(index.html)

Signal processing (scipy.signal)

Convolution

convolve (generated/scipy.signal.convolve.html#scipy.signal.convolve)(in1, in2[, mode])	Convolve two N-dimensional arrays.
correlate (generated/scipy.signal.correlate.html#scipy.signal.correlate)(in1, in2[, mode])	Cross-correlate two N-dimensional arrays.
fftconvolve (generated/scipy.signal.fftconvolve.html#scipy.signal.fftconvolve)(in1, in2[, mode])	Convolve two N-dimensional arrays using FFT.
convolve2d (generated/scipy.signal.convolve2d.html#scipy.signal.convolve2d)(in1, in2[, mode, boundary, fillvalue])	Convolve two 2-dimensional arrays.
correlate2d (generated/scipy.signal.correlate2d.html#scipy.signal.correlate2d)(in1, in2[, mode, boundary, ...])	Cross-correlate two 2-dimensional arrays.
sepfir2d (generated/scipy.signal.sepfir2d.html#scipy.signal.sepfir2d)((input, hrow, hcol) -> output)	Description:

B-splines

bspline (generated/scipy.signal.bspline.html#scipy.signal.bspline)(x, n)	B-spline basis function of order n.
cubic (generated/scipy.signal.cubic.html#scipy.signal.cubic)(x)	A cubic B-spline.
quadratic (generated/scipy.signal.quadratic.html#scipy.signal.quadratic)(x)	A quadratic B-spline.
gauss_spline (generated/scipy.signal.gauss_spline.html#scipy.signal.gauss_spline)(x, n)	Gaussian approximation to B-spline basis function of order n.
cspline1d (generated/scipy.signal.cspline1d.html#scipy.signal.cspline1d)(signal[, lamb])	Compute cubic spline coefficients for rank-1 array.
qspline1d (generated/scipy.signal.qspline1d.html#scipy.signal.qspline1d)(signal[, lamb])	Compute quadratic spline coefficients for rank-1 array.
cspline2d (generated/scipy.signal.cspline2d.html#scipy.signal.cspline2d)((input {, lambda, precision}) -> ck)	Description:
qspline2d (generated/scipy.signal.qspline2d.html#scipy.signal.qspline2d)((input {, lambda, precision}) -> qk)	Description:

Table Of Contents (index.html)

Previous topic

scipy.optimize.show_options (generated/scipy.optimize.shc

Next topic

scipy.signal.convolve (generated/scipy.signal.convo

cspline1d_eval (generated/scipy.signal.cspline1d_eval.html#scipy.signal.cspline1d_eval)(cj, newx[, dx, x0])	Evaluate a spline at the new set of points.
qspline1d_eval (generated/scipy.signal.qspline1d_eval.html#scipy.signal.qspline1d_eval)(cj, newx[, dx, x0])	Evaluate a quadratic spline at the new set of points.
spline_filter (generated/scipy.signal.spline_filter.html#scipy.signal.spline_filter)(Iin[, lmbda])	Smoothing spline (cubic) filtering of a rank-2 array.

Filtering

order_filter (generated/scipy.signal.order_filter.html#scipy.signal.order_filter)(a, domain, rank)	Perform an order filter on an N-dimensional array.
medfilt (generated/scipy.signal.medfilt.html#scipy.signal.medfilt)(volume[, kernel_size])	Perform a median filter on an N-dimensional array.
medfilt2d (generated/scipy.signal.medfilt2d.html#scipy.signal.medfilt2d)(input[, kernel_size])	Median filter a 2-dimensional array.
wiener (generated/scipy.signal.wiener.html#scipy.signal.wiener)(im[, mysize, noise])	Perform a Wiener filter on an N-dimensional array.
symiirorder1 (generated/scipy.signal.symiirorder1.html#scipy.signal.symiirorder1)((input, c0, z1 {, ...)	Implement a smoothing IIR filter with mirror-symme boundary conditions using a cascade of first-order sections.
symiirorder2 (generated/scipy.signal.symiirorder2.html#scipy.signal.symiirorder2)((input, r, omega {, ...)	Implement a smoothing IIR filter with mirror-symme boundary conditions using a cascade of second-ord sections.
lfilter (generated/scipy.signal.lfilter.html#scipy.signal.lfilter)(b, a, x[, axis, zi])	Filter data along one-dimension with an IIR or FIR fil
lfiltic (generated/scipy.signal.lfiltic.html#scipy.signal.lfiltic)(b, a, y[, x])	Construct initial conditions for lfilter.
lfilter_zi (generated/scipy.signal.lfilter_zi.html#scipy.signal.lfilter_zi)(b, a)	Compute an initial state *zi* for the lfilter function tha corresponds to the steady state of the step respons
filtfilt (generated/scipy.signal.filtfilt.html#scipy.signal.filtfilt)(b, a, x[, axis, padtype, padlen])	A forward-backward filter.
savgol_filter (generated/scipy.signal.savgol_filter.html#scipy.signal.savgol_filter)(x, window_length, polyorder[, ...])	Apply a Savitzky-Golay filter to an array.
deconvolve (generated/scipy.signal.deconvolve.html#scipy.signal.deconvolve)(signal, divisor)	Deconvolves *divisor* out of signal (http://docs.python.org/dev/library/signal.html#moc signal).
hilbert (generated/scipy.signal.hilbert.html#scipy.signal.hilbert)(x[, N, axis])	Compute the analytic signal, using the Hilbert transform.
hilbert2 (generated/scipy.signal.hilbert2.html#scipy.signal.hilbert2)(x[, N])	Compute the '2-D' analytic signal of *x*
decimate (generated/scipy.signal.decimate.html#scipy.signal.decimate)(x, q[, n, ftype, axis])	Downsample the signal by using a filter.
detrend (generated/scipy.signal.detrend.html#scipy.signal.detrend)(data[, axis, type, bp])	Remove linear trend along axis from data.
resample (generated/scipy.signal.resample.html#scipy.signal.resample)(x, num[, t, axis, window])	Resample *x* to *num* samples using Fourier method a the given axis.

Filter design

bilinear (generated/scipy.signal.bilinear.html#scipy.signal.bilinear)(b, a[, fs])	Return a digital filter from an analog one using a bilinear transform.
findfreqs (generated/scipy.signal.findfreqs.html#scipy.signal.findfreqs)(num, den, N)	Find an array of frequencies for computing the response of a filter.
firwin (generated/scipy.signal.firwin.html#scipy.signal.firwin)(numtaps, cutoff[, width, window, ...])	FIR filter design using the window method.
firwin2 (generated/scipy.signal.firwin2.html#scipy.signal.firwin2)(numtaps, freq, gain[, nfreqs, ...])	FIR filter design using

	the window method.
freqs (generated/scipy.signal.freqs.html#scipy.signal.freqs)(b, a[, worN, plot])	Compute frequency response of analog filter.
freqz (generated/scipy.signal.freqz.html#scipy.signal.freqz)(b[, a, worN, whole, plot])	Compute the frequency response of a digital filter.
iirdesign (generated/scipy.signal.iirdesign.html#scipy.signal.iirdesign)(wp, ws, gpass, gstop[, analog, ...])	Complete IIR digital and analog filter design.
iirfilter (generated/scipy.signal.iirfilter.html#scipy.signal.iirfilter)(N, Wn[, rp, rs, btype, analog, ...])	IIR digital and analog filter design given order and critical points.
kaiser_atten (generated/scipy.signal.kaiser_atten.html#scipy.signal.kaiser_atten)(numtaps, width)	Compute the attenuation of a Kaiser FIR filter.
kaiser_beta (generated/scipy.signal.kaiser_beta.html#scipy.signal.kaiser_beta)(a)	Compute the Kaiser parameter *beta*, given the attenuation *a*.
kaiserord (generated/scipy.signal.kaiserord.html#scipy.signal.kaiserord)(ripple, width)	Design a Kaiser window to limit ripple and width of transition region.
savgol_coeffs (generated/scipy.signal.savgol_coeffs.html#scipy.signal.savgol_coeffs)(window_length, polyorder[, ...])	Compute the coefficients for a 1-d Savitzky-Golay FIR filter.
remez (generated/scipy.signal.remez.html#scipy.signal.remez)(numtaps, bands, desired[, weight, Hz, ...])	Calculate the minimax optimal filter using the Remez exchange algorithm.
unique_roots (generated/scipy.signal.unique_roots.html#scipy.signal.unique_roots)(p[, tol, rtype])	Determine unique roots and their multiplicities from a list of roots.
residue (generated/scipy.signal.residue.html#scipy.signal.residue)(b, a[, tol, rtype])	Compute partial-fraction expansion of b(s) / a(s).
residuez (generated/scipy.signal.residuez.html#scipy.signal.residuez)(b, a[, tol, rtype])	Compute partial-

	fraction expansion of b(z) / a(z).
invres (generated/scipy.signal.invres.html#scipy.signal.invres)(r, p, k[, tol, rtype])	Compute b(s) and a(s) from partial fraction expansion.
invresz (generated/scipy.signal.invresz.html#scipy.signal.invresz)(r, p, k[, tol, rtype])	Compute b(z) and a(z) from partial fraction expansion.

Lower-level filter design functions:

abcd_normalize (generated/scipy.signal.abcd_normalize.html#scipy.signal.abcd_normalize)([A, B, C, D])	Check state-space matrices and ensure they are two-dimensional.
band_stop_obj (generated/scipy.signal.band_stop_obj.html#scipy.signal.band_stop_obj) (wp, ind, passb, stopb, gpass, ...)	Band Stop Objective Function for order minimization.
besselap (generated/scipy.signal.besselap.html#scipy.signal.besselap)(N)	Return (z,p,k) for analog prototype of an Nth order Bessel filter.
buttap (generated/scipy.signal.buttap.html#scipy.signal.buttap)(N)	Return (z,p,k) for analog prototype of Nth order Butterworth filter.
cheb1ap (generated/scipy.signal.cheb1ap.html#scipy.signal.cheb1ap)(N, rp)	Return (z,p,k) for Nth order Chebyshev type I analog lowpass filter.
cheb2ap (generated/scipy.signal.cheb2ap.html#scipy.signal.cheb2ap)(N, rs)	Return (z,p,k) for Nth order Chebyshev type I analog lowpass filter.
cmplx_sort (generated/scipy.signal.cmplx_sort.html#scipy.signal.cmplx_sort)(p)	Sort roots based on magnitude.
ellipap (generated/scipy.signal.ellipap.html#scipy.signal.ellipap)(N, rp, rs)	Return (z,p,k) of Nth order elliptic analog lowpass filter.
lp2bp (generated/scipy.signal.lp2bp.html#scipy.signal.lp2bp)(b, a[, wo, bw])	Transform a lowpass filter prototype to a bandpass filter.
lp2bs (generated/scipy.signal.lp2bs.html#scipy.signal.lp2bs)(b, a[, wo, bw])	Transform a lowpass filter prototype to a bandstop filter.
lp2hp (generated/scipy.signal.lp2hp.html#scipy.signal.lp2hp)(b, a[, wo])	Transform a lowpass filter prototype to a highpass filter.

lp2lp (generated/scipy.signal.lp2lp.html#scipy.signal.lp2lp)(b, a[, wo])	Transform a lowpass filter prototype to a different frequency.
normalize (generated/scipy.signal.normalize.html#scipy.signal.normalize)(b, a)	Normalize polynomial representation of a transfer function.

Matlab-style IIR filter design

butter (generated/scipy.signal.butter.html#scipy.signal.butter)(N, Wn[, btype, analog, output])	Butterworth digital and analog filter design.
buttord (generated/scipy.signal.buttord.html#scipy.signal.buttord)(wp, ws, gpass, gstop[, analog])	Butterworth filter order selection.
cheby1 (generated/scipy.signal.cheby1.html#scipy.signal.cheby1)(N, rp, Wn[, btype, analog, output])	Chebyshev type I digital and analog filter design.
cheb1ord (generated/scipy.signal.cheb1ord.html#scipy.signal.cheb1ord)(wp, ws, gpass, gstop[, analog])	Chebyshev type I filter order selection.
cheby2 (generated/scipy.signal.cheby2.html#scipy.signal.cheby2)(N, rs, Wn[, btype, analog, output])	Chebyshev type II digital and analog filter design.
cheb2ord (generated/scipy.signal.cheb2ord.html#scipy.signal.cheb2ord)(wp, ws, gpass, gstop[, analog])	Chebyshev type II filter order selection.
ellip (generated/scipy.signal.ellip.html#scipy.signal.ellip)(N, rp, rs, Wn[, btype, analog, output])	Elliptic (Cauer) digital and analog filter design.
ellipord (generated/scipy.signal.ellipord.html#scipy.signal.ellipord)(wp, ws, gpass, gstop[, analog])	Elliptic (Cauer) filter order selection.
bessel (generated/scipy.signal.bessel.html#scipy.signal.bessel)(N, Wn[, btype, analog, output])	Bessel/Thomson digital and analog filter design.

Continuous-Time Linear Systems

freqresp (generated/scipy.signal.freqresp.html#scipy.signal.freqresp)(system[, w, n])	Calculate the frequency response of a continuous-time system.
lti (generated/scipy.signal.lti.html#scipy.signal.lti)(*args, **kwords)	Linear Time Invariant class which simplifies representation.
lsim (generated/scipy.signal.lsim.html#scipy.signal.lsim)(system, U, T[, X0, interp])	Simulate output of a continuous-time linear system.
lsim2 (generated/scipy.signal.lsim2.html#scipy.signal.lsim2)(system[, U, T, X0])	Simulate output of a continuous-time linear system, by using the ODE solver scipy.integrate.odeint (generated/scipy.integrate.odeint.html#scipy.integrate.odeint).
impulse (generated/scipy.signal.impulse.html#scipy.signal.impulse)(system[, X0, T, N])	Impulse response of continuous-time system.
impulse2 (generated/scipy.signal.impulse2.html#scipy.signal.impulse2)(system[, X0, T, N])	Impulse response of a single-input, continuous-time linear system.
step (generated/scipy.signal.step.html#scipy.signal.step)(system[, X0, T, N])	Step response of continuous-time system.
step2 (generated/scipy.signal.step2.html#scipy.signal.step2)(system[, X0, T, N])	Step response of continuous-time system.
bode (generated/scipy.signal.bode.html#scipy.signal.bode)(system[, w, n])	Calculate Bode magnitude and phase data of a continuous-time system.

Discrete-Time Linear Systems

dlsim (generated/scipy.signal.dlsim.html#scipy.signal.dlsim)(system, u[, t, x0])	Simulate output

	of a discrete-time linear system.
dimpulse (generated/scipy.signal.dimpulse.html#scipy.signal.dimpulse)(system[, x0, t, n])	Impulse response of discrete-time system.
dstep (generated/scipy.signal.dstep.html#scipy.signal.dstep)(system[, x0, t, n])	Step response of discrete-time system.

LTI Representations

tf2zpk (generated/scipy.signal.tf2zpk.html#scipy.signal.tf2zpk)(b, a)	Return zero, pole, gain (z,p,k) representation from a numerator, denominator representation of a linear filter.
zpk2tf (generated/scipy.signal.zpk2tf.html#scipy.signal.zpk2tf)(z, p, k)	Return polynomial transfer function representation from zeros
tf2ss (generated/scipy.signal.tf2ss.html#scipy.signal.tf2ss)(num, den)	Transfer function to state-space representation.
ss2tf (generated/scipy.signal.ss2tf.html#scipy.signal.ss2tf)(A, B, C, D[, input])	State-space to transfer function.
zpk2ss (generated/scipy.signal.zpk2ss.html#scipy.signal.zpk2ss)(z, p, k)	Zero-pole-gain representation to state-space representation
ss2zpk (generated/scipy.signal.ss2zpk.html#scipy.signal.ss2zpk)(A, B, C, D[, input])	State-space representation to zero-pole-gain representation.
cont2discrete (generated/scipy.signal.cont2discrete.html#scipy.signal.cont2discrete)(sys, dt[, method, alpha])	Transform a continuous to a discrete state-space system.

Waveforms

chirp (generated/scipy.signal.chirp.html#scipy.signal.chirp)(t, f0, t1, f1[, method, phi, vertex_zero])	Frequency-swept cosine generator.
gausspulse (generated/scipy.signal.gausspulse.html#scipy.signal.gausspulse)(t[, fc, bw, bwr, tpr, retquad, ...])	Return a Gaussian modulated sinusoid:
max_len_seq (generated/scipy.signal.max_len_seq.html#scipy.signal.max_len_seq)(nbits[, state, length, taps])	Maximum Length Sequence (MLS) generator
sawtooth (generated/scipy.signal.sawtooth.html#scipy.signal.sawtooth)(t[, width])	Return a periodic sawtooth

	or triangle waveform.
square (generated/scipy.signal.square.html#scipy.signal.square)(t[, duty])	Return a periodic square-wave waveform.
sweep_poly (generated/scipy.signal.sweep_poly.html#scipy.signal.sweep_poly)(t, poly[, phi])	Frequency-swept cosine generator, with a time-dependent frequency.

Window functions

get_window (generated/scipy.signal.get_window.html#scipy.signal.get_window) (window, Nx[, fftbins])	Return a window.
barthann (generated/scipy.signal.barthann.html#scipy.signal.barthann)(M[, sym])	Return a modified Bartlett-Hann window.
bartlett (generated/scipy.signal.bartlett.html#scipy.signal.bartlett)(M[, sym])	Return a Bartlett window.
blackman (generated/scipy.signal.blackman.html#scipy.signal.blackman)(M[, sym])	Return a Blackman window.
blackmanharris (generated/scipy.signal.blackmanharris.html#scipy.signal.blackmanharris) (M[, sym])	Return a minimum 4-term Blackman-Harris window.
bohman (generated/scipy.signal.bohman.html#scipy.signal.bohman)(M[, sym])	Return a Bohman window.
boxcar (generated/scipy.signal.boxcar.html#scipy.signal.boxcar)(M[, sym])	Return a boxcar or rectangular window.
chebwin (generated/scipy.signal.chebwin.html#scipy.signal.chebwin)(M, at[, sym])	Return a Dolph-Chebyshev window.
cosine (generated/scipy.signal.cosine.html#scipy.signal.cosine)(M[, sym])	Return a window with a simple cosine shape.
flattop (generated/scipy.signal.flattop.html#scipy.signal.flattop)(M[, sym])	Return a flat top window.
gaussian (generated/scipy.signal.gaussian.html#scipy.signal.gaussian)(M, std[, sym])	Return a Gaussian window.
general_gaussian (generated/scipy.signal.general_gaussian.html#scipy.signal.general_gaussian)(M, p, sig[, sym])	Return a window with a generalized Gaussian shape.
hamming (generated/scipy.signal.hamming.html#scipy.signal.hamming)(M[, sym])	Return a

	Hamming window.
hann (generated/scipy.signal.hann.html#scipy.signal.hann)(M[, sym])	Return a Hann window.
kaiser (generated/scipy.signal.kaiser.html#scipy.signal.kaiser)(M, beta[, sym])	Return a Kaiser window.
nuttall (generated/scipy.signal.nuttall.html#scipy.signal.nuttall)(M[, sym])	Return a minimum 4-term Blackman-Harris window according to Nuttall.
parzen (generated/scipy.signal.parzen.html#scipy.signal.parzen)(M[, sym])	Return a Parzen window.
slepian (generated/scipy.signal.slepian.html#scipy.signal.slepian)(M, width[, sym])	Return a digital Slepian (DPSS) window.
triang (generated/scipy.signal.triang.html#scipy.signal.triang)(M[, sym])	Return a triangular window.

Wavelets

cascade (generated/scipy.signal.cascade.html#scipy.signal.cascade)(hk[, J])	Return (x, phi, psi) at dyadic points `K/2**J` from filter coefficients.
daub (generated/scipy.signal.daub.html#scipy.signal.daub)(p)	The coefficients for the FIR low-pass filter producing Daubechies wavelets.
morlet (generated/scipy.signal.morlet.html#scipy.signal.morlet)(M[, w, s, complete])	Complex Morlet wavelet.
qmf (generated/scipy.signal.qmf.html#scipy.signal.qmf)(hk)	Return high-pass qmf filter from low-pass
ricker (generated/scipy.signal.ricker.html#scipy.signal.ricker)(points, a)	Return a Ricker wavelet, also known as the “Mexican hat wavelet”.
cwt (generated/scipy.signal.cwt.html#scipy.signal.cwt)(data, wavelet, widths)	Continuous wavelet transform.

Peak finding

find_peaks_cwt (generated/scipy.signal.find_peaks_cwt.html#scipy.signal.find_peaks_cwt)(vector, widths[, wavelet, ...])	Attempt to find the peaks in a 1-D array.
argrelmin (generated/scipy.signal.argrelmin.html#scipy.signal.argrelmin)(data[, axis, order, mode])	Calculate the relative minima of *data*.
argrelmax (generated/scipy.signal.argrelmax.html#scipy.signal.argrelmax)(data[, axis, order, mode])	Calculate the relative

	maxima of *data*.
argrelextrema (generated/scipy.signal.argrelextrema.html#scipy.signal.argrelextrema)(data, comparator[, axis, …])	Calculate the relative extrema of *data*.

Spectral Analysis

periodogram (generated/scipy.signal.periodogram.html#scipy.signal.periodogram)(x[, fs, window, nfft, detrend, …])	Estimate power spectral density using a periodogram.
welch (generated/scipy.signal.welch.html#scipy.signal.welch)(x[, fs, window, nperseg, noverlap, …])	Estimate power spectral density using Welch’s method.
lombscargle (generated/scipy.signal.lombscargle.html#scipy.signal.lombscargle)(x, y, freqs)	Computes the Lomb-Scargle periodogram.
vectorstrength (generated/scipy.signal.vectorstrength.html#scipy.signal.vectorstrength)(events, period)	Determine the vector strength of the events corresponding to the given period.