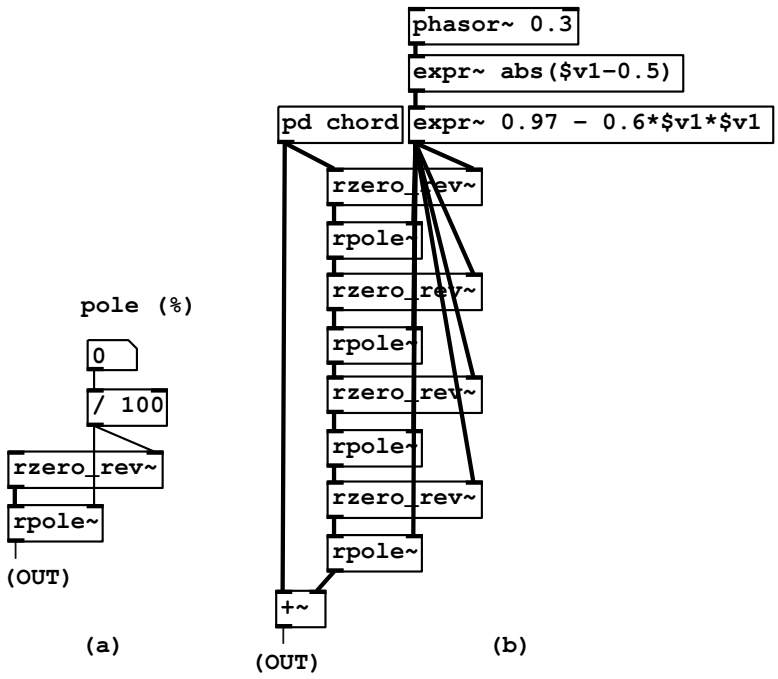


Рис. 8.33: Всепропускающие фильтры: (a) создание всепропускающего фильтра из элементарных фильтров; (b) использование четырёх всепропускающих фильтров для построения фазера.

мый выход, чтобы получить единый вещественный сигнал, эквивалентный двухполюсному фильтру с полюсами в P и $\overline{P}$. С каким весом вы бы взяли два выхода?

4. Предположим, что вы хотите сконструировать пиковый фильтр с коэффициентом усиления 2 на 1000 герц и шириной полосы 200 герц (со частотой семплирования 44100 герц). Где, приблизительно, вы бы установили полюс и ноль?

5. В той же ситуации, где бы вы установили (верхний) полюс и ноль для удаления синусоиды на 1000 герц целиком, при уменьшении на 3 децибела на частоте 1001 герц?

6. Комплексный фильтр с одним полюсом возбуждается импульсом для создания тона на 1000 герц, который спадает на 10 децибел за одну секунду (при частоте семплирования 44100 герц). Где бы вы поместили полюс? Какова величина “q”?

Глава 9

Анализ и синтез Фурье

Среди примеров применения фильтров, обсуждаемых в Главе 8, мы видели, как использовать преобразование частоты сигналов в комбинации с низкочастотным фильтром для нахождения их амплитуды и фазы синусоидальной компоненты сигнала (стр. 252). В этой главе мы усовершенствуем этот метод, который будет называться *анализом Фурье*. В простейшей форме анализ Фурье берёт любой входящий периодический сигнал (с периодом N) и выводит комплекснозначные амплитуды его N возможных синусоидальных компонент. Эти N комплексных амплитуд могут быть теоретически использованы для точной реконструкции первоначального сигнала. Эта реконструкция называется *синтезом Фурье*.

В этой главе мы начнём с рассмотрения теории анализа Фурье и ресинтезе периодических семплированных сигналов. Затем мы покажем как применить те же методы к произвольным сигналам, периодическим или нет. Наконец, мы разработаем некоторые стандартные приложения, такие как фазовый вокодер.

9.1 Анализ Фурье периодических сигналов

Предположим, что $X[n]$ есть комплекснозначный сигнал, который повторяется каждые N семплов. (Мы продолжаем использование комплекснозначных сигналов вместо действительнозначных для упрощения математических расчётов.) Поскольку период равен N, значения $X[n]$ для $n = 0, \ldots, N-1$ определяют $X[n]$ для всех целых значений n.

Предположим далее, что $X[n]$ может быть записан в виде суммы комплексных синусоид частот 0, $2\pi/N, 4\pi/N, \ldots, 2(N-1)\pi/N$. Это — начинающиеся с нулевой гармоники сигнала периода N. Мы остановились на N-ом члене, поскольку следующий имел бы частоту 2π, эквивалентную частоте 0, которая уже попала в список.

Для данного значения X мы хотим найти комплексные амплитуды гармоник. Предположим, что нам нужна k-ая гармоника, где $0 \leqslant k < N$. Частота этой

гармоники равна $2\pi k/N$. Мы можем найти её комплексную амплитуду, модулируя X вниз на $2\pi k/N$ радиан/семпл по частоте, так что происходит модуляция k-ой гармоники на частоту ноль. Затем мы пропускаем сигнал через ФНЧ с такой частотой среза, что ничего кроме гармоники с нулевой частотой не остаётся. По сути, мы можем сделать это путем усреднения по очень большому количеству семплов; но поскольку сигнал повторяется каждые N семплов, это громадное усреднение даёт такой же результат, как усреднение по N семплам. Короче говоря, для измерения синусоидальной компоненты периодического сигнала, смодулируйте его на DC, а затем усредните за один период.

Пусть $\omega = 2\pi/N$ есть основная частота, соответствующая периоду N, и U — комплексное число с единичным модулем и аргументом ω:

$$U = \cos(\omega) + i\sin(\omega).$$

Пусть k-ая гармоника сигнала $X[n]$ имеет вид:

$$P_k[n] = A_k\left[U^k\right]^n,$$

где A_k есть комплексная амплитуда гармоники, а частота гармоники равна:

$$\angle(U^k) = k\angle(U) = k\omega.$$

В настоящий момент мы предполагаем, что сигнал $X[n]$ может реально быть записан в виде суммы n гармоник или, другими словами:

$$X[n] = A_0\left[U^0\right]^n + A_1\left[U^1\right]^n + \cdots + A_{N-1}\left[U^{N-1}\right]^n.$$

С помощью фильтрации аргумента методом выше, мы надеемся получить возможность измерить каждый A_k, умножая на синусоиду частоты $-k\omega$ и усредняя по периоду:

$$A_k = \frac{1}{N}\left(\left[U^{-k}\right]^0 X[0] + \left[U^{-k}\right]^1 X[1] + \cdots + \left[U^{-k}\right]^{N-1} X[N-1]\right).$$

Это настолько полезная формула, что она получила специальное обозначение. *Преобразование Фурье* сигнала $X[n]$ по N семплам определяется как:

$$\mathcal{FT}\left\{X[n]\right\}(k) = V^0 X[0] + V^1 X[1] + \cdots + V^{N-1} X[N-1],$$

где $V = U^{-k}$. Преобразование Фурье является функцией переменной k, равной произведению N и амплитуды k-ой входной гармоники. До сих пор k принимало целые значения, но формула имеет смысл для любых значений k, если мы определим V в более общем виде, как:

$$V = \cos(-k\omega) + i\sin(-k\omega),$$

где, как перед этим, $\omega = 2\pi/N$ есть (угловая) основная частота, ассоциированная с периодом N.

9.1.1 Периодичность преобразования Фурье

Если $X[n]$ такой же сигнал, как указано выше, который повторяется каждые N семплов, то преобразование Фурье $X[n]$ также повторяет себя каждые N единиц частоты, то есть

$$\mathcal{FT}\left\{X[n]\right\}(k+N) = \mathcal{FT}\left\{X[n]\right\}(k)$$

для всех вещественных значений k. Это немедленно следует из определения преобразования Фурье, поскольку множитель

$$V = \cos(-k\omega) + i\sin(-k\omega)$$

не меняется, когда мы добавляем N (или любое кратное N) к k.

9.1.2 Преобразование Фурье как аддитивный синтез

Теперь рассмотрим произвольный сигнал $X[n]$, который повторяется каждые N семплов. (Ранее мы предполагали, что $X[n]$ может быть получен как сумма синусоид, и мы ещё не выяснили, может ли любой периодический сигнал $X[n]$ быть получен таким способом.) Пусть $Y[k]$ обозначает преобразование Фурье X для $k = 0, \ldots, N-1$:

$$Y[k] = \mathcal{FT}\left\{X[n]\right\}(k)$$

$$= \left[U^{-k}\right]^0 X[0] + \left[U^{-k}\right]^1 X[1] + \cdots + \left[U^{-k}\right]^{N-1} X[N-1]$$

$$= \left[U^0\right]^k X[0] + \left[U^{-1}\right]^k X[1] + \cdots + \left[U^{-(N-1)}\right]^k X[N-1].$$

Во втором варианте мы переставили экспоненты, чтобы показать, что $Y[k]$ является суммой комплексных синусоид с комплексными амплитудами $X[m]$ и частотами — $m\omega$ для $m = 0, \ldots, N-1$. Другими словами, $Y[k]$ может рассматриваться на правах ряда Фурье, чья m-ая компонента имеет силу $X[-m]$. (Выражение $X[-m]$ имеет смысл, поскольку X является периодическим сигналом). Мы можем также выразить амплитуду гармоник $Y[k]$ в терминах их собственного преобразования Фурье. Их приравнивание даёт:

$$\frac{1}{N}\mathcal{FT}\left\{Y[k]\right\}(m) = X[-m].$$

Это, в свою очередь, имеет тот смысл, что $X[-m]$ может быть получен с помощью суммирования синусоид с амплитудами $Y[k]/N$. Положив $n = -m$ получаем:

$$X[n] = \tfrac{1}{N}\mathcal{FT}\left\{Y[k]\right\}(-n)$$

$$= \left[U^0\right]^n Y[0] + \left[U^1\right]^n Y[1] + \cdots + \left[U^{N-1}\right]^n Y[N-1].$$

Это показывает, что любой периодический сигнал $X[n]$ действительно может быть получен как сумма синусоид. Кроме того, формула явно показывает, как

восстановить $X[n]$ из его преобразования Фурье $Y[k]$, если мы знаем его значения для целых $k = 0, \ldots, N-1$.

9.2 Свойства преобразований Фурье

В этом параграфе мы рассмотрим, что происходит, когда мы берём преобразование Фурье (комплексной) синусоиды. Простейшая такая синусоида есть "DC", специальная синусоида с частотой ноль. После получения её преобразования Фурье, мы обнаружим некоторые свойства преобразования Фурье, которые позволят нам применить результат к любой другой синусоиде.

9.2.1 Преобразование Фурье DC

Положим $X[n] = 1$ для всех n (который повторяется с любым требуемым целым периодом $N > 1$). Исходя из предыдущего обсуждения, мы ожидаем обнаружить, что

$$\mathcal{FT}\left\{X[n]\right\}(k) = \begin{cases} N, & \text{если } k = 0 \\ 0, & \text{когда } k = 1, \ldots, N-1 \end{cases}$$

Однако нам будет часто требоваться знать результат для нецелых значений k, и для этого нет ничего лучше, чем вычислить значение напрямую:

$$\mathcal{FT}\left\{X[n]\right\}(k) = V^0 X[0] + V^1 X[1] + \cdots + V^{N-1} X[N-1],$$

где V, как и раньше, комплексное число с единичным модулем и аргументом $-k\omega$. Это — геометрическая прогрессия; и если $V \neq 1$, мы получаем:

$$\mathcal{FT}\left\{X[n]\right\}(k) = \frac{V^N - 1}{V - 1}.$$

Теперь мы симметризуем числитель и знаменатель тем же способом, как ранее делали в §7.3. Для того чтобы проделать это, положим:

$$\xi = \cos(\pi k/N) - i\sin(\pi k/N)$$

так что $\xi^2 = V$. Тогда, умножая на подходящую степень ξ числитель и знаменатель, получаем:

$$\mathcal{FT}\left\{X[n]\right\}(k) = \xi^{N-1}\frac{\xi^N - \xi^{-N}}{\xi - \xi^{-1}}.$$

Теперь нетрудно упростить числитель:

$$\xi^N - \xi^{-N} = (\cos(\pi k) - i\sin(\pi k)) - (\cos(\pi k) + i\sin(\pi k)) = -2i\sin(\pi k)$$

и аналогично для знаменателя:

$$\mathcal{FT}\left\{X[n]\right\}(k) = \Big(\cos(\pi k(N-1)/N) - i\sin(\pi k(N-1)/N)\Big)\frac{\sin(\pi k)}{\sin(\pi k/N)}$$

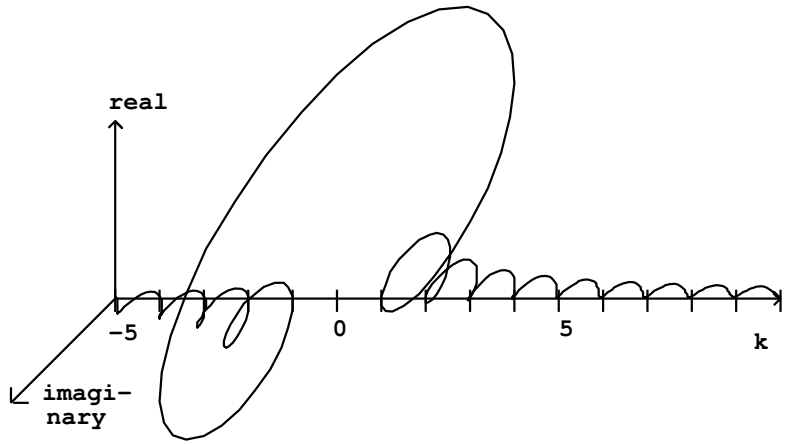


Рис. 9.1: Преобразование Фурье сигнала, состоящего из всех единиц. Здесь $N = 100$ и значения показаны для k в диапазоне от -5 до 10. Результат является комплексно-значным и показан как проекция с действительной осью, соответствующей плоскости рисунка, и мнимой осью, перпендикулярной к ней.

Независимо от того, $V = 1$ или нет, мы получаем

$$\mathcal{FT}\left\{X[n]\right\}(k) = \Big(\cos(\pi k(N-1)/N) - i\sin(\pi k(N-1)/N)\Big)\, D_N(k)$$

где $D_N(k)$, известное как *ядро Дирихле*, определяется как

$$D_N(k) = \begin{cases} N, & \text{если } k = 0 \\ \frac{\sin(\pi k)}{\sin(\pi k/N),} & \text{если } k \neq 0,\ -N < k < N \end{cases}$$

Рисунок 9.1 показывает преобразование Фурье $X[n] = 1$ для $N = 100$. Преобразование повторяется каждые 100 семплов, с пиком в $k = 0$, другим в $k = 100$ и так далее. На рисунке предпринята попытка изобразить поведение как модуля, так и фазы с использованием 3-мерного графика, спроектированного на плоскость. Фазовый член

$$\cos(\pi k(N-1)/N) - i\sin(\pi k(N-1)/N)$$

действует как закручивание значений $\mathcal{FT}\left\{X[n]\right\}(k)$ вокруг оси k с периодом, приблизительно равным двум. Ядро Дирихле $D_N(k)$, показанное на рис. 9.2, управляет модулем $\mathcal{FT}\left\{X[n]\right\}(k)$. Оно имеет пик шириной в две единицы вокруг $k = 0$. Он окружён боковыми лепестками единичной ширины, меняющими знак и постепенно убывающими по модулю, когда k растёт или убывает от нуля. Фазовый член даёт закручивание около π радиан каждый раз, когда ядро Дирихле меняет знак, так что произведение обоих членов остаётся приблизительно в одной и той же комплексной полуплоскости для $k > 1$ (и в противоположной полуплоскости для $k < -1$). Фаза поворачивается приблизительно на 2π радиан на пике от $k = -1$ до $k = 1$.

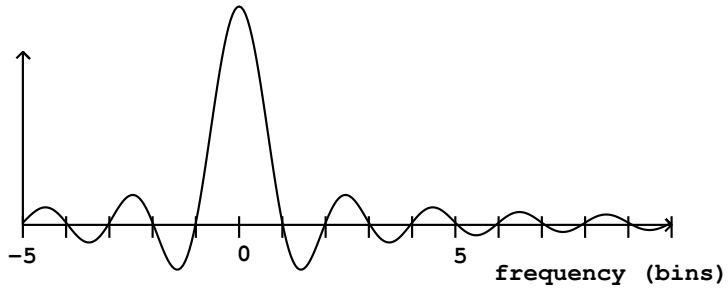


Рис. 9.2: Ядро Дирихле для $N = 100$

9.2.2 Сдвиги и изменения фазы

Параграф §7.2 показал, как сдвиг сигнала по времени меняет фазы его синусоидальных компонент, а §8.4.3 показал, как умножение сигнала на комплексную синусоиду сдвигает составляющие его частоты. Эти два эффекта имеют соответствующие аналоги для преобразования Фурье.

Во-первых, мы рассмотрим временной сдвиг. Если $X[n]$, как обычно, комплекснозначный сигнал, который повторяется каждые N семплов, пусть $Y[n]$ будет $X[n]$, задержанный на d семплов

$$Y[n] = X[n-d],$$

который также повторяется каждые N семплов, как X. Мы можем трансформировать преобразование Фурье $Y[n]$ следующим образом:

$$\begin{aligned}\mathcal{FT}\left\{Y[n]\right\}(k) &= V^0Y[0] + V^1Y[1] + \cdots + V^{N-1}Y[N-1] \\ &= V^0X[-d] + V^1X[-d+1] + \cdots + V^{N-1}X[-d+N-1] \\ &= V^dX[0] + V^{d+1}X[1] + \cdots + V^{d+N-1}X[N-1] \\ &= V^d\left(V^0X[0] + V^1X[1] + \cdots + V^{N-1}X[N-1]\right) \\ &= V^d\mathcal{FT}\left\{X[n]\right\}(k).\end{aligned}$$

(Третья строчка есть всего лишь вторая, в которой члены просуммированы в другом порядке). Таким образом мы получаем *формулу временного сдвига для преобразований фурье*:

$$\mathcal{FT}\left\{X[n-d]\right\}(k) = \Big(\cos(-dk\omega) + i\sin(-dk\omega)\Big)\,\mathcal{FT}\left\{X[n]\right\}(k) \tag{9.1}$$

Преобразование Фурье $X[n-d]$ равно произведению фазового множителя и преобразования Фурье $X[n]$. Фаза меняется на $-idk\omega$, что является линейной функцией k.

Теперь предположим, что мы изменяем наш первоначальный сигнал $X[n]$, умножая его на комплексную экспоненту Z^n с угловой частотой α:

$$Y[n] = Z^nX[n]$$

$$Z = \cos(\alpha) + i\sin(\alpha).$$

Его преобразование Фурье есть:

$$\mathcal{FT}\left\{Y[n]\right\}(k) = V^0 Y[0] + V^1 Y[1] + \cdots + V^{N-1} Y[N-1]$$

$$= V^0 X[0] + V^1 Z X[1] + \cdots + V^{N-1} Z^{N-1} X[N-1]$$

$$= (VZ)^0 X[0] + (VZ)^1 X[1] + \cdots + (VZ)^{N-1} X[N-1]$$

$$= \mathcal{FT}\left\{X[n]\right\}(k - \frac{\alpha}{\omega}).$$

Таким образом, мы получаем *формулу фазового сдвига для преобразований фурье*:

$$\mathcal{FT}\left\{(\cos(\alpha) + i\sin(\alpha))X[n]\right\}(k) = \mathcal{FT}\left\{X[n]\right\}(k - \frac{\alpha N}{2\pi}). \tag{9.2}$$

9.2.3 Преобразование Фурье синусоиды

Мы можем использовать формулу фазового сдвига, приведённую выше, для нахождения преобразования Фурье любой комплексной синусоиды Z^n с угловой частотой α, просто устанавливая $X[n] = 1$ в формуле и используя преобразование Фурье для DC:

$$\mathcal{FT}\left\{Z^n\right\}(k) = \mathcal{FT}\left\{1\right\}(k - \frac{\alpha}{\omega})$$

$$= \left[\cos(\Phi(k)) + i\sin(\Phi(k))\right] D_N(k - \frac{\alpha}{\omega}),$$

где D_N есть ядро Дирихле, и Φ — “некрасивый” фазовый коэффициент:

$$\Phi(k) = -\pi \cdot (k - \frac{\alpha}{\omega}) \cdot (N-1)/N.$$

Если частота синусоиды α является целым кратным основной частоты ω, ядро Дирихле сдвигается налево или направо на целое число. В этом случае пересечение нулевого уровня ядром Дирихле соответствует целым значениям k, так что имеется только одна ненулевая гармоника. Это изображено на рис. 9.3 (часть a).

Часть (b) показывает результат, когда частота α попадает на половину расстояния между двумя целыми значениями. Гармоники имеют амплитуды, спадающие, примерно, как $1/k$ в обоих направлениях, измеряемых от действующей частоты α. В этом случае энергия должна распределяться по множеству гармоник при том, что мы начинали с единственной синусоиды, что может показаться удивительным на первый взгляд. Однако как показано на рис. 9.4, сигнал повторяется с периодом N, что не соответствует частоте синусоиды. В результате имеется разрыв в начале каждого периода, и энергия разбросана по широкому диапазону частот.

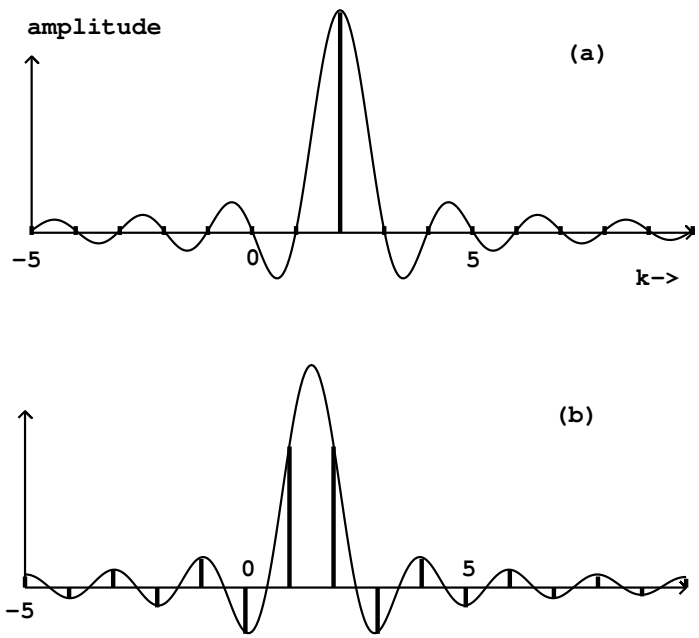


Рис. 9.3: Преобразования Фурье комплексных синусоид с $N = 100$: (a) с частотой 2ω; (b) с частотой 1.5ω. (Эффект вращения фазы не показан.)

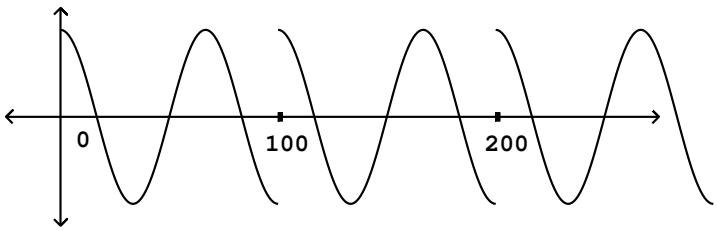


Рис. 9.4: Комплексная синусоида с частотой $\alpha = 1.5\omega = 3\pi/N$, вынужденная повторяться каждые N семплов. (N произвольно установлено на 100; показана только действительная часть.)

9.3 Анализ Фурье непериодических сигналов

Большинство сигналов не являются периодическими и даже периодические сигналы могут иметь неизвестный период. Таким образом, мы должны быть готовы к тому, чтобы производить анализ Фурье сигналов без удобного предположения о том, что анализируемый сигнал повторяется с фиксированным периодом N. Конечно, мы можем просто взять N семплов сигнала и продолжить его периодическим образом; это в основном то, что мы делали в предыдущем параграфе, в котором чистая синусоида давала нам сложное преобразование Фурье (рис. 9.3, часть b).

Однако было бы лучше получить в результате отклик на чистую синусоиду, лучше локализованный вокруг соответствующего значения k. Мы можем осуществить это с использованием техники огибающей, впервые введённой на рис. 2.7. Применение этого метода к анализу Фурье не только улучшит наш анализ, но также прольёт новый свет на циклический семплер огибающей Главы 2.

Для данного сигнала $X[n]$, периодического или нет, определённого на точках от 0 до $N-1$, метод состоит в построении огибающей сигнала, перед тем как делать для него анализ Фурье. Форма огибающей известна как оконная функция. Для данной функции $w[n]$, *оконное преобразование Фурье* есть

$$\mathcal{FT}\left\{w[n]X[n]\right\}(k).$$

Сколько бумаги было исписано по поводу конструкции подходящих функций окна для частных ситуаций! Но здесь мы будем рассматривать простейшую функцию, называемую функцией окна Ханна (это имя иногда искажают в “окно Хеннинга” в кругах DSP-шников). Окно Ханна определяется как

$$w[n] = \frac{1}{2} - \frac{1}{2}\cos(2\pi n/N).$$

Нетрудно проанализировать эффект от умножения сигнала на окно Ханна, перед тем как сделать преобразование Фурье, поскольку окно Ханна может быть записано как сумма трёх комплексных экспонент:

$$w[n] = \frac{1}{2} - \frac{1}{4}U^n - \frac{1}{4}U^{-n},$$

где, как и раньше, U — комплексное число с единичным модулем и аргументом $2\pi/N$. Теперь мы можем вычислить оконное преобразование Фурье синусоиды Z^n с угловой частотой α, как и раньше. Фазы на выходе имеют довольно запутанный вид, и мы заменим их на упрощённые аппроксимации:

$$\begin{aligned}
&\mathcal{FT}\left\{w[n]Z^n\right\}(k) \\
&= \mathcal{FT}\left\{\frac{1}{2}Z^n - \frac{1}{4}(UZ)^n - \frac{1}{4}(U^{-1}Z)^n\right\}(k) \\
&\approx \left[\cos(\Phi(k)) + i\sin(\Phi(k))\right] M(k - \frac{\alpha}{\omega}),
\end{aligned}$$

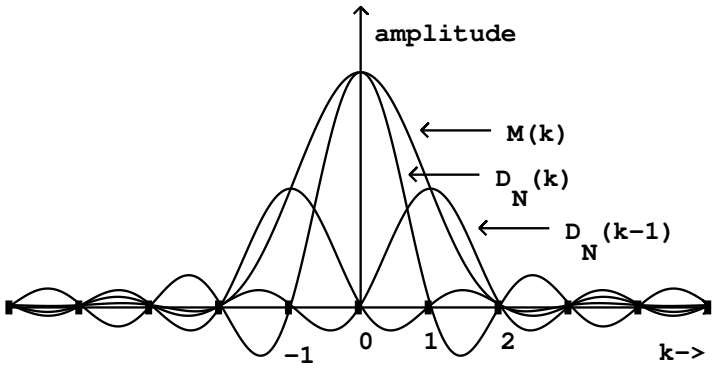


Рис. 9.5: Модуль $M(k)$ преобразования Фурье функции окна Ханна. Это сумма трёх (сдвинутых и масштабированных) копий ядра Дирихле D_N при $N = 100$.

где (приближённый) фазовый член есть

$$\Phi(k) = -\pi \cdot (k - \frac{\alpha}{\omega})$$

и функция, задающая модуль:

$$M(k) = \left[\frac{1}{2}D_N(k) + \frac{1}{4}D_N(k+1) + \frac{1}{4}D_N(k-1)\right].$$

Функция, задающая модуль $M(k)$, построена графически на рис. 9.5. Три компоненты ядра Дирихле также показаны по отдельности.

Основной лепесток $M(k)$ имеет ширину в четыре гармоники и в два раза шире основного лепестка ядра Дирихле. Боковые лепестки, с другой стороны, имеют гораздо меньшие величины. Каждый боковой лепесток $M(k)$ является суммой трёх боковых лепестков $D_N(k)$, один, уменьшенный на $1/2$, и другие, противоположные по знаку, уменьшенные на $1/4$. Они не компенсируются полностью, но очень близко к этому.

Боковые лепестки достигают своих максимальных значений в районе срединных точек, и мы можем оценить их амплитуды там с использованием аппроксимации:

$$D_N(k) \approx \frac{N \sin(\pi k)}{\pi k}.$$

Установка $k = 3/5, 5/2, \ldots$ даёт амплитуды лепестков, относящиеся к пиковой высоте N так, что

$$\frac{2}{3\pi} \approx -13\text{дб}, \ \frac{2}{5\pi} \approx -18\text{дб}, \ \frac{2}{7\pi} \approx -21\text{дб}, \ \frac{2}{9\pi} \approx -23\text{дб}, \ldots$$

Боковые лепестки спадают последовательно всё медленне, так что десятый уменьшается только до ≈ -30 дб, а 32-ой до ≈ -40 дб. С другой стороны, боковые

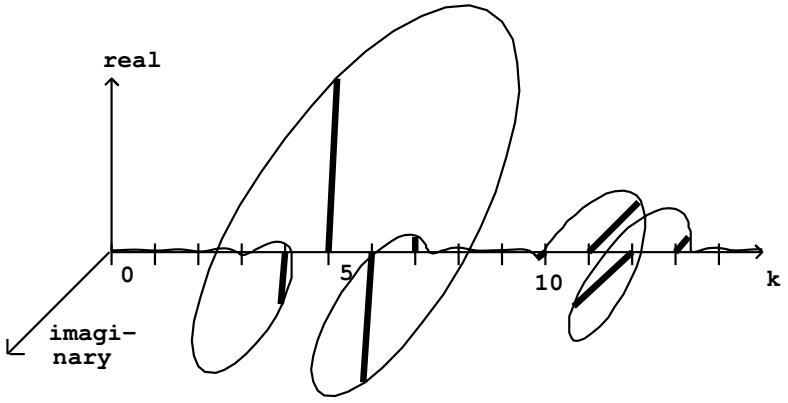


Рис. 9.6: Преобразование Фурье с окном Ханна сигнала с двумя синусоидальными компонентами, с частотами, равными произведениям 5.3 и 10.6 на основную частоту, с различными комплексными амплитудами.

лепестки окна Ханна уменьшаются на

$$\frac{2}{5\pi} - \frac{1}{2}[\frac{2}{3\pi} + \frac{2}{7\pi}] \approx -32.30\text{дб}$$

и -42, -49, -54 и -59 дб для следующих четырёх боковых лепестков.

Это показывает, что применение окна Ханна перед преобразованием Фурье позволит нам лучше изолировать синусоидальные компоненты. Если сигнал имеет много синусоидальных компонентов, то боковые лепестки, порожденные каждым из них, будут мешать главному лепестку среди всех остальных. Уменьшение амплитуды боковых лепестков уменьшает эту интерференцию.

Рисунок 9.6 показывает анализ Фурье с использованием окна Ханна сигнала с двумя синусоидальными компонентами. Эти компоненты отделены друг от друга примерно на пять основных частот ω, и для каждой из них мы ясно видим форму преобразования Фурье окна Ханна. Четыре точки анализа Фурье лежат внутри основного лепестка $M(k)$, соответствующего каждой синусоиде. Амплитуда и фаза индивидуальных синусоид отражаются в амплитудах и фазах (четырёхточечных в ширину) пиков. Четыре точки внутри пика, которые попадают на целые значения k, последовательно отличаются по фазе на половину цикла.

Для полного извлечения гармоник сигнала, мы должны выбрать анализируемый размер N достаточно большим, так чтобы $\omega = 2\pi/N$ не превышало четверти частотной разницы между близлежащими гармониками. Для периодического сигнала, например, гармоники разделены на основную частоту. Для анализа, полностью извлекающего гармоники, анализируемый период N должен быть равен как минимум четырём периодам сигнала.

В некоторых случаях хорошо работает, если разрешенить пикам перекрываться, до тех пор, пока центр каждого пика изолирован от других пиков; в этом случае, правило четырёх периодов может быть ослаблено до трёх или даже меньшего количества.

9.4 Анализ Фурье и реконструкция аудиосигналов

Анализ Фурье может иногда использоваться для извлечения синусоидальных компонент в аудиосигнале. Даже когда это нельзя сделать, можно разделить сигнал на частотные регионы, в том смысле, что для каждого k, k-ая точка преобразования Фурье подвергается воздействию только компонент, близких к номинальной частоте $k\omega$. Это предоставляет много интересных операций, которые мы можем проделать над сигналом с помощью преобразования Фурье, преобразования его результата и реконструкции нового трансформированного сигнала из модифицированного преобразования.

Рисунок 9.7 показывает, как выполнить анализ Фурье, модификацию и реконструкцию аудиосигнала. Первый шаг заключается в разделении сигнала на *окна*, которые являются отрезками сигнала по N семплов каждое, обыкновенно с некоторым перекрытием. Затем каждое окно получает форму с помощью умножения на функцию окна (например, Ханна). Потом вычисляется преобразование Фурье для N точек $k = 0, 1, \ldots, N-1$. (Иногда требуется вычислить преобразование Фурье для большего числа точек, чем это, но здесь будет достаточно этих N точек.)

Анализ Фурье даёт нам двумерный массив комплексных чисел. Пусть H обозначает *размер скачка* (“hop size”), число семплов каждого окна, которые предшествуют следующему окну. Тогда, для каждого $m = \ldots, 0, 1, \ldots$, m-ое окно состоит из N точек, начинающихся с точки mH. Точка n окна m есть $mH + n$. Оконное преобразование Фурье есть, таким образом,

$$S[m,k] = \mathcal{FT}\{w(n)X[n-mH]\}(k).$$

Это одновременно и функция времени (m в единицах H семплов), и частоты (k, как кратное основной частоты ω). Фиксирование номера окна m и рассматривание оконного преобразования Фурье как функции k

$$S[k] = S[m,k]$$

даёт меру моментального спектра сигнала $X[n]$. С другой стороны, фиксируя частоту k, мы можем ввести в рассмотрение k-ый канал N-канального сигнала:

$$C[m] = S[m,k].$$

С этой точки зрения, оконное преобразование Фурье разделяет первоначальный сигнал $X[n]$ на N узких частотных регионов, называемых *полосами*.

После вычисления оконного преобразования Фурье мы применяем любое требуемое изменение. На приведенном рисунке модификация состоит просто в замене верхней половины спектра на ноль, что даёт высокоизбирательный низкочастотный фильтр. (Две другие возможные модификации, узкополосное компандирование и вокодер, описываются в следующих параграфах.)

Наконец, мы восстанавливаем выходной сигнал. Для того чтобы сделать это, мы применяем обратное преобразование Фурье (обозначенное “iFT” на рисунке).

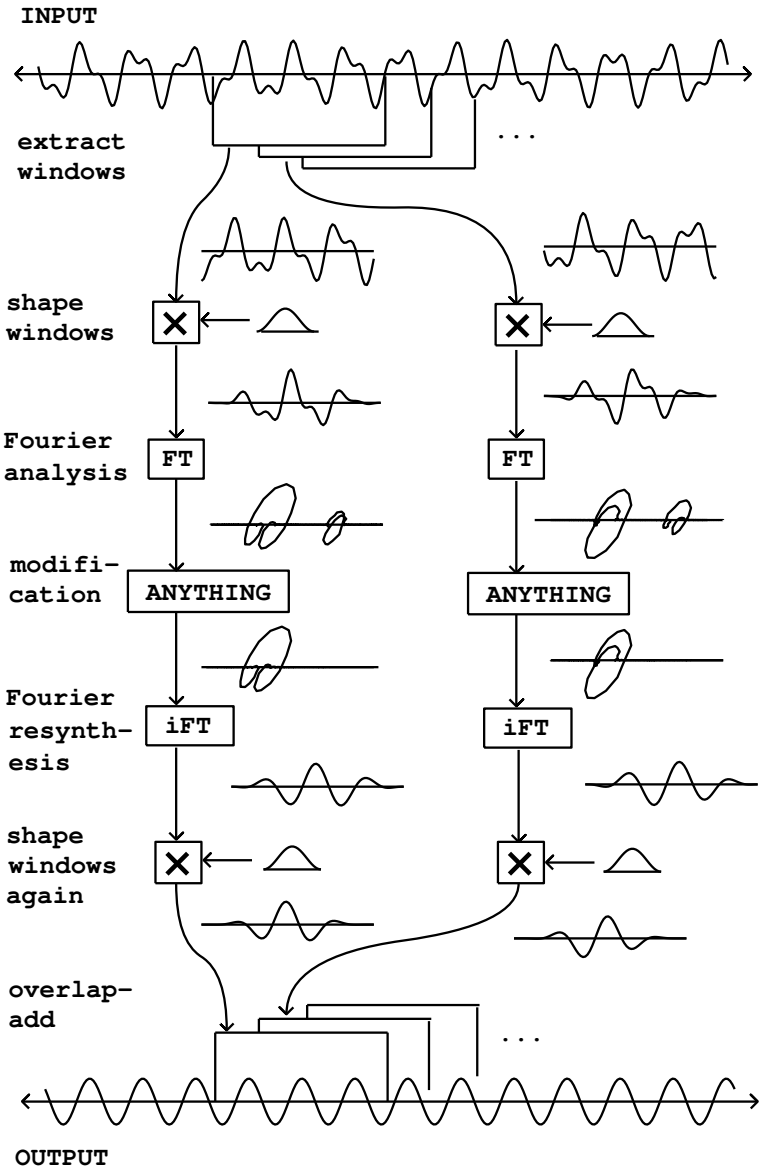


Рис. 9.7: Анализ со скользящим окном и ресинтез аудиосигнала с использованием преобразований Фурье. В этом примере сигнал фильтруется умножением преобразования Фурье на требуемый частотный отклик.

Как показано в §9.1.2, это может быть сделано с помощью взятия другого преобразования Фурье с нормированием и перевернув результат в обратном направлении. В случае, когда восстановленное окно не стремится гладко к нулю на своих концах, мы применяем функцию окна Ханна второй раз. Проделывая это для каждого окна входа последовательно, мы затем складываем выходы, используя то же перекрытие, что и для анализа.

Если мы используем окно Ханна с перекрытием на три четверти (то есть выбирая N кратным четырём и начиная следующее окно после $H = N/4$ семплов предыдущего), мы можем реконструировать первоначальный сигнал точно, опуская шаг "модификации". Это связано с тем, что iFT (обратное преобразование Фурье) обращает действие FT (преобразование Фурье) и, таким образом, мы умножаем каждое окно на квадрат функции Ханна. Выход тем самым есть вход, умноженному на квадрат функции Ханна, с добавлением перекрытия на три четверти. Простая проверка показывает, что это приводит к константе 3/2, так что выход равен входу, умноженному на постоянный множитель.

Возможность точного восстановления входного сигнала полезна, потому что некоторые типы модификаций могут выполняться постепенно, и поэтому выход может быть сделан плавно изменяющимся между входным сигналом и некоторой его преобразованной версией.

9.4.1 Узкополосный компандер

Компандером называется устройство, которое усиливает сигнал с переменным коэффициентом усиления, зависящим от измеряемой амплитуды сигнала. Этот термин является производным от "compressor" ("сжиматель") и "expander" ("расширитель"). Коэффициент усиления компрессора уменьшается, когда уровень входа растёт, так что *динамический диапазон*, то есть общее изменение уровня сигнала, уменьшается. Экспандер, наоборот, увеличивает динамический диапазон. Часто коэффициент усиления зависит не только от текущего уровня сигнала, но и от его истории; например, скорость изменения может ограничиваться или может осуществляться временная задержка.

С использованием анализа и ресинтеза Фурье мы можем осуществлять индивидуальное компандирование для узкополосных каналов. Если $C[m]$ есть одна такая полоса, мы применяем коэффициент усиления $g[m]$ к ней, что даёт $g[m]C[m]$. Хотя $C[m]$ является комплексным числом, коэффициент усиления будет неотрицательным действительным числом. В общем, коэффициент усиления может быть функцией не только $C[m]$, но также любого или всех предыдущих семплов в канале: $C[m-1]$, $C[m-2]$ и так далее. Здесь мы рассмотрим простейшую ситуацию, в которой коэффициент усиления является просто функцией модуля текущего значения $|C[m]|$.

Схема на рис. 9.8 показывает одно очень полезное приложение компандеров, на-

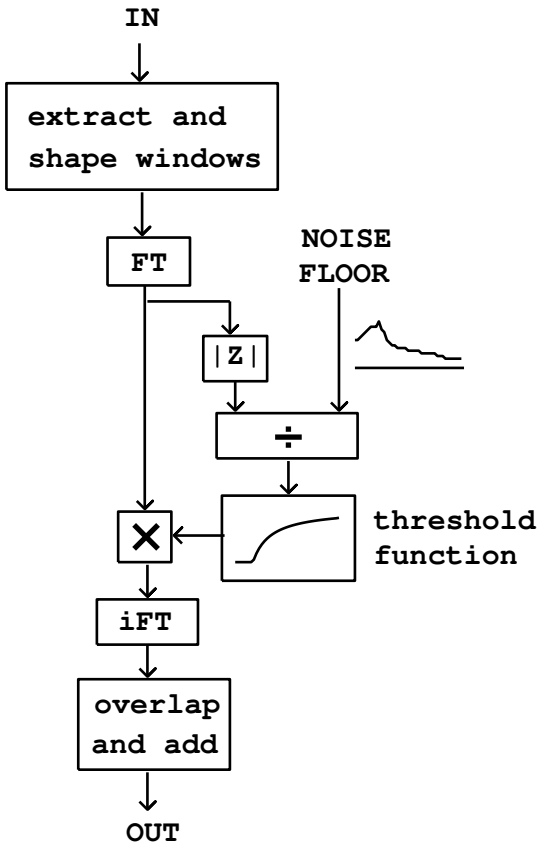


Рис. 9.8: Блок-схема узкополосного подавления шума с помощью компандера.

зываемое *пороговый шумоподавитель*[1]. Здесь коэффициент усиления $g[m]$ зависит от амплитуды канала $C[m]$ и уровня шума, который является функцией f номера канала k. Для ясности мы будем подставлять частотный индекс k в коэффициент усиления, теперь записываемый как $g[m,k]$, и в оконное преобразование Фурье $S[m,k] = C[m]$. Коэффициент усиления даётся формулой

$$g[m,k] = \begin{cases} 1 - f[k]/|S[m,k]|, & \text{если } |S[m,k]| > f[k] \\ 0, & \text{иначе.} \end{cases}$$

Пока модуль $S[m,k]$ будет меньше, чем пороговое значение $f[k]$, коэффициент усиления равен нулю и, таким образом, амплитуда $S[m,k]$ заменяется на ноль. В противном случае, умножение на $g[m,k]$ уменьшает модуль до величины $|S[m,k]| - f[k]$. Поскольку коэффициент усиления является неотрицательным действительным числом, фаза сохраняется.

На рисунке коэффициент усиления вычисляется как пороговая функция отношения $x = |S[m,k]|/f[k]$ — модуля сигнала к уровню шума; пороговое значение есть $g(x) = 1 - 1/x$, когда $x < 1$, и ноль в противном случае, хотя без труда можно подставить и другие функции порогового значения.

Этот метод полезен для удаления шума из записанного звука. Мы или измеряем, или задаём уровень шума $f[k]$. В силу конструкции функции усиления $g[m,k]$, только амплитуды, которые выше уровня шума, поступают на выход. Поскольку это проделывается для узких частотных полос, иногда можно убрать значительную часть шума, в то время как сигнал сам по себе в частотных диапазонах, где он громче чем уровень шума, практически сохраняется.

Данный метод также полезен в качестве подготовки перед применением к звуку нелинейной операции, такой как искажение. Часто лучше всего искажать только наиболее характерные частоты звука. Вычитание шумоизолированного звука из оригинала дает остаточный сигнал, который может быть пропущен без искажений.

9.4.2 Наложение тембра (классический вокодер)

Вторым приложением анализа и ресинтеза Фурье является изменяемый во времени фильтр, имеющий возможность создавать звук, следующий спектральной огибающей другого. Это устройство широко известно в кругах электронной музыки как *вокодер*, название которого, не очень корректно, восходит к первоначальному прибору вокального анализа и синтеза лабораторий Белла. Метод, излагаемый здесь, более корректно называть наложением тембра. Используются два входных сигнала: один для фильтрации, а другой для управления фильтром с помощью спектральной огибающей, изменяемой во времени. Оконное преобразование Фурье используется как для управляющего сигнального входа, чтобы измерить спектральную огибающую, так и для фильтруемого входа, для того чтобы задать фильтрацию.

[1] часто используется термин *нойз-гейт*

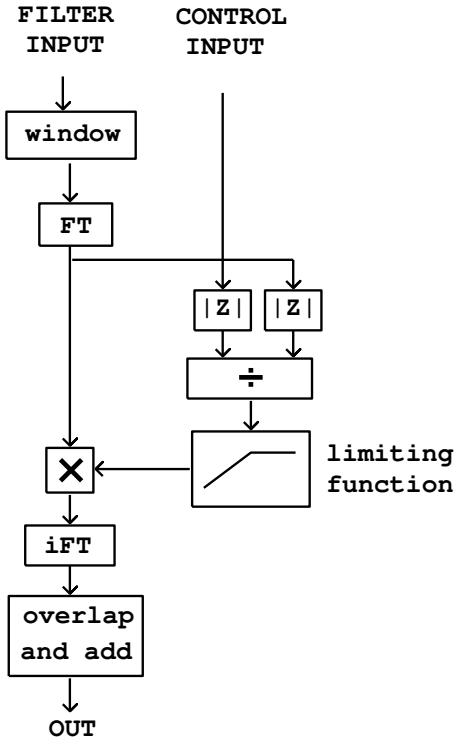


Рис. 9.9: Блок-схема наложения тембра (известного как эффект “вокодер”).

Блок-схема для наложения тембра показана на рис. 9.9. Как и в предыдущем примере, наложение тембра производится умножением комплекснозначного оконного преобразования Фурье фильтруемого входа на неотрицательные действительные числа, меняя тем самым их модули, но оставляя фазы нетронутыми. Здесь трюк состоит в том, что мы просто хотим заменить первоначальные модули $|S(m,k)|$ на модули, полученные от управляющего сигнала (назовём их, скажем, $|T(m,k)|$). Необходимый коэффициент усиления есть, таким образом,

$$g[m,k] = \frac{|T[m,k]|}{|S[m,k]|}.$$

На практике лучше ограничить коэффициент усиления некоторым максимальным значением (которое может зависеть от частоты), поскольку в противном случае каналы, не содержащие ничего кроме шума, боковые лепестки или даже ошибки отбрасывания знаков вычислений, могут увеличиться до слышимого уровня. Так что перед использованием, для коэффициента усиления применяется подходящая ограничивающая функция.

9.5 Фаза

До сих пор мы оперировали с сигналами, изменяя модули их оконных преобразований Фурье, но оставляя фазы неизменными. Модули кодируют спектральную

огибающую звука. Фазы, с другой стороны, кодируют частоты и время, в том смысле, что изменение фазы от одного окна к другому накапливается по времени в согласии с частотой. Для того чтобы сделать преобразование, которое допускает независимое управление по частоте и времени, требуется анализ и восстановление фазы.

В примерах анализа/синтеза предыдущего параграфа, фаза выхода копируется напрямую из фазы входа. Это является приемлемым, когда выходной сигнал соответствует времени входного сигнала. Иногда требуется как-то изменить время, например, растянуть или сжать его. С другой стороны, выходная фаза может зависеть от более чем одного входа, например, при звуковом морфинге[2].

Рисунок 9.10 показывает, как фаза преобразования Фурье меняется от окна к окну, при заданной исходной синусоиде на входе. Частота синусоиды есть $\alpha = 3\omega$, так что пик в преобразовании Фурье имеет центр при $k = 3$. Если первоначальная фаза есть ϕ, тогда окрестные фазы могут быть заданы как

$$\begin{array}{lll}
\angle S[0,2] = \phi + \pi & \angle S[0,3] = \phi & \angle S[0,4] = \phi + \pi \\
\angle S[1,2] = \phi + H\alpha + \pi & \angle S[1,3] = \phi + H\alpha & \angle S[1,4] = \phi + H\alpha + \pi \\
\angle S[2,2] = \phi + 2H\alpha + \pi & \angle S[2,3] = \phi + 2H\alpha & \angle S[2,4] = \phi + 2H\alpha + \pi.
\end{array}$$

Это даёт прекрасный способ оценки частоты α: возьмём произвольный канал, чья амплитуда подчиняется синусоиде и возьмём разность двух последовательных фаз, чтобы получить $H\alpha$:

$$H\alpha = \angle S[1,3] - \angle S[0,3]$$

$$\alpha = \frac{\angle S[1,3] - \angle S[0,3] + 2p\pi}{H},$$

где p — целое число. Имеется H возможных частот, разделённых промежутками $2\pi/H$. Если мы используем 4-х кратное перекрытие, то есть $P = N/4$, частоты разделены промежутком $8\pi/N = 4\omega$. К счастью, это ширина основного лепестка функции Ханна, так что не более одного возможного значения α может соответствовать любой измеренной разности фаз внутри основного лепестка пика. Корректное значение выбираемого p равно такому, которое даёт частоту, наиболее близкую к номинальной частоте канала $k\omega$.

При вычислении фаз для синтеза нового или модифицированного сигнала, необходимо поддерживать подходящую взаимосвязь фаз между последовательными окнами ресинтеза и, также, одновременно между смежными каналами. Однако эти два семейства соотношений не всегда совместимы. Мы прежде всего позаботимся о соотношениях между последовательными окнами ресинтеза и отложим беспокойство о фазовых соотношениях между каналами.

Предположим, что мы хотим построить m-ый спектр $S[m,k]$ для ресинтеза (при построенном предыдущем, номер $m-1$). Предположим, что мы хотим фазовое

[2]Морфинг — звуковой эффект, заключающийся в наложении характеристик одного звука на другой. (*прим. перев.*)

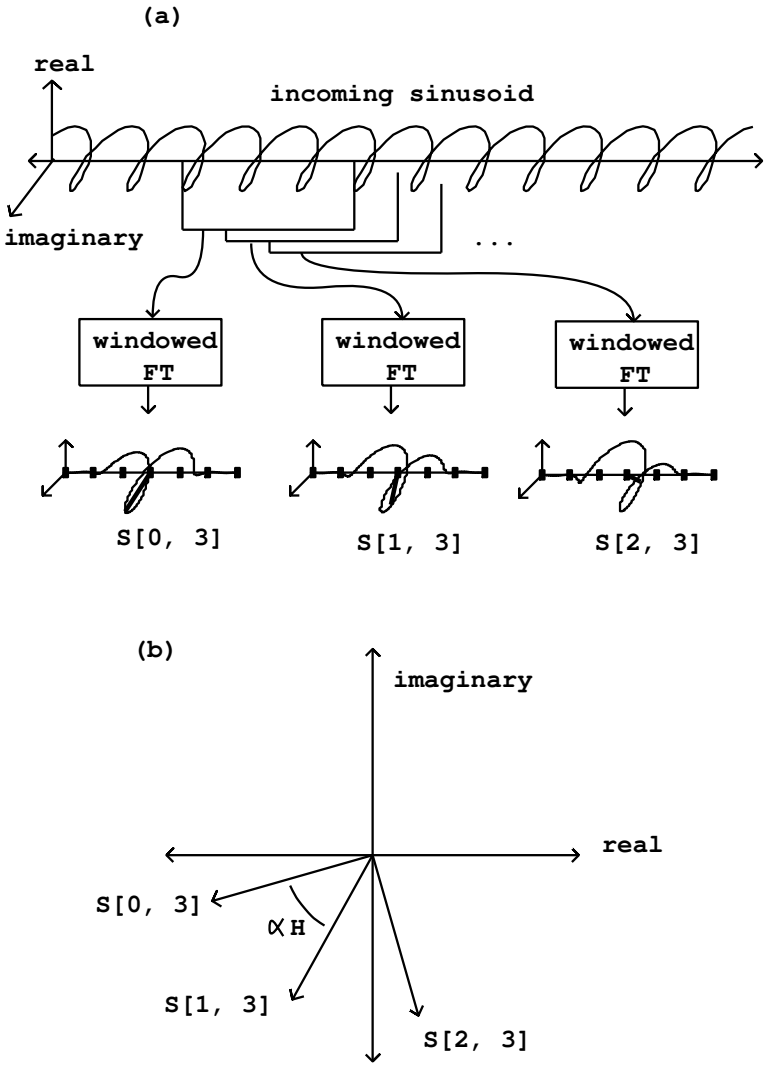


Рис. 9.10: Фазы при оконном преобразовании Фурье: (a) анализ комплексной синусоиды в трёх последовательных окнах; (b) результат для одного канала ($k = 3$), для трёх окон.

соотношение между окнами $m-1$ и m такое же, как у сигнала $x[n]$, но фазы окна $m-1$ могут возникнуть из тех или иных соображений и не обязаны отвечать нашим желаниям.

Для того чтобы найти, насколько сильно фаза каждого канала должна отличаться от предыдущей, мы проделываем два анализа сигнала $x[n]$, разделённых той же величиной скачка H, которую мы используем при ресинтезе:

$$T[k] = \mathcal{FT}(W(n)X[n])(k)$$

$$T'[k] = \mathcal{FT}(W(n)X[n+H])(k).$$

Рисунок 9.11 показывает процесс накопления фазы, при котором каждая фаза выхода зависит от предыдущей фазы выхода и фазовой разности для двух оконных анализов входа. Рисунок 9.12 иллюстрирует фазовое соотношение в комплексной плоскости. Фаза нового выхода $S[m,k]$ должна быть равна предыдущей плюс разность между фазами двух анализов:

$$\begin{aligned}\angle S[m,k] &= \angle S[m-1,k] + (\angle T'[k] - \angle T[k]) \\ &= \angle\left(\frac{S[m-1,k]T'[k]}{T[k]}\right).\end{aligned}$$

Здесь мы использовали тот факт, что умножение или деление двух комплексных чисел даёт сумму или разность их аргументов.

Если требуемый модуль есть действительное число a, тогда мы должны установить $S[m,k]$ на значение

$$S[m,k] \;=\; a \;\cdot\; \left|\frac{S[m-1,k]T'[k]}{T[k]}\right|^{-1} \;\cdot\; \frac{S[m-1,k]T'[k]}{T[k]}$$

Модули второго и третьего коэффициентов равны единице, так что модуль $S[m,k]$ сводится к a; первые два коэффициента являются действительными числами, так что аргумент управляется последним множителем.

Если мы так же хотим завершить модулем из спектра T, мы можем положить $a = |T'[k]|$ и упростить:

$$S[m,k] \;=\; \left|\frac{S[m-1,k]}{T[k]}\right|^{-1} \;\cdot\; \frac{S[m-1,k]T'[k]}{T[k]}$$

9.5.1 Фазовые соотношения между каналами

В схеме выше фаза каждого $S[m,k]$ зависит только от предыдущего значения того же канала. Фазовое соотношение между соседними каналами оставляется произвольным. Иногда такой подход работает хорошо, но в других ситуациях

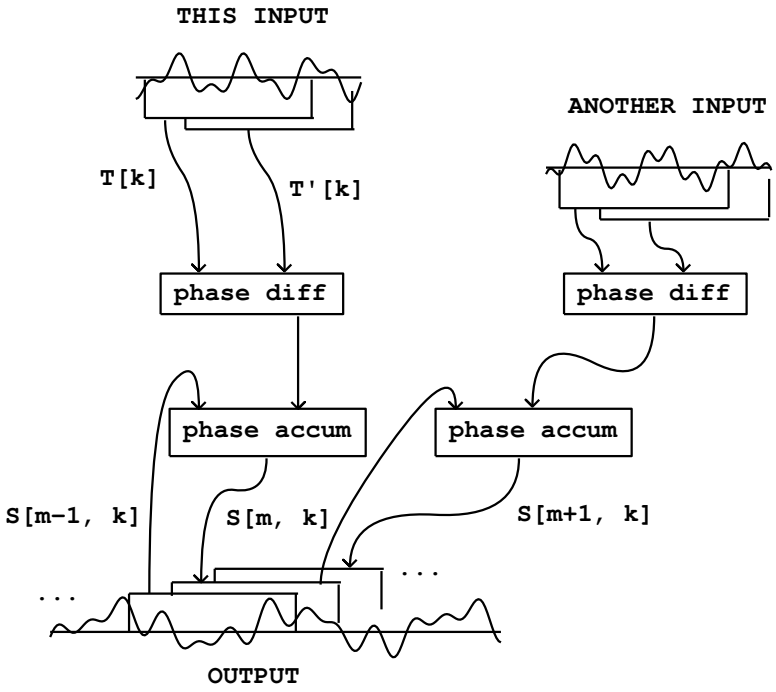


Рис. 9.11: Изменение фазы при ресинтезе. Каждая фаза, как у $S[m,k]$ здесь, зависит от предыдущей выходной фазы и разности между входными фазами.

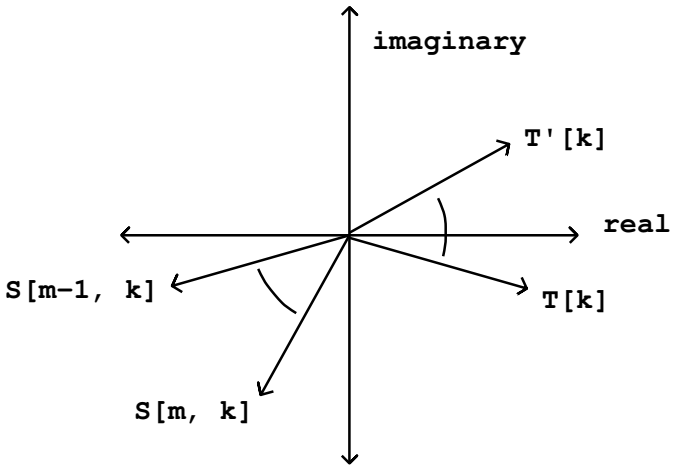


Рис. 9.12: Фазы одного канала окон анализа и два последовательных окна ресинтеза.

несогласованность соседних каналов приводит к нежелательному эффекту хора. В идеале мы бы хотели, чтобы не только между $S[m,k]$ и $S[m,k+1]$ было бы такое же фазовое соотношение, как между $T'[k]$ и $T'[k+1]$, но также и между $S[m,k]$ и $S[m-1,k]$ было бы тоже фазовое соотношение, как между $T'[k]$ и $T[k]$.

Эти $2N$ уравнений для N фаз в общем случае не будут иметь решения, но мы можем изменить приведённое выше уравнение для $S[m,k]$ так, что всякий раз, когда возникает решение для чрезмерно ограниченной системы уравнений, алгоритм реконструкции фокусируется на этом решении. Этот подход называется *фиксацией фазы* [Puc95b] и обладает достоинством большей простоты, хотя доступны и более сложные методы [LD97].

Требуемое выходное фазовое соотношение в точке $m-1$ есть

$$\angle T[k+1] - \angle T[k] = \angle S[m-1,k+1] - \angle S[m-1,k]$$

или, после перегруппировки:

$$\angle \left\{ \frac{S[m-1,k+1]}{T[k+1]} \right\} = \angle \left\{ \frac{S[m-1,k]}{T[k]} \right\}.$$

Другими словами, фаза отношения S/T не должна зависеть от k. Принимая это во внимание, мы можем переписать рекурсивную функцию для $S[m,k]$:

$$S[m,k] \;=\; |R[k]|^{-1} \cdot R[k]T'[k]$$

где

$$R[k] \;=\; \frac{\overline{T[k]} \;\cdot\; S[m-1,k]}{|S[m-1,k]|},$$

и, в силу предыдущего уравнения, $R[k]$ должно быть согласовано по фазе. Теперь трюк заключается в том, чтобы заменить $R[k]$ для каждого k на сумму трёх соседних значений. Вычисление даёт:

$$S[m,k] \;=\; |R'[k]|^{-1} \;\cdot\; R'[k]T'[k],$$

где

$$R'[k] = R[k+1] + R[k] + R[k-1].$$

Если каналы уже находятся в корректных фазовых соотношениях, это не даёт эффекта (результирующая фаза будет та же, как если бы использовалось лишь $R[k]$). Но, в общем, сумма будет содержать два соседних члена в $k+1$:

$$R'[k+1] = R[k+2] + R[k+1] + R[k]$$

так что R' будет лучше приближаться к одной точке в том же направлении, что R. Итерационное применение будет, в конечном счёте, выстраивать все R' к одному значению фазы по мере того, как фазовые соотношения между измеряемыми спектрами T и T' это позволяют.

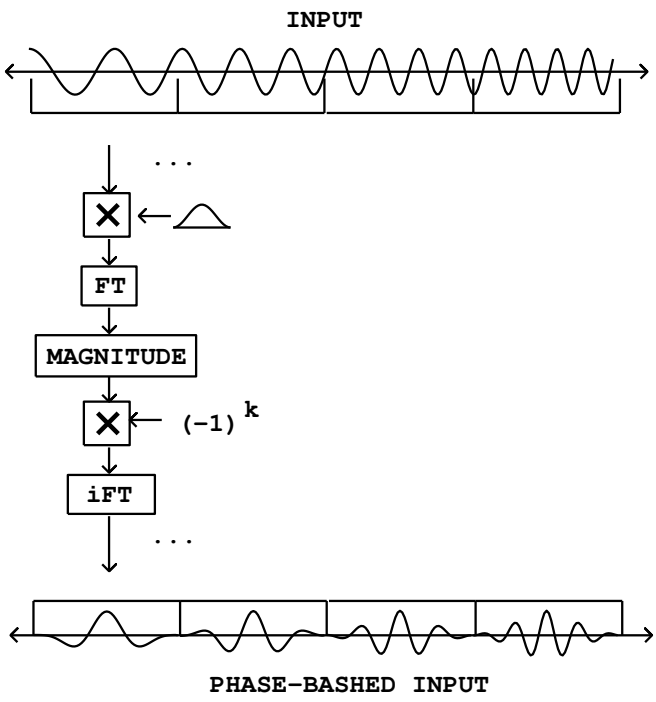


Рис. 9.13: Фазовые биения записанного звука (на примере синусоиды с увеличивающейся частотой) для получения последовательности волновых таблиц осциллятора.

9.6 Фазовые биения

В параграфе §2.3, посвящённом семплированию с огибающей, мы видели, как создать периодическую форму волны из предварительно записанного звука, заимствуя тембр изначального звука, воспроизводимого, тем не менее, с указанной высотой. Если окно семпла сделано изменяемым во времени, получающийся тембр также меняется, имитируя записанный звук.

Возникает одна важная проблема, состоящая в том, что если мы берём формы волн из различных окон семпла (или разных семплов), то нет гарантии, что их фазы будут соответствовать друг другу. Если это не так, результат может быть неприятным, поскольку случайные изменения фазы прослушиваются как флуктуации частоты. Такая ситуация может быть исправлена с использованием анализа и ресинтеза Фурье [Puc05].

Рисунок 9.13 показывает простой способ использования анализа Фурье для выравнивания фаз в последовательности окон при записи. Мы просто берём *быстрое преобразование Фурье* (FFT) окна и далее устанавливаем каждую фазу на ноль для чётных значений k и на π для нечётных. Фаза в центре окна есть, таким образом, ноль как для чётных, так и для нечётных k. Для установки фаз (аргументов комплексных амплитуд в спектре) нужным образом, мы, во-первых, ищем модуль, который может рассматриваться, как комплексное число с аргументом ноль. Дальнейшее умножение на $(-1)^k$ устанавливает амплитуду положительной

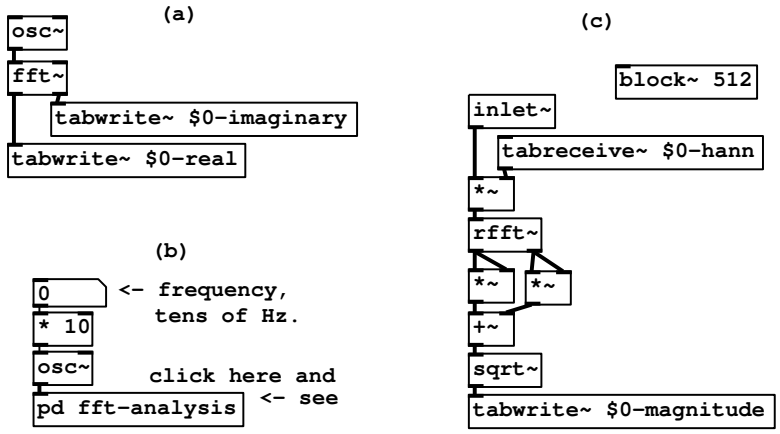


Рис. 9.14: Анализ Фурье в Pd: (a) объект fft~; (b) использование сабпатча для управления размером блока преобразования Фурье; (c) сабпатч: использование действительного преобразования Фурье (объект fft~) и оконной функции Ханна.

или отрицательной попеременно. После этого мы берём обратное преобразование Фурье даже без применения окна; возможно, мы захотим это сделать каким-либо образом позднее, как показано на рис. 2.7. Результат может быть объединен с техникой модуляции Главы 6 для создания мощных инструментов вокального и других типов имитационного синтезов.

9.7 Примеры

9.7.1 Анализ и ресинтез Фурье в Pd

Пример I01.Fourier.analysis.pd (рис. 9.14, часть a) демонстрирует вычисление преобразования Фурье аудиосигнала с использованием объекта fft~:

> fft~: *быстрое преобразование Фурье.* Два входа принимают аудиосигналы, представляющие из себя действительную и мнимую части комплекснозначного сигнала. Размер окна N задаётся размером блока Pd. Одно преобразование Фурье выполняется для каждого звукового блока. Вычислительная нагрузка быстрого преобразования Фурье (FFT — Fast Fourier Transform) [Smi03] примерно равна вычислениям от 5 до 15 объектов osc~ в типичных конфигурациях. Алгоритм FFT в его простейшей форме использует в качестве N степень двойки, что также (обыкновенно) является ограничением, накладываемым на размер блока в Pd.

Пример I02.Hann.window.pd (рис. 9.14, часть b и c) показывает, как управлять размером блока с использованием объекта block~, как применять окно Ханна, и

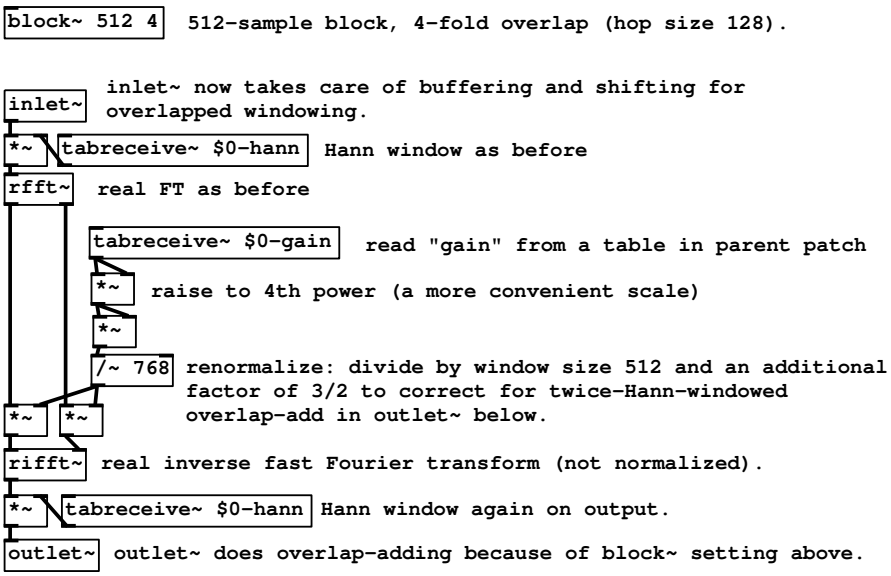


Рис. 9.15: Анализ и ресинтез Фурье, с использованием block~ для определения четырехкратного перекрытия, и rifft~ для ресинтеза сигнала после модификации.

другой вариант преобразования Фурье. Часть (b) показывает обращение к сабпатчу, которое, в свою очередь, показано в части (c). Новыми объектами являются:

rfft~ : *вещественное быстрое преобразование Фурье*. Мнимая часть входа предполагается равной нулю. Заполняются только первые $N/2+1$ каналов выхода (остальные определяются с помощью симметрии). Это экономит половину вычислительного времени по сравнению с (более общим) объектом fft~ .

tabreceive~ : циклически выводит содержание волновой таблицы. Каждый блок вычисления выводит одни и те же N семплов таблицы.

В этом примере таблица “$0-hann” содержит окно Ханна размером 512 семплов, согласованное с указанныи размером звукового блока. Сигнал для анализа появляется (из родительского патча) через объект inlet~ . Амплитуды канала (выход объекта rfft~) ограничиваются действительнозначными модулями: действительные и мнимые части возводятся в квадрат по отдельности, квадраты складываются и результат проходит через объект sqrt~ . Наконец, модуль записывается (с управляющим соединением, не показанным на рисунке) с помощью tabwrite~ в другую таблицу “$0-magnitude” для графического отображения.

Пример I03.resynthesis.pd (рис. 9.15) показывает, как анализировать и производить ресинтез аудиосигнала, следуя стратегии рис. 9.7. Как и раньше, имеется сабпатч для установки подходящего размера блока; рисунок показывает только сабпатч. Нам нужен один новый объект для обратного преобразования Фурье:

rifft~ : *вещественное обратное быстрое преобразование Фурье*. Используются первые $N/2+1$ точек входа (взятых как пара действительная/мнимая части)

с установкой соответствующих значений для других каналов по симметрии для восстанавления вещественного выхода. Никакая нормализация не производится, так что последовательно соединенная пара rfft~ / rifft~ приводит к коэффициенту усиления N. Объект ifft~ также является допустимым: он вычисляет ненормированное обращение объекта fft~, восстанавливающее комплекснозначный выход.

В сабпатче для объекта block~ указан второй аргумент, который определяет коэффициент перекрытия, равный 4. Это заставляет сабпатч запускаться четыре раза каждые 512 семплов, с регулярными интервалами в 128 семплов. Объект inlet~ производит необходимую буферизацию и переупорядочивание семплов, так что его вывод всегда даёт 512 последних семплов входа. В другом направлении объект outlet~ складывает сегменты его предыдущих четырёх входов для реализации схемы сложения с перекрытием, показанной на рис. 9.7. Блоки, состоящие из 512 семплов, умножаются на окно Ханна как на входе, так и на выходе. Если объекты rfft~ и rifft~ были соединены без каких-либо модификаций между ними, выход точно восстанавливает входной сигнал.

Тем не менее, применена модификация: каждый канал умножается на (действительный положительный) коэффициент усиления. Комплекснозначная амплитуда каждого канала масштабируется отдельным умножением действительной и мнимой частей на коэффициент усиления. Коэффициент усиления (который зависит от канала) берётся из другой таблицы, названной “$0-gain”. Результатом является графический фильтр-эквалайзер; рисуя указателем курсора в графическом окне этой таблицы, вы можете создавать кривые зависимости коэффициента усиления от частоты.

Существует неотъемлемая задержка из-за использования block~ для увеличения размера блока (но её нет, если он используется, как показано в Главе 7, для уменьшения размера блока относительно родительского окна). Задержка может быть измерена от входа до выхода сабпатча и равна разности между двумя размерами блоков. В этом примере задержка буферизации составляет $512 - 64 = 448$ семплов. Задержка, связанная с величиной блоков, не зависит от величины перекрытия, а только от размера блоков.

9.7.2 Узкополосный компандер: подавление шума

Пример I04.noisegate.pd (рис. 9.16) показывает пример узкополосного компандера, использующего анализ Фурье и восстановление. (Это реализация блок-схемы рис. 9.8.) Часть (a) рисунка показывает конфигурацию фильтра, сходную с предыдущим примером, за исключением того, что коэффициент усиления для каждого канала теперь является функцией модуля канала.

Для каждого k, если $s[k]$ обозначает мощность, сосредоточенную в канале k, и $m[k]$ — уровень маскировки (уровень, имеющий предполагаемо большую мощность, чем мощность шума для канала k), тогда коэффициент усиления для канала k

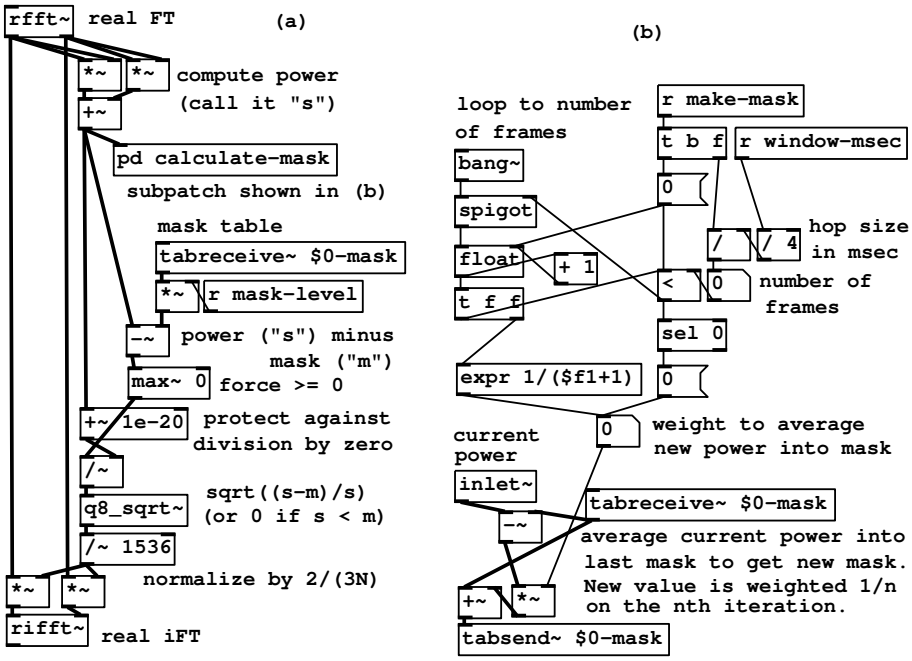


Рис. 9.16: Подавление шума в качестве примера узкополосного компандера: (a) анализ и восстановление сигнала; (b) вычисление “маски”.

даётся выражением

$$\begin{cases} \sqrt{\frac{s[k]-m[k]}{s[k]}}, & \text{если } s[k] > m[k] \\ 0, & \text{иначе.} \end{cases}$$

Мощность в k-ом канале уменьшается на $m[k]$, если это возможно, или заменяется на ноль.

Маска сама по себе является результатом измерения среднего шума в каждом канале, который содержится в таблице “$0-mask”, умноженного на величину, называемую “уровнем маскировки”. Средний шум вычисляется в сабпатче (pd calculate-mask), чьё содержание показано в части (b) рисунка. Для вычисления маски мы используем два новых объекта:

bang~ : отправка “bang” перед каждым блоком вычисления. Сообщение “bang” появляется в логическое время первого семпла в каждом блоке (наиболее раннее логическое время, чьё управляющее вычисление воздействует на этот блок, но не на предыдущий), следуя схеме, показанной на рис. 3.2.

tabsend~ : объект, являющийся компаньоном tabreceive~ , циклически записывая входной сигнал в таблицу, используя первые N семплов таблицы.

Процесс усреднения мощности начинается с отправки необходимой продолжительности времени в миллисекундах на “make-mask”. Патч вычисляет эквивалентное этому времени число блоков b и генерирует последовательность весовых коэффициентов: $1, 1/2, 1/3, \ldots, 1/b$, с которыми каждая из мощностей последовательных b

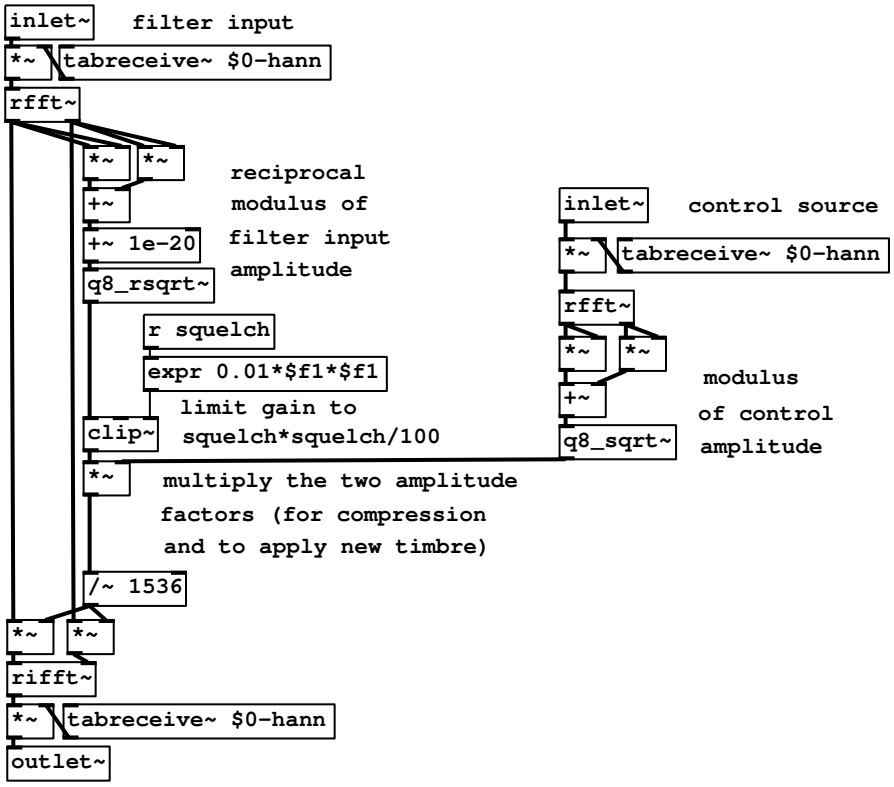


Рис. 9.17: Наложение тембра.

блоков усредняется с тем, что содержится в таблице маскировок для предыдущего блока. В конце b блоков таблица содержит одинаково взвешенное среднее всех b измерений мощности. После этого вес для усреднения новых измерений мощности становится равным нулю, так что измеренное среднее перестаёт меняться.

Для использования этого патча при подавлении классического шума требуется по крайней мере несколько секунд образца записанного шума без наличия полезного сигнала. Запись этого образца идет в патч, а продолжительность отправляется на “make-mask”, в результате таблица “$0-mask” содержит среднюю измеренную мощность шума для каждого канала. Далее, предполагая, например, что шумовая часть сигнала редко превышает его одной десятой средней мощности, мы устанавливаем “уровень маскировки” равным 10 и сигнал, шум которого должен быть подавлен, посылается через часть (a) патча. Шум будет почти удалён, но те каналы, в которых сигнал превышает, скажем, двадцатикратную мощность шума, будут уменьшены лишь на 3дб, а каналы с большей мощностью существенно меньше. (Конечно, действительное подавление шума может быть не самым интересным применением патча; можно пытаться маскировать любой сигнал от каких-либо других.)

9.7.3 Наложение тембра (“вокодер”)

Пример I05.compressor.pd (рис. 9.17) представляет из себя другой компандер канала, который представлен в качестве подготовки к примеру I06.timbre.stamp.pd, который мы разберём следующим. Это немного изменённая реализация схемы наложения тембра рис. 9.9.

Имеется два входа: один, который подвергается фильтрации, слева (его преобразование Фурье используется для ресинтеза после модификации абсолютных значений), и второй справа, который действует как источник управления. Грубо говоря, если абсолютное значение первого сигнала равно $f[k]$ (для входа фильтра), а второго — $c[k]$ (для источника управления), мы просто “отбеливаем” вход фильтра, умножая его на $1/f[k]$ и далее накладываем абсолютные значения управляющего сигнала на результат с помощью дальнейшего умножения на $c[k]$. На практике мы должны ограничить коэффициент усиления некоторым разумным максимальным значением. В рассматриваемом патче это делается с помощью ограничения “отбеливающих” сигнал коэффициентов $1/f[k]$ определённым максимальным значением с помощью объекта clip~. Предел управляется параметром “подавление” (“squelch”), который возводится в квадрат и делится на 100 для отображения значений от 0 до 100 на используемый диапазон.

Другая возможная схема состоит в ограничении коэффициента усиления после нахождения отношения $c[k]/f[k]$. Ограничение коэффициента усиления может, так или иначе, зависеть от частоты. Иногда также полезно поднимать коэффициент усиления с весомым коэффициентом p между 0 и 1; если это 1 — получаем наложение тембра, и если 0 — сигнал проходит через фильтр без изменения, а значения между ними дают гладкую интерполяцию между двумя такими вариантами.

9.7.4 Управление временем с помощью фазового вокодера

Фазовый вокодер обыкновенно относят к общей технике перехода от (комплекснозначных) амплитуд канала к парам, состоящим из (действительнозначных) модулей и скоростей изменения фазы (“частот”) и обратно, как описано на рис. 9.11 (§9.5). В примере I07.phase.vocoder.pd (рис. 9.18) мы используем эту технику со специфической целью растяжения или сжатия по времени записанного звука с управлением в режиме реального времени. Таким образом, в любой момент реального времени мы управляем позицией в записанном звуке, который мы слушаем. Использованы два новых объекта:

lrshift~: *сдвиг звукового блока налево или направо* (в соответствии с указанным аргументом). Если аргумент положительный, каждый выходной блок является копией входного, сдвинутого направо на указанное число семплов, с заполнением нулями слева. Отрицательный аргумент сдвигает налево, заполняя нулями правую часть.

q8_rsqrt~: *быстрое приблизительное вычисление обратного к квадратному*

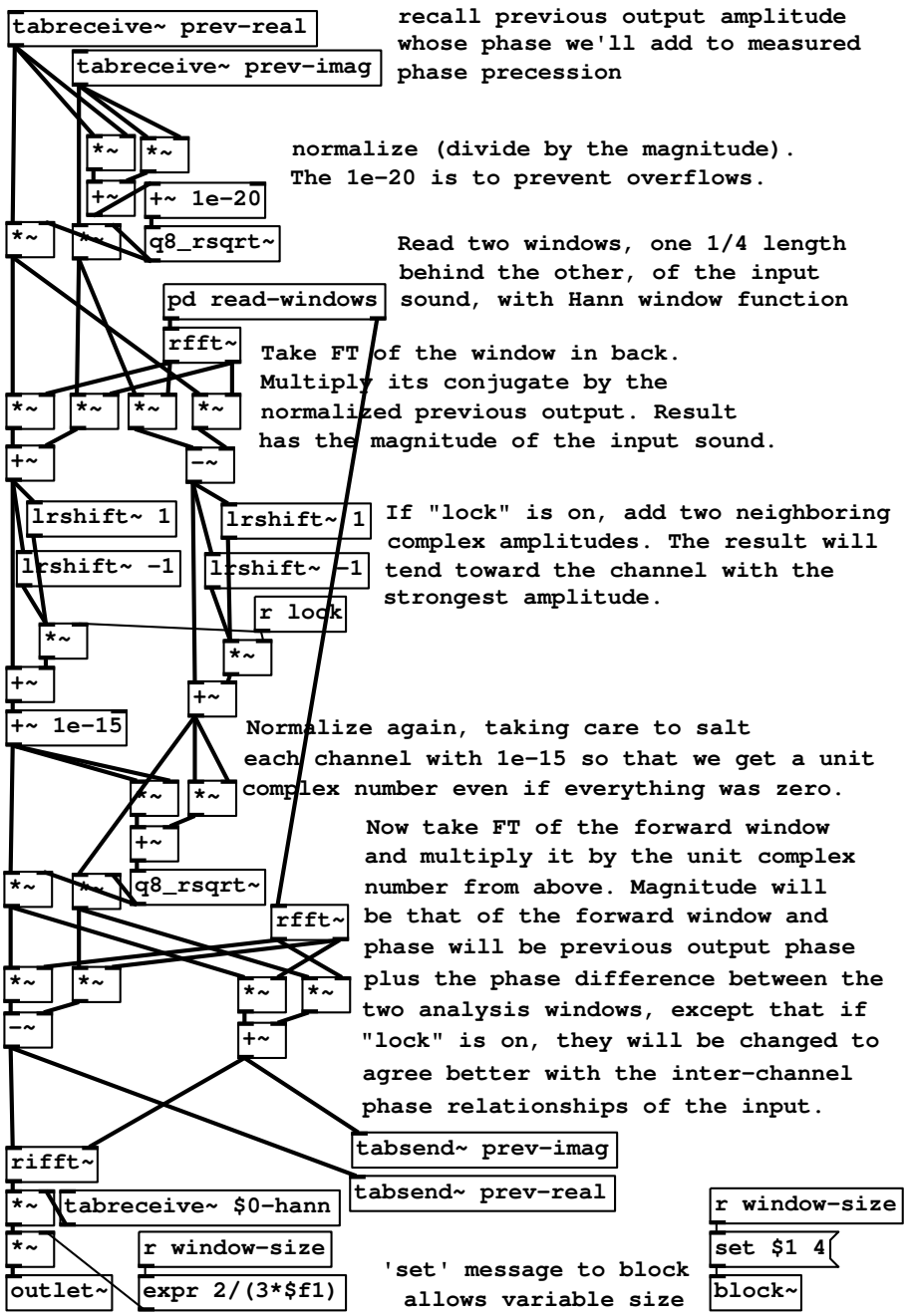


Рис. 9.18: Фазовый вокодер для растяжения и сжатия времени.

корню. Выводит обратное квадратного корня входа с точностью около $\approx 1/256$, с использованием гораздо меньшего количества вычислений, чем требует точное вычисление квадратного корня и взятие обратного.

Процесс начинается с сабпатча pd read-windows, который выводит два блока записанного звука, помещённые в окна Ханна — "последний" и "передний" — со смещением на 1/4 окна по записи. Показанное окно использует два выхода сабпатча для установки изменений амплитуды и фазы каждого канала его собственного выхода.

Два верхних объекта tabreceive~ выводят предыдущий блок комплексных амплитуд, посланных на объект rifft~ внизу, соответствующий $S[m-1,k]$ в дискуссии §9.5. В целом патч вычисляет $S[m,k]$ и затем его обратное преобразование Фурье с окном Ханна на выходе.

После нормализации $S[m-1,k]$ его комплексно сопряжённое (нормированное обратное) умножается на оконное преобразование Фурье "последнего" окна $T[k]$, давая в результате $R[k]$ на стр. 278. Далее, в зависимости от значения параметра "lock", вычисленное значение $R[k]$, по условию, заменяется на версию с зафиксированной фазой $R'[k]$. Это выполняется с использованием объектов lrshift~, чьи выходы складываются с $R[k]$, если "lock" установлен на единицу. Затем результат нормируется и умножается на преобразование Фурье с использованием окна Ханна, "переднего" окна $T'[k]$ для получения $S[m,k]$.

Три других применения техник анализа и ресинтеза Фурье, не показанные здесь в виде иллюстраций, можно посмотреть в документации Pd. Во-первых, Пример I08.pvoc.reverb.pd показывает, как сделать фазовый вокодер с обратными связями, как в ревербераторе, за исключением того, что индивидуальные каналы заменяются на входной сигнал, когда он мощнее рециркулирующих сигналов. В результате получается более связно звучащий эффект реверберации, чем это можно сделать классическим способом с использованием линий задержки.

Пример I09.sheep.from.goats.pd демонстрирует (несовершенный) метод отделения сигналов с определённой высотой от шумовых, канал за каналом, основываясь на согласованности фазы, которую мы должны ожидать от синусоиды, помещённой в окно Ханна. Если три настраиваемых канала приблизительно на π радиан разнятся в фазе один от другого, они полагаются принадлежащими пику синусоиды. Каналы, принадлежащие пикам синусоиды, заменяются на ноль для извлечения шумовой составляющей сигнала, или все другие заменяются на ноль, чтобы дать синусоидальную часть.

Пример I10.phase.bash.pd возвращает нас к циклическому семплеру волновой таблицы рис. 2.7 и показывает, как выровнять фазы так, чтобы все компоненты сигнала имели нулевую фазу в точках 0, N, $2N$, и так далее. Тем самым, две копии циклического семплера, каждый независимо выводящий N семплов, могут связно микшироваться. Синтезированная, изменяемая по высоте версия первоначального звукового файла может быть сделана при помощи циклического микширования.

Упражнения

1. Сигнал $x[n]$ равен 1 для $n = 0$ и 0 в противном случае (импульс). Что является его (N-точечным) преобразованием Фурье как функцией k?

2. Более того, предполагая что N — чётное число, что будет преобразованием Фурье, если $x[n]$ равен 1 при $n = N/2$ вместо $n = 0$?

3. Для каких целых k преобразование Фурье N-точечной функции окна Ханна ненулевое?

4. Какое значение N будет необходимо для анализа Фурье периодического тона в 100 герц (со частотой семплирования 44100 герц) с использованием окна Ханна для получения полного разрешения всех гармоник тона (в смысле, имеющих неперекрывающиеся пики в спектре)?

5. Предположим, что делается N-точечное преобразование Фурье комплексной синусоидой частоты 2.5ω, где $\omega = 2\pi/N$ есть основная частота. Какой процент энергии сигнала лежит в основном лепестке, каналов $k = 2$ и $k = 3$? Если сигнал помещён в окно Ханна, какой тогда процент энергии находится в основном лепестке (который соответствует каналам от 1 до 4)?

Глава 10

Классические формы волн

Вплоть до настоящего момента мы в основном использовали три подхода к синтезу повторяющихся форм волн: аддитивный синтез (Глава 1), синтез волновых таблиц (Глава 2) и вейвшейпинг (Главы 5 и 6). Эта глава вводит четвёртый подход, при котором формы волн строятся явным образом из линейных отрезков с управляемыми конечными точками. Этот подход исторически важен, как минимум, как другие, и был доминирующим в течение периода аналогового синтеза, примерно в 1965–1985 годы. За недостатком лучшего названия мы будем использовать термин *классические формы волн* для обозначения форм волн, составленных из линейных отрезков.

Среди многих других вариантов они включают *пилообразную*, *треугольную* и *прямоугольную* волны, изображённые на рис. 10.1. Наиболее характерными особенностями классических форм волн являются или разрывные скачки (изменения в величине), или углы (изменения наклона). На этом рисунке пилообразная и прямоугольная волны имеют скачки (один за цикл для пилообразной, и два — для прямоугольной) и постоянный наклон и постоянный наклон в других местах (отрицательный для пилообразной волны и нулевой для прямоугольной). Треугольная волна не имеет прерывистых скачков, но её наклон резко изменяется два раза за цикл.

Для эффективного использования классических форм волн полезно понять, как форма волны отражается на её ряде Фурье. (Для его вычисления нам нужны основы, заложенные в Главе 9, — это единственная причина того, что эта глава появилась здесь, а не раньше.) Нам будут также нужны стратегии для цифрового синтеза классических форм волн. Эти формы волн показывают свою восприимчивость к проблемам свертки спектра больше, чем те, которые мы рассматривали раньше, так что нам нужно особенно внимательно отнестись к управлению ими.

В общем, наша стратегия для предсказания и управления свертки спектра — для начала рассмотреть те семплированные формы волн, чей период является целым числом N. Далее, если мы хотим получить форму волны нецелого периода (назовём его, скажем, τ), мы аппроксимируем τ как отношение двух целых чисел N/R. По крайней мере концептуально, мы можем синтезировать далее требуемую

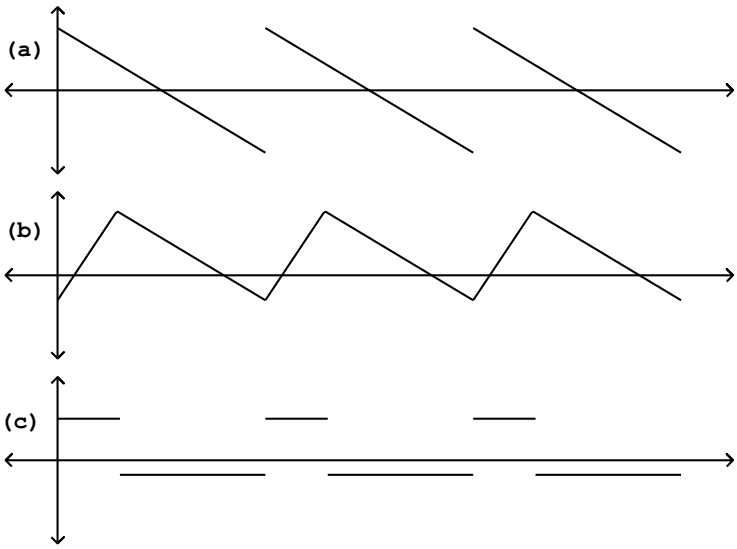


Рис. 10.1: Классические формы волн: (a) пилообразная, (b) треугольная, (c) прямоугольная волны, показанные как функции непрерывной переменной (без семплирования).

форму волны с периодом N и далее взять только каждый R-ый семпл выхода. На этом последнем шаге субдискретизации ("downsampling") происходит свёртывание спектра, и внимательный контроль поможет нам управлять им.

10.1 Симметрии и ряды Фурье

Перед тем как осуществить численный анализ рядов Фурье классических форм волн, мы сделаем паузу, чтобы сделать два полезных наблюдения о симметриях в формах волн и соответствующих им симметриях рядов Фурье. Во-первых, ряд Фурье может состоять только из гармоник с чётными или нечётными номерами; это отражено в симметриях, связывающих форму волны с её смещением на половину цикла. Во-вторых, ряд Фурье может содержать только вещественные или чисто мнимые коэффициенты (соответствующие косинусам или синусам). Это отражено в симметриях, связывающих форму волны с её обращением по времени.

В этом параграфе мы будем предполагать, что изучаемая нами форма волны имеет целый период N, и более того, для простоты, что N является чётным (если это не так, мы можем просто произвести передискретизацию с коэффициентом два). Мы знаем из Главы 9, что любая (вещественная или комплекснозначная) форма волны $X[n]$ может быть записана в виде ряда Фурье (коэффициенты которого мы обозначим $A[k]$):

$$X[n] = A[0] + A[1]U^n + \cdots + A[N-1]U^{(N-1)n},$$

что эквивалентно

$$X[n] = A[0] + A[1](\cos(\omega n) + i\sin(\omega n)) + \cdots$$

$$+A[N-1](\cos(\omega(N-1)n) + i\sin(\omega(N-1)n)),$$

где $\omega = 2\pi/N$ есть основная частота формы волны и

$$U = \cos(\omega) + i\sin(\omega)$$

— комплексное число с единичным модулем и аргументом ω.

Для анализа симметрии первого вида, мы задерживаем сигнал на половину цикла. Поскольку $U^{N/2} = -1$ мы получаем:

$$X[n + N/2] = A[0] - A[1]U^n + A[2]U^{2n} \pm \cdots$$

$$+A[N-2]U^{(N-2)n} - A[N-1]U^{(N-1)n}$$

В действительности задержка в половину цикла меняет знак каждого второго члена в ряде Фурье. Мы комбинируем это выражение с первоначальным рядом двумя различными способами. Пусть X' обозначает их полусумму:

$$X'[n] = \frac{X[n] + X[n + N/2]}{2} = A[0] + A[2]U^{2n} + \cdots + A[N-2]U^{(N-2)n}$$

и X'' — полуразность:

$$X''[n] = \frac{X[n] - X[n + N/2]}{2} = A[1]U^n + A[3]U^{3n} + \cdots + A[N-1]U^{(N-1)n};$$

мы видим, что X' состоит только из гармоник с чётными номерами (включая DC) и X'' — только с нечётными.

Более того, если X оказался равен себе при сдвиге на половину цикла, то есть, если $X[n] = X[n + N/2]$, тогда (рассматривая определения X' и X'') мы получаем $X'[n] = X[n]$ и $X''[n] = 0$. Это означает, что в этом случае $X[n]$ имеет только гармоники с чётными номерами. Действительно, это не должно являться сюрпризом, поскольку в этом случае $X[n]$ должен был бы повторяться каждые $N/2$ семплов, так что его основная частота в два раза больше, чем для периода N.

Точно так же, если $X[n] = -X[n + N/2]$, то X может иметь только гармоники с нечётными номерами. Это позволяет нам просто расщепить любую требуемую форму волны в сумму её гармоник с чётными и нечётными номерами. (Это эквивалентно использованию гребенчатого фильтра для извлечения чётных и нечётных гармоник; см. Главу 7.)

Для вывода соотношений симметрии второго вида мы сравниваем $X[n]$ с его обращением по времени $X[-n]$ (или с $X[N-n]$, поскольку X повторяется каждые N семплов). Ряд Фурье принимает вид:

$$X[-n] = A[0] + A[1](\cos(\omega n) - i\sin(\omega n)) + \cdots$$

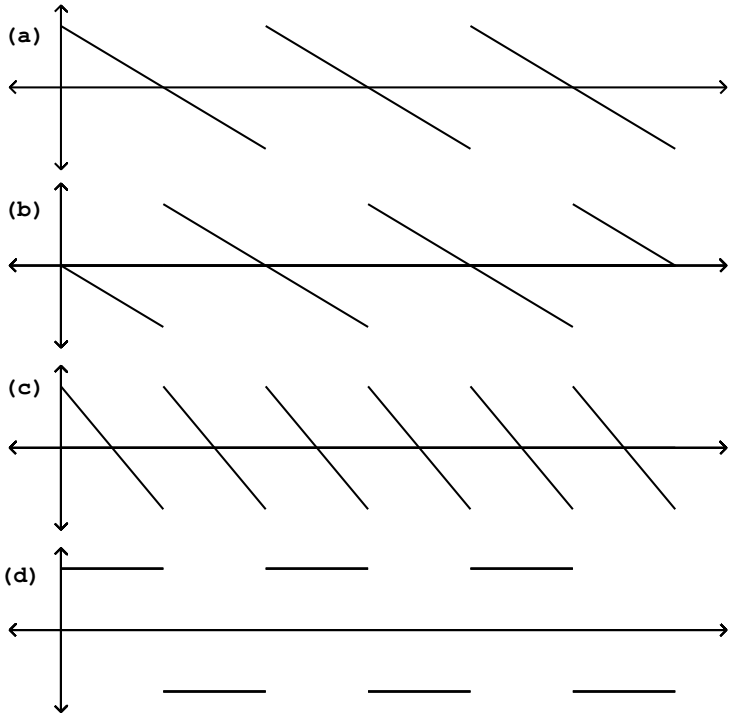


Рис. 10.2: Использование соотношений симметрии для извлечения чётных и нечётных гармоник для пилообразной волны: (a) первоначальная пилообразная волна; (b) сдвинутая на 1/2 цикла; (c) их сумма (другая пилообразная волна с удвоенной частотой); (d) их разность (квадратная волна).

$$+A[N-1](\cos(\omega(N-1)n) - i\sin(\omega(N-1)n))$$

(поскольку функция косинус чётная и синус нечётная). Таким же образом, как выше, мы можем извлечь косинусы, образуя $X'[n]$ как полусумму:

$$X'[n] = \frac{X[n] + X[-n]}{2} = A[0] + A[1]\cos(\omega n) + \cdots + A[N-1]\cos(\omega(N-1)n)$$

и $X''[n]$ как полуразность, делённую на i:

$$X''[n] = \frac{X[n] - X[-n]}{2i} = A[1]\sin(\omega n) + \cdots + A[N-1]\sin(\omega(N-1)n).$$

Таким образом, если $X[n]$ удовлетворяет условию $X[-n] = X[n]$, то ряд Фурье состоит только из косинусов; если же $X[-n] = -X[n]$, тогда он состоит только из синусов; и, как раньше, мы можем разложить любой $X[n]$ (который повторяется каждые N семплов) в сумму двух слагаемых.

10.1.1 Пилообразные волны и симметрия

В качестве примера мы применяем сдвиговую симметрию (чётные и нечётные гармоники) к пилообразной волне. Рисунок 10.2 (часть a) показывает первоначальную пилообразную волну, и часть (b) показывает результат сдвига на половину

цикла. Сумма этих двух волн (часть c) совершает скачек, как только это происходит с одной из двух копий, и создает прямолинейный отрезок во время линейного изменения обеих волн; таким образом, это в результате даёт пилообразную волну с периодом, равным половине первоначального (с удвоенной основной частотой). Взятие разности двух пилообразных волн (часть d) даёт форму волны с нулевым наклоном за исключением мест разрывов. Разрывы, приходящие из первой пилообразной волны, дают прыжки в том же направлении (с отрицательной полуоси на положительную), в отличие от разрывов от второй сдвинутой волны, которые переворачиваются и дают прыжки с положительной полуоси на отрицательную. Результатом является *квадратная волна* — особый вид прямоугольной волны, у которой два образующих отрезка имеют одинаковую длину.

Эта симметрия была использована для замечательного эффекта в конструкции аналоговых синтезаторов Дона Буклы; вместо задействования одного генератора пилы, Букла сконструировал осциллятор, который выводит части, содержащие чётные и нечётные гармоники по отдельности, так что плавное микширование между ними позволяет непрерывное управление соответствующими силами чётных и нечётных гармоник в аналоговой форме волны.

10.2 Препарирование классических форм волн

Среди нескольких выводов, которые мы можем сделать из разложения пилообразной волны на чётные и нечётные гармоники (рисунок 10.2), один состоит в том, что квадратная волна может быть разложена в линейную комбинацию двух пилообразных волн. Мы разовьём эту идею и покажем, как представить любую классическую форму волны, имеющую только скачки (разрывы в значениях), но не углы (разрывы в наклоне), как сумму пилообразных волн различных фаз и амплитуд. Затем мы продвинемся ещё дальше, показав, как генерировать формы волн с углами (или в дополнение, или вместо скачков) с использованием другой элементарной формы волны, которую мы назовём *параболической волной*.

Во-первых, предположим, что форма волны периода N имеет разрывы в j различных точках $L_1, \ldots, L_j$, лежащих внутри цикла от 0 до N, в которых форма волны совершает скачки $d_1, \ldots, d_j$. Отрицательное значение, например, d_1 означало бы, что форма волны совершает скачек от большего к меньшему значению в точке L_1, и положительное значение d_1 означало бы, что совершается прыжок от более низкого к более высокому значению.

Например, рис. 10.3 (часть a) показывает классическую форму волны с двумя скачками: $(L_1, d_1) = (0.3N, -0.3)$ и $(L_2, d_2) = (0.6N, 1.3)$. Части (b) и (c) показывают пилообразные волны, каждая с одним из двух приведённых скачков. Сумма этих пилообразных волн восстанавливает форму волны части (a), за исключением возможного появления константы (DC).

Пилообразная волна с единичным скачком в точке ноль задаётся как

$$s[n] = n/N - 1/2$$

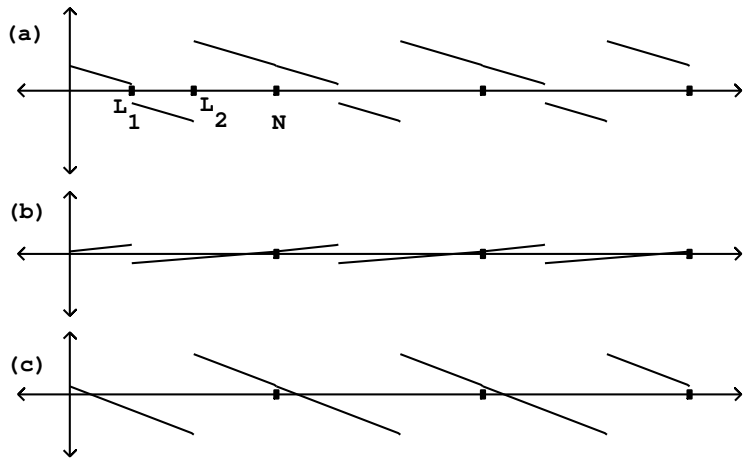


Рис. 10.3: Препарирование формы волны: (a) первоначальная форма волны с двумя разрывами; (b и c) двухкомпонентные пилообразные волны.

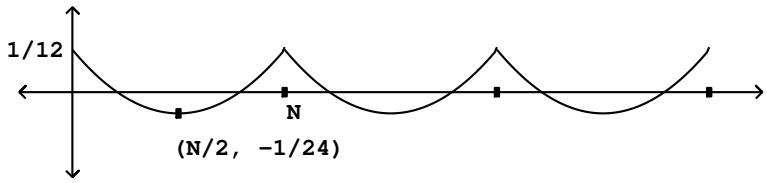


Рис. 10.4: Параболическая волна.

на периоде $0 \leqslant n \leqslant N-1$ и повторяется для остальных значений n. Пилообразная волна со скачком (L,d) задаётся как $s'[n] = ds[n-L]$. Сумма всех составляющих пилообразных волн есть:

$$x[n] = d_1 s[n-L_1] + \cdots + d_j s[n-L_j].$$

Наклоны отрезков формы волны части (a) рисунка все одинаковы и равны сумме наклонов составляющих пилообразных волн:

$$-\frac{d_1 + \cdots + d_j}{N}.$$

Квадратная и прямоугольная волны имеют горизонтальные линейные сегменты (с нулевым наклоном); если такое происходит в произвольной ситуации, скачки должны давать в сумме ноль: $d_1 + \ldots + d_j = 0$.

Для разложения классических форм волн с углами мы используем параболическую волну, которая за один период от 0 до N принимает значения

$$p[n] = \frac{1}{2}\left(\frac{n}{N} - \frac{1}{2}\right)^2 - \frac{1}{24},$$

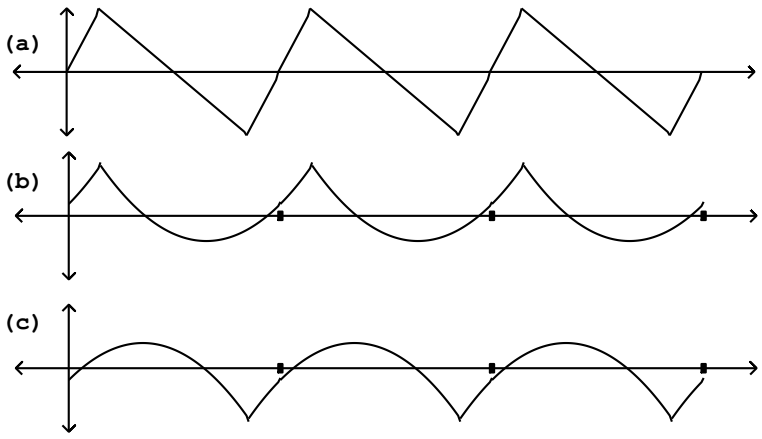


Рис. 10.5: Разложение треугольной волны (часть a) на две параболических волны (b и c).

как показано на рис. 10.4. Это — полином второго порядка (квадратичный) переменной n, заданный так, что он достигает минимума на середине цикла в точке $n = N/2$, компонента DC равна нулю (или, другими словами, среднее значение за цикл формы волны равно нулю), и так, что наклон меняется разрывно на $-1/N$ в начале цикла.

Для построения формы волны с любым желаемым числом углов (предположим, что они расположены в точках $M_1, \ldots, M_l$, а соответствующие изменения наклонов равны $c_1, \ldots, c_l$) мы суммируем необходимые параболические волны:

$$x[n] = -Nc_1 p[n - M_1] - \cdots - Nc_l p[n - M_l].$$

Пример показан графически на рис. 10.5.

10.3 Ряды Фурье элементарных форм волн

В общем, для данной повторяющейся формы волны $X[n]$, мы можем вычислить коэффициенты ряда Фурье $A[k]$ напрямую, вычисляя преобразование Фурье:

$$A[k] = \frac{1}{N}\mathcal{FT}\{X[n]\}(k)$$

$$= \frac{1}{N}\left[X[0] + U^{-k}X[1] + \cdots + U^{-(N-1)k}X[N-1]\right].$$

но прямолинейные выкладки для пилообразной и параболической волн будут требовать целые страницы вычислений (и несколько меньше, если бы мы использовали дифференциальное исчисление). Вместо этого, мы полагаемся на свойства преобразования Фурье для получения связи преобразования Фурье сигнала $x[n]$ с его первой *разностной производной*, определённой как $x[n] - x[n-1]$. Первая

разностная производная параболической волны окажется пилообразной волной, а первая разностная производная пилообразной волны будет достаточно просто вычисляться напрямую и, таким образом, мы получим требуемый ряд Фурье.

В целом, для вычисления силы k-ой гармоники мы будем делать предположение, что N значительно больше, чем k, или равносильно, что k/N ничтожно мало.

Мы начинаем с *формулы сдвига по времени для преобразования фурье* (стр. 262), устанавливающей временной сдвиг в один семпл:

$$\mathcal{FT}\{x[n-1]\} = [\cos(k\omega) - i\sin(k\omega)]\,\mathcal{FT}\{x[n]\}$$
$$\approx (1 - i\omega k)\mathcal{FT}\{x[n]\}.$$

Здесь мы используем предположение, что поскольку N гораздо больше, чем k, то $k\omega = 2\pi k/N$ гораздо меньше единицы, и мы можем провести аппроксимацию:

$$\cos(k\omega) \approx 1\ ,\ \sin(k\omega) \approx k\omega,$$

что является хорошим приближением с небольшой ошибкой порядка $(k/N)^2$. Используя этот результат для вычисления, получаем:

$$\mathcal{FT}\{x[n] - x[n-1]\} \approx i\omega k\mathcal{FT}\{x[n]\}.$$

10.3.1 Пилообразная волна

Во-первых, мы применяем этот подход к пилообразной волне $s[n]$. Для $0 \leqslant n \leqslant N$ получаем:

$$s[n] - s[n-1] = -\frac{1}{N} + \begin{cases} 1, & \text{если } n = 0 \\ 0, & \text{иначе.} \end{cases}$$

Игнорируя постоянное смещение $-1/N$, это даёт *импульс*, равный нулю везде, за исключением одного семпла за цикл. Сумма в преобразовании Фурье содержит только одно слагаемое, и мы получаем:

$$\mathcal{FT}\{s[n] - s[n-1]\}(k) = 1,\ k \neq 0,\ -N < k < N\ .$$

Далее мы применяем формулу для разности, полученную ранее, что даёт:

$$\mathcal{FT}\{s[n]\}(k) \approx \frac{1}{i\omega k} = \frac{-iN}{2\pi k},$$

что справедливо для целых k, малых по сравнению с N, но с $k \neq 0$. (Для получения второго варианта в формуле мы заметили, что $\omega = 2\pi/N$ и $1/i = -i$.)

Такой анализ не даёт нам компоненту DC, поскольку мы должны были бы делить на $k = 0$. Вместо этого, мы можем вычислить член, соответствующий DC, напрямую, как сумму всех точек формы волны: это приблизительно ноль по причине симметрии.

Для получения ряда Фурье в терминах привычных действительнозначных синуса и косинуса, мы комбинируем соответствующие члены для отрицательных и положительных значений k. Первая гармоника ($k = \pm 1$) есть

$$\begin{aligned}
&\frac{1}{N}\left[\mathcal{FT}\{s[n]\}(1)\cdot U^n + \mathcal{FT}\{s[n]\}(-1)\cdot U^{-n}\right] \\
&\approx \frac{-i}{2\pi}\left[U^n - U^{-n}\right] \\
&= \frac{\sin(\omega n)}{\pi}
\end{aligned}$$

и, сходным образом, k-ая гармоника есть

$$\frac{\sin(k\omega n)}{k\pi}$$

$$s[n] \approx \frac{1}{\pi}\left[\sin(\omega n) + \frac{\sin(2\omega n)}{2} + \frac{\sin(3\omega n)}{3} + \cdots\right].$$

10.3.2 Параболическая волна

Тот же анализ, с некоторыми отличиями в знаке и нормализации, работает для параболических волн. Во-первых, вычислим разностную производную:

$$\begin{aligned}
p[n] - p[n-1] &= \frac{\left(\frac{n}{N} - \frac{1}{2}\right)^2 - \left(\frac{n-1}{N} - \frac{1}{2}\right)^2}{2} \\
&= \frac{\left(\frac{n}{N} - \frac{N}{2N}\right)^2 - \left(\frac{n}{N} - \frac{N-2}{2N}\right)^2}{2} \\
&= \frac{\frac{2n}{N^2} - \frac{1}{N} + \frac{1}{N^2}}{2} \\
&\approx -s[n]/N.
\end{aligned}$$

Таким образом (опять при $k \neq 0$, малом по сравнению с N), мы получаем:

$$\begin{aligned}
\mathcal{FT}\{p[n]\}(k) &\approx \frac{-1}{N}\cdot\frac{-iN}{2\pi k}\cdot\mathcal{FT}\{s[n]\}(k) \\
&\approx \frac{-1}{N}\cdot\frac{-iN}{2\pi k}\cdot\frac{-iN}{2\pi k} \\
&= \frac{N}{4\pi^2 k^2}
\end{aligned}$$

и, как и перед этим, мы получаем ряд Фурье:

$$p[n] \approx \frac{1}{2\pi^2}\left[\cos(\omega n) + \frac{\cos(2\omega n)}{4} + \frac{\cos(3\omega n)}{9} + \cdots\right]$$

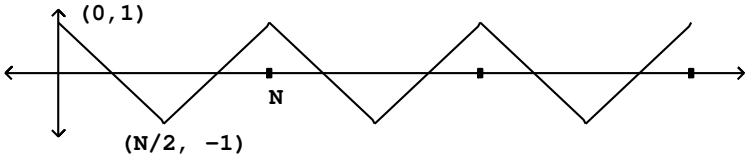


Рис. 10.6: Симметричная треугольная волна, полученная суперпозицией параболических волн с парами (M,c), равными (0,8) и $(N/2, -8)$.

10.3.3 Квадратная и симметричная треугольная волны

Для того чтобы увидеть, как получить ряд Фурье для классических форм волн в общем, рассмотрим квадратную волну,

$$x[n] = s[n] - s[n - \frac{N}{2}],$$

равную 1/2 на первой половине цикла ($0 < n < N/2$) и $-1/2$ на второй. Подставляя дважды ряд Фурье для $s[n]$, мы получаем искомый ряд Фурье:

$$\begin{aligned} x[n] \approx \frac{1}{\pi} \Bigg[& \sin(\omega n) + \frac{\sin(2\omega n)}{2} + \frac{\sin(3\omega n)}{3} + \cdots \\ & -\sin(\omega n) + \frac{\sin(2\omega n)}{2} - \frac{\sin(3\omega n)}{3} \pm \cdots \Bigg] \\ = \frac{2}{\pi} \Bigg[& \sin(\omega n) + \frac{\sin(3\omega n)}{3} + \frac{\sin(5\omega n)}{5} + \cdots \Bigg] \end{aligned}$$

Симметричная треугольная волна (рисунок 10.6), заданная как

$$x[n] = 8p[n] - 8p[n - \frac{N}{2}]$$

аналогичным образом приводит к

$$x[n] \approx \frac{8}{\pi^2} \left[\cos(\omega n) + \frac{\cos(3\omega n)}{9} + \frac{\cos(5\omega n)}{25} + \cdots \right].$$

10.3.4 Общая (несимметричная) треугольная волна

Общая форма несимметричной треугольной волны показана на рис. 10.7. Здесь мы разместили цикл таким образом, что, во-первых, компонента DC равна нулю (так что два угла имеют одинаковые, но противоположные по знаку высоты) и, во-вторых, так что средняя точка наиболее короткого сегмента приходится на точку $(0, 0)$.

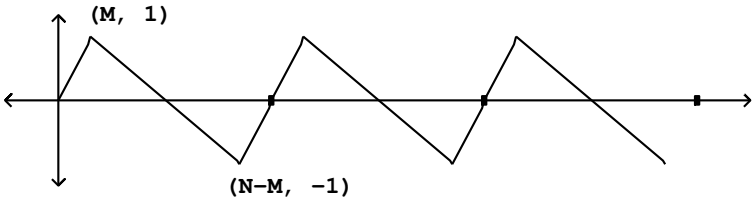


Рис. 10.7: Несимметричная треугольная волна с вершинами в (M,1) и (N-M, -1).

Два линейных сегмента имеют наклоны, равные $1/M$ и $-2/(N-2M)$, так что разложение на параболические волны даётся выражением:

$$x[n] = \frac{N^2}{MN - 2M^2}(p[n-M] - p[n+M])$$

(здесь мы используем периодичность $p[n]$, которая позволяет заменить $p[n-(N-M)]$ на $p[n+M]$).

Наиболее общий способ действия с линейными комбинациями элементарных (параболических и/или пилообразных) волн состоит в возвращении к комплексным рядам Фурье, как мы делали при нахождении рядов для самих элементарных волн. Но в этом частном случае мы можем использовать тригонометрическое тождество для избежания дополнительной работы при прямом и обратном преобразованиях. Во-первых, мы подставляем действительнозначное преобразование Фурье:

$$x[n] = \frac{N^2}{2\pi^2(MN - 2M^2)}\Big[\cos(\omega(n-M)) - \cos(\omega(n+M))$$
$$+\frac{\cos(2\omega(n-M)) - \cos(2\omega(n+M))}{4} + \cdots\Big].$$

Сейчас мы используем тождество

$$\cos(a) - \cos(b) = 2\sin(\frac{b-a}{2})\sin(\frac{a+b}{2}),$$

так что, например,

$$\cos(\omega(n-M)) - \cos(\omega(n+M)) = 2\sin(2\pi M/N)\sin(\omega n).$$

(Здесь опять мы использовали определение $\omega = 2\pi/N$.) Это является упрощением, поскольку первый синус в произведении не зависит от n; это только амплитудный коэффициент. Применение тождества ко всем членам ряда для $x[n]$ даёт:

$$x[n] = a[1]\sin(\omega n) + a[2]\sin(2\omega n) + \cdots,$$

где амплитуды компонент даются выражением

$$a[k] = \frac{1}{\pi^2(M/N - 2(M/N)^2)} \cdot \frac{\sin(2\pi kM/N)}{k^2}$$

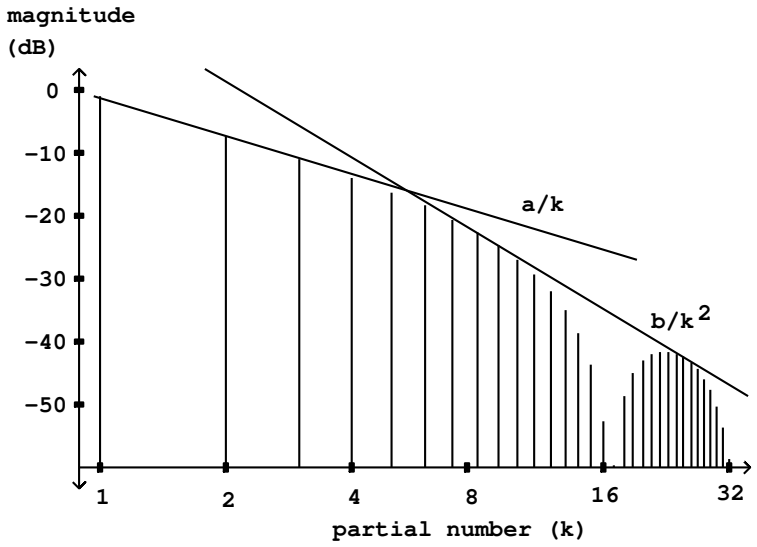


Рис. 10.8: Спектр абсолютных значений треугольной волны с $M/N = 0.33$. Два линейных сегмента показывают поведение $1/k$ и $1/k^2$ на низких и высоких частотах.

Заметим, что результат не зависит от величин M и N по отдельности, а только от их отношения M/N (это не является сюрпризом, поскольку форма волны зависит от отношения). Если мы рассматриваем маленькие значения k:

$$k < \frac{1}{4M/N}$$

аргумент функции sin меньше $\pi/2$ и, используя аппроксимацию $\sin(\theta) \approx \theta$, мы находим, что $a[k]$ убывает как $1/k$, так же как гармоники пилообразной волны. Но для больших значений k синус колеблется между -1 и 1, так что амплитуды спадают иррегулярно, как $1/k^2$.

Рисунок 10.8 показывает силу гармоник при M/N, установленном на 0.03; здесь наш прогноз состоит в том, что зависимость $1/k$ должна продолжаться до $k \approx 1/(4 \cdot 0.03) \approx 8.5$, в приблизительном соответствии с рисунком.

Другой способ увидеть, почему гармоники должны вести себя как $1/k$ для маленьких значений k, и как $1/k^2$ — впоследствии, состоит в сравнении периода данной гармоники с длиной короткого сегмента $2M$. Для гармоник, имеющих номера ниже $N/4M$, период равен как минимум удвоенной длине короткого сегмента и на этом масштабе форма волны почти не отличима от пилообразной волны. Для гармоник, имеющих номера, превышающие $N/2M$, два угла треугольной волны разделены как минимум одним периодом, и на этих более высоких частотах два угла (каждый с зависимостью от частоты $1/k^2$) отделены друг от друга. На рисунке выемка на 17-й гармонике возникает на длине волны $N/2M \approx 1/17$, при которой два угла разделены на один цикл; поскольку углы противоположны по знаку, они уничтожают друг друга.

10.4 Предсказание и управление сверткой спектра

Теперь мы вернемся к реальной ситуации, при которой период формы волны нельзя предположить имеющим произвольную длину и целое значение. Предположим (для определённости), что мы хотим синтезировать тоны на 440 герц (ля первой октавы), и что мы используем частоту семплирования 44100 герц, так что период составляет около 100.25 семплов. Теоретически, для данной очень высокой частоты семплирования, мы должны ожидать, что 50-я гармоника имеет величину 1/50, по сравнению с основной частотой, и её частота составляет приблизительно 20 кГц. Если мы семплируем эту форму волны с частотой ниже 44100 герц, тогда гармоники, превышающие эту частоту, будут накладываться друг на друга, как описано в §3.1. Относительная сила сворачивающихся гармоник будет порядка −32 децибел — достаточно слышимой. Если основная частота далее увеличивается, большее число более громких гармоник достигает частоты Найквиста (половина скорости семплирования) и начинает свёртываться.

Проблемы свёртывания гораздо меньше декларируются для форм волн только с углами (вместо скачков) в виду более быстрого спадания гармоник с высокими частотами; например, симметричная треугольная волна на 440 герц спала бы в два раза или на −64 децибела. Однако в целом формы волн с разрывами являются лучшей отправной точкой для субтрактивного синтеза (наиболее популярной классической техники). На тот случай, если у вас была такая надежда, субтрактивная фильтрация не сможет убрать свертку спектра, если она уже присутствует в аудиосигнале.

10.4.1 Пересемплирование

В качестве первой линии защиты против свертки спектра мы можем синтезировать форму волны с максимально возможной частотой семплирования, применить низкочастотный фильтр с частотой среза, установленной на частоту Найквиста (для первоначальной частоты семплирования) и затем пересемплировать. Например, в сценарии выше (частота семплирования — 44100, тон — 440 герц) мы можем генерировать пилообразную волну со скоростью семплов $16 \cdot 44100 = 705600$ герц. Нам нужно только позаботиться о частотах, не превышающих $705600 - 20000 = 685600$ герц (так что они свёртываются в слышимые частоты; свёртывание до ультразвуковых частот обычно нас не касается), так первая проблемная гармоника есть $685600/440 = 1558$, чья амплитуда равна −64 дб по отношению к амплитуде основной частоты.

Это затухание падает на 6 дб при увеличении частоты на октаву, так что пилообразная волна на 10 кГц порадует только 37 дб падения по отношению к основной частоте самой громкой свернувшейся гармоники. С другой стороны, увеличение частоты семплирования на дополнительный коэффициент два уменьшает свёртывание на такое же значение. Если мы действительно хотим достичь 60 децибел подавления свёртывания (как-либо до 10 кГц основной частоты) мы будем должны пересемплировать с коэффициентом 256, с частотой семплирования около

11 миллионов герц.

10.4.2 “Коварные” треугольные волны

Для низких основных частот пересемплирование является простым способом достижения адекватной защиты от свертки спектра. Если мы хотим иметь более высокие частоты, нам потребуется подход сложнее. Одна из возможностей состоит в замене разрывов с помощью линейных функций или, другими словами, замена составляющих пилообразных волн на треугольные волны, что проделывалось в §10.3.4 с величинами M/N, достаточно малыми, чтобы результат звучал как пилообразная волна, но достаточно большими, чтобы держать свертку спектра под контролем.

Возвращаясь к рис. 10.8, предположим, например, что мы имитируем пилообразную волну при помощи треугольной с M, равным двум семплам, так что первая впадина[1] попадает на частоту Найквиста. Гармоники выше этой впадины (17-ая гармоника на рисунке) будут свёртываться; наихудшая из них приблизительно на 40 дб меньше основной частоты. С другой стороны, сила гармоник начинает спадать быстрее, чем для тех гармоник настоящей пилообразной волны, которые находятся в районе половины частоты Найквиста. Это приемлемо в некоторых, но не во всех ситуациях.

Стратегия использования треугольных волн может быть объединена с пересемплированием для дальнейшего улучшения ситуации. В контексте рис. 10.8 предположим опять, что мы пересемплируем с коэффициентом 4 и устанавливаем первую впадину на первоначальную скорость семплирования. Гармоники до частоты Найквиста (8-ая по отношению к основной частоте гармоника, показанная на рисунке) очень хорошо соответствуют гармоникам настоящей пилообразной волны. Свертка спектра происходит только для гармоники с номером 48 и даёт результат на 52 дб ниже основной частоты. Это общее поведение остаётся для любой основной частоты, вплоть до приблизительно одной четверти частоты семплирования (после которой M превышает $N/2$). Установка частоты выемки на первоначальную частоту семплирования эквивалентно установке сегмента длины 2M на один семпл (с первоначальной частотой семплирования).

10.4.3 Склейка переходов

С точки зрения, рассматриваемой в этой главе, энергия спектральных составляющих классических волн может быть отнесена полностью их скачкам и углам. Это, конечно же, искусственный подход: в действительности энергия получается из формы волны в целом. При расчете спектра классических волн скачки и углы используются в качестве основы для рассчетлв, и это возможно потому, что вся форма волны определяется их положением и величиной.

[1] 16-я гармоника на рисунке. (*прим. перев.*)

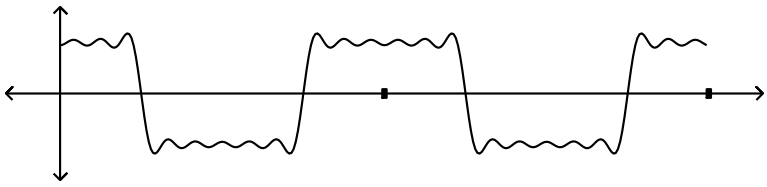


Рис. 10.9: Квадратная волна, ограниченная полосой с гармониками 1, 3, 5, 7, 9 и 11. Это может приблизительно рассматриваться как ряд ступенчатой функции с ограниченным спектром, записанный от начала до конца.

Используя этот приём дальше, можно попытаться решить задачу создания вариантов классических форм волн с ограниченным спектром с помощью создания ограниченных по спектру вариантов скачков и углов. Поскольку скачки представляют более серьёзную опасность свертки спектра, мы сосредоточимся здесь на них, хотя описанный здесь подход отлично работает также для углов.

Для создания ступенчатой функции с ограниченным спектром всё, что нам нужно сделать — это складывать так много компонентов Фурье квадратной волны, сколько мы хотим, чтобы получить ступенчатую функцию для любого из скачков. Рисунок 10.9 показывает частичную сумму Фурье, соответствующую квадратной волне, в которой использованы гармоники 1, 3, 5, 7, 9 и 11. В качестве частоты среза можно взять 12ω (если ω — основная частота).

Если бы мы удвоили период квадратной волны, для получения той же среза частоты мы бы могли сложить опять много гармоник, например, вплоть до 23-й. Постоянно расширяя этот процесс, мы увидели бы в итоге идеальную ступенчатую функцию с ограниченной полосой два раза (за произвольно длинный) период.

На практике мы можем неплохо справиться, пользуясь всего двумя первыми гармониками (на основной и утроенной основной частоте). Рисунок 10.10 (часть a) показывает аппроксимацию квадратной волны, состоящую из двух гармоник. Частота среза равна четырём основным частотам; таким образом, если период формы волны составляет восемь семплов, отрезание происходит на частоте Найквиста. Часть (b) рисунка показывает, как мы могли бы использовать эту ступенчатую функцию для синтезирования приблизительно квадратной волны с удвоенным периодом. Если частота отрезания равна частоте Найквиста, период формы волны части (b) составляет 16 семплов. Каждый переход занимает 4 семпла, поскольку квадратная волна с ограниченной полосой имеет период в восемь семплов.

Мы можем сделать пилообразную волну с ограниченным спектром, связывая четырёхсемпловые переходы линейными отрезками, так что конец получающейся функции гладко сшивается со следующим концом, как показано в части (c) рисунка. Имеется один переход за период, так что период должен состоять как минимум из четырёх семплов; наиболее высокая основная частота, которую мы можем синтезировать этим путём, равна половине частоты Найквиста. Для этой или более низкой основной частоты все результаты свертки спектра оказываются как минимум на 60 дб тише основной частоты.

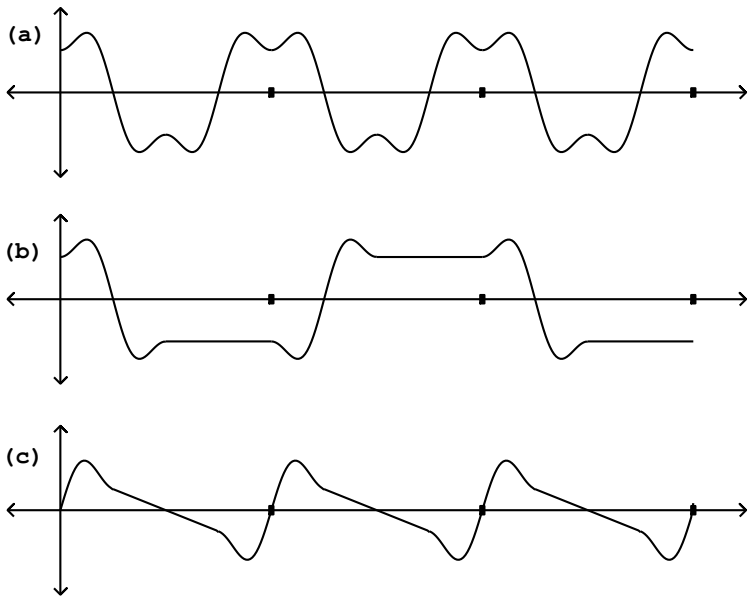


Рис. 10.10: Растягивание квадратной волны с ограниченной полосой: (a) первоначальная форма волны; (b) после склейки переходов горизонтальными отрезками; (c) использование того же ступенчатого перехода для пилообразной волны.

Рисунок 10.11 показывает, как генерировать пилообразную волну с плавными переходами. Двумя параметрами являются: f — основная частота, и b — предел полосы, который предполагается, как минимум, не меньше f. Мы начнём с цифровой пилообразной волны (фазора), имеющей диапазон от -0.5 до 0.5 по амплитуде. Переход будет иметь место в середине цикла, когда фазор пересекает ноль. Волновая таблица проходится за постоянный промежуток времени $1/b$ независимо от f. Просмотр таблицы берётся без зацикливания, так что входы вне диапазона выводятся или как -0.5, или как 0.5.

В конце цикла фазор совершает разрывный скачёк от -0.5 до 0.5, но выход переходной таблицы совершает равный ему скачёк, но с противоположным знаком, так что результат является непрерывным. Во время той части формы волны, на которой переходная таблица читается в одной или другой концевых точках, выход описывает отрезок прямой линии.

10.5 Примеры

10.5.1 Комбинирование пилообразных волн

Пример J01.even.odd.pd (рис. 10.12, часть a) показывает, как комбинировать пилообразные волны парами для извлечения чётных и нечётных гармоник. Получающиеся формы волн показаны на рис. 10.3. Пример J02.trapezoids.pd (часть b рисунка) демонстрирует комбинирование трёх пилообразных волн с произвольными

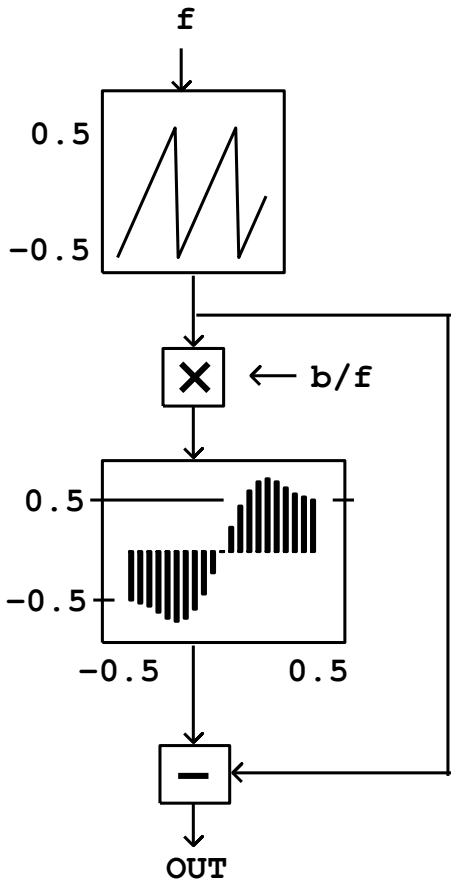


Рис. 10.11: Блок-схема для создания пилообразной волны с плавными переходами.

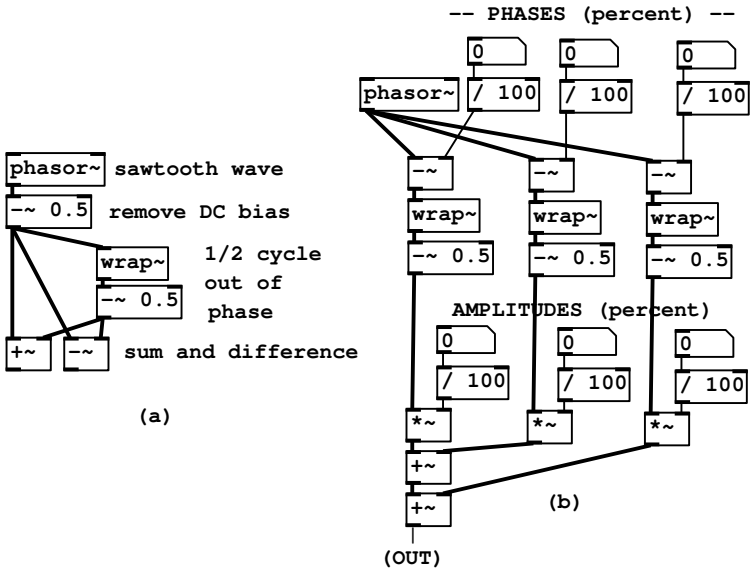


Рис. 10.12: Комбинирование пилообразных волн: (a) сложение и вычитание пилообразных волн, не совпадающих на половину цикла по фазе, для извлечения чётных и нечётных гармоник; (b) комбинирование трёх пилообразных волн с произвольными амплитудами и фазами.

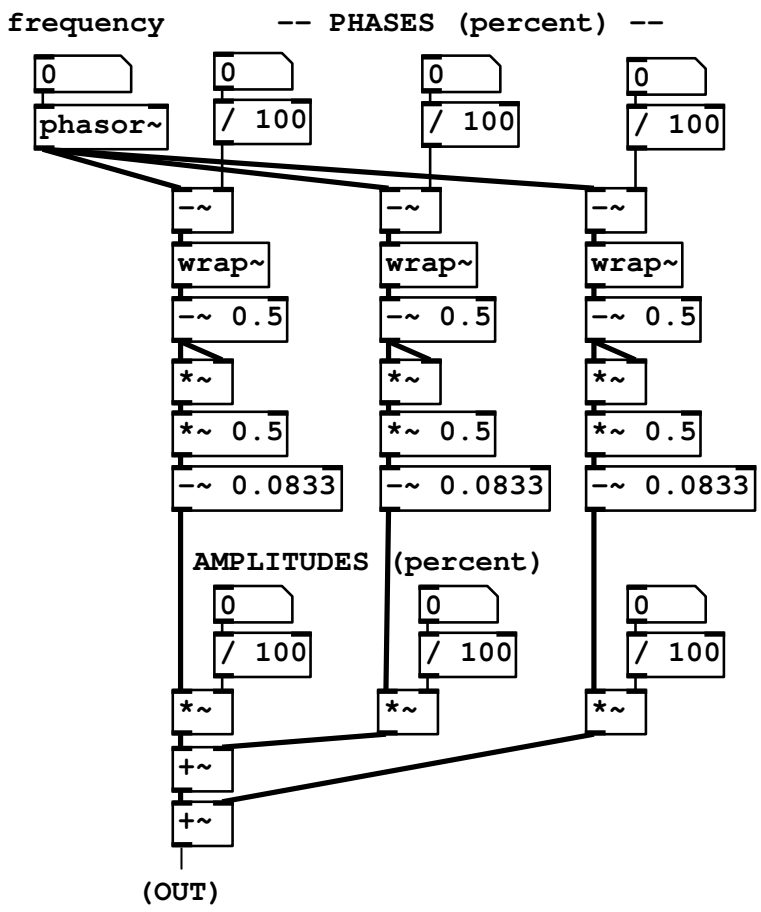


Рис. 10.13: Комбинирование пилообразных волн: (a) сложение и вычитание пилообразных волн, не совпадающих на половину цикла по фазе, для извлечения чётных и нечётных гармоник; (b) комбинирование трёх пилообразных волн с произвольными амплитудами и фазами.

амплитудами и фазами; получающаяся классическая форма волны имеет до трёх скачков и не имеет углов. Три линейных отрезка горизонтальны, если сумма всех трёх скачков равна нулю; в противном случае, отрезки наклонены для компенсации несбалансированных скачков, так что результат повторяется от одного периода к следующему.

Пример J03.pulse.width.mod.pd (не показан) комбинирует две пилообразные волны противоположных знаков со слегка разными частотами, так что относительная фаза меняется непрерывно. Их сумма представляет из себя прямоугольную волну, чья ширина варьируется по времени. Это известно как *широтно-импульсная модуляция* (“ШИМ”).

Пример J04.corners.pd (рис. 10.13) показывает, как складывать параболические волны для создания комбинированных форм волн с тремя углами. Каждая параболическая волна вычисляется из пилообразной волны (пробегающей от -0.5 до 0.5) путём возведения её в квадрат, умножения на 0.5 и вычитания компоненты

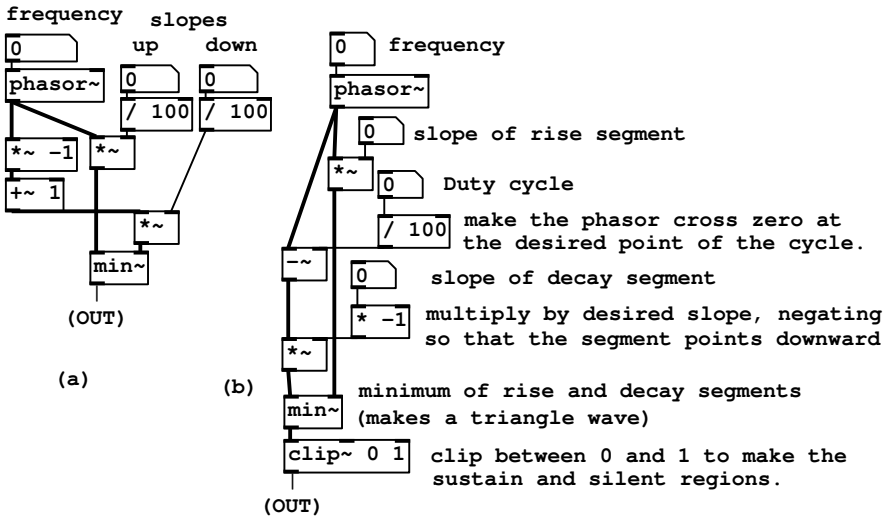


Рис. 10.14: Альтернативные методы получения форм волн с углами: (a) треугольная волна как минимум двух линейных сегментов; (b) отрезание треугольной волны для создания “огибающей”.

DC $-1/12$ или -0.08333. Патч комбинирует три таких параболических волны с управляемыми амплитудами и фазами. Если амплитуды дают в сумме ноль, получающаяся форма волны состоит из линейных сегментов с углами расположенными согласно трём фазам и изменяет наклон в соответствии с тремя амплитудами.

10.5.2 Получение пилообразных волн с ограниченным спектром

Пример J05.triangle.pd (рис. 10.14, часть a) показывает простой путь для создания треугольной волны, у которой определены только наклон восходящего и падающего сегментов. Фазор обеспечивает растущий фрагмент (его амплитуда является наклоном), и, тот же фазор, вычтенный из единицы, даёт убывающий фрагмент. Минимум двух линейных функций соответствует растущему фазору вплоть до их пересечения и далее соответствует падающему фазору, возвращающемуся в ноль в конце цикла.

Треугольная волна может быть отрезана сверху и снизу трапециевидной волны, которая может быть использована как аудиочастотный импульс или, при низкой основной частоте, как повторяющаяся огибающая ASR (attack/sustain/release — атака/удерживание/завершение). Патч J06.enveloping.pd (рис. 10.14, часть b) демонстрирует это. Тот же растущий фрагмент используется как в предыдущем примере, и убывающий фрагмент отличается только тем, что его фаза устанавливается так, что он спадает до нуля в управляемой точке (не обязательно в конце цикла как раньше). Объект clip~ предотвращает рост выше 1 (так что, если пересечение двух сегментов выше единицы, мы получаем горизонтальный сегмент “удерживания”), а

также спад ниже нуля, так что если спадающий фрагмент достигает нуля, выход будет нулём всю оставшуюся часть цикла.

Пример J07.oversampling.pd показывает, как использовать пересемплирование для уменьшения свёртывания при использовании объекта phasor~ в качестве пилообразной аудиоволны. Сабпатч, работающий в 16 раз быстрее основной частоты семплирования, содержит объект phasor~ и фильтр Баттерворта с тремя полюсами и тремя нулями для уменьшения амплитуд гармоник выше частоты Найквиста родительского патча (работающего с первоначальной частотой семплирования), так что спектр выхода не будет свёртываться при пересемплировании объектом outlet~. Пример J08.classicsynth.pd демонстрирует использование пересемплирующих фазоров как генераторов сигналов для создания имитации классического синтезатора, производящего субтрактивный синтез.

Пример J09.bandlimited.pd показывает, как использовать склейку переходов в качестве альтернативного способа генерирования пилообразной волны с контролируемой сверткой спектра. Этот метод проще (и, обыкновенно, менее затратный для вычисления), чем метод пересемплирования. С другой стороны, такой метод зависит от внутреннего использования частоты обратной основной в качестве аудиосигнала (для управления амплитудой считывающего сигнала, который считывает переходную таблицу), и, таким же образом, как для техники PAF Главы 6, необходимо позаботиться, чтобы избежать щелчков, если основная частота меняется разрывно.

Упражнения

1. Объект phasor~ имеет частоту 441 герц (с частотой семплирования 44100). Какова амплитуда компоненты DC? Основной частоты? Гармоники на 22050 герц (какие из приведённых гармоник свёртываются)?

2. Квадратная волна осциллирует между 1 и -1. Какова у нее амплитуда RMS?

3. В §10.3 квадратная волна была представлена как нечётная форма волны, чей ряд Фурье состоит из синусов (но не из косинусов). Если квадратная волна опережается на 1/8 цикла по фазе, так что она появляется в виде чётной функции, каким станет её ряд Фурье?

4. Значение прямоугольной волны равно 1 для 1/4 цикла и ноль для 3/4 цикла. Каковы силы её гармоник с частотами, равными произведению 0, 1, 2, 3 и 4 на основную частоту.

5. Чему равно $1 + 1/9 + 1/25 + 1/49 + 1/81 + \ldots$?

Словарь терминов

ADSR ADSR-огибающая — функция, описывающая изменения какого-либо параметра во времени, используемая в синтезаторах звука. ADSR-огибающая позволяет описывать подобные изменения с помощью небольшого числа параметров, описывающих четыре разные стадии огибающей. Название ADSR является сокращением от названий каждой из этих стадий (**A**ttack-**D**ecay-**S**ustain-**R**elease). 86, 87, 91–93, 100–103, 245

Csound язык программирования для работы со звуком. Создатель — Барри Веркоу (Barry Vercoe), сотрудник Массачусетского технологического института и композитор. xi, 8, 15, 61, 63, 95, 156, 209

DC Direct Current, постоянный электрический ток. 114, 122, 129, 142, 145, 197, 223, 231, 240, 247, 250, 252–254, 258, 260, 263, 291, 293, 295, 296, 298

FM Frequency Modulation, Частотная модуляция — вид модуляции, при которой модулирующий сигнал управляет частотой несущего колебания. 126–128, 138, 153

IRCAM Institut de Recherche et Coordination Acoustique/Musique, исследовательская организация в Париже, созданная по поручению Жоржа Помпиду композитором Пьером Булезом для современных музыкальных и музыковедческих исследований. 153

Max/MSP язык визуального программирования музыки и мультимедиа. xi, 8, 15, 63, 95, 156

MIDI Musical Instrument Digital Interface — стандарт на формат обмена данными между электронными музыкальными инструментами. 7, 25, 63, 98, 131

RMS (Root Mean Square), Среднее квадратичное — число s, равное квадратному корню из среднего арифметического квадратов данных чисел a_1,a_2,...,a_n

$$s = \sqrt{\frac{a_1^2 + a_2^2 + \ldots + a_n^2}{n}}$$

. 3, 20, 25, 28, 46, 82, 140, 196

SuperCollider виртуальная среда и язык программирования для аудиосинтеза в реальном времени. 15

VCA (Voltage Controlled Amplifier) — усилитель, управляемый напряжением. 11, 116

VCF (Voltage Controlled Filter) — фильтр, управляемый напряжением. 245

VCO (Voltage Controlled Oscillator) — осциллятор, управляемый электрическим напряжением. 11

Предметный указатель

MIDI, 7

аддитивный синтез, 13
амплитуда, 1, 3
вейвшейпинг, 26
децибел, 4
окно, 3
пиковая амплитуда, 3

семпл, 1
синусоида, 1
среднеквадратичная амплитуда, 3
юнит-генератор, 8

Литература

[Bal03] Mark Ballora. *Essentials of Music Technology.* Prentice Hall, Upper Saddle River, New Jersey, 2003.

[Ble01] Barry A. Blesser. An interdisciplinary synthesis of reverberation viewpoints. *Journal of The Audio Engineering Society*, 49:867–903, 2001.

[Bou00] Richard Boulanger, editor. *The Csound book.* MIT Press, Cambridge, Massachusetts, 2000.

[Bru77] Marc Le Brun. A derivation of the spectrum of fm with a complex modulating wave. *Computer Music Journal*, 1(4):51–52, 1977.

[Bru79] Marc Le Brun. Digital waveshaping synthesis. *Journal of The Audio Engineering Society*, 27:250–266, 1979.

[Cha80] Hal Chamberlin. *Musical applications of microprocessors.* Hayden, Rochelle Park, N. J., 1980.

[Cho73] John M. Chowning. The synthesis of complex audio spectra by means of frequency modulation. pages 526–534, 1973.

[Cho89] John Chowning. Frequency modulation synthesis of the singing voice. *Current Directions in Computer Music Research*, pages 57–64, 1989.

[DJ85] C. Dodge and T.A. Jerse. *Computer Music: Synthesis, Composition, and Performance.* Longman music series. Schirmer Books, 1985.

[GM77] John M. Grey and James A. Moorer. Perceptual evaluations of synthesized musical instrument tones. *The Journal of the Acoustical Society of America*, 62(2):454–462, 08 1977.

[Har87] W. M. Hartmann. Digital waveform generation by fractional addressing. *The Journal of the Acoustical Society of America*, 82(6):1883–1891, 12 1987.

[KS83] Kevin Karplus and Alex Strong. Digital synthesis of plucked-string and drum timbres. *Computer Music Journal*, 7(2):43–55, 1983.

[LD97] Jean Laroche and M. Dolson. Phase-vocoder: about this phasiness business. *Proceedings of 1997 Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 4 pp.–, 1997.

[Mat69] Max V. Mathews. *The Technology of Computer Music.* MIT Press, Cambridge, Massachusetts, 1969.

[Moo90] F.R. Moore. *Elements of Computer Music.* Prentice Hall, 1990.

[PB87] T.W. Parks and C.S. Burrus. *Digital Filter Design.* Topics in Digital Signal Processing. Wiley, 1987.

[Puc95a] Miller Puckette. Formant-based audio synthesis using nonlinear distortion. *J. Audio Eng. Soc*, 43(1/2):40–47, 1995.

[Puc95b] Miller Puckette. Phase-locked vocoder. *Proceedings of 1995 Workshop on Applications of Signal Processing to Audio and Accoustics*, pages 222–225, 1995.

[Puc01] Miller S. Puckette. *Synthesizing Sounds with Specified, Time-Varying Spectra.* Michigan Publishing, 2001.

[Puc05] Miller Puckette. Phase bashing for sample-based formant synthesis. In *International Conference on Mathematics and Computing*, 2005.

[Reg93] Phillip A. Regalia. *Special filter design.* Wiley, 1993.

[RM69] Jean-Claude Risset and Max V. Mathews. Analysis of musical-instrument tones. *Physics Today*, 22(2):23–30, 02 1969.

[RMW+02] Thomas D. Rossing, Richard F. Moore, Paul A. Wheeler, David Miles Huber, and Robert E. Runstein. *The Science of Sound.* Addison Wesley, San Francisco, third edition, 2002.

[Roa01] Curtis Roads. *Microsound.* MIT Press, Cambridge, Massachusetts, 2001.

[Sch77] Bill Schottstaedt. The simulation of natural instrument tones using frequency modulation with a complex modulating wave. *Computer Music Journal*, 1(4):46–50, 1977.

[Smi03] Julius O. Smith. *Mathematics of the discrete Fourier transform (DFT) : with audio applications.* BookSurge, North Charleston, 2003.

[Ste96] K. Steiglitz. *A DSP Primer: With Applications to Digital Audio and Computer Music.* Addison-Wesley Pub., 1996.

[Str85] John Strawn. Digital audio signal processing. *William Kaufmann, Los Altos, California*, 1985.

[Str95] Allen Strange. *Electronic music: systems, techniques, and controls.* W. C. Brown, Dubuque, Iowa, 1995.