Manual for Package: mathematics
Revision 33M

Karl Kästner

July 25, 2023

Contents

1 calendar

1.1 days_per_month

1.2 isnight

2 mathematics

`mathematical functions of various kind`

2.1 cast_byte_to_integer

` cast byte to integer`

3 complex-analysis

`operations on complex numbers`

3.1 complex_exp_product_im_im

```
 product of the imaginary part of two complex exponentials

the product has two frequency components

input :
      c : complex amplitudes
      o : frequencies
output :
      cp : amplitude of the product
      op : frequencies of the product
```

3.2 complex_exp_product_im_re

```
 product of the imaginary part of one and the real part of a second
 complex exponential

the product has two frequency components

input :
      c : complex amplitudes
      o : frequencies
output :
      cp : amplitude of the product
      op : frequencies of the product
```

3.3 complex_exp_product_re_im

```
the product has two frequency components

 product of the imaginary part of one and the real part of a second
 complex exponential

input :
      c : complex amplitudes
      o : frequencies
output :
      cp : amplitude of the product
      op : frequencies of the product
```

3.4 complex_exp_product_re_re

```
 product of the real part of two complex exponentials

re(c1 exp(io1x))*re(c2 exp(io2x)) =
      1/2*(    real(c1*c2*exp(i*(n1+n2)*o*x)) ...
            + real(conj(c1)*c2*exp(i*(n2-n1)*o*x)) )

the product has two frequency components

input :
      c : complex amplitudes
      o : frequencies
output :
      cp : amplitude of the product
      op : frequencies of the product
```

3.5 croots

```
nth-roots of a complex number

input:
c : complex number
n : order of root
   n must be rational, to obtain n solutions
   otherwise no finite set of solutions exists

r : roots of the complex number
```

3.6 root_complex

```
root of a complex number
```

3.7 test_imroots

4 derivation

derivation of several functions by means of symbolic computation

4.1 derive_acfar1

4.2 derive_ar1_spectral_density

4.3 derive_ar2param

4.4 derive_arc_length

4.5 derive_fourier_power

4.6 derive_fourier_power_exp

4.7 derive_laplacian_curvilinear

4.8 derive_laplacian_fourier_piecewise_linear

4.9 derive_logtripdf

4.10 derive_phase_drift_inv

4.11 derive_smooth1d_parametric

4.12 derive_spectral_density_bandpass_initial_condition

5 derivation/master

5.1 derive_bc_one_sided

5.2 derive_convergence

5.3 derive_error_fdm

5.4 derive_fdm_poly

5.5 derive_fdm_power

5.6 derive_fdm_taylor

5.7 derive_fdm_vargrid

5.8 derive_fem_2d_mass

5.9 derive_fem_error_2d

5.10 derive_fem_error_3d

5.11 derive_fem_sym_2d

5.12 derive_grid_constants

5.13 derive_interpolation

5.14 derive_laplacian

5.15 derive_limit

5.16 derive_nc_1d

5.17 derive_nc_1d_

5.18 derive_nc_2d

5.19 derive_nonuniform_symmetric

%

5.20 derive_richardson

5.21 derive_sum

5.22 nn

5.23 test_derive

5.24 test_derive_fdm_poly

5.25 test_filter

5.26 test_vargrid

6 derivation

`derivation of several functions by means of symbolic computation`

6.1 simplify_atan

`symbolic simplification of the arcus tangent`

7 mathematics

`mathematical functions of various kind`

7.1 entropy

8 finance

8.1 derive_skewrnd_walsh_paramter

8.2 gbb_geostd_entire_series

8.3 gbb_mean

8.4 gbb_simulate

8.5 gbb_std

8.6 gbm_bridge

8.7 gbm_cdf

8.8 gbm_fit

8.9 gbm_fit_old

8.10 gbm_geomean

8.11 gbm_geostd

8.12 gbm_inv

8.13 gbm_mean

8.14 gbm_mean_entire_series

8.15 gbm_median

8.16 gbm_moment2par

8.17 gbm_moment2par_entire_series

8.18 gbm_pdf

8.19 gbm_simulate

8.20 gbm_skewness

8.21 gbm_std

8.22 gbm_std_entire_series

8.23 gbm_transform_time_step

8.24 put_price_black_scholes

8.25 skewgbm_simulate

8.26 skewrnd_walsh

9 finance/test

9.1 test_gbm

9.2 test_gbm_pdf

9.3 test_skewrnd_walsh

10 fourier/@STFT

10.1 STFT

```
class for short time fourier transform

Note: the interval Ti should be set to at leat 2*max(T), as
    otherwise coefficients
      tend to oscillate in the presence of noise
Note: for convenience, the independent variable is labeled as time
    (t),
      but the independent variable is arbitrary, so it works
          likewise in space
```

10.2 itransform

```
inverse of the short time fourier transform
```

10.3 stft_

```
static wrapper for STFT
```

10.4 stftmat

```
transformation matrix for the short time fourier transform
```

10.5 transform

```
short time fourier transform
```

11 fourier

```
support and analysis functions both for the discrete (fast) fourier
    transform (dft/fft)
and continuous fourier analysis (fourier series)
```

11.1 amplitude_from_peak

```
amplitude and standard deviation of the amplitude of a frequency
    component
represented by a peak in the fourier domain
input :
h : peak height
w : peak width at half height

output:
a : amplitude in real space
s : standard deviation of the frequency (!)
```

11.2 caesaro_weight

11.3 dftmtx_man

```
fourier matrix in matlab style with a limited number of rows,
columns of higher frequencies are omitted

input :
n  : number of samples
nr : number of columns
```

```
output :
F : fourier matrix
```

11.4 example_fourier_window

11.5 fft2_cartesian2radial

11.6 fft_man

```
fast fourier transform for complex input data

input:
F : data in real space

output :

F : fourier transformation of F
```

11.7 fft_rotate

11.8 fftsmooth

```
smooth the fourier transform and determine upper and lower bound
    confidence intervals

input :
f :
sfunc : a smoothing function (for example fir convolution with
    rectangular window)
        returns filtered (mean) value and normalized fir window
nf    : window length
nsigma : number of standard deviations for confidnce intervals


output :
```

```
ff : filtered fourier transform
l : lower bound
u : upper bound
```

11.9 fix_fourier

```
fill gaps (missing data) by means of fourier extrapolation

fix periodic data series with fourier interpolation
longest gap should not exceed 1/2 of the shortest time span of
    interest (1/cutoff frequency)
 note: this limit equals the position of first side lobe of the ft
     of a rectangular window with gap length
```

11.10 fourier_2d_padd

11.11 fourier_2d_quadrants

11.12 fourier_axis

```
return axis of frequencies and periods for the discrete fourier
    transform
as computed by fft (matlab-style)

input:
X : sample locations (equal interval)
L : length of samples
n : number of samples

output :
f    : frequencies
T    : periods
mask : mask for 1/2 of the fourier transform
       (as both halves are complex conjugates)
N    : frequency id
```

11.13 fourier_axis_2d

```
frequency axis of the 2d fourier transform as computed by Matlab
function [fx, fy, fr, ft, Tx, Ty, mask, N] = fourier_axis_2d(L,n)
```

11.14 fourier_cesaro_correction

11.15 fourier_coefficient_piecewise_linear

```
fourier series coefficients of a piecewise linear function
(not coefficient of discrete fourier transform)
function can be discontinuous between intervals
scales domain length to 2pi

input :
l,r : end points of piecewise linear function
lval, rval : values at end points
L : length of domain
n : number of samples/highest frequency

output :
a, b : coefficients for frequency components
```

11.16 fourier_coefficient_piecewise_linear_1

```
fourier series coefficients of a piecewise linear function
(not coefficient of discrete fourier transform)
function can be discontinuous between intervals
scales domain length to 2pi

input :
X : end points of piecewise linear function
Y : values at end points

output :
ab : coefficients for frequency components
```

11.17 fourier_coefficient_ramp3

```
fourier series coefficient of a ramp
```

11.18 fourier_coefficient_ramp_pulse

```
fourier series coefficient of a ramp pules
```

11.19 fourier_coefficient_ramp_step

```
fourier coefficient of a ramp-step
```

11.20 fourier_coefficient_square_pulse

```
fourier series coefficients of a square pulse
```

11.21 fourier_complete_negative_half_plane

11.22 fourier_cubic_interaction_coefficients

11.23 fourier_derivative

```
derivative via fourier transform
exponential convergence for periodic functions
results in spurious oscillations for aperiodic functions

input:
x : data, sampled in equal intervals
k : order of the derivative

dx : kth-derivative of x

note : 1) the derivative converges with spectral accuracy, i.e. is
         exact up to rounding condition for L sufficiently large
           and x being periodic
        2) the derivative converges with order p, when x has only
           p-continous derivatives, including discontinuous
               derivatives
           over the boundary
        3) discontinuous derivatives result in gibbs phenomenon
```

11.24 fourier_derivative_matrix_1d

11.25 fourier_derivative_matrix_2d

11.26 fourier_expand

```
expand values of fourier series
```

11.27 fourier_fit

```
fit a fourier series to a set of sample points that are not spaced
    in
equal intervals
```

11.28 fourier_freq2ind

11.29 fourier_interpolate

```
interpolate samples y sampled at moments (location) t to locations
    ti
```

11.30 fourier_matrix

```
transformation matrix for a continuous fourier series
(not for the discrete dft/fft)
```

11.31 fourier_matrix2

```
transformation matrix for a continuous fourier series
(not for the discrete dft/fft)
```

11.32 fourier_matrix3

```
transformation matrix for the continous fourier transform
this is a matrix with (2*n+1) real columns
```

11.33 fourier_matrix_exp

```
transformation matrix for a continuous fourier series
(not for the discrete dft/fft)
```

11.34 fourier_multiplicative_interaction_coefficients

11.35 fourier_power

```
powers of a continuous fourier series in sin/cos form

powers of a^p = (ur + u1 sin(ot) + u2 sin(ot+dp))^p
phase of first component assumed 0

frequencies higher than 2-omega ignored in input
frequencies higher than 3-omega not computed
```

11.36 fourier_power_exp

```
powers of the continuous fourier series
      a^p = (ur + u1 sin(ot) + u2 sin(ot+dp))^p
phase of first component assumed 0

higher orders than 2 ignored input
higher order than 3 not computed in output

y = a_0 + sum (a_j sin(jot) + b_j cos(jot))
  = Real(sum_{i=0}^inf c_i exp(1i*omega), c_i = a_i + b_i

NOT the alternative sum_{i=-inf}^inf \tilde c_i, tile c_j = 1/2 a_j
     + 1/2i b_j
```

11.37 fourier_predict

```
expand a continous fourier series at times t
```

11.38 fourier_quadratic_interaction_coefficients

11.39 fourier_random_phase_walk

```
evaluete fourier series where the phase undergoes a brownian motion
```

11.40 fourier_range

```
approximate range of a continous Fourier series with 2 components
range(y) = max(y) - min(y)
```

11.41 fourier_regress

```
fit a continous fourier series to a set of sample points not
    sampled
at equal intervals
```

11.42 fourier_resampled_fit

```
fits coefficients of a continuous fourier transform,
but stores them as resampled values
```

11.43 fourier_resampled_predict

```
interpolates a continuous fourier series that has been stored as
    values
at their support points
```

11.44 fourier_series_signed_square

```
coefficients of the Fourier series of Q|Q|
      Q|Q| = Q_a^2 y                             (8.5)
         = | cos a + cos t | (cos a + cos t) (8.6)
         = a0 + a1 cos t + ... + an cos n t      (8.7)
          cos a is midrange
          cos t is tidal variation
c.f Dronkers 1964, eq. 8.10
```

11.45 fourier_transform

```
continuous fourier transformation of y
(not discrete fourier transformation dft/fft)

input:
      b : data sampled at equal intervals
      T : length of data in time or space, i.e. position of last
          sample if
          position of first sample is 0
     T_max : maximum period to include

output :
      A  : fourier matrix
     p  : fourier transformation of b
      tt : TODO
```

11.46 fourier_transform_fractional

11.47 fourier_truncate_negative_half_plane

11.48 hyperbolic_fourier_box

11.49 idftmtx_man

```
inverse matrix for the discrete fourier transform in matlab style
with a limited number of columns, thus ignoring higher frequencies
keep 2nc+1 columns (mean and conj-complex pairs of nc frequencies)
```

11.50 laplace_2d_pwlinear

```
solution to the Laplacian in two dimensions for a finite
    rectangular domain
with piecewise constant boundary conditions
linear system with 4 unknowns per freqency component
these are coefficients of s,c,sh,ch
      (pu*(s + c) + qu*(s' + c'))*(shu + chu) = ru      % upper bc
      (pd*(s + c) + qd*(s' + c'))*(shd + chd) = rd      % lower bc
      ( (sl + cl)*( pl*(shl + chl) + ql*(shl' + chl')) = rl % left
          bc
      ( (sr + cr)*( pr*(shr + chr) + qr*(shr' + chr')) = rr % right
           bc

 least squares with piecewise integration
[x0,p,q,r] piecewise linear polynomials at the boundaries
```

11.51 mean_fourier_power

11.52 moments_fourier_power

11.53 nanfft

```
discrete fourier transform of a data series with gaps
```

11.54 peaks

```
peaks of the power spectrum of a disctrete fourier transform

rule for peaks: there is no higher value left or right of the "peak
    "
              until the signal drops to p*y_peak, p = 0.5

works best, when spectrum has been smoothened

input :
f : frequency
y : absolute value of fourier transform (power spectrum)
L : length in space or time of series

output :

a0 : amplitude
s0 : standard deviation (error?) of amplitude
w0 : width of peak
lambda = wave length (period?)
pdx : index of peak
f : frequency (if not given as input)
```

11.55 roots_fourier

```
zeros of continuous fourier series series

      f  = a_0 + sum_j=^n a_i cos(j x) + b_i sin(j x)
```

11.56 spectral_density

```
spectral density
```

11.57 std_fourier_power

11.58 test_complex_exp_product

11.59 test_fourier_filter

11.60 test_idftmtx

11.61 var_fourier_power

12 mathematics

```
mathematical functions of various kind
```

12.1 gaussfit_quantile

13 geometry/@Geometry

13.1 Geometry

13.2 arclength

```
 arc length of a two dimensional curve

 8th order accurate
 does not require the segments length to vary smoothly

 note: the curve can be considered parametric, e.g. x = x(t), y=y(t)
      and
        and t = t(s), but the error term contains derivatives of t,
      thus a non smooth t (strongly varying distance between points)
      requires the scaling as done below
```

13.3 arclength_old

```
 arc length of a two dimensional function
```

13.4 arclength_old2

```
arc length of a two dimensional function
```

13.5 base_point

```
base point (fusspunkt), i.e. point on a line with shortest distance
to another point
```

13.6 base_point_limited

```
base point (Fusspunkt) of a point on a line
```

13.7 centroid

```
centroid of a polygone
```

13.8 cosa_min_max

13.9 cross2

```
cross product in two dimensions
```

13.10 curvature

```
curvature of a function in two dimensions
```

13.11 ddot

```
sum of squares of cos of inner angles of triangle
```

13.12 distance

```
equclidan distance between two points
```

13.13 distance2

```
euclidean distance between two points
this function requires a and be of equal dimensions, or the least
    the first pair or second pair to be a scalar
```

13.14 dot

```
dot product
```

13.15 edge_length

```
edge length
```

13.16 enclosed_angle

```
angle enclosed between two lines
```

13.17 enclosing_triangle

```
smallest enclosing equilateral triangle with bottom site paralle to
     X axis
```

13.18 hexagon

```
coordinates of a hexagon, scaled and rotated
```

13.19 inPolygon

```
flag points contained in a polygon
much faster than matlab internal function
```

13.20 inTetra

```
flag points contained in tetrahedron
```

13.21 inTetra2

```
flag points contained in tetrahedron
```

13.22 inTriangle

```
flag points contained in triangle
function [flag, c] = inTriangle(P1,P2,P3,P0)
```

13.23 intersect

```
intersect between two lines
```

13.24 lineintersect

```
intersect of two lines
```

13.25 lineintersect1

```
intersect of two lines
```

13.26 minimum_distance_lines

```
minimum distance of two lines in three dimensions
```

13.27 mittenpunkt

```
mittenpunkt of a triangle
```

13.28 nagelpoint

```
nagelpoint of a triangle
```

13.29 onLine

13.30 orthocentre

```
orthocentre of triangle
```

13.31 plumb_line

13.32 poly_area

```
area of a polygon
function A = poly_area(x,y)
```

13.33 poly_edges

```
edges of a polygon
```

13.34 poly_set

```
associate point at arbitary location with a polygon it is contained
     in
and assign the value of the polygon to it
```

13.35 poly_width

```
width of polygon width holes by surface normals
holes / islands separated with NaN
order of points of outer boundary must be cw
order of points of holes must be ccw
note that this function does not give the true width for expanding
    sections
use voronoi polygons for this
```

13.36 polyxpoly

```
intersections of two polygons
```

13.37 project_to_curve

```
closest point on a curve with respect to a point at distance to the
     curve
```

13.38 quad_isconvex

13.39 random_disk

```
draw random points on the unit disk
```

13.40 random_simplex

```
random point inside of a triangle
```

13.41 sphere_volume

```
volume of a sphere
function v = sphere_volume(r)
```

13.42 tetra_volume

```
volume of a tetrahedron
```

13.43 tobarycentric

```
cartesian to barycentric coordinates
```

13.44 tobarycentric1

```
cartesian to barycentric coordinates
```

13.45 tobarycentric2

```
cartesian to barycentric coordinates
```

13.46 tobarycentric3

```
cartesian to barycentric coordinates
```

13.47 tri_angle

```
cos of angles of a triangle
```

13.48 tri_area

```
angle of a triangle
```

13.49 tri_centroid

```
centroid of a triangle
```

13.50 tri_distance_opposit_midpoint

```
distance between corner of a triangle and its opposing mid-point
```

13.51 tri_edge_length

```
edge length of a triangle
```

13.52 tri_edge_midpoint

```
mid point of a triangle
```

13.53 tri_excircle

```
excircle of a triangle
```

13.54 tri_height

```
height of a triangle
```

13.55 tri_incircle

```
incircle of a triangle
```

13.56 tri_isacute

```
flag acute triangles
```

13.57 tri_isobtuse

```
flag obntuse triangles
```

13.58 tri_semiperimeter

```
semiperimeter of a triangle
```

13.59 tri_side_length

```
edge lenght of triangle
```

14 geometry

14.1 Polygon

```
Simple 2D polygon class

  Polygon properties:
      x - x coordinates of polygon
      y - y coordinates of polygon
      nnodes - number of nodes in the polygon

  Polygon methods:
      in - checks whether given points lie inside, on the edge, or
          outside of the polygon
      area - returns the area of the polygon
      centerline - computes the centerline of the river
      iscw - check whether polygon is clockwise
      reverse - reverse the order of the polygon
```

14.2 bounding_box

```
bounding box of X
```

14.3 curvature_1d

```
curvature of a sampled parametric curve in two dimensions
```

14.4 cvt

```
centroidal voronoi tesselation
```

14.5 deg_to_frac

```
degree, minutes and seconds to fractions
```

14.6 ellipse

```
return points on an ellipse
n : number of points
ci : confidence interval, i.e. for 1 sigma
```

14.7 ellipseX

```
x-coordinates of y-coordinates of an ellipse
```

14.8 ellipseY

14.9 first_intersect

```
get first intersection between lines in A and B
```

14.10 golden_ratio

```
golden ratio
```

14.11 hypot3

```
hypothenuse in 3D
```

14.12 meanangle

```
weighted mean of angles
```

14.13 meanangle2

```
mean angle
```

14.14 meanangle3

```
mean angle
```

14.15 meanangle4

```
mean angle
```

14.16 mediangle

```
median angle
angle, that has the smallest squared distance to all others
```

14.17 medianangle2

```
median angle

input
alpha : x*m, [rad] angle

ouput
ma    : 1*m, [rad] median angle
sa    : 1*m, [rad] standard error of median angle for uncorrelated
    error
```

14.18 pilim

```
limit to +- pi
```

14.19 streamline_radius_of_curvature

```
streamline radius of curvature
simplifies when rotatate to streamwise coordinates to R = 1/dv/ds *
     u
```

15 histogram/@Histogram

15.1 2x

15.2 Histogram

15.3 bimodes

15.4 cdf

15.5 cdfS

15.6 chi2test

15.7 cmoment

15.8 cmomentS

15.9 entropy

15.10 entropyS

15.11 export_csv

15.12 iquantile

15.13 kstest

15.14 kurtosis

15.15 kurtosisS

15.16 mean

15.17 meanS

15.18 median

15.19 medianS

15.20 mode

15.21 modeS

15.22 moment

15.23 momentS

15.24 pdf

15.25 quantile

15.26 quantileS

15.27 resample

15.28 setup

15.29 skewness

15.30 skewnessS

15.31 stairs

15.32 stairsS

15.33 std

15.34 stdS

15.35 var

15.36 varS

16 histogram

16.1 hist_man

16.2 histadapt

16.3 histconst

16.4 pdf_poly

16.5 plotcdf

16.6 test_histogram

17 mathematics

```
mathematical functions of various kind
```

17.1 imrotmat

18 linear-algebra

18.1 averaging_matrix_2

18.2 colnorm

```
 norms of columns
```

18.3 condest_

```
estimation of the condition number
```

18.4 connectivity_matrix

19 linear-algebra/coordinate-transformation

19.1 barycentric2cartesian

```
barycentric to cartesian coordinates
```

19.2 barycentric2cartesian3

```
convert barycentric to cartesian coordinates
```

19.3 cartesian2barycentric

```
cartesian to barycentric coordinates
```

19.4 cartesian_to_unit_triangle_basis

```
transform coodinates into unit triangle
```

19.5 ellipsoid2geoid

19.6 example_approximate_utm_conversion

19.7 latlon2utm

```
transform latitude and longitude to WGS84 UTM
```

19.8 latlon2utm_simple

19.9 lowrance_mercator_to_wgs84

```
convert lowrance coordinates to wgs84

based on spreadsheet by D Whitney King and Patty B at Lowrance
```

19.10 nmea2utm

```
convert nmea messages to utm coordinates
```

19.11 sn2xy

```
convert sn to xy coordinates
```

19.12 unit_triangle_to_cartesian

```
transform coordinates in unit triangle to cartesian coordinates
```

19.13 utm2latlon

```
convert wgs84 utm to latitute and longitude
```

19.14 xy2nt

project all points onto the cross section and assign them nz-coordinates

transform coordinate into N-T reference
rotate coordinate, so that cross section goes along x-axis
then x and y are n and t respectively scaled by width
N and T coordinates

19.15 xy2sn

convert cartesian to streamwise coordiantes

19.16 xy2sn_java

use java port for speed up

19.17 xy2sn_old

transform points from cartesian into streamwise coordinates

NOTE : prefer the java version, this has some problems with round off

20 linear-algebra

20.1 deflation_matrix

20.2 det2x2

2x2 matrix inverse of 2x2 matrices stacked along dim 3

20.3 det3x3

```
determinant of stacked 3x3 matrices
```

20.4 det4x4

```
determinant of stacked 4x4 matrices
```

20.5 diag2x2

```
diagonal of stacked 2x2 matrices
```

20.6 down

20.7 eig2x2

```
eigenvalues of stacked 2x2 matrices
```

21 linear-algebra/eigenvalue

21.1 eig_bisection

21.2 eig_inverse

21.3 eig_inverse_iteration

21.4 eig_power_iteration

22 linear-algebra/eigenvalue/jacobi-davidson

22.1 afun_jdm

22.2 davidson

22.3 jacobi_davidson

22.4 jacobi_davidson_qr

22.5 jacobi_davidson_qz

22.6 jacobi_davidson_simple

22.7 jdqr

```
% Read/set parameters
% Initiate global variables
% Return if eigenvalueproblem is trivial
% Initialize V, W:
%   V,W orthonormal, A*V=W*R+Qschur*E, R upper triangular
% The JD loop (Standard)
%    V orthogonal, V orthogonal to Qschur
%    V*V=eye(j), Qschur'*V=0,
```

```
%    W=A*V, M=V'*W
%
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%    Both V and W orthonormal and orthogonal w.r.t. Qschur
%    V*V=eye(j), Qschur'*V=0, W'*W=eye(j), Qschur'*W=0
%    (A*V-tau*V)=W*R+Qschur*E, E=Qschur'*(A*V-tau*V), M=W'*V
%
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%    V W AV.
%    Both V and W orthonormal and orthogonal w.r.t. Qschur, AV=A*V-
    tau*V
%    V*V=eye(j), W'*W=eye(j), Qschur'*V=0, Qschur'*W=0,
%    (I-Qschur*Qschur')*AV=W*R, M=W'*V; R=W'*AV;
```

```
%
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%    W orthonormal, V and W orthogonal to Qschur,
%    W'*W=eye(j), Qschur'*V=0, Qschur'*W=0
%    W=(A*V-tau*V)-Qschur*E, E=Qschur'*(A*V-tau*V),
%    M=W'*V
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
 W=V*Q; V=V(:,1:j)/R; E=E/R; R=eye(j); M=Q(1:j,:)'/R;
 W=V*H; V(:,j+1)=[];R=R'*R; M=H(1:j,:)';
%======== ARNOLDI (for initializing spaces)
    ================================
%=========== END ARNOLDI
    ============================================
% not accurate enough M=Rw'\(M/Rv);
```

```
%=========== COMPUTE SORTED JORDAN FORM
    ==================================
% compute vectors and matrices for skew projection
% solve preconditioned system
% 0 step of bicgstab eq. 1 step of bicgstab
% Then x is a multiple of b
% HIST=[0,1];
 explicit preconditioning
% compute norm in l-space
%  HIST=[HIST;[nmv,rnrm/snrm]];
% sufficient accuracy. No need to update r,u
 implicit preconditioning
% collect the updates for x in l-space
% but, do the orth to Q implicitly
% compute norm in l-space
% HIST=[HIST;[nmv,rnrm/snrm]];
% sufficient accuracy. No need to update r,u
% Do the orth to Q explicitly
% In exact arithmetic not needed, but
% appears to be more stable.
%  plot(HIST(:,1),log10(HIST(:,2)+eps),'*'), drawnow, pause
% 0 step of gmres eq. 1 step of gmres
% Then x is a multiple of b
%======================================================================

% 0 step of gmres eq. 1 step of gmres
% Then x is a multiple of b
 HIST=1;
% Lucky break-down
 HIST=[HIST;(gamma~=0)/sqrt(rho)];
% Lucky break-down
% solve in least square sense
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow,% pause
 r=r/rho; rho=1;
%  HIST=rho;
%  HIST=[HIST;rho];
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow,% pause
% HIST = rho;
%  HIST=[HIST;rho];
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow, pause
% HIST = rho;
%  HIST=[HIST;rho];
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow, pause
%------ compute schur form -------------
 A*Q=Q*S, Q'*Q=eye(size(A));
```

```
% transform real schur form to complex schur form
%------ find order eigenvalues ---------------
%------ reorder schur form ----------------
%------ compute qz form ----------------
%------ sort eigenvalues ---------------
%------ sort qz form -------------------
% i>j, move ith eigenvalue to position j
% determine dimension
% defaults
%% 'v'
```

22.8 jdqr_sleijpen

```
% Read/set parameters
% Initiate global variables
% Return if eigenvalueproblem is trivial
% Initialize V, W:
%  V,W orthonormal, A*V=W*R+Qschur*E, R upper triangular
% The JD loop (Standard)
%   V orthogonal, V orthogonal to Qschur
%   V*V=eye(j), Qschur'*V=0,
%   W=A*V, M=V'*W
%
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%   Both V and W orthonormal and orthogonal w.r.t. Qschur
%   V*V=eye(j), Qschur'*V=0, W'*W=eye(j), Qschur'*W=0
%   (A*V-tau*V)=W*R+Qschur*E, E=Qschur'*(A*V-tau*V), M=W'*V
%
% Compute approximate eigenpair and residual
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%    V W AV.
%    Both V and W orthonormal and orthogonal w.r.t. Qschur, AV=A*V-
    tau*V
%    V*V=eye(j), W'*W=eye(j), Qschur'*V=0, Qschur'*W=0,
%    (I-Qschur*Qschur')*AV=W*R, M=W'*V; R=W'*AV;
%
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%    W orthonormal, V and W orthogonal to Qschur,
%    W'*W=eye(j), Qschur'*V=0, Qschur'*W=0
%    W=(A*V-tau*V)-Qschur*E, E=Qschur'*(A*V-tau*V),
%    M=W'*V
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
 W=V*Q; V=V(:,1:j)/R; E=E/R; R=eye(j); M=Q(1:j,:)'/R;
 W=V*H; V(:,j+1)=[];R=R'*R; M=H(1:j,:)';
%======== ARNOLDI (for initializing spaces)
    ===============================
%=========== END ARNOLDI
    ============================================
% not accurate enough M=Rw'\(M/Rv);
%=========== COMPUTE SORTED JORDAN FORM
    ==================================
% compute vectors and matrices for skew projection
% solve preconditioned system
% 0 step of bicgstab eq. 1 step of bicgstab
% Then x is a multiple of b
% HIST=[0,1];
 explicit preconditioning
% compute norm in l-space
%  HIST=[HIST;[nmv,rnrm/snrm]];
% sufficient accuracy. No need to update r,u
 implicit preconditioning
% collect the updates for x in l-space
% but, do the orth to Q implicitly
% compute norm in l-space
% HIST=[HIST;[nmv,rnrm/snrm]];
% sufficient accuracy. No need to update r,u
% Do the orth to Q explicitly
% In exact arithmetic not needed, but
% appears to be more stable.
%  plot(HIST(:,1),log10(HIST(:,2)+eps),'*'), drawnow, pause
% 0 step of gmres eq. 1 step of gmres
% Then x is a multiple of b
%=====================================================================

% 0 step of gmres eq. 1 step of gmres
```

```
% Then x is a multiple of b
 HIST=1;
% Lucky break-down
 HIST=[HIST;(gamma~=0)/sqrt(rho)];
% Lucky break-down
% solve in least square sense
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow,% pause
 r=r/rho; rho=1;
%  HIST=rho;
%  HIST=[HIST;rho];
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow,% pause
% HIST = rho;
%  HIST=[HIST;rho];
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow, pause
% HIST = rho;
%  HIST=[HIST;rho];
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow, pause
%------ compute schur form -------------
 A*Q=Q*S, Q'*Q=eye(size(A));
% transform real schur form to complex schur form
%------ find order eigenvalues ---------------
%------ reorder schur form ----------------
%------ compute qz form ----------------
%------ sort eigenvalues ---------------
%------ sort qz form -------------------
% i>j, move ith eigenvalue to position j
% determine dimension
% defaults
%% 'v'
```

22.9 jdqr_vorst

```
% Read/set parameters
% Initiate global variables
% Return if eigenvalueproblem is trivial
% Initialize V, W:
%   V,W orthonormal, A*V=W*R+Qschur*E, R upper triangular
% The JD loop (Standard)
%    V orthogonal, V orthogonal to Qschur
%    V*V=eye(j), Qschur'*V=0,
%    W=A*V, M=V'*W
%
% Compute approximate eigenpair and residual
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%    Both V and W orthonormal and orthogonal w.r.t. Qschur
%    V*V=eye(j), Qschur'*V=0, W'*W=eye(j), Qschur'*W=0
%    (A*V-tau*V)=W*R+Qschur*E, E=Qschur'*(A*V-tau*V), M=W'*V
%
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%    V W AV.
%    Both V and W orthonormal and orthogonal w.r.t. Qschur, AV=A*V-
    tau*V
%    V*V=eye(j), W'*W=eye(j), Qschur'*V=0, Qschur'*W=0,
%    (I-Qschur*Qschur')*AV=W*R, M=W'*V; R=W'*AV;
%
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
% The JD loop (Harmonic Ritz values)
%    W orthonormal, V and W orthogonal to Qschur,
%    W'*W=eye(j), Qschur'*V=0, Qschur'*W=0
%    W=(A*V-tau*V)-Qschur*E, E=Qschur'*(A*V-tau*V),
%    M=W'*V
% Compute approximate eigenpair and residual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check for convergence
% Expand the partial Schur form
 Rschur=[[Rschur;zeros(1,k)],Qschur'*MV(u)]; k=k+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand preconditioned Schur matrix PinvQ
% Check for shrinking the search subspace
% Solve correction equation
% Expand the subspaces of the interaction matrix
 W=V*Q; V=V(:,1:j)/R; E=E/R; R=eye(j); M=Q(1:j,:)'/R;
 W=V*H; V(:,j+1)=[];R=R'*R; M=H(1:j,:)';
%======== ARNOLDI (for initializing spaces)
    ================================
%=========== END ARNOLDI
    =============================================
% not accurate enough M=Rw'\(M/Rv);
%=========== COMPUTE SORTED JORDAN FORM
    ==================================
% accepted separation between eigenvalues:
```

```
% no preconditioning
% solve left preconditioned system
% compute vectors and matrices for skew projection
% precondion and project r
% solve preconditioned system
% no preconditioning
% solve two-sided expl. precond. system
% compute vectors and matrices for skew projection
% precondion and project r
% solve preconditioned system
% "unprecondition" solution
%%%% u(:,j+1)=Atilde*u(:,j)
%%%% r(:,j+1)=Atilde*r(:,j)
%------ compute schur form -------------
 A*Q=Q*S, Q'*Q=eye(size(A));
% transform real schur form to complex schur form
%------ find order eigenvalues ---------------
%------ reorder schur form ----------------
%------ compute qz form ----------------
%------ sort eigenvalues ---------------
%------ sort qz form -------------------
% i>j, move ith eigenvalue to position j
% determine dimension
% defaults
```

22.10 jdqz

```
% Read/set parameters
% Return if eigenvalueproblem is trivial
% Initialize target, test space and interaction matrices
% V=RepGS(Qschur,V); [AV,BV]=MV(V); %%% more stability??
% W=RepGS(Zschur,eval(testspace)); %%% dangerous if sigma~lambda
% Solve the preconditioned correction equation
% Expand the subspaces and the interaction matrices
% Check for stagnation
% Solve projected eigenproblem
% Compute approximate eigenpair and residual
%=== an alternative, but less stable way of computing z =====
% display history
% save history
% check convergence
% EXPAND Schur form
% Expand preconditioned Schur matrix MinvZ=M\Zschur
% check for conjugate pair
% To detect whether another eigenpair is accurate enough
% restart if dim(V)> jmax
% Initialize target, test space and interaction matrices
```

```
% additional stabilisation. May not be needed
% V=RepGS(Zschur,V); [AV,BV]=MV(V);
% end add. stab.
% Solve the preconditioned correction equation
% expand the subspaces and the interaction matrices
% Check for stagnation
% compute approximate eigenpair
% Compute approximate eigenpair and residual
% display history
% save history
% check convergence
% expand Schur form
% ZastQ=Z'*Q0
% the final Qschur
% check for conjugate pair
% t perp Zschur, t in span(Q0,imag(q))
% To detect whether another eigenpair is accurate enough
% restart if dim(V)> jmax
%======== END JDQZ
    ====================================================
%========================================================================

%======== PREPROCESSING
    ===============================================
%========================================================================

%======== ARNOLDI (for initial spaces)
    ================================
%% then precond=I and target = 0: apply Arnoldi with A
%======== END ARNOLDI
    =================================================
%========================================================================

%======== POSTPROCESSING
    ==============================================
%========================================================================

%======== SORT QZ DECOMPOSITION INTERACTION MATRICES
    ==================
%======== COMPUTE SORTED JORDAN FORM
    ==================================
%======== END JORDAN FORM
    =============================================
%======== OUTPUT
    ======================================================
%========================================================================

%======== UPDATE PRECONDITIONED SCHUR VECTORS
    =========================
```

```
%=======================================================================

%=======================================================================

%======== SOLVE CORRECTION EQUATION
    ===================================
%=======================================================================

% solve preconditioned system
%=======================================================================

%======== LINEAR SOLVERS
    ==============================================
%=======================================================================

% [At,Bt]=MV(x); At=theta(2)*At-theta(1)*Bt;
% xtol=norm(r-At+Z*(Z'*At))/norm(r);
%===== Iterative methods
    ==============================================
% 0 step of bicgstab eq. 1 step of bicgstab
% Then x is a multiple of b
% HIST=[0,1];
 explicit preconditioning
% compute norm in l-space
% HIST=[HIST;[nmv,rnrm/snrm]];
% sufficient accuracy. No need to update r,u
 implicit preconditioning
% collect the updates for x in l-space
% but, do the orth to Z implicitly
% compute norm in l-space
% HIST=[HIST;[nmv,rnrm/snrm]];
% sufficient accuracy. No need to update r,u
% Do the orth to Z explicitly
% In exact arithmetic not needed, but
% appears to be more stable.
% plot(HIST(:,1),log10(HIST(:,2)+eps),'*'), drawnow
% 0 step of gmres eq. 1 step of gmres
% Then x is a multiple of b
%=======================================================================

% 0 step of gmres eq. 1 step of gmres
% Then x is a multiple of b
 HIST=1;
% Lucky break-down
 HIST=[HIST;(gamma~=0)/sqrt(rho)];
% Lucky break-down
% solve in least square sense
 HIST=log10(HIST+eps); J=[0:size(HIST,1)-1]';
 plot(J,HIST(:,1),'*'); drawnow
```

```
%======== END SOLVE CORRECTION EQUATION
    ===============================
%=======================================================================

%======== BASIC OPERATIONS
    ============================================
%=======================================================================

 y(1:5,1), pause
%======== COMPUTE r AND z
    =============================================
% E*u=Q*sigma, sigma(1,1)>sigma(2,2)
%======== END computation r and z
    =====================================
%=======================================================================

%======== Orthogonalisation
    ===========================================
%=======================================================================

%======== END Orthogonalisation
    ======================================
%=======================================================================

%======== Sorts Schur form
    ============================================
%=======================================================================

   kappa=max(norm(A,inf)/max(norm(B,inf),1.e-12),1);
   kappa=2^(round(log2(kappa)));
%------ compute the qz factorization -------
%------ scale the eigenvalues --------------
%------ sort the eigenvalues ---------------
%------ swap the qz form -------------------
% repeat SwapQZ if angle is too small
%=======================================================================

%=======================================================================

% i>j, move ith eigenvalue to position j
% compute q s.t. C*q=(t(i,1)*S-s(i,1)*T)*q=0
 C*P=Q*R
 check whether last but one diag. elt r nonzero
 C*q
% end computation q
%======== END sort QZ decomposition interaction matrices
    ==============
%=======================================================================
```

```
%======== INITIALIZATION
    ==============================================
%=======================================================================

%=======================================================================

% defaults            %%%% search for 'xx' in fieldnames
%% 'ma'
%% 'sch'
%% 'to'
%% 'di'
% jmin=nselect+p0 %%%% 'jmi'
% jmax=jmin+p1  %%%% 'jma'
%% 'te'
%% 'pai'
%% 'av'
%% 'tr'
%% 'fix'
%% 'ns'
%% 'ch'
%% 'lso'
%% 'ls_m'
%% 'ls_t'
%% 'ls_e'
%% 'ty'
%% 'l_'
%% 'u_'
%% 'p_'
%% 'sca'
%% 'v0'
 initiation
 'standard'
 'harmonic'
 'searchspace'
%=======================================================================

% or Operator_Form=3 or Operator_Form=5???
%=======================================================================

%========= DISPLAY FUNCTIONS
    ===========================================
%=======================================================================

%=======================================================================

%=======================================================================

%=======================================================================
```

22.11 mfunc_jdm

22.12 mgs

22.13 minres_

22.14 mv_jacobi_davidson

23 linear-algebra

23.1 first

23.2 gershgorin_circle

```
range of eigenvalues determined by the gershgorin circle theorem
```

23.3 haussdorff

```
haussdorf dimension
box counting: count cectangles passed through by line (covered by
    polygon)

Koch snow flake 3:4 -> 1.2619
Kantor set     2:3, (4:9) -> 0.6309
quadrat        4:2, 9:3, 16:4 -> 2
```

23.4 ieig2x2

```
reconstruct matrix from eigenvalue decomposition
```

23.5 inv2x2

```
2x2 inverse of stacked matrices
```

23.6 inv3x3

23.7 inv4x4

```
inverse of stacked 4x4 matrices
```

23.8 kernel2matrix

24 linear-algebra/lanczos

24.1 arnoldi

24.2 arnoldi_new

24.3 eigs_lanczos_man

24.4 lanczos

24.5 lanczos_

24.6 lanczos_biorthogonal

24.7 lanczos_biorthogonal_improved

24.8 lanczos_ghep

24.9 mv_lanczos

24.10 reorthogonalise

24.11 test_lanczos

25 linear-algebra

25.1 laplacian_eigenvalue

25.2 laplacian_eigenvector

25.3 laplacian_power

25.4 least_squares_perpendicular_offset

25.5 left

```
left element of vector, leftmost column is extrapolated
```

26 linear-algebra/linear-systems

26.1 gmres_man

```
break on convergence
```

26.2 minres_recycle

27 linear-algebra

27.1 lpmean

```
mean of pth-power of a
```

27.2 lpnorm

```
norm of lth-power of a
```

27.3 matvec3

```
matrix-vector product of stacked matrices and vectors
```

27.4 max2d

```
maximum value and i-j index for matrix
```

27.5 mid

```
mid point between neighbouring vector elements
```

27.6 mpoweri

```
approximation of A^p, where p is not integer by quadtratic
    interpolation
```

27.7 mtimes2x2

27.8 mtimes3x3

```
product of stacked 3x3 matrices
```

27.9 nannorm

```
norm of a vector, skips nan-values
```

27.10 nanshift

```
shift vector, but set out of range values to NaN
```

27.11 nl

```
number rows (lines) of a matrix

analogue to unix nl command
```

27.12 normalise

```
normalise a vector or the columns of a matrix
note that the columns are independently normalised, and hence not
    necessarily
orthogonal to each other use the gram schmidt algorithm for this (
    qr or orth)
```

27.13 normalize1

```
normalize columns in x to [-1,1]
```

27.14 normrows

27.15 orth2

```
make matrix A orhogonal to B
```

27.16 orth_man

```
orthogonalize the columns of A
```

27.17 orthogonalise

```
make x orthogonal to Y
```

27.18 padd2

```
padd values around a 2d (image) matrix, constant exprapolation
```

27.19 paddext

```
padd values to vactor
not suitable for noisy data
order = 0 : constant extrapolation (hold)
order = 1 : linear extrapolation
```

27.20 paddval1

```
padd values at end of x
```

27.21 paddval2

```
padd values to x
```

28 linear-algebra/polynomial

28.1 chebychev

```
c.f. Dronkers 1964, eq. 8.15, p. 300
chebycheff polynomials
function c = chebychev(x,n)
```

28.2 piecewise_polynomial

```
evaluate piecewise polynomial
```

28.3 roots1

```
roots of linear functions
```

28.4 roots2

```
roots of quadratic function
c1 x^2 + c2 x + c3 = 0
```

28.5 roots2poly

28.6 roots3

28.7 roots4

28.8 roots_piecewise_linear

28.9 test_roots4

28.10 vanderi_1d

```
vandermonde matrix of an integral
```

29 linear-algebra

29.1 randrot

```
random rotation matrix
```

29.2 right

```
get right column by shifting columns to left
extrapolate rightmost column
```

29.3 rot2

```
rotation matrix from angle
```

29.4 rot2dir

```
rotation matrix from direction vector
```

29.5 rot3

29.6 rotR

29.7 rownorm

29.8 simmilarity_matrix

29.9 spnorm

```
frobenius norm
```

29.10 spzeros

```
allocate a sparze matrix of zeros
```

29.11 test_roots3

29.12 transform_minmax

29.13 transpose3

```
transpose stacked 3x3 matrices
```

29.14 transposeall

29.15 up

29.16 vander_nd

29.17 vanderd_2d

30 logic

```
bitwise operations on integers
```

30.1 bitor_man

```
bitwise OR of the numbers of the columns of A

input:
      A (positive integer)
```

31 master/plot

31.1 attach_boundary_value

31.2 cartesian_polar

31.3 img_vargrid

31.4 plot_basis_functions

31.5 plot_convergence

31.6 plot_dof

31.7 plot_eigenbar

31.8 plot_error_estimation

31.9 plot_error_estimation_2

31.10 plot_error_fem

31.11 plot_fdm_kernel

31.12 plot_fdm_vs_fem

31.13 plot_fem_accuracy

31.14 plot_function_and_grid

31.15 plot_hat

31.16 plot_hydrogen_wf

31.17 plot_mesh

31.18 plot_mesh_2

31.19 plot_refine

31.20 plot_refine_3d

31.21 plot_runtime

31.22 plot_spectrum

31.23 plot_wavefunction

32 master/ported

32.1 assemble_2d_phi_phi

32.2 assemble_3d_dphi_dphi

32.3 assemble_3d_phi_phi

32.4 dV_2d_

32.5 derivative_2d

32.6 derivative_3d

32.7 element_neighbour_2d

32.8 prefetch_2d_

32.9 promote_2d_3_10

32.10 promote_2d_3_15

32.11 promote_2d_3_21

32.12 promote_2d_3_6

32.13 promote_3d_4_10

32.14 promote_3d_4_20

32.15 promote_3d_4_35

32.16 vander_2d

32.17 vander_3d

33 mathematics

```
mathematical functions of various kind
```

33.1 monotoneous_indices

33.2 nearest_fractional_timestep

34 number-theory

34.1 ceiln

```
floor to leading n-digits
```

34.2 digitsb

```
number of digits with respect to specified base
```

34.3 floorn

```
floor to n-digits
```

34.4 iseven

```
true for even numbers in X
```

34.5 multichoosek

```
all combinations of lenght k from set values with repetitions
c.f. nchoosek, combinations without repetition

input :
      x : scalar integer or vector of arbitrary numbers
      k : length of subsets
output :
      if x scalar : number of combinations
      if x vector : the exact combinations
```

34.6 nchoosek_man

```
vecotrised binomial coefficient
b = N!/K!(N-K)!
```

34.7 pythagorean_triple

```
pythagorean triple
```

34.8 roundn

```
round to n digits
```

35 numerical-methods

35.1 advect_analytic

35.2 advection_kernel

36 numerical-methods/differentiation

36.1 derivative1

```
first derivative on variable mesh
second order accurate
```

36.2 derivative2

```
second derivative on a variable mesh
```

37 numerical-methods/finite-difference

37.1 cdiff

```
differences of columns of X
degree = 1 : central first order differences
degreee = 2 : central second order differences
```

37.2 cdiffb

```
differences of columns of X
degree = 1 : central first order differences
degreee = 2 : central second order differences
TODO use difference matrix function for simplicity
```

37.3 central_difference

37.4 cmean

```
single gaussian smoothing step with kernel 1/4*[1,2,1]
```

37.5 cmean2

37.6 derivative_matrix_1_1d

```
finite difference matrix of first derivative in one dimensions
n : number of grid points
h = L/(n+1) constant step with

function [D1, d1] = derivative_matrix_1d(n,L,order)
```

37.7 derivative_matrix_2_1d

```
finite derivative matrix of second derivative in one dimension
```

37.8 derivative_matrix_2d

```
finite difference derivative matrix in two dimensions
```

37.9 derivative_matrix_curvilinear

```
derivative matrix on a curvilinear grid
```

37.10 derivative_matrix_curvilinear_2

```
derivative matrix on a two dimensional curvilinear grid
the grid has not necessarily to be orthogonal
```

37.11 difference_kernel

```
difference kernels for equispaced grids
c.f. Computing the Spectrum of the Confined Hydrogen Atom, Kastner,
     2012
```

37.12 diffusion_matrix_2d_anisotropic

37.13 diffusion_matrix_2d_anisotropic2

37.14 directional_neighbour

37.15 distmat

```
distance matrix for a 2 dimensional rectangular matrix
```

37.16 downwind_difference

37.17 gradpde2d

```
objective function gradiend on two dimensional regular grid
numeric gradient for non-linear least squares optimisation
of a PDE on a rectangular grid
x_* = min(f(x))
f = (v(x) - v(x_*))^2 = f(x) + A dx + O(dx^2)
a_ij = df_i/dx_j
```

37.18 laplacian

37.19 laplacian_fdm

```
finite difference matrix of the laplacian
BC
```

37.20 lrmean

```
mean of the left and right element
```

38 numerical-methods/finite-difference/master

38.1 fdm_adaptive_grid

38.2 fdm_adaptive_refinement_old

38.3 fdm_assemble_d1_2d

38.4 fdm_assemble_d2_2d

38.5 fdm_confinement

38.6 fdm_d_vargrid

38.7 fdm_h_unstructured

38.8 fdm_hydrogen_vargrid

38.9 fdm_mark_unstructured_2d

38.10 fdm_plot

38.11 fdm_plot_series

38.12 fdm_refine_2d

38.13 fdm_refine_3d

38.14 fdm_refine_unstructured_2d

38.15 fdm_schroedinger_2d

38.16 fdm_schroedinger_3d

38.17 relocate

39 numerical-methods/finite-difference

39.1 mid

```
mid point between neighbouring vector elements
```

39.2 pwmid

```
segment end point to segment mid point transformation for regular 1
    d grids
```

39.3 ratio

```
ratio of two subsequent values
```

39.4 steplength

```
step length of a vector if it were equispaced
```

39.5 swapoddeven

```
swap odd and even elements in a vector
```

39.6 test_derivative_matrix_2d

39.7 test_derivative_matrix_curvilinear

39.8 test_difference_kernel

39.9 upwind_difference

40 numerical-methods/finite-element

40.1 Mesh_2d_java

40.2 Tree_2d_java

40.3 assemble_1d_dphi_dphi

40.4 assemble_1d_phi_phi

40.5 assemble_2d_dphi_dphi_java

40.6 assemble_2d_phi_phi_java

40.7 assemble_3d_dphi_dphi_java

40.8 assemble_3d_phi_phi_java

40.9 boundary_1d

40.10 boundary_2d

40.11 boundary_3d

40.12 check_area_2d

40.13 circmesh

40.14 cropradius

40.15 display_2d

40.16 display_3d

40.17 distort

40.18 err_2d

40.19 estimate_err_2d_3

40.20 example_1d

40.21 example_2d

40.22 explode

40.23 fem_2d

40.24 fem_2d_heuristic_mesh

40.25 fem_get_2d_radial

40.26 fem_interpolation

40.27 fem_plot_1d

40.28 fem_plot_1d_series

40.29 fem_plot_2d

40.30 fem_plot_2d_series

40.31 fem_plot_3d

40.32 fem_plot_3d_series

40.33 fem_plot_confine_series

40.34 fem_radial

```
adaptive grid
constant grid
```

40.35 flip_2d

40.36 get_mesh_arrays

40.37 hashkey

41 numerical-methods/finite-element/int

41.1 int_1d_equal

41.2 int_1d_equal_exp

41.3 int_1d_gauss

41.4 int_1d_gauss_1

```
w : weights
      2/(1-xi^2)(P'_n(xi))^2
b : baricentric coordinates
    ith-root of legendre polynomial of order n
second order, midpoint rule
function [w, b, flag] = int_1d_gauss_1()
```

41.5 int_1d_gauss_2

41.6 int_1d_gauss_3

41.7 int_1d_gauss_4

41.8 int_1d_gauss_5

41.9 int_1d_gauss_6

41.10 int_1d_gauss_lobatto

41.11 int_1d_gauss_n

41.12 int_1d_nc_2

41.13 int_1d_nc_3

41.14 int_1d_nc_4

41.15 int_1d_nc_5

41.16 int_1d_nc_6

41.17 int_1d_nc_7

41.18 int_1d_nc_7_hardy

41.19 int_2d_gauss_1

41.20 int_2d_gauss_12

41.21 int_2d_gauss_13

41.22 int_2d_gauss_16

41.23 int_2d_gauss_19

41.24 int_2d_gauss_25

41.25 int_2d_gauss_3

41.26 int_2d_gauss_33

41.27 int_2d_gauss_4

41.28 int_2d_gauss_6

41.29 int_2d_gauss_7

41.30 int_2d_gauss_9

41.31 int_2d_nc_10

41.32 int_2d_nc_15

41.33 int_2d_nc_21

41.34 int_2d_nc_3

41.35 int_2d_nc_6

41.36 int_3d_gauss_1

41.37 int_3d_gauss_11

41.38 int_3d_gauss_14

41.39 int_3d_gauss_15

41.40 int_3d_gauss_24

41.41 int_3d_gauss_4

41.42 int_3d_gauss_45

41.43 int_3d_gauss_5

41.44 int_3d_nc_11

41.45 int_3d_nc_4

41.46 int_3d_nc_6

41.47 int_3d_nc_8

42 numerical-methods/finite-element

42.1 interpolation_matrix

42.2 mark

42.3 mark_1d

42.4 mesh_1d_uniform

42.5 mesh_3d_uniform

42.6 mesh_interpolate

42.7 neighbour_1d

42.8 old

42.9 pdeeig_1d

42.10 pdeeig_2d

42.11 pdeeig_3d

42.12 polynomial_derivative_1d

42.13 potential_const

42.14 potential_coulomb

42.15 potential_harmonic_oscillator

42.16 project_circle

42.17 project_rectangle

42.18 promote_1d_2_3

42.19 promote_1d_2_4

42.20 promote_1d_2_5

42.21 promote_1d_2_6

42.22 quadrilaterate

42.23 recalculate_regularity_2d

42.24 refine_1d

42.25 refine_2d_21

42.26 refine_2d_structural

42.27 regularity_1d

42.28 regularity_2d

42.29 regularity_3d

```
{       T = [1 2 3 4];
}
```

42.30 relocate_2d

42.31 test_circmesh

42.32 test_hermite

42.33 tri_assign_points

42.34 triangulation_uniform

42.35 vander_1d

```
van der Monde matrix
```

42.36 vanderd_1d

42.37 vanderi_1d

43 numerical-methods/finite-volume/@Advection

43.1 Advection

```
FVM treatment of the Advection equation
```

43.2 dot_advection

```
advection equation
```

44 numerical-methods/finite-volume/@Burgers

44.1 burgers_split

```
viscous Burgers' equation,
mixed analytic and numerical derivative in frequency space
by splitting sheme
u_t = -(0.5*u^2)_x + c*u_xx
```

44.2 dot_burgers_fdm

```
viscous burgers' equation
u_t = -d/dx (1/2*u^2) + c d^2/dx^2 u_xx
```

44.3 dot_burgers_fft

```
viscous Burgers' equation in frequency space
u_t + (0.5*u^2)_x = c*u_xx
```

45 numerical-methods/finite-volume/@Finite_Volume

45.1 Finite_Volume

```
finite volume method for partial differential equations 1+1
    dimensions
(time and space)
```

45.2 apply_bc

```
apply boundary conditions
```

45.3 solve

```
solve the the PDE by successively stepping in time
this is a trivial implmentation with constant step length
severity of diffusive error depends on dt/dx-ratio
stability depends on wave height
                     printf('Progress %2.1f%% %2.1fs\n',100*(t-Ti
                         (1))/(Ti(2)-Ti(1)),t_real);
```

45.4 step_split_strang

```
step in time, treat inhomogeneous part by Strang splitting
this scheme is not suitable for stationary solutions, for example
steady shallow water flow
```

45.5 step_unsplit

```
step in time, without splitting the inhomogeneous term
```

46 numerical-methods/finite-volume/@Flux_Limiter

46.1 Flux_Limiter

```
class of flux limiters
```

46.2 beam_warming

```
beam warming sheme
low resolution
note: works only if sign of eigenvalues point into the same
    direction according to RL
```

46.3 fromm

```
fromme limiter
low res
```

46.4 lax_wendroff

```
lax wendroff scheme
second order accurate, but no tvd
this is effectively not a limiter
eq. 6.39 in randall, leveque
```

46.5 minmod

```
min-mod schock limiter
```

46.6 monotized_central

```
monotonized central flux limiter
```

46.7 muscl

```
muscl flux limiter
```

46.8 superbee

```
superbee limiter
```

46.9 upwind

```
godunov scheme
godunov, first order accurate
```

46.10 vanLeer

```
van Leer limiter
```

47 numerical-methods/finite-volume/@KDV

47.1 dot_kdv_fdm

```
korteweg de vries equation
u_t + (0.5*u^2)_x = c*u_xxx
```

47.2 dot_kdv_fft

```
korteweg de vries equation
compute derivatives in frequency space
u_t + (0.5*u^2)_x = c*u_xxx
```

47.3 kdv_split

```
korteweg de vries equation in frequency space,
derivative treated by splitting scheme
```

48 numerical-methods/finite-volume/@Reconstruct_Average_Evolve

48.1 Reconstruct_Average_Evolve

```
Reconstruct Average Evolve Finite Volume Method for treatment of
    1+1D pdes

McCronack Scheme
err = O(dt^2) + O(dx^2), except as discontinuities
error:
      h_xxx(3:end-2) = 1/dx^3*( -0.5*h(1:end-4) + h(2:end-3) - h(4:
          end-1) + 0.5*h(5:end) );
      th = -1/6*dx^2*qh_.*(1 - (qh_*dt/dx).^2).*h_xxx;
```

48.2 advect_highres

```
single time step for the reconstruct evolve algorithm
```

48.3 advect_lowress

```
single time step
low resolution
```

49 numerical-methods/finite-volume

49.1 Godunov

```
Godunov, upwind method for systems of pdes
```

49.2 Lax_Friedrich

```
Lax-Friedrich-Method
for hyperbolic conservation laws
err = O(dt) + O(dx)
|a dt/dx| < 1
```

49.3 Measure

49.4 Roe

```
non linear roe solver for the SWE (randall, leveque 15.3.1)

The roe solver guarantess:
- A is diagonalisable with real eigenvalues (15.12)
- can be determined by a closed formula
- is an efficient replacement for true Rieman solver
```

49.5 fv_swe

```
wrapper for solving SWE
```

49.6 staggered_euler

```
forward euler method with staggered grid
```

49.7 staggered_grid

```
staggered grid approximation to the SWE
```

50 numerical-methods

50.1 grid2quad

```
extract rectangular elements of a structured grid
in form of an unstructured quad-mesh format
```

51 numerical-methods/integration

51.1 cumintL

```
cumulative integral from left to right
```

51.2 cumintR

```
cumulative integral from right to left
```

51.3 cumint_trapezoidal

```
integrate y along x with the trapezoidal rule
```

51.4 int_1d_gauss_laguerre

51.5 int_trapezoidal

```
integrate y along x with the trapezoidal rule
```

52 numerical-methods/interpolation/@Kriging

52.1 Kriging

```
class for Kriging interpolation
```

52.2 estimate_semivariance

```
estimate the parameter of the semivariance model for Kriging
    interpolation
              % set up the regression matrix and solve for
                  parameters
```

52.3 interpolate_

```
interpolate with Krieging method

this function may interpolate several quantities per coordinate,
using the same variogram, if the semivariance of the quantities
    differs,
the user may prefer to estimate the semivariance and interpolate
    each quantity
individually

Xs  : source point coordinates
Vs  : value at source points
Xt  : targe point coordinates
Vt  : value at target points
E2t : squared interpolation error at target points
```

53 numerical-methods/interpolation/@RegularizedInterpolator

53.1 RegularizedInterpolator1

```
class for regularized interpolation (Thikonov) on a 1D mesh
```

53.2 init

```
initialize the interpolator with a set of sampling points
```

54 numerical-methods/interpolation/@RegularizedInterpolator2

54.1 RegularizedInterpolator2

```
class for regularized interpolation on an unstructures mesh (
    interpolation)
```

54.2 init

```
initialize the interpolator with a set of point samples
```

55 numerical-methods/interpolation/@RegularizedInterpolator3

55.1 RegularizedInterpolator3

```
class for regularized interpolation (Tikhonov) on a triangulation
(unstructured mesh)
```

55.2 init

```
initialize the interpolator with a set of sampling points
```

56 numerical-methods/interpolation

56.1 IDW

```
spatial averaging by inverse distance weighting
```

56.2 IPoly

```
polynomial interpolation class
```

56.3 IRBM

```
interpolate by the radial basis function method
                     fprintf(1,'Progress IRBM: %d%%\n',round(100*
                         idx/size(Xi,1)));
```

56.4 ISparse

```
sparse interpolation class
```

56.5 Inn

```
nearest neighbour interpolation
```

56.6 Interpolator

```
interpolator super-class
                            fprintf(1,'Progress: %f%% %fs\n',100*
                                idx/size(Xt,1),t);
```

56.7 fixnan

```
fill nan-values in vector with gaps
```

56.8 idw1

```
spatial average ny inverse distance weighting
```

56.9 idw2

```
spatial average by inverse distance weighting
```

56.10 inner2outer

```
linear interpolation of segment mit point to grid points at segment
     ends
assumes equal grid spacing
```

56.11 inner2outer2

```
interpolate from element (segment) centres to edge points
```

56.12 interp1_circular

56.13 interp1_limited

```
interpolate values, but not beyond a certain distance
this function is idempotent, i.e. it will not extrapolate over into
     gaps
exceedint the limit and thus not spuriously extend the series when
    called a second time on the same data
```

56.14 interp1_man

```
interpolate
```

56.15 interp1_piecewise_linear

56.16 interp1_save

```
make interpolation save to round off errors
the matlab internal interpolation suffers from rounding errors,
    which
are unacceptable when values of X and Y are large (for example UTm
    coordinates)
this normalization prevents this
```

56.17 interp1_slope

```
quadratic interpolation returning value and derivative(s)
```

56.18 interp1_smooth

56.19 interp1_unique

```
matlab fails to interpolate, when x values are not unique
this function makes the values unique before use
```

56.20 interp2_man

```
nearest neighbour interpolation in two dimensions
```

56.21 interp_angle

```
interpolate an angle
```

56.22 interp_fourier

```
interpolation by the fourier method
```

56.23 interp_fourier_batch

```
batch interpolation by the fourier interpolation
```

56.24 interp_sn

```
interpolate along streamwise coordinates
This gives similar result to setting aspect ratio for sN to
    infinity,
but not quite,as the input point set is not dense (scale for sN to
    infinity does not work)
              sdx  = sdx(sdx_);
```

56.25 interp_sn2

```
interpolation in streamwise coordinates
```

56.26 interp_sn3

56.27 interp_sn_

56.28 limit_by_distance_1d

```
smooth subsequent values along a curve such that
      v(x0+dx) < v(x0) + (ratio-1)*dx
if v is the edge length in a resampled polygon, then v_i/v_(i+1) <
    ratio
      ratio^1 = exp(a*1)
```

56.29 resample1

```
interpolation along a parametric curve with variable step width
```

56.30 resample_d_min

```
resample a function
```

56.31 resample_vector

```
resample a track so that velocity vectors do not run into each
    other
```

56.32 test_interp1_limited

57 numerical-methods

57.1 inverse_complex

57.2 maccormack_step

57.3 minmod

58 numerical-methods/multigrid

58.1 mg_interpolate

58.2 mg_restrict

59 numerical-methods/ode/@BVPS_Characteristic

59.1 BVPS_Characteristic

```
solve coupled first- and second-order 1D boundary-value problems
```

59.2 assemble1_A

```
assemble the discretisation matrix for a first order ode
(mean component, zero frequency)
```

59.3 assemble1_A_Q

```
assemble the discretisation matrix for a first order ode
(mean component, zero frequency)
```

59.4 assemble2_A

```
assemble the discretisation matrix for a second-order ode
(non-zero frequency component)
```

59.5 assemble_AA

```
assemble the discretisation matrix for each channel
iteratively calls assembly for each frequency components
```

59.6 assemble_AAA

```
assemble the discretisation matrix for the entire network
iteratively calls assembly for each channel
```

59.7 assemble_Ic

59.8 bvp1c

59.9 check_arguments

59.10 couple_junctions

59.11 derivative

59.12 init

59.13 inner2outer_bvp2c

59.14 reconstruct

59.15 resample

59.16 solve

```
solve system of non-linear second order odes (in more than one
    variable)
as boundary value problems

odefun provides ode coefficients c:
c(x,1) y''(x) + c(x,2) y'(x) + c(x,3) y = c(x,4)
   c_1 y"    + c_2 y'      + c_3 y + c_4 = c_4

subject to the boundary conditions
bcfun provides v and p and optionally q, so that:

b_1 y + b_2 y' = f
   q(x,1)*( p(x,1) y_l(x) + p(x,2) y_l'(x)
 + q(x,2)*( p(x,1) y_r(x) + p(x,2) y_r'(x) = v(x)
where q weighs the waves travelling from left to right and right to
     left (default [1 1])
```

59.17 test_assemble1_A

59.18 test_assemble2_A

60 numerical-methods/ode/@Time_Stepper

60.1 Time_Stepper

60.2 solve

61 numerical-methods/ode

61.1 bvp2fdm

```
solve system of non-linear second order odes (in more than one
    variable)
as boundary value problems by the finite difference method

odefun provides ode coefficients c:
c(x,1) y''(x) + c(x,2) y'(x) + c(x,3) y = c(x,4)
 c_1 y" + c_2 y' + c_3 y + c_4 = 0

subject to the boundary conditions
bcfun provides v and p and optionally q, so that:

b_1 y + b_2 y' = f
   q(x,1)*( p(x,1) y_l(x) + p(x,2) y_l'(x)
 + q(x,2)*( p(x,1) y_r(x) + p(x,2) y_r'(x) = v(x)
where q weighs the waves travelling from left to right and right to
     left (default [1 1])
```

61.2 bvp2wavetrain

```
solve second order boundary value problem by repeated integration
```

61.3 bvp2wavetwopass

```
two pass solution for the linearised wave equation
solve first for the wave number k, and then for y
```

61.4 ivp_euler_forward

```
solve intial value problem by the euler forward method
```

61.5 ivp_euler_forward2

61.6 ivprk2

```
solve initial value problem by the two step runge kutta method
```

61.7 ode2_matrix

```
transformation matrix of second order ode
to left and right going wave

c = odefun(x)
c1 y'' + c2' y + c3 y == 0
y = y_p + y_m, left and right going wave
d/dx [y_p, y_m] = A*[y_m, y_p]
```

61.8 ode2characteristic

```
second order odes
transmittded and reflected wave
```

61.9 step_trapezoidal

```
single trapezoidal step
```

61.10 test_bvp2

62 numerical-methods/optimisation

62.1 aitken_iteration

62.2 anderson_iteration

62.3 armijo_stopping_criterion

```
armijo stopping criterion for optimizations
```

62.4 astar

```
astar path finding alforithm
```

62.5 binsearch

```
binary search on a line
```

62.6 bisection

```
bisection
```

62.7 box1

```
test objective function for optimisation routines
```

62.8 box2

62.9 cauchy

62.10 cauchy2

```
solve non-linear system by cuachy's method
slower than quadratic optimisation, but does not require a hessian
fun : objective function, returns
      f  : scalar, objective function value
      g  : nx1, gradient
x   : nx1, initial position
opt : options
```

62.11 directional_derivative

```
directional (projected) derivative
d : derivative, highest first
p : series expansion around x0
```

62.12 dud

```
optimization by the dud algorithm
```

62.13 extreme3

```
extract maxima by quadratic approximation from sampled function val
    (t)
intended to be called after [mval, mid] = max(val) for refinement
    of
locatian and maximum

input
t    : sampling time (uniformly spaced)
v    : values at sampling times
ouput:
tdx  : index where extremum should be computed
t0   : location of the extremum
val0 : value of extremum

v'(dt0) = 0 and v''(dt0) determines type of extremum
```

62.14 extreme_quadratic

62.15 ftest

62.16 fzero_bisect

62.17 fzero_newton

62.18 grad

```
numerical gradient
```

62.19 hessian

```
numerical hessian
```

62.20 hessian_from_gradient

```
numerical hessian from gradient
```

62.21 hessian_projected

```
numerical hessian projected to one dimenstion
```

62.22 line_search

```
bisection routine
```

62.23 line_search2

```
bisection method

fun : objective funct
x0  : start value
f0  : objective function value at x0
g   : gradient at x0
p   : search direction from x0 (p = g for steepest descend)
h   : initial step length (default 1)
lb  : lower bound for x
up  : upper bound for x
```

62.24 line_search_polynomial

```
polynomial line search
fun : objective funct
x0  : start value
f0  : objective function value at x0
g   : gradient at x0
dir : search direction from x0 (p = g for steepest descend)
h   : initial step length (default 1)
lb  : lower bound for x
up  : upper bound for x
```

62.25 line_search_polynomial2

```
cubic line search
fun : objective funct
x0  : start value
f0  : objective function value at x0
g   : gradient at x0
dir : search direction from x0 (p = g for steepest descend)
h   : initial step length (default 1)
lb  : lower bound for x
up  : upper bound for x
```

62.26 line_search_quadratic

```
quadratic line search
fun : objective funct
x0  : start value
f0  : objective function value at x0
g   : gradient at x0
dir : search direction from x0 (p = g for steepest descend)
h   : initial step length (default 1)
lb  : lower bound for x
up  : upper bound for x
```

62.27 line_search_quadratic2

```
quadratic line search
```

62.28 line_search_wolfe

```
line search by wolfe method
c.f.: OPTIMIZATION THEORY AND METHODS - Nonlinear Programming, Sun,
     Yuan
```

62.29 ls_bgfs

```
least squares by the bgfs method
```

62.30 ls_broyden

```
least squares by the broyden method
for rectangular / non symmetric systems
      Numerical Optimization nocedal
      Practical Methods of  Optimization fletcher
c.f. gerber 1981
c.f. fletcher 1978 (more advanced, not used here)
c.f. Kelley 1999 ch. 4

BGFS:
Broyden 1965
Fletcher 1970
Goldfarb 1970
Shanno 1970
```

62.31 ls_generalized_secant

```
least squares by the secant method
Barnes, 1965
Wolfe, 1959
Fletcher 1980, 6.3
seber 2003
gerber
```

62.32 nlcg

```
non-linear conjugate gradient
input:
x   : nx1 start vectort
opt : struct options
fdx : gradient constraint
```

62.33 nlls

```
non-linear least squares
```

62.34 picard

```
picard iteration
```

62.35 poly_extrema

`extrema of a polynomial`

62.36 quadratic_function

`evaluate quadratic function in higher dimensions`

62.37 quadratic_programming

`optimize by quadratic programming`

62.38 quadratic_step

`single step of the quadratic programming`

62.39 rosenbrock

`rosenbrock test function`

62.40 sqrt_heron

`Heron's method for the square root`

62.41 test_directional_derivative

62.42 test_dud

62.43 test_fzero_newton

62.44 test_line_search_quadratic2

62.45 test_ls_generalized_secant

62.46 test_nlcg_6_order

62.47 test_nlls

```
f = w'*(p*abs(x-1).^4) + w'*(1-p)*abs(x-1).^2;
```

63 numerical-methods/pde

63.1 laplacian2d_fundamental_solution

64 numerical-methods/piecewise-polynomials

64.1 Hermite1

```
hermite polynomial interpolation in 1d
```

64.2 hp2_fit

```
fit a hermite polynomial
coefficients are derivative free
x0  : left point of first segment
x1  : right point of last segment
n   : number of segments
x   : sample x-value
val : sample y-value
c   : coefficients (values at points, no derivatives)
```

64.3 hp2_predict

```
prediction with pw hermite polynomial
c are values at support points
```

64.4 hp_predict

```
predict with piecewise hermite polynomial
```

64.5 hp_regress

```
fit piecewise hermite polynomial
coefficients are values and derivatives
```

64.6 lp_count

```
lagrangian basis for interpolation
count number of valid samples
```

64.7 lp_predict

```
lagrangian basis piecwie interpolation, predicor
```

64.8 lp_regress

64.9 lp_regress_

65 numerical-methods

65.1 step_advect_euler_explicit

65.2 step_advection_diffusion_euler_implicit

65.3 step_advection_diffusion_trapezoidal

65.4 step_diffuse_analytic

```
analytic solution to the heat equation
the spectral solution is not positivity preserving as it results in
spurious oscillations, this is avoided here, by integrating over
    segments
rather than sampling at gridpoints
```

65.5 step_diffuse_euler_explicit

65.6 step_diffuse_euler_implicit

65.7 step_diffuse_spectral

65.8 step_diffuse_trapezoidal

65.9 step_react_euler_explicit

65.10 step_react_euler_implicit

65.11 step_react_midpoint

65.12 step_react_ralston

65.13 step_react_ralston_exp

65.14 step_react_ralston_exp_2

65.15 step_react_semi_analytic

65.16 step_react_trapezoidal

65.17 test_adams_bashforth

66 mathematics

`mathematical functions of various kind`

66.1 oversampleNZ

67 pdes

67.1 heat_equation_fundamental_solution

67.2 heat_equation_fundamental_std_to_time

67.3 heat_equation_std

67.4 heat_equation_width

67.5 heat_equation_width_to_time

68 regression/@PolyOLS

68.1 PolyOLS

```
class for polynomial least squares
```

68.2 coefftest

68.3 detrend

```
detrending by polynomial regression
```

68.4 fit

```
fit a polynomial function
like polyfit, but returns parameter error estimates
TODO automatically activate scaleflag
```

68.5 fit_

```
fit a polynomial function
```

68.6 predict

```
predict polynomial function values
```

68.7 predict_

68.8 slope

```
slope by linear regression
```

69 regression/@PowerLS

69.1 PowerLS

```
class for power law regression
```

69.2 fit

```
fit a power law
like polyfit, but returns parameter error estimates
```

69.3 predict

```
predict with power law
      S2 = diag((A*obj.C)*A');
      L  = Y - S;
      U  = Y + S;
```

69.4 predict_

70 regression/@Theil

70.1 Theil

```
Kendal-Theil-Sen robust regression
```

70.2 detrend

```
linear detrending of a set of samples by the Theil-Senn Slope
```

70.3 fit

```
fit slope and intercept to a set of sample with the Theil-Sen
    method

c     : confidence interval c = 2*ns*normcdf(1) for ns-sigma
    intervals
param : itercept and slope
P : confidence interval
```

70.4 predict

```
predict values and confidence intervals with the Theil-Sen method
```

70.5 slope

```
fit the slope with the Theil-Sen method
```

71 regression

```
linear and non-linear regression
```

71.1 Theil_Multivariate

```
extension of the Theil-Senn regression to higher dimensions by
means of the Gauss-Seidel iteration
```

71.2 areg

```
regression using the pth-fraction of samples with smallest residual
```

71.3 ginireg

```
gini regression
```

71.4 hesssimplereg

```
hessian, gradient and objective function value of the simple
    regression
rhs = p(1) + p(2) x + eps
```

71.5 l1lin

```
solve ||Ax - b||_L1 by means of linear programming
```

71.6 lsq_sparam

```
parameter covariance of the least squares regression

fun : model function for predtiction
b   : sample values
f(p) = b
p   : parameter at point of evaluation (preferably optimum)
```

71.7 polyfitd

```
fit a polynomial of order n to a set of sampled values and sampled
    values
of the derivative

x0 must contain at least for conditioning as otherwise the
    intercept
cannot be determined
```

71.8 regression_method_of_moments

```
fit linear function ||a b x = y||_L2 by the method of moments
y+eps = alpha + beta*x
```

71.9 robustlinreg

```
fit a linear function by splitting the x-values at their median
      (med(y_left) - med(y_right))/(med(x_left)-med(x_right)
this approach performs poorly compared to the theil-senn operator
```

71.10 theil2

```
Theil senn-estimator for two dimensions (glm)
```

71.11 theil_generalised

```
generalization of the Theil-Senn operator to higher dimensions,
for arbitrary functions such as polynomials and multivariate
    regression
either higher order polynomials or glm
c.f. "On theil's fitting method", Pegoraro, 1991
```

71.12 total_least_squares

```
total least squares
```

71.13 weighted_median_regression

```
weighted median regression
c.f. Scholz, 1978
```

72 set-theory

72.1 issubset

```
test if set B is subset of A in O(n)-runtime

A : first set
B : second set
P : set of primes (auxiliary)
```

73 mathematics

```
mathematical functions of various kind
```

73.1 shuffle_index

74 signal-processing

74.1 asymwin

```
creates asymmetrical filter windows
filter will always have negative weights
```

75 signal-processing/autocorrelation

75.1 acf_radial

75.2 acfar1

```
Autocorrelation function of the finite AR1 process

a_k = 1/(n-k)sum x_ix_i+1 + (xi + xi+k)mu + mu^2
    = r^k + 1/n sum_ij + 1/n
       pause
```

75.3 acfar1_2

```
autocorrelation of the ar1 process
```

75.4 acfar2

```
impulse response of the ar2 process
```

75.5 acfar2_2

```
autocorrelation of the ar2 process
X_i + a1 X_i-1 + a2 X_i-2 = 0
```

75.6 ar1_cutoff_frequency

75.7 ar1_effective_sample_size

```
effective sample size correction for autocorrelated series
```

75.8 ar1_mse_mu_single_sample

```
standard error of a single sample of an ar1 correlated process
```

75.9 ar1_mse_pop

```
variance of the population mean of a single realisation around zero

E[(mu_N-0)^2] = E[mu_N^]
```

75.10 ar1_mse_range

```
mean standard error of the mean of a range of values taken from an
    ar1 process
```

75.11 ar1_spectrum

```
spectrum of the ar1 process
```

75.12 ar1_to_tikhonov

```
convert ar1 correlation to tikhonovs lambda
```

75.13 ar1_var_factor

```
variance correction factor for an autocorrelated finite process
n   : [1 .. inf] population size
m   : [1 .. n]  samples size
rho : [ -1 < rho < 1 (for convergence) ] correlation of samples
```

75.14 ar1_var_factor_

```
variance of an autocorrelated finite process
```

75.15 ar1_var_range2

```
variance of sub sample starting at the end of the series

from the finite length first order autocorrelated process

s2 = 1/m^2 sum_i^m sum_j^m rho^-|i-j|
```

75.16 ar1delay

```
approximate acf by the ar1 process
acf: autocovariance or autocorrerlation function
nf : skip first samples (for mixed geometric-arithmetic series (
    ARMA)
```

75.17 ar1delay_old

```
autocorrelation of the residual
```

75.18 ar2_acf2c

```
determine coefficients of the ar2 process from the first two lags
    of the
autocorrelation function
```

75.19 ar2conv

```
coefficients of the ar2 process determined from the two leading
    correlations
of the acf [1,r1,r2,...]
```

75.20 ar2dof

```
effective samples size for the ar2 process
```

75.21 ar2param

```
ar2 parameter estimation from first two terms of acf
acf = [1 a1 a2 ...]
```

75.22 autocorr2

75.23 autocorr_angular

75.24 autocorr_bandpass

75.25 autocorr_decay_rate

```
estimate exponential decay of the autocorrelation
```

75.26 autocorr_effective_sample_size

```
effective sample size from acf
```

75.27 autocorr_fft

```
estimate sample autocorrelation function
```

75.28 autocorr_forest

75.29 autocorr_genton

```
autocorrelation function
```

75.30 autocorr_highpass

75.31 autocorr_lowpass

75.32 autocorr_periodic_additive_noise

75.33 autocorr_periodic_windowed

75.34 autocorr_radial

75.35 autocorr_radial_hexagonal_pattern

75.36 autocorrelation_max

76 signal-processing

76.1 average_wave_shape

```
extract waves with varying length from a wave train and and average
     their shape
```

76.2 bandpass

```
bandpass filter
```

76.3 bandpass_continuous_cdf

76.4 bartlett

```
Effective sample size factor for bartlett window
c.f. thiebaux
c.f spectral analysis-jenkins, eq. (6.3.27)
c = acf
note: results seams always to be 1 tac too low
T : reduction factor for dof
for ar1 with a = rho^k = exp(-k/L), T = 2L
```

76.5 bin1d

```
bin values of v sampled at x into bins bounded by "edges"
apply function v to it
```

76.6 bin2d

```
bin values of V sampled at X and Y into the grid structured grid ex
    ,ey
apply function func to all walues in the bin
func = mean : default
func = sum : non-normalized frequency histogram in 2D
```

76.7 binormrnd

```
generate two correlated normally distributed vectors
```

76.8 coherence

76.9 conv1_man

```
convolutions with padding
```

76.10 conv2_man

```
convolution in 2d
```

76.11 conv2z

76.12 conv30

```
convolve with rectangular window of lenght n
circular boundaries
```

76.13 conv_

```
convolution of a with b
```

76.14 conv_centered

```
convolve x with filter window f
when length of f is even, this guarantees a symmetric result (no
    off by on
displacement) by making the lenght of f odd at first
```

76.15 convz

76.16 cosexpdelay

76.17 csmooth

```
smooth recursively with [1,2,1]/4 kernel
function x = csmooth(x,n,p,circ)
```

76.18 daniell_window

```
Daniell window for smoothing the power spectrum
c.f. Daniell 1946
Bloomfield 2000
meko 2015
```

76.19 db2neper

```
convert decibel to neper
```

76.20 db2power

```
power ratio from db
```

76.21 derive_bandpass_continuous_scale

76.22 derive_danielle_weight

76.23 derive_limit_0_acfar

76.24 detect_peak

```
detect peaks in a vector
requires function value to fall to p*max before new value is
    allowed
```

76.25 determine_phase_shift

76.26 determine_phase_shift1

```
average phase and phase shift per time step of a train of waves
```

76.27 doublesum_ij

```
double sum of r^i
```

76.28 effective_mask_size

76.29 effective_sample_size_to_ar1

```
convert effective sample size to ar1 correlation
```

76.30 fcut2Lw_gausswin

76.31 fcut_gausswin

76.32 filt_hodges_lehman

77 signal-processing/filters

77.1 circfilt2

```
smooth (filter) the 2D image z with a circular disk of radius nf
apply periodic boundary conditions
```

77.2 filter1

```
filter along one dimension
```

77.3 filter2

```
filter columns of x (matlab does only support vector input)
```

77.4 filter_

```
invalidate values that exceed n-times the robust standard deviation
```

77.5 filter_r_to_f0

77.6 filter_rho_to_f0

77.7 filter_twosided

77.8 filteriir

```
filter adcp t-n data over time

v : nz,nt  : values to be filtered
H : nt,1   : depth of ensemble
last : nt,1 : last bin above bottom that can be sampled without
    side lobe interference
nf : scalar : number of reweighted iterations

when samples
- distance to bed is reference (advantageous for near-bed suspended
     transport)
TODO for wash load: distance to surface is more relevant
```

```
interpolate depending on z

when depth changes, neighbouring indices do not correspond to same
    relative position in the water column
relative poisition in the colum (s-coordinate) smoothes values
near the bed: absolute distance to bed is chosen
near surface: absolute distance to surface is chosen
-> cubic transformation of index

faster and avoid alising (smoothing along z)
      resample ensemble to same number of bins in S -> filter ->
          resample back
      use nonlinear transform z-s coordinates
-> resampling has to be local (Hi -> H-filtered)

filtered profile coordinates to sample coordinates
      zf -> zi (special transform)
corresponding indices and fractions
filtration step (update of hf and vf)
sample coordinates to updated profile coordinates
(the inverse step is actually not necessary)
write filtered value
```

77.9 filterp

77.10 filterp1

```
fir filter with some fancy extras
```

77.11 filterstd

77.12 gaussfilt2

```
smooth (filter) the 2D image z with a gaussian window
apply periodic boundary conditions
```

77.13 lowpass_discrete

```
design coefficients of a low pass filter with specified cut of
    frequency
and sampling period
alalogue low pass with pole at s=-omega_c=1/tau=1/RC
Ha = tau/(tau + s) = 1/(1 + omega_c*s)
```

77.14 meanfilt2

```
filter with a rectangular window along both dimensions
```

77.15 medfilt1_man

```
moving median filter, supports columnwise operation
```

77.16 medfilt1_man2

```
moving median filter with special treatment of boundaries
```

77.17 medfilt1_padded

```
median filter with padding
```

77.18 medfilt1_reduced

```
median filter with padding
```

77.19 trifilt1

```
filter with triangular window
trifilt1 is ident to twice applying rectfilt1 (meanfilt1) with half
     the domain size
note : inifnitely many convolution yield a gaussian
```

77.20 trifilt2

filter with a triangular window along both dimensions

78 signal-processing

78.1 firls_man

design finite impulse response filter by the least squares method

78.2 fit_spectral_density

fit spectral densities (probability distributions)

78.3 fit_spectral_density_2d

fit spectral densities

78.4 fit_spectral_density_radial

fit spectral densities

78.5 flattopwin

the flat top window

78.6 frequency_response_boxcar

frquency response of a boxcar filter

78.7 freqz_boxcar

frequncy response of a boxcar filter

78.8 gaussfilt1

```
filter data series with a gaussian window, assumes periodic bc
```

78.9 hanchangewin

```
hanning window for change point detection
```

78.10 hanchangewin2

```
nanning window for chage point detection
```

78.11 hanwin

```
hanning filter window
```

78.12 hanwin_

```
hanning filter window
```

78.13 high_pass_1d_simple

78.14 kaiserwin

```
kaiser filter window
```

78.15 kalman

```
Kalman filter
```

78.16 lanczoswin

```
Lanczos window
```

78.17 last

```
lake tail, but for matrices
```

78.18 maxfilt1

78.19 meanfilt1

```
moving average filter with special treatment of the boundaries
```

78.20 mid_term_single_sample

```
variance of single sample, mid term
```

78.21 minfilt1

78.22 minmax

78.23 mu2ar1

```
error variance of the mean of the finite length ar1 process
```

```
(mu)^2 = (sum epsi)^2 = sum_i sum_j eps_i eps_j = sum_ii(rho,n)/n^2
this has the limit s^2 for rho->1
```

78.24 mysmooth

78.25 nanautocorr

```
autocorrelation with nan-values
```

78.26 nanmedfilt1

```
medfilt1, skipping nans
```

78.27 neper2db

```
convert neper to db
```

78.28 oscillator_noisy

79 signal-processing/passes

79.1 bandpass1d

79.2 bandpass1d_fft

```
filter input vector with a spatial (two-sided) bandpass in fourier
    space
```

79.3 bandpass1d_implicit

79.4 bandpass2

```
bandpass filter
```

79.5 bandpass2d

79.6 bandpass2d_convolution

79.7 bandpass2d_fft

79.8 bandpass2d_ideal

79.9 bandpass2d_implicit

```
bandpass filter the surface x by solving the implicit relation:
```

79.10 bandpass2d_iso

79.11 bandpass_arg

```
determine correlation coefficient from frequency of mode for the
    symmetric
```

79.12 bandpass_f0_to_rho

```
correlation coefficient for the pth-order symmetric bandpass filter
     with
maximum at f0 (when rho_lp = rho_hp)
```

79.13 bandpass_max

79.14 bandpass_max2

79.15 highpass

```
high pass filter
```

79.16 highpass1d_fft_cos

```
filter the input vector with a cosine-shaped highpass in frequency
    space
```

79.17 highpass1d_implicit

79.18 highpass2d_fft

79.19 highpass2d_ideal

79.20 highpass2d_implicit

79.21 highpass_arg

79.22 highpass_fc_to_rho

79.23 lowpass

```
low pass filter
```

79.24 lowpass1d_fft

79.25 lowpass1d_implicit

79.26 lowpass2

```
design low pass filter with cutoff-frequency f1
```

79.27 lowpass2d_anisotropic

79.28 lowpass2d_convolution

```
this function is computationally inefficient and serves merely for
    illustration
and tests
```

79.29 lowpass2d_fft

```
note : this function is for testing purposes only,
      directly multiply the ft of the signal with the ft of the
          filter
      to obtain the filtered signal in a single step
function y = lowpass2d_fft(x,rho,a,order)
```

79.30 lowpass2d_ideal

```
lowpass filter the input x in the Frequency Domain

TODO no need to provide dx, follows from size of x
function [y,S,R,r]=lowpass2d_ideal(x,L,dx,varargin)
```

79.31 lowpass2d_implicit

```
function [y] = lowpass2d_implicit(x,rho,a,order,direct)
```

79.32 lowpass_arg

79.33 lowpass_fc_to_rho

79.34 lowpass_iir

```
iir-low pass
```

79.35 lowpass_iir_symmetric

```
two-sided iir low pass filter (for symmetry)
```

79.36 lowpassfilter2

```
low-pass filter of data
```

80 signal-processing

80.1 peaks_man

```
peaks of a periodogram
```

81 signal-processing/periodogram

81.1 periodogram

```
compute the normalized periodogram
```

81.2 periodogram_2d

```
compute the normalized periodogram in two dimensions
```

81.3 periodogram_align

81.4 periodogram_angular

```
input:
      Sxy : nxn
ouput:
      Sra : n/2*(pi*n/2)
      angle
function [Sa,angle,A] = periodogram_angular(Sxy,L,nf)
```

81.5 periodogram_bartlett

```
estimate the spectral density nonparametrically with Bartlett's
    method
```

81.6 periodogram_bootstrap

81.7 periodogram_confidence_interval

```
confidence interval for periodogram values
```

81.8 periodogram_filter

81.9 periodogram_median

81.10 periodogram_normalize

81.11 periodogram_normalize_2d

81.12 periodogram_p_value

81.13 periodogram_qq

```
 qq-plot of a spectral density estimate by smoothing against the
     expected
 beta-density
```

81.14 periodogram_quantiles

```
 quantiles of a periodogram
```

81.15 periodogram_radial

```
function [Sr,fri,se,count] = periodogram_radial(S2d,L)

 compute the radially averaged density

 input:
 S2d : 2-dimensional density or periodogram
```

```
L =[Lx,Ly] : domain length

output:

S_r.mu        : radially averaged periodogram
S_r.normalized : normalized radially averaged periodogram
A             : matris operator s that Sr = (A*A')^-1 A'*S2d

f_r : radial freqeuncies, at which radial periodogram is determined
      discretized in same interval as the 2d-density : f = 1/L

Definitions:
      radial wavenumber, identical to circumferences of circles
          centred at origin with radial frequency fr
             k_r = 2*pi*f_r

       radially averaged periodogram:
      S_r(k_r) = 1/k_r int_0^{k_r} S2d(k_r,s) d s
                = 1/(2 pi) int_0^{2 pi} S2d(k_r,theta) d theta
                ~ 1/(2 pi) sum^nt S2d(k_r,theta) * (2*pi/nt)
                ~ 1/nt sum^nt S2d(k_r,theta)

             nt ~ k_r/df = k_r*L
normalization:
     S_r.normalize = S_r/int_0^inf S_r dfr
                   ~ S_r/(sum_0^nr S_r Delta fr)

note : the radially averaged "periodogram", is actually a density
    estimate,
      for radial frequencies fr hat are not small

when S is flattened into a vector, the isotropic part of the 2D
    density can be recovered with:
S_iso    = (A*S_radial)
S_radial = A^-1 S_hat
```

81.16 periodogram_std

```
standard deviation of a periodogram
```

81.17 periodogram_test_periodicity

```
test a periodogram for hidden periodic frequency components
```

```
function [p,ratio,maxShat,mdx,fdx,S] = periodogram_test_periodicity
    (fx,Shat,nf,fmin,fmax,S,mode)

input:
      fx : frequengcies
      Shat : corresponding periodogram values
    nf   : number of bins to test for periodicity, ignored when S
        is given
      fmin, fmax : frequency range limits to test
      S    : exact (a priori known theoretical spectral density,
          must not be estimated from the periodogram)
      mode : automatically set to "exact", when S given
            inclusive : estimate density by smoothing including the
                 central bin
            exclusive : estimate density by smoothing excluding the
                 central bin
      note: inclusive and exclusive lead to different distribution
           but identical p-values

TODO pass L and not fx
```

81.18 periodogram_test_periodicity_2d

```
test a periodogram for hidden periodic frequency components

[pn,stat,out] = periodogram_test_periodicity_2d(b, L, nf, bmsk,
    fmsk, ns, significance_level)

input:
      b    (nx * ny): image to test for presence of hidden
          periodicities,
           i.e. periodicities where the frequency is not known a
               priori
      L    : domain size in arbitrary units, default is n
            only effects scaling of complementary outout Shat and
                Sbar
            does not effect test as it cancels out in the tested
                ratio Shat/Sbar
    nf   : nfr or [nfx, nfy]
            radius of circular disk (in number of bins) used for
                smoothing
           the periodogram to estimate the spectral density,
            or axes of ellipses for smoothing
            when b is not square a good choice is nfx/nfy ~ Lx/Ly
    bmsk : mask in real space selecting parts of the image to
        include in
            the analysis. default is the entire image
```

```
        the mask can have non-integer values to feather the
            borders of the mask
  fmsk : mask in frequency selecting frequencies to test for
      periodicity
        default is all frequencies
        note: when b is real, one half plane can always be
            excluded
        because of symmetry. This slightly increases the
            significance
  n_mc : number of samples for the monte-carlo determination of
        the test statistics, mc is only used when parts of the
            image are masked
        otherwise the analytic test statistic is used
  siginificance_level :

output :
  pn   : p-value of largest frequency component with largest
      ratio Shat/Sbar
       when testing all frequency components selected by fmsk
  stat.max.ratio   : max ratio value of Shat/Sbar
  stat.max.Shat    : periodigoram value of frequency component
      with max ratio
  stat.max.Shat_rel : spectral energy contained frequency
      component with max ratio
  stat.max.fx      : x-component of frequency at max ratio
  stat.max.fy      : y-component of frequency of max ratio
  stat.intShat_sig : spectral energy contained in all
      significant frequency bins
  stat.p1          : p-value of all frequency components
  stat.pn          : p-value of all frequency components,
      corrected for multiple comparisons

influence of masking the input file:
           - the root-mean-square energy of the ordinates is
               proportional
             to the number of unmasked points
           - values in the periodogram are not any more linearly
               independent
             so that the dof of the filter window is not nf^2
```

81.19 periodogram_test_stationarity

```
test a periodogram for statoinarity
note : the method works, but is of little practical use,
as it requires about 50 periods and a small dx to detect a
    frequency change by a factor of 2
```

81.20 periodogram_welsh

82 signal-processing

82.1 polyfilt1

```
polynomial filter,
can be achieved by iteratively processing the data with
a mean (zero-order) filter
```

82.2 qmedfilt1

```
medfilt1, after fitting a quadratic polynomial
```

82.3 quadratfilt1

82.4 quadratwin

82.5 randar1

```
generate random ar1 process
e1 = randar1(sigma,p,n,m)
```

82.6 randar1_dual

```
draw random variables of two corrlated ar1 processes
```

82.7 randar2

```
generate ar2 process
```

82.8 randarp

```
randomly generate the instance of an ar-p process
```

82.9 rectwin

```
rectangular window
```

82.10 recursive_sum

82.11 select_range

82.12 smooth1d_parametric

```
smooth position of p0=x0,y0 between p1=x1,y1 and p2=x2,y2,
so that distance to p1 and p2 becomes equal
and the chord length remains the same
```

82.13 smooth2

```
smooth vectos of X
```

82.14 smooth_man

82.15 smooth_parametric

```
smooth a parametric function given in x-y coordinates
      matvec2x2(R,[dxc;dyc])
```

82.16 smooth_parametric2

```
parametrically smooth the curve
```

82.17 smooth_with_splines

82.18 smoothfft

```
filter with fast fourier transform
```

83 signal-processing/spectral-density

83.1 hex_angular_pdf

83.2 hex_angular_pdf_max

83.3 hex_angular_pdf_max2par

83.4 spectral_density_ar2

83.5 spectral_density_area

```
integrate the spectral density over the positive half axis
```

83.6 spectral_density_estimate_2d

83.7 spectral_density_flat

```
flat spectral density of a random vector woth iid elements
```

83.8 spectral_density_forest

83.9 spectral_density_gausswin

83.10 spectral_density_lorentzian

```
lorentzian spectral density
```

83.11 spectral_density_lorentzian_max

```
mode (maximum) of the lorentzian spectral density
```

83.12 spectral_density_lorentzian_max2par

```
transform maximum of the lorentzian spectral density to its
    distribution parameters
```

83.13 spectral_density_lorentzian_scale

```
normalization scale of the lorentzian spectral density
```

83.14 spectral_density_maximum_bias_corrected

83.15 spectral_density_periodic_additive_noise

83.16 spectral_density_rectwin

83.17 spectral_density_wperiodic

84 signal-processing

84.1 spectrogram

```
spectrogram
```

84.2 sum_i_lag

```
sum of ar1 matrix with lag
sum_i=1^n rho^|i-k|
```

84.3 sum_ii

```
sum of ar1 matrix
sum_i=1^n sum_j=1^n rho^|i-j|
this is for the variance, take square root for the standard
    deviation factor
```

84.4 sum_ii_

84.5 sum_ij

```
sum of ar1 matrix
sum_{i=1}^n sum_{j=1}^m r^|i-j|
```

84.6 sum_ij_

84.7 sum_ij_partial_

84.8 sum_multivar

```
sum of matrix entries of bivariate ar1 process
```

84.9 test_acfar1

84.10 tikhonov_to_ar1

```
convert coefficient of the tikhonov regularization to correlatioon
    of the ar1 process
```

84.11 trapwin

```
trapezoidal filter window
```

84.12 triwin

```
triangular filter window
```

84.13 triwin2

```
triangular filter window
```

84.14 tukeywin_man

84.15 varar1

```
error variance of a single sample of a finite length ar1 process
with respect to the mean, averaged over the population
```

84.16 welch_spectrogram

```
welch spectrogram
```

84.17 wfilt

```
filter with window
```

84.18 winbandpass

```
filter with bandpass
```

85 signal-processing/windows

85.1 circwin

85.2 danielle_window

```
danielle fourier window
```

85.3 gausswin

85.4 gausswin1

85.5 gausswin2

85.6 radial_window

```
radial filter window in the 2d-frequency domain
```

85.7 range_window

```
range of values within a certain range of indices (window)
```

85.8 rectwin_cutoff_frequency

85.9 std_window

```
moving block standard deviation
```

85.10 window2d

85.11 window_make_odd

86 signal-processing

86.1 winfilt0

```
filter with window
```

86.2 winlength

```
window length for desired cutoff frequency
power at fc is halved
H(wf) = 1/sqrt(2) H(f)
if the filter window were used as a low pass filter
note: the user should prefer a windowed ideal low pass filter
TODO, relate this to DOF
```

86.3 wmeanfilt

```
mean filter with window
```

86.4 wmedfilt

```
median filter with window
```

86.5 wordfilt

```
weighted order filter
```

86.6 wordfilt_edgeworth

```
weighed order filter
```

86.7 wrapphase

86.8 xar1

86.9 xcorr_man

```
cross correlation of two sampled ar1 processes
```

87 sorting

87.1 sort2

```
sort two numbers
```

87.2 sort2d

```
sort elements of matrix in X
returns row and column index of sorted values
```

88 spatial-pattern-analysis/@Spatial_Pattern

88.1 Spatial_Pattern

```
class for analysis of remotely sensed and model generated
    vegetation patterns
```

88.2 analyze_grid

```
analyze a 2D spatial pattern, estimate regularity and test for
    periodicity
```

88.3 analyze_transect

```
analyze 1D transect through a spatial pattern,
either remotely sensed or model generated
```

88.4 clear_1d_properties

88.5 clear_2d_properties

88.6 fit_parametric_densities

fit parametric spectral densities to the empirical density

88.7 imread

read an image file containing a pattern, mask and geospatial data

88.8 init

88.9 plot

plot the pattern or densities

88.10 plot_transect

plot 1D pattern

88.11 prepare_analysis

88.12 report

report statistics of analysis

88.13 resample_functions

resample empirical densities to a comman grid

88.14 tabulate

```
summarize properties of multiple patterns in a single struct
```

89 spatial-pattern-analysis/@Spatial_Pattern_Array

89.1 Spatial_Pattern_Array

```
container class for Spatial_Pattern objects
```

89.2 analyze

```
analyze spatial patterns
```

89.3 assign_regions

89.4 export_shp

89.5 fetch

```
determine the sampling interval for fetching images from the Google
     satellite
server and later processing
```

89.6 generate_filename

89.7 quality_check

90 spatial-pattern-analysis

90.1 approximate_ratio_distribution

```
input :
bmsk : region selected for periodicity test (smoothes the
    periodogram)
nf   : radius of smoothing window (in bins) for estimating the
    spectral density
nsample : number of repetitions to estimate the ratio distribution
         recommended at

output:
      pr    : probabilities for quantiles
      qr1   : quantiles of the distribution for bin m
      qrn   : quantiles of the distribution for the maximum of bins
           selected by fmsk
      ratio : ratios for each frequency bin and iteration (only for
           last block, for testing)
intput:
      bmsk : mask region pattern/interest in the real domain
      nf   : smoothing window radius in the frequency domain for
          density estimation
      ns   : number of samples for the monte-carlo simulation
      fmsk : mask frequencies of interest
      mdx  : selection of an a-priori known frequency bin

note the following complications:
     - problem 1 : ratio locally differs near fr=0, fx,fy=fmax and
         fx,fy=fmax/2
      - fits of the fisher or beta distribution are highly unstable
```

90.2 banded_pattern

90.3 hexagonal_pattern

```
function [z, x, y, xx, yy, xe, ye] = hexagonal_pattern(fc,n,L,
    angle0_rad,scale,sbm,p,q)

spot pattern of unit amplitude
output : z : pattern
         x : x-coordinate
         y : y-coordinate
```

```
Note : z_gap = 1 - z_spot
```

90.4 patch_size_1d

90.5 patch_size_2d

90.6 pattern_isotropic_rotated

90.7 reconstruct_isotropic_density

90.8 separate_isotropic_from_anisotropic_density

90.9 suppress_low_frequency_lobe

91 spatial-statistics

91.1 cov_cell_averages_1d

```
determine covariance between grid cell averged values of a
    stationary
stochastic process on an equispaced grid

cov(e_ij,e_kl) = E[ (e_ij - mu)*(e_kl - mu) ]
             = 1/dx^2 E[ int (e(x1) - mu) dx1 int (e(x2)-mu) dx2 ]
             = 1/dx^2 E[ int int (e(x1) - mu) (e(x2) - mu) dx1 dx2
                 ) ]
             = 1/dx^2 int int cov(x2-x1) dx1 dx2
```

```
f_ij = int_(x_i - dx/2)^(x_i+dx/2) f(x) dx

integrals approximated by Gauss' method
```

91.2 cov_cell_averages_2d

```
determine covariance between grid cell averged values of a
    stationary
stochastic process on an equispaced grid

cov(e_ij,e_kl) = E[ (e_ij - mu)*(e_kl - mu) ]
             = 1/dx^2 1/dy^2 E[ int int (e(x1,y1) - mu) dx1 dy1 int
                  int (e(x2,y2)-mu) dx2 dy2 ]
             = 1/dx^2 1/dy^2 E[ int int int int (e(x1,y1) - mu) (e(
                 x2,y2) - mu) dx1 dy1 dx2 dy2) ]
             = 1/dx^2 1/dy^2 int int int int cov(x2-x1,y2-y1) dx1
                 dy1 dx2 dy2

f_ij = int_(x_i - dx/2)^(x_i+dx/2) int_(y_i-dy/2)^(y_j+dy/2) f(x,y)
     dx dy

integrals approximated by equal spaced mid-point intervals,
this allows to reduce the double-integral along each dimension into
     a
single integral and hence to reduce the computational effort from m
    ^4 to m^2
```

92 special-functions

92.1 bessel_sphere

```
spherical Bessel function of the first kind
```

92.2 besseliln_large_x

92.3 beta_man

92.4 betainc_man

92.5 digamma_man

92.6 exp10

92.7 hankel_sphere

```
spherical Hankel function for the far field (incident plane wave)
first kind
```

92.8 hermite

```
probabilistic's hermite polynomial by recurrence relation

input :
n : order
x : value

output:
f : H_n(x)
df : d/dx H_n(x)
```

92.9 laguerre_roots

92.10 lambertw_numeric

```
lambert-w function
```

92.11 legendre_man

```
legendre polynomials
```

92.12 neumann_sphere

```
spherical Neumann function
Bessel function of the second kind
```

93 statistics

93.1 atan_s2

```
stadard deviation of the arcus tangens by means of taylor expansion
```

93.2 binomial

```
generalized binomial coefficient, working for non-integer arguments
    ,
in contrast to the matlab buildin function nchoosek
```

94 statistics/circular

94.1 circular_fmoment

94.2 circular_fquantile

94.3 circular_fstd

94.4 circular_fvar

95 statistics

95.1 coefficient_of_determination

95.2 conditional_expectation_normal

95.3 correlation_confidence_pearson

```
confience intervals of the correlation coefficient
c.f. Fischer 1921
```

96 statistics/distributions

96.1 PDF

```
class for quasi-distributions from a set of sampling points
```

97 statistics/distributions/anisotropic

97.1 anisotropic_pattern

97.2 anisotropic_pattern_acf

97.3 anisotropic_pattern_pdf

98 statistics/distributions/beta

98.1 beta_kurt

98.2 beta_mean

98.3 beta_moment2par

```
transform central moments (mean and sd) to parameters of the beta
    function
```

98.4 beta_skew

98.5 beta_std

99 statistics/distributions/bivariate-normal

99.1 binorm_separation_coefficient

```
separation coefficient of a bimodal normal distribution
```

99.2 binormcdf

```
bio-modal gaussian distribution
```

99.3 binormfit

```
fit sum of to normal distribution to a histogram
```

99.4 binormpdf

100 statistics/distributions/chi2

100.1 chi2_kurt

100.2 chi2_mean

100.3 chi2_skew

100.4 chi2_std

101 statistics/distributions/circular-normal

101.1 wnormpdf

```
wrapped normal distribution to the unit circle
c.f. stephens
```

102 statistics/distributions/edgeworth

102.1 edgeworth_cdf

```
edgeworth expansion of an unknown cumulative distribution
with mean mu, standard deviation sigma, and third and fourth
    cumulants
c.f. Rao 2010
```

102.2 edgeworth_pdf

```
probability density of and unknown distribution
with mean mu, standard deviation sigma, and third and fourth
    cumulants
c.f. Rao 2010
```

103 statistics/distributions/exp

103.1 exppdf_max2par

104 statistics/distributions/fisher

104.1 fisher_mean

104.2 fisher_moment2par

104.3 fisher_std

105 statistics/distributions/gamma

105.1 gamma_mean

105.2 gamma_mode

105.3 gamma_mode2par

105.4 gamma_moment2par

```
transform modes (mu,sd) to parameters of the gamma distribution
```

105.5 gamma_std

105.6 gamma_stirling

105.7 gampdf_man

105.8 generalized_gamma_mean

106 statistics/distributions/hotelling-t2

106.1 t2cdf

```
Hotelling's T-squared cumulative distribution
```

106.2 t2inv

```
inverse of Hotelling's T-squared cumulative distribution
```

107 statistics/distributions/kurt-normal

107.1 kurtncdf

107.2 kurtnpdf

108 statistics/distributions/log-triangular

108.1 logtrialtcdf

```
pdf of a logarithmic triangular distribution
```

108.2 logtrialtinv

```
inverse of the logarithmic triangular distribution
= (d F log(a) log(b) + a log(b) - b log(a) - d F log(a) log(c) - a
    log(c) + d F log(b) log(c) + b log(c) - d F log^2(b))/((log(a)
    - log(b)) W((a^(-1/(log(a) - log(b))) (b^(-log(c)/log(a) - 1/
    log(a)) c)^(-log(a)/(log(a) - log(b))) (-d F log^2(b) + a log(b
    ) + d F log(a) log(b) + d F log(c) log(b) - b log(a) - a log(c)
     + b log(c) - d F log(a) log(c)))/(log(a) - log(b))))
x = (d F log(a) log(b) + a log(b) - b log(a) - d F log(a) log(c) - a
     log(c) + d F log(b) log(c) + b log(c) - d F log^2(b))/((log(a)
    - log(b)) W((a^(-1/(log(a) - log(b))) (b^(-log(c)/log(a) - 1/log
    (a)) c)^(-log(a)/(log(a) - log(b))) (-d F log^2(b) + a log(b) +
    d F log(a) log(b) + d F log(c) log(b) - b log(a) - a log(c) + b
    log(c) - d F log(a) log(c)))/(log(a) - log(b))))
```

108.3 logtrialtmean

```
mean of the logarithmic triangular distribution
```

108.4 logtrialtpdf

```
density of the logarithmic triangular distribution
```

108.5 logtrialtrnd

108.6 logtricdf

`cumulative distribution of the logarithmic triangular distribution`

108.7 logtriinv

`invere of the logarithmic triangular distribution`

108.8 logtrimean

`mean of the logarithmic triangular distribution`

108.9 logtripdf

`probability density of the logarithmic triangular distribution`

108.10 logtrirnd

109 statistics/distributions/log-uniform

109.1 logu_median

`median of the log-uniform distribution`

109.2 logucdf

`probability density of the logarithmic uniform distribution`

109.3 logucm

```
central moments of the log-uniform distribution
```

109.4 loguinv

```
inverse of the log-uniform distribution
```

109.5 logumean

```
mean of the log-uniform distribution
```

109.6 logupdf

```
pdf of the log uniform distribution
```

109.7 logurnd

```
random numbers following a log-uniform distribution
```

109.8 loguvar

```
variance of the log-uniform distribution
```

110 statistics/distributions/loglog

110.1 loglogpdf

111 statistics/distributions/lognormal

111.1 logn_corr

```
function corr_eaeb = logn_corr(lr,lmu_a,lmu_b,lsd_a,lsd_b)

correlation of two log-normal random variables, where the log of
    the variables
is correlated with r
```

111.2 logn_cov

```
covariance of two log-normally distributed random variables,
cov(ea,eb) = cov(exp(mua + sa*za),exp(mub + sb*zb))
where za, zb are standard normal distributed and correlated
```

111.3 logn_mean

111.4 logn_mode

```
mode (maximum) of the log-normal density
```

111.5 logn_mode2par

111.6 logn_moment2par

```
transform the mode (mu,sd) to parameters of the log normal
    distribution
```

111.7 logn_moment2par_correlated

111.8 logn_param2moment

```
transform parameters to mode (mu, sd) for the log normal
    distribution
```

111.9 logn_skewness

111.10 logn_std

111.11 lognpdf_

```
log normal distribution called by modes rather than parameters
```

111.12 lognpdf_entropy

112 statistics/distributions/logskew

112.1 logskewcdf

112.2 logskewpdf

113 statistics/distributions/mises

113.1 mises_max2par

113.2 mises_std

113.3 mises_var

113.4 misesn_max2par

113.5 misesnpdf

113.6 misespdf

114 statistics/distributions

114.1 ncx2_moment2par

115 statistics/distributions/normal

115.1 normpdf_entropy

115.2 normpdf_mode

115.3 normpdf_mode2par

116 statistics/distributions/passes

116.1 bandpass1d_continuous_pdf

```
function [S_bp,Sc] = spectral_density_bandpass_continous(fx,fc,
    order,normalize,pp)

output :
S_bp : spectral density of the bandpass filter in continuos space
   limit case of the discrete bandpass for dx -> 0
Sc   : scale factor to normalize area to 1, if noramlize = true

input :
f     : frequency (abszissa)
fc    : central freqeuncy, location of maximum on abszissa
order : number of times filter is applied iteratively, not
    necessarily integer
normalize : normalize area under curve int_0^inf S(f) df = 1, if
    not maximum S(fc) = 1
pp    : powers for recombination of the lowpass filter
```

116.2 bandpass1d_continuous_pdf_max

```
maximum of the bandpass spectral density
```

116.3 bandpass1d_continuous_pdf_max2par

```
transform mode (maxima) of the bandpass spectral density into the
    paramter
of the underlying distribution
```

116.4 bandpass1d_continuous_pdf_scale

```
normaliztation scale of the spatial bandpass density
```

116.5 bandpass1d_discrete_pdf

```
spectral density of the discrete spatial (two-sided) bandpass
    filter
```

116.6 bandpass2d_discrete_pdf

116.7 bandpass2d_pdf_exact

```
function Sb = bandpass2d_pdf(fr,a,order)
not normalized, max (S) = 1;
```

116.8 bandpass2d_pdf_hankel

116.9 bandpass2d_pdf_mode

116.10 bandpass2d_pdf_mode2par

```
transform mode (maxima) of the bandpass spectral density into the
    paramter
of the underlying distribution
```

116.11 bandpass2d_pdf_scale

116.12 highpass1d_continuous_pdf

```
function [S_bp,Sc] = spectral_density_highpass_continous(fx,fc,
    order,normalize,pp)

output :
S_bp : spectral density of the highpass filter in continuos space
    limit case of the discrete highpass for dx -> 0
Sc   : scale factor to normalize area to 1, if noramlize = true

input :
```

```
f     : frequency (abszissa)
fc    : central freqeuncy, location of maximum on abszissa
order : number of times filter is applied iteratively, not
    necessarily integer
normalize : normalize area under curve int_0^inf S(f) df = 1, if
    not maximum S(fc) = 1
pp    : powers for recombination of the lowpass filter
```

116.13 highpass1d_discrete_cos_pdf

```
consine shaped spectral density of a highpass filter
```

116.14 highpass1d_disrete_pdf

```
spectral density of the pth-order high-pass

Note that there are two alternative definitions
S_hp = S_h1^p = (1 - S_l1)^p (recursive highpass-filtering)
or S_hp~ = (1 - S_lp^p) (1 - recursive lowpass-filtering)
here, recursive highpass filtering is represented

Sh = |F*Ah|^2p
Sh^(1/2p) = F Ah = F(I-Al)
         = F(I - Al)
         = F(I - F^-1 Sl^1/2 F)
         = (F F^-1 - Sl^1/2) F
         = (I - Sl^(1/2)) F

function [S_h,S_h1] = spectral_density_hp(f,r0,fmax,order,varargin)
```

116.15 highpass2d_discrete_pdf

116.16 highpass2d_pdf

116.17 highpass2d_pdf_hankel

116.18 lowpass1d_continuous_pdf

116.19 lowpass1d_continuous_pdf_scale

116.20 lowpass1d_discrete_pdf

116.21 lowpass1d_one_sided_pdf

116.22 lowpass2d_discrete_acf

```
truncated, not wrapped at the end
```

116.23 lowpass2d_discrete_pdf

116.24 lowpass2d_pdf

116.25 lowpass2d_pdf_hankel

```
spectral density of the two-dimensional lowpass filter with
    autocorrelation

r = exp(-a*sqrt(x^2 + y^2))

efficiently estimated with gauss-laguerre integration and 1D-FFT:

for a radially symmetric function, the radial density is
Sr(r) = S2d(r,0) = S2d(0,r)
```

```
with density S2d and autocorrelation R2d
S2d = F_2d^-1 (R2d)
by the slicing theorem:
S2d(x,0) = F_1d^-1 (int R2d(x,y) dy)
```

116.26 lowpass2d_pdf_series

117 statistics/distributions

117.1 pdfsample

```
pdf from sample distribution
Note: better use kernal density estimates
```

118 statistics/distributions/phase-drift

118.1 phase_drift_acf

118.2 phase_drift_acf_2d

118.3 phase_drift_cdf

118.4 phase_drift_inv

118.5 phase_drift_parallel_acf

118.6 phase_drift_parallel_pdf

118.7 phase_drift_parallel_pdf_max

118.8 phase_drift_parallel_pdf_max2par

118.9 phase_drift_parallel_pdf_mode2par

118.10 phase_drift_patch_size_distribution

118.11 phase_drift_pdf

```
spectral density of a fourier series where the phase undergoes
    brownian motion
with standard deviation s per unit distance
```

118.12 phase_drift_pdf_2d

118.13 phase_drift_pdf_mode

```
mode (maximum) of the spectral density of the fourier series with
    brownian phase
```

118.14 phase_drift_pdf_mode2par

```
transform mode to parameters of the brownian phase spectral density
```

118.15 phase_drift_pdf_reg2par

118.16 phase_drift_pdf_scale

```
normalization scale of the brownian phase spectral density
```

119 statistics/distributions/skew-normal

119.1 skew_generalized_normal_fit

119.2 skew_generalized_normpdf

119.3 skewcdf

119.4 skewparam_to_central_moments

119.5 skewpdf

```
skew-normal distribution
c.f. Azzalini 1985
```

119.6 skewpdf_entropy

120 statistics/distributions/triangular

120.1 tricdf

```
cumulative distribution of the log-triangular distribution
```

120.2 triinv

```
inverse of the triangular distribution
```

120.3 trimedian

```
median of the triangular distribution
```

120.4 tripdf

```
probability density of the triangular distribution
```

120.5 trirnd

```
random numbers of the triangular distribution
```

121 statistics/distributions/weibull

121.1 wbl_std

122 statistics/distributions/wrapped-normal

122.1 normpdf_wrapped

122.2 normmpdf_wrapped_mode

122.3 normpdf_wrapped_mode2par

123 statistics

123.1 error_propagation_fraction

123.2 error_propagation_product

123.3 example_standard_error_of_sample_quantiles

123.4 f_var_finite

```
reduction of variance when sampling from a finite population
without replacement
```

123.5 gaussfit3

123.6 gaussfit_quantile

123.7 geoserr

123.8 geostd

123.9 hodges_lehmann_correlation

```
hodges_lehmann correlatoon coefficient
c.f. Shamos 1976
c.f. Bickel and Lehmann 1976
c.f. rousseeuw 1993
c.f. Shevlyakov 2011
```

123.10 hodges_lehmann_dispersion

```
dispersion determined by the hodges lehman method
asymptotic efficiency of dispersion estimates:
standard deviation:    E(s - hat s)/s = 2/sqrt(2 n) ~ 0.707/sqrt(n)
               (100%)
hodges lehmann dispersion E(s-\hat s)/s = (pi/3)^2 /(sqrt(2 n)) ~
    0.775/sqrt(n) (91%)
mad                    E(s-\hat s)/s ~ 1.17 s/sqrt(n)
                   (60%)
c.f. Shamos 1976
c.f. Bickel and Lehmann 1976
c.f. rousseeuw 1993
nb: rousseeuw uses the 25th percentile, which is more efficient for
     small sample sizes
```

124 statistics/information-theory

124.1 akaike_information_criterion

```
akaike information criterion

serr : rmse of model prediction
n : effective sample size
k : number of parameters

c.f. akaike (1974)
c.f. sugiura 1978
```

124.2 bayesian_information_criterion

```
bayesian information criterion
```

125 statistics

125.1 jackknife_block

125.2 kurtosis_bias_corrected

```
bias corrected kurtosis
```

125.3 limit

```
limit a by lower and upper bound
```

125.4 logfactorial

```
approximate log of the factorial
```

125.5 lognfit_quantile

125.6 max_exprnd

125.7 maxnnormals

```
expected maximum of n normal variables
c.f. Wolperts
this is the median, not the mean of the maximum!
see median of gumbel
```

125.8 mean_angle

125.9 mean_max_n

125.10 mean_min_n

125.11 midrange

```
mid range of columns of X
```

125.12 minavg

```
solution of the minimum variance problem
minimise the variance of the weighted sum of n-independent
random variables with equal mean and individual variance
```

125.13 mode_man

126 statistics/moment-statistics

126.1 autocorr_man3

```
autoccorrelation of the columns of X
```

126.2 autocorr_man4

```
autocorrelation for x if x is a vector, or indivvidually for the
columns of x if x is a matrix

c.f. box jenkins 2008 eq. 2.1.12

Note that it is faster to compute the acf in frequency space
as done in the matlab internal function
```

126.3 autocorr_man5

```
autocorrellation of the columns of X
```

126.4 blockserr

```
estimate the standard error of potetially sequentilly correlated
    data
by blocking
block length should be sufficiently larger than correlation length
and sufficiently smaller than data length
this uses a sliding block approach, which reduces the variation of
    the error estimate
```

126.5 comoment

```
non-central higher order moments of the multivariate normal
    distribution

c.f. Moments and cumulants of the multivariate real and complex
    Gaussian distributions

note : there seem to be some typos in the original paper,
      for x^4 cii^2, the square seems to be missing
mu : nx1 mean vector
C  : nxn covariance matrix
k  : nx1 powers of variables in moments
```

126.6 corr_man

```
correlation of two vectors
```

126.7 cov_man

```
covariance matrix of two vectors
```

126.8 dof

```
mininum number of support points
for a polynomial of degree order in dim dimensions
```

126.9 edgeworth_quantile

```
inverse edgeworth expansion
c.f. cornis fisher 1937
c.f. Rao 2010
c.f. 2.50 in hall
CHERNOZHUKOV 3.3
```

126.10 effective_sample_size

```
effective sample size of the weighted mean of uncorrelated data
c.f. Kish
```

126.11 f_correlation

```
correction factor for standard error of the mean of n ar1-
    correlated iid samples
```

126.12 f_finite

```
reduction factor of standard error for sampling from a finite
    distribution
without replacement
```

126.13 lmean

```
mean of x.^l, not of abs
```

126.14 lmoment

```
l-moment of vector x
```

126.15 maskmean

```
mean of the masked values of X
```

126.16 masknanmean

126.17 mean1

```
mean of x
```

126.18 mean_man

```
mean and standard error of X
```

126.19 mse

```
mean squared error of residual vector res
this is de-facto the std for an unbiased residual
```

126.20 nanautocorr_man1

```
autocorrelation of a vector with nan-values
```

126.21 nanautocorr_man2

```
autocorrelation of a vector with nan-values
```

126.22 nanautocorr_man4

```
compute autocorrelation for x if x is a vector, or indivvidually
    for the
columns of x if x is a matrix
box jenkins 2008 eq. 2.1.12
TODO nan is problematic!
Note that it is faster to compute the acf in frequency space
as done in the matlab internal function
```

126.23 nancorr

```
(co)-correlation matrix when samples a NaN
```

126.24 nancumsum

```
cumulative sum, setting nan values to zero
```

126.25 nanlmean

```
mean of the l-th power of the absolute value of x
```

126.26 nanr2

```
coefficient of determination when samples are invalid
```

126.27 nanrms

```
root mean square value when sample contains nan-values
```

126.28 nanrmse

```
root mean square error from vector of residuals
this is de-facto the std for an unbiased residual
```

126.29 nanserr

```
standard error of x with respect to mean when x contains nan values
```

126.30 nanwmean

```
weighted mean
min_x sum w (x-mu)^2 => mu = sum(wx)/sum(w)
varargin can be dim
function [mu serr] = nanwmean(w,x)
```

126.31 nanwstd

```
weighed standard deviation
```

126.32 nanwvar

```
weighted variance of columns, corrected for degrees of freedom (
    bessel)

s^2 = sum(w*(x-sum(wx)/sum(w))^2)/sum(w)
```

126.33 nanxcorr

126.34 pearson

```
pearson correlation coefficient
```

126.35 pearson_to_kendall

```
conversion of pearson to kendall correlation coefficient
c.f. Kruskal 1958
```

126.36 pool_samples

```
pooled mean and standard deviation of several groups of different
    size, mean and standard deviation
```

126.37 qmean

```
trimmed mean
```

126.38 range_mean

126.39 rmse_

```
root mean square error computed from a residual vector
this is de-facto the std for an unbiased residual
```

126.40 serr

```
standard error of the mean of a set of uncorrelated samples
```

126.41 serr1

126.42 test_qskew

126.43 test_qstd_qskew_optimal_p

126.44 wautocorr

```
autocorrelation for x if x is a vector, or indivvidually for the
columns of x if x is a matrix
samples can be weighted

c.f. box jenkins 2008 eq. 2.1.12

c.f. autocorr_man4

Note that it is faster to compute the acf in frequency space
as done in the matlab internal function
```

126.45 wcorr

```
correlation of two vectors when samples are weighted
```

126.46 wcov

```
covariance of two vectors when samples are weighted
```

126.47 wdof

```
effective degrees of freedom for weighted samples
```

126.48 wkurt

```
kurtosis with weighted samples
```

126.49 wmean

```
weighted mean

min_x sum w (x-mu)^2 => mu = sum(wx)/sum(w)

varargin can be dim
function [mu serr] = wmean(w,x)
```

126.50 wrms

```
weighted root mean square
```

126.51 wserr

```
weighted root mean square error
```

126.52 wskew

```
skewness of a weighted set of samples
function sk = wskew(w,x)
```

126.53 wstd

```
weighed standard deviation
```

126.54 wvar

```
weighted variance of columns, corrected for degrees of freedom (
    bessel)
variance of the weighted sample mean of samples with same mean (but
     not necessarily same variance)
s^2 = sum (w^2(x-sum(wx)^2))

s2_mu : error of mean, s2_mu : sd of prediction
```

127 statistics

127.1 nangeomean

127.2 nangeostd

```
geometric standard deviation ignoring nan-values
```

128 statistics/nonparametric-statistics

128.1 kernel1d

```
X   : ouput x axis bins
xi  : samples along x
m   : number of bins in X
fun : kernel function
pdf : propability density of xi
```

128.2 kernel2d

```
kernel density estimate in two dimensions
```

129 statistics

129.1 normalize_exponential_random_variable

129.2 normmoment

```
expected norm of x.^n, when values x in x are iid normal with mu
    and sigma
```

129.3 normpdf2

```
pdf of the bivariate normal distribution
```

130 statistics/order-statistics

130.1 hodges_lehmann_location

```
hodges lehman location estimator

Asymptotic rms efficency of location estimte:
     mean:          1 s/sqrt(n)
     hodges lehman: sqrt(pi/3)*s ~ 1.0233 s/sqrt(n)
     median:        pi/2 s/sqrt(n) ~ 1.25 s / sqrt(n)
```

130.2 kendall

```
kendall correlation coefficient
```

130.3 kendall_to_pearson

```
convert kendall rank correlation coefficient to the person product
    moment
correlation coefficient

c.f. Kruskal, 1958, p. 823
```

130.4 mad2sd

```
transform median absolute deviation to standard deviation
for normal distributed values
```

130.5 madcorr

```
proxy correlation by median absolute deviation
```

130.6 median2_holder

130.7 median_ci

```
median and its confidence intervals under assumption of normality
se_me = sqrt(1/2 pi) 1.25331 * sd/sqrt(n)
```

130.8 median_man

```
median and confidence intervals
c is a P value for the confidence interval,
default is 0.95 (2-sigma)
median of the colums of X
```

130.9 mediani

```
index of median, if median is not unique, any of the values is
    chosen
```

130.10 nanmadcorr

```
proxy correlation by median absolute deviation
```

130.11 nanwmedian

```
weighted median, skips nan-values
```

130.12 nanwquantile

```
weighted quantile, skips nan values
```

130.13 oja_median

```
two dimensional oja median
note: the multivariate median is not unique

oja 1983, for extension to multivariate function, see chaudhri
```

130.14 qkurtosis

```
kurosis computed for quantiles

Note : this is a measurement of shape-tailedness and yields the
    same value for the
      normal distribution as "kurtosis"
      However, this is a separate statistic and hence requires
          different
      methods for calculating P-values and hypothesis testing
```

130.15 qmoments

```
moments estimated from quantiles
```

130.16 qskew

```
skewness estimated from quantiles

Note : this is a measurement of shape-symmetry and yields the same
    value for the
      skew-normal distribution as "skewness"
      However, this is an own statistic and hence requires
          different
      methods for calculating P-values and hypothesis testing
```

130.17 qskewq

```
skewness estimated by quantiles
```

130.18 qstdq

```
proxy standard deviation determined by quantiles
```

130.19 quantile1_optimisation

130.20 quantile2_breckling

```
qunatile regression
```

130.21 quantile2_chaudhuri

```
quantile regression
```

130.22 quantile2_projected

```
quantile in two dimensions
```

130.23 quantile2_projected2

```
spatial qunatile for chosen direction
```

130.24 quantile_envelope

130.25 quantile_regression_simple

```
simple quantile regression
```

130.26 ranking

```
ranking for spearman statistics
```

130.27 spatial_median

```
c.f. Oja 2008
is this the same as the oja simplex median (c.f. small 1990)?
```

130.28 spatial_quantile

```
spatial quantile
```

130.29 spatial_quantile2

```
spatial quantile
```

130.30 spatial_quantile3

```
spatial quantile
```

130.31 spatial_rank

```
unsigned rank
```

130.32 spatial_sign

```
spatial sign
```

130.33 spatial_signed_rank

```
signed rank
Note: this is only a true rank if X is normal with zero mean,
    abitrary variance
```

130.34 spearman

```
spearman's product moment coefficient
```

130.35 spearman_rank

130.36 spearman_to_pearson

```
conversion of spearman rank to person product moment correlation
    coefficient
```

130.37 wmedian

```
weighted median
```

130.38 wquantile

```
weighted quantile
```

131 statistics

131.1 qstd

131.2 quantile_extrap

131.3 quantile_sin

132 statistics/random-number-generation

132.1 laplacernd

```
random number of laplace distribution
```

132.2 randc

```
correlate to correlated standard normally distributed vectors
```

132.3 skewness2param

132.4 skewpdf_central_moments

132.5 skewrnd

```
random numbers of the skew normal distribution
```

133 statistics

133.1 range

```
range and mid range of input
```

133.2 resample_with_replacement

134 statistics/resampling-statistics/@Jackknife

134.1 Jackknife

```
class for leave out 1 (delete 1) Jackknife estimates

note 1 : the 1-delete jackknife does not yield consistend estimates
     for all functions,
       in particular it will perform poorly on robust estimation
           functions
       this is overcome by the d-delete jacknife, where d has to
           exceed the breakdown point
       of the estimating function, for example sqrt(n) for the
           median
       as this leads to unreasonably large number of repetitions,
           bootstrap
       is recommended for large sample cases (or blocking for
           sequential data)
note 2 : as a linearisation, jackknife underestimates the error
    variance in case of
        dependence in the data
note 3 : studentisation and the leave out 1 jackknife are related
note 4 : the double 1 sample jacknife performs iferior to the d1
    jacknife
```

134.2 estimated_STATIC

```
jacknife estimate of mean, bias and standard error
theta0 : estimate from all samples
thetad : set of estimates obtained by leaving out one data point
    each
        last dimension of theta is assumed to be the jackknife
            dimension
```

134.3 matrix1_STATIC

```
matrix of estimation for leaving out two samples at a time
```

134.4 matrix2

```
matrix of estimations for jacknive with two samples left out
```

135 statistics/resampling-statistics

135.1 block_jackknife

135.2 jackknife_moments

```
moments determined by the jacknife

func : function of interest on the samples (e.g. mean)
A    : parameter matrix
       columns : parameters
       rows    : samples of the parameter sets
d   : number of samples left out
```

135.3 moving_block_jackknife

```
blocked Jacknfife for autocorrelated data
sliding block, statistically more efficient but computationally
    expensive
note, number of blocks must be sufficiently large h ~ sqrt(n)? << n
```

135.4 randblockserr

```
standard error of sequentilly correlated data by blocking
block length should be sufficiently larger than correlation length
and sufficiently smaller than data length
this uses a sliding block approach, which reduces the variation of
    the error estimate
TODO this does not work, randomly picking samples does not reveal
    the correlation
```

135.5 resample

```
resample a vector and apply function to it

TODO, should be with replacement

n  : number of samples
m  : number of subsamples
cx : maximum number of combinations
```

136 statistics

136.1 scale_quantile_sd

```
scale factor for the standard deviation
of the asymtpotic distibution of sample quantiles
(for normal distribution)
see cadwell, 1952
```

136.2 sd_sample_quantiles

136.3 spatialrnd

136.4 trimmed_mean

```
trimmed mean
```

136.5 ttest2_man

```
two-sample t-test
here posix return value standard: h = 0 accepted, h = 1 failed
note: the matlab logic is inverse : h = 1 accepted, h = 0 failed
two sided univariate t-test
```

136.6 ttest_man

```
two-sample t-test
unequal sample size
equal variance
```

136.7 ttest_paired

```
paired t-test
unequal sample size
equal variance
more powerfull than unpaired test, as long as correlation between
    x1 and x2 > 0
```

136.8 uniformnpdf

136.9 wgeomean

```
weighted geometric mean
function mu = wgeomean(w,x)
```

136.10 wgeovar

```
variance of the weighted geometric mean
```

136.11 wharmean

```
weighted harmonic mean
```

136.12 wharstd

136.13 wharvar

137 stochastic

137.1 brownian_drift_hitting_probability

137.2 brownian_drift_hitting_probability2

137.3 brownian_field

```
simulate Fractional Brownian field on unit disk, with Hurst
    parameter 'H';
Reference:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

137.4 brownian_field_scaled

```
generate a square (fractal brownian) field where the variance is
    ellyptic,
i.e. increasing at different rates in both axes
this is facilitated by cropping and stretching
TODO can the kernel directly be adapted?
```

137.5 brownian_motion_1d_acf

137.6 brownian_motion_1d_cov

137.7 brownian_motion_1d_fft

137.8 brownian_motion_1d_fourier

137.9 brownian_motion_1d_interleave

137.10 brownian_motion_1d_laplacian

137.11 brownian_motion_2d_cov

137.12 brownian_motion_2d_fft

137.13 brownian_motion_2d_fft_old

137.14 brownian_motion_2d_fourier

137.15 brownian_motion_2d_interleave

137.16 brownian_motion_2d_interleaving

137.17 brownian_motion_2d_kahunen

137.18 brownian_motion_2d_laplacian

137.19 brownian_motion_with_drift_hitting_probability

138 stochastic/geometric-ar1

138.1 geometric_ar1_2d_generate

138.2 geometric_ar1_2d_generate_1

```
realization of the spatial geometric ornstein (geometric ar1)
    process
averaged over grid cells
```

138.3 geometric_ar1_2d_grid_cell_averaged_cov

138.4 geometric_ar1_2d_grid_cell_averaged_generate

```
simulate a grid cell averaged stochastic process exp(z)
where z follows a geometric Ornstein-Uhlenbeck (AR1) process
with mean lmu, standard deviatian sd and stationary autocorrelation

val = exp(z)
mean(z) = lmu
std(z) ) ls
corr(z(0,0),z(x,y)) = exp(-sqrt(x^2 + y^2)/theta)
```

138.5 geometric_ar1_2d_grid_cell_averaged_moment2par

138.6 geometric_ar1_2d_grid_cell_averaged_std

```
mean of the values val
covariance fucntion of the values
```

139 stochastic

139.1 ornstein_uhlenbeck_cov

139.2 ornstein_uhlenbeck_mean

139.3 ornstein_uhlenbeck_spectral_density

139.4 ornstein_uhlenbeck_std

140 mathematics

`mathematical functions of various kind`

140.1 ternary_diagram

141 test/finance

141.1 test_gbb_mean

141.2 test_gbb_std

141.3 test_gbm_mean

141.4 test_gbm_mean_entire_series

141.5 test_gbm_moment2par

141.6 test_gbm_moment2par_entire_series

141.7 test_gbm_std

141.8 test_gbm_std_entire_series

142 test/fourier

142.1 test_fourier_freq2ind

143 test/master

143.1 dat_test_lanczos_3d_k_20_n_40

143.2 poisson2d_blk

143.3 qr_implicit_givens_2

143.4 spectral_derivative_2d

143.5 test_2d_eigensolver_hydrogen

143.6 test_2d_refine

143.7 test_3d_eigensolver_hydrogen

143.8 test_FEM

143.9 test_Mesh_3d

143.10 test_arnoldi

143.11 test_arpackc

143.12 test_assemble

143.13 test_assembly_performance

143.14 test_bc_one_sided

143.15 test_compare_solvers

143.16 test_complete

143.17 test_convergence

143.18 test_convergence_b

143.19 test_df_2d

143.20 test_eig_algs

143.21 test_eig_inverse

143.22 test_eigs_lanczos

143.23 test_eigs_lanczos_1

143.24 test_eigs_lanczos_2

143.25 test_eigs_lanczos_performance

143.26 test_fdm

143.27 test_fdm_d_vargrid

143.28 test_fdm_spectral

143.29 test_fem

143.30 test_fem_1d

143.31 test_fem_1d_higher_order

143.32 test_fem_2d_adaptive

143.33 test_fem_2d_higher_order

143.34 test_fem_3d_higher_order

143.35 test_fem_3d_refine

143.36 test_fem_b

143.37 test_fem_derivative

143.38 test_fem_quadrature

143.39 test_final

143.40 test_fix_substitution

143.41 test_forward

143.42 test_get_sparse_arrays

143.43 test_harmonic_oscillator

143.44 test_high_order_fdm_periodic_bc

143.45 test_hydrogen_wf

143.46 test_ichol

143.47 test_interpolation

143.48 test_inverse_problem

143.49 test_it_vs_exact

143.50 test_jama

143.51 test_jd

143.52 test_jdqz

143.53 test_lanczos_2

143.54 test_lanczos_biorthogonal

143.55 test_laplacian

143.56 test_laplacian_non_uniform

143.57 test_laplacian_simple

143.58 test_mesh_2d_uniform

143.59 test_mesh_2d_uniform_2

143.60 test_mesh_circle

143.61 test_mesh_generation

143.62 test_mesh_interpolate

143.63 test_mg

143.64 test_minres_recycle

143.65 test_multigrid

143.66 test_nc

143.67 test_nonuniform_symmetric

143.68 test_pde

143.69 test_permutation

143.70 test_poison_fem

143.71 test_polar

143.72 test_potential

143.73 test_powers

143.74 test_precondition

143.75 test_project_rectangle

143.76 test_qr

143.77 test_quantum_well

143.78 test_radial_adaptive

143.79 test_radial_confinement

143.80 test_radial_fixes

143.81 test_refine_2d

143.82 test_refine_2d_b

143.83 test_refine_3d

143.84 test_refine_structural

143.85 test_regularisation

143.86 test_round_off

143.87 test_schrödinger_potentials

143.88 test_uniform_mesh

143.89 test_vargrid

144 test/numerical-methods/optimisation

144.1 test_extreme3

145 test/numerical-methods

145.1 test_advection_kernel

146 test/signal-processing/autocorrelation

146.1 test_acf

146.2 test_acf_bias

146.3 test_acfar1_2

146.4 test_acfar1_3

146.5 test_acfar1_4

146.6 test_acfar2

146.7 test_ar1_var_factor

146.8 test_ar1_var_factor_2

146.9 test_ar1_var_mu_single_sample

146.10 test_ar1_var_pop

146.11 test_ar1_var_pop_1

146.12 test_ar1delay

146.13 test_ar2

146.14 test_phase_drift_acf

147 test/signal-processing/passes

147.1 test_bandpass2d

147.2 test_bandpass2d_ideal

147.3 test_lowpass1d_fft

147.4 test_lowpass1d_implicit

147.5 test_lowpass2d_anisotropic

147.6 test_lowpass2d_fft

147.7 test_lowpass2d_rho

148 test/signal-processing/periodogram

148.1 test_periodicity_test_2d

148.2 test_periodogram_bartlett_se

148.3 test_periodogram_gauss

148.4 test_periodogram_radial

148.5 test_periodogram_test

148.6 test_periodogram_test_periodicity_2d

148.7 test_periogogram_significance

149 test/signal-processing/spectral-density

149.1 test_phase_drift_parallel_pdf

149.2 test_phase_drift_parallel_pdf_mode2par

149.3 test_phase_drift_pdf

149.4 test_phase_drift_pdf_2d

149.5 test_phase_drift_pdf_mode

149.6 test_phase_drift_pdf_mode2par

149.7 test_phase_drift_pdf_scale

149.8 test_spectral_density_2

149.9 test_spectral_density_bandpass_2d

149.10 test_spectral_density_bandpass_2d_max2par

149.11 test_spectral_density_bandpass_continuous

```
title(sprintf('n %d L %g %g%%',[n,L,1e2*rmse(idx,jdx)]));
```

149.12 test_spectral_density_bandpass_continuous_1

149.13 test_spectral_density_bandpass_maximum

149.14 test_spectral_density_bandpass_scale

149.15 test_spectral_density_bp

149.16 test_spectral_density_bp_2d

149.17 test_spectral_density_bp_approx

149.18 test_spectral_density_flat

149.19 test_spectral_density_hp_cos

149.20 test_spectral_density_lorentzian_max

149.21 test_spectral_density_lorentzian_scale

149.22 test_spectral_density_lowpass

149.23 test_spectral_density_lowpass_continuous

149.24 test_spectral_density_lowpass_continuous_1

149.25 test_spectral_density_maxiumum_bias_corrected

150 test/signal-processing

150.1 test_autocorrelation_max

150.2 test_cdf_bandpass_continuous

150.3 test_fit_spectral_density

150.4 test_phase_drift_cdf

151 test/spatial-pattern-analysis

151.1 test_approximate_ratio_distribution

151.2 test_approximate_ratio_quantile

151.3 test_separate_isotropic_density

152 test/spatial-statistics

152.1 test_cov_cell_averages_1d

152.2 test_cov_cell_averages_2d

153 test/statistics/distributions/anisotropic

153.1 test_anisotropic_pattern

153.2 test_anisotropic_pattern_pdf

154 test/statistics/distributions/gamma

154.1 test_generalized_gamma_mean

155 test/statistics/distributions/log-uniform

155.1 test_logurnd

156 test/statistics/distributions/lognormal

156.1 test_logn_cov

157 test/statistics/distributions/mises

157.1 test_mises_std

158 test/statistics/distributions/passes

158.1 test_bandpass2d_pdf

158.2 test_bandpass2d_pdf_hankel

158.3 test_bandpass2d_pdf_mode

158.4 test_lowpass2d_pdf_hankel

158.5 test_lowpass2d_pdf_series

159 test/statistics/distributions/skew-normal

159.1 test_skew_generalized_normpdf

160 test/statistics/distributions

160.1 test_normpdf_wrapped

161 test/statistics/distributions/weibull

161.1 test_wbl_std

162 test/statistics/moment-statistics

162.1 test_wmean

163 test/statistics

163.1 test_fisher_moment2par

163.2 test_gamma_mode

163.3 test_normalize_exponential_random_variable

164 test/stochastic

164.1 test_brownian_field

164.2 test_brownian_field_scaled

165 test/stochastics

165.1 test_brownian_surface

166 test

166.1 test_S

166.2 test_advect_analytic

166.3 test_asymbp

166.4 test_bandwidth

166.5 test_bartlett_angle

166.6 test_bartlett_distribution

166.7 test_bartlett_expansion

166.8 test_beta

166.9 test_betainc

166.10 test_bivariate_covariance_term

166.11 test_brownian_drift_hitting_probability

166.12 test_brownian_drift_hitting_probability2

166.13 test_brownian_motion_1d

166.14 test_brownian_motion_2d_cov

166.15 test_brownian_motion_2d_fft

166.16 test_brownian_noise_1d

166.17 test_brownian_noise_2d

166.18 test_brownian_noise_interleave

166.19 test_coherence

166.20 test_combined_spectral_density

166.21 test_continuous_fourier_transform

166.22 test_convexity

166.23 test_d2

166.24 test_determine_phase_shift

166.25 test_diffuse_analytic

166.26 test_diffusion_matrix

166.27 test_ellipse

166.28 test_error_propagation_fraction

166.29 test_f

166.30 test_f2

166.31 test_fit_2d_spectral_density

166.32 test_fourier

166.33 test_fourier_derivative

166.34 test_fourier_derivative_1

166.35 test_fourier_integral

166.36 test_fourier_mask_covariance_matrix

166.37 test_ft_bp

166.38 test_gam

166.39 test_gamma_distribution

166.40 test_gampdf_man

166.41 test_gaussfit3

166.42 test_gaussian_flat

166.43 test_geoserr

166.44 test_hexagonal_pattern

166.45 test_iafrate

166.46 test_implicit_ode

166.47 test_imrotmat

166.48 test_integration

166.49 test_ivp

166.50 test_jacobian

166.51 test_lanczoswin

166.52 test_laplacian_power

166.53 test_lognfit_quantile

166.54 test_ls_perpendicular_offset

166.55 test_madcorr

166.56 test_mask

166.57 test_max_normal

166.58 test_moments

166.59 test_moments_fourier_power

166.60 test_mtimes3x3

166.61 test_noisy_oscillator

166.62 test_nonperiodic_pattern

166.63 test_normaliztation

166.64 test_ols

166.65 test_parcorr

166.66 test_positivity_preserving

166.67 test_randar1

166.68 test_randar1_multivariate

166.69 test_randar2

166.70 test_ratio_distributions

166.71 test_sd_rectwin

166.72 test_spatialrnd

166.73 test_spectrum_additivity

166.74 test_stationarity

166.75 test_stationarity2

166.76 test_sum_ij

166.77 test_sum_multivar

166.78 test_trifilt1

166.79 test_wautocorr

166.80 test_wavelet_transform

166.81 test_whittle

166.82 test_window

166.83 test_wordfilt

166.84 test_xar1_mid_term

167 mathematics

```
mathematical functions of various kind
```

167.1 trapezoidal_fixed

168 wavelet

168.1 contiuous_wavelet_transform

```
continuous wavelet transform
follows "The Illustrated Wavelet Transform Handbook: Introductory
    Theory and ..."
```

168.2 cwt_man

```
continuous fourier transform
as of time of implmentation, the matlab interal cwt is affected by
serious round-off errors and has issues with the scaling,
which is not the case here
```

168.3 cwt_man2

168.4 example_wavelets

168.5 phasewrap

```
wrap the phase to +/- pi
```

168.6 test_cwt_man

168.7 test_phasewrap

168.8 test_wavelet

168.9 test_wavelet2

168.10 test_wavelet_analysis

168.11 test_wavelet_reconstruct

168.12 test_wtc

168.13 wavelet

```
wavelet windows
```

168.14 wavelet_reconstruct

```
iverses wavelet transform for single frequency
(reconstruction of time series)
n : window lengths in multiples of filter period 1/f0
```

168.15 wavelet_transform

```
wavelet transform for single frequency
n : window lengths in multiples of filter period 1/f0
```