CHAPTER 33

The z-Transform
Z-преобразование

Just as analog filters are designed using the Laplace transform, recursive digital filters are developed with a parallel technique called the z-transform. The overall strategy of these two transforms is the same: probe the impulse response with sinusoids and exponentials to find the system's poles and zeros. The Laplace transform deals with differential equations, the s-domain, and the s-plane. Correspondingly, the z-transform deals with difference equations, the z-domain, and the z-plane. However, the two techniques are not a mirror image of each other; the s-plane is arranged in a rectangular coordinate system, while the z-plane uses a polar format. Recursive digital filters are often designed by starting with one of the classic analog filters, such as the Butterworth, Chebyshev, or elliptic. A series of mathematical conversions are then used to obtain the desired digital filter. The z-transform provides the framework for this mathematics. The Chebyshev filter design program presented in Chapter 20 uses this approach, and is discussed in detail in this chapter.

Также, как аналоговые фильтры разработаны, используя преобразование Лапласа, рекурсивные цифровые фильтры разработаны с параллельной методикой называемой z-трансформантой. Полная стратегия из этих двух трансформант - тот же самый: зондируйте импульсную передаточную функцию с синусоидами и показательными функциями, чтобы найти полюса системы и нули. Преобразование Лапласа имеет дело с дифференциальными уравнениями, s-доменом, и s-плоскостью. Соответственно, z-трансформанта имеет дело с дифференциальными уравнениями, z-доменом, и z-плоскостью. Однако, два метода - не зеркальное изображение друг друга; s-плоскость размещается в прямоугольной системе координат, в то время как z-плоскость использует полярный формат. Рекурсивные цифровые фильтры часто разрабатываются, начинаясь с одного из классических аналоговых фильтров, типа Буттерворта, Чебышевского, или эллиптического. Ряд математических преобразований тогда используется, чтобы получить желательный цифровой фильтр. Z-трансформанта обеспечивает структуру для этой математики. Программа проекта фильтра Чебышева, представленная в главе 20 использует этот подход, и обсуждена подробно в этой главе.

The Nature of the z-Domain
Характер(природа) z-домена

To reinforce that the Laplace and z-transforms are parallel techniques, we will start with the Laplace transform and show how it can be changed into the z-transform. From the last chapter, the Laplace transform is defined by the relationship between the time domain and s-domain signals:

Чтобы укрепить это, преобразования Лапласа и z-преобразования - параллельные методы, мы начнем с преобразования Лапласа и покажем, как это может быть изменено в z-трансформанту. В прошлой главе, преобразование Лапласа определено отношениями между сигналами домена времени и s-домена:

$$X(s) = \int_{t=-\infty}^{\infty} x(t)\, e^{-st}\, dt$$

where $x(t)$ and $X(s)$ are the time domain and s-domain representation of the signal, respectively. As discussed in the last chapter, this equation analyzes the time domain signal in terms of sine and cosine waves that have an exponentially changing amplitude. This can be understood by replacing the complex variable, s, with its equivalent expression, $\sigma + j\omega$. Using this alternate notation, the Laplace transform becomes:

Где $x(t)$ и $X(s)$ - представление сигнала домена времени и s-домена, соответственно. Как обсуждено в прошлой главе, это уравнение анализирует сигнал домена времени в терминах синуса волн и косинуса, которые имеют амплитуду, изменяющуюся по экспоненте. Это может быть понято, заменяя комплексную переменную, s, ее эквивалентным выражением, $\sigma + j\omega$. Использование этой альтернативной системы обозначений, преобразование Лапласа становится:

$$X(\sigma,\omega) = \int_{t=-\infty}^{\infty} x(t)\, e^{-\sigma t} e^{-j\omega t}\, dt$$

If we are only concerned with *real* time domain signals (the usual case), the top and bottom halves of the s-plane are mirror images of each other, and the term, $e^{-j\omega t}$, reduces to simple cosine and sine waves. This equation identifies each *location* in the s-plane by the two parameters, σ and ω. The *value* at each location is a complex number, consisting of a real part and an imaginary part. To find the real part, the time domain signal is multiplied by a cosine wave with a frequency of ω, and an amplitude that changes exponentially according to the decay parameter, σ. The value of the real part of $X(\sigma, \omega)$ is then equal to the integral of the resulting waveform. The value of the imaginary part $X(\sigma, \omega)$ of is found in a similar way, except using a sine wave. If this doesn't sound very familiar, you need to review the previous chapter before continuing.

Если мы заинтересованы только сигналами домена реального времени (обычный случай), верхние и нижние половины s-плоскости - зеркальные изображения друг друга, и термина, $e^{-j\omega t}$, приводит к простым волнам косинуса и синуса. Это уравнение идентифицирует каждое расположение в s-плоскости этими двумя параметрами, σ и ω. Значение в каждом расположении - комплексное число, состоящее из вещественной части и мнимой части. Чтобы найти вещественную часть, сигнал домена времени умножен на волну косинуса с частотой ω, и амплитудой, которая изменяется по экспоненте согласно параметру распада, σ. Значение вещественной части $X(\sigma, \omega)$ тогда равно интегралу заканчивающейся формы волны. Значение мнимой части $X(\sigma, \omega)$ найдено подобным способом, кроме использования волны синуса. Если это не звучит очень знакомым, Вы должны делать обзор предыдущей главы перед продолжением.

The Laplace transform can be changed into the z-transform in three steps. The first step is the most obvious: change from continuous to discrete signals. This is done by replacing the time variable, t, with the sample number, n, and changing the integral into a summation:

Преобразование Лапласа может быть изменено в z-трансформанту в трех шагах. Первый шаг наиболее очевиден: изменение непрерывных сигналов в дискретные. Это сделано, заменяя переменную времени, t, выборкой с номером, n, и заменяя интегрирование в суммированием:

$$X(\sigma,\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-\sigma n} e^{-j\omega n}$$

Notice that $X(\sigma, \omega)$ uses parentheses, indicating it is *continuous*, not discrete. Even though we are now dealing with a discrete time domain signal, $x[n]$, the parameters σ and ω can still take on a continuous range of values. The second step is to rewrite the exponential term. An exponential signal can be mathematically represented in either of two ways:

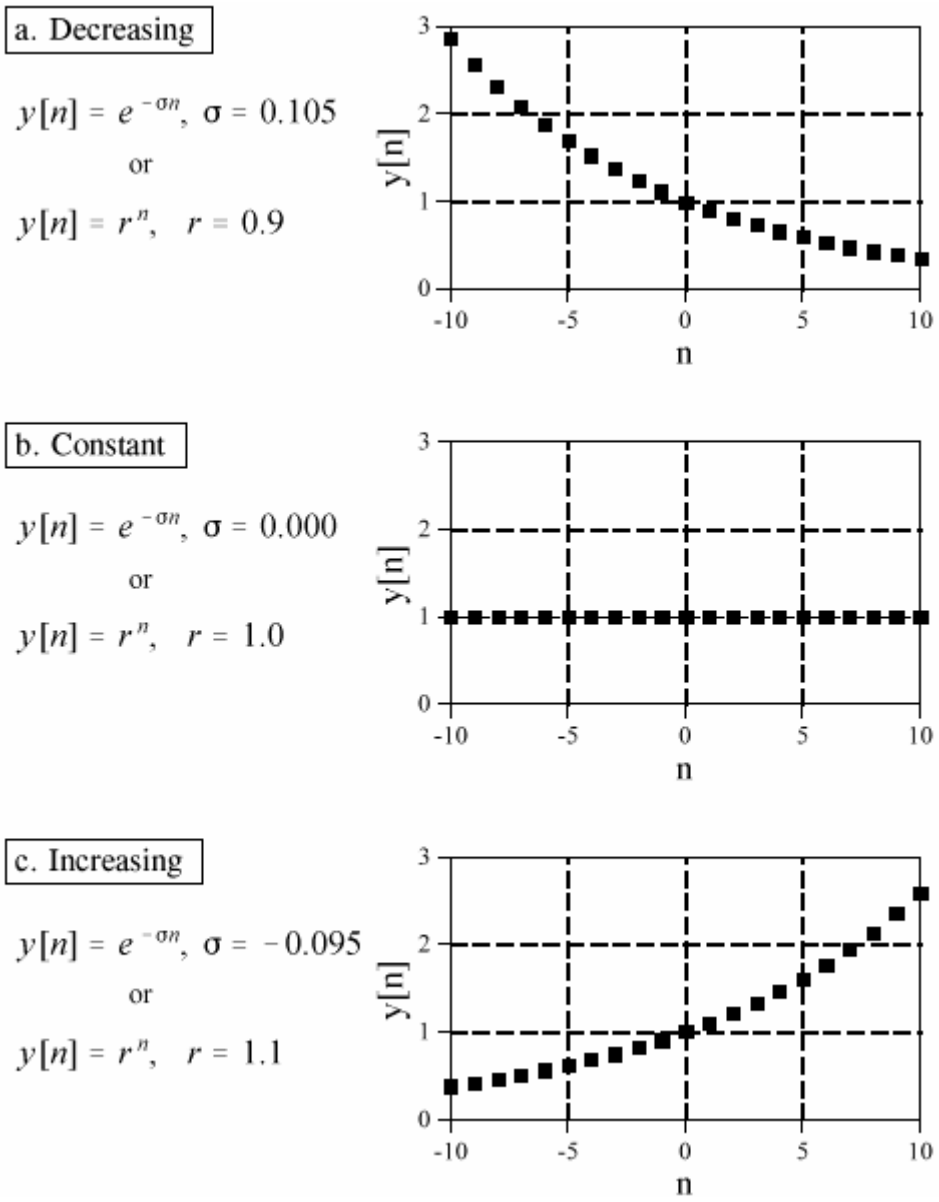
Обратите внимание, $X(\sigma, \omega)$ использует круглые скобки, указывая это что сигнал непрерывен, а не дискретен. Даже притом, что мы теперь имеем дело с дискретным сигналом домена времени, $x[n]$, параметры σ и ω могут принимать непрерывный диапазон значений. Второй шаг должен перезаписать экспоненциальный термин. Показательный сигнал может быть математически представлен одним из двух способов:

$$y[n] = e^{-\sigma n} \quad \text{or} \quad y[n] = r^n$$

As illustrated in Fig. 33-1, both these equations generate an exponential curve. The first expression controls the decay of the signal through the parameter, σ. If σ is positive, the waveform will *decrease* in value as the sample number, n, becomes larger. Likewise, the curve will progressively *increase* if σ is negative. If σ is exactly zero, the signal will have a constant value of *one*.

Как иллюстрировано в рис. 33-1, оба эти уравнения генерируют экспоненциальную кривую. Первое выражение управляет распадом сигнала через параметр, σ. Если σ является положительной, форма волны *уменьшится* в значении как выборка номер, n, становится большей. Аналогично, кривая прогрессивно *увеличится*, если σ является отрицательной. Если σ является точно нулем, сигнал будет иметь постоянное значение *единицы*.

РИСУНОК 33-1
Экспоненциальные сигналы. Показательные функции могут быть представлены в двух различных математических формах. Преобразование Лапласа использует один путь, в то время как z-трансформанта использует другой.



The second expression uses the parameter, *r*, to control the decay of the waveform. The waveform will decrease if $r < 1$, and increase if $r > 1$. The signal will have a constant value when $r = 1$. These two equations are just different ways of expressing the same thing. One method can be swapped for the other by using the relation:

Второе выражение использует параметр, *r*, управляющий распадом формы волны. Форма волны уменьшится, если $r < 1$, и увеличивается, если $r > 1$. Сигнал будет иметь постоянное значение когда $r = 1$. Эти два уравнения - только различные пути выражения той же самой вещи. Один метод может меняться для другого, используя отношение:

$$r^n = \left[e^{\ln(r)}\right]^n = e^{n\ln(r)} = e^{-\sigma n}$$

The second step of converting the Laplace transform into the z-transform is completed by using the *other* exponential form:

Второй шаг преобразования трансформанты Лапласа в z-трансформанту закончен, используя другую показательную форму:

$$r^n = [e^{\ln(r)}]^n = e^{n\ln(r)} = e^{-\sigma n}$$

where: $\sigma = -\ln(r)$

While this is a perfectly correct expression of the z-transform, it is not in the most compact form for complex notation. This problem was overcome in the Laplace transform by introducing a new complex variable, *s*, defined to be: $s = \sigma + j\omega$. In this same way, we will define a new variable for the z- transform:

В то время как это - совершенно правильное выражение z-трансформанты, это не в наиболее компактной форме для комплексной системы обозначений. Эта проблема была преодолена в преобразовании Лапласа, представляя новую комплексную переменную, *s*, определена, чтобы быть: $s = \sigma + j\omega$. Этим же способом, мы определим новую переменную для z-трансформанты:

$$z = re^{-j\omega}$$

This is defining the complex variable, *z*, as the polar notation combination of the two real variables, *r* and ω. The third step in deriving the z-transform is to replace: *r* and ω, with *z*. This produces the standard form of the z-transform:

Это определяет комплексную переменную, *z*, как комбинацию полярной системы обозначений из двух вещественных переменных, *r* и ω. Третьим шагом в получение z-трансформанты должно заменить: *r* и ω, на *z*. Это производит стандартную форму z-трансформанты:

УРАВНЕНИЕ 33-1. Z-трансформанта.
Z-трансформанта определяет отношения между сигналом домена времени, $x[n]$, и сигналом z-домена, $X[n]$.

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$$

Why does the z-transform use r^n instead of $e^{-\sigma n}$, and *z* instead of *s*? As described in Chapter 19, recursive filters are implemented by a set of *recursion coefficients*. To analyze these systems in the z-domain, we must be able to convert these recursion coefficients into the z-domain *transfer function*, and back again. As we will show shortly, defining the z-transform in this manner (r^n and *z*) provides the simplest means of moving between these two important representations. In fact, defining the z-domain in this way makes it *trivial* to move from one representation to the other.

Почему z-трансформанта использует r^n вместо $e^{-\sigma n}$, и *z* вместо *s* ? Как описано в главе 19, рекурсивные фильтры осуществлены набором *коэффициентов рекурсии*. Чтобы анализировать эти системы в z-домене, мы должны быть способны преобразовать эти коэффициенты рекурсии в *функцию преобразования(передаточную функцию)* z-домена, и опять назад. Как мы покажем вскоре, определение z-трансформанты этим способом (r^n и *z*) обеспечивает самые простые средства перемещения между этими двумя важными представлениями. Фактически, определение z-домена таким образом, делает тривиальным двигаться от одного представления до другого.

Figure 33-2 illustrates the difference between the Laplace transform's s-plane, and the z-transform's z-plane. Locations in the s-plane are identified by two parameters: σ, the exponential decay variable along the horizontal axis, and ω, the frequency variable along the vertical axis. In other words, these two real parameters are arranged in a *rectangular* coordinate system. This geometry results from defining s, the complex variable representing position in the s-plane, by the relation: $s = \sigma + j\omega$.

Рисунок 33-2 иллюстрирует разность между s-плоскостью преобразования Лапласа, и z-плоскостью z-трансформанты. Расположения в s-плоскости идентифицированы двумя параметрами: σ, переменная экспоненциального распада по горизонтальной оси, и ω, частотная переменная по вертикальной оси. Другими словами, эти два вещественных параметра размещаются в прямоугольной системе координат. Эта геометрия следствие определения s, комплексная переменная, представляющая позицию в s-плоскости, отношением: $s = \sigma + j\omega$.

In comparison, the z-domain uses the variables: r and ω, arranged in *polar* coordinates. The distance from the origin, r, is the value of the exponential decay. The angular distance measured from the positive horizontal axis, ω, is the frequency. This geometry results from defining z by: $z = re^{-j\omega}$. In other words, the complex variable representing position in the z-plane is formed by combining the two real parameters in a polar form.

Для сравнения, z-домен использует переменные: r и ω, размещаемые в полярных координатах. Расстояние от начала координат, r, является значением экспоненциального распада. Угловое расстояние, измеренное от положительной горизонтальной оси, ω, является частотой. Эта геометрия следствия определения z: $z = re^{-j\omega}$. Другими словами, комплексная переменная, представляющая позицию в z-плоскости сформирована, объединяя два вещественных параметра в полярной форме.

These differences result in vertical *lines* in the s-plane matching *circles* in the z-plane. For example, the s-plane in Fig. 33-2 shows a pole-zero pattern where all of the poles & zeros lie on vertical lines. The equivalent poles & zeros in the z-plane lie on circles concentric with the origin. This can be understood by examining the relation presented earlier: $\sigma = -\ln(r)$. For instance, the s-plane's vertical axis (i.e., $\sigma = 0$) corresponds to the z-plane's **unit circle** (that is $r = 1$). Vertical lines in the left half of the s-plane correspond to circles inside the z-plane's unit circle. Likewise, vertical lines in the right half of the s-plane match with circles on the outside of the z-plane's unit circle. In other words, the left and right sides of the s-plane correspond to the interior and the exterior of the unit circle, respectively. For instance, a continuous system is unstable when poles occupy the *right half* of the s-plane. In this same way, a discrete system is unstable when poles are *outside* the unit circle in the z-plane. When the time domain signal is completely real (the most common case), the upper and lower halves of the z-plane are mirror images of each other, just as with the s-domain.

Эти различия приводят к вертикальным *строкам* в s-плоскости, соответствующим *кругам* в z-плоскости. Для примера, s-плоскость в рис. 33-2 показывает образец нуль-полюса, где все полюсы и нули лежат на вертикальных строках. Эквивалентные полюса и нули в z-плоскости лежат на кругах, концентрических с началом координат. Это может быть понято, исследуя отношение, представленное ранее: $\sigma = -\ln(r)$. Например, вертикальная ось s-плоскости (то есть, $\sigma = 0$) соответствует **модулю круга** в z-плоскости (которая является $r = 1$). Вертикальные строки в левой половине s-плоскости соответствуют кругам внутри круга модуля z-плоскости. Аналогично, вертикальные строки в правой половине s-плоскости соответствуют кругам на внешней стороне модуля круга z-плоскости. Другими

словами, левые и правые стороны s-плоскости соответствуют внутренней области и внешней области модулю круга, соответственно. Например, непрерывная система непостоянна, когда полюса занимают *правую половину* s-плоскости. В этом тот же самый путь, дискретная система непостоянна, когда полюса - *вне* модуля круга в z-плоскости. Когда сигнал домена времени полностью вещественный (наиболее обычный случай), верхние и нижние половины z-плоскости - зеркальные изображения друг друга, так же, как с s-доменом.

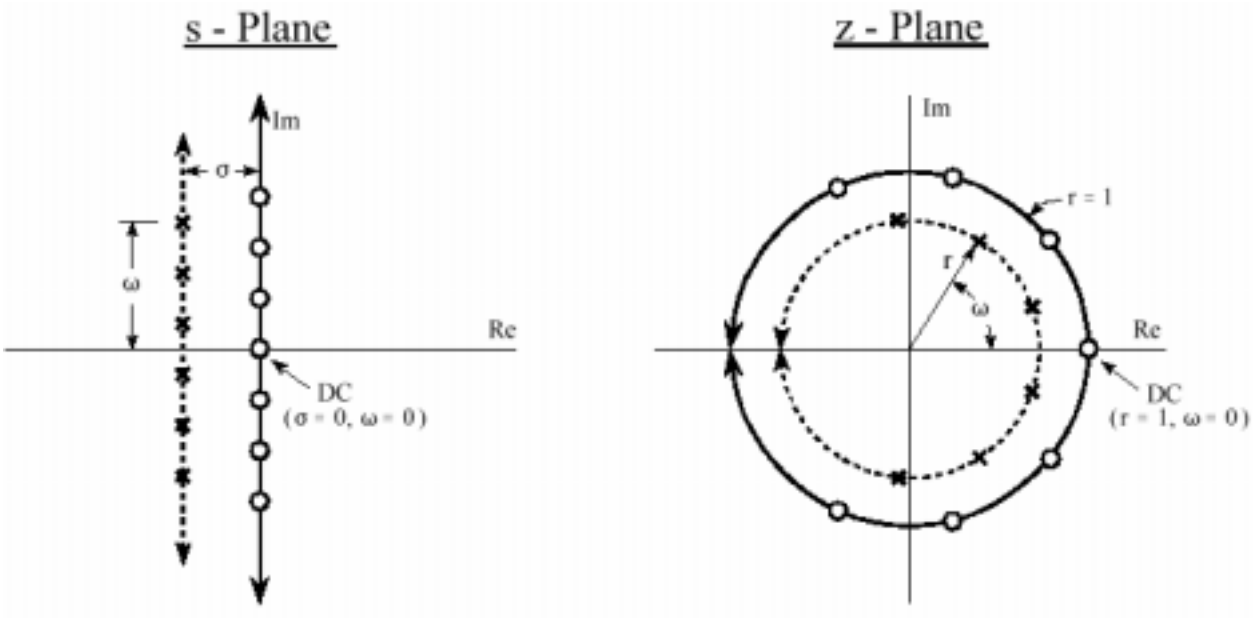


FIGURE 33-2
Relationship between the s-plane and the z-plane. The s-plane is a rectangular coordinate system with σ expressing the distance along the real (horizontal) axis, and ω the distance along the imaginary (vertical) axis. In comparison, the z-plane is in polar form, with r being the distance to the origin, and ω the angle measured to the positive horizontal axis. Vertical lines in the s-plane, such as illustrated by the example poles and zeros in this figure, correspond to circles in the z-plane.

РИСУНОК 33-2
Отношения между s-плоскостью и z-плоскостью. S-плоскость - прямоугольная система координат с σ выражение расстояния по вещественной (горизонтальной) оси, и ω расстояние по мнимой (вертикальной) оси. Для сравнения, z-плоскость находится в полярной форме, с r, являющимся расстоянием к началу координат, и ω угол, измеренный к положительной горизонтальной оси. Вертикальные строки в s-плоскости, типа иллюстрированного полюсами примера и нулями в этом рисунке, соответствуют кругам в z-плоскости.

Pay particular attention to how the frequency variable, ω, is used in the two transforms. A *continuous* sinusoid can have any frequency between DC and infinity. This means that the s-plane must allow ω to run from negative to positive infinity. In comparison, a *discrete* sinusoid can only have a frequency between DC and one-half the sampling rate. That is, the frequency must between 0 and 0.5 when expressed as a fraction of the sampling rate, or between 0 and π when expressed as a natural frequency (i.e., $\omega = 2\pi f\omega$). This matches the geometry of the z-plane when we interpret to be an angle expressed in *radians*. That is, the positive frequencies correspond to angles of 0 to π radians, while the negative frequencies correspond to 0 to $-\pi$ radians. Since the z-plane express frequency in a different way than the s-plane, some authors use different symbols to distinguish the two. A common notation is to use Ω (an upper case omega) to represent frequency in the z-domain, and ω (a lower case omega) for frequency in the s-domain. In this book we will use ω to represent both types of frequency, but look for this in other DSP material.

Обратите особое внимание на то, как частотная переменная, ω, используется в двух трансформантах. Непрерывная синусоида может иметь любую частоту между постоянным током и бесконечностью. Это означает, что s-плоскость должна позволить ω работать от

отрицательной до положительной бесконечности. Для сравнения, дискретная синусоида может только иметь частоту между постоянным током и половиной частоты выборки. То есть частота должна между 0 и 0.5 когда выражено как дробь частоты выборки, или между 0 и ω, когда выражено как собственная частота (то есть, $\omega = 2\pi f\omega$). Это соответствует геометрии z-плоскости, когда мы интерпретируем, чтобы быть углом, выраженным в радианах. То есть положительные частоты соответствуют углам ω к π радиан, в то время как отрицательные частоты соответствуют 0 к - π Радианы. Так как z-плоскость выражает частоту различным способом, чем s-плоскость, некоторые авторы используют различные символы, чтобы отличать их. Общая система обозначений должна использовать Ω (омега верхнего регистра) чтобы представить частоту в z-домене, и ω (омега строчных букв) для частоты в s-домене. В этой книге мы будем использовать ω для представления обоих типов частоты, но искать(отыскивать; обращать на это внимание?) это в другом материале ЦОС.

In the s-plane, the values that lie along the vertical axis are equal to the frequency response of the system. That is, the Laplace transform, evaluated at $\sigma = 0$, is equal to the Fourier transform. In an analogous manner, the frequency response in the z-domain is found along the unit circle. This can be seen by evaluating the z-transform (Eq. 33-1) at $r = 1$, resulting in the equation reducing to the Discrete Time Fourier Transform (DTFT). This places zero frequency (DC) at a value of *one* on the horizontal axis in the s-plane. The spectrum's positive frequencies are positioned in a counter-clockwise pattern from this DC position, occupying the upper semicircle. Likewise the negative frequencies are arranged from the DC position along the clockwise path, forming the lower semicircle. The positive and negative frequencies in the spectrum meet at the common point $\omega = \pi$ of and $\omega = -\pi$. This circular geometry also corresponds to the frequency spectrum of a discrete signal being *periodic*. That is, when the frequency angle is increased beyond π, the same values are encountered as between 0 and π. When you run around in a circle, you see the same scenery over and over.

В s-плоскости, значения, которые лежат по вертикальной оси, равны частотной характеристике системы. То есть преобразование Лапласа, оцененное в $\sigma = 0$, является равным преобразованию Фурье. Аналогичным способом, частотная характеристика в z-домене найдена по кругу модуля. Это может быть замечено, оценивая z-трансформанту (уравнение 33-1) в $r = 1$, приводя к сокращению уравнения Преобразования Фурье Дискретным Временем (DTFT). Это размещает нулевую частоту (постоянный ток) в значение *один* на горизонтальной оси в s-плоскости. Положительные частоты спектра позиционированы в противоположном часовой стрелке направлении от этой позиции постоянного тока, занимая верхний полукруг. Аналогично отрицательные частоты размещаются от позиции постоянного тока по часовой стрелке пути, формируя более низкий полукруг. Положительные и отрицательные частоты в спектре встречаются в общей точке $\omega = \pi$ и $\omega = -\pi$. Эта круговая(кольцевая) геометрия также соответствует спектру частот дискретного сигнала, являющегося *периодическим*. То есть когда частотный угол увеличен вне π, с теми же самыми значениями сталкиваются как между 0 и π. Когда Вы работаете вокруг в круге, Вы видите тот же самый пейзаж много раз.

Analysis of Recursive Systems
Анализ Рекурсивных Систем

As outlined in Chapter 19, a recursive filter is described by a **difference equation**:
Как выделено в главе 19, рекурсивный фильтр описан **дифференциальным уравнением**:

УРАВНЕНИЕ 33-2
Дифференциальное уравнение. См. главу 19 для подробностей.

$$y[n] = a_0x[n] + a_1x[n-1] + a_2x[n-2] + \cdots + b_1y[n-1] + b_2y[n-2] + b_3y[n-3] + \cdots$$

where *x*[ ] and *y*[ ] are the input and output signals, respectively, and the "*a*" and "*b*" terms are the **recursion coefficients**. An obvious use of this equation is to describe how a programmer would implement the filter. An equally important aspect is that it represents a mathematical relationship between the input and output that must be continually satisfied. Just as continuous systems are controlled by *differential* equations, recursive discrete systems operate in accordance with this *difference* equation. From this relationship we can derive the key characteristics of the system: the impulse response, step response, frequency response, pole-zero plot, etc.

Где *x*[ ] и *y*[ ] - сигналы ввода и вывода, соответственно, и термины "a", и "b" - коэффициенты рекурсии. Очевидное использование этого уравнения должно описать, как программист осуществил бы фильтр. Одинаково важный аспект - то, что это представляет математические отношения между вводом и выводом, который должен непрерывно удовлетворяться. Так же, как непрерывные системы управляются *дифференциальными* уравнениями, рекурсивные дискретные системы работают в соответствии с этим *дифференциальным* уравнением. От этих отношений мы можем получать ключевые характеристики системы: импульсная передаточная функция, реакция на скачок(переходная характеристика), частотная характеристика, график нуля полюса, и т.д.

We start the analysis by taking the z-transform (Eq. 33-1) of both sides of Eq. 33-2. In other words, we want to see what this controlling relationship looks like in the z-domain. With a fair amount of algebra, we can separate the relation into: *Y*[*z*]/*X*[*z*], that is, the z-domain representation of the output signal divided by the z-domain representation of the input signal. Just as with the Laplace transform, this is called the **system's transfer function**, and designate it by *H*[*z*]. Here is what we find:

Мы начинаем анализ, беря z-трансформанту (уравнение 33-1) обеих сторон уравнения 33-2. Другими словами, мы хотим видеть то, что эти отношения управления напоминают в z-домене. Со справедливым количеством алгебры, мы можем отделить отношение в: *Y*[*z*]/*X*[*z*], то есть представление z-домена сигнала выхода, разделенного представлением z-домена входного сигнала. Так же, как с преобразованием Лапласа, это называется **функцией преобразования системы**, и определять это *H*[*z*]. Имеется то, что мы находим:

EQUATION 33-3
Transfer function in polynomial form. The recursion coefficients are directly identifiable in this relation.

$$H[z] = \frac{a_0 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3} + \cdots}{1 - b_1z^{-1} - b_2z^{-2} - b_3z^{-3} - \cdots}$$

УРАВНЕНИЕ 33-3
Функция преобразования в полиномиальной форме. Коэффициенты рекурсии непосредственно идентифицируемы в этом отношении.

This is one of two ways that the transfer function can be written. This form is important because it directly contains the recursion coefficients. For example, suppose we know the recursion coefficients of a digital filter, such as might be provided from a design table:

Это - один из двух путей, которыми функция преобразования может быть написана. Эта форма важна, потому что это непосредственно содержит коэффициенты рекурсии. На-

пример, предположите, что мы знаем, коэффициенты рекурсии цифрового фильтра, типа(такие как) можно было бы обеспечивать от таблицы проекта:

a0 = 0.389	
a1 = -1.558	b1 = 2.161
a2 = 2.338	b2 = -2.033
a3 = -1.558	b3 = 0.878
a4 = 0.389	b4 = -0.161

Without having to worry about nasty complex algebra, we can directly write down the system's transfer function:

Без того, чтобы иметь необходимость, чтобы волноваться относительно противной комплексной алгебры, мы можем непосредственно записывать функцию преобразования системы:

$$H[z] = \frac{0.389 - 1.558z^{-1} + 2.338z^{-2} - 1.558z^{-3} + 0.389z^{-4}}{1 - 2.161z^{-1} + 2.033z^{-2} - 0.878z^{-3} + 0.161z^{-4}}$$

Notice that the "b" coefficients enter the transfer function with a *negative* sign in front of them. Alternatively, some authors write this equation using additions, but change the sign of all the "b" coefficients. Here's the problem. If you are given a set of recursion coefficients (such as from a table or filter design program), there is a 50-50 chance that the "b" coefficients will have the opposite sign from what you expect. If you don't catch this discrepancy, the filter will be grossly unstable.

Обратите внимание, что коэффициенты "b" вводят функцию преобразования типа с отрицательным знаком перед ними. Альтернативно, некоторые авторы записывают это уравнение, используя сложение, но изменяют знак всех коэффициентов "b". Имеется проблема. Если Вам дают набор коэффициентов рекурсии (типа от таблицы или программы проекта фильтра), имеется шанс 50 на 50, что коэффициенты "b" будут иметь противоположный знак, от того, что Вы ожидаете. Если Вы не уловите это несоответствие, фильтр будет чрезвычайно непостоянен.

Equation 33-3 expresses the transfer function using *negative* powers of z, such as: z^{-1}, z^{-2}, z^{-3}, etc. After an actual set of recursion coefficients have been plugged in, we can convert the transfer function into a more conventional form that uses *positive* powers: i.e., z, z^2, z^3 , … . By multiplying both the numerator and denominator of our example by z^4, we obtain:

Уравнение 33-3 экспресс-функция преобразования типа, используя отрицательные степени z, типа: z^{-1}, z^{-2}, z^{-3}, и т.д. После того, как фактический набор коэффициентов рекурсии был подключен в, мы можем преобразовывать функцию преобразования(передаточную функцию) в более обычную форму, которая использует положительные степени: то есть, z, z^2, z^3 , … . Умножая и числитель и знаменатель нашего примера на z^4, мы получаем:

$$H[z] = \frac{0.389z^4 - 1.558z^3 + 2.338z^2 - 1.558z + 0.389}{z^4 - 2.161z^3 + 2.033z^2 - 0.878z + 0.161}$$

Positive powers are often easier to use, and they are *required* by some z-domain techniques. Why not just rewrite Eq. 33-3 using positive powers and forget about negative powers entirely? We can't! The trick of dividing the numerator and denominator by the highest power of *z* (such as in our *z* 4 example) can only be used if the number of recursion coefficients is already known. Equation 33-3 is written for an *arbitrary* number of coefficients. The point is, both positive and negative powers are routinely used in DSP and you need to know how to convert between the two forms.

Положительные степени часто более легкие для использования, и они требуются некоторыми методами z-домена. Почему не перезаписывать уравнение 33-3, используя только положительные степени, и, полностью забывая относительно отрицательных степеней? Мы не можем! Уловка деления числителя и знаменателя самой высокой степенью *z* (типа z^4 в нашем примере) может использоваться, если только число коэффициентов рекурсии уже известно. Уравнение 33-3 написано в произвольном числе коэффициентов. Пункт, и положительные и отрицательные степени обычно используются в ЦОС, и Вы должны знать, как преобразовать между двумя формами.

The transfer function of a recursive system is useful because it can be manipulated in ways that the recursion coefficients cannot. This includes such tasks as: combining cascade and parallel stages into a single system, designing filters by specifying the pole and zero locations, converting analog filters into digital, etc. These operations are carried out by algebra performed in the s-domain, such as: multiplication, addition, and factoring. After these operations are completed, the transfer function is placed in the form of Eq. 33-3, allowing the new recursion coefficients to be identified.

Функция преобразования рекурсивной системы полезна, потому что это может управляться способами, которыми коэффициенты рекурсии не могут. Это включает такие задачи как: объединение каскада и параллельных стадий в единственную(отдельную) систему, проектирование фильтров, определяя полюс и нулевые расположения, преобразовывая аналоговые фильтры в цифровые, и т.д. Эти операции выполнены алгеброй, выполненной в s-домене, типа: умножение, сложение, и разложение на множители. После того, как эти операции закончены, функция преобразования помещена в форму уравнения 33-3, позволяя новым коэффициентам рекурсии быть идентифицированными.

Just as with the s-domain, an important feature of the z-domain is that the transfer function can be expressed as **poles** and **zeros**. This provides the second general form of the z-domain:

EQUATION 33-4
Transfer function in pole-zero form.

УРАВНЕНИЕ 33-4
Функция преобразования в форму нуля полюса.

$$H[z] = \frac{(z - z_1)(z - z_2)(z - z_3)\cdots}{(z - p_1)(z - p_2)(z - p_3)\cdots}$$

Each of the poles (p_1, p_2, p_3, …) and zeros (z_1, z_2, z_3 …) is a complex number. To move from Eq. 33-4 to 33-3, multiply out the expressions and collect like terms. While this can involve a tremendous amount of algebra, it is straightforward in principle and can easily be written into a computer routine. Moving from Eq. 33-3 to 33-4 is more difficult because it requires *factoring* of the polynomials. As discussed in Chapter 32, the quadratic equation can be used for the factoring if the transfer function is second order or less (i.e., there are no powers of z higher than z^2). Algebraic methods cannot be used to factor systems greater than second order and numerical methods must be employed. Fortunately, this is seldom needed; digital filter design *starts* with the pole-zero locations (Eq. 33-4) and *ends* with the recursion coefficients (Eq. 33-3), not the other way around.

Каждый из полюсов (p_1, p_2, p_3, …) и нулей (z_1, z_2, z_3 …) - комплексное число. Чтобы двигаться от уравнения 33-4 до 33-3, умножитесь из выражений, и приведите подобные члены. В то время как это может включать в себя огромное количество алгебры, это прямое в принципе и может легко быть написано в компьютерную подпрограмму. Перемещение от уравнения от 33-3 до 33-4 более трудно, потому что это требует разложения на множители многочленов. Как обсуждено в главе 32, квадратное уравнение может использоваться для разложения на множители, если функция преобразования – второго порядка или меньше (то есть, не имеется никаких степеней z выше, чем z^2). Алгебраические методы не могут использоваться с степенями системы больше чем второго порядка и численные методы должны использоваться. К счастью, это редко необходимо; цифровой проект фильтра *начинается* с расположениями нуля полюса (уравнение 33-4) и *заканчивается* коэффициентами рекурсии (уравнение 33-3), не наоборот.

As with all complex numbers, the pole and zero locations can be represented in either polar or rectangular form. Polar notation has the advantage of being more consistent with the natural organization of the z-plane. In comparison, rectangular form is generally preferred for mathematical work, that is, it is usually easier to manipulate: $\sigma + j\omega$, as compared with: $re^{j\omega}$.

Как со всеми комплексными числами, полюс и нулевые расположения могут быть представлен или в полярной или в прямоугольной форме. Полярная система обозначений имеет преимущество, являясь более совместимой с естественной организацией z-плоскости. Для сравнения, прямоугольная форма вообще предпочитается для математической работы, то есть обычно проще управлять: $\sigma + j\omega$, по сравнению с: $re^{j\omega}$.

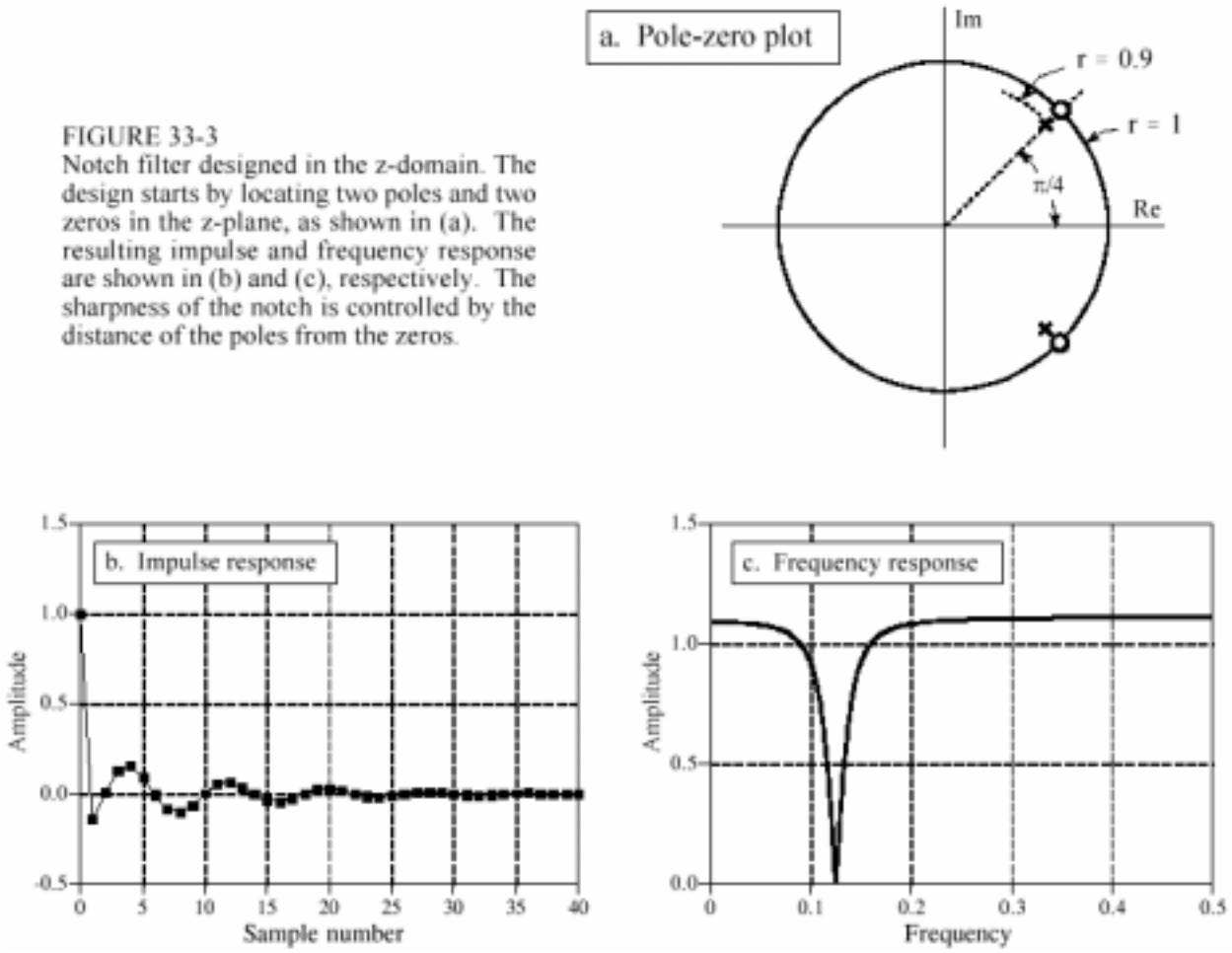


FIGURE 33-3
Notch filter designed in the z-domain. The design starts by locating two poles and two zeros in the z-plane, as shown in (a). The resulting impulse and frequency response are shown in (b) and (c), respectively. The sharpness of the notch is controlled by the distance of the poles from the zeros.

As an example of using these equations, we will design a notch filter by the following steps: (1) specify the pole-zero placement in the z-plane, (2) write down the transfer function in the form of Eq. 33-4, (3) rearrange the transfer function into the form of Eq. 33-3, and (4) identify the recursion coefficients needed to implement the filter. Fig. 33-3 shows the example we will use: a notch filter formed from two poles and two zeros located at

Как пример использования этих уравнений, мы будем проектировать фильтр-пробку следующими шагами: (1) определяют, что размещение нуля полюса в z-плоскости, (2) записывает функцию преобразования в форме уравнения 33-4, (3) перестраивают функцию преобразования в форму уравнения 33-3, и (4) идентифицируют коэффициенты рекурсии, необходимые, чтобы осуществить фильтр. Рис. 33-3 показывает пример, который мы будем использовать: фильтр-пробка, сформированная из двух полюсов и двух нулей, расположенных в

В прямоугольной форме:	В полярной форме:
$z_1 = 1.00e^{j(\pi/4)}$	$z_1 = 0.7071 + j\ 0.7071$
$z_2 = 1.00e^{j(-\pi/4)}$	$z_2 = 0.7071 - j\ 0.7071$
$p_1 = 0.90e^{j(\pi/4)}$	$p_1 = 0.6364 + j\ 0.6364$
$p_2 = 0.90e^{j(-\pi/4)}$	$p_2 = 0.6364 - j\ 0.6364$

To understand why this is a notch filter, compare this pole-zero plot with Fig. 32-6, a notch filter in the s-plane. The only difference is that we are moving along the *unit circle* to find the frequency response from the z-plane, as opposed to moving along the *vertical axis* to find the frequency response from the s-plane. From the polar form of the poles and zeros, it can be seen that the notch will occur at a natural frequency of , corresponding to 0.125 of the $\pi/4$ sampling rate.

Чтобы понимать, почему это - фильтр-пробка, сравните этот график нуля полюса с рис. 32-6, фильтр-пробка в s-плоскости. Единственная разность - то, что мы перемещаемся по кругу модуля, чтобы найти частотную характеристику от z-плоскости, в противоположность перемещению по вертикальной оси, чтобы найти частотную характеристику от s-плоскости. От полярной формы полюсов и нулей, может быть замечено, что бороздка(зубец) произойдет в собственной частоте, соответствуя 0.125 из $\pi/4$ частоты выборки.

Since the pole and zero locations are known, the transfer function can be written in the form of Eq. 33-4 by simply plugging in the values:

Так как расположения полюсов и нулей известно, функция преобразования может быть записана в форме уравнения 33-4, просто подставляя значения:

$$H(z) = \frac{[z - (0.7071 + j\,0.7071)]\,[z - (0.7071 - j\,0.7071)]}{[z - (0.6364 + j\,0.6364)]\,[z - (0.6364 - j\,0.6364)]}$$

To find the recursion coefficients that implement this filter, the transfer function must be rearranged into the form of Eq. 33-3. To start, expand the expression by multiplying out the terms:

Чтобы находить коэффициенты рекурсии, которые осуществляют этот фильтр, функция преобразования, должна быть перестроена в форму уравнения 33-3. Для начала, разворачивают выражение, перемножая его (много)члены:

$$H(z) = \frac{z^2 - 0.7071z + j0.7071z - 0.7071z + 0.7071^2 - j0.7071^2 - j0.7071z + j0.7071^2 - j^2 0.7071^2}{z^2 - 0.6364z + j0.6364z - 0.6364z + 0.6364^2 - j0.6364^2 - j0.6364z + j0.6364^2 - j^2 0.6364^2}$$

Next, we collect like terms and reduce. As long as the upper half of the z-plane is a mirror image of the lower half (which is always the case if we are dealing with a *real* impulse response), all of the terms containing a will "*j* " cancel out of the expression:

Затем, мы собираем(группируем) подобные члены и приводим. Пока верхняя половина z-плоскости - зеркальное изображение нижней половины (который всегда имеет место, если мы имеем дело с вещественной импульсной передаточной функцией), все члены, содержащие значение "*j*" отменяются(исключаются) из выражения:

$$H[z] = \frac{1.000 - 1.414z + 1.000z^2}{0.810 - 1.273z + 1.000z^2}$$

Since the transfer function is now in the form of Eq. 33-3, the recursive coefficients can be directly extracted by inspection:

Так как функция преобразования - теперь в форме уравнения 33-3, рекурсивные коэффициенты могут быть непосредственно извлечены осмотром:

a0 = 1.000
a1 = -1.414 b1 = 1.273
a2 = 1.000 b2 = -0.810

This example provides the general strategy for obtaining the recursion coefficients from a pole-zero plot. In specific cases, it is possible to derive simpler equations directly relating the pole-zero positions to the recursion coefficients. For example, a system containing two poles and two zeros, called as **biquad**, has the following relations:

Этот пример обеспечивает общую стратегию для получения коэффициентов рекурсии от графика нуля полюса. В отдельных(специфических) случаях, возможно получить более простые уравнения, непосредственно связывающие позиции нуля полюса с коэффициентами рекурсии. Например, система, содержащая два полюса и два нуля, называемые как **biquad**(**биквадратная**), имеет следующие отношения:

EQUATION 33-5
Biquad design equations. These equations give the recursion coefficients, a_0 , a_1, a_2, b_1, b_2, from the position of the poles: r_p & ω_p, and the zeros: r_0 & ω_0.

УРАВНЕНИЕ 33-5
Проектирование Biquad(биквадратных) уравнений. Эти уравнения дают коэффициенты рекурсии, a_0 , a_1, a_2, b_1, b_2, из позиции полюсов: r_p и ω_p, и нулей: r_0 и ω_0.

$$\begin{aligned} a_0 &= 1 \\ a_1 &= -2r_0\cos(\omega_0) \\ a_2 &= r_0^2 \\ b_1 &= 2r_p\cos(\omega_p) \\ b_2 &= -r_p^2 \end{aligned}$$

After the transfer function has been specified, how do we find the frequency response? There are three methods: one is mathematical and two are computational (programming). The mathematical method is based on finding the values in the z-plane that lie on the unit circle. This is done by evaluating the transfer function, $H(z)$ at $r = 1$. Specifically, we start by writing down the, transfer function in the form of either Eq. 33-3 or 33-4. We then replace each z with $e^{-j\omega}$ (that is, $re^{-j\omega}$ with $r = 1$). This provides a mathematical equation of the frequency response, $H(\omega)$. The problem is, the resulting expression is in a very inconvenient form. A significant amount of algebra is usually required to obtain something recognizable, such as the magnitude and phase. While this method provides an exact equation for the frequency response, it is difficult to automate in computer programs, such as needed in filter design packages.

После того, как функция преобразования определена, как мы находим частотную характеристику? Имеется три метода: один математический, и два - вычислительных (программирование). Математический метод основан на обнаружении значений в z-плоскости, которые лежат на круге модуля. Это сделано, оценивая функцию преобразования, $H(z)$ в $r = 1$. Определение, мы начинаем, записывая, функцию преобразования в форме уравнения 33-3 или 33-4. Мы тогда заменяем каждый z с $e^{-j\omega}$ (то есть $re^{-j\omega}$ с $r = 1$). Это обеспечивает математическое уравнение частотной характеристики, $H(\omega)$. Проблема, полученное выражение находится в очень неудобной форме. Существенное количество алгебры обычно требуется, чтобы получить кое-что распознаваемое, типа величины и фазы. В то время как этот метод обеспечивает точное уравнение для частотной характеристики, трудно автоматизировать в компьютерных программах, типа необходимого в пакетах проекта фильтра.

The second method for finding the frequency response also uses the approach of evaluating the z-plane on the unit circle. The difference is that we only calculate *samples* of the frequency response, not a mathematical solution for the entire curve. A computer program loops through, perhaps, 1000 equally spaced frequencies between $\omega = 0$ and $\omega = \pi$. Think of an ant moving between 1000 discrete points on the upper half of the z-plane's unit circle. The magnitude and phase of the frequency response are found at each of these location by evaluating the transfer function.

Второй метод для обнаружения частотной характеристики также использует подход оценки z-плоскости на круге модуля. Разница в только том, что мы вычисляем *выборки* час-

тотного ответа, не математическое решение для полной кривой. Циклы компьютерной программы через, возможные, 1000 одинаково раздельных частот между $\omega = 0$ и $\omega = \pi$. Думайте о муравье, перемещающемся между 1000 дискретных точек на верхней половине круга модуля z-плоскости. Величина и фаза частотной характеристики найдены в каждом этом расположении, оценивая функцию преобразования(передаточную функцию).

This method works well and is often used in filter design packages. Its major limitation is that it does not account for *round-off noise* affecting the system's characteristics. Even if the frequency response found by this method looks perfect, the implemented system can be completely unstable!

Этот метод работает хорошо и часто используется в пакетах проекта фильтра. Его главное ограничение - то, что это не объясняет воздействие *шума округления* на характеристики системы. Даже если частотная характеристика, найденная этими совершенными взглядами метода, осуществленная система может быть полностью непостоянна(нестабильна)!

This brings up the third method: find the frequency response from the recursion coefficients that are actually used to implement the filter. To start, we find the impulse response of the filter by passing an impulse through the system. In the second step, we take the DFT of the impulse response (using the FFT, of course) to find the system's frequency response. The only critical item to remember with this procedure is that enough samples must be taken of the impulse response so that the discarded samples are *insignificant*. While books could be written on the theoretical criteria for this, the practical rules are much simpler. Use as many samples as you *think* are necessary. After finding the frequency response, go back and repeat the procedure using twice as many samples. If the two frequency responses are adequately similar, you can be assured that the truncation of the impulse response hasn't fooled you in some way.

Это поднимает третий метод: найдите частотную характеристику от коэффициентов рекурсии, которые фактически используются, чтобы осуществить фильтр. Сначала, мы находим импульсную передаточную функцию фильтра, пропуская импульс через систему. Во втором шаге, мы берем ДПФ ответа импульса (используя БПФ, конечно) чтобы найти частотную характеристику системы. Единственный критический пункт, чтобы помнить с этой процедурой - то, что достаточно выборок должны быть приняты импульсной передаточной функции так, чтобы отвергнутые выборки были *незначащие*. В то время как книги могли быть написаны на теоретических критериях для этого, практические правила намного более просты. Использование так много выборок, как Вы думаете(понимаете), необходимо. После обнаружения частотной характеристики, возвратитесь, и повторите процедуру, используя вдвое больше выборок. Если эти две частотных характеристики соответственно подобны, Вы можете быть уверены, что усечение импульсной передаточной функции не ввело Вас в заблуждение некоторым способом.

Cascade and Parallel Stages
Каскад и Параллельные Стадии

Sophisticated recursive filters are usually designed in stages to simplify the tedious algebra of the z-domain. Figure 33-4 illustrates the two common ways that individual stages can be arranged: cascaded stages and parallel stages with added outputs. For example, a low-pass and high-pass stage can be cascaded to form a *band-pass* filter. Likewise, a parallel combination of low-pass and high-pass stages can form a *band-reject* filter. We will call the two stages being combined *system 1* and *system 2*, with their recursion coefficients being called: a_0, a_1, a_2, b_1, b_2 and A_0, A_1, A_2, B_1, B_2, respectively. Our goal is to combine these stages (in cascade or parallel)

into a single recursive filter, which we will call *system 3*, with recursion coefficients given by: a_0, a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4.

Сложные рекурсивные фильтры обычно разрабатываются постепенно, чтобы упростить утомительную алгебру z-домена. Рисунок 33-4 иллюстрирует два обычных пути, которыми индивидуальные стадии могут размещаться: каскадные стадии и параллельные стадии с добавленными выходами. Например, фильтр низкой частоты и стадия фильтра верхних частот могут быть расположены каскадом, чтобы формировать *полосовой фильтр*. Аналогично, параллельная комбинация стадий фильтра низкой частоты и фильтра верхних частот может формировать *полосовой(заграждающий*; *режекторный)* фильтр. Мы назовем две стадии объединяемой *системой 1* и *системой 2*, с их коэффициентами рекурсии называемыми: a_0, a_1, a_2, b_1, b_2 и A_0, A_1, A_2, B_1, B_2, соответственно. Наша цель состоит в том, чтобы объединить эти стадии (в каскаде или параллельно) в единый рекурсивный фильтр, которым мы назовем *система 3*, с коэффициентами рекурсии, данными: a_0, a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4.

As you recall from previous chapters, the frequency responses of systems in a cascade are combined by *multiplication*. Also, the frequency responses of systems in parallel are combined by *addition*. These same rules are followed by the z-domain transfer functions. This allows recursive systems to be combined by moving the problem into the z-domain, performing the required multiplication or addition, and then returning to the recursion coefficients of the final system.

Как Вы помните из предыдущих глав, частотные характеристики систем в каскаде объединены умножением. Также, частотные характеристики систем в параллельном объединены сложением. Они те же самые правила сопровождаемые в соответствии с функциями преобразования типа z-домена. Это позволяет рекурсивным системам быть объединенным, перемещая проблему в z-домен, выполняя требуемое умножение или сложение, и затем возвращаясь коэффициентам рекурсии конечной системе.

As an example of this method, we will work out the algebra for combining two biquad stages in a cascade. The transfer function of each stage is found by writing Eq. 33-3 using the appropriate recursion coefficients. The transfer function of the entire system, , is then found by multiplying the transfer $H[z]$ functions of the two stage:

$$H[z] = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - b_1 z^{-1} - b_2 z^{-2}} \times \frac{A_0 + A_1 z^{-1} + A_2 z^{-2}}{1 - B_1 z^{-1} - B_2 z^{-2}}$$

Multiplying out the polynomials and collecting like terms:
Умножая многочлены и приведя подобны члены:

$$H[z] = \frac{a_0A_0 + (a_0A_1 + a_1A_0)z^{-1} + (a_0A_2 + a_1A_1 + a_2A_0)z^{-2} + (a_1A_2 + a_2A_1)z^{-3} + (a_2A_2)z^{-4}}{1 - (b_1 + B_1)z^{-1} - (b_2 + B_2 - b_1B_1)z^{-2} - (-b_1B_2 - b_2B_1)z^{-3} - (-b_2B_2)z^{-4}}$$

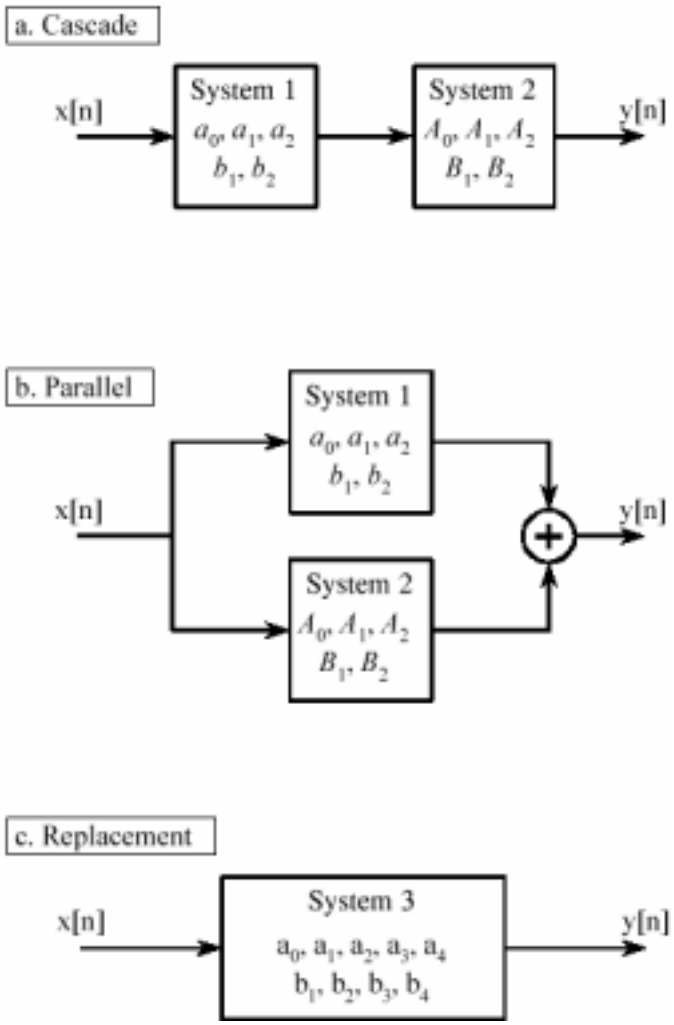


FIGURE 33-4
Combining cascade and parallel stages. The z-domain allows recursive stages in a cascade, (a), or in parallel, (b), to be combined into a single system, (c).
РИСУНОК 33-4
Объединение каскада и параллельных стадий. Z-домен позволяет рекурсивные стадии в каскаде, (a), или в параллельном, (b), быть объединенным в единую систему, (c).

Since this is in the form of Eq. 33-3, we can directly extract the recursion coefficients that implement the cascaded system:

Так как это находится в форме уравнения 33-3, мы можем непосредственно извлекать коэффициенты рекурсии, которые осуществляют каскадную систему:

$$
\begin{aligned}
a_0 &= a_0A_0 \\
a_1 &= a_0A_1 + a_1A_0 & b_1 &= b_1 + B_1 \\
a_2 &= a_0A_2 + a_1A_1 + a_2A_0 & b_2 &= b_2 + B_2 - b_1B_1 \\
a_3 &= a_1A_2 + a_2A_1 & b_3 &= -b_1B_2 - b_2B_1 \\
a_4 &= a_2A_2 & b_4 &= -b_2B_2
\end{aligned}
$$

The obvious problem with this technique is the large amount of algebra needed to multiply and rearrange the polynomial terms. Fortunately, the entire algorithm can be expressed in a short computer program, shown in Table 33-1. Although the cascade and parallel combinations require different mathematics, they use nearly the same program. In particular, only one line of code is different between the two algorithms, allowing both to be combined into a single program.

Очевидная проблема с этой методикой - большое количество алгебры, необходимой, чтобы умножать и перестраивать полиномиальные члены. К счастью, полный алгоритм может быть выражен в короткой компьютерной программе, показан в таблице 33-1. Хотя каскадные и параллельные комбинации требуют различной математики, они используют почти ту же самую программу. В частности только одна строка программы отлична между этими двумя алгоритмами, позволяя обоим быть объединенной в единственную(отдельную) программу.

```
100 'COMBINING RECURSION COEFFICIENTS OF CASCADE AND PARALLEL STAGES
110 '
120 ' 'INITIALIZE VARIABLES
130 DIM A1[8], B1[8] 'a and b coefficients for system 1, one of the stages
140 DIM A2[8], B2[8] 'a and b coefficients for system 2, one of the stages
150 DIM A3[16], B3[16] 'a and b coefficients for system 3, the combined system
160 '
170 'Indicate cascade or parallel combination
180 INPUT "Enter 0 for cascade, 1 for parallel: ", CP%
190 '
200 GOSUB XXXX 'Mythical subroutine to load: A1[ ], B1[ ], A2[ ], B2[ ]
210 '
220 FOR I% = 0 TO 8 'Convert the recursion coefficients into transfer functions
230 B2[I%] = -B2[I%]
240 B1[I%] = -B1[I%]
250 NEXT I%
260 B1[0] = 1
270 B2[0] = 1
280 '
290 FOR I% = 0 TO 16 'Multiply the polynomials by convolving
300 A3[I%] = 0
310 B3[I%] = 0
320 FOR J% = 0 TO 8
330 IF I%-J% < 0 OR I%-J% > 8 THEN GOTO 370
340 IF CP% = 0 THEN A3[I%] = A3[I%] + A1[J%] * A2[I%-J%]
350 IF CP% = 1 THEN A3[I%] = A3[I%] + A1[J%] * B2[I%-J%] + A2[J%] * B1[I%-J%]
```

```
360 B3[I%] = B3[I%] + B1[J%] * B2[I%-J%]
370 NEXT J%
380 NEXT I%
390 '
400 FOR I% = 0 TO 16 'Convert the transfer function into recursion coefficients.
410 B3[I%] = -B3[I%]
420 NEXT I%
430 B3[0] = 0
440 ' 'The recursion coefficients of the combined system now
450 END 'reside in A3[ ] & B3[ ]
```

TABLE 33-1
Combining cascade and parallel stages. This program combines the recursion coefficients of stages in cascade or parallel. The recursive coefficients for the two stages being combined enter the program in the arrays: A1[ ], B1[ ], & A2[ ], B2[ ]. The recursion coefficients that implement the entire system leave the program in the arrays: A3[ ], B3[ ].

ТАБЛИЦА 33-1
Объединение каскада и параллельных стадий. Эта программа объединяет коэффициенты рекурсии стадий в каскаде или параллельно. Рекурсивные коэффициенты для двух объединяемых стадий вводят программу в массивы: A1[ ], B1[ ], и A2[ ], B2[ ]. Коэффициенты рекурсии, которые осуществляют полную систему, оставляют программу в массивах: A3[ ], B3[ ].

This program operates by changing the recursive coefficients from each of the individual stages into transfer functions in the form of Eq. 33-3 (lines 220-270). After combining these transfer functions in the appropriate manner (lines 290-380), the information is moved back to being recursive coefficients (lines 400 to 430).

Эта программа работает, изменяя рекурсивные коэффициенты от каждой из индивидуальных стадий в функции преобразования типа в форме уравнения 33-3 (строки 220-270). После объединения этих функций преобразования соответствующим способом (строки 290-380), информация перемещена назад, будучи рекурсивными коэффициентами (строки от 400 до 430).

The heart of this program is how the transfer function polynomials are represented and combined. For example, the numerator of the first stage being combined is: $a_0 + a_1^{z-1} + a_2^{z-2} + a_3^{z-3}$, … . This polynomial is represented in the program by storing the coefficients: a_0, a_1, a_2, a_3, …, in the array: A1[0], A1[1], A1[2], A1[3]… . Likewise, the numerator for the second stage is represented by the values stored in: A2[0], A2[1], A2[2], A2[3]… , and the numerator for the combined system in: A3[0], A3[1], A3[2], A3[3] … . The idea is to represent and manipulate *polynomials* by only referring to their *coefficients*. The question is, how do we calculate A3[ ], given that A1[ ], A2[ ], and A3[ ] all represent polynomials? The answer is that when two polynomials are multiplied, their coefficients are *convolved*. In equation form: A1[ ] * A2[ ] = A3[ ]. This allows a standard convolution algorithm to find the transfer function of cascaded stages by convolving the two numerator arrays and the two denominator arrays.

Основа(сердце) этой программы - то, как многочлены функции преобразования представлены и объединены. Например, числитель первой объединяемой стадии: $a_0 + a_1^{z-1} + a_2^{z-2} + a_3^{z-3}$, … . Этот полиномиальный представлен в программе, сохраняя коэффициенты: a0, a1, a2, a3, …, в массиве: A1[0], A1[1], A1[2], A1[3] … . Аналогично, числитель для второй стадии представлен значениями, сохраненными в: A2[0], A2[1], A2[2], A2[3] …, и числитель для объединенной системы в: A3[0], A3[1], A3[2], A3[3] … . Идея должна представлять и управлять *многочленом* только что касается их коэффициентов. Вопрос, как мы вычисляем A3[ ], учитывая что A1[ ], A2[ ], и A3[ ] все представляют многочлены? Ответ - то, что, когда два многочлена умножены, их коэффициенты свернуты(скручены). В форме уравнения: A1[ ] * A2[ ] = A3[ ]. Это позволяет стандартному алгоритму свертки находить

функцию преобразования типа каскадных стадий, скручивая(свертывая) два массива числителей и два массива знаменателей.

The procedure for combining parallel stages is slightly more complicated. In algebra, fractions are added according to:
Процедура для объединения параллельных стадий слегка больше усложнена. По алгебре, дроби добавлены согласно:

$$\frac{w}{x} + \frac{y}{z} = \frac{w \cdot z + y \cdot z}{x \cdot z}$$

Since each of the transfer functions is a fraction (one polynomial divided by another polynomial), we combine stages in parallel by multiplying the denominators, and adding the cross products in the numerators. This means that the denominator is calculated in the same way as for cascaded stages, but the numerator calculation is more elaborate. In line 340, the numerators of *cascaded* stages are convolved to find the numerator of the combined transfer function. In line 350, the numerator of the *parallel* stage combination is calculated as the sum of the two numerators convolved with the two denominators. Line 360 handles the denominator calculation for both cases.

Так как каждая из функций преобразования - дробь (один многочлен разделен на другой многочлен), мы объединяем стадии параллельно, умножая знаменатели, и складывая векторные произведения в числителях. Это означает, что знаменатель рассчитан таким же образом, что и каскадные стадии, но вычисление числителя более сложно. В строке 340, числители *каскадных* стадий свернуты(скручены), чтобы найти числитель объединенной функции преобразования. В строке 350, числитель *параллельной* комбинации стадий рассчитан как сумма из двух числителей, свернутых(скрученных) с этими двумя знаменателями. Строка 360 руководит вычислением знаменателя для обоих случаев.

Spectral Inversion
Спектральная Инверсия

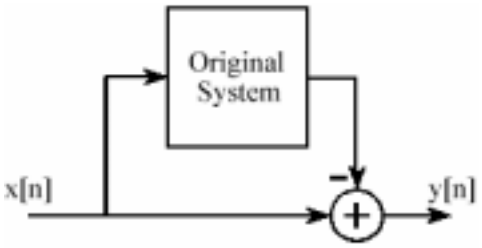
Chapter 14 describes an FIR filter technique called *spectral inversion*. This is a way of changing the filter kernel such that the frequency response is flipped top-for-bottom. All the passbands are changed into stopbands, and vice versa. For example, a low-pass filter is changed into high-pass, a band-pass filter into band-reject, etc. A similar procedure can be done with recursive filters, although it is far less successful.

Глава 14 описывает методику КИХ-фильтра, называемую *спектральной инверсией*. Это - путь изменения ядра фильтра так, что частотная характеристика зеркально отражена " вершина-основание ". Все полосы пропускания изменены в полосы задерживания, и наоборот. Например, фильтр нижних частот изменен в фильтр верхних частот, полосовой фильтр в полосовой(заграждающий), и т.д. Подобная процедура может быть сделана с рекурсивными фильтрами, хотя это гораздо менее успешно.

As illustrated in Fig. 33-5, spectral inversion is accomplished by subtracting the output of the system from the original signal. This procedure can be viewed as combining two stages in parallel, where one of the stages happens to be the *identity system* (the output is identical to the input). Using this approach, it can be shown that the "b" coefficients are left unchanged, and the modified "a" coefficients are given by:

Как иллюстрировано в рис. 33-5, спектральная инверсия выполнена, вычитая выход системы от первоначального сигнала. Эта процедура может быть просмотрена как объединение двух стадий параллельно, где одна из стадий, случается, система тождественна (выход идентична вводу). Используя этот подход, можно показать, что коэффициенты "b" оставлены неизменяемыми, и изменяемые коэффициенты "a" даются:

FIGURE 33-5
Spectral inversion. This procedure is the same as subtracting the output of the system from the original signal.
РИСУНОК 33-5
Спектральная инверсия. Эта процедура - то же самое как вычитание выхода системы от первоначального сигнала.



EQUATION 33-6
Spectral inversion. The frequency response of a recursive filter can be flipped top-for-bottom by modifying the "a" coefficients according to these equations. The original coefficients are shown in italics, and the modified coefficients in roman. The "b" coefficients are not changed. This method usually provides poor results.
УРАВНЕНИЕ 33-6
Спектральная инверсия. Частотная характеристика рекурсивного фильтра может быть зеркально отражена " вершина-основание ", изменяя коэффициенты "a" согласно этим уравнениям. Первоначальные коэффициенты показываются курсивом, и изменяемые коэффициенты в католике(римском?). Коэффициенты "b" не изменены. Этот метод обычно дает плохие результаты.

$$\begin{aligned} a_0 &= 1 - a_0 \\ a_1 &= -a_1 - b_1 \\ a_2 &= -a_2 - b_2 \\ a_3 &= -a_3 - b_3 \\ &\vdots \end{aligned}$$

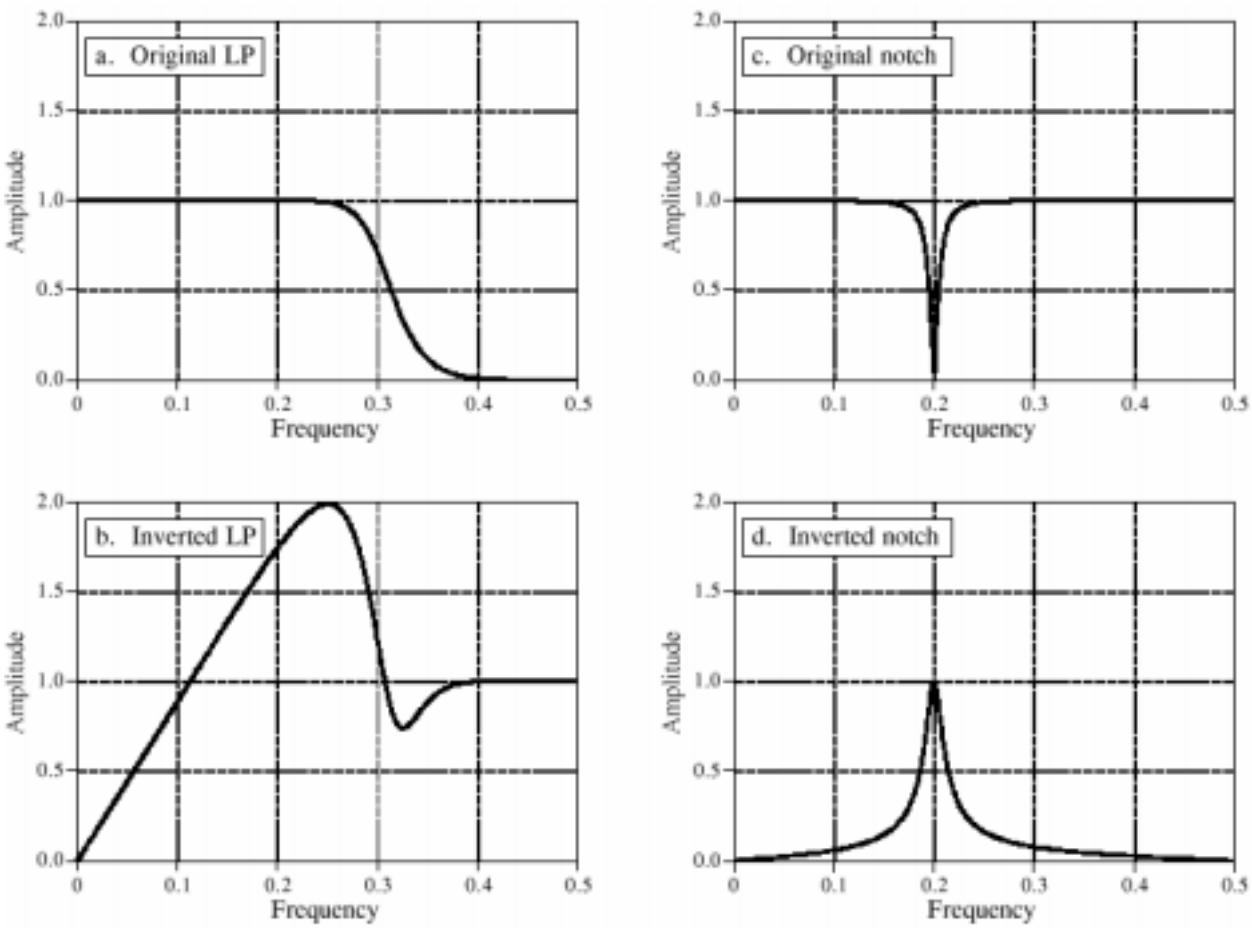


FIGURE 33-6
Examples of spectral inversion. Figure (a) shows the frequency response of a 6 pole low-pass Butterworth filter. Figure (b) shows the corresponding high-pass filter obtained by spectral inversion; its a mess! A more successful case is shown in (c) and (d) where a notch filter is transformed in to a band-pass frequency response.
РИСУНОК 33-6
Примеры спектральной инверсии. Рисунок (a) показывает частотную характеристику из 6 полюсов фильтра Буттерворта низкой частоты. Рисунок (b) показывает соответствующий фильтр верхних частот, полученный спектральной инверсией; это беспорядок! Более успешный случай показывается в (c) и (d), где фильтр-пробка преобразован в к полосовой частотной характеристике.

Figure 33-6 shows spectral inversion for two common frequency responses: a low-pass filter, (a), and a notch filter, (c). This results in a high-pass filter, (b), and a band-pass filter, (d), respectively. How do the resulting frequency responses look? The high-pass filter is absolutely terrible! While the band-pass is better, the peak is not as sharp as the notch filter from which it was derived. These mediocre results are especially disappointing in comparison to the excellent performance seen in Chapter 14. Why the difference? The answer lies in something that is often forgotten in filter design: the *phase response*.

Рисунок 33-6 показывает спектральную инверсию для двух обычных частотных характеристик: фильтра нижних частот, (a), и фильтра-пробки, (c). Это приводит к фильтру верхних частот, (b), и полосовому фильтру, (d), соответственно. Как заканчивающиеся частотные характеристики смотрят? Фильтр верхних частот абсолютно ужасен! В то время как полосовой лучше, пик - не столь острый как фильтр-пробка, из которого это было получено. Эти посредственные результаты особенно неутешительны по сравнению с превосходной эффективностью, отмеченной в главе 14. Почему разность? Ответ находится в коечем, что часто забывается в проекте фильтра: *фазочастотная характеристика*.

To illustrate how phase is the culprit, consider a system called the **Hilbert transformer**. The Hilbert transformer is not a specific device, but any system that has the frequency response: Magnitude = 1 and phase = 90 degrees, for all frequencies. This means that any sinusoid passing through a Hilbert transformer will be unaffected in amplitude, but changed in phase by one-quarter of a cycle. Hilbert transformers can be analog or discrete (that is, hardware or software), and are commonly used in communications for various modulation and demodulation techniques.

Чтобы иллюстрировать, как фаза - преступник, рассматривает систему называемую Гильбертовым трансформатором. Гильбертов трансформатор - не удельное устройство, но любая система, которая имеет частотную характеристику: Величина = 1 и фаза = 90 градусов, для всех частот. Это означает, что любая синусоида, проходящая через Гильбертов трансформатор будет незатронута в амплитуде, но изменена в фазе с одной четвертью из периода. Гильбертовы трансформаторы могут быть аналоговые или дискретные (то есть аппаратные средства или программное обеспечение), и обычно используются в связи для различных методов модуляции и демодуляции.

Now, suppose we spectrally invert the Hilbert transformer by subtracting its output from the original signal. Looking only at the *magnitude* of the frequency responses, we would conclude that the entire system would have an output of *zero*. That is, the magnitude of the Hilbert transformer's output is identical to the magnitude of the original signal, and the two will cancel. This, of course, is completely incorrect. Two sinusoids will exactly cancel only if they have the same magnitude *and* phase. In reality, the frequency response of this composite system has a magnitude of $\sqrt{2}$, and a phase shift of -45 degrees. 2 Rather than being zero (our naive guess), the output is *larger* in amplitude than the input!

Теперь, предположите, что мы спектрально инвертируем Гильбертов трансформатор, вычитая его выход от первоначального сигнала. Смотря только на *величину* частотных характеристик, мы заключили бы, что полная система будет иметь выход *нуля*. То есть величина выхода Гильбертова трансформатора идентична величине первоначального сигнала, и эти два(вход и выход?) отменяют(компенсируются). Это, конечно, является полностью неправильным. Две синусоиды точно отменят(компенсируют друг друга?), если только они имеют одну и ту же величину и фазу. В действительности, частотная характеристика этой составной системы имеет величину $\sqrt{2}$, и сдвиг фаз -45 градусов. Скорее чем являющийся нулем (наше наивное предположение), выход *больший* в амплитуде, чем ввод!

Spectral inversion works well in Chapter 14 because of the specific kind of filter used: *zero phase*. That is, the filter kernels have a left-right symmetry. When there is no phase shift introduced by a system, the subtraction of the output from the input is dictated solely by the magnitudes. Since recursive filters are plagued with phase shift, spectral inversion generally produces unsatisfactory filters.

Спектральная инверсия работает хорошо в главе 14 из-за специфического вида используемого фильтра: *нулевая фаза*. То есть ядра фильтра имеют лево - правую симметрию. Когда не имеется никакого сдвига фаз, представленного системой, вычитание выхода от ввода диктуется исключительно величинами. Так как рекурсивные фильтры мучаются(наказание; бич) со сдвигом фаз, спектральная инверсия вообще производит неудовлетворительные фильтры.

Gain Changes

Изменение усиления

Suppose we have a recursive filter and need to modify the recursion coefficients such that the output signal is changed in amplitude. This might be needed, for example, to insure that a filter has unity gain in the passband. The method to achieve this is very simple: multiply the "a" coefficients by whatever factor we want the gain to change by, and leave the "b" coefficients alone.

Предположим, что мы имеем рекурсивный фильтр и должны изменить коэффициенты рекурсии такой, что сигнал выхода изменен в амплитуде. Это могло бы быть необходимо, например, обеспечивать, чтобы фильтр имел единичное усиление в полосе пропускания. Метод достичь этого очень прост: умножьте коэффициенты "a" на любой коэффициент, на который мы хотим, чтобы усиление изменилось, и оставьте коэффициенты "b" неизменными.

Before adjusting the gain, we would probably like to know its current value. Since the gain must be specified at a frequency in the *passband*, the procedure depends on the type of filter being used. Low-pass filters have their gain measured at a frequency of *zero*, while high-pass filters use a frequency of 0.5, the maximum frequency allowable. It is quite simple to derive expressions for the gain at both these special frequencies. Here's how it is done.

Перед корректировкой усиления, мы вероятно хотели бы знать его текущее значение. Так как усиление должно быть определено в частоте *полосы пропускания*, процедура зависит от типа используемого фильтра. Фильтры нижних частот измеряют их усиление в частоте *нуля*, в то время как фильтры верхних частот используют частоту 0.5, максимально допустимой частоты. Весьма просто получить выражения для усиления обоих этих специальных частот. Имеется, как это сделано.

First, we will derive an equation for the gain at zero frequency. The idea is to force each of the input samples to have a value of *one*, resulting in each of the output samples having a value of *G*, the gain of the system we are trying to find. We will start by writing the recursion equation, the mathematical relationship between the input and output signals:

Во первых, мы получим уравнение для усиления в нулевой частоте. Идея состоит в том, чтобы вынудить каждую из входных выборок иметь значение *единицы*, приводя к каждой из выборок выхода, имеющих значение G, усиление системы, которую мы пробуем находить. Мы начнем, запись уравнения рекурсии, математическими отношениями между сигналами ввода и вывода:

$$y[n] = a_0x[n] + a_1x[n-1] + a_2x[n-2] + \cdots + b_1y[n-1] + b_2y[n-2] + b_3y[n-3] + \cdots$$

Next, we plug in *one* for each input sample, and *G* for each output sample. In other words, we force the system to operate at zero frequency. The equation becomes:
Затем, мы подключаем *единицу* для каждой входной выборки, и G для каждой выборки выхода. Другими словами, мы вынуждаем систему работать в нулевой частоте. Уравнение становится:

$$G = a_0 + a_1 + a_2 + a_3 + \cdots + b_1G + b_2G + b_3G + b_4G \cdots$$

Solving for *G* provides the gain of the system at a frequency of 0.5, using its recursion coefficients:
Решение для G обеспечивает усиление системы в частоте 0.5, используя ее коэффициенты рекурсии:

EQUATION 33-7
DC gain of recursive filters. This relation provides the DC gain from the recursion coefficients.
УРАВНЕНИЕ 33-7
Усиление постоянного тока рекурсивных фильтров. Это отношение обеспечивает усиление постоянного тока от коэффициентов рекурсии.

$$G = \frac{a_0 + a_1 + a_2 + a_3 \cdots}{1 - (b_1 + b_2 + b_3 \cdots)}$$

To make a filter have a gain of *one* at DC, calculate the existing gain by using this relation, and then divide all the "a" coefficients by *G*.
Делая фильтр, имеющий усиление по постоянному току *единица*, вычисляют существующее усиление, используя это отношение, и затем делят все коэффициенты "a" *G*.

The gain at a frequency of 0.5 is found in a similar way: we force the input and output signals to operate at this frequency, and see how the system responds. At a frequency of 0.5, the samples in the input signal alternate between -1 and 1. That is, successive samples are: 1, -1, 1, -1, 1, -1, 1, etc. The corresponding output signal also alternates in sign, with an amplitude equal to the gain of the system: *G*, -*G*, *G*, -*G*, *G*, -*G*, etc. Plugging these signals into the recursion equation:

Усиление в частоте 0.5 найдена подобным способом: мы вынуждаем сигналы ввода и вывода работать в этой частоте, и см., как система отвечает. В частоте 0.5, выборки во входном сигнале чередуются между -1 и 1. То есть последовательные выборки: 1, -1, 1, -1, 1, -1, 1, и т.д. Соответствующий сигнал выхода также чередуется в знаке, с амплитудой равной усилению системы: *G*, -*G*, *G*, -*G*, *G*, -*G*, и т.д. Подключение этих сигналов в уравнение рекурсии:

$$G = a_0 - a_1 + a_2 - a_3 + \cdots - b_1 G + b_2 G - b_3 G + b_4 G \cdots$$

Solving for *G* provides the gain of the system at a frequency of 0.5, using its recursion coefficients:
Решение для *G* обеспечивает усиление системы в частоте 0.5, используя ее коэффициентов рекурсии:

EQUATION 33-8
Gain at maximum frequency. This relation gives the recursive filter's gain at a frequency of 0.5, based on the system's recursion coefficients.
УРАВНЕНИЕ 33-8
Усиление в максимальной частоте. Это отношение дает усиление рекурсивного фильтра в частоте 0.5, основанный на коэффициентах рекурсии системы.

$$G = \frac{a_0 - a_1 + a_2 - a_3 + a_4 \cdots}{1 - (-b_1 + b_2 - b_3 + b_4 \cdots)}$$

Just as before, a filter can be normalized for unity gain by dividing all of the "a" coefficients by this calculated value of *G*. Calculation of Eq. 33-8 in a computer program requires a method for generating negative signs for the odd coefficients, and positive signs for the even coefficients. The most common method is to multiply each coefficient by $(-1)^k$, where *k* is the index of the coefficient being worked on. That is, as *k* runs through the values: 0, 1, 2, 3, 4, 5, 6 etc., the expression, $(-1)^k$, takes on the values: 1, -1, 1, -1, 1, -1, 1 etc.

Так же, как прежде, фильтр может быть нормализован для единичного усиления, разделяя все коэффициенты "a" этим расчетным значением *G*. Вычисление уравнения 33-8 в компьютерной программе требует метода для производства отрицательных знаков для нечетных коэффициентов, и положительных знаков для четных коэффициентов. Наиболее обычный метод состоит в том, чтобы умножить каждый коэффициент на $(-1)^k$, где *k* - ин-

декс коэффициента, над которым разрабатывают. То есть как *k* выполняется через значения: 0, 1, 2, 3, 4, 5, 6 и т.д., выражение, (-1)*k*, принимает значения: 1, -1, 1, -1, 1, -1, 1 и т.д.

Chebyshev-Butterworth Filter Design
Проект Фильтра Chebyshev-Butterworth

A common method of designing recursive digital filters is shown by the Chebyshev-Butterworth program presented in Chapter 20. It starts with a pole-zero diagram of an *analog* filter in the s-plane, and converts it into the desired *digital* filter through several mathematical *transforms*. To reduce the complexity of the algebra, the filter is designed as a cascade of several stages, with each stage implementing one pair of poles. The recursive coefficients for each stage are then combined into the recursive coefficients for the entire filter. This is a very sophisticated and complicated algorithm; a fitting way to end this book. Here's how it works.

Обычный метод проектирования рекурсивных цифровых фильтров показывается программой Chebyshev-Butterworth, представленной в главе 20. Это начинается с диаграммой нуля полюса *аналогового* фильтра в s-плоскости, и преобразовывает это в желательный *цифровой* фильтр через несколько математических *трансформант*. Чтобы приводить сложность алгебры, фильтр разработан как каскад нескольких стадий, с каждой стадией, осуществляющей одну пару полюсов. Рекурсивные коэффициенты для каждой стадии тогда объединены в рекурсивные коэффициенты для полного фильтра. Это - очень изощренный и сложный алгоритм; приспособление способ заканчивать эту книгу. Имеется, как это работает.

Loop Control
Управление цикла

Figure 33-7 shows the program and flowchart for the method, duplicated from Chapter 20. After initialization and parameter entry, the main portion of the program is a loop that runs through each pole-pair in the filter. This loop is controlled by block 11 in the flowchart, and the FOR-NEXT loop in lines 320 & 460 of the program. For example, the loop will be executed three times for a 6 pole filter, with the loop index, P%, taking on the values 1,2,3. That is, a 6 pole filter is implemented in three stages, with two poles per stage.

Рисунок 33-7 показвает программу и блок-схему для метода, дублированного от главы 20. После инициализации и входа параметра, основная часть программы - цикл, который выполняется через каждую пару полюса в фильтре. Этот цикл управляется блоком в блок-схеме, и для следующего цикла в строках 320 и 460 из программы. Например, цикл будет выполнен три раза для 6 полюсов фильтра, с индексом цикла, P%, принимая значения 1,2,3. То есть 6 полюсов фильтра осуществлены в трех стадиях, с двумя полюсами в стадию.

Combining Coefficients
Объединение Коэффициентов

During each loop, subroutine 1000 (listed in Fig. 33-8) calculates the recursive coefficients *for that stage*. These are returned from the subroutine in the five variables: A0, A1, A2, B1, B2. In step 10 of the flowchart (lines 360-440), these coefficients are combined with the coefficients of all the previous stages, held in the arrays: A[ ] and B[ ]. At the end of the first loop, A[ ] and B[ ] hold the coefficients for stage one. At the end of the second loop, A[ ] and B[ ] hold the coefficients of the cascade of stage one and stage two. When all the loops have been completed, A[ ] and B[ ] hold the coefficients needed to implement the entire filter.

В течение каждого цикла, подпрограмма 1000 (перечисленная в рис. 33-8) вычисляет рекурсивные коэффициенты *для той стадии*. Они возвращены от подпрограммы в этих пяти переменных: A0, A1, A2, B1, B2. Шаг 10 из блок-схемы (строки 360-440), эти коэффициенты, объединены с коэффициентами всех предыдущих стадий, проведенных(поддержанных) в массивах: A[ ] и B[ ]. В конце первого цикла, A[ ] и B[ ] проводят(держат) коэффициенты для одной стадии. В конце второго цикла, A[ ] и B[ ] проводят(держат) коэффициенты одной стадии каскада, и организуют две. Когда все циклы были закончены, A[ ] и B[ ] проводят коэффициенты необходимые для осуществления полного фильтра.

The coefficients are combined as previously outlined in Table 33-1, with a few modifications to make the code more compact. First, the index of the arrays, A[ ] and B[ ], is shifted by *two* during the loop. For example, a_0 is held in A[2], a_1 & b_1 are held in A[3] & B[3], etc. This is done to prevent the program from trying to access values outside the defined arrays. This shift is removed in block 12 (lines 480-520), such that the final recursion coefficients reside in A[ ] and B[ ] without an index offset.

Коэффициенты объединены как предварительно выделено в таблице 33-1, с несколькими модификациями, чтобы делать код более компактным. Во первых, индекс массивов, A[ ] и B[ ], сдвинут двумя в течение цикла. Например, a_0 проведен(поддержан) в A[2], a_1 и b_1 проведены(поддержаны) в A[3] и B[3], и т.д. Это сделано, чтобы предотвратить программу от попытки обратиться к значениям вне определенных массивов. Этот сдвиг удален в блоке (строки 480-520), так, что конечные(заключительные) коэффициенты рекурсии постоянно находятся в A[ ] и B[ ] без смещения индекса.

Second, A[ ] and B[ ] must be initialized with coefficients corresponding to the *identity* system, not all zeros. This is done in lines 180 to 240. During the first loop, the coefficients for the first stage are combined with the information initially present in these arrays. If all zeros were initially present, the arrays would always remain zero. Third, two temporary arrays are used, TA[ ] and TB[ ]. These hold the old values of A[ ] and B[ ] during the convolution, freeing A[ ] and B[ ] to hold the new values.

Во вторых, A[ ] и B[ ] должны быть инициализированы с коэффициентами, соответствующими системе тождества, не все нули. Это сделано в строках от 180 до 240. В течение первого цикла, коэффициенты для первой стадии объединены с информацией, первоначально представленной в этих массивах. Если бы все нули были первоначально инициализированы, массивы всегда остались бы нулевыми. Третье, два временных массива используются, TA[ ] и TB[ ]. Они считают старые значения A[ ] и B[ ] в течение свертки, освобождая A[ ] и B[ ] проводить(держать) новые значения.

To finish the program, block 13 (lines 540-670) adjusts the filter to have a unity gain in the passband. This operates as previously described: calculate the existing gain with Eq. 33-7 or 33-8, and divide all the "a" coefficients to normalize. The intermediate variables, SA and SB, are the sums of the "a" and "b" coefficients, respectively.

Чтобы закончить программу, блок (строки 540-670) корректирует фильтр, чтобы иметь единичное усиление в полосе пропускания. Это работает как предварительно описано: вычислите существующее усиление с уравнением 33-7 или 33-8, и делите все коэффициенты "a", чтобы нормализовать. Промежуточные переменные, SA и SB, являются суммами коэффициентов "a" и "b", соответственно.

Calculate Pole Locations in the s-Plane

Вычислите Расположения Полюса в s-плоскости

Regardless of the type of filter being designed, this program begins with a Butterworth low-pass filter in the *s-plane*, with a cutoff frequency of $\omega = 1$. As described in the last chapter, Butterworth filters have poles that are equally spaced around a circle in the s-plane. Since the filter is low-pass, no zeros are used. The radius of the circle is *one*, corresponding to the cutoff frequency of $\omega = 1$. Block 3 of the flowchart (lines 1080 & 1090) calculate the location of each pole-pair in rectangular coordinates. The program variables, RP and IP, are the real and imaginary parts of the pole location, respectively. These program variables correspond to σ and ω, where the pole-pair is located at $\sigma \pm j\omega$. This pole location is calculated from the number of poles in the filter and the stage being worked on, the program variables: NP and P%, respectively.

Независимо от типа разрабатываемого фильтра, эта программа начинается с фильтра нижних частот Буттерворта в *s-плоскости*, с частотой отсечки $\omega = 1$. Как описано в прошлой главе, фильтры Буттерворта имеют полюса, которые одинаково располагаются вокруг круга в s-плоскости. Так как фильтр низкой частоты, никакие нули не используются. Радиус круга *еденица*, соответствует частоте отсечки $\omega = 1$. Блок блок-схема (строки 1080 и 1090) вычисляет расположение каждой пары полюса в прямоугольных координатах. Переменные программы, RP и IP, являются вещественными и мнимыми частями расположения полюса, соответственно. Эти переменные программы соответствуют σ и ω, где пара полюса расположена в $\sigma \pm j\omega$. Это расположение полюса рассчитано от числа полюсов в фильтре и стадии, над которой разрабатывают, переменные программы: NP и P%, соответственно.

Warp from Circle to Ellipse

Деформация от Круга до Эллипса

To implement a Chebyshev filter, this *circular* pattern of poles must be transformed into an *elliptical* pattern. The relative flatness of the ellipse determines how much ripple will be present in the passband of the filter. If the pole location on the circle is given by: σ and ω, the corresponding location on the ellipse, σ' and ω', is given by:

Чтобы осуществлять Чебышевский фильтр, этот *круговой*(*кольцевой*) образец полюсов должен быть преобразован в *эллиптический* образец. Относительная пологость эллипса определяет, сколько ряби будет присутствовать в полосе пропускания фильтра. Если расположение полюса на круге дается σ and ω, соответствующее расположение на эллипсе, σ' и ω', дается:

EQUATION 33-9

Circular to elliptical transform. These equations change the pole location on a circle to a corresponding location on an ellipse. The variables, *NP* and *PR*, are the number of poles in the filter, and the percent ripple in the passband, respectively. The location on the circle is given by σ and ω, and the location on the ellipse by σ' and ω'. The variables **,** v, and k, are used only to make the equations shorter.

$$\sigma' = \sigma \sinh(v)/k$$

$$\omega' = \omega \cosh(v)/k$$

where:

$$v = \frac{\sinh^{-1}(1/\epsilon)}{NP}$$

$$k = \cosh\left(\frac{1}{NP}\cosh^{-1}\frac{1}{\epsilon}\right)$$

$$\epsilon = \left[\left(\frac{100}{100 - PR}\right)^2 - 1\right]^{1/2}$$

УРАВНЕНИЕ 33-9

Преобразование круга к эллипсу. Эти уравнения изменяют расположение полюса на круге к соответствующему расположению на эллипсе. Переменные, NP и PR, являются числом полюсов в фильтре, и процентом ряби в полосе пропускания, соответственно. Расположение на круге дается σ и ω, и расположение на эллипсе σ' и ω'. Переменные, v, и k, используются только, чтобы делать уравнения короче.

These equations use hyperbolic sine and cosine functions to define the ellipse, just as ordinary sine and cosine functions operate on a circle. The flatness of the ellipse is controlled by the variable: *PR*, which is numerically equal to the percentage of ripple in the filter's passband. The variables: ϵ, v, and k are used to reduce the complexity of the equations, and are represented in the program by: *ES*, *VX* and *KX*, respectively. In addition to converting from a circle to an ellipse, these equations correct the pole locations to keep a unity cutoff frequency. Since many programming languages do not support hyperbolic functions, the following identities are used:

Эти уравнения используют гиперболические функции синуса и косинуса, чтобы определить эллипс, так же, как обычные функции синуса и косинуса работают на круге. Пологость эллипса управляется переменной: *PR*, которая является в цифровой форме равным проценту от ряби в полосе пропускания фильтра. Переменные: є,*v*, и *k* используются, чтобы привести сложность уравнений, и представлены в программе: *ES*, *VX* и *KX*, соответственно. В дополнение к преобразованию от круга до эллипса, эти уравнения исправляют расположения полюса, чтобы сохранить частоту отсечки единицы. Так как много языков программирования не поддерживают гиперболические функции, следующие тождества используются:

$$\sinh(x) = \frac{e^{x} - e^{-x}}{2}$$

$$\cosh(x) = \frac{e^{x} + e^{-x}}{2}$$

$$\sinh^{-1}(x) = \log_e \left[x + (x^2 + 1)^{1/2}\right]$$

$$\cosh^{-1}(x) = \log_e \left[x + (x^2 - 1)^{1/2}\right]$$

These equations produce illegal operations for $PR \geq 30$ and $PR = 0$. To use this program to calculate Butterworth filters (i.e., zero ripple, $PR = 0$), the program lines that implement these equations must be bypassed (line 1120).

Эти уравнения производят незаконные операции для $PR \geq 30$ и $PR = 0$. Чтобы использовать эту программу, чтобы вычислить фильтры Буттерворта (то есть обнулить рябь, $PR = 0$), строки программы, которые осуществляют эти уравнения, должны быть обойдены (строка 1120).

Continuous to Discrete Conversion
Преобразование Непрерывный к Дискретному

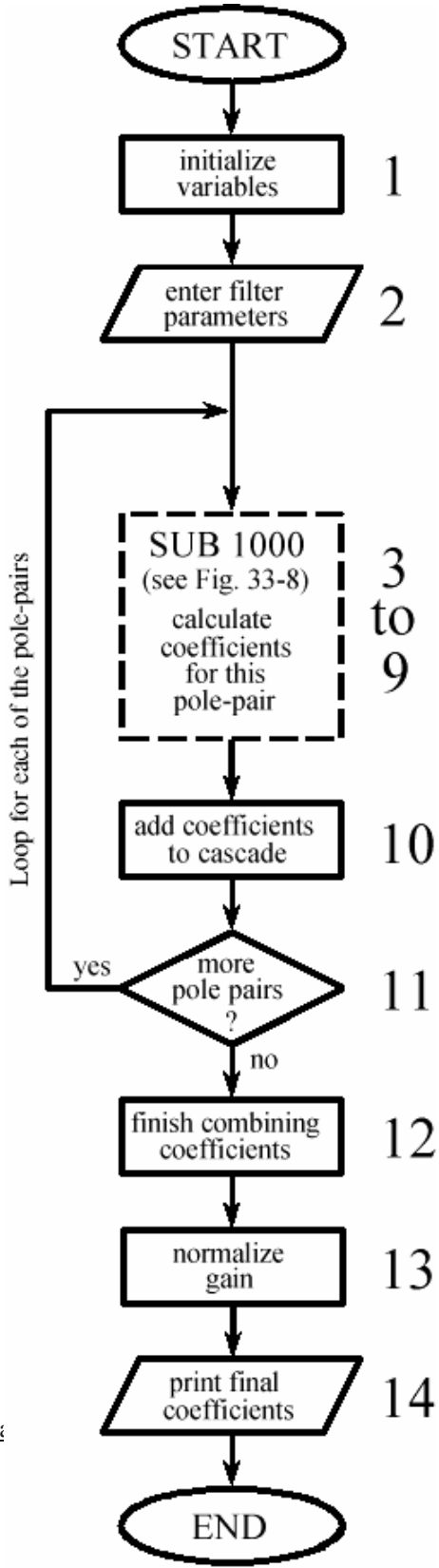
The most common method of converting a pole-zero pattern from the s-domain into the z-domain is the **bilinear transform**. This is a mathematical technique of *conformal mapping*, where one complex plane is algebraically distorted or warped into another complex plane. The bilinear transform changes $H(s)$, into $H(z)$, by the substitution:

Наиболее обычный метод преобразования образца нуля полюса от s-домена в z-домен - **билинейная трансформанта**. Это - математическая методика *конформного отображения*, где одна комплексная плоскость алгебраически искажена или деформирована в дру-

гую комплексную плоскость. Билинейная трансформанта изменяет(заменяет) $H(s)$, в $H(z)$, заменой(замещением; подстановкой):

EQUATION 33-10
The Bilinear transform. This substitution maps every point in the s-plane into a corresponding piont in the z-plane.
УРАВНЕНИЕ 33-10
Билинейная трансформанта. Эта замена отображает каждую точку в s-плоскости, передавая точку в z-плоскости.

$$s \rightarrow \frac{2(1 - z^{-1})}{T(1 + z^{-1})}$$



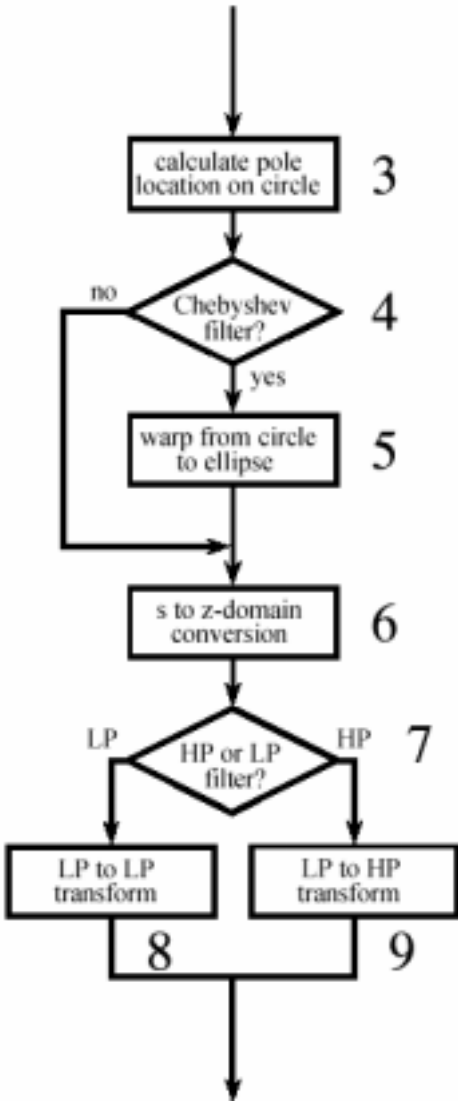

```
100 'CHEBYSHEV FILTER- COEFFICIENT CALCULATION
110 '
120 'INITIALIZE VARIABLES
130 DIM A[22] 'holds the "a" coefficients
140 DIM B[22] 'holds the "b" coefficients
150 DIM TA[22] 'internal use for combining stages
160 DIM TB[22] 'internal use for combining stages
170 '
180 FOR I% = 0 TO 22
190 A[I%] = 0
200 B[I%] = 0
210 NEXT I%
220 '
230 A[2] = 1
240 B[2] = 1
250 PI = 3.14159265
260 'ENTER THE FILTER PARAMETERS
270 INPUT "Enter cutoff frequency (0 to .5): ", FC
280 INPUT "Enter 0 for LP, 1 for HP filter: ", LH
290 INPUT "Enter percent ripple (0 to 29): ", PR
300 INPUT "Enter number of poles (2,4,...20): ", NP
310 '
```

```
320 FOR P% = 1 TO NP/2 'LOOP FOR EACH POLE-ZERO PAIR
330 '
340 GOSUB 1000 'The subroutine in Fig. 33-8
350 '
360 FOR I% = 0 TO 22 'Add coefficients to the cascade
370 TA[I%] = A[I%]
380 TB[I%] = B[I%]
390 NEXT I%
400 '
410 FOR I% = 2 TO 22
420 A[I%] = A0*TA[I%] + A1*TA[I%-1] + A2*TA[I%-2]
430 B[I%] = TB[I%] - B1*TB[I%-1] - B2*TB[I%-2]
440 NEXT I%
450 '
460 NEXT P%
470 '
480 B[2] = 0 'Finish combining coefficients
490 FOR I% = 0 TO 20
500 A[I%] = A[I%+2]
510 B[I%] = -B[I%+2]
520 NEXT I%
530 '
540 SA = 0 'NORMALIZE THE GAIN
550 SB = 0
560 FOR I% = 0 TO 20
570 IF LH = 0 THEN SA = SA + A[I%]
580 IF LH = 0 THEN SB = SB + B[I%]
590 IF LH = 1 THEN SA = SA + A[I%] * (-1)^I%
600 IF LH = 1 THEN SB = SB + B[I%] * (-1)^I%
610 NEXT I%
620 '
630 GAIN = SA / (1 - SB)
640 '
650 FOR I% = 0 TO 20
660 A[I%] = A[I%] / GAIN
670 NEXT I%
680 ' 'The final recursion coefficients are
690 END 'in A[ ] and B[ ]
```

РИСУНОК 33-7

Проект фильтра Чебышева-Буттерворта. Эта программа была предварительно представлена таблицей 20-4 и таблицей 20-5 в главе 20. Рисунок 33-8 показывает блок-схему программы для подпрограммы 1000, вызываемой от строки 340 из этой основной программы.

```
1000 'THIS SUBROUTINE IS CALLED FROM FIG. 33-7, LINE 340
1010 '
1020 'Variables entering subroutine: PI, FC, LH, PR, HP, P%
1030 'Variables exiting subroutine: A0, A1, A2, B1, B2
1040 'Variables used internally: RP, IP, ES, VX, KX, T, W, M, D, K,
1050 ' X0, X1, X2, Y1, Y2
1060 '
1070 ' 'Calculate pole location on unit circle
1080 RP = -COS(PI/(NP*2) + (P%-1) * PI/NP)
1090 IP = SIN(PI/(NP*2) + (P%-1) * PI/NP)
1100 '
1110 ' 'Warp from a circle to an ellipse
1120 IF PR = 0 THEN GOTO 1210
1130 ES = SQR( (100 / (100-PR))^2 -1 )
1140 VX = (1/NP) * LOG( (1/ES) + SQR( (1/ES^2) + 1) )
1150 KX = (1/NP) * LOG( (1/ES) + SQR( (1/ES^2) - 1) )
1160 KX = (EXP(KX) + EXP(-KX))/2
```



```
1170 RP = RP * ( (EXP(VX) - EXP(-VX) ) /2 ) / KX
1180 IP = IP * ( (EXP(VX) + EXP(-VX) ) /2 ) / KX
1190 '
1200 ' 's-domain to z-domain conversion
1210 T = 2 * TAN(1/2)
1220 W = 2*PI*FC
1230 M = RP^2 + IP^2
1240 D = 4 - 4*RP*T + M*T^2
1250 X0 = T^2/D
1260 X1 = 2*T^2/D
1270 X2 = T^2/D
1280 Y1 = (8 - 2*M*T^2)/D
1290 Y2 = (-4 - 4*RP*T - M*T^2)/D
1300 '
1310 ' 'LP TO LP, or LP TO HP
1320 IF LH = 1 THEN K = -COS(W/2 + 1/2) / COS(W/2 - 1/2)
1330 IF LH = 0 THEN K = SIN(1/2 - W/2) / SIN(1/2 + W/2)
1340 D = 1 + Y1*K - Y2*K^2
1350 A0 = (X0 - X1*K + X2*K^2)/D
1360 A1 = (-2*X0*K + X1 + X1*K^2 - 2*X2*K)/D
1370 A2 = (X0*K^2 - X1*K + X2)/D
1380 B1 = (2*K + Y1 + Y1*K^2 - 2*Y2*K)/D
1390 B2 = (-K^2 - Y1*K + Y2)/D
1400 IF LH = 1 THEN A1 = -A1
1410 IF LH = 1 THEN B1 = -B1
1420 '
1430 RETURN
```

FIGURE 33-8
Subroutine called from Figure 33-7.
РИСУНОК 33-8
Подпрограмма, вызываемая из программы в рисунке 33-7.

That is, we write an equation for $H(s)$, and then replaced each s with the above expression. In most cases, $T = 2\tan(1/2) = 1.093$ is used. This results in the s-domain's frequency range of 0 to π radians/second, being mapped to the z-domain's frequency range of 0 to π radians. Without going into more detail, the bilinear transform has the desired properties to convert from the s-plane to the z-plane, such as vertical lines being mapped into circles. Here is an example of how it works. For a continuous system with a single pole-pair located at $p_1 = \sigma + j\omega$ and $p_2 = \sigma - j\omega$, the s-domain transfer function is given by:

То есть мы записываем уравнение для $H(s)$, и затем заменяем каждый s вышеупомянутым выражением. В большинстве случаев, используется $T = 2\tan(1/2) = 1.093$. Это приводит к диапазону частотного s-домена от 0 до π радиан/секунда, отображаемый к частотному диапазону z-домена 0 to π радиан. Не вдаваясь в подробности, билинейная трансформанта имеет желательные свойства, чтобы преобразовать от s-плоскости в z-плоскости, типа вертикальных строк, отображаемых в кругах. Имеется пример того, как это работает. Для непрерывной системы с однополюсной парой, расположенной в $p_1 = \sigma + j\omega$ и $p_2 = \sigma - j\omega$, функция преобразования s-домена дается:

$$H(s) = \frac{1}{(s - p_1)(s - p_2)}$$

The bilinear transform converts this into a discrete system by replacing each s with the expression given in Eq. 33-10. This creates a z-domain transfer function also containing two poles. The problem is, the substitution leaves the transfer function in a very unfriendly form:

Билинейная трансформанта преобразовывает это в дискретную систему, заменяя каждый *s* выражением, данным в уравнении 33-10. Это создает функцию преобразования z-домена, также содержащую два полюса. Проблема, замена оставляет функцию преобразования в очень недружелюбной форме:

$$H(z) = \frac{1}{\left(\frac{2(1-z^{-1})}{T(1+z^{-1})} - (\sigma + j\omega)\right)\left(\frac{2(1-z^{-1})}{T(1+z^{-1})} - (\sigma - j\omega)\right)}$$

Working through the long and tedious algebra, this expression can be placed in the standard form of Eq. 33-3, and the recursion coefficients identified as:

Работая через длинную и утомительную алгебру, это выражение может быть помещено в стандартную форму уравнения 33-3, и коэффициентов рекурсии, идентифицированных как:

EQUATION 33-11
Bilinear transform for two poles. The pole-pair is located at $\sigma \pm \omega$ in the s-plane, and a_0, a_1, a_2, b_1, b_2 are the recursion coefficients for the discrete system.

УРАВНЕНИЕ 33-11
Билинейная трансформанта для двух полюсов. Пара полюса расположена в $\sigma \pm \omega$ в s-плоскости, и a_0, a_1, a_2, b_1, b_2 - коэффициенты рекурсии для дискретной системы.

$$a_0 = T^2/D$$
$$a_1 = 2T^2/D$$
$$a_2 = T^2/D$$
$$b_1 = (8 - 2MT^2)/D$$
$$b_2 = (-4 - 4\sigma T - MT^2)/D$$

where:
$$M = \sigma^2 + \omega^2$$
$$T = 2\tan(1/2)$$
$$D = 4 - 4\sigma T + MT^2$$

The variables *M*, *T*, and *D* have no physical meaning; they are simply used to make the equations shorter.
Переменные *M*, *T*, и *D* не имеют никакого физического значения; они просто используются, чтобы делать уравнения короче.

Lines 1200-1290 use these equations to convert the location of the s-domain pole-pair, held in the variables, RP and IP, directly into the recursive coefficients, held in the variables, X0, X1, X2, Y1, Y2. In other words, we have calculated an intermediate result: the recursion coefficients for one stage of a *low-pass* filter with a cutoff frequency of *one*.

Строки 1200-1290 использование этих уравнений, чтобы преобразовать расположение пары полюса s-домена, проведенной(поддержанной) в переменных, RP и IP, непосредственно в рекурсивные коэффициенты, проведенные(поддержанные) в переменных, X0, X1, X2,

Y1, Y2. Другими словами, мы вычислили промежуточный результат: коэффициенты рекурсии для одной стадии *фильтра нижних частот* с частотой отсечки *единица*.

Low-pass to Low-pass Frequency Change
Преобразование фильтра низкой частоты в фильтр низкой частоты изменяя частоту отсечки

Changing the frequency of the recursive filter is also accomplished with a conformal mapping technique. Suppose we know the transfer function of a recursive low-pass filter with a unity cutoff frequency. The transfer function of a similar low-pass filter with a new cutoff frequency, *W*, is obtained by using a **low-pass to low-pass transform**. This is also carried out by *substituting variables*, just as with the bilinear transform. We start by writing the transfer function of the unity cutoff filter, and then replace each z^{-1} with the following:

Изменение частоты рекурсивного фильтра также выполнено с методикой конформного отображения. Предположим, что мы знаем функцию преобразования о рекурсивном фильтре нижних частот с критической частотой(частотой отсечки) единица. Функция преобразования типа подобного фильтра нижних частот с новой частотой отсечки, *W*, получена, используя фильтр низкой частоты к преобразованию в фильтр низкой частоты с другой частотой отсечки. Это также выполнено, заменяя переменные, так же, как с билинейной трансформантой. Мы начинаем, запись функции фильтра останова единицы, и затем заменяем каждый z^{-1} со следующим:

EQUATION 33-12
Low-pass to low-pass transform. This is a method of changing the cutoff frequency of low-pass filters. The original filter has a cutoff frequency of unity, while the new filter has a cutoff frequency of *W*, in the range of 0 to π.

$$z^{-1} \rightarrow \frac{z^{-1} - k}{1 - kz^{-1}}$$

where:

$$k = \frac{\sin(1/2 - W/2)}{\sin(1/2 + W/2)}$$

УРАВНЕНИЕ 33-12
Фильтр низкой частоты в фильтр низкой частоты. Это - метод изменения частоты отсечки фильтров нижних частот. Первоначальный фильтр имеет частоту отсечки единица, в то время как новый фильтр имеет частоту отсечки *W*, в диапазоне 0 к π.

This provides the transfer function of the filter with the new cutoff frequency. The following design equations result from applying this substitution to the biquad, i.e., no more than two poles and two zeros:

Это обеспечивает функцию преобразования фильтра новой частотой отсечки. Следующие уравнения проекта следствие применения этой замены к biquad(биквадрату?), то есть, не больше, чем два полюса и два нуля:

EQUATION 33-13
Low-pass to low-pass conversion. The recursion coefficients of the filter with unity cutoff are shown in italics. The coefficients of the low-pass filter with a cutoff frequency of *W* are in roman.

$$a_0 = (a_0 - a_1 k + a_2 k^2)/D$$
$$a_1 = (-2a_0 k + a_1 + a_1 k^2 - 2a_2 k)/D$$
$$a_2 = (a_0 k^2 - a_1 k + a_2)/D$$
$$b_1 = (2k + b_1 + b_1 k^2 - 2b_2 k)/D$$
$$b_2 = (-k^2 - b_1 k + b_2)/D$$

where:

$$D = 1 + b_1 k - b_2 k^2$$
$$k = \frac{\sin(1/2 - W/2)}{\sin(1/2 + W/2)}$$

УРАВНЕНИЕ 33-13
Преобразование фильтра низкой частоты в фильтр низкой частоты. Коэффициенты рекурсии фильтра с остановом единицы показываются в курсиве. Коэффициенты фильт-

ра нижних частот с частотой отсечки W находятся в католике(римском шрифте?).

Low-pass to High-pass Frequency Change
Изменение Фильтр низкой частоты в Фильтр верхних частот изменяя частоту

The above transform can be modified to change the response of the system from low-pass to high-pass while simultaneously changing the cutoff frequency. This is accomplished by using a **low-pass to high-pass transform**, via the substitution:

Вышеупомянутая трансформанта может изменяться, чтобы изменить ответ системы от фильтра низкой частоты в фильтр верхних частот при одновременном изменении частоты отсечки. Это выполнено, используя фильтр низкой частоты преобразованием в фильтр верхних частот, через замену(подстановку?):

$$z^{-1} \rightarrow \frac{-z^{-1} - k}{1 + kz^{-1}}$$

where:

$$k = -\frac{\cos(W/2 + 1/2)}{\cos(W/2 - 1/2)}$$

EQUATION 33-14
Low-pass to high-pass transform. This substitution changes a low-pass filter into a high-pass filter. The cutoff frequency of the low-pass filter is *one*, while the cutoff frequency of the high- pass filter is *W*.
УРАВНЕНИЕ 33-14
Преобразование фильтр низкой частоты в фильтр верхних частот. Эта замена изменяет фильтр нижних частот в фильтр верхних частот. Частота останова фильтра нижних частот единица, в то время как частота отсечки высокочастотного фильтра - *W*.

As before, this can be reduced to design equations for changing the coefficients of a biquad stage. As it turns out, the equations are identical to those of Eq. 33-13, with only two minor changes. The value of *k* is different (as given in Eq. 33-14), and two coefficients, and , are negated in value. a 1 b 1 These equations are carried out in lines 1330 to 1410 in the program, providing the desired cutoff frequency, and the choice of a high-pass or low-pass response.

Как прежде, это может быть сокращено, чтобы проектировать уравнения для изменения(замены) коэффициентов стадии biquad. Как это собирается, уравнения идентичны таковым уравнения 33-13, только с двумя незначительными изменениями. Значение *k* различно (как дано в уравнении 33-14), и двух коэффициентах, и, инвертировано в значении a_1 b и b_1. Эти уравнения выполнены в строках от 1330 до 1410 в программе, обеспечивая желательную частоту отсечки, и выбор фильтра верхних частот или ответа фильтра низкой частоты.

The Best and Worst of DSP
Лучшее и Худшее в ЦОС

This book is based on a simple premise: **most DSP techniques can be used and understood with a minimum of mathematics**. The idea is to provide scientists and engineers *tools* for solving the DSP problems that arise in their non-DSP research or design activities.

Эта книга основана на простой предпосылке: большинство методов ЦОС может использоваться и понято с минимумом математики. Идея состоит в том, чтобы обеспечить ученых и проектировщиков инструментальными средствами для решения проблем ЦОС, которые возникают в их не - ЦОС, исследованиях или действий проектирования.

These last four chapters are the other side of the coin: DSP techniques that can *only* be understood through extensive math. For example, consider the Chebyshev-Butterworth filter just described. This is the *best* of DSP, a series of elegant mathematical steps leading to an optimal solution. However, it is also the *worst* of DSP, a design method so complicated that most scientists and engineers will look for another alternative.

Последние четыре главы - другая сторона монеты: методы ЦОС, которые могут *только* быть поняты через обширную математику. Например, рассмотрите фильтр Чебышева-Буттерворта, только описанный. Это - *лучшее* из ЦОС, ряд изящных математических шагов, ведущих к оптимальному решению. Однако, это также самое *плохое* из ЦОС, метод проекта настолько сложен, что большинство ученых и инженеров будет искать другую альтернативу.

Where do you fit into this scheme? This depends on *who* your are and *what* you plan on using DSP for. The material in the last four chapters provides the theoretical basis for signal processing. If you plan on pursuing a *career* in DSP, you need to have a detailed understanding of this mathematics. On the other hand, specialists in other areas of science and engineering only need to know how DSP is *used*, not how it is *derived*. To this group, the theoretical material is more of a background, rather than a central topic.

Где Вы вписываетесь в эту схему? Это зависит от того, кем вы являетесь и для чего Вы планируете использовать ЦОС. Материал в последних четырех главах обеспечивает теоретическое основание для обработки сигнала. Если Вы планируете преследовать карьеру в ЦОС, Вы должны иметь детальное понимание этой математики. С другой стороны, специалисты в других областях науки и техники должны знать, только, как ЦОС используется, не, как это получено. К этой группе, теоретический материал - большее количество подготовки, скорее чем центральная тема.