

APPUNTI SUI FILTRI

Signal Processing Toolbox di Matlab

Filtri di Bessel

Analog Bessel lowpass filters have maximally flat group delay at zero frequency and retain nearly constant group delay across the entire passband. Filtered signals therefore maintain their waveshapes in the passband frequency range. Frequency mapped and digital Bessel filters, however, do not have this maximally flat property; this toolbox supports only the analog case for the complete Bessel filter design function. Bessel filters generally require a higher filter order than other filters for satisfactory stopband attenuation. $|H(j1)| < 1/\sqrt{2}$ and decreases as filter order n increases.

BESSELAP Bessel analog lowpass filter prototype.

$[Z,P,K] = \text{BESSELAP}(N)$ returns the zeros, poles, and gain for an N -th order normalized prototype Bessel analog lowpass filter. The cutoff or 3dB frequency is equal to 1 for $N = 1$ and decreases as N increases.

BESSELF Bessel analog filter design.

$[B,A] = \text{BESSELF}(N,W_n)$ designs an N 'th order lowpass analog Bessel filter and returns the filter coefficients in length $N+1$ vectors B and A . The cut-off frequency W_n must be greater than 0.

If W_n is a two-element vector, $W_n = [W_1 W_2]$, **BESSELF** returns an order $2N$ bandpass filter with passband $W_1 < W < W_2$.

$[B,A] = \text{BESSELF}(N,W_n,\text{'high'})$ designs a highpass filter.

$[B,A] = \text{BESSELF}(N,W_n,\text{'stop'})$ is a bandstop filter if $W_n = [W_1 W_2]$.

When used with three left-hand arguments, as in

$[Z,P,K] = \text{BESSELF}(\dots)$, the zeros and poles are returned in length N column vectors Z and P , and the gain in scalar K .

When used with four left-hand arguments, as in

$[A,B,C,D] = \text{BESSELF}(\dots)$, state-space matrices are returned.

Filtri di Butterworth

BUTTORD Butterworth filter order selection.

$[N, W_n] = \text{BUTTORD}(W_p, W_s, R_p, R_s)$ returns the order N of the lowest order digital Butterworth filter that loses no more than R_p dB in the passband and has at least R_s dB of attenuation in the stopband.

W_p and W_s are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to π radians/sample). For example,

Lowpass: $W_p = .1, W_s = .2$

Highpass: $W_p = .2, \quad W_s = .1$
Bandpass: $W_p = [.2 \ .7], \quad W_s = [.1 \ .8]$
Bandstop: $W_p = [.1 \ .8], \quad W_s = [.2 \ .7]$

BUTTORD also returns W_n , the Butterworth natural frequency (or, the "3 dB frequency") to use with BUTTER to achieve the specifications.

$[N, W_n] = \text{BUTTORD}(W_p, W_s, R_p, R_s, 's')$ does the computation for an analog filter, in which case W_p and W_s are in radians/second.

When R_p is chosen as 3 dB, the W_n in BUTTER is equal to W_p in BUTTORD.

BUTTAP Butterworth analog lowpass filter prototype.

$[Z,P,K] = \text{BUTTAP}(N)$ returns the zeros, poles, and gain for an N -th order normalized prototype Butterworth analog lowpass filter. The resulting filter has N poles around the unit circle in the left half plane, and no zeros.

BUTTER Butterworth digital and analog filter design.

$[B,A] = \text{BUTTER}(N,W_n)$ designs an N th order lowpass digital Butterworth filter and returns the filter coefficients in length $N+1$ vectors B (numerator) and A (denominator). The coefficients are listed in descending powers of z . The cutoff frequency W_n must be $0.0 < W_n < 1.0$, with 1.0 corresponding to half the sample rate.

If W_n is a two-element vector, $W_n = [W_1 \ W_2]$, BUTTER returns an order $2N$ bandpass filter with passband $W_1 < W < W_2$.

$[B,A] = \text{BUTTER}(N,W_n,'high')$ designs a highpass filter.

$[B,A] = \text{BUTTER}(N,W_n,'stop')$ is a bandstop filter if $W_n = [W_1 \ W_2]$.

When used with three left-hand arguments, as in

$[Z,P,K] = \text{BUTTER}(\dots)$, the zeros and poles are returned in length N column vectors Z and P , and the gain in scalar K .

When used with four left-hand arguments, as in

$[A,B,C,D] = \text{BUTTER}(\dots)$, state-space matrices are returned.

$\text{BUTTER}(N,W_n,'s')$, $\text{BUTTER}(N,W_n,'high','s')$ and $\text{BUTTER}(N,W_n,'stop','s')$ design analog Butterworth filters. In this case, W_n is in [rad/s] and it can be greater than 1.0.

Filtri di Chebyshev type 1

CHEB1ORD Chebyshev Type I filter order selection.

$[N, W_n] = \text{CHEB1ORD}(W_p, W_s, R_p, R_s)$ returns the order N of the lowest order digital Chebyshev Type I filter that loses no more than R_p dB in the passband and has at least R_s dB of attenuation in the stopband.

W_p and W_s are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to π radians/sample). For example,

Lowpass: $W_p = .1, \quad W_s = .2$
Highpass: $W_p = .2, \quad W_s = .1$

Bandpass: $W_p = [.2 \ .7]$, $W_s = [.1 \ .8]$

Bandstop: $W_p = [.1 \ .8]$, $W_s = [.2 \ .7]$

CHEB1ORD also returns W_n , the Chebyshev natural frequency to use with CHEBY1 to achieve the specifications.

$[N, W_n] = \text{CHEB1ORD}(W_p, W_s, R_p, R_s, 's')$ does the computation for an analog filter, in which case W_p and W_s are in radians/second.

CHEB1AP Chebyshev Type I analog lowpass filter prototype.

$[Z,P,K] = \text{CHEB1AP}(N,R_p)$ returns the zeros, poles, and gain of an N-th order normalized analog prototype Chebyshev Type I lowpass filter with R_p decibels of ripple in the passband.

Chebyshev Type I filters are maximally flat in the stopband.

CHEBY1 Chebyshev Type I digital and analog filter design.

$[B,A] = \text{CHEBY1}(N,R,W_n)$ designs an Nth order lowpass digital Chebyshev filter with R decibels of peak-to-peak ripple in the passband. CHEBY1 returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The cutoff frequency W_n must be $0.0 < W_n < 1.0$, with 1.0 corresponding to half the sample rate. Use $R=0.5$ as a starting point, if you are unsure about choosing R.

If W_n is a two-element vector, $W_n = [W_1 \ W_2]$, CHEBY1 returns an order $2N$ bandpass filter with passband $W_1 < W < W_2$.

$[B,A] = \text{CHEBY1}(N,R,W_n,'high')$ designs a highpass filter.

$[B,A] = \text{CHEBY1}(N,R,W_n,'stop')$ is a bandstop filter if $W_n = [W_1 \ W_2]$.

When used with three left-hand arguments, as in

$[Z,P,K] = \text{CHEBY1}(\dots)$, the zeros and poles are returned in length N column vectors Z and P, and the gain in scalar K.

When used with four left-hand arguments, as in

$[A,B,C,D] = \text{CHEBY1}(\dots)$, state-space matrices are returned.

$\text{CHEBY1}(N,R,W_n,'s')$, $\text{CHEBY1}(N,R,W_n,'high','s')$ and

$\text{CHEBY1}(N,R,W_n,'stop','s')$ design analog Chebyshev Type I filters.

In this case, W_n is in [rad/s] and it can be greater than 1.0.

Filtri di Chebyshev type 2

CHEB2ORD Chebyshev Type II filter order selection.

$[N, W_n] = \text{CHEB2ORD}(W_p, W_s, R_p, R_s)$ returns the order N of the lowest order digital Chebyshev Type II filter that loses no more than R_p dB in the passband and has at least R_s dB of attenuation in the stopband.

W_p and W_s are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to π radians/sample). For example,

Lowpass: $W_p = .1$, $W_s = .2$

Highpass: $W_p = .2$, $W_s = .1$

Bandpass: $W_p = [.2 \ .7]$, $W_s = [.1 \ .8]$

Bandstop: $W_p = [.1 \ .8]$, $W_s = [.2 \ .7]$

CHEB2ORD also returns W_n , the Chebyshev natural frequency to use with CHEBY2 to achieve the specifications.

$[N, W_n] = \text{CHEB2ORD}(W_p, W_s, R_p, R_s, 's')$ does the computation for an analog filter, in which case W_p and W_s are in radians/second.

CHEB2AP Chebyshev Type II analog lowpass filter prototype.

$[Z,P,K] = \text{CHEB2AP}(N,R_s)$ returns the zeros, poles, and gain of an N -th order normalized analog prototype Chebyshev Type II lowpass filter with R_s decibels of ripple in the stopband.

Chebyshev Type II filters are maximally flat in the passband.

CHEBY2 Chebyshev Type II digital and analog filter design.

$[B,A] = \text{CHEBY2}(N,R,W_n)$ designs an N th order lowpass digital Chebyshev filter with the stopband ripple R decibels down and stopband edge frequency W_n . CHEBY2 returns the filter coefficients in length $N+1$ vectors B (numerator) and A (denominator).

The cutoff frequency W_n must be $0.0 < W_n < 1.0$, with 1.0 corresponding to half the sample rate. Use $R = 20$ as a starting point, if you are unsure about choosing R .

If W_n is a two-element vector, $W_n = [W_1 W_2]$, CHEBY2 returns an order $2N$ bandpass filter with passband $W_1 < W < W_2$.

$[B,A] = \text{CHEBY2}(N,R,W_n, 'high')$ designs a highpass filter.

$[B,A] = \text{CHEBY2}(N,R,W_n, 'stop')$ is a bandstop filter if $W_n = [W_1 W_2]$.

When used with three left-hand arguments, as in

$[Z,P,K] = \text{CHEBY2}(\dots)$, the zeros and poles are returned in length N column vectors Z and P , and the gain in scalar K .

When used with four left-hand arguments, as in

$[A,B,C,D] = \text{CHEBY2}(\dots)$, state-space matrices are returned.

$\text{CHEBY2}(N,R,W_n, 's')$, $\text{CHEBY2}(N,R,W_n, 'high', 's')$ and $\text{CHEBY2}(N,R,W_n, 'stop', 's')$ design analog Chebyshev Type II filters.

In this case, W_n is in [rad/s] and it can be greater than 1.0.

Filtri Ellittici

ELLIPORD Elliptic filter order selection.

$[N, W_n] = \text{ELLIPORD}(W_p, W_s, R_p, R_s)$ returns the order N of the lowest order digital elliptic filter that loses no more than R_p dB in the passband and has at least R_s dB of attenuation in the stopband.

W_p and W_s are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to π radians/sample). For example,

Lowpass: $W_p = .1, W_s = .2$

Highpass: $W_p = .2, W_s = .1$

Bandpass: $W_p = [.2 .7], W_s = [.1 .8]$

Bandstop: $W_p = [.1 .8], W_s = [.2 .7]$

ELLIPORD also returns W_n , the elliptic natural frequency to use with ELLIP to achieve the specifications.

`[N, Wn] = ELLIPORD(Wp, Ws, Rp, Rs, 's')` does the computation for an analog filter, in which case `Wp` and `Ws` are in radians/second.

NOTE: If `Rs` is much much greater than `Rp`, or `Wp` and `Ws` are very close, the estimated order can be infinite due to limitations of numerical precision.

ELLIPAP Elliptic analog lowpass filter prototype.

`[Z,P,K] = ELLIPAP(N,Rp,Rs)` returns the zeros, poles, and gain of an `N`-th order normalized prototype elliptic analog lowpass filter with `Rp` decibels of ripple in the passband and a stopband `Rs` decibels down.

ELLIP Elliptic or Causer digital and analog filter design.

`[B,A] = ELLIP(N,Rp,Rs,Wn)` designs an `N`th order lowpass digital elliptic filter with `Rp` decibels of peak-to-peak ripple and a minimum stopband attenuation of `Rs` decibels. `ELLIP` returns the filter coefficients in length `N+1` vectors `B` (numerator) and `A` (denominator). The cutoff frequency `Wn` must be $0.0 < Wn < 1.0$, with 1.0 corresponding to half the sample rate. Use `Rp = 0.5` and `Rs = 20` as starting points, if you are unsure about choosing them.

If `Wn` is a two-element vector, `Wn = [W1 W2]`, `ELLIP` returns an order `2N` bandpass filter with passband $W1 < W < W2$.

`[B,A] = ELLIP(N,Rp,Rs,Wn,'high')` designs a highpass filter.

`[B,A] = ELLIP(N,Rp,Rs,Wn,'stop')` is a bandstop filter if `Wn = [W1 W2]`.

When used with three left-hand arguments, as in `[Z,P,K] = ELLIP(...)`, the zeros and poles are returned in length `N` column vectors `Z` and `P`, and the gain in scalar `K`.

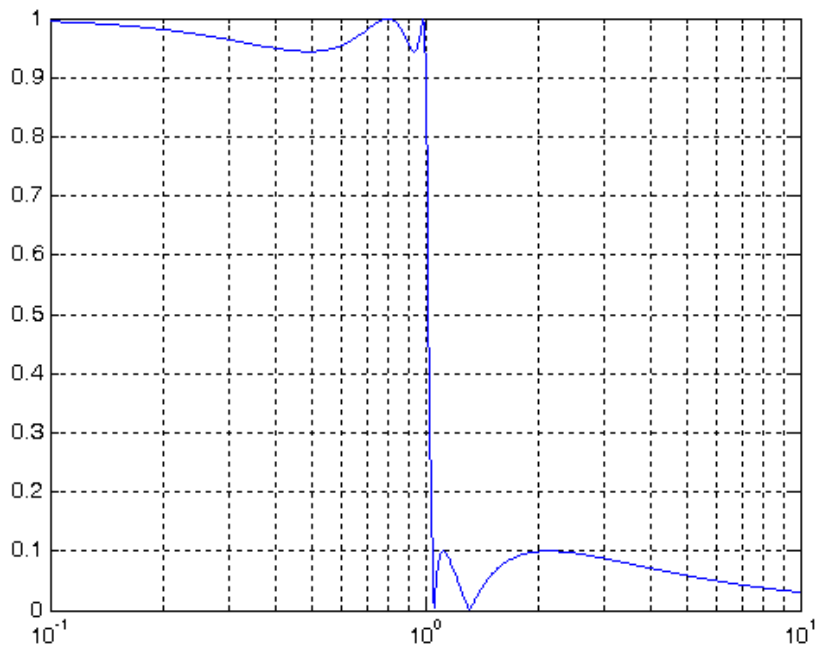
When used with four left-hand arguments, as in `[A,B,C,D] = ELLIP(...)`, state-space matrices are returned.

`ELLIP(N,Rp,Rs,Wn,'s')`, `ELLIP(N,Rp,Rs,Wn,'high','s')` and `ELLIP(N,Rp,Rs,Wn,'stop','s')` design analog elliptic filters. In this case, `Wn` is in [rad/s] and it can be greater than 1.0.

ESEMPIO 1

Vediamo come disegnare il diagramma del modulo di un filtro ellittico di grado 5, con massima attenuazione in banda passante di 0.5 dB e minima attenuazione in banda attenuata di 20 dB:

```
>> [z, p, k] = ellipap(5, 0.5, 20);
>> w = logspace(-1,1,1000);
>> h = freqs(k*poly(z), poly(p), w);
>> att = -20 .* log10(abs(h));
>> semilogx(w, abs(h)), grid; % diagramma del modulo
>> figure;
>> semilogx(w, att), grid; % diagramma di Bode dell'attenuazione
```



dove le routines usate sono:

POLY Convert roots to polynomial.

POLY(A), when A is an N by N matrix, is a row vector with N+1 elements which are the coefficients of the characteristic polynomial, $\text{DET}(\text{lambda}*\text{EYE}(\text{SIZE}(\text{A})) - \text{A})$.

POLY(V), when V is a vector, is a vector whose elements are the coefficients of the polynomial whose roots are the elements of V. For vectors, **ROOTS** and **POLY** are inverse functions of each other, up to ordering, scaling, and roundoff error.

ROOTS(POLY(1:20)) generates Wilkinson's famous example.

FREQS Laplace-transform (s-domain) frequency response.

H = FREQS(B,A,W) returns the complex frequency response vector H of the filter B/A:

$$H(s) = \frac{B(s)}{A(s)} = \frac{b(1)s^{nb-1} + b(2)s^{nb-2} + \dots + b(nb)}{a(1)s^{na-1} + a(2)s^{na-2} + \dots + a(na)}$$

given the numerator and denominator coefficients in vectors B and A.

The frequency response is evaluated at the points specified in vector W (in rad/s). The magnitude and phase can be graphed by calling **FREQS(B,A,W)** with no output arguments.

[H,W] = FREQS(B,A) automatically picks a set of 200 frequencies W on

which the frequency response is computed. `FREQS(B,A,N)` picks N frequencies.

ROOTS Find polynomial roots.

`ROOTS(C)` computes the roots of the polynomial whose coefficients are the elements of the vector C . If C has $N+1$ components, the polynomial is $C(1)*X^N + \dots + C(N)*X + C(N+1)$.

POLYVAL Evaluate polynomial.

$Y = \text{POLYVAL}(P,X)$, when P is a vector of length $N+1$ whose elements are the coefficients of a polynomial, is the value of the polynomial evaluated at X .

$$Y = P(1)*X^N + P(2)*X^{(N-1)} + \dots + P(N)*X + P(N+1)$$

If X is a matrix or vector, the polynomial is evaluated at all points in X . See also `POLYVALM` for evaluation in a matrix sense.

$Y = \text{POLYVAL}(P,X,[],\text{MU})$ uses $\text{XHAT} = (X - \text{MU}(1))/\text{MU}(2)$ in place of X . The centering and scaling parameters MU are optional output computed by `POLYFIT`.

$[Y,\text{DELTA}] = \text{POLYVAL}(P,X,S)$ or $[Y,\text{DELTA}] = \text{POLYVAL}(P,X,S,\text{MU})$ uses the optional output structure S provided by `POLYFIT` to generate error estimates, $Y \pm \text{delta}$. If the errors in the data input to `POLYFIT` are independent normal with constant variance, $Y \pm \text{DELTA}$ contains at least 50% of the predictions.

Esempio 2

Vogliamo progettare un filtro passa-basso con $\omega_p = 10$ kHz, $\omega_A = 15$ kHz, $\alpha_p = 0.1$ dB e $\alpha_A = 25$ dB. Proviamo con un filtro di Chebyshev del primo tipo con $n = 5$ e $R_s = 0.1$ dB.

```
>> [b, a] = cheby1(5, 0.1, 10000, 's');
>> w = logspace(1, 5, 5000);
>> h = freqs(b, a, w);
>> semilogx(w, abs(h)), grid;
>> att = -20 .* log10(abs(h));
>> figure;
>> semilogx(w, att), grid;
```

Vogliamo ora conoscere il valore dell'attenuazione a $\omega_A = 15000$ rad/s, limite inferiore della banda attenuata:

```
>> h1 = polyval(b, 15000*j)/polyval(a, 15000*j);
>> att1 = -20 * log10(abs(h1));
```

Si trova: $\text{att1} = 19.4988$ dB. Questo è il massimo valore di attenuazione che si può ottenere con questo filtro. Se le specifiche richiedono una attenuazione superiore, per esempio di 25 dB, allora si deve cambiare o l'ordine del filtro, o tipo di filtro:

1) Aumentiamo l'ordine del filtro a 6. Ripetendo le operazioni fatte sopra con $n = 6$, otteniamo $att1 = 27.8160$ dB. Soddisfiamo quindi le specifiche del filtro.

2) Passiamo ad un filtro ellittico con $n = 5$:

```
>> [be,ae] = ellip(5, 0.1, 25, 10000, 's');  
>> w = logspace(1, 5, 5000);  
>> he = freqs(be, ae, w);  
>> atte = -20 .* log10(abs(he));  
>> semilogx(w, atte), grid;
```

Ora dobbiamo trovare l'attenuazione per $\omega_A = 15000$ rad/s.

```
>> he = polyval(be, 15000*j)/polyval(ae, 15000*j);  
>> atte = -20 * log10(abs(he));
```

Risulta $atte = 33.1137$ dB. Si potrebbe controllare se un filtro ellittico di ordine $n = 4$ soddisfa ugualmente le specifiche.

Utilizziamo ora le funzioni per determinare l'ordine minimo di un filtro:

```
>> [n, Wn] = cheblord(10000, 15000, 0.1, 25, 's');
```

Si ottiene $n = 6$ e $Wn = 10000$, come avevamo già sperimentato prima.

Se invece consideriamo i filtri ellittici:

```
>> [n, Wn] = ellipord(10000, 15000, 0.1, 25, 's');
```

Si ottiene $n = 4$ e $Wn = 10000$. E' evidente che un filtro di grado 4 è sufficiente, come avevamo sospettato prima.