Manual for Package: mathematics
Revision 17M

Karl Kästner

December 22, 2021

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_smooth1d_parametric

4.11 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 gbm_cdf

8.3 gbm_fit

8.4 gbm_fit_old

8.5 gbm_inv

8.6 gbm_mean

8.7 gbm_median

8.8 gbm_pdf

8.9 gbm_simulate

8.10 gbm_skewness

8.11 gbm_std

8.12 gbm_transform_time_step

8.13 put_price_black_scholes

8.14 skewgbm_simulate

8.15 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 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
```

11.3 example_fourier_window

11.4 fft_derivative

```
derivative by 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
```

11.5 fft_man

```
fast fourier transform for complex input data

input:
F : data in real space

output :

F : fourier transformation of F
```

11.6 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
```

11.7 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
```

11.8 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
```

11.9 fourier_cesaro_correction

11.10 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
```

11.11 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
```

11.12 fourier_coefficient_ramp3

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

11.13 fourier_coefficient_ramp_pulse

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

11.14 fourier_coefficient_ramp_step

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

11.15 fourier_coefficient_square_pulse

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

11.16 fourier_cubic_interaction_coefficients

11.17 fourier_derivative

```
coefficients of the derivative of a fourier series
not of discrete fourier transform (fft)
```

11.18 fourier_derivative_matrix_1d

11.19 fourier_derivative_matrix_2d

11.20 fourier_expand

```
expand values of fourier series
```

11.21 fourier_fit

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

11.22 fourier_interpolate

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

11.23 fourier_matrix

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

11.24 fourier_matrix2

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

11.25 fourier_matrix3

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

11.26 fourier_matrix_exp

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

11.27 fourier_multiplicative_interaction_coefficients

11.28 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
```

11.29 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
```

11.30 fourier_predict

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

11.31 fourier_quadratic_interaction_coefficients

11.32 fourier_random_phase_walk

11.33 fourier_range

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

11.34 fourier_regress

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

11.35 fourier_resampled_fit

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

11.36 fourier_resampled_predict

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

11.37 fourier_signed_square

```
coefficients of the fourier series of | cos a + cos t | (cos a +
    cos t)
in general
          cos a is midrange
          cos t is tidal variation
c.f Dronkers
```

11.38 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
```

11.39 fourier_transform_fractional

11.40 hyperbolic_fourier_box

11.41 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)
```

11.42 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
```

11.43 nanfft

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

11.44 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)
```

11.45 roots_fourier

```
zeros of continuous fourier series series

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

11.46 spectral_density

```
spectral density
```

11.47 test_complex_exp_product

11.48 test_fourier_filter

11.49 test_idftmtx

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 pf 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

```
n-points on an ellipse
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 medianangle

```
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 mathematics

```
mathematical functions of various kind
```

15.1 hexagonal_pattern

16 histogram/@Histogram

16.1 2x

16.2 Histogram

16.3 bimodes

16.4 cdf

16.5 cdfS

16.6 chi2test

16.7 cmoment

16.8 cmomentS

16.9 entropy

16.10 entropyS

16.11 export_csv

16.12 iquantile

16.13 kstest

16.14 kurtosis

16.15 kurtosisS

16.16 mean

16.17 meanS

16.18 median

16.19 medianS

16.20 mode

16.21 modeS

16.22 moment

16.23 momentS

16.24 pdf

16.25 quantile

16.26 quantileS

16.27 resample

16.28 setup

16.29 skewness

16.30 skewnessS

16.31 stairs

16.32 stairsS

16.33 std

16.34 stdS

16.35 var

16.36 varS

17 histogram

17.1 hist_man

17.2 histadapt

17.3 histconst

17.4 pdf_poly

17.5 plotcdf

17.6 test_histogram

18 mathematics

`mathematical functions of various kind`

18.1 imrotmat

19 linear-algebra

19.1 averaging_matrix_2

19.2 colnorm

```
norms of columns
```

19.3 condest_

```
estimation of the condition number
```

19.4 connectivity_matrix

20 linear-algebra/coordinate-transformation

20.1 barycentric2cartesian

```
barycentric to cartesian coordinates
```

20.2 barycentric2cartesian3

```
convert barycentric to cartesian coordinates
```

20.3 cartesian2barycentric

```
cartesian to barycentric coordinates
```

20.4 cartesian_to_unit_triangle_basis

```
transform coodinates into unit triangle
```

20.5 ellipsoid2geoid

20.6 example_approximate_utm_conversion

20.7 latlon2utm

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

20.8 latlon2utm_simple

20.9 lowrance_mercator_to_wgs84

```
convert lowrance coordinates to wgs84

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

20.10 nmea2utm

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

20.11 sn2xy

```
convert sn to xy coordinates
```

20.12 unit_triangle_to_cartesian

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

20.13 utm2latlon

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

20.14 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
```

20.15 xy2sn

```
convert cartesian to streamwise coordiantes
```

20.16 xy2sn_java

```
use java port for speed up
```

20.17 xy2sn_old

```
transform points from cartesian into streamwise coordinates

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

21 linear-algebra

21.1 det2x2

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

21.2 det3x3

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

21.3 det4x4

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

21.4 diag2x2

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

21.5 down

21.6 eig2x2

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

22 linear-algebra/eigenvalue

22.1 eig_bisection

22.2 eig_inverse

22.3 eig_inverse_iteration

22.4 eig_power_iteration

23 linear-algebra/eigenvalue/jacobi-davidson

23.1 afun_jdm

23.2 davidson

23.3 jacobi_davidson

23.4 jacobi_davidson_qr

23.5 jacobi_davidson_qz

23.6 jacobi_davidson_simple

23.7 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'
```

23.8 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'
```

23.9 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
```

23.10 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
    ============================================
%=========================================================================

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

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

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

23.11 mfunc_jdm

23.12 mgs

23.13 minres_

23.14 mv_jacobi_davidson

24 linear-algebra

24.1 first

24.2 gershgorin_circle

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

24.3 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
```

24.4 ieig2x2

```
reconstruct matrix from eigenvalue decomposition
```

24.5 inv2x2

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

24.6 inv3x3

24.7 inv4x4

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

24.8 kernel2matrix

25 linear-algebra/lanczos

25.1 arnoldi

25.2 arnoldi_new

25.3 eigs_lanczos_man

25.4 lanczos

25.5 lanczos_

25.6 lanczos_biorthogonal

25.7 lanczos_biorthogonal_improved

25.8 lanczos_ghep

25.9 mv_lanczos

25.10 reorthogonalise

25.11 test_lanczos

26 linear-algebra

26.1 left

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

27 linear-algebra/linear-systems

27.1 gmres_man

```
break on convergence
```

27.2 minres_recycle

28 linear-algebra

28.1 lpmean

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

28.2 lpnorm

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

28.3 matvec3

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

28.4 max2d

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

28.5 mid

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

28.6 mpoweri

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

28.7 mtimes2x2

28.8 mtimes3x3

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

28.9 nannorm

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

28.10 nanshift

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

28.11 nl

```
number rows (lines) of a matrix

analogue to unix nl command
```

28.12 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)
```

28.13 normalize1

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

28.14 normrows

28.15 orth2

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

28.16 orth_man

```
orthogonalize the columns of A
```

28.17 orthogonalise

```
make x orthogonal to Y
```

28.18 padd2

28.19 paddext

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

28.20 paddval1

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

28.21 paddval2

```
padd values to x
```

29 linear-algebra/polynomial

29.1 chebychev

```
chebycheff polynomials
```

29.2 piecewise_polynomial

```
evaluate piecewise polynomial
```

29.3 roots1

```
roots of linear functions
```

29.4 roots2

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

29.5 roots2poly

29.6 roots3

29.7 roots4

29.8 roots_piecewise_linear

29.9 test_roots4

29.10 vanderi_1d

```
vandermonde matrix of an integral
```

30 linear-algebra

30.1 randrot

```
random rotation matrix
```

30.2 right

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

30.3 rot2

```
rotation matrix from angle
```

30.4 rot2dir

```
rotation matrix from direction vector
```

30.5 rot3

30.6 rotR

30.7 rownorm

30.8 simmilarity_matrix

30.9 spnorm

```
frobenius norm
```

30.10 spzeros

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

30.11 test_roots3

30.12 transform_minmax

30.13 transpose3

```
transpose stacked 3x3 matrices
```

30.14 transposeall

30.15 up

30.16 vanderd_2d

31 logic

```
bitwise operations on integers
```

31.1 bitor_man

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

 input:
       A (positive integer)
```

32 master/plot

32.1 attach_boundary_value

32.2 cartesian_polar

32.3 img_vargrid

32.4 plot_basis_functions

32.5 plot_convergence

32.6 plot_dof

32.7 plot_eigenbar

32.8 plot_error_estimation

32.9 plot_error_estimation_2

32.10 plot_error_fem

32.11 plot_fdm_kernel

32.12 plot_fdm_vs_fem

32.13 plot_fem_accuracy

32.14 plot_function_and_grid

32.15 plot_hat

32.16 plot_hydrogen_wf

32.17 plot_mesh

32.18 plot_mesh_2

32.19 plot_refine

32.20 plot_refine_3d

32.21 plot_runtime

32.22 plot_spectrum

32.23 plot_wavefunction

33 master/ported

33.1 assemble_2d_phi_phi

33.2 assemble_3d_dphi_dphi

33.3 assemble_3d_phi_phi

33.4 dV_2d_

33.5 derivative_2d

33.6 derivative_3d

33.7 element_neighbour_2d

33.8 prefetch_2d_

33.9 promote_2d_3_10

33.10 promote_2d_3_15

33.11 promote_2d_3_21

33.12 promote_2d_3_6

33.13 promote_3d_4_10

33.14 promote_3d_4_20

33.15 promote_3d_4_35

33.16 vander_2d

33.17 vander_3d

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/differentiation

35.1 derivative1

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

35.2 derivative2

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

36 numerical-methods/finite-difference

36.1 cdiff

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

36.2 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
```

36.3 central_difference

36.4 cmean

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

36.5 cmean2

36.6 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)
```

36.7 derivative_matrix_2_1d

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

36.8 derivative_matrix_2d

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

36.9 derivative_matrix_curvilinear

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

36.10 derivative_matrix_curvilinear_2

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

36.11 difference_kernel

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

36.12 diffusion_matrix_2d_anisotropic

36.13 diffusion_matrix_2d_anisotropic2

36.14 distmat

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

36.15 downwind_difference

36.16 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
```

36.17 laplacian

36.18 laplacian_fdm

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

36.19 lrmean

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

37 numerical-methods/finite-difference/master

37.1 fdm_adaptive_grid

37.2 fdm_adaptive_refinement_old

37.3 fdm_assemble_d1_2d

37.4 fdm_assemble_d2_2d

37.5 fdm_confinement

37.6 fdm_d_vargrid

37.7 fdm_h_unstructured

37.8 fdm_hydrogen_vargrid

37.9 fdm_mark_unstructured_2d

37.10 fdm_plot

37.11 fdm_plot_series

37.12 fdm_refine_2d

37.13 fdm_refine_3d

37.14 fdm_refine_unstructured_2d

37.15 fdm_schroedinger_2d

37.16 fdm_schroedinger_3d

37.17 relocate

38 numerical-methods/finite-difference

38.1 mid

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

38.2 pwmid

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

38.3 ratio

```
ratio of two subsequent values
```

38.4 steplength

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

38.5 swapoddeven

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

38.6 test_derivative_matrix_2d

38.7 test_derivative_matrix_curvilinear

38.8 test_difference_kernel

38.9 upwind_difference

39 numerical-methods/finite-element

39.1 Mesh_2d_java

39.2 Tree_2d_java

39.3 assemble_1d_dphi_dphi

39.4 assemble_1d_phi_phi

39.5 assemble_2d_dphi_dphi_java

39.6 assemble_2d_phi_phi_java

39.7 assemble_3d_dphi_dphi_java

39.8 assemble_3d_phi_phi_java

39.9 boundary_1d

39.10 boundary_2d

39.11 boundary_3d

39.12 check_area_2d

39.13 circmesh

39.14 cropradius

39.15 display_2d

39.16 display_3d

39.17 distort

39.18 err_2d

39.19 estimate_err_2d_3

39.20 example_1d

39.21 example_2d

39.22 explode

39.23 fem_2d

39.24 fem_2d_heuristic_mesh

39.25 fem_get_2d_radial

39.26 fem_interpolation

39.27 fem_plot_1d

39.28 fem_plot_1d_series

39.29 fem_plot_2d

39.30 fem_plot_2d_series

39.31 fem_plot_3d

39.32 fem_plot_3d_series

39.33 fem_plot_confine_series

39.34 fem_radial

```
adaptive grid
constant grid
```

39.35 flip_2d

39.36 get_mesh_arrays

39.37 hashkey

40 numerical-methods/finite-element/int

40.1 int_1d_gauss

40.2 int_1d_gauss_1

40.3 int_1d_gauss_2

40.4 int_1d_gauss_3

40.5 int_1d_gauss_4

40.6 int_1d_gauss_5

40.7 int_1d_gauss_6

40.8 int_1d_gauss_lobatto

40.9 int_1d_gauss_n

40.10 int_1d_nc_2

40.11 int_1d_nc_3

40.12 int_1d_nc_4

40.13 int_1d_nc_5

40.14 int_1d_nc_6

40.15 int_1d_nc_7

40.16 int_1d_nc_7_hardy

40.17 int_2d_gauss_1

40.18 int_2d_gauss_12

40.19 int_2d_gauss_13

40.20 int_2d_gauss_16

40.21 int_2d_gauss_19

40.22 int_2d_gauss_25

40.23 int_2d_gauss_3

40.24 int_2d_gauss_33

40.25 int_2d_gauss_4

40.26 int_2d_gauss_6

40.27 int_2d_gauss_7

40.28 int_2d_gauss_9

40.29 int_2d_nc_10

40.30 int_2d_nc_15

40.31 int_2d_nc_21

40.32 int_2d_nc_3

40.33 int_2d_nc_6

40.34 int_3d_gauss_1

40.35 int_3d_gauss_11

40.36 int_3d_gauss_14

40.37 int_3d_gauss_15

40.38 int_3d_gauss_24

40.39 int_3d_gauss_4

40.40 int_3d_gauss_45

40.41 int_3d_gauss_5

40.42 int_3d_nc_11

40.43 int_3d_nc_4

40.44 int_3d_nc_6

40.45 int_3d_nc_8

41 numerical-methods/finite-element

41.1 interpolation_matrix

41.2 mark

41.3 mark_1d

41.4 mesh_1d_uniform

41.5 mesh_3d_uniform

41.6 mesh_interpolate

41.7 neighbour_1d

41.8 old

41.9 pdeeig_1d

41.10 pdeeig_2d

41.11 pdeeig_3d

41.12 polynomial_derivative_1d

41.13 potential_const

41.14 potential_coulomb

41.15 potential_harmonic_oscillator

41.16 project_circle

41.17 project_rectangle

41.18 promote_1d_2_3

41.19 promote_1d_2_4

41.20 promote_1d_2_5

41.21 promote_1d_2_6

41.22 quadrilaterate

41.23 recalculate_regularity_2d

41.24 refine_1d

41.25 refine_2d_21

41.26 refine_2d_structural

41.27 regularity_1d

41.28 regularity_2d

41.29 regularity_3d

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

41.30 relocate_2d

41.31 test_circmesh

41.32 test_hermite

41.33 tri_assign_points

41.34 triangulation_uniform

41.35 vander_1d

```
 van der Monde matrix
```

41.36 vanderd_1d

41.37 vanderi_1d

42 numerical-methods/finite-volume/@Advection

42.1 Advection

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

42.2 dot_advection

```
advection equation
```

43 numerical-methods/finite-volume/@Burgers

43.1 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
```

43.2 dot_burgers_fdm

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

43.3 dot_burgers_fft

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

44 numerical-methods/finite-volume/@Finite_Volume

44.1 Finite_Volume

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

44.2 apply_bc

```
apply boundary conditions
```

44.3 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);
```

44.4 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
```

44.5 step_unsplit

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

45 numerical-methods/finite-volume/@Flux_Limiter

45.1 Flux_Limiter

```
class of flux limiters
```

45.2 beam_warming

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

45.3 fromm

```
fromme limiter
low res
```

45.4 lax_wendroff

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

45.5 minmod

```
min-mod schock limiter
```

45.6 monotized_central

```
monotonized central flux limiter
```

45.7 muscl

```
muscl flux limiter
```

45.8 superbee

```
superbee limiter
```

45.9 upwind

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

45.10 vanLeer

```
van Leer limiter
```

46 numerical-methods/finite-volume/@KDV

46.1 dot_kdv_fdm

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

46.2 dot_kdv_fft

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

46.3 kdv_split

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

47 numerical-methods/finite-volume/@Reconstruct_Average_E

47.1 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;
```

47.2 advect_highres

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

47.3 advect_lowress

```
single time step
low resolution
```

48 numerical-methods/finite-volume

48.1 Godunov

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

48.2 Lax_Friedrich

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

48.3 Measure

48.4 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
```

48.5 fv_swe

```
wrapper for solving SWE
```

48.6 staggered_euler

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

48.7 staggered_grid

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

49 numerical-methods

49.1 grid2quad

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

50 numerical-methods/integration

50.1 cumintL

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

50.2 cumintR

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

50.3 int_trapezoidal

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

51 numerical-methods/interpolation/@Kriging

51.1 Kriging

```
class for Kriging interpolation
```

51.2 estimate_semivariance

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

51.3 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
```

52 numerical-methods/interpolation/@RegularizedInterpolator1

52.1 RegularizedInterpolator1

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

52.2 init

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

53 numerical-methods/interpolation/@RegularizedInterpolator

53.1 RegularizedInterpolator2

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

53.2 init

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

54 numerical-methods/interpolation/@RegularizedInterpolator

54.1 RegularizedInterpolator3

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

54.2 init

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

55 numerical-methods/interpolation

55.1 IDW

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

55.2 IPoly

```
polynomial interpolation class
```

55.3 IRBM

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

55.4 ISparse

```
sparse interpolation class
```

55.5 Inn

```
nearest neighbour interpolation
```

55.6 Interpolator

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

55.7 fixnan

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

55.8 idw1

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

55.9 idw2

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

55.10 inner2outer

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

55.11 inner2outer2

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

55.12 interp1_circular

55.13 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
```

55.14 interp1_man

```
interpolate
```

55.15 interp1_piecewise_linear

55.16 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
```

55.17 interp1_slope

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

55.18 interp1_smooth

55.19 interp1_unique

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

55.20 interp2_man

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

55.21 interp_angle

```
interpolate an angle
```

55.22 interp_fourier

```
interpolation by the fourier method
```

55.23 interp_fourier_batch

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

55.24 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_);
```

55.25 interp_sn2

```
interpolation in streamwise coordinates
```

55.26 interp_sn3

55.27 interp_sn_

55.28 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)
```

55.29 resample1

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

55.30 resample_d_min

```
resample a function
```

55.31 resample_vector

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

55.32 test_interp1_limited

56 numerical-methods

56.1 inverse_complex

56.2 maccormack_step

56.3 minmod

57 numerical-methods/multigrid

57.1 mg_interpolate

57.2 mg_restrict

58 numerical-methods/ode/@BVPS_Characteristic

58.1 BVPS_Characteristic

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

58.2 assemble1_A

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

58.3 assemble1_A_Q

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

58.4 assemble2_A

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

58.5 assemble_AA

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

58.6 assemble_AAA

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

58.7 assemble_Ic

58.8 bvp1c

58.9 check_arguments

58.10 couple_junctions

58.11 derivative

58.12 init

58.13 inner2outer_bvp2c

58.14 reconstruct

58.15 resample

58.16 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])
```

58.17 test_assemble1_A

58.18 test_assemble2_A

59 numerical-methods/ode/@Time_Stepper

59.1 Time_Stepper

59.2 solve

60 numerical-methods/ode

60.1 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])
```

60.2 bvp2wavetrain

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

60.3 bvp2wavetwopass

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

60.4 ivp_euler_forward

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

60.5 ivp_euler_forward2

60.6 ivprk2

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

60.7 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]
```

60.8 ode2characteristic

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

60.9 step_trapezoidal

```
single trapezoidal step
```

60.10 test_bvp2

61 numerical-methods/optimisation

61.1 aitken_iteration

61.2 anderson_iteration

61.3 armijo_stopping_criterion

```
armijo stopping criterion for optimizations
```

61.4 astar

```
astar path finding alforithm
```

61.5 binsearch

```
binary search on a line
```

61.6 bisection

```
bisection
```

61.7 box1

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

61.8 box2

61.9 cauchy

61.10 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
```

61.11 directional_derivative

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

61.12 dud

```
optimization by the dud algorithm
```

61.13 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
```

61.14 extreme_quadratic

61.15 ftest

61.16 fzero_bisect

61.17 fzero_newton

61.18 grad

```
numerical gradient
```

61.19 hessian

```
numerical hessian
```

61.20 hessian_from_gradient

```
numerical hessian from gradient
```

61.21 hessian_projected

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

61.22 line_search

```
bisection routine
```

61.23 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
```

61.24 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
```

61.25 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
```

61.26 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
```

61.27 line_search_quadratic2

```
quadratic line search
```

61.28 line_search_wolfe

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

61.29 ls_bgfs

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

61.30 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
```

61.31 ls_generalized_secant

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

61.32 nlcg

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

61.33 nlls

```
non-linear least squares
```

61.34 picard

```
picard iteration
```

61.35 poly_extrema

```
extrema of a polynomial
```

61.36 quadratic_function

```
evaluate quadratic function in higher dimensions
```

61.37 quadratic_programming

```
optimize by quadratic programming
```

61.38 quadratic_step

```
single step of the quadratic programming
```

61.39 rosenbrock

```
rosenbrock test function
```

61.40 sqrt_heron

```
Heron's method for the square root
```

61.41 test_directional_derivative

61.42 test_dud

61.43 test_fzero_newton

61.44 test_line_search_quadratic2

61.45 test_ls_generalized_secant

61.46 test_nlcg_6_order

61.47 test_nlls

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

62 numerical-methods/pde

62.1 laplacian2d_fundamental_solution

63 numerical-methods/piecewise-polynomials

63.1 Hermite1

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

63.2 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)
```

63.3 hp2_predict

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

63.4 hp_predict

```
predict with piecewise hermite polynomial
```

63.5 hp_regress

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

63.6 lp_count

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

63.7 lp_predict

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

63.8 lp_regress

63.9 lp_regress_

64 numerical-methods

64.1 test_adams_bashforth

65 regression/@PolyOLS

65.1 PolyOLS

```
class for polynomial least squares
```

65.2 coefftest

65.3 detrend

```
detrending by polynomial regression
```

65.4 fit

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

65.5 fit_

```
fit a polynomial function
```

65.6 predict

```
predict polynomial function values
```

65.7 predict_

65.8 slope

```
slope by linear regression
```

66 regression/@PowerLS

66.1 PowerLS

```
class for power law regression
```

66.2 fit

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

66.3 predict

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

66.4 predict_

67 regression/@Theil

67.1 Theil

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

67.2 detrend

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

67.3 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
```

67.4 predict

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

67.5 slope

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

68 regression

linear and non-linear regression

68.1 Theil_Multivariate

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

68.2 areg

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

68.3 ginireg

```
gini regression
```

68.4 hesssimplereg

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

68.5 l1lin

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

68.6 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)
```

68.7 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
```

68.8 regression_method_of_moments

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

68.9 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
```

68.10 theil2

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

68.11 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
```

68.12 total_least_squares

```
total least squares
```

68.13 weighted_median_regression

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

69 set-theory

69.1 issubset

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

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

70 mathematics

mathematical functions of various kind

70.1 shuffle_index

71 signal-processing

71.1 acf_bp

71.2 acf_decay_rate

71.3 acf_effective_sample_size

```
effective sample size from acf
```

71.4 acf_genton

```
autocorrelation function
```

71.5 acf_hp

71.6 acf_lp

71.7 acf_periodic_additive_noise

71.8 acf_periodic_windowed

71.9 acf_radial

71.10 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
```

71.11 acfar1_2

```
autocorrelation of the ar1 process
```

71.12 acfar2

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

71.13 acfar2_2

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

71.14 ar1_cutoff_frequency

71.15 ar1_effective_sample_size

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

71.16 ar1_mse_mu_single_sample

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

71.17 ar1_mse_pop

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

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

71.18 ar1_mse_range

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

71.19 ar1_spectrum

```
spectrum of the ar1 process
```

71.20 ar1_to_tikhonov

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

71.21 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
```

71.22 ar1_var_factor_

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

71.23 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|
```

71.24 ar1delay

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

71.25 ar1delay_old

```
autocorrelation of the residual
```

71.26 ar2_acf2c

71.27 ar2conv

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

71.28 ar2dof

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

71.29 ar2param

```
ar2 parameter estimation from first two terms of acf

acf = [1 a1 a2 ...]
```

71.30 asymwin

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

71.31 autocorr2

71.32 autocorr_fft

```
autocorrelation function
```

71.33 bandpass

```
bandpass filter
```

71.34 bandpass1d

71.35 bandpass1d_fft

71.36 bandpass1d_implicit

71.37 bandpass2

```
bandpass filter
```

71.38 bandpass2d

71.39 bandpass2d_2

71.40 bandpass2d_fft

71.41 bandpass2d_implicit

71.42 bandpass2d_iso

71.43 bandpass_max

71.44 bandpass_max2

71.45 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
```

71.46 bin1d

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

71.47 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
```

71.48 binormrnd

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

71.49 coherence

71.50 conv1_man

```
convolutions with padding
```

71.51 conv2_man

```
convolution in 2d
```

71.52 conv2z

71.53 conv30

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

71.54 conv_

```
convolution of a with b
```

71.55 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
```

71.56 convz

71.57 cosexpdelay

71.58 csmooth

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

71.59 daniell_window

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

71.60 danielle_window

```
danielle fourier window
```

71.61 db2neper

```
convert decibel to neper
```

71.62 db2power

```
power ratio from db
```

71.63 derive_bandpass_normalization_and_zeros

71.64 derive_danielle_weight

71.65 derive_limit_0_acfar

71.66 detect_peak

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

71.67 digital_low_pass_filter

```
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)
```

71.68 doublesum_ij

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

71.69 effective_sample_size_to_ar1

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

71.70 filt_hodges_lehman

71.71 filter1

```
filter along one dimension
```

71.72 filter2

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

71.73 filter_

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

71.74 filter_arg

71.75 filter_f0_to_rho

71.76 filter_r_to_f0

71.77 filter_rho_to_f0

71.78 filter_twosided

71.79 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
```

71.80 filterp

71.81 filterp1

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

71.82 filterstd

71.83 firls_man

```
design finite impulse response filter by the least squares method
```

71.84 fit_spectral_density

71.85 flattopwin

```
the flat top window
```

71.86 frequency_response_boxcar

```
frquency response of a boxcar filter
```

71.87 freqz_boxcar

```
frequncy response of a boxcar filter
```

71.88 gaussfilt1

```
filter data series with a gaussian window
```

71.89 hanchangewin

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

71.90 hanchangewin2

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

71.91 hanwin

```
hanning filter window
```

71.92 hanwin_

```
hanning filter window
```

71.93 high_pass_1d_simple

71.94 highpass

```
high pass filter
```

71.95 highpass1d_implicit

71.96 highpass2d_fft

71.97 highpass2d_implicit

71.98 jackknife_block

71.99 kaiserwin

```
kaiser filter window
```

71.100 kalman

```
Kalman filter
```

71.101 lanczoswin

```
Lanczos window
```

71.102 last

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

71.103 lowpass

```
low pass filter
```

71.104 lowpass1d_fft

71.105 lowpass1d_implicit

71.106 lowpass2

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

71.107 lowpass2d_2

71.108 lowpass2d_anisotropic

71.109 lowpass2d_fft

71.110 lowpass2d_implicit

71.111 lowpass_iir

```
iir-low pass
```

71.112 lowpass_iir_symmetric

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

71.113 lowpassfilter2

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

71.114 maxfilt1

71.115 meanfilt1

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

71.116 medfilt1_man

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

71.117 medfilt1_man2

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

71.118 medfilt1_padded

```
median filter with padding
```

71.119 medfilt1_reduced

```
median filter with padding
```

71.120 mid_term_single_sample

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

71.121 minfilt1

71.122 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
```

71.123 mysmooth

71.124 nanautocorr

```
autocorrelation with nan-values
```

71.125 nanmedfilt1

```
medfilt1, skipping nans
```

71.126 neper2db

```
convert neper to db
```

71.127 peaks_man

```
peaks of a periodogram
```

71.128 periodogram

71.129 periodogram_annular

71.130 periodogram_bartlett

```
TODO welsh sliding window
```

71.131 periodogram_bootstrap

71.132 periodogram_confidence_interval

71.133 periodogram_median

71.134 periodogram_p_value

71.135 periodogram_qq

71.136 periodogram_quantiles

71.137 periodogram_std

71.138 periodogram_test

71.139 polyfilt1

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

71.140 qmedfilt1

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

71.141 randar1

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

71.142 randar1_dual

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

71.143 randar2

```
generate ar2 process
```

71.144 randarp

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

71.145 range_window

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

71.146 rectwin

```
rectangular window
```

71.147 recursive_sum

71.148 select_range

71.149 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
```

71.150 smooth2

```
smooth vectos of X
```

71.151 smooth_man

71.152 smooth_parametric

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

71.153 smooth_parametric2

```
parametrically smooth the curve
```

71.154 smooth_with_splines

71.155 smoothfft

```
filter with fast fourier transform
```

71.156 spectral_density_ar2

71.157 spectral_density_area

71.158 spectral_density_bandpass_continuous

71.159 spectral_density_bandpass_discrete

71.160 spectral_density_brownian_phase

71.161 spectral_density_flat

71.162 spectral_density_highpass

71.163 spectral_density_lorentzian

71.164 spectral_density_lowpass

71.165 spectral_density_radial

71.166 spectral_density_wperiodic

71.167 spectrogram

`spectrogram`

71.168 std_window

```
moving block standard deviation
```

71.169 sum_i_lag

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

71.170 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
```

71.171 sum_ii_

71.172 sum_ij

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

71.173 sum_ij_

71.174 sum_ij_partial_

71.175 sum_multivar

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

71.176 test_acfar1

71.177 test_acfar1_2

71.178 test_acfar1_3

71.179 test_acfar1_4

71.180 test_acfar2

71.181 test_ar1_var_factor

71.182 test_ar1_var_factor_2

71.183 test_ar1_var_mu_single_sample

71.184 test_ar1_var_pop

71.185 test_ar1_var_pop_1

71.186 test_ar1delay

71.187 test_bivariate_covariance_term

71.188 test_convexity

71.189 test_lanczoswin

71.190 test_madcorr

71.191 test_randar1

71.192 test_randar1_multivariate

71.193 test_randar2

71.194 test_sum_ij

71.195 test_sum_multivar

71.196 test_trifilt1

71.197 test_wautocorr

71.198 test_wavelet_transform

71.199 test_wordfilt

71.200 test_xar1_mid_term

71.201 tikhonov_to_ar1

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

71.202 trapwin

```
trapezoidal filter window
```

71.203 trifilt1

```
filter with triangular window
```

71.204 trifilt2

71.205 triwin

```
triangular filter window
```

71.206 triwin2

```
triangular filter window
```

71.207 varar1

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

71.208 welch_spectrogram

```
welch spectrogram
```

71.209 wfilt

```
filter with window
```

71.210 winbandpass

```
filter with bandpass
```

71.211 window2d

71.212 window_make_odd

71.213 winfilt0

```
filter with window
```

71.214 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
```

71.215 wmeanfilt

```
mean filter with window
```

71.216 wmedfilt

```
median filter with window
```

71.217 wordfilt

```
weighted order filter
```

71.218 wordfilt_edgeworth

```
weighed order filter
```

71.219 wrapphase

71.220 xar1

71.221 xcorr_man

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

72 sorting

72.1 sort2

```
sort two numbers
```

72.2 sort2d

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

73 special-functions

73.1 bessel_sphere

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

73.2 digamma_man

73.3 exp10

73.4 hankel_sphere

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

73.5 hermite

```
probabilistic's hermite polynomial by recurrence relation

input :
n : order
x : value

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

73.6 legendre_man

```
legendre polynomials
```

73.7 neumann_sphere

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

74 statistics

74.1 atan_s2

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

74.2 beta_kurt

74.3 beta_mean

74.4 beta_mode_to_parameter

```
transform modes (mean and sd) to paramets of the beta function
```

74.5 beta_skew

74.6 beta_std

74.7 chi2_kurt

74.8 chi2_mean

74.9 chi2_skew

74.10 chi2_std

74.11 coefficient_of_determination

74.12 conditional_expectation_normal

74.13 correlation_confidence_pearson

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

75 statistics/distributions

75.1 PDF

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

75.2 binorm_separation_coefficient

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

75.3 binormcdf

```
bio-modal gaussian distribution
```

75.4 binormfit

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

75.5 binormpdf

75.6 edgeworth_cdf

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

75.7 edgeworth_pdf

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

75.8 gam_moment2param

75.9 logn_mean

75.10 logn_mode

75.11 logn_moment2param

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

75.12 logn_param2moment

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

75.13 logn_std

75.14 lognpdf_

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

75.15 pdfsample

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

75.16 t2cdf

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

75.17 t2inv

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

76 statistics

76.1 example_standard_error_of_sample_quantiles

76.2 f_mean

76.3 f_std

76.4 f_var_finite

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

76.5 gam_mean

76.6 gam_std

76.7 gamma_mode_to_parameter

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

76.8 gaussfit3

76.9 gaussfit_quantile

76.10 geoserr

76.11 geostd

76.12 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
```

76.13 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
```

77 statistics/information-theory

77.1 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
```

77.2 bayesian_information_criterion

```
bayesian information criterion
```

78 statistics

78.1 kurtncdf

78.2 kurtnpdf

78.3 kurtosis_bias_corrected

```
bias corrected kurtosis
```

78.4 limit

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

78.5 logfactorial

```
approximate log of the factorial
```

78.6 loglogpdf

78.7 lognfit_quantile

78.8 logskewcdf

78.9 logskewpdf

79 statistics/logu

79.1 lambertw_numeric

```
lambert-w function
```

79.2 logtrialtcdf

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

79.3 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))))
```

79.4 logtrialtmean

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

79.5 logtrialtpdf

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

79.6 logtrialtrnd

79.7 logtricdf

```
cumulative distribution of the logarithmic triangular distribution
```

79.8 logtriinv

```
invere of the logarithmic triangular distribution
```

79.9 logtrimean

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

79.10 logtripdf

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

79.11 logtrirnd

79.12 logucdf

```
probability density of the logarithmic uniform distribution
```

79.13 logucm

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

79.14 loguinv

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

79.15 logumean

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

79.16 logupdf

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

79.17 logurnd

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

79.18 loguvar

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

79.19 medlogu

```
median of the log-uniform distribution
```

79.20 test_logurnd

79.21 tricdf

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

79.22 triinv

```
inverse of the triangular distribution
```

79.23 trimedian

```
median of the triangular distribution
```

79.24 tripdf

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

79.25 trirnd

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

80 statistics

80.1 max_exprnd

80.2 maxnnormals

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

80.3 mean_generalized_gampdf

80.4 midrange

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

80.5 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
```

80.6 mode_man

81 statistics/moment-statistics

81.1 autocorr_man3

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

81.2 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
```

81.3 autocorr_man5

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

81.4 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
```

81.5 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
```

81.6 corr_man

```
correlation of two vectors
```

81.7 cov_man

```
covariance matrix of two vectors
```

81.8 dof

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

81.9 edgeworth_quantile

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

81.10 effective_sample_size

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

81.11 f_correlation

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

81.12 f_finite

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

81.13 lmean

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

81.14 lmoment

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

81.15 maskmean

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

81.16 masknanmean

81.17 mean1

```
mean of x
```

81.18 mean_man

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

81.19 mse

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

81.20 nanautocorr_man1

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

81.21 nanautocorr_man2

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

81.22 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
```

81.23 nancorr

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

81.24 nancumsum

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

81.25 nanlmean

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

81.26 nanr2

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

81.27 nanrms

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

81.28 nanrmse

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

81.29 nanserr

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

81.30 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)
```

81.31 nanwstd

```
weighed standard deviation
```

81.32 nanwvar

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

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

81.33 nanxcorr

81.34 pearson

```
pearson correlation coefficient
```

81.35 pearson_to_kendall

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

81.36 pool_samples

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

81.37 qmean

```
trimmed mean
```

81.38 range_mean

81.39 rmse_

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

81.40 serr

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

81.41 serr1

81.42 test_qskew

81.43 test_qstd_qskew_optimal_p

81.44 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
```

81.45 wcorr

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

81.46 wcov

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

81.47 wdof

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

81.48 wkurt

```
kurtosis with weighted samples
```

81.49 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)
```

81.50 wrms

```
weighted root mean square
```

81.51 wserr

```
weighted root mean square error
```

81.52 wskew

```
skewness of a weighted set of samples
```

81.53 wstd

```
weighed standard deviation
```

81.54 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
```

82 statistics

82.1 nangeomean

82.2 nangeostd

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

83 statistics/nonparametric-statistics

83.1 kernel1d

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

83.2 kernel2d

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

84 statistics

84.1 normmoment

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

84.2 normpdf2

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

85 statistics/order-statistics

85.1 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)
```

85.2 kendall

```
kendall correlation coefficient
```

85.3 kendall_to_pearson

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

c.f. Kruska, 1985
```

85.4 mad2sd

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

85.5 madcorr

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

85.6 median2_holder

85.7 median_ci

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

85.8 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
```

85.9 mediani

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

85.10 nanmadcorr

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

85.11 nanwmedian

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

85.12 nanwquantile

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

85.13 oja_median

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

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

85.14 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
```

85.15 qmoments

```
moments estimated from quantiles
```

85.16 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
```

85.17 qskewq

```
skewness estimated by quantiles
```

85.18 qstdq

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

85.19 quantile1_optimisation

85.20 quantile2_breckling

```
qunatile regression
```

85.21 quantile2_chaudhuri

```
quantile regression
```

85.22 quantile2_projected

```
quantile in two dimensions
```

85.23 quantile2_projected2

```
spatial qunatile for chosen direction
```

85.24 quantile_envelope

85.25 quantile_regression_simple

```
simple quantile regression
```

85.26 ranking

```
ranking for spearman statistics
```

85.27 spatial_median

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

85.28 spatial_quantile

```
spatial quantile
```

85.29 spatial_quantile2

```
spatial quantile
```

85.30 spatial_quantile3

```
spatial quantile
```

85.31 spatial_rank

```
unsigned rank
```

85.32 spatial_sign

```
spatial sign
```

85.33 spatial_signed_rank

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

85.34 spearman

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

85.35 spearman_rank

85.36 spearman_to_pearson

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

85.37 wmedian

```
weighted median
```

85.38 wquantile

```
weighted quantile
```

86 statistics

86.1 qstd

86.2 quantile_extrap

86.3 quantile_sin

87 statistics/random-number-generation

87.1 laplacernd

```
random number of laplace distribution
```

87.2 randc

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

87.3 skewness2param

87.4 skewpdf_central_moments

87.5 skewrnd

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

87.6 skewrnd2

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

88 statistics

88.1 range

```
mid range
```

88.2 resample_with_replacement

89 statistics/resampling-statistics/@Jackknife

89.1 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
```

89.2 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
```

89.3 matrix1_STATIC

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

89.4 matrix2

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

90 statistics/resampling-statistics

90.1 block_jackknife

90.2 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
```

90.3 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
```

90.4 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
```

90.5 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
```

91 statistics

91.1 scale_quantile_sd

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

91.2 sd_sample_quantiles

91.3 skew_generalized_normal_fit

91.4 skew_generalized_normpdf

91.5 skewcdf

91.6 skewparam_to_central_moments

91.7 skewpdf

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

91.8 spatialrnd

91.9 test_mean_generalized_gampdf

91.10 test_skew_generalized_normpdf

91.11 trimmed_mean

```
trimmed mean
```

91.12 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
```

91.13 ttest_man

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

91.14 ttest_paired

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

91.15 wgeomean

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

91.16 wgeovar

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

91.17 wharmean

```
weighted harmonic mean
```

91.18 wharstd

91.19 wharvar

91.20 wnormpdf

92 mathematics

```
mathematical functions of various kind
```

92.1 ternary_diagram

93 test/master

93.1 dat_test_lanczos_3d_k_20_n_40

93.2 poisson2d_blk

93.3 qr_implicit_givens_2

93.4 spectral_derivative_2d

93.5 test_2d_eigensolver_hydrogen

93.6 test_2d_refine

93.7 test_3d_eigensolver_hydrogen

93.8 test_FEM

93.9 test_Mesh_3d

93.10 test_arnoldi

93.11 test_arpackc

93.12 test_assemble

93.13 test_assembly_performance

93.14 test_bc_one_sided

93.15 test_compare_solvers

93.16 test_complete

93.17 test_convergence

93.18 test_convergence_b

93.19 test_df_2d

93.20 test_eig_algs

93.21 test_eig_inverse

93.22 test_eigs_lanczos

93.23 test_eigs_lanczos_1

93.24 test_eigs_lanczos_2

93.25 test_eigs_lanczos_performance

93.26 test_fdm

93.27 test_fdm_d_vargrid

93.28 test_fdm_spectral

93.29 test_fem

93.30 test_fem_1d

93.31 test_fem_1d_higher_order

93.32 test_fem_2d_adaptive

93.33 test_fem_2d_higher_order

93.34 test_fem_3d_higher_order

93.35 test_fem_3d_refine

93.36 test_fem_b

93.37 test_fem_derivative

93.38 test_fem_quadrature

93.39 test_final

93.40 test_fix_substitution

93.41 test_forward

93.42 test_get_sparse_arrays

93.43 test_harmonic_oscillator

93.44 test_high_order_fdm_periodic_bc

93.45 test_hydrogen_wf

93.46 test_ichol

93.47 test_interpolation

93.48 test_inverse_problem

93.49 test_it_vs_exact

93.50 test_jama

93.51 test_jd

93.52 test_jdqz

93.53 test_lanczos_2

93.54 test_lanczos_biorthogonal

93.55 test_laplacian

93.56 test_laplacian_non_uniform

93.57 test_laplacian_simple

93.58 test_mesh_2d_uniform

93.59 test_mesh_2d_uniform_2

93.60 test_mesh_circle

93.61 test_mesh_generation

93.62 test_mesh_interpolate

93.63 test_mg

93.64 test_minres_recycle

93.65 test_multigrid

93.66 test_nc

93.67 test_nonuniform_symmetric

93.68 test_pde

93.69 test_permutation

93.70 test_poison_fem

93.71 test_polar

93.72 test_potential

93.73 test_powers

93.74 test_precondition

93.75 test_project_rectangle

93.76 test_qr

93.77 test_quantum_well

93.78 test_radial_adaptive

93.79 test_radial_confinement

93.80 test_radial_fixes

93.81 test_refine_2d

93.82 test_refine_2d_b

93.83 test_refine_3d

93.84 test_refine_structural

93.85 test_regularisation

93.86 test_round_off

93.87 test_schrödinger_potentials

93.88 test_uniform_mesh

93.89 test_vargrid

94 test

94.1 test_bandpass2d

94.2 test_gaussfit3

94.3 test_geoserr

94.4 test_hexagonal_pattern

94.5 test_lognfit_quantile

94.6 test_lowpass1d_implicit

94.7 test_lowpass2d_anisotropic

94.8 test_lowpass2d_fft

94.9 test_max_normal

94.10 test_mtimes3x3

95 wavelet

95.1 contiuous_wavelet_transform

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

95.2 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
```

95.3 example_wavelets

95.4 phasewrap

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

95.5 test_cwt_man

95.6 test_phasewrap

95.7 test_wavelet

95.8 test_wavelet2

95.9 test_wavelet_analysis

95.10 test_wavelet_reconstruct

95.11 test_wtc

95.12 wavelet

```
wavelet windows
```

95.13 wavelet_reconstruct

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

95.14 wavelet_transform

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