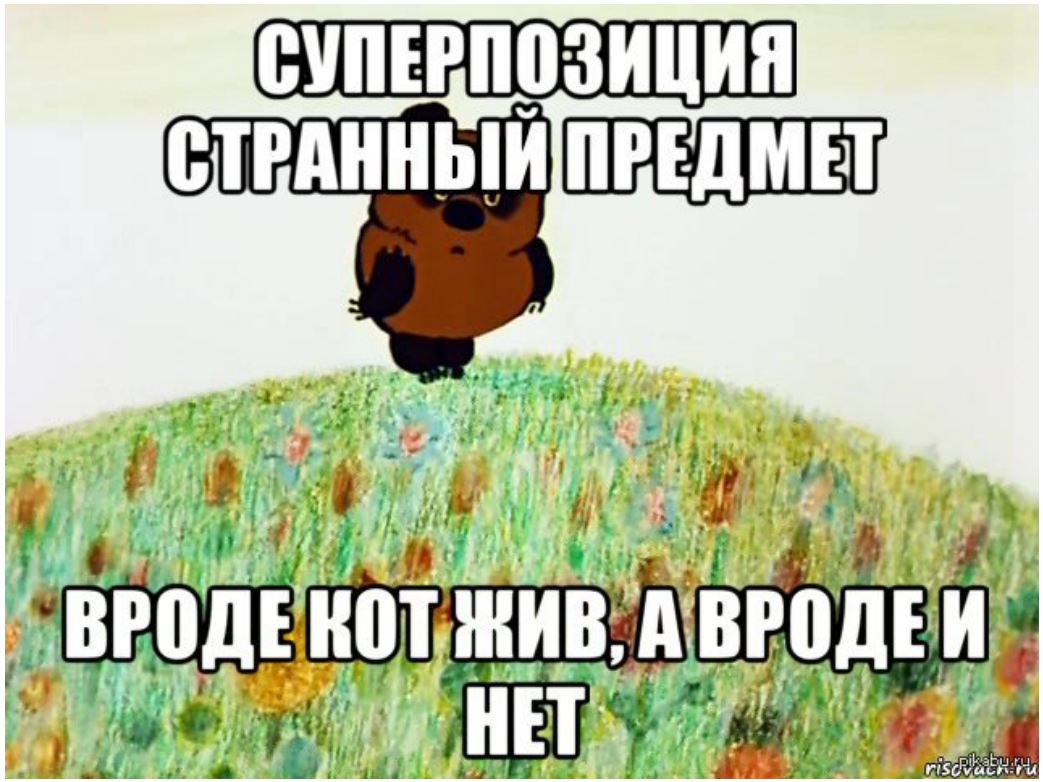
Математическое моделирование РТУ и С

Лекция 7. Моделирование линейных звеньев

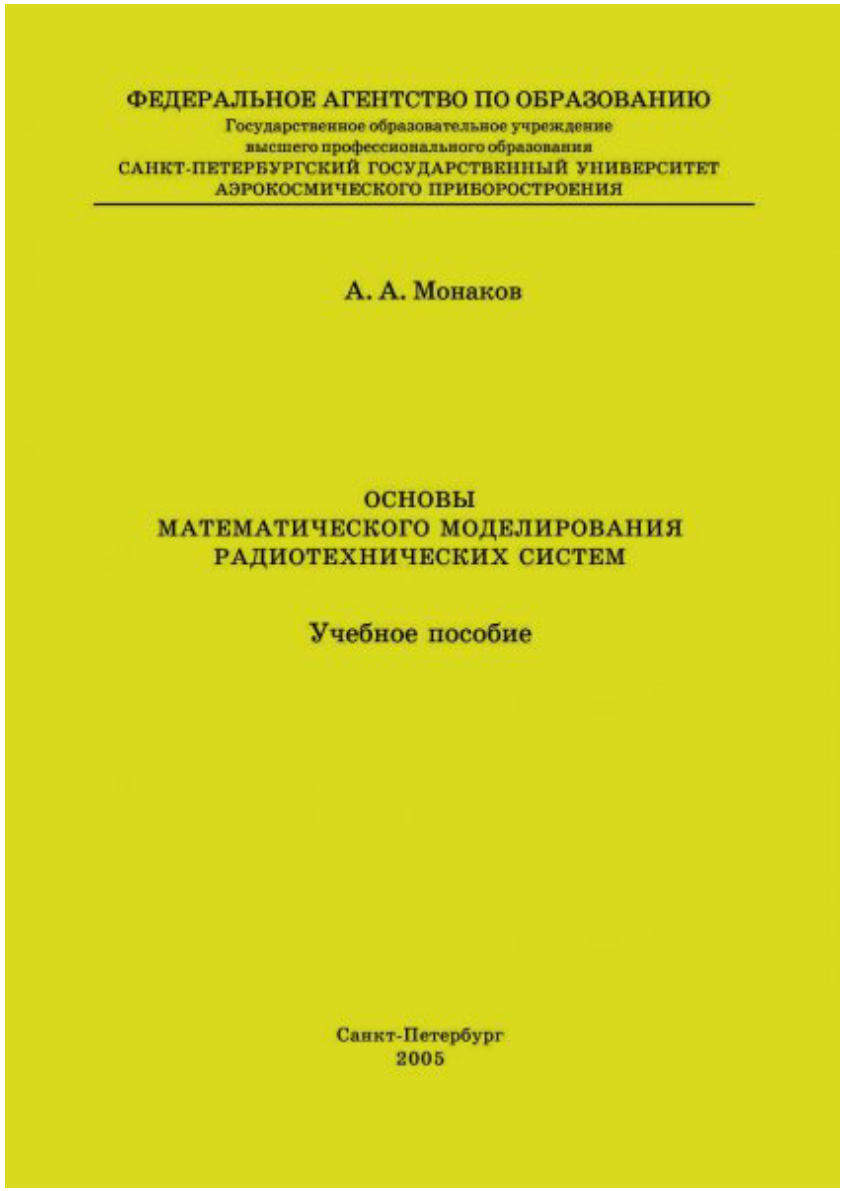


Преподаватель:
Корогодин Илья
korogodin@srns.ru

Литература

Монаков А.А. Основы математического моделирования радиотехнических систем. Учебное пособие. – СПб.: ГУАП, 2005. – 100с.

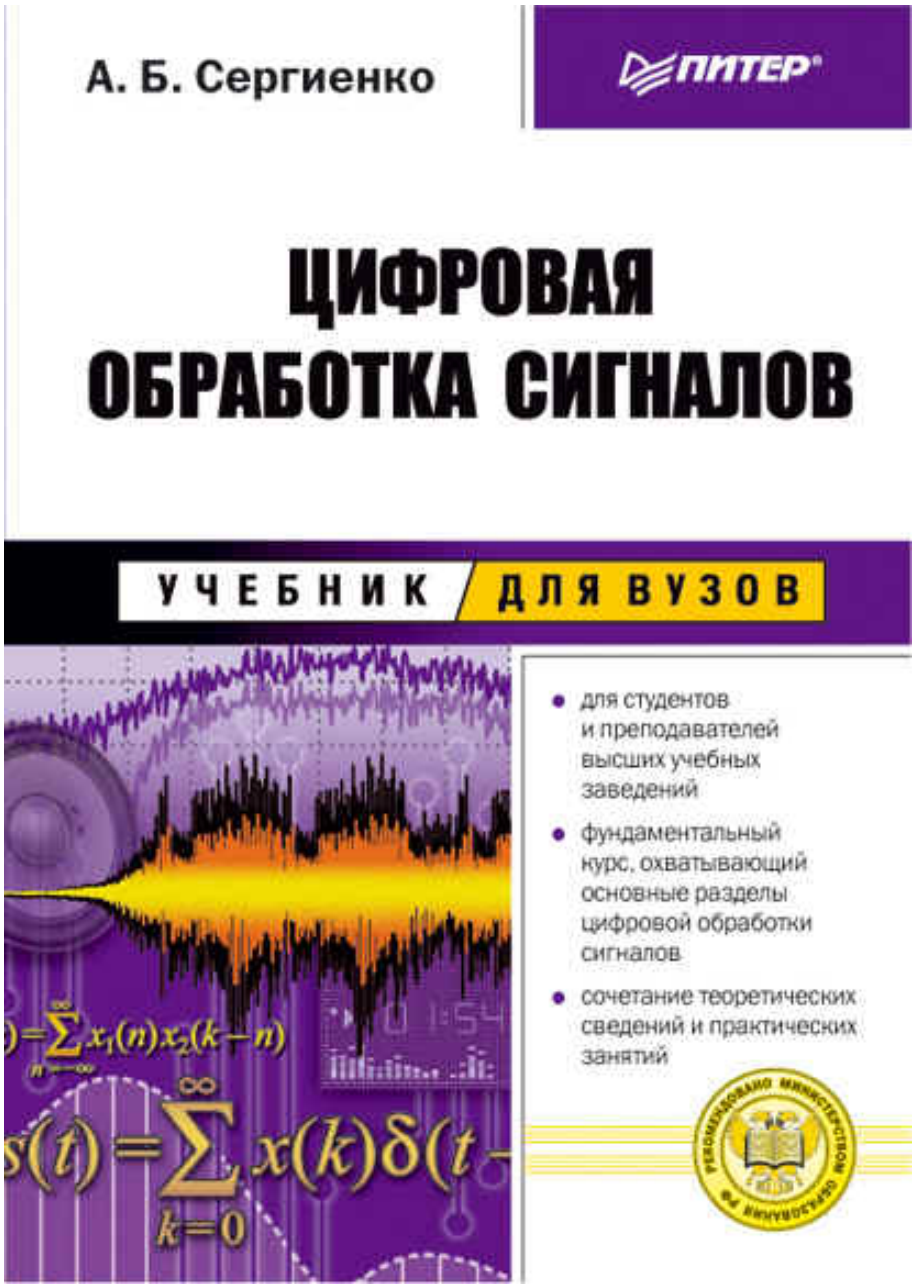
Глава 2, Раздел 2.1: Моделирование линейных звеньев



Литература

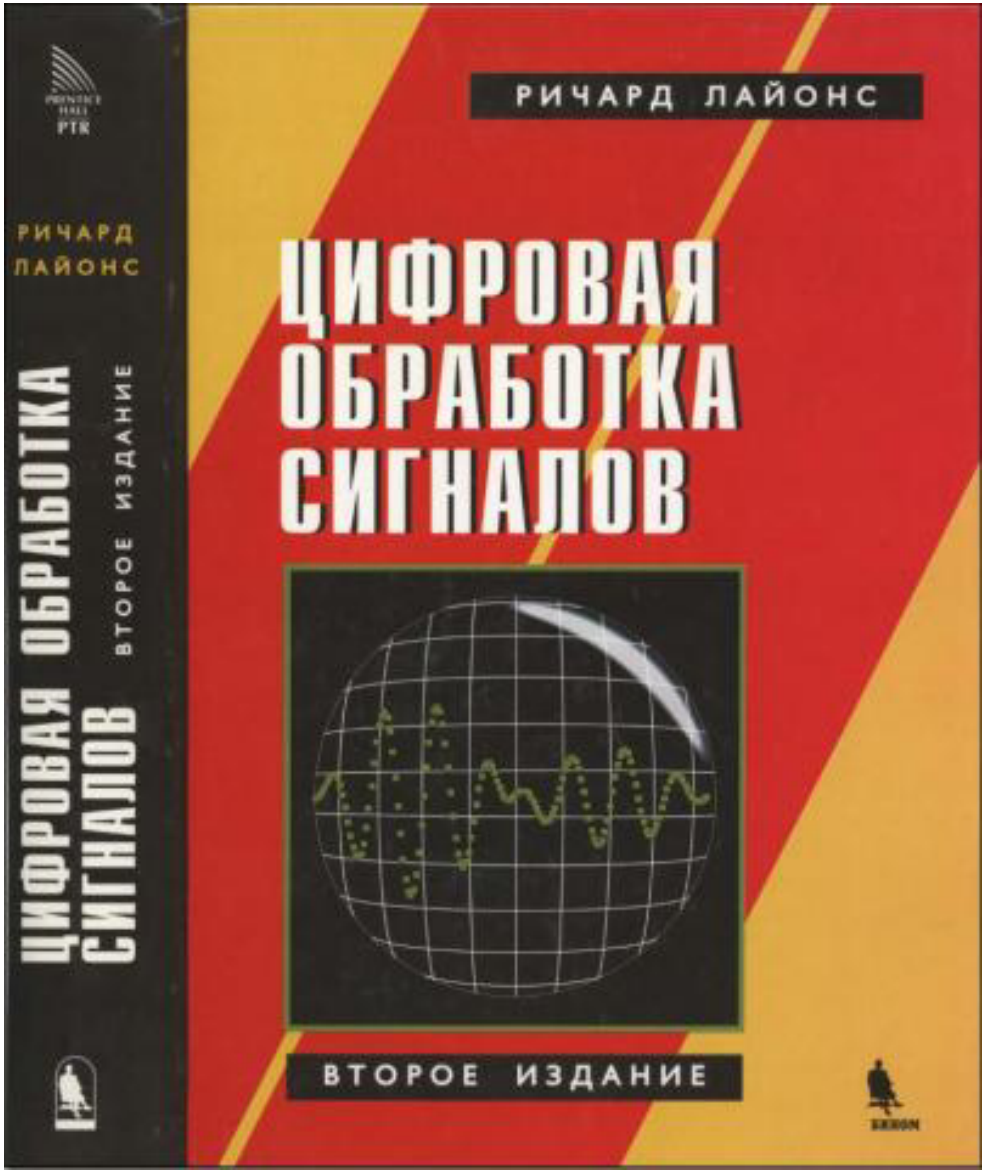
А.Б.Сергиенко. Цифровая обработка сигналов. СПб, Питер, 2002. — 608 с.: ил.

А.Б.Сергиенко.
Signal Processing Toolbox – обзор:
http://matlab.exponenta.ru/signalprocess/book2/index.php



Литература

Ричард Лайонс - Цифровая обработка сигналов / Understanding Digital Signal Processing, 2006



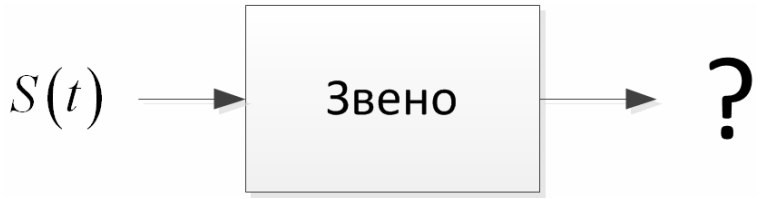
Глава 5. Фильтры с импульсной характеристикой конечной длины

Глава 6. Фильтры с импульсной характеристикой бесконечной длины

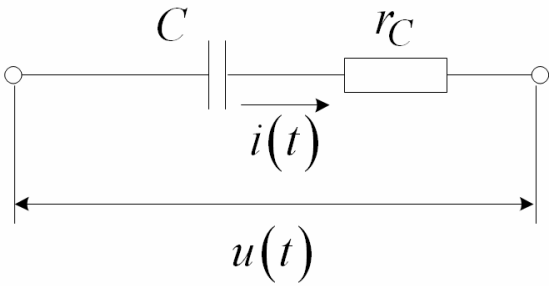
Глава 7. Специальные КИХ-фильтры нижних частот

Преобразование сигналов

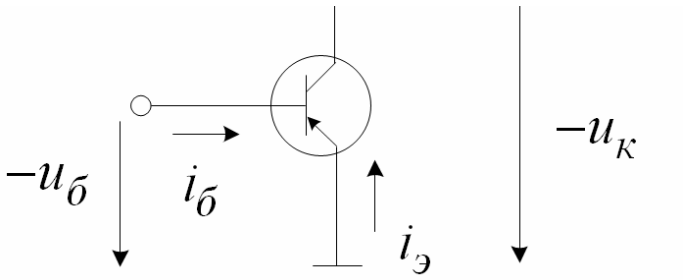
Любое обрабатывающее радиосигнал устройство может быть представлено как совокупность линейных и нелинейных звеньев.



Формально отличие в дифурах (линейные/нелинейные):



$$u(t) = i(t) r_C + \frac{1}{C} \int_0^t i(t)\, dt$$



$$i = I_s \left(\exp\left(u / nV \right) - 1 \right)$$

Но важно следствие - при действии суммы сигналов отклик звена есть суперпозиция откликов на каждое воздействие в отдельности:

$$K(p) \sum \alpha_i S_i = \sum \alpha_i K(p) S_i$$

Коэффициент передачи

Линейное звено описывается дифуром:

$$a_M y^{(M)}(t) + \ldots + a_1 y^{(1)}(t) + a_0 y(t) = b_N x^{(N)}(t) + \ldots + b_1 x^{(1)}(t) + b_0 x(t)$$

Нам достаточно научиться его решать для воздействия $x(t) = e^{j\omega t}$,

а потом воспользоваться преобразованием Фурье и линейностью

Решение лежит на поверхности - $y(t) = \dot{U} e^{j\omega t}$:

$$(j\omega)^M a_M \dot{U} e^{j\omega t} + \ldots + (j\omega) a_1 \dot{U} e^{j\omega t}, + a_0 \dot{U} e^{j\omega t}$$

$$= (j\omega)^N b_N e^{j\omega t} + \ldots + (j\omega) b_1 e^{j\omega t}, + b_0 e^{j\omega t}$$

откуда

$$\dot{U} = \dot{U}(j\omega) = \frac{(j\omega)^N b_N + \ldots + (j\omega) b_1, + b_0}{(j\omega)^M a_M + \ldots + (j\omega) a_1 + a_0}$$

Коэффициент передачи

Нетрудно заметить, что в этом случае

$$y(t)=\dot{U}x(t)$$

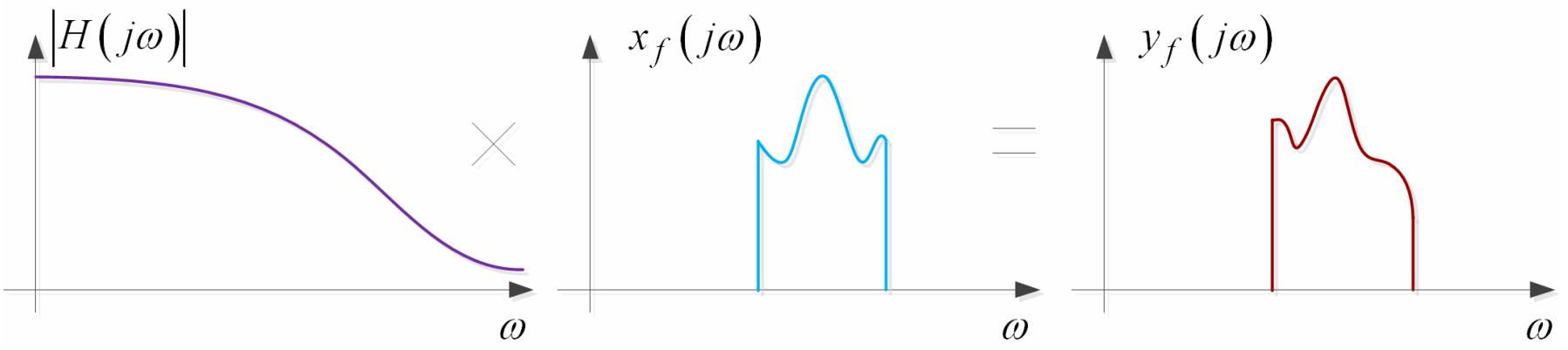
Да это же не только комплексная амплитуда,
но ещё и коэффициент передачи (transfer function)!

Обозначим $s=j\omega$

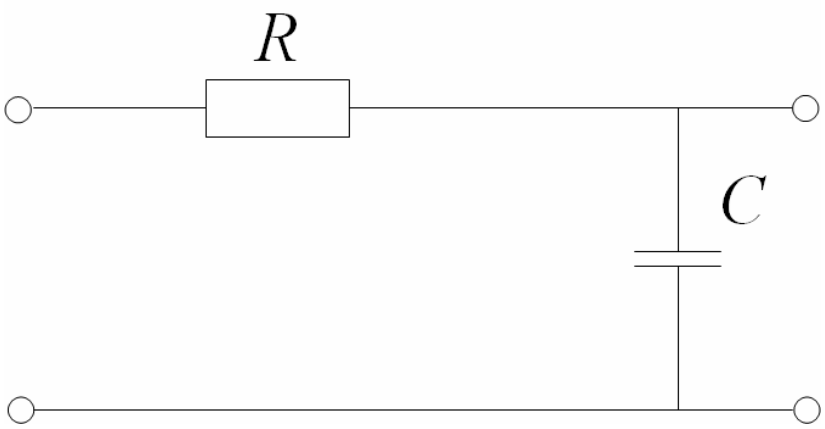
$$H(s)=\dot{U}(s)=\frac{b_N s^N+\ldots+b_1 s+b_0}{a_M s^M+\ldots+a_1 s+a_0}$$

Благодаря линейности для сигналов с произвольным спектром имеем

$$y_f(s)=H(s)x_f(s),\ y_f(s)=f\{y(t)\},\ x_f(s)=f\{x(t)\}$$



Коэффициент передачи



$$x(t) = i(t)R + y(t),\ \ i(t) = C\frac{dy(t)}{dt} \Rightarrow$$

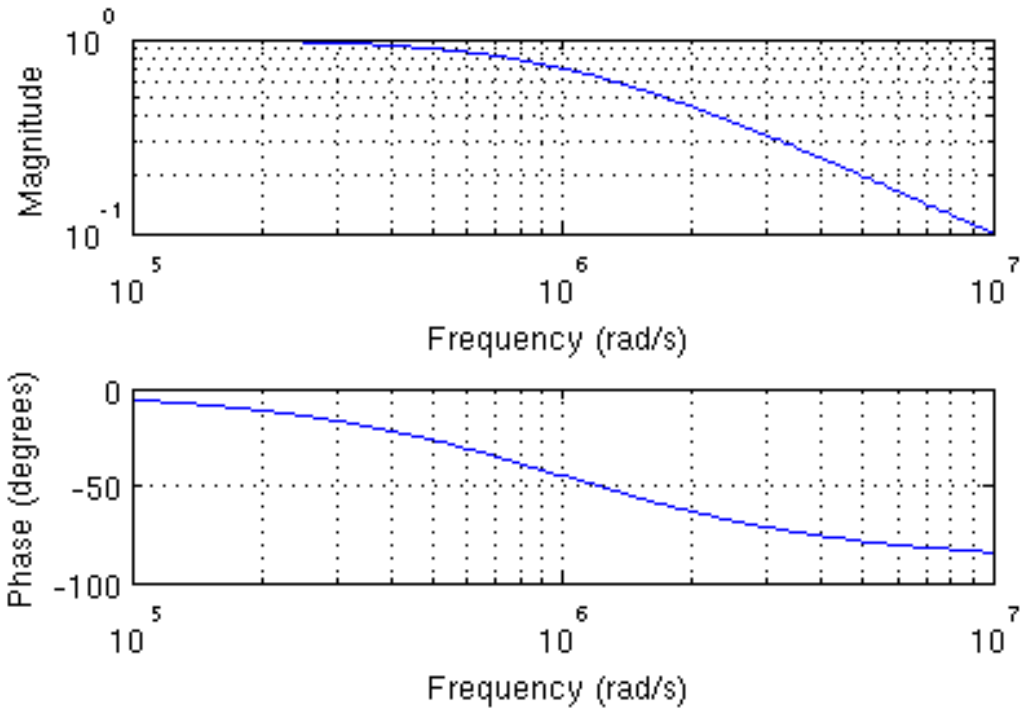
$$RC\frac{dy(t)}{dt} + y(t) = x(t) \Rightarrow$$

$$a_1 = RC;\ a_0 = 1;\ b_0 = 1; \Rightarrow$$

$$H(s) = \frac{1}{RC \cdot s + 1}$$

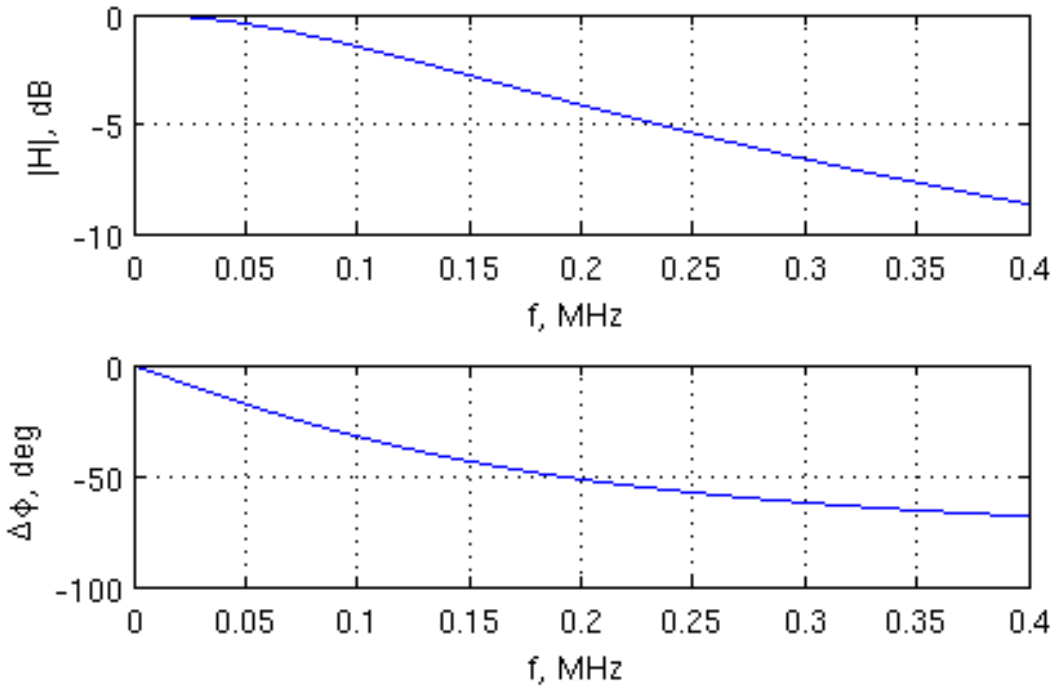
В MATLAB есть функции для работы с линейными звеньями

```
clear all; clc; close all;

RC = 1e-6;

a = [RC 1];
b = [1];

freqs(b, a);
```



Коэффициент передачи

```
clear all; clc; close all;
RC = 1e-6;

a = [RC 1]; b = [1];
Fs = 100; Fmax = 4e5; f = 0:Fs:Fmax;
H = freqs(b, a, 2*pi*f);

figure(1); subplot(2,1,1);
plot(f/1e6, 20*log10(abs(H)));
subplot(2,1,2);
plot(f/1e6, rad2deg(unwrap(angle(H))));
```



freqs

Frequency response of analog filters

Syntax

```
h = freqs(b,a,w)
[h,w] = freqs(b,a,n)
freqs
```

Description

`freqs` returns the complex frequency response $H(j\omega)$ (Laplace transform) of an analog filter

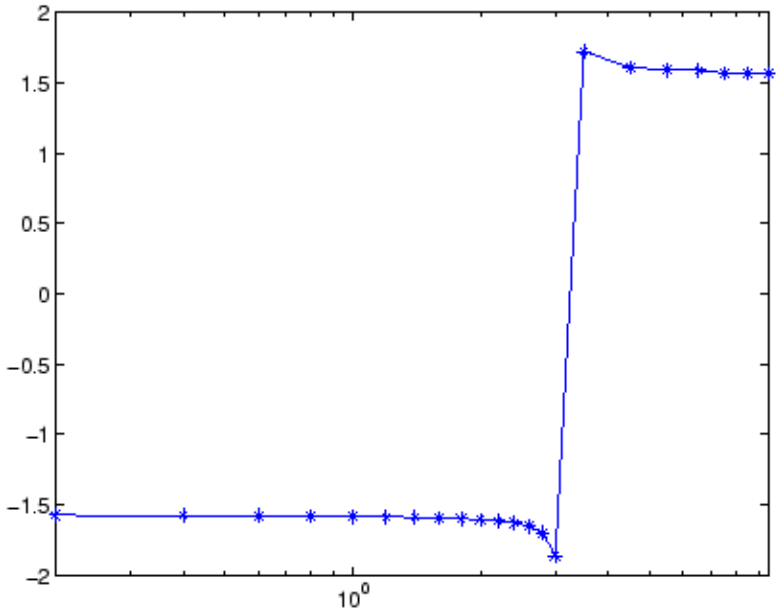
$$H(s) = \frac{B(s)}{A(s)} = \frac{b(1)s^n + b(2)s^{n-1} + \cdots + b(n+1)}{a(1)s^m + a(2)s^{m-1} + \cdots + a(m+1)}$$

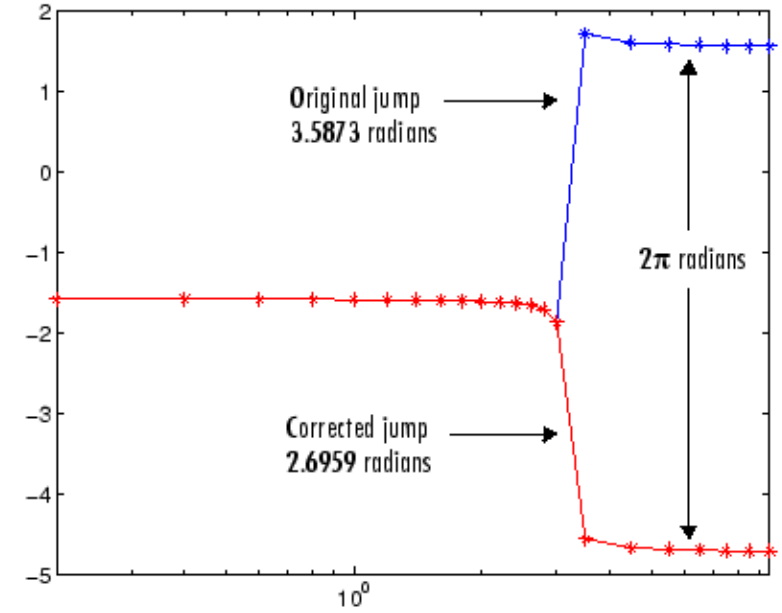
given the numerator and denominator coefficients in vectors b and a.

`h = freqs(b,a,w)` returns the complex frequency response of the analog filter specified by coefficient vectors b and a. `freqs` evaluates the frequency response along the imaginary axis in the complex plane at the angular frequencies in rad/sec specified in real vector `w`, where `w` is a vector containing more than one frequency.

`[h,w] = freqs(b,a,n)` uses n frequency points to compute the frequency response h, where n is a real, scalar value. The frequency vector `w` is auto-generated and has length n. If you omit n as an input, 200 frequency points are used. If you do not need the generated frequency vector returned, you can use the form `h = freqs(b,a,n)` to return only the frequency response h.

Функция unwrap





unwrap

Correct phase angles to produce smoother phase plots

Syntax

```
Q = unwrap(P)
Q = unwrap(P,tol)
Q = unwrap(P,[],dim)
Q = unwrap(P,tol,dim)
```

Description

`Q = unwrap(P)` corrects the radian phase angles in a vector P by adding multiples of $\pm 2\pi$ when absolute jumps between consecutive elements of P are greater than or equal to the default jump tolerance of π radians. If P is a matrix, unwrap operates columnwise. If P is a multidimensional array, unwrap operates on the first nonsingleton dimension.

`Q = unwrap(P,tol)` uses a jump tolerance `tol` instead of the default value, π.

`Q = unwrap(P,[],dim)` unwraps along `dim` using the default tolerance.

`Q = unwrap(P,tol,dim)` uses a jump tolerance of `tol`.

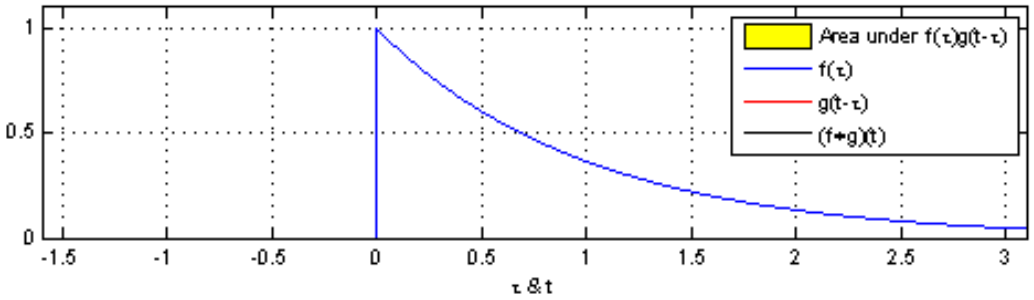
Импульсная характеристика

Умножению в частотной области соответствует свертка во временной

$$y(t)=\int_{-\infty}^{+\infty} h(\tau)x(t-\tau)d\tau$$

$$h(t)=\frac{1}{2\pi}\int_{-\infty}^{+\infty} H(j\omega)e^{j\omega t}d\omega$$

Для нашей RC-цепи и П-импульса:



А можно и через преобразование Лапласа: $$h(t)=\frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} H(s)e^{st}ds$$

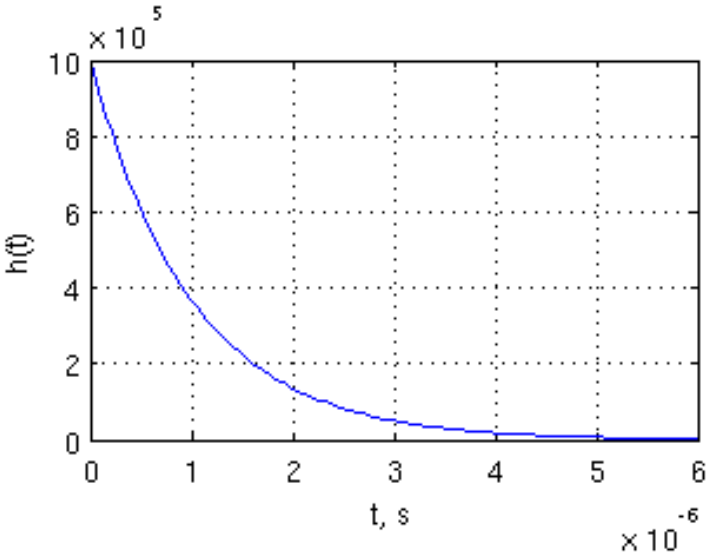
Это интегрирование по линии, параллельной мнимой оси.
Выбором сигмы все особенности подынтегральной функции оставляют слева от контура интегрирования.

Обратно: $$H(j\omega)=\int_{0}^{+\infty} h(t)e^{-j\omega t}dt \leftrightarrow H(s)=\int_{0}^{+\infty} h(t)e^{-st}dt$$

Импульсная характеристика

Для построения импульсной характеристики можно воспользоваться функциями Control System Toolbox

```
RC = 1e-6; a = [RC 1]; b = [1];

sys = tf(b,a);
[y,t] = impulse(sys); % Без `[y, t] =`
% сразу построит график

figure(1); plot(t, y);
xlabel('t, s'); ylabel('h(t)');
grid on
```



impulse

Impulse response plot of dynamic system

Syntax

```
impulse
impulse(sys)
impulse(sys,t)
```

Description

`impulse` calculates the unit impulse response of a dynamic system. The impulse response is the response to a Dirac input $\delta(t)$ for continuous-time systems and to a unit pulse at $t = 0$ for discrete-time systems. Zero initial state is assumed in the state-space case. When invoked without left-hand arguments, this function plots the impulse response on the screen.

tf

Convert unconstrained MPC controller to linear transfer function

Syntax

```
sys=tf(MPCobj)
```

Description

The `tf` function computes the transfer function of the linear controller `ss(MPCobj)` as an LTI system in `tf` form corresponding to the MPC controller when the constraints are not active. The purpose is to use the linear equivalent control in Control System Toolbox software for sensitivity and other linear analysis.

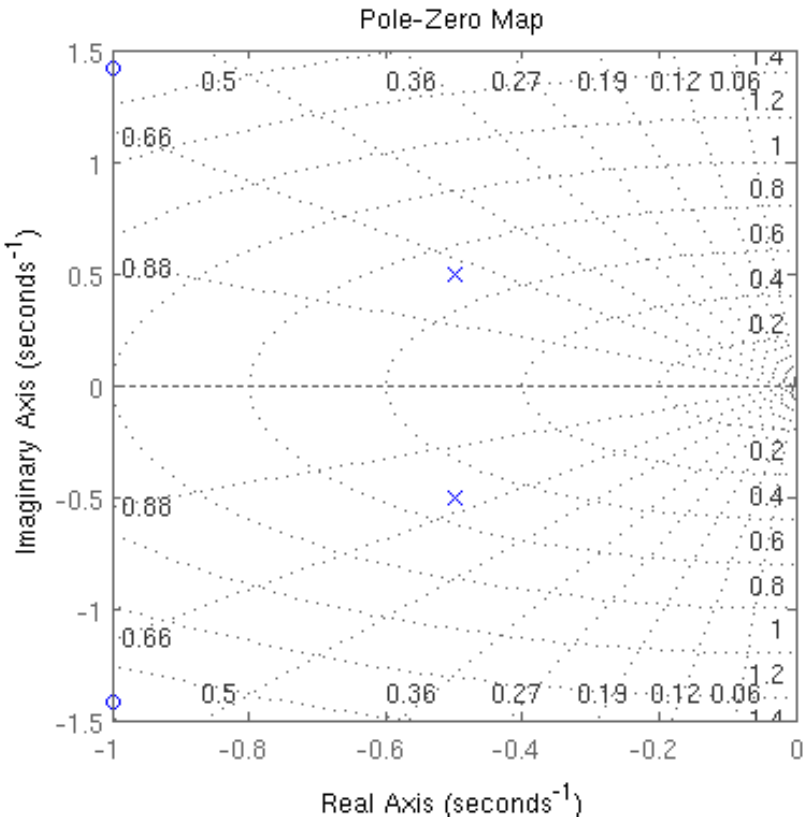
Нули и полюсы

Функцию передачи можно представить в виде

$$H(s) = k\frac{(s - z_N)(s - z_{N-1})\cdot\ldots\cdot(s - z_1)}{(s - p_M)(s - p_{M-1})\cdot\ldots\cdot(s - p_1)}$$

здесь $k = \frac{b_N}{a_M}$ **- коэффициент усиления,**

z_i **- нули,** p_i **- полюсы.**



tf2zp

Convert transfer function filter parameters to zero–pole–gain form

Syntax

```
[z,p,k]=tf2zp(b,a)
```

Description

`tf2zp` finds the zeros, poles, and gains of a continuous–time transfer function.

> **Note** You should use `tf2zp` when working with positive powers ($s^2 + s + 1$), such as in continuous–time transfer functions. A similar function, `tf2zpk`, is more useful when working with transfer functions expressed in inverse powers ($1 + z^{-1} + z^{-2}$), which is how transfer functions are usually expressed in DSP.

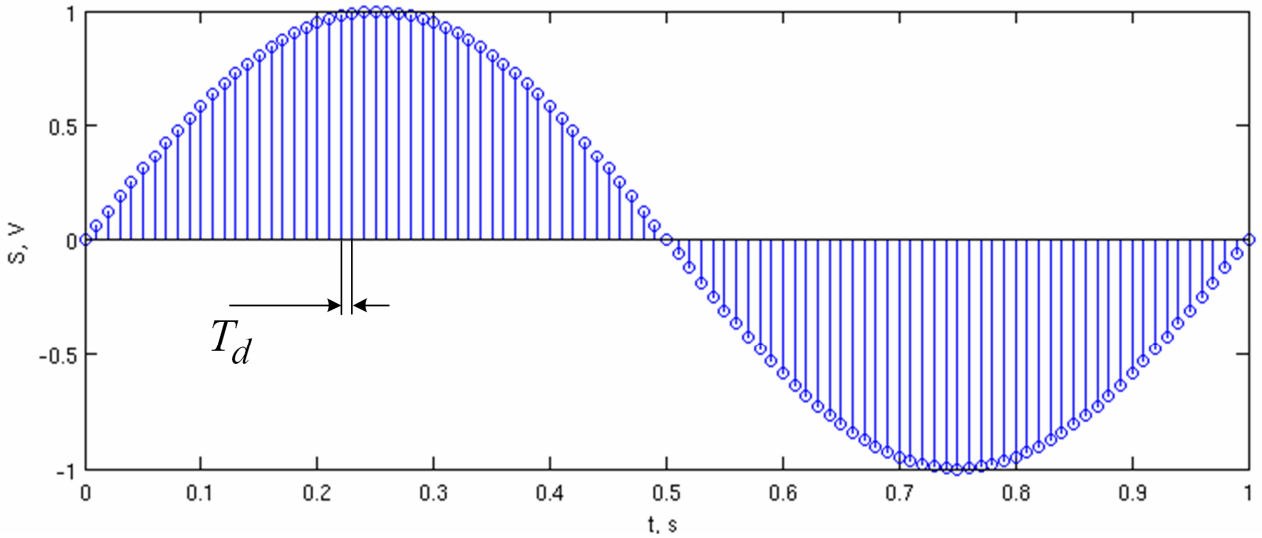
`[z,p,k] = tf2zp(b,a)` finds the matrix of zeros z, the vector of poles p, and the associated vector of gains k from the transfer function parameters b and a:

- The numerator polynomials are represented as columns of the matrix b.
- The denominator polynomial is represented in the vector a.

$\mathrm{Re}(p_i) < 0 \Leftrightarrow$ **устойчивость**

Цифровые фильтры

Всё это здорово, наглядно и удобно описывает аналоговые системы, но нам же нужно уметь их моделировать – получать отклик на сигнал



И входные, и выходные сигналы в машине мы представляем в виде дискретных последовательностей:

$$x_k = x\left(t_k\right) \rightarrow y_k \approx y\left(t_k\right)$$

Нужна модель, для которой это приближенное равенство выполняется как можно точнее.

Да это же задача синтеза цифрового фильтра по аналоговому прототипу!

Импульсная характеристика

Т.к. система линейна, то может описываться только уравнением вида:

$$y_k + a_1 y_{k-1} + a_2 y_{k-2} + \ldots + a_M y_{k-M} =$$

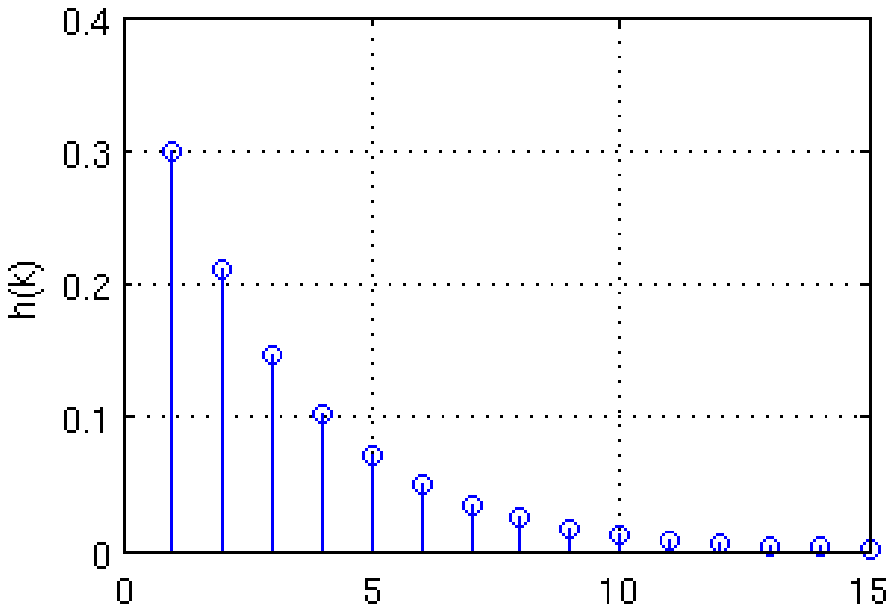
$$= b_0 x_k + b_1 x_{k-1} + b_2 x_{k-2} + \ldots + b_N x_{k-N}$$

Непрерывные линейные системы характеризуются импульсной характеристикой – откликом на дельта-функцию.

Для дискретной системы можем найти отклик на единичный импульс:

$$y_k = y_{k-1} + 0.3 \cdot \left(x_k - y_{k-1} \right) = 0.7 \cdot y_{k-1} + 0.3 \cdot x_k$$ **- ФНЧ**

```
clear all; close all; clc
x = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
km = 1:length(x);
for k = km
    if k > 1
        y(k) = 0.7*y(k-1) + 0.3*x(k);
    else
        y(k) = 0.3*x(k);
    end
end
figure(1); stem(km, y)
```

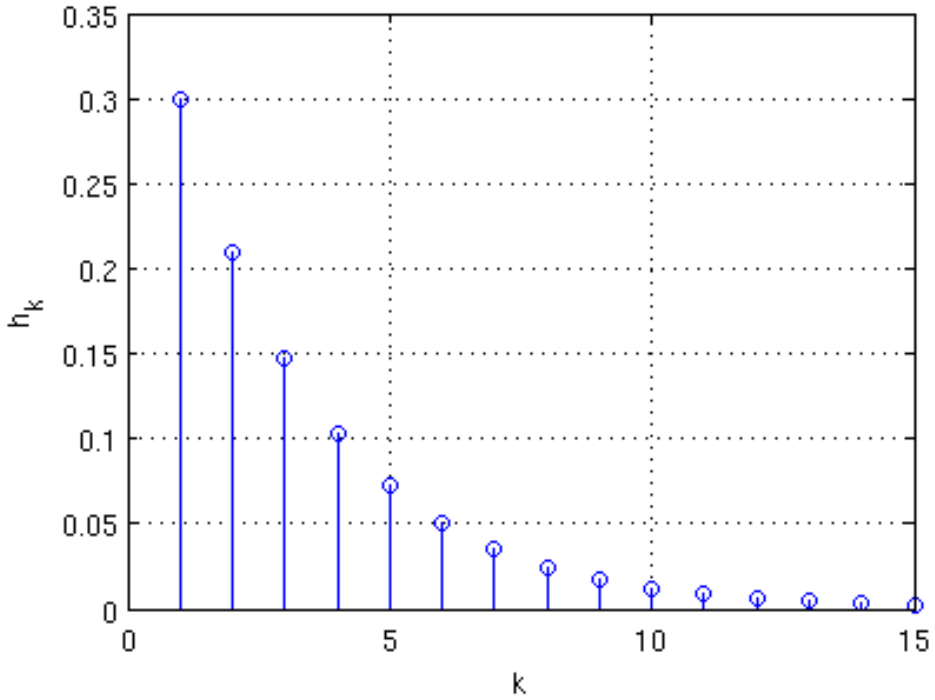


impz()

$$y_k = 0.7 \cdot y_{k-1} + 0.3 \cdot x_k \rightarrow$$

$$\rightarrow a_0 = 1,\ a_1 = -0.7,\ b_0 = 0.3$$

```
clear all; close all; clc

a = [1 -0.7]; b = [0.3];
h = impz(b, a, 15);

figure(1); stem(h);
xlabel('k'); ylabel('h_k'); grid on
```



impz

Impulse response of digital filter

Syntax

```
[h,t] = impz(b,a)
[h,t] = impz(b,a,n)
[h,t] = impz(b,a,n,fs)
impz(b,a)
impz(Hd)
```

Description

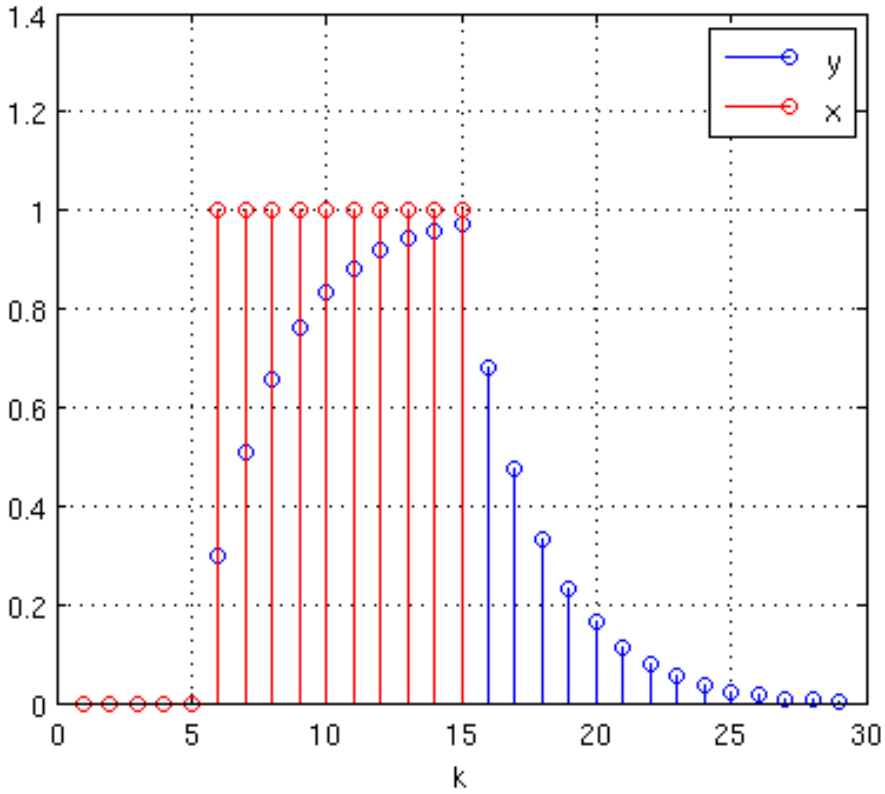
`[h,t] = impz(b,a)` computes the impulse response of the filter with numerator coefficients b and denominator coefficients a. `impz` chooses the number of samples and returns the response in the column vector h and sample times in the column vector t (where `t = [0:n-1]'`, and `n = length(t)` is computed automatically).

`[h,t] = impz(b,a,n)` computes n samples of the impulse response when n is an integer (`t = [0:n-1]'`). If n is a vector of integers, impz computes the impulse response at those integer locations, starting the response computation from 0 (and `t = n` or `t = [0 n]`). If, instead of n, you include the empty vector `[]` for the second argument, the number of samples is computed automatically by default.

Свертка: conv(), deconv()

Т.к. система линейная, то нынешний выход есть сумма реакций:

$$y_k = \sum_{n=-\infty}^{k} x_n h_{k-n}$$



Можем воспользоваться функциями conv и deconv

```
clear all; close all; clc

a = [-0.7]; b = [0.3];

xh = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
km = 2:length(xh); h(1) = b(1)*xh(1);
for k = km
        h(k) = -a(1)*h(k-1) + b(1)*xh(k);
end

x = [0 0 0 0 0 1 1 1 1 1 1 1 1 1 1];

y = conv(x, h);
xdec = deconv(y, h);

figure(1)
stem(1:length(y), y); hold on
stem(1:length(xdec), xdec, 'r'); hold off
grid on; legend('y', 'x'); xlabel('k')
```

Transfer function

Вспоминаем РЦС, z-преобразование и его свойства:

$$y_k = \sum_{n=-\infty}^{+\infty} x_n h_{k-n} = \sum_{n=-\infty}^{k} x_n h_{k-n} \Leftrightarrow$$

$$\Leftrightarrow Y(z) = H(z)X(z),\ H(z) = \sum_{k=0}^{\infty} h_k z^{-k}$$

Или из уравнения:

$$y_k + a_1 y_k z^{-1} + \ldots + a_M y_k z^{-M} = b_0 x_k + b_1 x_k z^{-1} + \ldots + b_N x_k z^{-N}$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + \ldots + a_M z^{-M}}$$

Связь с преобразованием Фурье:

$$H(j\omega) = H(z)\Big|_{z=e^{j\omega T}} = \sum_{k=0}^{\infty} h_k e^{-j\omega T} \qquad H(z) = \frac{0.3}{1 - 0.7 z^{-1}}$$

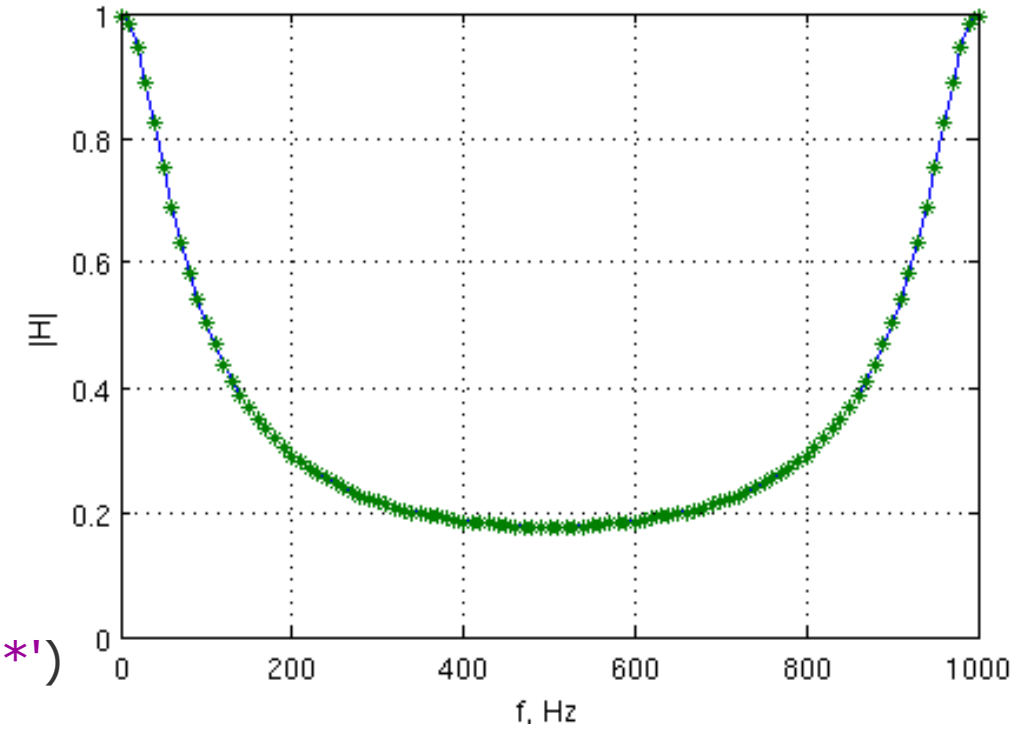
Transfer function

```
clear all; close all; clc
a = [-0.7]; b = [0.3];

xp = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
km = 1:length(x);
for k = km
    if k > 1
        h(k) = -a(1)*h(k-1) + b(1)*xh(k);
    else
        h(k) = b(1)*xh(k);
    end
end

T = 0.001; f = 0:(1/T/100):(1/T);
z = exp(1i*2*pi*f*T);

H2 = 0;
for k = km
    H2 = H2 + h(k) * z.^-k;
end

H1 = 0.3 ./ (1 - 0.7 * z.^-1);
figure(1); plot(f, abs(H1), f, abs(H2), '*')
xlabel('f, Hz'); ylabel('|H|'); grid('on');
```

$$H(j\omega) = H(z)\Big|_{z=e^{j\omega T}} = \sum_{k=0}^{\infty} h_k e^{-j\omega T}$$

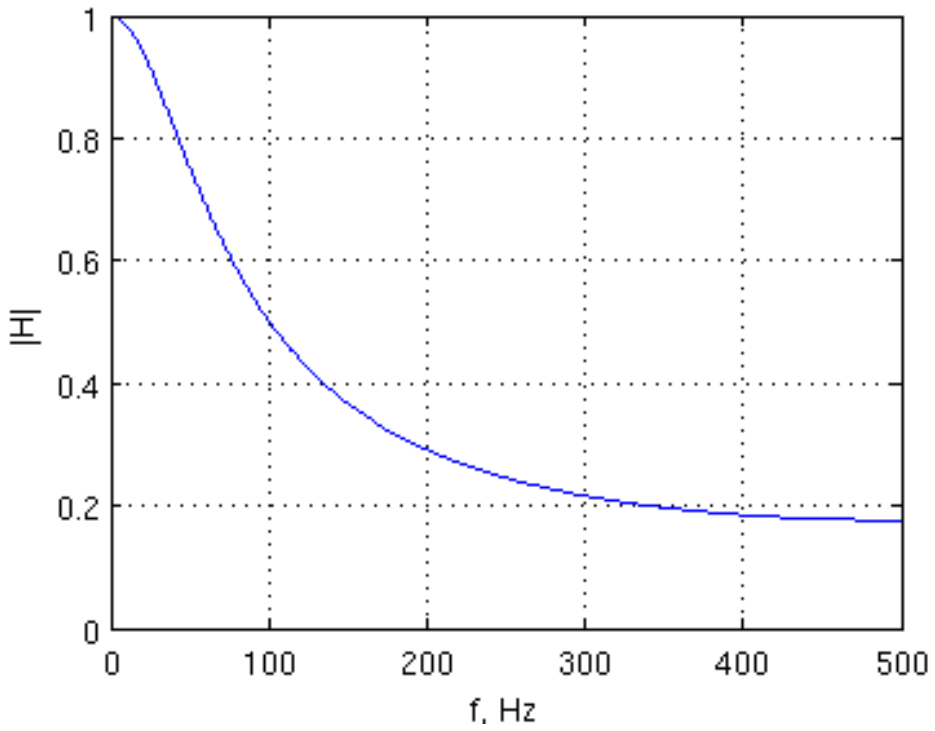
$$y_k = 0.7 \cdot y_{k-1} + 0.3 \cdot x_k \rightarrow$$

$$H(z) = \frac{0.3}{1 - 0.7 z^{-1}}$$



freqz()

```
clear all; close all; clc

a = [1 -0.7]; b = [0.3];
H = freqz(b, a);

T = 0.001;
f = ( (1:length(H)) - 1)/ length(H)*1/T/2 ;

figure(1); plot(f, abs(H));
ylabel('|H|'); grid on; xlabel('f, Hz');
```



freqz

Frequency response of digital filter

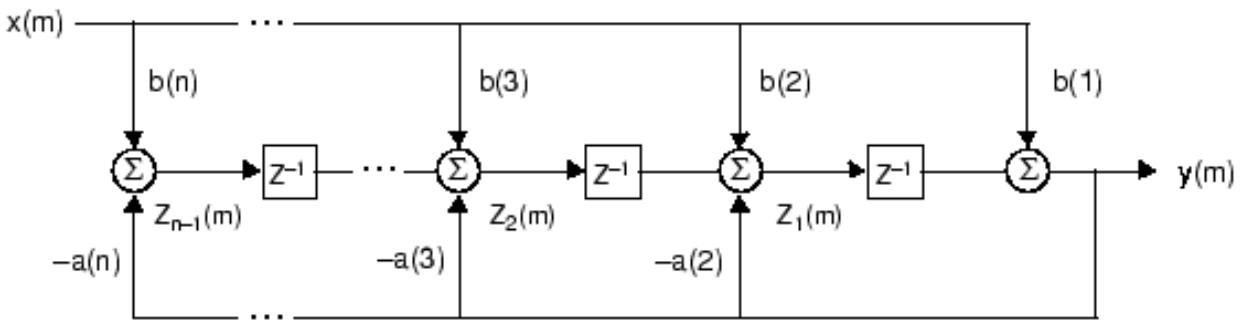
Syntax

```
[h,w]=freqz(b,a,n)
h=freqz(b,a,w)
[h,w]=freqz(b,a,n,'whole')
[h,f]=freqz(b,a,n,fs)
h=freqz(b,a,f,fs)
[h,f]=freqz(b,a,n,'whole',fs)
freqz(b,a,...)
freqz(Hd)
```

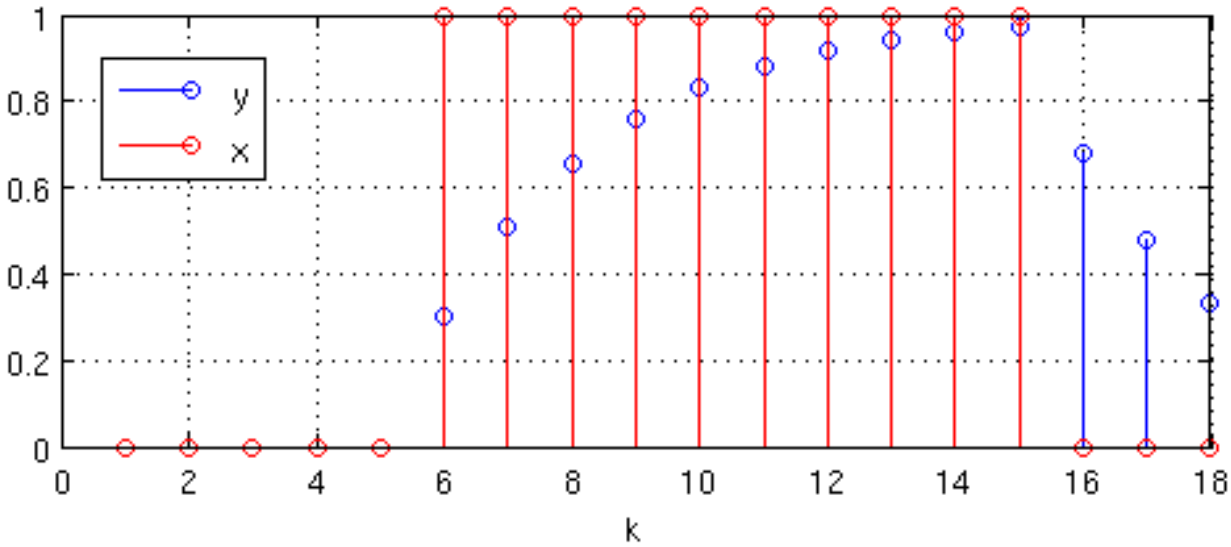
Description

`[h,w] = freqz(b,a,n)` returns the frequency response vector h and the corresponding angular frequency vector w for the digital filter whose transfer function is determined by the (real or complex) numerator and denominator polynomials represented in the vectors b and a, respectively. The vectors h and w are both of length n. n must be a positive integer greater than or equal to two. The angular frequency vector w has values ranging from 0 to π radians per sample. If you do not specify the integer n, or you specify it as the empty vector `[]`, the frequency response is calculated using the default value of 512 samples.

filter()




```
clear all; close all; clc

a = [1 -0.7]; b = [0.3];

x = [0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0];
y = filter(b, a, x);

figure(1)
stem(1:length(y), y); hold on
stem(1:length(x), x, 'r'); hold off
grid on; legend('y', 'x'); xlabel('k')
```



filter

1-D digital filter

Syntax

```
y = filter(b,a,X)
[y,zf] = filter(b,a,X)
[y,zf] = filter(b,a,X,zi)
y = filter(b,a,X,zi,dim)
[...] = filter(b,a,X,[],dim)
```

Description

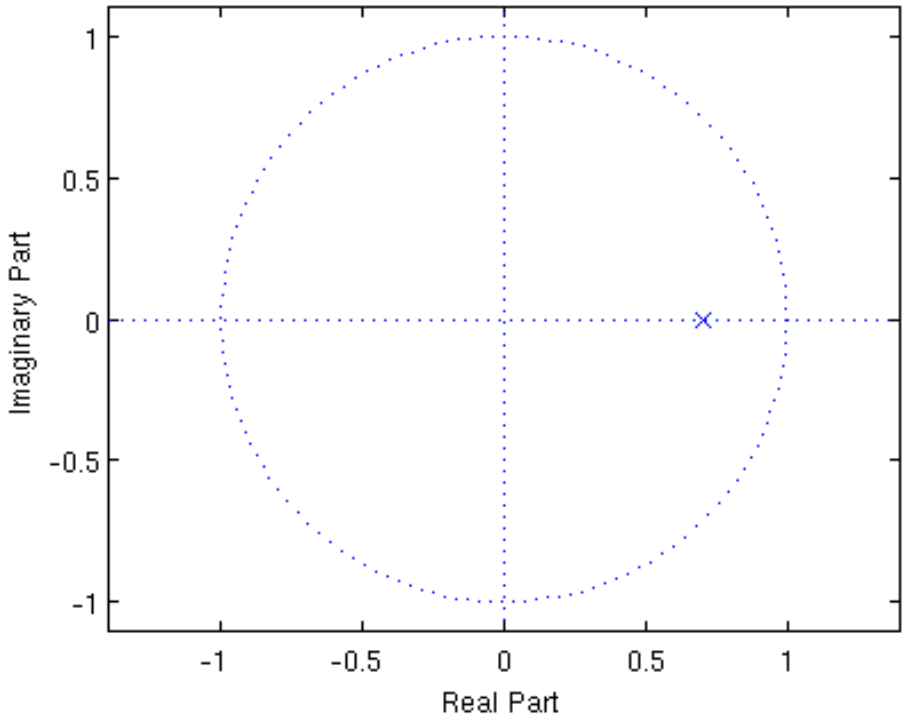
The `filter` function filters a data sequence using a digital filter which works for both real and complex inputs. The filter is a *direct form II transposed* implementation of the standard difference equation (see "Algorithm").

`y = filter(b,a,X)` filters the data in vector X with the filter described by numerator coefficient vector b and denominator coefficient vector a. If `a(1)` is not equal to 1, `filter` normalizes the filter coefficients by `a(1)`. If `a(1)` equals 0, `filter` returns an error.

Нули и полюсы

Коэффициенты ПФ – строки, нули/полюсы - столбцы

```
clear all; close all; clc

a = [1 -0.7]; b = [0.3];
[z, p, k] = tf2zp(b, a);

% zplane(b, a);
zplane(z, p);
```



$$H(z) = k\frac{\left(1-z_1 z^{-1}\right)\left(1-z_2 z^{-1}\right)\cdot\ldots\cdot\left(1-z_N z^{-1}\right)}{\left(1-p_1 z^{-1}\right)\left(1-p_2 z^{-1}\right)\cdot\ldots\cdot\left(1-p_M z^{-1}\right)}$$

zplane

Zero-pole plot for filter

Syntax

```
zplane(Hq)
zplane(Hq,'plotoption')
zplane(Hq,'plotoption','plotoption2')
[zq,pq,kq] = zplane(Hq)
[zq,pq,kq,zr,pr,kr] = zplane(Hq)
```

Description

This function displays the poles and zeros of quantized filters, as well as the poles and zeros of the associated unquantized reference filter.

`zplane(Hq)` plots the zeros and poles of a quantized filter Hq in the current figure window. The poles and zeros of the quantized and unquantized filters are plotted by default. The symbol `o` represents a zero of the unquantized reference filter, and the symbol `x` represents a pole of that filter. The symbols `□` and `+` are used to plot the zeros and poles of the quantized filter Hq. The plot includes the unit circle for reference.

$$|p_i| < 1 \Leftrightarrow$$ **устойчивость**

Метод инвариантности h(t)

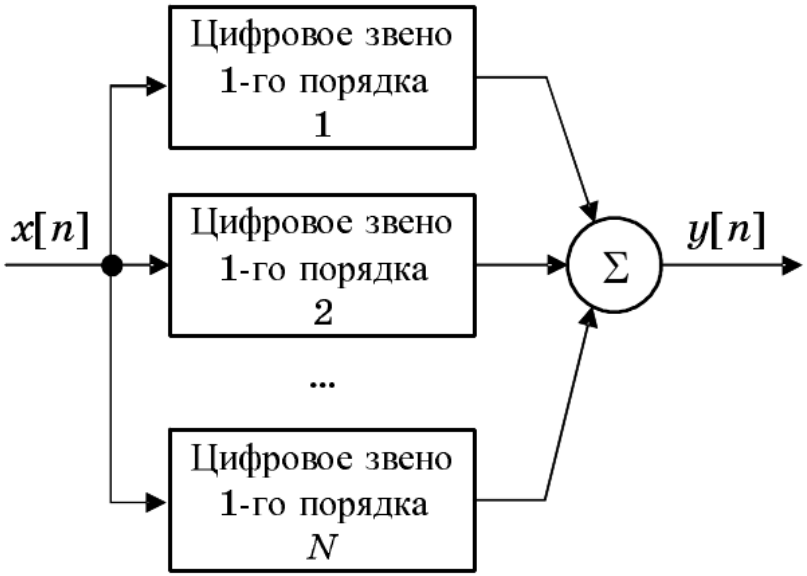
Реализовывать цифровые фильтры в MATLAB научились, вернемся к задаче синтеза цифрового фильтра по аналоговому прототипу

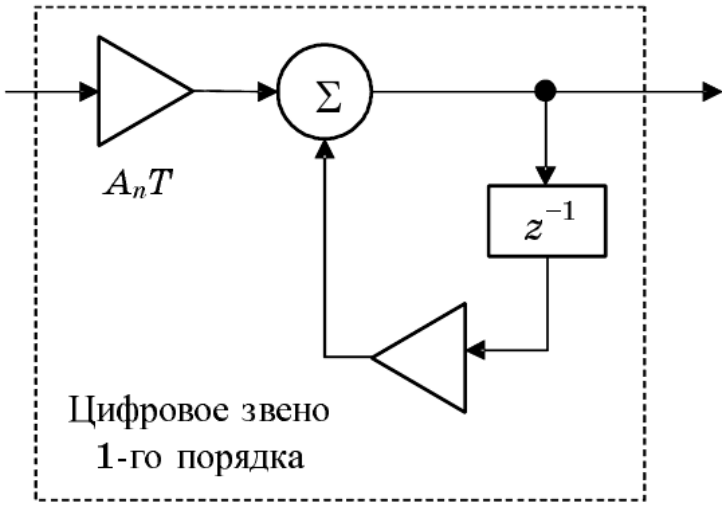
$$x_k = x\left(t_k\right) \to y_k \approx y\left(t_k\right); \quad a_j^s, b_j^s \to a_j^z, b_j^z - ???$$

Метод инвариантности импульсной характеристики: $h_k = h\left(t_k\right)$

$$h(t) = \sum_{n=1}^{N} A_n e^{p_n^s t} \to h_k = \sum_{n=1}^{N} A_n e^{p_n^s kT}, \; A_n = H(s)\left(s - z_n^s\right)\Big|_{s=z_n^s}$$

$$H(z) = \sum_{k=1}^{\infty} h_k z^{-k} = \sum_{n=1}^{N}\sum_{k=1}^{\infty} A_n e^{p_n^s kT} z^{-k} = \sum_{n=1}^{N} \frac{A_n}{1 - e^{p_n^s kT} z^{-k}}$$



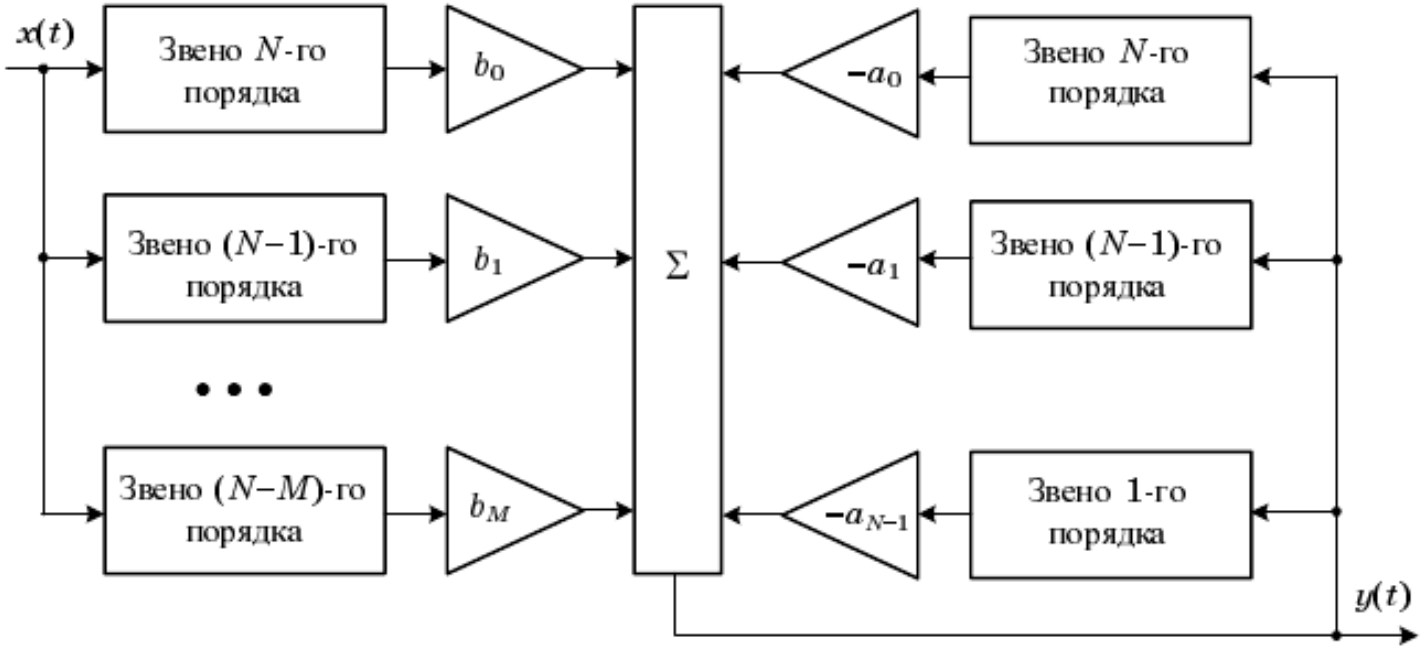


Метод билинейного преобразования

Поделим на s^M

$$H(s)=\frac{b_N s^N+\ldots+b_1 s+b_0}{a_M s^M+\ldots+a_1 s+a_0}=\frac{b_N\left(\frac{1}{s}\right)^{M-N}+\ldots+b_1\left(\frac{1}{s}\right)^{M-1}+b_0\left(\frac{1}{s}\right)^{M}}{a_M+\ldots+a_1\left(\frac{1}{s}\right)^{M-1}+a_0\left(\frac{1}{s}\right)^{M}}$$

Да это же пачки интеграторов!

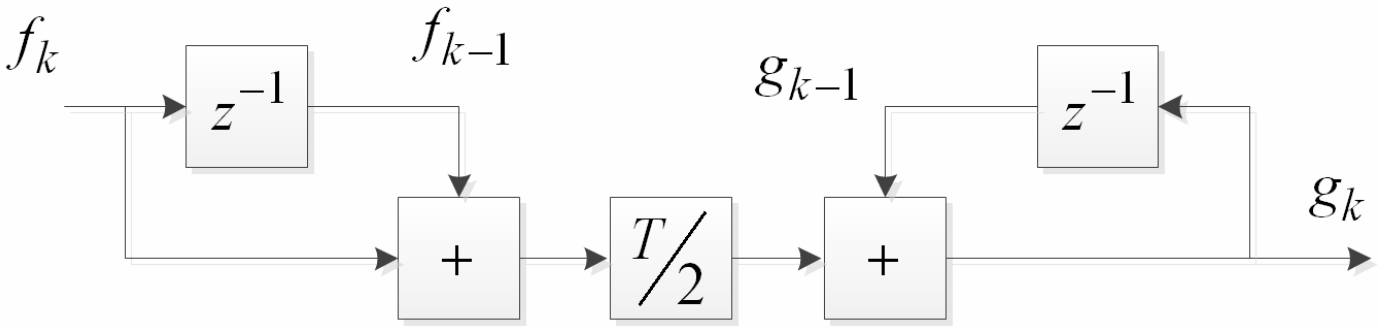


Метод билинейного преобразования

А давайте аналоговый интегратор заменим цифровым!

Интегрировать будем методом трапеций:

$$g_k = g_{k-1} + \frac{\left(f_{k-1} + f_k \right) T}{2}$$



$$H_i(z) = \left(1 + z^{-1}\right) \cdot T/2 \cdot \frac{1}{1 - z^{-1}} \Leftrightarrow \frac{1}{s}$$

Итого, в качестве s должны использовать:

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)$$

Это преобразование и называется билинейным

Метод билинейного преобразования

Полуплоскость переменной *s* отображается в окружность единичного радиуса в плоскости переменной *z*.

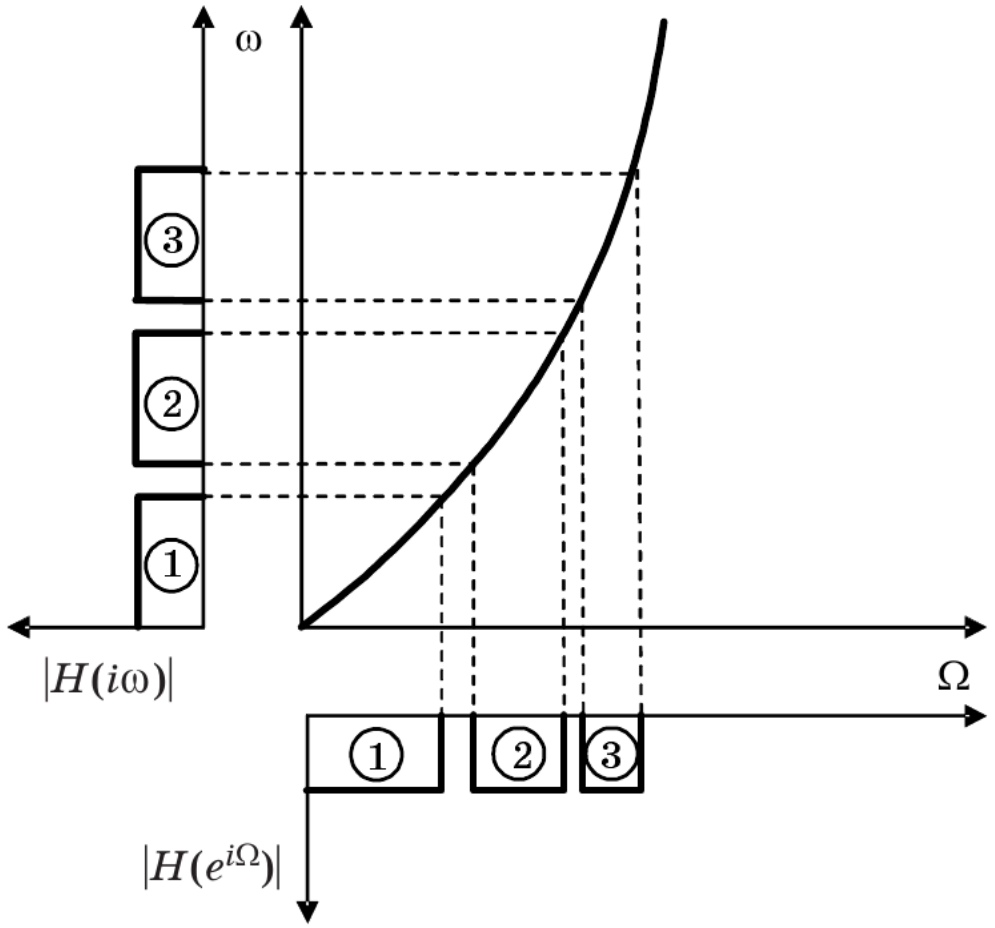
Есть явление деформации оси частот.

$$s = j\omega_s,\ z = e^{j\Omega} = e^{j\omega_z T} \rightarrow$$

$$\rightarrow \quad \omega_s = \frac{2}{T} tg \frac{\omega_z T}{2}$$

Можно заранее скомпенсировать в аналоговом прототипе

$$\omega_s' = \frac{2}{T} tg \frac{\omega_s T}{2}$$



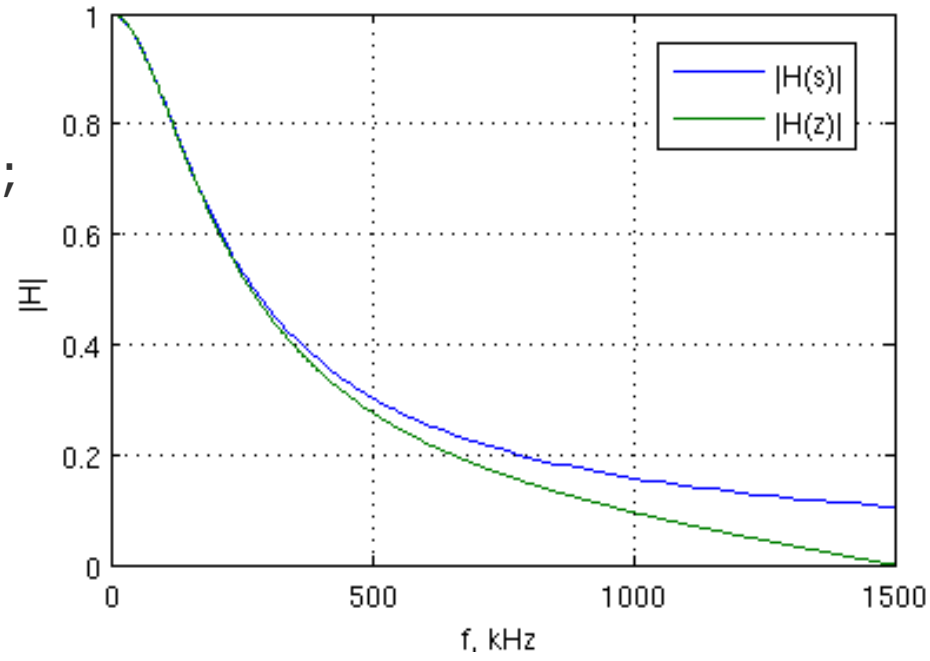
Метод билинейного преобразования

Пример. Смоделируем RC цепь, что использовали ранее.

$$H(s)=\frac{1}{RC\cdot s+1}\rightarrow H(z)=\frac{1}{RC\cdot\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}+1}=\frac{1+z^{-1}}{1+RC\cdot\frac{2}{T}+\left(1-RC\cdot\frac{2}{T}\right)z^{-1}}$$

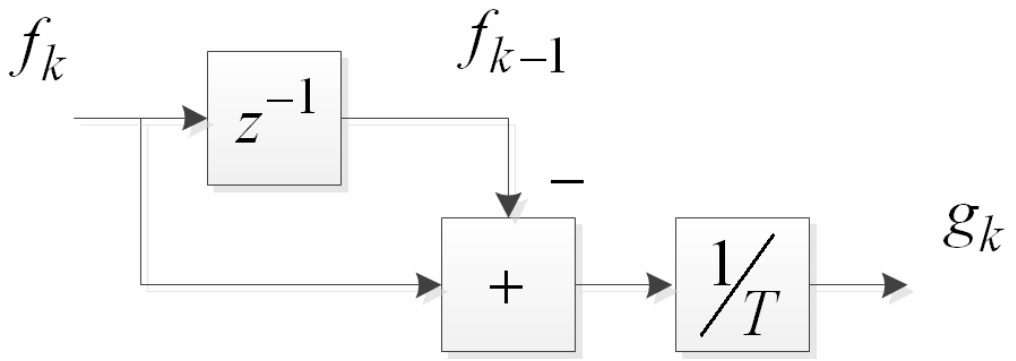
```
clear all; clc; close all;

RC = 1e-6;
T = RC/3;

as = [RC 1]; bs = [1];
[Hs, w] = freqs(bs, as);

az = [1+2*RC/T, 1-2*RC/T]; bz = [1, 1];
Hz = freqz(bz, az);

figure(1)
plot(w/2/pi/1000, abs(Hs), ...
    ((1:length(Hz)) - 1)/ length(Hz) *
1/T/2/1000, abs(Hz))
xlabel('f, kHz'); ylabel('|H|')
legend('|H(s)|', '|H(z)|'); grid on
```

$$b_0^z=1,\ b_1^z=1,\ a_0^z=1+\frac{2RC}{T},\ a_1^z=1-\frac{2RC}{T}$$



Метод замены дифференциалов

Если 1/s – интегратор, то
s – дифференциатор!



$$H_i(z) = \left(1 - z^{-1}\right) \cdot 1/T \Leftrightarrow s$$

Итого, в качестве s должны использовать: $s = \dfrac{1 - z^{-1}}{T}$

Не стоит использовать когда:
- **нули передаточной функции аналогового прототипа имеют вещественную часть больше удвоенной частоты дискретизации**
- **нулей у передаточной функции аналогового прототипа нет**

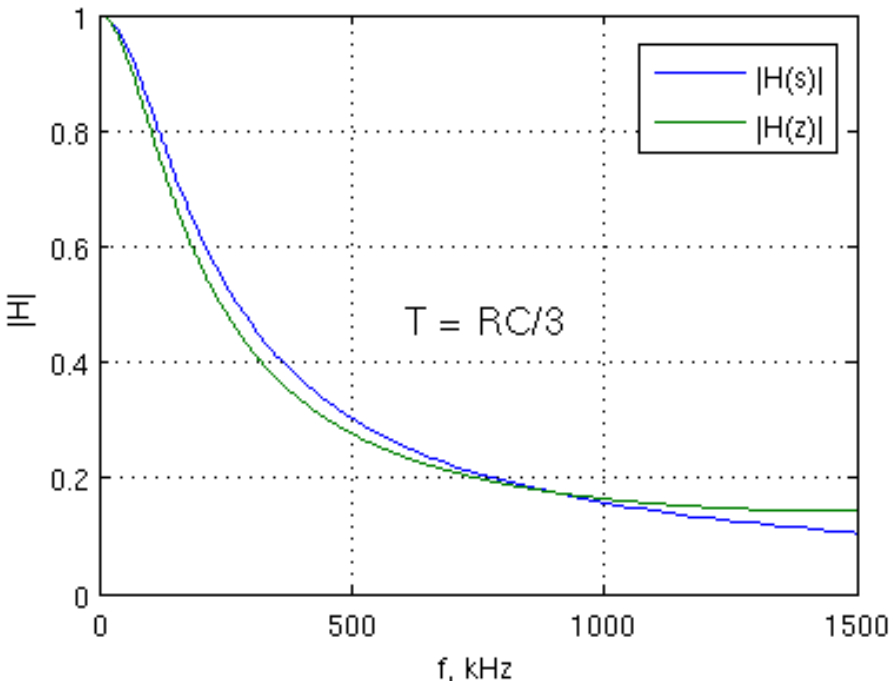
Метод замены дифференциалов

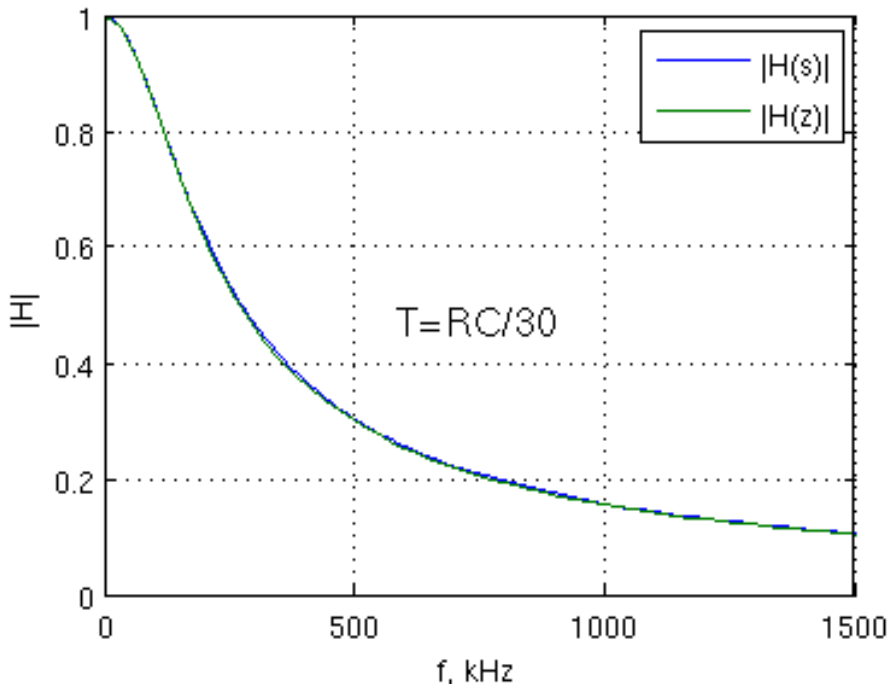
Пример. Смоделируем RC цепь.

$$H(s)=\frac{1}{RC\cdot s+1}\rightarrow H(z)=\frac{1}{RC\cdot\frac{1-z^{-1}}{T}+1}=\frac{1}{1+\frac{RC}{T}-\frac{RC}{T}z^{-1}}$$

```
...
az = [1+RC/T, -RC/T]; bz = [1];
Hz = freqz(bz, az);
…
```

$$b_0^z=1,\ a_0^z=1+\frac{RC}{T},\ a_1^z=-\frac{RC}{T}$$





Ограничение ИХ

При использовании перечисленных методов мы можем ограничивать длительность импульсной характеристики.
Получаем при этом КИХ-фильтр со всеми его плюсами. НО.

Но это же применение прямоугольного окна!

$$y_k = \sum_{n=-\infty}^{+\infty} x_n h_{k-n} w_{k-n}$$

$$\Leftrightarrow Y(z) = X(z)\left(H(z) * W(z)\right)$$

А значит и коэффициент передачи свернется со спектром окна!

Получим эффект Гиббса.

Выбором окна можем управлять расползанием спектра и неравномерностью.