Manual for Package: mathematics
Revision 2:12M

Karl Kästner

August 17, 2020

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_ar2param

4.3 derive_arc_length

4.4 derive_fourier_power

4.5 derive_fourier_power_exp

4.6 derive_laplacian_curvilinear

4.7 derive_laplacian_fourier_piecewise_linear

4.8 derive_logtripdf

4.9 derive_smooth1d_parametric

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

7.2 exp10

7.3 filter_twosided

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_expand

```
expand values of fourier series
```

11.19 fourier_fit

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

11.20 fourier_interpolate

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

11.21 fourier_matrix

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

11.22 fourier_matrix2

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

11.23 fourier_matrix3

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

11.24 fourier_matrix_exp

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

11.25 fourier_multiplicative_interaction_coefficients

11.26 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
```

11.27 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
```

11.28 fourier_predict

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

11.29 fourier_quadratic_interaction_coefficients

11.30 fourier_range

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

11.31 fourier_regress

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

11.32 fourier_resampled_fit

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

11.33 fourier_resampled_predict

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

11.34 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
```

11.35 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
```

11.36 hyperbolic_fourier_box

11.37 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)
```

11.38 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
```

11.39 nanfft

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

11.40 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)
```

11.41 roots_fourier

```
zeros of continuous fourier series series

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

11.42 spectral_density

```
spectral density
```

11.43 test_complex_exp_product

11.44 test_fourier_filter

11.45 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
```

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
```

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 histogram/@Histogram

15.1 2x

15.2 Histogram

15.3 bimodes

15.4 cdf

15.5 cdfS

15.6 chi2test

15.7 cmoment

15.8 cmomentS

15.9 entropy

15.10 entropyS

15.11 iquantile

15.12 kstest

15.13 kurtosis

15.14 kurtosisS

15.15 mean

15.16 meanS

15.17 median

15.18 medianS

15.19 mode

15.20 modeS

15.21 moment

15.22 momentS

15.23 pdf

15.24 quantile

15.25 quantileS

15.26 setup

15.27 skewness

15.28 skewnessS

15.29 stairs

15.30 stairsS

15.31 std

15.32 stdS

15.33 var

15.34 varS

16 histogram

16.1 hist_man

16.2 histadapt

16.3 histconst

16.4 pdf_poly

16.5 plotcdf

16.6 test_histogram

17 linear-algebra

17.1 averaging_matrix_2

17.2 colnorm

```
norms of columns
```

17.3 condest_

```
estimation of the condition number
```

18 linear-algebra/coordinate-transformation

18.1 barycentric2cartesian

```
barycentric to cartesian coordinates
```

18.2 barycentric2cartesian3

```
convert barycentric to cartesian coordinates
```

18.3 cartesian2barycentric

```
cartesian to barycentric coordinates
```

18.4 cartesian_to_unit_triangle_basis

```
transform coodinates into unit triangle
```

18.5 ellipsoid2geoid

18.6 example_approximate_utm_conversion

18.7 latlon2utm

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

18.8 latlon2utm_simple

18.9 lowrance_mercator_to_wgs84

```
convert lowrance coordinates to wgs84

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

18.10 nmea2utm

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

18.11 sn2xy

```
convert sn to xy coordinates
```

18.12 unit_triangle_to_cartesian

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

18.13 utm2latlon

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

18.14 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
```

18.15 xy2sn

```
convert cartesian to streamwise coordiantes
```

18.16 xy2sn_java

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

18.17 xy2sn_old

```
transform points from cartesian into streamwise coordinates

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

19 linear-algebra

19.1 det2x2

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

19.2 det3x3

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

19.3 det4x4

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

19.4 diag2x2

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

19.5 eig2x2

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

20 linear-algebra/eigenvalue

20.1 eig_bisection

20.2 eig_inverse

20.3 eig_inverse_iteration

20.4 eig_power_iteration

21 linear-algebra/eigenvalue/jacobi-davidson

21.1 afun_jdm

21.2 davidson

21.3 jacobi_davidson

21.4 jacobi_davidson_qr

21.5 jacobi_davidson_qz

21.6 jacobi_davidson_simple

21.7 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'
```

21.8 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'
```

21.9 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
```

21.10 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
    ===========================================
%=========================================================================

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

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

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

21.11 mfunc_jdm

21.12 mgs

21.13 minres_

21.14 mv_jacobi_davidson

22 linear-algebra

22.1 first

22.2 gershgorin_circle

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

22.3 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
```

22.4 ieig2x2

```
reconstruct matrix from eigenvalue decomposition
```

22.5 inv2x2

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

22.6 inv3x3

22.7 inv4x4

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

23 linear-algebra/lanczos

23.1 arnoldi

23.2 arnoldi_new

23.3 eigs_lanczos_man

23.4 lanczos

23.5 lanczos_

23.6 lanczos_biorthogonal

23.7 lanczos_biorthogonal_improved

23.8 lanczos_ghep

23.9 mv_lanczos

23.10 reorthogonalise

23.11 test_lanczos

24 linear-algebra/linear-systems

24.1 gmres_man

```
break on convergence
```

24.2 minres_recycle

25 linear-algebra

25.1 lpmean

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

25.2 lpnorm

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

25.3 matvec3

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

25.4 max2d

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

25.5 mid

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

25.6 mpoweri

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

25.7 mtimes2x2

25.8 mtimes3x3

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

25.9 nannorm

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

25.10 nanshift

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

25.11 nl

```
number rows (lines) of a matrix

analogue to unix nl command
```

25.12 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)
```

25.13 normalize1

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

25.14 normrows

25.15 orth2

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

25.16 orth_man

```
orthogonalize the columns of A
```

25.17 orthogonalise

```
make x orthogonal to Y
```

25.18 paddext

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

25.19 paddval1

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

25.20 paddval2

```
padd values to x
```

26 linear-algebra/polynomial

26.1 chebychev

```
chebycheff polynomials
```

26.2 piecewise_polynomial

```
evaluate piecewise polynomial
```

26.3 roots1

```
roots of linear functions
```

26.4 roots2

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

26.5 roots2poly

26.6 roots3

26.7 roots4

26.8 roots_piecewise_linear

26.9 test_roots4

26.10 vanderi_1d

```
vandermonde matrix of an integral
```

27 linear-algebra

27.1 randrot

```
random rotation matrix
```

27.2 right

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

27.3 rot2

```
rotation matrix from angle
```

27.4 rot2dir

```
rotation matrix from direction vector
```

27.5 rot3

27.6 rotR

27.7 rownorm

27.8 simmilarity_matrix

27.9 spnorm

```
frobenius norm
```

27.10 spzeros

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

27.11 test_roots3

27.12 transform_minmax

27.13 transpose3

```
transpose stacked 3x3 matrices
```

27.14 transposeall

28 logic

```
bitwise operations on integers
```

28.1 bitor_man

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

 input:
       A (positive integer)
```

29 master/plot

29.1 attach_boundary_value

29.2 cartesian_polar

29.3 img_vargrid

29.4 plot_basis_functions

29.5 plot_convergence

29.6 plot_dof

29.7 plot_eigenbar

29.8 plot_error_estimation

29.9 plot_error_estimation_2

29.10 plot_error_fem

29.11 plot_fdm_kernel

29.12 plot_fdm_vs_fem

29.13 plot_fem_accuracy

29.14 plot_function_and_grid

29.15 plot_hat

29.16 plot_hydrogen_wf

29.17 plot_mesh

29.18 plot_mesh_2

29.19 plot_refine

29.20 plot_refine_3d

29.21 plot_runtime

29.22 plot_spectrum

29.23 plot_wavefunction

30 master/ported

30.1 assemble_2d_phi_phi

30.2 assemble_3d_dphi_dphi

30.3 assemble_3d_phi_phi

30.4 dV_2d_

30.5 derivative_2d

30.6 derivative_3d

30.7 element_neighbour_2d

30.8 prefetch_2d_

30.9 promote_2d_3_10

30.10 promote_2d_3_15

30.11 promote_2d_3_21

30.12 promote_2d_3_6

30.13 promote_3d_4_10

30.14 promote_3d_4_20

30.15 promote_3d_4_35

30.16 vander_2d

30.17 vander_3d

31 mathematics

mathematical functions of various kind

31.1 myexp

32 number-theory

32.1 ceiln

floor to leading n-digits

32.2 digitsb

number of digits with respect to specified base

32.3 floorn

floor to n-digits

32.4 iseven

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

32.5 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
```

32.6 nchoosek_man

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

32.7 pythagorean_triple

```
pythagorean triple
```

32.8 roundn

```
round to n digits
```

33 numerical-methods/differentiation

33.1 derivative1

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

33.2 derivative2

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

34 numerical-methods/finite-difference

34.1 cdiff

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

34.2 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
```

34.3 cmean

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

34.4 cmean2

34.5 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)
```

34.6 derivative_matrix_2_1d

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

34.7 derivative_matrix_2d

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

34.8 derivative_matrix_curvilinear

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

34.9 derivative_matrix_curvilinear_2

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

34.10 difference_kernel

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

34.11 distmat

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

34.12 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
```

34.13 laplacian

34.14 laplacian_fdm

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

34.15 left

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

34.16 lrmean

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

35 numerical-methods/finite-difference/master

35.1 fdm_adaptive_grid

35.2 fdm_adaptive_refinement_old

35.3 fdm_assemble_d1_2d

35.4 fdm_assemble_d2_2d

35.5 fdm_confinement

35.6 fdm_d_vargrid

35.7 fdm_h_unstructured

35.8 fdm_hydrogen_vargrid

35.9 fdm_mark_unstructured_2d

35.10 fdm_plot

35.11 fdm_plot_series

35.12 fdm_refine_2d

35.13 fdm_refine_3d

35.14 fdm_refine_unstructured_2d

35.15 fdm_schroedinger_2d

35.16 fdm_schroedinger_3d

35.17 relocate

36 numerical-methods/finite-difference

36.1 mid

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

36.2 pwmid

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

36.3 ratio

```
ratio of two subsequent values
```

36.4 steplength

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

36.5 swapoddeven

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

36.6 test_derivative_matrix_2d

36.7 test_derivative_matrix_curvilinear

36.8 test_difference_kernel

37 numerical-methods/finite-element

37.1 Mesh_2d_java

37.2 Tree_2d_java

37.3 assemble_1d_dphi_dphi

37.4 assemble_1d_phi_phi

37.5 assemble_2d_dphi_dphi_java

37.6 assemble_2d_phi_phi_java

37.7 assemble_3d_dphi_dphi_java

37.8 assemble_3d_phi_phi_java

37.9 boundary_1d

37.10 boundary_2d

37.11 boundary_3d

37.12 check_area_2d

37.13 circmesh

37.14 cropradius

37.15 display_2d

37.16 display_3d

37.17 distort

37.18 err_2d

37.19 estimate_err_2d_3

37.20 example_1d

37.21 example_2d

37.22 explode

37.23 fem_2d

37.24 fem_2d_heuristic_mesh

37.25 fem_get_2d_radial

37.26 fem_interpolation

37.27 fem_plot_1d

37.28 fem_plot_1d_series

37.29 fem_plot_2d

37.30 fem_plot_2d_series

37.31 fem_plot_3d

37.32 fem_plot_3d_series

37.33 fem_plot_confine_series

37.34 fem_radial

```
adaptive grid
constant grid
```

37.35 flip_2d

37.36 get_mesh_arrays

37.37 hashkey

38 numerical-methods/finite-element/int

38.1 int_1d_gauss

38.2 int_1d_gauss_1

38.3 int_1d_gauss_2

38.4 int_1d_gauss_3

38.5 int_1d_gauss_4

38.6 int_1d_gauss_5

38.7 int_1d_gauss_6

38.8 int_1d_gauss_lobatto

38.9 int_1d_gauss_n

38.10 int_1d_nc_2

38.11 int_1d_nc_3

38.12 int_1d_nc_4

38.13 int_1d_nc_5

38.14 int_1d_nc_6

38.15 int_1d_nc_7

38.16 int_1d_nc_7_hardy

38.17 int_2d_gauss_1

38.18 int_2d_gauss_12

38.19 int_2d_gauss_13

38.20 int_2d_gauss_16

38.21 int_2d_gauss_19

38.22 int_2d_gauss_25

38.23 int_2d_gauss_3

38.24 int_2d_gauss_33

38.25 int_2d_gauss_4

38.26 int_2d_gauss_6

38.27 int_2d_gauss_7

38.28 int_2d_gauss_9

38.29 int_2d_nc_10

38.30 int_2d_nc_15

38.31 int_2d_nc_21

38.32 int_2d_nc_3

38.33 int_2d_nc_6

38.34 int_3d_gauss_1

38.35 int_3d_gauss_11

38.36 int_3d_gauss_14

38.37 int_3d_gauss_15

38.38 int_3d_gauss_24

38.39 int_3d_gauss_4

38.40 int_3d_gauss_45

38.41 int_3d_gauss_5

38.42 int_3d_nc_11

38.43 int_3d_nc_4

38.44 int_3d_nc_6

38.45 int_3d_nc_8

39 numerical-methods/finite-element

39.1 interpolation_matrix

39.2 mark

39.3 mark_1d

39.4 mesh_1d_uniform

39.5 mesh_3d_uniform

39.6 mesh_interpolate

39.7 neighbour_1d

39.8 old

39.9 pdeeig_1d

39.10 pdeeig_2d

39.11 pdeeig_3d

39.12 polynomial_derivative_1d

39.13 potential_const

39.14 potential_coulomb

39.15 potential_harmonic_oscillator

39.16 project_circle

39.17 project_rectangle

39.18 promote_1d_2_3

39.19 promote_1d_2_4

39.20 promote_1d_2_5

39.21 promote_1d_2_6

39.22 quadrilaterate

39.23 recalculate_regularity_2d

39.24 refine_1d

39.25 refine_2d_21

39.26 refine_2d_structural

39.27 regularity_1d

39.28 regularity_2d

39.29 regularity_3d

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

39.30 relocate_2d

39.31 test_circmesh

39.32 test_hermite

39.33 tri_assign_points

39.34 triangulation_uniform

39.35 vander_1d

```
van der Monde matrix
```

39.36 vanderd_1d

39.37 vanderi_1d

40 numerical-methods/finite-volume/@Advection

40.1 Advection

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

40.2 dot_advection

```
advection equation
```

41 numerical-methods/finite-volume/@Burgers

41.1 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
```

41.2 dot_burgers_fdm

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

41.3 dot_burgers_fft

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

42 numerical-methods/finite-volume/@Finite_Volume

42.1 Finite_Volume

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

42.2 apply_bc

```
apply boundary conditions
```

42.3 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);
```

42.4 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
```

42.5 step_unsplit

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

43 numerical-methods/finite-volume/@Flux_Limiter

43.1 Flux_Limiter

```
class of flux limiters
```

43.2 beam_warming

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

43.3 fromm

```
fromme limiter
low res
```

43.4 lax_wendroff

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

43.5 minmod

```
min-mod schock limiter
```

43.6 monotized_central

```
monotonized central flux limiter
```

43.7 muscl

```
muscl flux limiter
```

43.8 superbee

```
superbee limiter
```

43.9 upwind

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

43.10 vanLeer

```
van Leer limiter
```

44 numerical-methods/finite-volume/@KDV

44.1 dot_kdv_fdm

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

44.2 dot_kdv_fft

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

44.3 kdv_split

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

45 numerical-methods/finite-volume/@Reconstruct_Average_Evolve

45.1 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;
```

45.2 advect_highres

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

45.3 advect_lowress

```
single time step
low resolution
```

46 numerical-methods/finite-volume

46.1 Godunov

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

46.2 Lax_Friedrich

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

46.3 Measure

46.4 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
```

46.5 fv_swe

```
wrapper for solving SWE
```

46.6 staggered_euler

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

46.7 staggered_grid

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

47 numerical-methods

47.1 grid2quad

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

48 numerical-methods/integration

48.1 cumintL

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

48.2 cumintR

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

48.3 int_trapezoidal

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

49 numerical-methods/interpolation/@Kriging

49.1 Kriging

```
class for Kriging interpolation
```

49.2 estimate_semivariance

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

49.3 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
```

50 numerical-methods/interpolation/@RegularizedInterpolator

50.1 RegularizedInterpolator1

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

50.2 init

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

51 numerical-methods/interpolation/@RegularizedInterpolator

51.1 RegularizedInterpolator2

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

51.2 init

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

52 numerical-methods/interpolation/@RegularizedInterpolator3

52.1 RegularizedInterpolator3

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

52.2 init

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

53 numerical-methods/interpolation

53.1 IDW

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

53.2 IPoly

```
polynomial interpolation class
```

53.3 IRBM

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

53.4 ISparse

```
sparse interpolation class
```

53.5 Inn

```
nearest neighbour interpolation
```

53.6 Interpolator

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

53.7 fixnan

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

53.8 idw1

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

53.9 idw2

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

53.10 inner2outer

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

53.11 inner2outer2

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

53.12 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
```

53.13 interp1_man

```
interpolate
```

53.14 interp1_piecewise_linear

53.15 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
```

53.16 interp1_slope

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

53.17 interp1_smooth

53.18 interp1_unique

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

53.19 interp2_man

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

53.20 interp_angle

```
interpolate an angle
```

53.21 interp_fourier

```
interpolation by the fourier method
```

53.22 interp_fourier_batch

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

53.23 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_);
```

53.24 interp_sn2

```
interpolation in streamwise coordinates
```

53.25 interp_sn3

53.26 interp_sn_

53.27 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)
```

53.28 resample1

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

53.29 resample_d_min

```
resample a function
```

53.30 resample_vector

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

53.31 test_interp1_limited

54 numerical-methods

54.1 inverse_complex

54.2 maccormack_step

55 numerical-methods/ode

55.1 bvp1c

55.2 bvp1c_assemble

55.3 bvp2_check_arguments

55.4 bvp2c

```
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])
```

55.5 bvp2c2

```
solve second order boundary value problem via roots of the
    characteristic
polynomial

input:

x   : [nx1] discretized domain
      n : number of vertices
```

```
    nxc = n-1 : number of segments

bc  : struct : boundary condition
      bc.p(1)*y(0) + bc.pd(2)*y'(0) = bc.val(1)
      bc.p(2)*y(L) + bc.pd(2)*y'(L) = bc.val(2)

output:

A   : [2*nxc x 2*ns] disrcretisation matrix
rhs : [2*nxc x 1] right hand size

y = A^-1 rhs
```

55.6 bvp2c_assemble

55.7 bvp2c_derivative

55.8 bvp2c_resample

55.9 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])
```

55.10 bvp2wavetrain

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

55.11 bvp2wavetwopass

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

55.12 bvp_row

55.13 ivp_euler_forward

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

55.14 ivprk2

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

55.15 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]
```

55.16 ode2characteristic

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

55.17 ode_adams_bashforth

55.18 step_trapezoidal

```
single trapezoidal step
```

55.19 test_bvp2

56 numerical-methods/optimisation

56.1 armijo_stopping_criterion

```
armijo stopping criterion for optimizations
```

56.2 astar

```
astar path finding alforithm
```

56.3 binsearch

```
binary search on a line
```

56.4 bisection

```
bisection
```

56.5 box1

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

56.6 box2

56.7 cauchy

56.8 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
```

56.9 directional_derivative

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

56.10 dud

```
optimization by the dud algorithm
```

56.11 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
```

56.12 extreme_quadratic

56.13 ftest

56.14 fzero_bisect

56.15 fzero_newton

56.16 grad

```
numerical gradient
```

56.17 hessian

```
numerical hessian
```

56.18 hessian_from_gradient

```
numerical hessian from gradient
```

56.19 hessian_projected

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

56.20 line_search

```
bisection routine
```

56.21 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
```

56.22 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
```

56.23 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
```

56.24 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
```

56.25 line_search_quadratic2

```
quadratic line search
```

56.26 line_search_wolfe

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

56.27 ls_bgfs

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

56.28 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
```

56.29 ls_generalized_secant

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

56.30 nlcg

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

56.31 nlls

```
non-linear least squares
```

56.32 picard

```
picard iteration
```

56.33 poly_extrema

```
extrema of a polynomial
```

56.34 quadratic_function

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

56.35 quadratic_programming

```
optimize by quadratic programming
```

56.36 quadratic_step

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

56.37 rosenbrock

```
rosenbrock test function
```

56.38 sqrt_heron

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

56.39 test_directional_derivative

56.40 test_dud

56.41 test_fzero_newton

56.42 test_line_search_quadratic2

56.43 test_ls_generalized_secant

56.44 test_nlcg_6_order

56.45 test_nlls

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

57 numerical-methods/pde

57.1 laplacian2d_fundamental_solution

58 numerical-methods/piecewise-polynomials

58.1 Hermite1

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

58.2 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)
```

58.3 hp2_predict

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

58.4 hp_predict

```
predict with piecewise hermite polynomial
```

58.5 hp_regress

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

58.6 lp_count

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

58.7 lp_predict

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

58.8 lp_regress

58.9 lp_regress_

59 numerical-methods

59.1 test_adams_bashforth

60 regression/@PolyOLS

60.1 PolyOLS

```
class for polynomial least squares
```

60.2 coefftest

60.3 detrend

```
detrending by polynomial regression
```

60.4 fit

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

60.5 fit_

```
fit a polynomial function
```

60.6 predict

```
predict polynomial function values
```

60.7 predict_

60.8 slope

```
slope by linear regression
```

61 regression/@PowerLS

61.1 PowerLS

```
class for power law regression
```

61.2 fit

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

61.3 predict

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

61.4 predict_

62 regression/@Theil

62.1 Theil

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

62.2 detrend

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

62.3 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
```

62.4 predict

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

62.5 slope

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

63 regression

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

63.1 Theil_Multivariate

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

63.2 areg

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

63.3 ginireg

```
gini regression
```

63.4 hesssimplereg

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

63.5 l1lin

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

63.6 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)
```

63.7 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
```

63.8 regression_method_of_moments

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

63.9 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
```

63.10 theil2

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

63.11 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
```

63.12 total_least_squares

```
total least squares
```

63.13 weighted_median_regression

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

64 set-theory

64.1 issubset

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

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

65 signal-processing

65.1 acf_effective_sample_size

```
effective sample size from acf
```

65.2 acf_genton

```
autocorrelation function
```

65.3 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
```

65.4 acfar1_2

```
autocorrelation of the ar1 process
```

65.5 acfar2

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

65.6 acfar2_2

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

65.7 ar1_cutoff_frequency

65.8 ar1_effective_sample_size

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

65.9 ar1_mse_mu_single_sample

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

65.10 ar1_mse_pop

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

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

65.11 ar1_mse_range

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

65.12 ar1_spectrum

```
spectrum of the ar1 process
```

65.13 ar1_to_tikhonov

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

65.14 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
```

65.15 ar1_var_factor_

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

65.16 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|
```

65.17 ar1delay

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

65.18 ar1delay_old

```
autocorrelation of the residual
```

65.19 ar2conv

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

65.20 ar2dof

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

65.21 ar2param

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

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

65.22 asymwin

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

65.23 autocorr_fft

```
autocorrelation function
```

65.24 bandpass

```
bandpass filter
```

65.25 bandpass2

```
bandpass filter
```

65.26 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
```

65.27 bartlett_spectrogram

```
bartlet spectrogramm
TODO sliding window
```

65.28 bin1d

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

65.29 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
```

65.30 binormrnd

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

65.31 conv1_man

```
convolutions with padding
```

65.32 conv2_man

```
convolution in 2d
```

65.33 conv2z

65.34 conv30

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

65.35 conv_

```
convolution of a with b
```

65.36 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
```

65.37 convz

65.38 cosexpdelay

65.39 csmooth

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

65.40 daniell_window

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

65.41 danielle_window

```
danielle fourier window
```

65.42 db2neper

```
convert decibel to neper
```

65.43 db2power

```
power ratio from db
```

65.44 derive_danielle_weight

65.45 derive_limit_0_acfar

65.46 detect_peak

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

65.47 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)
```

65.48 doublesum_ij

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

65.49 effective_sample_size_to_ar1

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

65.50 filt_hodges_lehman

65.51 filter1

```
filter along one dimension
```

65.52 filter2

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

65.53 filter_

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

65.54 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
```

65.55 filterp

65.56 filterp1

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

65.57 filterstd

65.58 firls_man

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

65.59 flattopwin

```
the flat top window
```

65.60 frequency_response_boxcar

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

65.61 freqz_boxcar

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

65.62 gaussfilt1

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

65.63 hanchangewin

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

65.64 hanchangewin2

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

65.65 hanwin

```
hanning filter window
```

65.66 hanwin_

```
hanning filter window
```

65.67 highpass

```
high pass filter
```

65.68 kaiserwin

```
kaiser filter window
```

65.69 kalman

```
Kalman filter
```

65.70 lanczoswin

```
Lanczos window
```

65.71 last

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

65.72 lowpass

```
low pass filter
```

65.73 lowpass2

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

65.74 lowpass_iir

```
iir-low pass
```

65.75 lowpass_iir_symmetric

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

65.76 lowpassfilter2

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

65.77 maxfilt1

65.78 meanfilt1

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

65.79 medfilt1_man

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

65.80 medfilt1_man2

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

65.81 medfilt1_padded

```
median filter with padding
```

65.82 medfilt1_reduced

```
median filter with padding
```

65.83 mid_term_single_sample

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

65.84 minfilt1

65.85 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
```

65.86 mysmooth

65.87 nanautocorr

```
autocorrelation with nan-values
```

65.88 nanmedfilt1

```
medfilt1, skipping nans
```

65.89 neper2db

```
convert neper to db
```

65.90 peaks_man

```
peaks of a periodogram
```

65.91 polyfilt1

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

65.92 qmedfilt1

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

65.93 randar1

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

65.94 randar1_dual

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

65.95 randar2

```
generate ar2 process
```

65.96 randarp

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

65.97 range_window

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

65.98 rectwin

```
rectangular window
```

65.99 recursive_sum

65.100 select_range

65.101 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
```

65.102 smooth2

```
smooth vectos of X
```

65.103 smooth_man

65.104 smooth_parametric

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

65.105 smooth_parametric2

```
parametrically smooth the curve
```

65.106 smooth_with_splines

65.107 smoothfft

```
filter with fast fourier transform
```

65.108 spectrogram

```
spectrogram
```

65.109 std_window

```
moving block standard deviation
```

65.110 sum_i_lag

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

65.111 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
```

65.112 sum_ii_

65.113 sum_ij

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

65.114 sum_ij_

65.115 sum_ij_partial_

65.116 sum_multivar

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

65.117 test_acfar1

65.118 test_acfar1_2

65.119 test_acfar1_3

65.120 test_acfar1_4

65.121 test_acfar2

65.122 test_ar1_var_factor

65.123 test_ar1_var_factor_2

65.124 test_ar1_var_mu_single_sample

65.125 test_ar1_var_pop

65.126 test_ar1_var_pop_1

65.127 test_ar1delay

65.128 test_bivariate_covariance_term

65.129 test_convexity

65.130 test_lanczoswin

65.131 test_madcorr

65.132 test_randar1

65.133 test_randar1_multivariate

65.134 test_randar2

65.135 test_sum_ij

65.136 test_sum_multivar

65.137 test_trifilt1

65.138 test_wautocorr

65.139 test_wavelet_transform

65.140 test_wordfilt

65.141 test_xar1_mid_term

65.142 tikhonov_to_ar1

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

65.143 trapwin

```
trapezoidal filter window
```

65.144 trifilt1

```
filter with triangular window
```

65.145 triwin

```
triangular filter window
```

65.146 triwin2

```
triangular filter window
```

65.147 varar1

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

65.148 welch_spectrogram

```
welch spectrogram
```

65.149 wfilt

```
filter with window
```

65.150 winbandpass

```
filter with bandpass
```

65.151 window_make_odd

65.152 winfilt0

```
filter with window
```

65.153 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
```

65.154 wmeanfilt

```
mean filter with window
```

65.155 wmedfilt

```
median filter with window
```

65.156 wordfilt

```
weighted order filter
```

65.157 wordfilt_edgeworth

```
weighed order filter
```

65.158 xar1

65.159 xcorr_man

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

66 sorting

66.1 sort2

```
sort two numbers
```

66.2 sort2d

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

67 special-functions

67.1 bessel_sphere

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

67.2 digamma_man

67.3 hankel_sphere

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

67.4 hermite

```
probabilistic's hermite polynomial by recurrence relation

input :
n : order
x : value

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

67.5 legendre_man

```
legendre polynomials
```

67.6 neumann_sphere

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

68 statistics

68.1 atan_s2

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

68.2 beta_mode_to_parameter

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

68.3 coefficient_of_determination

68.4 conditional_expectation_normal

68.5 correlation_confidence_pearson

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

69 statistics/distributions

69.1 PDF

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

69.2 binorm_separation_coefficient

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

69.3 binormcdf

```
bio-modal gaussian distribution
```

69.4 binormfit

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

69.5 binormpdf

69.6 edgeworth_cdf

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

69.7 edgeworth_pdf

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

69.8 logn_mode2param

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

69.9 logn_param2mode

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

69.10 lognpdf_

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

69.11 pdfsample

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

69.12 t2cdf

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

69.13 t2inv

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

70 statistics

70.1 example_standard_error_of_sample_quantiles

70.2 f_var_finite

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

70.3 gamma_mode_to_parameter

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

70.4 gaussfit3

70.5 gaussfit_quantile

70.6 geoserr

70.7 geostd

70.8 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
```

70.9 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
```

71 statistics/information-theory

71.1 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
```

71.2 bayesian_information_criterion

```
bayesian information criterion
```

72 statistics

72.1 kurtncdf

72.2 kurtnpdf

72.3 kurtosis_bias_corrected

```
bias corrected kurtosis
```

72.4 limit

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

72.5 logfactorial

```
approximate log of the factorial
```

72.6 loglogpdf

72.7 lognfit_quantile

72.8 logskewcdf

72.9 logskewpdf

73 statistics/logu

73.1 lambertw_numeric

```
lambert-w function
```

73.2 logtrialtcdf

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

73.3 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))))
```

73.4 logtrialtmean

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

73.5 logtrialtpdf

`density of the logarithmic triangular distribution`

73.6 logtrialtrnd

73.7 logtricdf

`cumulative distribution of the logarithmic triangular distribution`

73.8 logtriinv

`invere of the logarithmic triangular distribution`

73.9 logtrimean

`mean of the logarithmic triangular distribution`

73.10 logtripdf

`probability density of the logarithmic triangular distribution`

73.11 logtrirnd

73.12 logucdf

`probability density of the logarithmic uniform distribution`

73.13 logucm

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

73.14 loguinv

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

73.15 logumean

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

73.16 logupdf

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

73.17 logurnd

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

73.18 loguvar

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

73.19 medlogu

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

73.20 test_logurnd

73.21 tricdf

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

73.22 triinv

```
inverse of the triangular distribution
```

73.23 trimedian

```
median of the triangular distribution
```

73.24 tripdf

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

73.25 trirnd

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

74 statistics

74.1 max_exprnd

74.2 maxnnormals

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

74.3 mean_generalized_gampdf

74.4 midrange

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

74.5 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
```

74.6 mode_man

75 statistics/moment-statistics

75.1 autocorr_man3

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

75.2 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
```

75.3 autocorr_man5

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

75.4 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
```

75.5 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
```

75.6 corr_man

```
correlation of two vectors
```

75.7 cov_man

```
covariance matrix of two vectors
```

75.8 dof

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

75.9 edgeworth_quantile

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

75.10 effective_sample_size

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

75.11 f_correlation

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

75.12 f_finite

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

75.13 lmean

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

75.14 lmoment

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

75.15 maskmean

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

75.16 masknanmean

75.17 mean1

```
mean of x
```

75.18 mean_man

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

75.19 mse

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

75.20 nanautocorr_man1

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

75.21 nanautocorr_man2

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

75.22 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
```

75.23 nancorr

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

75.24 nancumsum

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

75.25 nanlmean

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

75.26 nanr2

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

75.27 nanrms

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

75.28 nanrmse

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

75.29 nanserr

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

75.30 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)
```

75.31 nanwstd

```
weighed standard deviation
```

75.32 nanwvar

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

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

75.33 nanxcorr

75.34 pearson

```
pearson correlation coefficient
```

75.35 pearson_to_kendall

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

75.36 pool_samples

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

75.37 qmean

```
trimmed mean
```

75.38 range_mean

75.39 rmse_

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

75.40 serr

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

75.41 serr1

75.42 test_qskew

75.43 test_qstd_qskew_optimal_p

75.44 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
```

75.45 wcorr

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

75.46 wcov

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

75.47 wdof

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

75.48 wkurt

```
kurtosis with weighted samples
```

75.49 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)
```

75.50 wrms

```
weighted root mean square error
```

75.51 wserr

```
weighted root mean square error
```

75.52 wskew

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

75.53 wstd

```
weighed standard deviation
```

75.54 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
```

76 statistics

76.1 nangeomean

76.2 nangeostd

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

77 statistics/nonparametric-statistics

77.1 kernel1d

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

77.2 kernel2d

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

78 statistics

78.1 normmoment

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

78.2 normpdf2

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

79 statistics/order-statistics

79.1 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)
```

79.2 kendall

```
kendall correlation coefficient
```

79.3 kendall_to_pearson

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

c.f. Kruska, 1985
```

79.4 mad2sd

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

79.5 madcorr

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

79.6 median2_holder

79.7 median_ci

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

79.8 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
```

79.9 mediani

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

79.10 nanmadcorr

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

79.11 nanwmedian

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

79.12 nanwquantile

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

79.13 oja_median

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

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

79.14 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
```

79.15 qmoments

```
moments estimated from quantiles
```

79.16 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
```

79.17 qskewq

```
skewness estimated by quantiles
```

79.18 qstdq

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

79.19 quantile1_optimisation

79.20 quantile2_breckling

```
qunatile regression
```

79.21 quantile2_chaudhuri

```
quantile regression
```

79.22 quantile2_projected

```
quantile in two dimensions
```

79.23 quantile2_projected2

```
spatial qunatile for chosen direction
```

79.24 quantile_envelope

79.25 quantile_regression_simple

```
simple quantile regression
```

79.26 ranking

```
ranking for spearman statistics
```

79.27 spatial_median

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

79.28 spatial_quantile

```
spatial quantile
```

79.29 spatial_quantile2

```
spatial quantile
```

79.30 spatial_quantile3

```
spatial quantile
```

79.31 spatial_rank

```
unsigned rank
```

79.32 spatial_sign

```
spatial sign
```

79.33 spatial_signed_rank

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

79.34 spearman

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

79.35 spearman_rank

79.36 spearman_to_pearson

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

79.37 wmedian

```
weighted median
```

79.38 wquantile

```
weighted quantile
```

80 statistics

80.1 qstd

80.2 quantile_extrap

81 statistics/random-number-generation

81.1 laplacernd

```
random number of laplace distribution
```

81.2 randc

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

81.3 skewness2param

81.4 skewpdf_central_moments

81.5 skewrnd

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

81.6 skewrnd2

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

82 statistics

82.1 range

```
mid range
```

82.2 resample_with_replacement

83 statistics/resampling-statistics/@Jackknife

83.1 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
```

83.2 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
```

83.3 matrix1_STATIC

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

83.4 matrix2

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

84 statistics/resampling-statistics

84.1 block_jackknife

84.2 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
```

84.3 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
```

84.4 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
```

84.5 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
```

85 statistics

85.1 scale_quantile_sd

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

85.2 sd_sample_quantiles

85.3 skewpdf

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

85.4 test_mean_generalized_gampdf

85.5 trimmed_mean

```
trimmed mean
```

85.6 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
```

85.7 ttest_man

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

85.8 ttest_paired

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

85.9 wgeomean

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

85.10 wgeovar

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

85.11 wharmean

```
weighted harmonic mean
```

85.12 wharstd

85.13 wharvar

86 mathematics

`mathematical functions of various kind`

86.1 ternary_diagram

87 test/master

87.1 dat_test_lanczos_3d_k_20_n_40

87.2 poisson2d_blk

87.3 qr_implicit_givens_2

87.4 spectral_derivative_2d

87.5 test_2d_eigensolver_hydrogen

87.6 test_2d_refine

87.7 test_3d_eigensolver_hydrogen

87.8 test_FEM

87.9 test_Mesh_3d

87.10 test_arnoldi

87.11 test_arpackc

87.12 test_assemble

87.13 test_assembly_performance

87.14 test_bc_one_sided

87.15 test_compare_solvers

87.16 test_complete

87.17 test_convergence

87.18 test_convergence_b

87.19 test_df_2d

87.20 test_eig_algs

87.21 test_eig_inverse

87.22 test_eigs_lanczos

87.23 test_eigs_lanczos_1

87.24 test_eigs_lanczos_2

87.25 test_eigs_lanczos_performance

87.26 test_fdm

87.27 test_fdm_d_vargrid

87.28 test_fdm_spectral

87.29 test_fem

87.30 test_fem_1d

87.31 test_fem_1d_higher_order

87.32 test_fem_2d_adaptive

87.33 test_fem_2d_higher_order

87.34 test_fem_3d_higher_order

87.35 test_fem_3d_refine

87.36 test_fem_b

87.37 test_fem_derivative

87.38 test_fem_quadrature

87.39 test_final

87.40 test_fix_substitution

87.41 test_forward

87.42 test_get_sparse_arrays

87.43 test_harmonic_oscillator

87.44 test_high_order_fdm_periodic_bc

87.45 test_hydrogen_wf

87.46 test_ichol

87.47 test_interpolation

87.48 test_inverse_problem

87.49 test_it_vs_exact

87.50 test_jama

87.51 test_jd

87.52 test_jdqz

87.53 test_lanczos_2

87.54 test_lanczos_biorthogonal

87.55 test_laplacian

87.56 test_laplacian_non_uniform

87.57 test_laplacian_simple

87.58 test_mesh_2d_uniform

87.59 test_mesh_2d_uniform_2

87.60 test_mesh_circle

87.61 test_mesh_generation

87.62 test_mesh_interpolate

87.63 test_mg

87.64 test_minres_recycle

87.65 test_multigrid

87.66 test_nc

87.67 test_nonuniform_symmetric

87.68 test_pde

87.69 test_permutation

87.70 test_poison_fem

87.71 test_polar

87.72 test_potential

87.73 test_powers

87.74 test_precondition

87.75 test_project_rectangle

87.76 test_qr

87.77 test_quantum_well

87.78 test_radial_adaptive

87.79 test_radial_confinement

87.80 test_radial_fixes

87.81 test_refine_2d

87.82 test_refine_2d_b

87.83 test_refine_3d

87.84 test_refine_structural

87.85 test_regularisation

87.86 test_round_off

87.87 test_schrdinger_potentials

87.88 test_uniform_mesh

87.89 test_vargrid

88 test

88.1 test_gaussfit3

88.2 test_geoserr

88.3 test_lognfit_quantile

88.4 test_max_normal

88.5 test_mtimes3x3

89 mathematics

```
mathematical functions of various kind
```

89.1 vanderd_2d

90 wavelet

90.1 contiuous_wavelet_transform

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

90.2 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
```

90.3 example_wavelets

90.4 phasewrap

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

90.5 test_cwt_man

90.6 test_phasewrap

90.7 test_wavelet

90.8 test_wavelet2

90.9 test_wavelet_analysis

90.10 test_wavelet_reconstruct

90.11 test_wtc

90.12 wavelet

```
wavelet windows
```

90.13 wavelet_reconstruct

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

90.14 wavelet_transform

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

91 mathematics

```
mathematical functions of various kind
```

91.1 wrapphase