высоких частот упр. 4.10, используя значения $B=68$, 84, 92, 96, 98 и 99 Гц. Сравните новые результаты с теми, которые были получены в упр. 4.10, и прокомментируйте возникшую картину.

4.12. Проведите трансляцию программы с рис. 4.13 для получения полосно-непропускающего фильтра, имеющего шесть рекурсивных коэффициентов, полагая $T=0.005$. Получите (численно и графически) значения коэффициента усиления и фазы в точках $f=0, 1, \ldots, F$ Гц, полагая

a) $f_c=50$ Гц и $B=1$, 2, 5, 10, 25 Гц;

b) $B=2$ Гц и $f_c=2$, 4, 10 и 20 Гц.

Прокомментируйте результаты.

4.13. Переделайте программу, использованную в упр. 4.12, для случая заграждающего фильтра и проделайте те же вычисления.

4.14. Проведите трансляцию программы LPTB с рис. 4.15 для вычисления коэффициентов фильтров низких частот Баттеруорта. Повторите упр. 4.10, используя вместо программы LPSB программу LPTB. Сравните результаты этих двух упражнений и прокомментируйте их.

4.15. a) Напишите программу вычисления коэффициента усиления и фазы передаточной функции симметричного ИКО-фильтра.

b) Напишите программу для вычисления весов фильтра низких частот, в которой

$$b_k=\frac{\sin(2\pi BkT)}{\pi k}, \qquad k=0, \ldots, M.$$

c) Используя программы из пп. a) и b), составьте программу, которая строит графики коэффициента усиления, полагая $T=0.005$, $B=32$, 16, 8, 4, 2, 1 Гц и $M=8$, 16, 32, 64. Прокомментируйте результаты. Сравните их также с результатами, полученными в упр. 4.10.

4.16. Программу LPSPBG с рис. 4.17, реализующую алгоритм Поттера, Бикфорда и Глейза, примените в упр. 4.10 и сравните результаты, которые получатся в обоих случаях.

Глава 5

ПРАКТИЧЕСКИЕ АСПЕКТЫ ЦИФРОВОЙ ФИЛЬТРАЦИИ

5.1. ВВЕДЕНИЕ

В этой главе затронут круг вопросов, связанных с использованием на практике методов цифровой фильтрации. Среди них будут рассмотрены шум и искажение, неустойчивость, различные виды программной записи алгоритмов фильтрации, сглаживание данных и децимация.

Выбор конкретного фильтра невозможно подчинить какому-либо общему правилу. Дело в том, что как необходимый тип фильтра (например, полоснопропускающий или низких частот), так и те условия, в которых должна проводиться фильтрация, в различных приложениях существенно отличаются.

С другой стороны, во многих приложениях требуется только низкочастотная фильтрация. В этом случае в качестве своего рода отправной точки выбора может служить шестиполюсный фильтр низких частот Баттеруорта. Иначе говоря, правомерна такая постановка вопроса: «Нужно ли в данном приложении отказываться от шестиполюсного фильтра Баттеруорта?» Нередко, если оптимальный выбор неосуществим, наиболее подходящим оказывается именно этот тип фильтра.

5.2. ШУМ И ИСКАЖЕНИЕ

Во многих вычислительных операциях, производимых ЭВМ, возникает шум. При использовании арифметики с фиксированной точкой это преимущественно бывает в результате умножения. После перемножения меньшая значащая половина произведения стирается, а бóльшая значащая половина (усеченная или округленная) запоминается. В данном случае *величину шума* определяет разница между истинным ответом и числом, которое получилось после усечения или округления. Уровень такого шума колеблется в пределах примерно половины последнего значащего бита.

При использовании арифметики с плавающей точкой ошибками такого рода дело не ограничивается. Здесь возникают новые погрешности, обусловленные антипереполнением при сложении. Когда складываются два числа с плавающей точкой, одно из которых значительно больше другого, все меньшее число или его часть могут быть потеряны.

Реализации фильтров, рассмотренные в гл. 4, не учитывали особенностей, связанных с конечным представлением чисел. Поэтому на практике характеристики реализаций фильтров отклоняются от характеристик теоретических реализаций, предполагающих использование идеальных арифметических операций.

С этим связано усложнение задачи реализации фильтров на ЭВМ, имеющих уменьшенное число битов. Например, применение для этой цели мини-ЭВМ с длиной слова, равной 12—16 битам, приводит к большему ограничению возможностей, чем применение больших ЭВМ со словами, длина которых равна 24—60 битов.

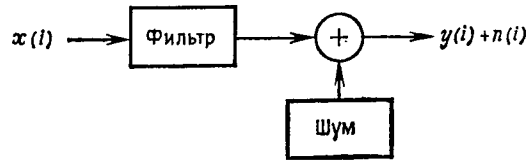

Рис. 5.1. Модель с шумом для идеального фильтра.

Для арифметики с конечным представлением чисел характерен ряд явлений, среди которых наиболее важными будут шум, искажение и неустойчивость. Они затронуты во многих работах. Назовем три сборника: первые два изданы под редакцией Рабинера и Рейдера (1972) и Оппенхейма и др. (1976), третий — под редакцией Лью (1975).

На рис. 5.1 изображена модель с шумом. К выходу идеального фильтра добавляются значения, порождаемые источником шума. В результате вместо последовательности значений $y(i)$ возникает последовательность $y(i)+n(i)$, где $n(i)$ — шум.

Характер шума в значительной степени зависит от используемой арифметики и от реализации фильтра.

Перечислим основные результаты, относящиеся к шуму.

1. Шум — функция числа битов, используемых в арифметических операциях; чем меньше это число, тем выше уровень шума.

2. Каскадные и параллельные[1]) реализации дают меньший по сравнению с комбинированными реализациями уровень шума. Если число полюсов или нулей равно четырем или больше четырех, то более предпочтительной будет каскадная форма фильтра. Это справедливо и для ИБО-, и для ИКО-фильтров.

3. Реализация фильтра в каскадной форме некоммутативна по отношению к выходу. Это означает следующее. При идеальных арифметических операциях выход фильтра не зависел бы от

[1]) Как правило, в дальнейшем параллельная и цепная реализации не будут затрагиваться. Хотя во многих случаях идеальные реализации таких типов имеют лучшие характеристики, чем другие, из-за большей сложности они оказываются непригодными для наших целей.

порядка расположения двухполюсных (двухнулевых) составляющих фильтра. В действительности порядок следования составляющих второго порядка в реализации фильтра влияет на шум. Поэтому возникает возможность выбора такой перестановки составляющих фильтров второго порядка, что полученный на выходе общего фильтра шум будет минимальным

Вычислительный шум включает в себя определенный «уровень» шума. Это нетрудно увидеть на примере идеализированной арифметической операции над числами с фиксированной точкой. Допустим, что для дробной части числа выделено m битов слова (само слово может иметь и более m битов). Тогда точно так же, как и при преобразовании в цифровую форму данных, при умножении каждый раз будет возникать ошибка, порядок которой равен примерно половине последнего значащего бита десятичной записи. Если предположить, что ошибки некоррелированы, то дисперсия ошибки после k последовательных умножений будет равна $k/12$ бит^2.

Таким образом, с учетом структуры слова получается уровень шума, равный $k2^{-2m}/12$ бит^2. Если перейти к децибелам, то

$$10\log_{10}(k2^{-2m}/12) = 10\log_{10}k - 20m\log_{10}2 - 10\log_{10}12 \approx$$
$$\approx 10\log_{10}k - 10.8 - 6m.$$

Следовательно, уровень шума имеет некоторую нижнюю границу, определенную числом битов, входящих в дробную часть слова ЭВМ. Уровень возрастает при увеличении числа умножений: чем больше фильтр, тем выше уровень шума.

Этот результат лишь приближенно отражает реальную картину. На практике, например, допущение о некоррелированности может не выполняться. Кроме того, полученный результат в каком-то смысле дает только нижнюю границу возможного шума; как будет показано, шум, порождаемый ИБО-фильтром, значительно возрастает, когда этот фильтр становится неустойчивым.

Одна из причин, по которой более предпочтительной является каскадная реализация фильтра, состоит в следующем: помимо того, что каждая каскадная составляющая вносит в данные свой собственный шум при фильтрации, *она также уменьшает шум, появившийся на предшествующих стадиях каскадной фильтрации.* Значит, в каскадном фильтре происходит и генерация, и сокращение шума. Это свойство в гораздо меньшей степени относится к комбинированным формам фильтров. Им более свойствен большой собственный шум, что, пожалуй, важно учитывать в приложениях.

Искажение. Искажение является результатом существенного расхождения передаточной функции фильтра в том виде, в ко-

тором он реализуется, с теоретической передаточной функцией. Обычно в качестве меры искажения берут среднеквадратичное значение

$$\varepsilon^2 = \int_0^F | H(f) - \hat{H}(f) |^2 \, df,$$

где ε^2 — среднеквадратичное искажение, $H(f)$ — теоретическая передаточная функция, $\hat{H}(f)$ — передаточная функция фильтра, реализованного на ЭВМ с конечной длиной слова.

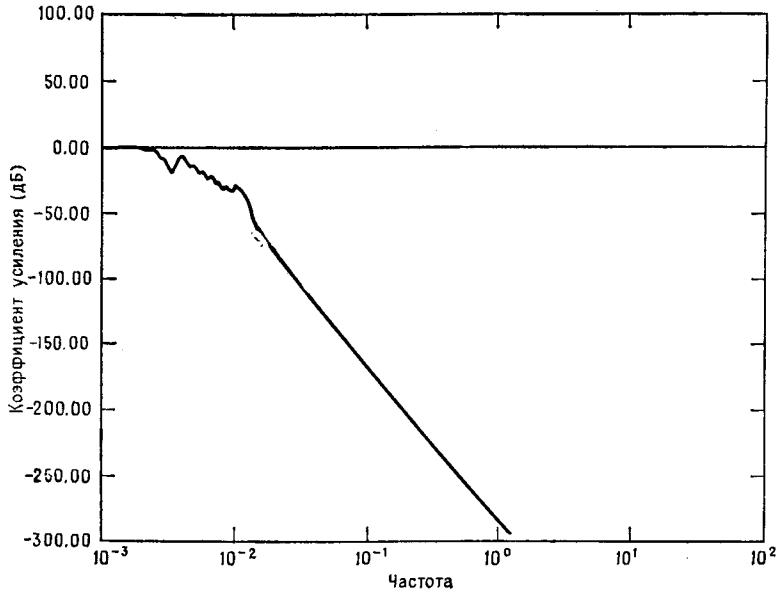

Рис. 5.2. Искажение передаточной функции, при котором она обретает очертания, свойственные низкополосному фильтру.

Первое и наиболее явное проявление искажения состоит в погрешностях и отклонениях от нужного вида передаточной функции. При уменьшении числа битов передаточные функции фильтров заметно искажаются (рис. 5.2.). Такое уменьшение числа битов приводит к тому, что исходная форма передаточной функции может измениться до неузнаваемости.

Искажение играет, пожалуй, меньшую роль, чем шум. К тому моменту, когда искажение, измеренное в линейном масштабе, станет заметным, шум, измеренный в логарифмическом масштабе, т. е. в децибелах, может оказаться достаточно большим для того, чтобы характеристика фильтра стала неудовлетворительной.

Разумеется, определение искажения тоже можно дать в логарифмической записи, т. е. в децибелах:

$$\varepsilon^2_{\text{дБ}} = 10 \int_0^F \log_{10} \left| \frac{H(f)}{\hat{H}(f)} \right|^2 df.$$

И в этом случае уровень шума будет проявляться сильнее.

Измерение $\hat{H}(f)$. Передаточную функцию $\hat{H}(f)$ можно измерять по крайней мере двумя способами.

Первый метод. На той же ЭВМ и с той же арифметикой, с которой вычисляется функция $\hat{H}(f)$ как преобразование Фурье фильтра, вычислить

$$\hat{H}(f) = \prod_{m=1}^{M/2} \frac{b_{0m} + b_{1m} \exp(-j\omega T) + b_{2m} \exp(-2j\omega T)}{1 + a_{1m} \exp(-j\omega T) + a_{2m} \exp(-2j\omega T)}.$$

Второй метод. Для каждой частоты f_p, в которой следует определить $\hat{H}(f)$, получить последовательность $x_p(i) = \sin(2\pi f_p i T)$ и пропустить ее через фильтр. Выход фильтра обозначим $y_p(i)$. Накопив достаточно длинную последовательность для переменного отклика, вычислить величину

$$s_p^2 = \frac{1}{N_p} \sum_{i=q}^{N_p+q-1} y_p^2(i).$$

В идеале f_p и N_p должны выбираться так, чтобы числу точек N_p соответствовало бы целое число периодов синусоиды. Параметр q выбирается таким образом, чтобы при $i \geqslant q$ выход $y_p(i)$ оставался устойчивым. Далее, вычисляется последовательность

$$z_p(i) = x_p(i)\, y_p(i),$$

для которой определяется последовательность корреляционных параметров

$$\rho_p = \frac{1}{s_p 2^{-1/2} N_p} \sum_{i=q}^{N_p+q-1} z_p(i),$$

где $2^{-1/2}$ и s_p — стандартные отклонения входа и выхода, используемые для нормировки. Допустим, что установившийся выход фильтра $y_p(i)$ имеет вид

$$y_p(i) = A_p \sin(2\pi f_p i T + \varphi_p), \quad \text{где} \quad i \geqslant q.$$

Тогда $s_p^2 = A_p^2/2$, и, значит,

$$z_p(i) = \sin(2\pi f_p i T)\, A_p \sin(2\pi f_p i T + \varphi_p) =$$
$$= \frac{A_p}{2} [\cos\varphi_p - \cos(4\pi f_p i T + \varphi_p)].$$

Суммирование на целом числе периодов членов последовательности $z_p(i)$ обращает в нуль крайнюю справа сумму косинусов, поэтому

$$\rho_p = \frac{\sqrt{2}}{s_p N_p} \sum_{i=q}^{N_p - q + 1} [\cos \varphi_p - \cos(4\pi f_p iT + \varphi_p)] =$$

$$= \left(\frac{\sqrt{2}}{(A_p/\sqrt{2})\, N_p} \right) \frac{A_p}{2} N_p \cos \varphi_p = \cos \varphi_p.$$

Таким образом, коэффициент усиления фильтра в точке f_p оказывается равным $A_p = \sqrt{2}\, s_p$, а его фаза φ_p — равной $\arccos(\rho_p)$. Разумеется, эти расчеты не застрахованы от некоторых ошибок, обусловленных тем, что в вычислениях использовано не целое число периодов синусоиды и что момент перехода q в установившийся режим определен неверно. Поэтому к всплескам в частотах следует относиться с известной осторожностью.

Более предпочтительным считается второй метод, поскольку в нем точнее отражаются стандартные операции для вычисления передаточной функции аналогового фильтра. Вместе с тем этот метод неудобно и невыгодно применять при вычислениях для больших наборов частот f_p, в особенности когда произведение BT невелико, где B — ширина полосы пропускания фильтра. Грубо говоря, для переменного отклика время достижения на выходе установившегося режима пропорционально $1/BT$. Поэтому при уменьшении значения BT требуется все большее время для получения установившейся части отклика.

Первая процедура, упомянутая выше, гораздо проще и удобнее для реализации. Однако ее нужно рассматривать скорее как указатель наличия шума и искажения, а не как средство для определения их силы.

5.3. УХУДШЕНИЕ ФИЛЬТРА

Хотя шум и искажение играют важную роль сами по себе, главным показателем качества цифрового фильтра остается устойчивость. В сущности и шум, и искажение, и неустойчивость суть только разные стороны одного и того же явления — ухудшения характеристики фильтра, связанного с использованием вычислений ограниченной точности.

Под неустойчивостью подразумевается появление бесконечных значений на выходе фильтра при конечных значениях входа. Примером совершенно неустойчивого фильтра служит, например, фильтр с разностным уравнением

$$y(i) = 2y(i-1) + x(i),$$

который в ответ на последовательность 1, 0, 0, ... выдает последовательность 1, 2, 4, ..., 2^i, ...

Условия устойчивости фильтра второго порядка получены в гл. 4. Область устойчивости снова изображена на рис. 5.3. На бесконечной плоскости всех возможных пар коэффициентов для фильтров второго порядка областью устойчивости фильтров служит небольшой треугольник, расположенный в начале координат. На рисунке показана точка $a_1 = -0.6$, $a_2 = 0.4$, которая попадает в треугольник коэффициентов устойчивых фильтров. Для оценки устойчивости нередко полезно обращаться к расстояниям d_1, d_2 и d_3. Чем ближе точка (a_1, a_2) расположена к сторонам треугольника, тем меньше оказывается одно из этих расстояний. В данном примере эти расстояния имеют следующие значения:

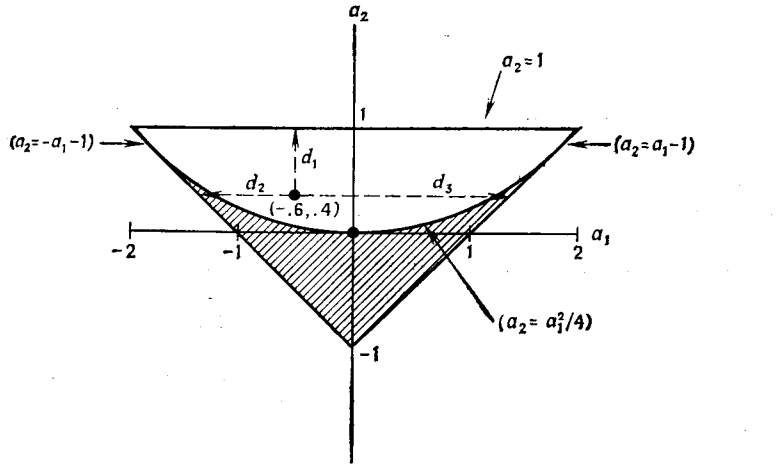

Рис. 5.3. Треугольник устойчивости для коэффициентов фильтра второго порядка. Каждая рекурсивная пара коэффициентов (a_1, a_2) определяет на плоскости одну точку. Если эта точка попадает в треугольник, то фильтр устойчив. Если она находится на границе или вне треугольника, то фильтр неустойчив.

$$d_1 = 0.6, \quad d_2 = 0.8, \quad d_3 = 2.0.$$

Для исследования устойчивости фильтра применяется и другой способ, в котором определяется некоторое пороговое значение. Как было показано Отнесом и Мак-Нейми (1970), для фильтров низких частот можно указать такую точку на шкале частот, что при значениях ширины полосы пропускания, меньших этой точки, цифровые фильтры теряют нужный вид передаточной функции и становятся неустойчивыми. Такое пороговое значение

выражается формулой

$$B_{t,N}=\frac{2^{-(t-1-\beta_N)/N}}{2\pi T},$$

где N — число полюсов фильтра, t — общее число битов слова применяемой ЭВМ, β_N — такое целое число, что

$$2^{\beta_N-1}<\binom{N}{\left[\frac{N}{2}\right]}\leqslant 2^{\beta_N}.$$

В табл. 5.1 приводятся значения порога для двух ЭВМ, работающих в режиме плавающей точки. Первая из них — IBM 704 (результаты для которой можно отнести также к ЭВМ IBM 709, 7090 и ЭВМ серии SRU 1106, 1107, 1108 и 1109), а вторая — IBM 360 (такой же формат используется в IBM 370 и ЭВМ серии Xerox 5, 6, 7 и т. д.). Значения, приведенные в таблице, относятся к каскадным реализациям фильтра, в комбинации составных частей которых входят фильтры низких частот Баттеруорта синусного и тангенсного типа; $T=0.005$, $N\geqslant 2$.

Кроме теоретических значений B для проверки на ЭВМ обоих типов были просчитаны $B_{\text{действ}}$ для реальных ситуаций (см. соответствующую колонку). Результаты почти одинаковы.

В качестве примера рассмотрим передаточные функции, изображенные на рис. 5.4. В этом примере для шестиполюсных фильтров значения B выбирались от 0.00031623 до 10 Гц. Расчеты производились на ЭВМ SRU 1108. При вычислениях передаточных функций использовался первый метод, в котором $\hat{H}(f)$

Таблица 5.1

Пороговые значения $B_{t,N}$ (в Гц) при нескольких значениях N для двух форматов слов ЭВМ с плавающей точкой

	Формат IBM 704 (t=36 бит)		Формат IBM 360 (t=32бит)	
N	B	$B_{\text{действ}}$	B	$B_{\text{действ}}$
2	0.0054	—	0.032	—
3	0.13	0.10	0.32	0.15
4	0.6	0.4	1.0	1.0
5	1.3	1.3	1.0	2.0
6	2.4	2.4	3.2—5.0 [a])	3.8

[a]) При $N=6$ вычислить порог трудно из-за шестнадцатеричного (4-битового) округления, применяемого на IBM 360; это порождает для данного значения ряд проблем.

получаются по конечным представлениям весов фильтров с помощью ЭВМ, используемой при самой фильтрации. Для фильтров рассматривалась *каскадная* реализация. Как и в табл. 5.1, возьмем $T=0.005$. Частота Найквиста равна в этом случае 100 Гц, что позволяет перейти к процентам на шкале частот. Действительные значения точек отсечки приводятся в табл. 5.2.

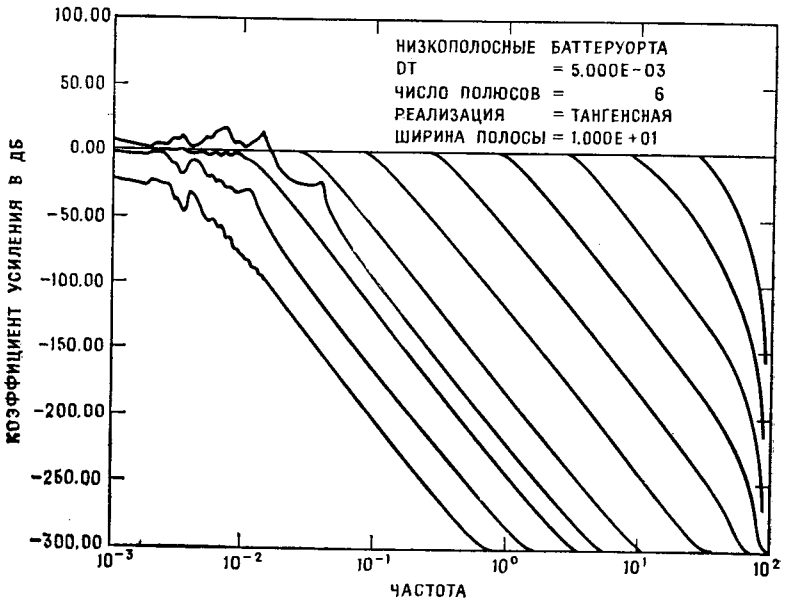

Рис. 5.4. Расчет десяти тангенсных низкополосных фильтров Баттеруорта с разными точками отсечки, выполненный на ЭВМ SRU 1108. Здесь $T=0.005$, $F=100$ Гц. Для фильтра с наименьшей точкой отсечки смещение настолько сильно, что он попал между седьмым и восьмым фильтрами.

Соответствующие веса составляющих фильтров приводятся в табл. 5.3.

После шестого фильтра в этом ряду происходит ухудшение характеристик фильтров. Это ухудшение выражается не только в том, что в полосе пропускания коэффициент усиления испытывает заметные всплески, но и в положении, которое начинает занимать 3-децибеловая точка. Точки отсечки 7—10 фильтров смещены вправо от того положения, которое они должны были занимать теоретически. Каждый из этих последних фильтров или может оказаться, или оказывается неустойчивым. В частности, частота отсечки для наименьшего значения B сместилась настолько сильно, что попала между значениями точек отсечки 0.0032 и 0.01 Гц.

Понятие плоскости коэффициентов и ее области устойчивых коэффициентов без труда переносится на случай фильтров комбинированного типа более высоких порядков. Например, для фильтра третьего порядка с разностным уравнением

$$y(i) = b_0 x(i) - a_1 y(i-1) - a_2 y(i-2) - a_3 y(i-3)$$

Таблица 5.2

Частоты отсечки и поведение 10 фильтров, приведенных на рис. 5.4

Фильтр	Точка отсечки (Гц)	Поведение фильтра
1	10.	Хорошее
2	3.1625	»
3	1.	»
4	0.3162	»
5	0.1	»
6	0.0316	»
7	0.01	Пограничное или плохое
8	0.0032	Плохое
9	0.001	»
10	0.0003	»

Таблица 5.3

Веса шестиполюсных тангенсных фильтров Баттеруорта (в каскадной реализации) с диапазоном частот отсечки от 0.01 Гц до 10 Гц (см. рис. 5.4)[a])

	B	b_0	a_{11}	a_{21}	a_{12}	a_{22}	a_{13}	a_{23}
1	10.00000	0.02046936		0.540254		0.641352		0.851887
			−1.464868		−1.561018		−1.761249	
2	3.16228	0.00231823		0.825146		0.868927		0.949944
			−1.816146		−1.859712		−1.940330	
3	1.00000	0.00024185		0.941106		0.956544		0.983972
			−1.940148		−1.955578		−1.982893	
4	0.31623	0.00002452		0.980991		0.986049		0.994871
			−1.980893		−1.985951		−1.994772	
5	0.10000	0.00000247		0.993949		0.995567		0.998375
			−1.993939		−1.995557		−1.998365	
6	0.03162	0.00000025		0.998083		0.998596		0.999486
			−1.998082		−1.998595		−1.999485	
7	0.01000	0.00000002		0.999393		0.999556		0.999837
			−1.999393		−1.999556		−1.999837	

[a]) Частота выборки — 200 выб./с. Для нерекурсивных членов указан только общий постоянный множитель; у каждой части фильтра имеется по три члена.

область устойчивых коэффициентов фильтров можно представлять как тетраэдр в пространстве троек всевозможных коэффициентов фильтров (a_1, a_2, a_3). Все тройки коэффициентов, которые попадают внутрь этого тетраэдра в трехмерном пространстве, соответствуют устойчивым фильтрам. Разумеется, что плоскость, определенная уравнением $a_3 = 0$, дает ту же плоскость, что была и в двумерном случае.

Аналогично для фильтра с бесконечной областью отклика (ИБО-фильтра), имеющего M полюсов, областью устойчивости будет многогранник в M-мерном пространстве. Коэффициенты $(a_1, \ldots, a_M)$, попадающие в него, соответствуют устойчивым фильтрам. Можно, конечно, и более подробно остановиться на этом случае, но мы не будем этого делать по той причине, что комбинированные формы фильтров при $M \geqslant 2$, вообще говоря, не рекомендуются. В этом случае обращаются к каскадным формам фильтров, которые могут быть устойчивыми даже тогда, когда соответствующий фильтр комбинированного вида будет неустойчивым. Короче говоря, каскадные реализации фильтров всегда имеют больший диапазон устойчивости, меньший уровень шума и искажений, чем комбинированные. Поэтому каскадные реализации предпочтительнее.

Разумеется, не обходится без исключений. В вычислительном плане каскадная реализация менее экономна, чем комбинированная. Так что для использования устойчивой (на данной ЭВМ) комбинированной формы фильтра при обработке больших объемов данных нет никаких препятствий.

Веса для комбинированных реализаций первых семи фильтров низких частот Баттеруорта синусного типа приводятся в табл. 5.4. Из этой таблицы видно, что веса устойчивых комбинированных фильтров имеют значительно большие абсолютные значения. Если для фильтров каскадного типа справедливы оценки

$$|a_1| < 2, \quad |a_2| < 1,$$

то соответствующие веса комбинированных фильтров оцениваются следующим образом:

$$|a_k| < \binom{M}{k} = \frac{M!}{k!\,(M-k)!}.$$

[*Замечание*. Эти оценки, дающие границы для коэффициентов, не могут служить условиями устойчивости. Хотя коэффициенты устойчивых фильтров обязательно удовлетворяют этим оценкам, можно привести и такие коэффициенты, лежащие в тех же границах, что соответствующие им фильтры будут неустойчивыми (например, $a_1 = 1$, $a_2 = -0.5$).]

Таким образом, комбинированная форма фильтра требует для реализации большего числа битов и в целой, и в дробной частях. Это приводит к тому, что общее число полезных битов при комбинированной реализации уменьшается.

Таблица 5.4

Веса шестиполюсных синусных фильтров Баттеруорта (в комбинированной реализации) с диапазоном частот отсечки от 0.01 Гц до 10 Гц [а])

	B	b_0	a_1	a_2	a_3	a_4	a_5	a_6
1	10.000000	0.512018-03		09.6857		06.4863		0.298003
			−4.79661		−10.5252		−2.14663	
2	3.162300	0.791620-06		13.1553		11.5766		0.581305
			−5.61649		−16.4476		−4.34912	
3	1.000000	0.904631-09		14.4005		13.8295		0.885697
			−5.87863		−18.8154		−5.42168	
4	0.316230	0.943210-12		14.8088		14.6205		0.962342
			−5.96162		−19.6191		−5.81099	
5	0.100000	0.955728-15		14.9394		14.8791		0.987935
			−5.98786		−19.8789		−5.93960	
6	0.031623	0.959648-18		14.9808		14.9617		0.996169
			−5.99616		−19.9616		−5.98083	
7	0.010000	0.953604-21		14.9939		14.9879		0.998787
			−5.99879		−19.9878		−5.99393	
	Предел	0.000000-00	−6.00000	15.0000	−20.0000	15.0000	−6.00000	1.000000

[а]) Скорость выборки — 200 выб./с.

Нерекурсивные коэффициенты как показатели устойчивости. В гл. 4 указывалось, что с целью нормировки фильтра обычно вводится постоянный множитель. При этом коэффициент усиления фильтра в полосе пропускания становится равным единице. Оказывается, этот постоянный множитель доставляет полезную информацию об устойчивости.

Поясним, как это получается. Предположим, что $\hat{H}(f)$ — передаточная функция до нормировки, $f_{\max}$ — частота, в которой значение $|\hat{H}(f)|^2$ достигает наибольшей величины. Передаточную функцию $H(f)$ определим в этом случае посредством формулы

$$H(f) = \frac{\hat{H}(f)}{|\hat{H}(f_{\max})|}.$$

Эту нормировку обычно проводят, умножая все нерекурсивные члены на $1/|\hat{H}(f_{\max})|$. Обозначим символами $\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_M$ нерекурсивные веса фильтра комбинированного вида до нормировки. Коэффициент $\hat{b}_0$ по определению полагается равным единице.

Тогда

$$b_k = \frac{\hat{b}_k}{|\hat{H}(f_{\max})|} = b\hat{b}_k,$$

где b — нормирующий множитель:

$$b = \frac{1}{|\hat{H}(f_{\max})|}.$$

Мы вскоре убедимся, что с уменьшением ширины полосы пропускания соответственно уменьшается значение b. В момент достижения такой ширины полосы пропускания, что соответствующее значение b окажется равным машинному нулю, возникают два взаимосвязанных явления. Легче всего их проследить на примере синусоиды $x(i)$ с максимальной частотой $f_{\max}$:

$$x(i) = \sin(2\pi f_{\max} iT), \quad i \geqslant 0.$$

Предположим, что i достаточно велико для того, чтобы был достигнут установившийся режим. Если ширина полосы пропускания стремится к нулю, то в частоте $f_{\max}$ рекурсивные веса дают в пределе —1 (читатель может проверить это, полагая сначала $f_{\max} = 0$, а затем переходя к общему случаю). Поэтому в том особом случае, когда синусоида берется в частоте $f_{\max}$ и достигается установившийся режим, рекурсивную составляющую фильтра можно записать в виде

$$-\sum_{k=1}^{M} a_k y(i-k) = \sin(2\pi f_{\max} iT + \varphi) - \varepsilon_i,$$

где ε_i — малый синусоидальный член (также имеющий частоту $f_{\max}$), а φ — фазовый угол, определяемый фильтром. Следовательно,

$$y_i = \sum_{k=0}^{M} b_k x(i-k) + \sin(2\pi f_{\max} iT + \varphi) - \varepsilon_i,$$

что равносильно соотношению

$$\varepsilon_i = b \sum_{k=0}^{M} \hat{b}_k x(i-k).$$

Таким образом, результатом воздействия нерекурсивной составляющей фильтра оказывается синусоида с относительно малой амплитудой, сокращающая член ε_i.

Первый случай (фиксированная точка). Если в этом случае b меньше 2^{-m}, где m — число битов, отведенных в слове ЭВМ для дробной части, то b оказывается равным нулю. Поэтому

на фильтр не подается никаких значений. Кроме того,

$$|\hat{H}(f_{\max})| = \frac{1}{b} = \frac{1}{0} = \infty,$$

и, значит, *фильтр неустойчив.*

Второй случай (плавающая точка). Этот случай чуть более сложен, но в основном сводится к тому же результату: хотя для чисел с плавающей точкой диапазон представления шире, при сложении одно число может выступать по отношению к другому в качестве нуля. Если слово с плавающей точкой имеет m битов для мантиссы и числа x и y связаны неравенством $|x| > |2^m y|$, то $x+y=x$. Таким образом, при суммировании в результате *антипереполнения* x полностью поглощает значение y. Значит, хотя y может оказаться и не настолько малой величиной, чтобы быть машинным нулем, при сложении в режиме плавающей точки она может выступить в качестве нуля.

Следовательно, если $b < 2^{-m}$, где m — число битов мантиссы, отведенных для мантиссы слова с плавающей точкой, то для данных, значения которых превосходят единицу, величина b может выступать в качестве нуля, и результат получится такой же, как и в первом случае.

Итак, множитель b позволяет сделать вывод об относительной устойчивости фильтра, реализованного в комбинированной форме.

То же относится и к постоянному множителю b^1: $b^1 = b^{2/M}$ в случае каскадной реализации.

Нормирующий множитель b помогает также определить число битов, необходимых для реализации фильтра. Величина b должна быть такой, чтобы $b > 2^{-m}$, где m — число битов дробной части машинного слова, используемого при реализации. Если прологарифмировать это неравенство (основание логарифма возьмем равным 2), то получим оценку $m > -\log_2 b$. Таким образом, по известному значению b легко определить минимальное значение m. Оказывается, что такое соотношение между b и m накладывает определенные условия на $H(f)$. Предположим, что значение b неизвестно, а для $H(f)$ известен только общий вид. Спрашивается, можно ли получить отсюда информацию о множителе b и, значит, о числе битов m? Ответ положительный.

Допустим, что передаточная функция $H(f)$ отвечает устойчивому фильтру и что ее нули расположены на верхней полуплоскости (на левой полуплоскости для переменной $j\omega$). Нетрудно установить, что ненормированная передаточная функция $\hat{H}(f)$ удовлетворяет в этом случае соотношению

$$\int_0^F \log_2 |\hat{H}(f)|^2 \, df = 0.$$

Как указано в работе Грэя и Мэркела (1974), из него вытекают некоторые важные следствия. В частности,

$$\int_0^F \log_2 |H(f)|^2 \, df = \int_0^F \log_2 |b\hat{H}(f)|^2 \, df =$$

$$= \int_0^F 2\log_2 b \, df + \int_0^F \log_2 |\hat{H}(f)|^2 \, df = 2F \log_2 b.$$

Следовательно,

$$\log_2 b = \frac{1}{2F} \int_0^F \log_2 |H(f)|^2 \, df.$$

Таким образом, число битов, необходимое для реализации данного нормированного фильтра с минимальной фазой, определяется формулой

$$m > -\frac{1}{2F} \int_0^F \log_2 |H(f)|^2 \, df.$$

Из этой оценки непосредственно следует, что фильтр не может быть реализован, если функция $|H(f)|^2$ равна 0 на каком-нибудь интервале частот f ненулевой длины.

Поясним на примерах, как используется эта формула.

1. Биномиальный фильтр M-го порядка (относящийся к ИКО-фильтрам), имеет веса

$$b_k = \binom{M}{k} 2^{-M}, \quad k = -M, \ \ldots, \ M,$$

и такую передаточную функцию $H(f)$, что

$$|H(f)|^2 = \cos^{2M}(\pi f T).$$

Следовательно,

$$-\int_0^F \log_2 |H(f)|^2 \, df = -2M \int_0^F \log_2 (\cos(\pi f T)) \, df = M,$$

и для дробной части слова ЭВМ требуется M битов. В справедливости этого результата помогает убедиться тот факт, что коэффициенты

$$b_0 = b_M = 1/2^M$$

при числе битов, меньшем M, обратятся в нули.

2. Фильтр низких частот первого порядка

$$y(i) = (1-\alpha)\, x(i) + \alpha y(i-1), \quad 0 < \alpha < 1,$$

имеет коэффициент b, равный $1-\alpha$. Поскольку $m > -\log_2 b$, при $(1-\alpha) = 2^{-N}$ имеем $m > N$. Эти соотношения вместе с интегральной формулой, на которой они основываются, дают важную информацию о фильтрах.

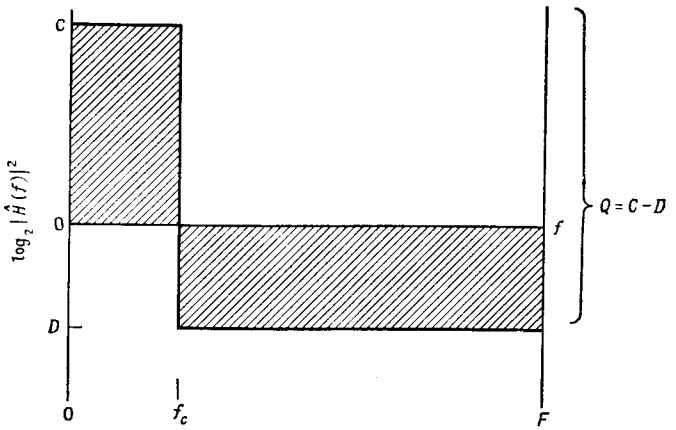

Рис. 5.5. Идеализированный низкополосный фильтр.

Рассмотрим ненормированный фильтр низких частот, показанный на рис. 5.5. Согласно интегральному соотношению

$$\int_0^F \log_2 |\hat{H}(f)|^2 \, df = 0,$$

должно выполняться равенство

$$Cf_c = -D(F - f_c).$$

В левой части этой формулы стоит выражение для площади прямоугольника, расположенного над нулевым уровнем, а справа — выражение для площади прямоугольника, расположенного под нулевым уровнем.

Решим это уравнение относительно C:

$$C = -D\left(\frac{F - f_c}{f_c}\right).$$

Значение коэффициента b определяется формулой $b = 2^{-C/2}$. Для примера выберем значение $f_c = F/4$, полагая $C - D = 20$. Это соответствует интервалу 60 дБ. После несложных подсчетов можно получить значение $b = 1/181$.

На рис. 5.6 изображен полученный результат, сопоставленный с реальным фильтром. В качестве последнего выступает шестиполюсный шестинулевой фильтр Чебышева типа 2 с гашением на 60 дБ в полосе заграждения и с частотой отсечки $f_c = F/4$. Нормирующий множитель оказывается равным 1/132. Фильтр Чебышева имеет значительно менее четкие очертания, чем фильтр идеальной формы, представленный на рис. 5.5.

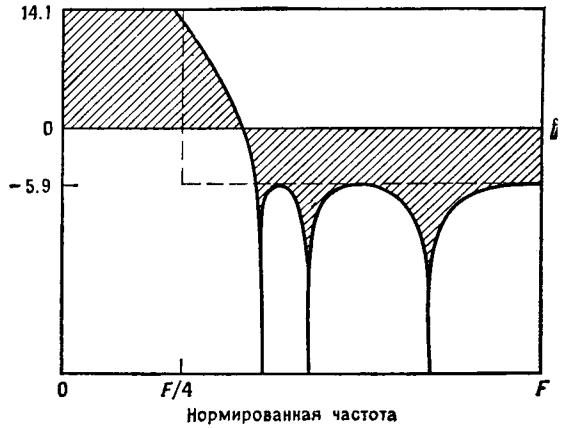

Рис. 5.6. Примерные очертания фильтра Чебышева второго типа. Интеграл для заштрихованной области равен нулю.

5.4. РЕАЛИЗАЦИЯ ФИЛЬТРА

В этом параграфе затронуты некоторые процедуры, применяемые в практической фильтрации.

На рис. 5.7 показана подпрограмма BFILT1. Она предназначена для получения одного значения выхода фильтра по одному значению на входе. Подпрограмма выглядит следующим образом.

1. Для связи с внешней программой FILTR1 в предложении COMMON перечисляются общие переменные: X — значение на входе (одно), Y — значение на выходе (тоже одно), B0 — единый для всех частей комбинированной реализации множитель, A1 — рекурсивные веса a_{1m}, A2 — рекурсивные веса a_{2m}, M — число полюсов фильтра.

2. Обращение к подпрограмме по основному названию BFILT1 происходит только один раз. При этом устанавливаются начальные значения, обнуляется память и т. д.

3. Вход в подпрограмму по названию FILT1 дает возможность получить одно новое значение выхода по одному новому значению входа. При этом значение входа размещается в области переменной X, затем совершается обращение к подпрограмме по названию FILT1 и значение выхода фильтра помещается в область переменной Y.

4. Веса фильтра следует поместить в область COMMON программы FILTR1.

```
      SUBROUTINE BFILT1
      COMMON /FILTR1/ X, Y, B0, A1(10), A2 (10), M
      DIMENSION Y0(11), Y1(11)
      DO 10 K = 1, 11
      Y0(K) = 0.
10    Y1(K) = 0.
      M2 = M - M/2
      RETURN
      ENTRY FILT1
      Y0(1) = X
      DO 20 K = 1, M2
      K1 = K + 1
      YI = B0 * Y0(K) - A1(K) * Y0(K1) - A2(K) * Y1(K1)
      Y1(K1) = Y0(K1)
20    Y0(K1) = YI
      Y = YI
      RETURN
      END
```

Рис. 5.7. Подпрограмма BFILT1. Обратите внимание на то, что она имеет две точки входа. Вход BFILT1 используется лишь один раз — для начала операции фильтрования. Вход FILT1, следующий за ним, используется по одному разу для каждой точки данных, которую следует подвергнуть воздействию фильтра.

Поясним роль этой подпрограммы на примере программы для фильтрации. Допустим, что основная программа имеет следующий вид:

```
COMMON XX(1000), YY(1000)
COMMON/FILTR1/X, Y, т. д.      } задание областей COMMON
.
.
.
CALL BFILT1                    } установка начальных значений для фильтрации
.
.
.
M = 6
T = 0.005
B = 25.                        } получение весов шестиполюсного синусного фильтра низких частот
CALL LPSB(...)
```



```
.
.
.
(получение любых данных XX)
.
.
.
DO 100 I = 1, 1000
X = XX(I)
CALL FILT1                     } цикл, в котором непосредственно проводится фильтрация
YY(I) = Y
100 CONTINUE
.
.
.
(вывод на дисплей или на другое устройство)
```

Операцию фильтрации данных можно непосредственно вставить в основную программу. Однако ее выделение в подпрограмму имеет ряд преимуществ.

Предложения COMMON используются по двум причинам. Во-первых, отсутствие данных в операторах вызова сокращает время работы программы. Во-вторых, в предложении ENTRY не требуется перечисления переменных. Это позволяет пользоваться программой на тех машинах, на которых возможность такого перечисления не предусмотрена.

Программа BFILT1 предназначена только для таких фильтров, у которых уравнение для каждой части каскадной реализации имеет вид

$$y^{(m)}(i) = b_0 y^{(m-1)}(i) - a_{1m} y^{(m)}(i-1) - a_{2m} y^{(m)}(i-2),$$

где $y^{(m-1)}(i)$ — вход m-й составляющей фильтра, а $y^{(m)}(i)$ — соответствующий выход этой составляющей. Таким образом, в каждом уравнении присутствует только один нерекурсивный член, коэффициент при котором остается одним и тем же для всех составляющих фильтра.

В более общем случае соответствующее уравнение имеет вид

$$y^{(m)}(i) = b_{0m} y^{(m-1)}(i) + b_{1m} y^{(m-1)}(i-1) + b_{m2} y^{(m-1)}(i-2) - a_{1m} y^{(m)}(i-1) - a_{2m} y^{(m)}(i-2).$$

На рис. 5.8 представлена подпрограмма BFILT2, которая полностью аналогична подпрограмме BFILT1. Единственное отличие связано с увеличением до трех числа нерекурсивных членов для каждой составляющей фильтра. С этим, естественно, связано увеличение необходимого объема памяти и числа перемещений данных.

Обе подпрограммы имеют один недостаток. При их работе значительная часть времени уходит на сдвиги данных.

В подпрограмме BFILT3 (рис. 5.9) сделана попытка обойти этот недостаток заменой непосредственного перемещения данных

```
      SUBROUTINE BFILT2
      COMMON /FILTR2/ X, Y, B0(10), B1(10), B2(10),
     *     A1(10), A2(10), M
      DIMENSION Y0(11), Y1(11), Y2(11)
      DO 10 K = 1,11
      Y0(K) = 0.
      Y1(K) = 0.
10    Y2(K) = 0.
      M2 = M - M/2
      RETURN
      ENTRY FILT2
      Y0(1) = X
      DO 20 K = 1, M2
      K1 = K + 1
      YI = B0(K) * Y0(K) + B1(K) * Y1(K) + B2(K) * Y2(K)
           - A1(K) * Y0(K1) - A2(K) * Y1(K1)
      Y2(K1) = Y1(K1)
      Y1(K1) = Y0(K)
20    Y0(K1) = YI
      Y2(1) = Y1(1)
      Y1(1) = Y0(1)
      Y = YI
      RETURN
      END
```

Рис. 5.8. Подпрограмма BFILT2 аналогична подпрограмме BFILT1; исключение составляет более широкий набор рекурсивных коэффициентов.

перестановкой индексов. Эта подпрограмма работает очень хорошо, особенно в том случае, если она написана не на алгоритмическом, а на машинном языке. В то же время подпрограмма на Фортране не всегда обеспечивает сокращение времени работы, поскольку на разных ЭВМ компиляция происходит неодинаково. Некоторые компиляторы, плохо приспособленные к работе с индексами, не дают более эффективной машинной подпрограммы. Поставив во главу угла скорость, следует или написать подпрограмму на машинном языке, или поэкспериментировать с программами на

Фортране и определить самую быструю из них для имеющейся ЭВМ.

На рис. 5.10 показана подпрограмма, в которой реализована циклическая буферная схема, применяемая для ИКО-фильтров.

```
      SUBROUTINE BFILT3
      COMMON /FILTR3/ X, Y, B0(10), B1(10), B2(10),
           A1(10), A2(10), M
      DIMENSION YM (11,3)
      DO 10 K = 1, 11
      DO 10 I = 1, 3
10    YM (K, I) = 0.
      M2 = M - M/2
      J1 = 1
      J2 = 3
      J3 = 2
      RETURN
C
C
      ENTRY FILT3
      YM (1, J1) = X
      DO 20  K = 1, M2
      K1 = K + 1
      YMM = B0(K) * YM(K,J1) + B1(K) * YM(K,J2)
     *     + B2(K) * YM(K,J3) - A1(K) * YM(K1,J2)
     *     - A2(K) * YM(K1,J3)
20    YM(K1,J1) = YMM
      J4 = J1
      J1 = J3
      J3 = J2
      J2 = J4
      Y = YMM
      RETURN
      END
```

Рис. 5.9. Подпрограмма BFILT3. Применяется аналогично программе BFILT2. В подпрограмме BFILT3 предусмотрена перестановочная буферная схема, обеспечивающая циклический сдвиг.

В этой программе используется большой внутренний вектор, предназначенный для запоминания 1001 значения. В программе этот массив назван BUFFER. Предполагается, что разностное

уравнение фильтрации имеет вид

$$y(i) = \sum_{k=-M/2}^{M/2} b_k x(i-k),$$

где M — нечетное. Если дополнительно предположить, что фильтр симметричен, то можно это уравнение переписать в виде

$$y(i) = b_0 x(i) + \sum_{k=1}^{M/2} b_k [x(i-k) + x(i+k)].$$

Поскольку внутри буфера никаких перемещений данных не происходит, время, необходимое для поиска нужной части данных, увеличивается по сравнению с более простыми, нециклическими буферными схемами. Как и в случае рекурсивного

```
      SUBROUTINE BFILT4
      COMMON /FILTR4/ X, Y, B(1001), M
      DIMENSION BUFFER (1001)
      DO 5 I = 1, 1001
5     BUFFER (I) = 0.
      LOC = 1
      M2 = M/2
      RETURN
      ENTRY FILT4
      BUFFER(LOC) = X
      L0 = LOC - M2
      IF (L0. LT. 1) L0 = L0 + M
      Y = BUFFER (L0) * B(1)
      DO 10 I = 1, M2
      LL = L0 - I
      IF (LL. LT. 1) LL = LL + M
      LR = L0 + I.
      IF (LR. GT. M) LR = LR - M
      Y = Y+(BUFFER(LL)+BUFFER(LR)) * B(I + 1)
10    CONTINUE
      LOC = LOC + 1
      IF (LOC. GT. M) LOC = LOC - M
      RETURN
      END
```

Рис. 5.10. Циклическая буферная схема для симметричных ИКО-фильтров.

фильтра, можно сделать эту процедуру весьма эффективной, переходя к машинному языку, хотя соответствующая программа на Фортране *могла* быть довольно плохой.

5.5. ДЕЦИМАЦИЯ

Собственно говоря, слово «децимация» означает потерю одной десятой военных сил в ходе сражения или в результате наказания[1]). В обработке сигналов этим словом принято называть удаление ненужной или излишней информации и соответственное уплотнение данных.

Как правило, сокращение выборки децимацией необходимо в целях экономии. Допустим, что в течение 30 мин проводится тестовое испытание со скоростью выборки, равной 400 выб./с. В результате будет получено $30 \cdot 60 \cdot 400 = 720\,000$ значений данных. Если попытаться изобразить их на графике, длина которого равна 30 дюймам, то на каждый дюйм попадет 24 000 значений. На каждую точку, которую может отметить графопостроитель, имеющий интервал 0.01 дм, приходится по 240 точек выборки. В этом случае при децимации должна быть оставлена для дальнейшей обработки не более чем 1/240 часть всей выборки.

В большинстве случаев децимации сопутствует необходимость низкополосной фильтрации. Эта фильтрация должна предотвратить подмену частот, т. е. появление в низких частотах информации, относящейся к высоким частотам.

В качестве примера рассмотрим временную последовательность

$$x(i) = \cos\left(\frac{3\pi i}{4}\right).$$

Первые десять точек этой последовательности изображены в верхней части рис. 5.11. Все эти точки принадлежат синусоиде с частотой, составляющей 75% частоты Найквиста. Для наглядности точки нанесены на график косинуса. Если каждую вторую точку, начиная с $i = 1$, выбросить, то получится последовательность, изображенная на нижней части рисунка. Точки этой последовательности можно принять за значения косинуса с частотой, равной 25% исходной частоты Найквиста.

Для предотвращения такой подмены частот следует применять низкополосную фильтрацию. Точка отсечки определяется в этом случае следующим образом.

Допустим, что индекс децимации равен p (т. е. в новой последовательности останется только каждое p-е значение исходного ряда), а исходная частота Найквиста равна $F = S/2 = 1/(2T)$ Гц. Тогда новая частота Найквиста должна быть равной

[1]) В Древнем Риме, во времена Республики, децимация служила наказанием за проявленную в сражении трусость: провинившийся отряд обычно делили на группы по 10 солдат и в каждой группе бросался жребий. Тот, кому выпадал жребий, подвергался ужасной каре: с него сдирали кожу его же товарищи.

p-й части исходной частоты Найквиста, т. е. F/p Гц, поскольку всякая частота, попадающая в интервал $(F/p, F)$, после децимации может появиться в частотах, меньших новой частоты Найквиста. Итак, децимации должна предшествовать фильтрация данных с помощью фильтра низких частот, точка отсечки которого равна $F' = 1/(2pT)$ Гц. Формулы децимации сведены в табл. 5.5.

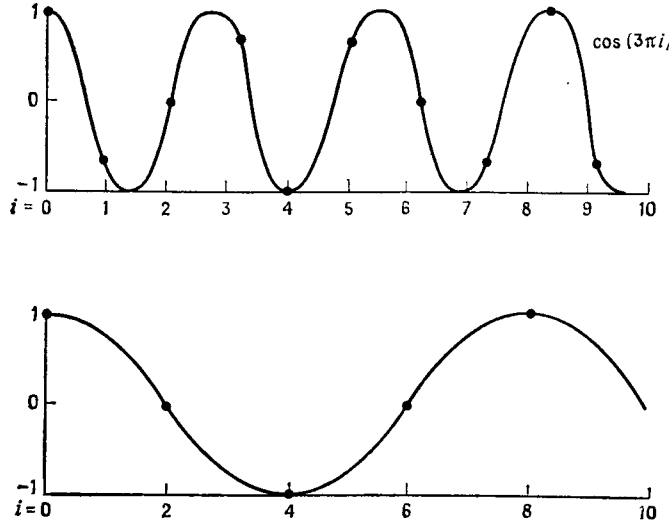

Рис. 5.11. Косинусоида до (вверху) и после (внизу) децимации, при которой выбрасывается каждая вторая точка.

Выбор фильтра. Точку отсечки фильтра, применяемого при децимации, можно выбрать несколькими способами. Одним таким стандартным способом предусматривается выбор точки отсечки

Таблица 5.5

Формулы дедимации в терминах величин T, S и F

	Через T	Через S	Через F
Скорость выборки и интервал	$S = \frac{1}{T}$	$T = \frac{1}{S}$	$S = 2F$ $T = 1/(2F)$
Частота Найквиста	$F = \frac{1}{2T}$	$F = \frac{S}{2}$	F
Частота Найквиста после децимации	$F' = \frac{1}{2pT}$	$F' = \frac{S}{2p}$	$F' = \frac{F}{p}$
Новый интервал и скорость	$T' = pT$ $S' = 1/(pT)$	$T' = p/S$ $S' = S/p$	$T' = p/(2F)$ $S' = 2F/p$

фильтра в частоте F', т. е. в 3-децибеловой точке. В этой точке мощность исходного сигнала падает наполовину, а соответствующая амплитуда уменьшается примерно на 30%. Примерно такое же уменьшение происходит для информации, которая имеет частоты чуть правее точки отсечки. Эта информация может появляться левее новой частоты Найквиста. Во втором способе предусматривается снижение амплитуды данных, которые могут вызывать подмену частот, до заданных пределов. Например, можно потребовать, чтобы в новой точке отсечки F' мощность сигнала падала бы на 60 дБ. Такое уменьшение мощности гарантирует применение фильтра низких частот с глушением на 60 дБ информации, расположенной правее точки F'. Данные с частотами, близкими к частоте Найквиста, в этом случае могут потеряться. Итак, при первом способе выбора 3-децибеловое падение мощности может вызывать подмену частот, что не позволяет делать выводы о низких частотах с достаточной уверенностью; во втором способе проблема подмены частот уже не играет такой существенной роли, но зато использование глушения на 60 дБ может приводить к потере полезных данных с частотами ниже F'.

Вслед за определением точки отсечки фильтра по величине F' возникает проблема выбора типа фильтра (рекурсивного — ИБО или нерекурсивного — ИКО). Помимо этого, нужно определить число коэффициентов конкретного типа фильтра. Преимущества и недостатки выбора того или иного фильтра сформулированы в табл. 5.6.

Как указано в таблице, при выборе нерекурсивных фильтров отпадает необходимость вычисления каждого значения выхода. В этом случае при децимации p к 1 нужно вычислять только каждое p-е значение выхода. В этом состоит главное преимущество выбора нерекурсивных фильтров.

Какого-то определенного правила выбора необходимого типа фильтра сформулировать нельзя. Помимо индекса децимации этот выбор диктуется многими другими факторами, в частности характером аппаратного оборудования программного обеспечения. Наша точка зрения по этому вопросу сводится к следующему.

Во-первых, мы считаем, что для децимации, проводимой на устройствах с фиксированной точкой (назовем здесь аппаратное оборудование специального назначения, мини- и микро-ЭВМ), более предпочтительным является выбор нерекурсивного фильтра.

Во-вторых, на ЭВМ широких возможностей или на вычислительных машинах для обработки больших массивов данных с плавающей точкой и объемом слова не меньше 32 битов более подходящими могут оказаться рекурсивные фильтры: исследователь заранее знает характер той информации, которую он может получить, и, значит, ему не нужно сначала анализировать фильтр и только потом применять его. Кроме того, важным преимуще-

Таблица 5.6

Преимущества и недостатки ИКО- и ИБО- фильтров при децимации

ИКО	ИБО
Преимущества	
1. Необходимо вычислять только каждую p-ю точку выхода	1. Объем необходимой памяти как для весов, так и для данных сильно сокращается
2. Фильтр всегда устойчив	2. Как правило, можно добиться, чтобы фильтр имел в полосе пропускания почти линейную фазу
3. Легко проводить бесфазовую фильтрацию	3. Часто наиболее эффективный фильтр (в вычислительном плане)
4. Фильтрацию можно проводить в режиме фиксированной точки на мини-ЭВМ	
Недостатки	
1. Может потребоваться большое число весовых коэффициентов	1. Должны вычисляться все значения на выходе. (Имеются формулировки, в которых не требуется, чтобы вычислялись все значения на выходе, однако они оказываются обычно менее эффективными)
2. Возможна зашумленность	2. Возможна зашумленность и даже неустойчивость
3. Может потребоваться большой объем памяти для весовых коэффициентов и для данных	3. Бесфазовая фильтрация невозможна (если только данные не проходят через один и тот же фильтр дважды, каждый раз в противоположных направлениях)
	4. Наилучшая реализация — в арифметике с плавающей точкой

ством такого выбора, особенно при обработке большого числа рядов, служит меньший объем памяти, необходимый для запоминания данных.

Децимацию может оказаться необходимым проводить в два и более приемов. Децимация в несколько этапов носит название *каскадной* децимации. Предположим, что $p=q^2$. Тогда процесс децимации естественно разбить на два шага, в каждом из которых проводится децимация q к 1. Например, допустим, что $f_s=1\,000\,000$ и необходима децимация с индексом $p=10\,000$ (т. е. 10 000 к 1). Для этого потребуется фильтр, точка отсечки которого располагается в частоте, равной 1/10 000 частоты Найквиста. На многих ЭВМ такого значения точки отсечки невозможно добиться. Вместе с тем разбиение процесса на два этапа с децимацией 100 к 1 на таких ЭВМ сделать нетрудно. Поэтому, если фильтр для одного этапа реализовать невозможно, прибегают к последовательной фильтрации.

По отношению к результатам фильтрации расположение фильтров в последовательной форме, вообще говоря, не коммутативно. Например, если требуется децимация 30 к 1, то с учетом того,

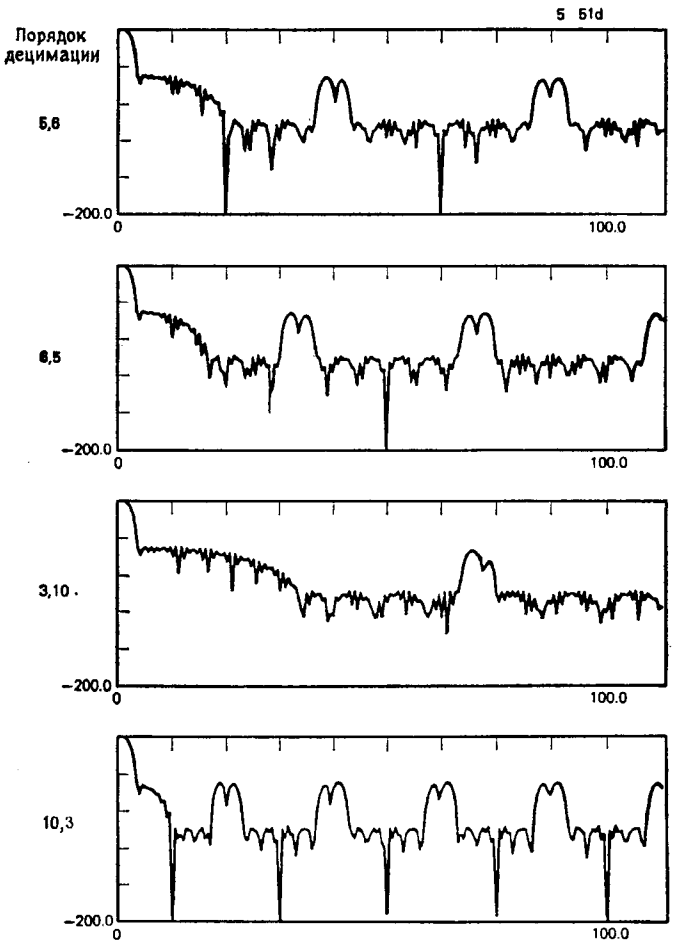

Рис. 5.12. Четыре варианта фильтров, осуществляющих децимацию 30 к 1.

что $30=2\cdot3\cdot5$, ее можно провести $3\cdot2!+3!=12$ различными способами (например, 5 к 1 и 6 к 1, 6 к 1 и 5 к 1, 3 к 1 и 10 к 1, 10 к 1 и 3 к 1 и т. д.).

На рис. 5.12 показаны передаточные функции для перечисленных в скобках вариантов каскадной децимации. Эти общие передаточные функции получены для нерекурсивных фильтров, реализованных с помощью процедуры Паркса — Мак-Клеллана,

рассмотренной в предыдущей главе. Параметры для фильтров, использованных перед децимацией, приведены в следующей таблице (частота Найквиста полагается равной 100 Гц):

Децимация	M	Частота отсечки (Гц)
3 к 1	18	13.3
5 к 1	30	8.
6 к 1	36	6.7
10 к 1	60	4.

Глушение в полосе заграждения этих фильтров равно примерно —46 дБ. При каскадной реализации этот уровень почти везде становится в два раза ниже, за исключением полос с подменой частот, где появились двойные выступы.

Хотя полоса пропускания во всех четырех показанных случаях и одинакова, полоса заграждения в каждой из них непохожа на другие. Например, полоса заграждения при децимации 5 к 1 и 6 к 1 выглядит иначе, чем в случае 6 к 1 и 5 к 1. Однако максимальные значения выступов в полосах заграждения нигде не превышают —46 дБ, что дает право говорить о том, что характеристика составного фильтра ничуть не хуже, чем исходного.

При фильтрации с помощью нерекурсивных фильтров, предотвращающей подмену частот, следует обратить внимание на следующее.

Во-первых, при фильтрации требуется только p-я часть выхода. Поэтому схему фильтрации необходимо изменить так, чтобы $p-1$ значений входа только запоминалось и никаких других операций с ними не проводилось. По p-й точке входа должно вычисляться одно значение выхода.

Во-вторых, значение M выбирается, как правило, от $5p$ до $10p$ или более в зависимости от необходимых характеристик в полосе заграждения (чем больше M, тем ниже значения коэффициента усиления в полосе заграждения).

В-третьих, вместо запоминания данных можно ограничиться только запоминанием *частных сумм*. Этот прием может сохранить значительный объем памяти.

Поясним, как это происходит. Главная экономия происходит за счет запоминания примерно M/p частных сумм вместо $5p-10p$ значений данных, которые запоминаются при обычных способах. Иначе говоря, если $M=rp$, то необходимо запоминание только r частных сумм. При этом для каждого нового значения входа выполняется r умножений этого значения на разные коэффициенты и каждое такое произведение добавляется к нужной частной сумме. Каждая частная сумма с p точками выступает в качестве значения выхода, и начинает вычисляться новая частная сумма.

Прием, в котором вычисляются частные суммы, позволяет сократить в p раз объем памяти, необходимый при других стандартных методах. Это имеет важное значение при обработке данных, поступающих по многим каналам.

5.6. ОБРАТНАЯ ДЕЦИМАЦИЯ

Иногда вместо уменьшения требуется увеличение скорости выборки. В частности, это может понадобиться для получения согласованных скоростей выборок двух несовпадающих временных рядов. Более подробно остановимся на этом в § 5.7. А сейчас покажем, как производится обратная децимация — процесс, при котором получаются новые значения между старыми. Эта операция включает в себя интерполяцию и выглядит следующим образом.

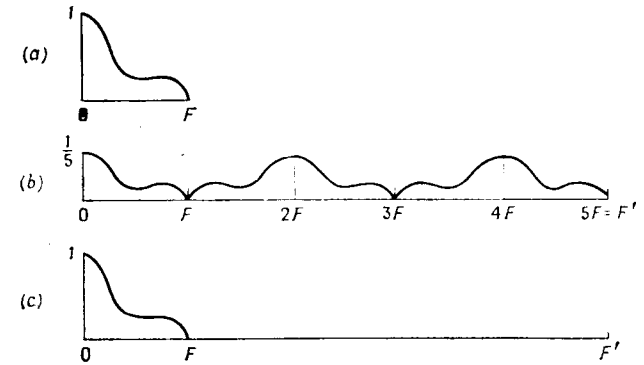

Рис. 5.13. Амплитуды спектров на разных стадиях обратной децимации функции данных.

Сначала между исходными данными вставляется необходимое число дополнительных нулей. При обратной децимации 1 к p между соседними данными вставляется $p-1$ нулей. Затем исходные данные умножаются на p. Наконец, новая последовательность исходных данных (умноженных на p) и нулей пропускается через фильтр низких частот, точка отсечки которого совпадает с исходной частотой Найквиста.

Предположим, например, что скорость выборки исходных данных равнялась 200 выб./с и требуется увеличить эту скорость до 1000 выб./с. Этому соответствует обратная децимация 1 к 5. Если исходные данные обозначить $x(0)$, $x(1)$, ..., то цифровая фильтрация будет проводиться с последовательностью $5x(0)$, 0, 0, 0, 0, $5x(1)$, 0, 0 и т. д.

Фильтр низких частот производит интерполяцию и уменьшение амплитуды данных. Из-за этого понижения мощности в слу-

чае обратной децимации в отличие от прямой децимации приходится умножать данные на p.

Получающуюся картину помогает понять рис. 5.13. Предположим, что амплитуда преобразования Фурье данных имеет вид, представленный на рис. 5.13, *а*. Добавление между исходными данными нулей приводит к изменениям, отраженным на рис. 5.13, *b*. Общая высота графика понизилась, а мощность периодически

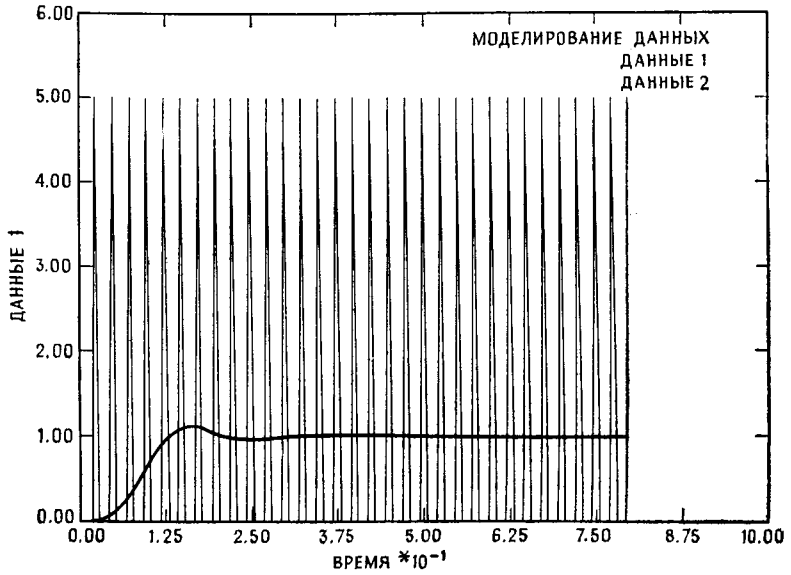

Рис. 5.14. Исходная функция и полученная из нее после обратной децимации 1 к 5. Функция в виде гребешка получена добавлением четырех нулей между единичными значениями исходной функции, умноженными на 5. Вторая функция получена после фильтрации с помощью подходящего цифрового фильтра.

распределилась на новом частотном диапазоне. Отметим, что на всех рисунках показан диапазон от 0 до частоты Найквиста. На рис. 5.13, *с* показан результат фильтрации, описанной выше. Вся мощность, располагавшаяся между F и F', пропала, поэтому никаких изменений из-за подмены частот в амплитудах не происходит.

В качестве второго примера приведенной процедуры рассмотрим последовательность данных до и после фильтрации, изображенную на рис. 5.14. Исходный ряд представлял собой последовательность из 33 точек, каждый элемент которой равнялся единице. Как и в предыдущем случае, индекс обратной децимации выбран равным 5. Между каждыми двумя единичными значениями при этой обратной децимации 1 к 5 вставляется по 4 нуля, а

сами значения умножаются на 5. При этом получается функция, показанная на рис. 5.14 (она имеет форму гребешка). Вторая функция на этом рисунке есть результат фильтрации гребнеобразной функции с помощью шестиполюсного фильтра низких частот.

Если исключить из рассмотрения начальный участок, соответствующий неустановившемуся режиму, то фильтрованные данные оказываются очень близкими к единице, т. е. данные снова имеют постоянное значение, равное единице.

5.7. СВЕДЕНИЕ К ОБЩЕЙ СКОРОСТИ ВЫБОРКИ

В некоторых задачах сравнения двух временных рядов (например, при вычислениях кросс-корреляции, кросс-спектра и передаточной функции) требуется получить для двух рядов одинаковую скорость выборки. Чтобы добиться такого согласования скоростей выборок, достаточно применить прямую или обратную децимации. Однако, как будет видно, следует соблюдать определенную осторожность, чтобы возникающие при этом потери информации не оказались слишком большими.

Предположим, что скорости выборок равны S_1 и S_2 соответственно. Рассмотрим два случая.

Первый случай. Пусть $S_1 = pS_2$, где p — целое число. В этом случае лучше всего провести децимацию первой последовательности с индексом p (т. е. оставить только p-ю часть исходной последовательности). Разумеется, можно сделать и иначе, т. е. провести обратную децимацию 1 к p второй последовательности. Однако этого не следует делать по двум причинам. Первая из них состоит в том, что число данных для обработки увеличится в p раз. Вторая причина связана с возможностью появления ложной информации о частотах выше $S_2/2$, не содержащейся в данных второй последовательности, если они были предварительно подвергнуты фильтрации.

Второй случай. В случае $S_1 \neq pS_2$ определяется наименьшее общее кратное S_1 и S_2, т. е. такое наименьшее число S_3, что и S_1, и S_2 делят его без остатка. Иначе говоря, существуют такие целые числа p и q, что $S_1 p = S_2 q = S_3$. Если S_1 и S_2 взаимно просты, то $p = S_2$ и $q = S_1$ (предельный случай).

После этого нужно выбрать большее из чисел p и q. Допустим, что это большее число есть p. Тогда для второй последовательности проводится обратная децимация 1 к q, а затем прямая децимация p к 1. Общая скорость выборки будет равна

$$\frac{S_2 q}{p} = \frac{S_3}{p} = \frac{S_1 p}{p} = S_1.$$

Предложенная процедура дает наименьшую из возможных скоростей выборки, что всегда желательно. Например, положим $S_1 = 800 = 2^5 \cdot 5^2$, $S_2 = 1000 = 2^3 \cdot 5^3$. Наименьшим общим кратным этих чисел будет $S_3 = 2^5 \cdot 5^3 = 4000$, т. е. $p = 5$, $q = 4$. Поэтому для второй последовательности нужно сначала провести обратную децимацию 1 к 4 (в результате получится скорость выборки 4000 выб./с), а затем провести прямую децимацию 5 к 1, чтобы получить скорость выборки, которая использовалась для получения первой последовательности.

Добавим, что при обеих децимациях фильтрации приводят к изменению фазового угла или к увеличению задержки, отвечающей второй последовательности. Это следует учитывать в тех задачах анализа, где фаза и задержка могут иметь значение, например в задачах анализа кросс-спектра.

5.8. КОМПЛЕКСНАЯ ДЕМОДУЛЯЦИЯ

Понятие комплексной демодуляции уже упоминалось в § 4.8 в связи с полоснопропускающими фильтрами. Собственно говоря, это понятие относится к умножению последовательности на комплексную экспоненту $\exp(j2\pi f_c iT)$. В результате такого умножения происходит сдвиг $X(f)$ на частоту f_c. Покажем, как это происходит. Пусть

$$X(f) = T \sum_{i=-\infty}^{\infty} x(i) \exp(-j2\pi f i T).$$

Определим последовательность $y(i)$ формулой

$$y(i) = x(i) \exp(j2\pi f_c i T).$$

Это равносильно умножению $x(i)$ на синус и косинус с частотой f_c Гц. Преобразование Фурье последовательности $y(i)$ в этом случае имеет вид

$$\begin{aligned} Y(s) &= T \sum_{i=-\infty}^{\infty} y(i) \exp(-j2\pi s i T) = \\ &= T \sum_{i=-\infty}^{\infty} x(i) \exp(j2\pi f_c i T) \exp(-j2\pi s i T) = \\ &= T \sum_{i=-\infty}^{\infty} x(i) \exp(-j2\pi (s - f_c) i T). \end{aligned}$$

Если переменную $s - f_c$ заменить величиной f, т. е. сделать замену $f = s - f_c$ (следовательно, $s = f + f_c$), то уравнение примет вид

$$Y(f + f_c) = T \sum_{i=-\infty}^{\infty} x(i) \exp(-j2\pi f i T) = X(f).$$

Таким образом, комплексной демодуляции во временной области отвечает сдвиг в частотной области.

Комплексная демодуляция обычно проводится при помощи двух фильтров, как это показано на рис. 5.15. В прямоугольнике справа вычисляются амплитуда и фаза входящих сигналов.

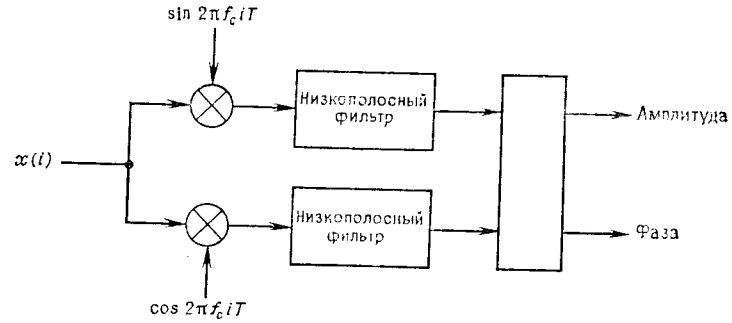

Рис. 5.15. Комплексная демодуляция.

Делается это следующим образом. Определим последовательности

$$x_s(i) = x(i) \sin(2\pi f_c i T), \quad x_c(i) = x(i) \cos(2\pi f_c i T)$$

и обозначим символом $y_s(i)$ последовательность значений, полученных фильтрацией из последовательности $x_s(i)$, а $y_c(i)$ — из последовательности $x_c(i)$. Последовательность значений амплитуды и фазы определим формулами

$$a(i) = \sqrt{2(y_s^2(i) + y_c^2(i))}, \quad \varphi(i) = \operatorname{arctg}\left(-\frac{y_s(i)}{y_c(i)}\right).$$

Действие комплексной демодуляции поясним на примере последовательности

$$x(i) = A \cos(2\pi f_c i T + \varphi_c),$$

где φ_c — произвольный фазовый угол. Тогда

$$\begin{aligned} x_s(i) &= A \cos(2\pi f_c i T + \varphi_c) \sin(2\pi f_c i T) = \\ &= \frac{A}{2} \{\sin[2\pi (f_c + f_c) i T + \varphi_c] + \sin[2\pi (f_c - f_c) i T - \varphi_c]\} = \\ &= \frac{A}{2} [\sin(4\pi f_c i T + \varphi_c) - \sin\varphi_c]. \end{aligned}$$

Если пренебречь задержкой по фазе, то для последовательности $y_s(i)$ получим выражение

$$y_s(i) = -\frac{A}{2} \sin\varphi_c$$

(слагаемое, соответствующее высокой частоте, при фильтрации исчезает). Аналогично

$$x_c(i) = A\cos(2\pi f_c iT + \varphi_c)\cos(2\pi f_c iT) =$$
$$= \frac{A}{2}\{\cos[2\pi(f_c + f_c)iT + \varphi_c] + \cos[2\pi(f_c - f_c)iT + \varphi_c]\} =$$
$$= \frac{A}{2}[\cos(4\pi f_c iT + \varphi_c) + \cos\varphi_c].$$

По тем же причинам, что и в случае $y_s(i)$, имеем

$$y_c(i) = \frac{A}{2}\cos\varphi_c,$$
$$y_s^2(i) + y_c^2(i) = \frac{A^2}{4}\sin^2\varphi_c + \frac{A^2}{4}\cos^2\varphi_c = \frac{A^2}{2}.$$

Поэтому

$$a(i) = \sqrt{2(y_s^2(i) + y_c^2(i))} = A$$

и

$$\varphi(i) = \operatorname{arctg}\left(-\frac{y_s(i)}{y_c(i)}\right) = \operatorname{arctg}\left(\frac{\sin\varphi_c}{\cos\varphi_c}\right) = \varphi_c.$$

Следует помнить, что задержка, которой пренебрегли, в фазовом угле присутствует. Несмотря на это, рассмотренная процедура используется в первую очередь для определения передаточных функций: на входе и выходе исследуемого устройства устанавливаются два одинаковых комплексных демодулятора. Коэффициент усиления определяется из отношения входной последовательности к выходной, а фаза — из разницы двух фазовых углов. Применение одинаковых фильтров в обоих комплексных демодуляторах вносит соответствующий фазовый угол.

Упражнения

5.1. Используя двойную точность в режиме плавающей точки, нетрудно смоделировать на Фортране операции, которые производятся при фильтрации в режиме фиксированной точки. Предположим, что числа с фиксированной точкой, для которых проводится моделирование, имеют m битов в дробной части. Тогда все вычисления, за исключением умножения и округления, могут непосредственно производиться с двойной точностью. Для указанных исключений нужно предусмотреть две подпрограммы.

Моделирование округления

```
SUBROUTINE SIMRND (AA,M)
DOUBLE PRECISION AA,A,FACT
FACT=2.D0**M
A=AA*FACT+0.5D0
A=A-DMOD (A,1.D0)
AA=A/FACT
RETURN
END
```


Моделирование умножения

```
SUBROUTINE SIMMLT (A,B,C,M)
DOUBLE PRECISION A,B,C,CC
CC=A*B/(2.D0**M)
CALL SIMRND (CC,M)
C=CC
RETURN
END
```

a) Моделируя вычисления с фиксированной точкой, при помощи приведенных подпрограмм повторите упр. 4.7 (b и c). Вычисления проведите при $M = 16$, 12, 8 и 4, $T = 0.005$ и только при одном значении $B = 4$ Гц. Для делений при вычислениях передаточной функции потребуется подпрограмма моделирования деления с фиксированной точкой SIMDIV. Напишите эту подпрограмму, просмотрев предварительно подпрограмму SIMMLT.

b) Параллельно вычислениям, моделирующим вычисления с фиксированной точкой, проведите те же вычисления с плавающей точкой. Используя результаты вычислений с плавающей точкой в качестве контрольных, определите погрешность. Для этого получите разности результатов для двух вариантов использованной арифметики, возведите эти разности в квадраты и осредните по всей длине вычислений. Прокомментируйте полученный для погрешности результат.

5.2. В этой задаче приводится способ, равнозначный методу измерения функции $\hat{H}(f)$, описанному под вторым номером в § 5.2. При решении этой задачи следует воспользоваться тем моделированием арифметики с фиксированной точкой, которое рассмотрено в предыдущем упражнении. Общая программа для решений должна включать в себя следующие элементы:

a) Подпрограмму SIMSIN для рекурсивного вычисления $\sin(2\pi f_p iT)$. При ее составлении нужно воспользоваться алгоритмом, описанным в § 4.3 в п. «Фильтр второго порядка как осциллятор». В этой программе следует использовать одинарную точность в режиме плавающей точки. Полученный выход должен преобразовываться в числа с двойной точностью и округляться при помощи подпрограммы SIMRND.

b) Подпрограмму SIMFIL для вычисления значений последовательности

$$y(i) = b_0 x(i) - a_1 y(i-1) - a_2 y(i-2)$$

моделируя действия с фиксированной точкой.

c) Управляющую программу, в которой предусмотрено следующее.

1) Ввод значения m или его задание в операторе данных.

2) Получение весов фильтра низких частот первого или второго порядка (в последнем случае обращением к LP2) и преобразование результатов с плавающей точкой в форму с фиксированной точкой, используя моделирование округления.

3) Использование подпрограммы SIMSIN для получения синусоиды заданной частоты и подпрограммы SIMFIL для фильтрации этой синусоиды.

4) Преобразование значений, полученных после достижения установившегося режима на выходе фильтра, в форму чисел с плавающей точкой и одинарной точностью; вычисление среднего квадрата мощности. Это дает коэффициент усиления и передаточную функцию фильтра в частоте, с которой генерировалась синусоида.

5) Вычисления по общей программе при значениях $T = 0.005$, $B = 4$, $f_p = 2$, 4, 8, 16, 32, 64 Гц, $m = 16$, 12, 8, 4.

Общую программу просчитайте и для фильтра первого, и для фильтра второго порядка. Объясните полученные результаты.

5.3. Устойчив ли фильтр

$$y(i) = 2x(i) - 4x(i-1) - 8x(i-2) + 2y(i-1) + 4y(i-2).$$

Объясните ответ. Бывают ли патологические случаи?

5.4. Веса биномиального фильтра M-го порядка (нерекурсивного) определяются формулой

$$b_k = \binom{M}{k} \Big/ 2^M, \quad \text{где} \quad k = -M, \ldots, M.$$

Покажите, что $|H(f)|^2 = \cos^{2M}(\pi f T)$. Получите выражение для 3-децибеловой точки (т. е. для такого значения f, что $10 \log_{10} |H(f)|^2 = 10 \log_{10} {}^1/_2$) через частоту и M. Что напоминает каскадная версия этого фильтра? Каковы нули этого фильтра?

5.5. Перепишите программу BFILT1. Используя ее, подпрограммы PRPLOT, LPSB и управляющую программу, аналогичную описанной в упр. 5.2, выполните следующие действия.

a) Фильтрацию последовательности $x(i) = 1, 0, 0, \ldots$, $i = 0, \ldots, 100$. Получите шесть таких последовательностей фильтрованных данных (по 101 точке в каждой), полагая во всех случаях $T = 0.005$.

b) Применяемый в каждом из шести случаев фильтр должен иметь шесть рекурсивных весов (три каскадные части). Точки отсечки должны быть равными 32, 16, 8, 4, 2, 1 Гц.

c) Прокомментируйте полученные результаты.

5.6. Напишите тестовую программу для сравнения эффективности подпрограмм BFILT2 и BFILT3. *Замечание*: на ЭВМ, которая предназначена для такой проверки, можно оценивать время работы обеих подпрограмм по-разному. Можно применить такую процедуру: получить по 1000 точек синусоиды; время фильтрации при помощи подпрограммы BFILT2 и время фильтрации при помощи подпрограммы BFILT3 выдать на печать. Если будет время, попытайтесь продолжить это сравнение и напишите подпрограмму BFILT3 на машинном языке. Просчитайте временной тест в этом случае. Это не простая задача, поэтому решать ее необязательно, хотя и желательно.

5.7. Напишите и проверьте работу подпрограммы BFILT4 для биномиального фильтра.

a) Положите $M = 8$, $T = 0.005$, $N = 1016$.

b) Получите и отфильтруйте синусоиду длины N. Отбросив первые и последние M точек, вычислите среднее и дисперсию значений выхода.

c) Частоту синусоиды f возьмите равной 9.3, 18.6, 37.2, 74.4 Гц. Прокомментируйте результаты.

5.8. Используя веса, полученные с помощью подпрограммы LPSPBG (рис. 4.17), повторите упр. 5.7, полагая $M = 8$, $T = 0.005$ и $B = 18.6$. Прокомментируйте результаты. Далее, вычислите передаточные функции обоих фильтров и сравните их. Что можно взять в качестве альтернативы подпрограммы TTRAN, которую в данном случае применять нельзя? Можно ли в этом случае выбрать FFTRAN? Если да, то почему?

5.9. Напишите тестовую программу для изучения вопросов, связанных с децимацией. В ней должно быть предусмотрено следующее.

a) Получение с помощью подпрограммы TDRAND 1024 значений данных, имеющих равномерное распределение. Из каждого значения должно вычитаться 0.5, чтобы среднее значение было равно 0.

b) Фильтрация данных с помощью подпрограмм BFILT1 и LPSB. Предполагается, что $T = 0.005$, фильтр имеет шесть полюсов и $B = 20$ Гц.

c) Децимация исходных и фильтрованных данных 4 к 1 ($p = 4$).

d) Вычисление для обеих последовательностей преобразования Фурье. (*Замечание*: для этого отдельно к каждой последовательности примените подпрограмму FFTRAN, задавая в качестве действительной части члены последовательности, а в качестве мнимой — нули. Существует и более эффективный способ такой операции, но он будет затронут только в следующей главе.) Затем — вычисление квадратов абсолютных величин полученных значений преобразований Фурье.

В каждом преобразовании Фурье должно оставаться по 129 точек. Почему? Получите графическое изображение результатов и сравните их.

Повторите действия при $B = 12.5$ и 25 Гц. Сравните полученные результаты со случаем 20 Гц.

5.10. Предположим, что требуется децимация 10 000 к 1. Попытайтесь для этой цели воспользоваться подходящим шестиполюсным фильтром, полученным при помощи подпрограммы LPSB. Удовлетворительны ли результаты? Что может служить разумной альтернативой?

5.11. Получите 64 точки равномерно распределенного псевдослучайного шума с помощью подпрограммы TDRAND. Проведите обратную децимацию 1 к 4, включающую необходимую операцию фильтрации. Прокомментируйте результаты. Вычислите и получите в виде графика квадраты абсолютных значений преобразования Фурье последовательности данных, подвергнутых обратной децимации, до и после фильтрации. Прокомментируйте результаты.