Manual for Package: mathematics

Karl Kastner

July 5, 2019

Contents

1 complex-analysis

operations on complex numbers

1.1 complex_exp_product_im_im

```
the product has two frequency components

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

1.2 complex_exp_product_im_re

```
 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
```

1.3 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
```

1.4 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
```

1.5 croots

```
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
```

1.6 root_complex

1.7 test_imroots

2 derivation

derivation of several functions by means of symbolic computation

2.1 derive_acfar1

2.2 derive_ar2param

2.3 derive_arc_length

2.4 derive_fourier_power

2.5 derive_fourier_power_exp

2.6 derive_laplacian_curvilinear

2.7 derive_laplacian_fourier_piecewise_linear

2.8 derive_logtripdf

2.9 derive_smooth1d_parametric

2.10 simplify_atan

3 fourier/@STFT

3.1 STFT

```
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
```

3.2 itransform

3.3 stft_

3.4 stftmat

3.5 transform

4 fourier

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

4.1 amplitude_from_peak

```
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 (!)
```

4.2 dftmtx_man

```
columns of higher frequencies are omitted

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

output :
F : fourier matrix
```

4.3 example_fourier_window

4.4 fft_derivative

```
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
```

4.5 fft_man

```
input:
F : data in real space

output :

F : fourier transformation of F
```

4.6 fftsmooth

```
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
```

4.7 fix_fourier

```
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
```

4.8 fourier_axis

```
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
```

4.9 fourier_coefficient_piecewise_linear

```
(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
```

4.10 fourier_coefficient_piecewise_linear_1

```
(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
```

4.11 fourier_coefficient_ramp3

4.12 fourier_coefficient_ramp_pulse

4.13 fourier_coefficient_ramp_step

4.14 fourier_coefficient_square_pulse

4.15 fourier_derivative

```
not of discrete fourier transform (fft)
```

4.16 fourier_expand

4.17 fourier_fit

```
equal intervals
```

4.18 fourier_interpolate

4.19 fourier_matrix

```
(not for the discrete dft/fft)
```

4.20 fourier_matrix2

```
(not for the discrete dft/fft)
```

4.21 fourier_matrix3

```
this is a matrix with (2*n+1) real columns
```

4.22 fourier_power

```
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
```

4.23 fourier_power_exp

```
      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
```

4.24 fourier_predict

4.25 fourier_range

```
range(y) = max(y) - min(y)
```

4.26 fourier_regress

```
at equal intervals
```

4.27 fourier_resampled_fit

```
but stores them as resampled values
```

4.28 fourier_resampled_predict

```
at their support points
```

4.29 fourier_signed_square

```
in general
          cos a is midrange
          cos t is tidal variation
c.f Dronkers
```

4.30 fourier_transform

```
(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
```

4.31 hyperbolic_fourier_box

4.32 idftmtx_man

```
with a limited number of columns, thus ignoring higher frequencies
keep 2nc+1 columns (mean and conj-complex pairs of nc frequencies)
```

4.33 laplace_2d_pwlinear

```
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
```

4.34 nanfft

4.35 peaks

```
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)
```

4.36 roots_fourier

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

4.37 spectral_density

4.38 test_complex_exp_product

4.39 test_idftmtx

5 geometry/@Geometry

5.1 Geometry

5.2 arclength

```
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
```

5.3 arclength_old

5.4 arclength_old2

5.5 base_point

```
to another point
```

5.6 base_point_limited

5.7 centroid

5.8 cross2

5.9 curvature

5.10 ddot

5.11 distance

5.12 distance2

```
this function requires a and be of equal dimensions, or the least
    the first pair or second pair to be a scalar
```

5.13 dot

5.14 edge_length

5.15 enclosed_angle

5.16 enclosing_triangle

5.17 hexagon

5.18 inPolygon

```
much faster than matlab internal function
```

5.19 inTetra

5.20 inTetra2

5.21 inTriangle

5.22 intersect

5.23 lineintersect

5.24 lineintersect1

5.25 minimum_distance_lines

5.26 mittenpunkt

5.27 nagelpoint

5.28 onLine

5.29 orthocentre

5.30 plumb_line

5.31 poly_area

5.32 poly_edges

5.33 poly_set

```
and assign the value of the polygon to it
```

5.34 poly_width

```
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
```

5.35 polyxpoly

5.36 project_to_curve

5.37 random_disk

5.38 random_simplex

5.39 sphere_volume

5.40 tetra_volume

5.41 tobarycentric

5.42 tobarycentric1

5.43 tobarycentric2

5.44 tobarycentric3

5.45 tri_angle

5.46 tri_area

5.47 tri_centroid

5.48 tri_distance_opposit_midpoint

5.49 tri_edge_length

5.50 tri_edge_midpoint

5.51 tri_excircle

5.52 tri_height

5.53 tri_incircle

5.54 tri_isacute

5.55 tri_isobtuse

5.56 tri_semiperimeter

5.57 tri_side_length

6 geometry

6.1 Polygon

```
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
```

6.2 bounding_box

6.3 curvature_1d

6.4 cvt

6.5 deg_to_frac

6.6 ellipse

6.7 ellipseX

6.8 ellipseY

6.9 first_intersect

6.10 golden_ratio

6.11 hypot3

6.12 meanangle

6.13 meanangle2

6.14 meanangle3

6.15 meanangle4

6.16 medianangle

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

6.17 medianangle2

```
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
```

6.18 pilim

6.19 streamline_radius_of_curvature

```
simplifies when rotatate to streamwise coordinates to R = 1/dv/ds *
     u
```

7 linear-algebra

7.1 averaging_matrix_2

7.2 colnorm

7.3 condest_

8 linear-algebra/coordinate-transformation

8.1 barycentric2cartesian

8.2 barycentric2cartesian3

8.3 cartesian2barycentric

8.4 cartesian_to_unit_triangle_basis

8.5 latlon2utm

8.6 lowrance_mercator_to_wgs84

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

8.7 nmea2utm

8.8 sn2xy

8.9 unit_triangle_to_cartesian

8.10 utm2latlon

8.11 xy2nt

```
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
```

8.12 xy2sn

8.13 xy2sn_java

8.14 xy2sn_old

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

9 linear-algebra

9.1 det2x2

9.2 det3x3

9.3 det4x4

9.4 diag2x2

9.5 eig2x2

9.6 first

9.7 gershgorin_circle

9.8 haussdorff

```
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
```

9.9 ieig2x2

9.10 inv2x2

9.11 inv3x3

9.12 inv4x4

9.13 lpmean

9.14 lpnorm

9.15 matvec3

9.16 max2d

9.17 mpoweri

9.18 mtimes2x2

9.19 mtimes3x3

9.20 nannorm

9.21 nanshift

9.22 nl

```
analogue to unix nl command
```

9.23 normalise

```
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)
```

9.24 normalize1

9.25 normrows

9.26 orth2

9.27 orth_man

9.28 orthogonalise

9.29 paddext

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

9.30 paddval1

9.31 paddval2

10 linear-algebra/polynomial

10.1 chebychev

10.2 piecewise_polynomial

10.3 roots1

10.4 roots2

```
c1 x^2 + c2 x + c3 = 0
```

10.5 vanderi_1d

10.6 vandermonde

11 linear-algebra

11.1 randrot

11.2 right

```
extrapolate rightmost column
```

11.3 rot2

11.4 rot2dir

11.5 rot3

11.6 rownorm

11.7 simmilarity_matrix

11.8 spnorm

11.9 spzeros

11.10 transpose3

11.11 transposeall

12 logic

bitwise operations on integers

12.1 bitor_man

```
input:
      A (positive integer)
```

13 number-theory

13.1 ceiln

13.2 digitsb

13.3 floorn

13.4 iseven

13.5 multichoosek

```
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
```

13.6 nchoosek_man

```
b = N!/K!(N-K)!
```

13.7 pythagorean_triple

13.8 roundn

14 numerical-methods/differentiation

14.1 derivative1

```
second order accurate
```

14.2 derivative2

15 numerical-methods/finite-difference

15.1 cdiff

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

15.2 cdiffb

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

15.3 cmean

15.4 derivative_matrix_1_1d

15.5 derivative_matrix_2_1d

15.6 derivative_matrix_2d

15.7 derivative_matrix_curvilinear

15.8 derivative_matrix_curvilinear_2

```
the grid has not necessarily to be orthogonal
```

15.9 difference_kernel

```
c.f. Computing the Spectrum of the Confined Hydrogen Atom, Kastner,
     2012
```

15.10 distmat

15.11 gradpde2d

```
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
```

15.12 laplacian

15.13 laplacian_fdm

```
BC
```

15.14 left

15.15 lrmean

15.16 mid

15.17 pwmid

15.18 ratio

15.19 steplength

15.20 swapoddeven

15.21 test_derivative_matrix_2d

15.22 test_derivative_matrix_curvilinear

15.23 test_difference_kernel

16 numerical-methods/finite-volume/@Advection

16.1 Advection

16.2 dot_advection

17 numerical-methods/finite-volume/@Burgers

17.1 burgers_split

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

17.2 dot_burgers_fdm

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

17.3 dot_burgers_fft

```
u_t + (0.5*u^2)_x = c*u_xx
```

18 numerical-methods/finite-volume/@Finite_Volume

18.1 Finite_Volume

```
(time and space)
```

18.2 apply_bc

```
apply boundary conditions
```

18.3 solve

```
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);
```

18.4 step_split_strang

```
this scheme is not suitable for stationary solutions, for example
steady shallow water flow
```

18.5 step_unsplit

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

19 numerical-methods/finite-volume/@Flux_Limiter

19.1 Flux_Limiter

19.2 beam_warming

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

19.3 fromm

```
low res
```

19.4 lax_wendroff

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

19.5 minmod

19.6 monotized_central

19.7 muscl

19.8 superbee

19.9 upwind

```
godunov, first order accurate
```

19.10 vanLeer

20 numerical-methods/finite-volume/@KDV

20.1 dot_kdv_fdm

```
u_t + (0.5*u^2)_x = c*u_xxx
```

20.2 dot_kdv_fft

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

20.3 kdv_split

```
derivative treated by splitting scheme
```

21 numerical-methods/finite-volume/@Reconstruct_Average_E

21.1 Reconstruct_Average_Evolve

```
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;
```

21.2 advect_highres

21.3 advect_lowress

```
low resolution
```

22 numerical-methods/finite-volume

22.1 Godunov

22.2 Lax_Friedrich

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

22.3 Measure

22.4 Roe

```
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
```

22.5 fv_swe

22.6 staggered_euler

22.7 staggered_grid

23 numerical-methods

23.1 grid2quad

```
in form of an unstructured quad-mesh format
```

24 numerical-methods/integration

24.1 cumintL

24.2 cumintR

24.3 int_trapezoidal

25 numerical-methods/interpolation/@Kriging

25.1 Kriging

25.2 estimate_semivariance

```
% set up the regression matrix and solve for
    parameters
```

25.3 interpolate_

```
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
```

26 numerical-methods/interpolation/@RegularizedInterpolator

26.1 RegularizedInterpolator1

26.2 init

27 numerical-methods/interpolation/@RegularizedInterpolator

27.1 RegularizedInterpolator2

27.2 init

28 numerical-methods/interpolation/@RegularizedInterpolator3

28.1 RegularizedInterpolator3

```
(unstructured mesh)
```

28.2 init

29 numerical-methods/interpolation

29.1 IDW

29.2 IPoly

29.3 IRBM

```
fprintf(1,'Progress IRBM: %d%%\n',round(100*
    idx/size(Xi,1)));
```

29.4 ISparse

29.5 Inn

29.6 Interpolator

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

29.7 fixnan

29.8 idw1

29.9 idw2

29.10 inner2outer

```
assumes equal grid spacing
```

29.11 inner2outer2

29.12 interp1_limited

```
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
```

29.13 interp1_man

29.14 interp1_save

```
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
```

29.15 interp1_smooth

29.16 interp1_unique

```
this function makes the values unique before use
```

29.17 interp2_man

29.18 interp_angle

29.19 interp_fourier

29.20 interp_fourier_batch

29.21 interp_sn

```
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_);
```

29.22 interp_sn2

29.23 interp_sn3

29.24 interp_sn_

29.25 limit_by_distance_1d

```
      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)
```

29.26 resample1

29.27 resample_d_min

29.28 resample_vector

29.29 test_interp1_limited

30 numerical-methods

30.1 inverse_complex

31 numerical-methods/ode

31.1 bvp2c

```
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])
```

31.2 bvp2c2

```
polynomial

input:

x   : [nx1] discretized domain
      n : number of vertices
     ns = 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*ns x 2*ns] disrcretisation matrix
rhs : [2*ns x 1] right hand size

y = A^-1 rhs
```

31.3 bvp2fdm

```
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])
```

31.4 bvp2wavetrain

31.5 bvp2wavetwopass

```
solve first for the wave number k, and then for y
```

31.6 ivp_euler_forward

31.7 ivprk2

31.8 ode2_matrix

```
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]
```

31.9 ode2characteristic

```
transmittded and reflected wave
```

31.10 step_trapezoidal

32 numerical-methods/optimisation

32.1 armijo_stopping_criterion

32.2 astar

32.3 binsearch

32.4 bisection

32.5 box1

32.6 box2

32.7 cauchy

32.8 cauchy2

```
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
```

32.9 directional_derivative

```
d : derivative, highest first
p : series expansion around x0
```

32.10 dud

32.11 extreme3

```
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
```

32.12 extreme_quadratic

32.13 ftest

32.14 grad

32.15 hessian

32.16 hessian_from_gradient

32.17 hessian_projected

32.18 line_search

32.19 line_search2

```
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
```

32.20 line_search_polynomial

```
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
```

32.21 line_search_polynomial2

```
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
```

32.22 line_search_quadratic

```
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
```

32.23 line_search_quadratic2

32.24 line_search_wolfe

```
c.f.: OPTIMIZATION THEORY AND METHODS - Nonlinear Programming, Sun,
     Yuan
```

32.25 ls_bgfs

32.26 ls_broyden

```
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
```

32.27 ls_generalized_secant

```
Barnes, 1965
Wolfe, 1959
Fletcher 1980, 6.3
seber 2003
gerber
```

32.28 nlcg

```
input:
x   : nx1 start vectort
opt : struct options
fdx : gradient constraint
```

32.29 nlls

32.30 picard

32.31 poly_extrema

32.32 quadratic_function

32.33 quadratic_programming

32.34 quadratic_step

32.35 rosenbrock

32.36 sqrt_heron

32.37 test_directional_derivative

32.38 test_dud

32.39 test_line_search_quadratic2

32.40 test_ls_generalized_secant

32.41 test_nlcg_6_order

32.42 test_nlls

33 numerical-methods/piecewise-polynomials

33.1 Hermite1

33.2 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)
```

33.3 hp2_predict

```
c are values at support points
```

33.4 hp_predict

33.5 hp_regress

```
coefficients are values and derivatives
```

33.6 lp_count

```
count number of valid samples
```

33.7 lp_predict

33.8 lp_regress

33.9 lp_regress_

34 regression/@PolyOLS

34.1 PolyOLS

34.2 coefftest

34.3 detrend

34.4 fit

```
like polyfit, but returns parameter error estimates
```

34.5 fit_

34.6 predict

34.7 predict_

34.8 slope

35 regression/@PowerLS

35.1 PowerLS

35.2 fit

```
like polyfit, but returns parameter error estimates
```

35.3 predict

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

35.4 predict_

36 regression/@Theil

36.1 Theil

36.2 detrend

36.3 fit

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

36.4 predict

36.5 slope

37 regression

linear and non-linear regression

37.1 Theil_Multivariate

```
means of the Gauss-Seidel iteration
```

37.2 areg

37.3 ginireg

37.4 hesssimplereg

```
rhs = p(1) + p(2) x + eps
```

37.5 l1lin

37.6 lsq_sparam

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

37.7 polyfitd

```
of the derivative

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

37.8 regression_method_of_moments

```
y+eps = alpha + beta*x
```

37.9 robustlinreg

```
      (med(y_left) - med(y_right))/(med(x_left)-med(x_right)
this approach performs poorly compared to the theil-senn operator
```

37.10 theil2

37.11 theil_generalised

```
for arbitrary functions such as polynomials and multivariate
    regression
either higher order polynomials or glm
c.f. "On theil's fitting method", Pegoraro, 1991
```

37.12 total_least_squares

37.13 weighted_median_regression

```
c.f. Scholz, 1978
```

38 set-theory

38.1 issubset

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

39 signal-processing

39.1 acf_effective_sample_size

39.2 acf_genton

39.3 acfar1

```
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
```

39.4 acfar1_2

39.5 acfar2

39.6 acfar2_2

```
X_i + a1 X_i-1 + a2 X_i-2 = 0
```

39.7 ar1_cutoff_frequency

39.8 ar1_effective_sample_size

39.9 ar1_mse_mu_single_sample

39.10 ar1_mse_pop

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

39.11 ar1_mse_range

39.12 ar1_spectrum

39.13 ar1_to_tikhonov

39.14 ar1_var_factor

```
n   : [1 .. inf] population size
m   : [1 .. n]  samples size
rho : [ -1 < rho < 1 (for convergence) ] correlation of samples
```

39.15 ar1_var_factor_

39.16 ar1_var_range2

```
from the finite length first order autocorrelated process

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

39.17 ar1delay

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

39.18 ar1delay_old

39.19 ar2conv

```
of the acf [1,r1,r2,...]
```

39.20 ar2dof

39.21 ar2param

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

39.22 asymwin

```
filter will always have negative weights
```

39.23 autocorr_fft

39.24 bandpass

39.25 bandpass2

39.26 bartlett

```
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
```

39.27 bartlett_spectrogram

```
TODO sliding window
```

39.28 bin1d

```
apply function v to it
```

39.29 bin2d

```
apply function func to all walues in the bin
func = mean : default
func = sum : non-normalized frequency histogram in 2D
```

39.30 binormrnd

39.31 conv1_man

39.32 conv2_man

39.33 conv2z

39.34 conv30

```
circular boundaries
```

39.35 conv_

39.36 conv_centered

```
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
```

39.37 convz

39.38 cosexpdelay

39.39 csmooth

39.40 daniell_window

```
c.f. Daniell 1946
Bloomfield 2000
meko 2015
```

39.41 danielle_window

39.42 db2neper

39.43 db2power

39.44 derive_danielle_weight

39.45 derive_limit_0_acfar

39.46 detect_peak

```
requires function value to fall to p*max before new value is
    allowed
```

39.47 digital_low_pass_filter

```
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)
```

39.48 doublesum_ij

39.49 effective_sample_size_to_ar1

39.50 filt_hodges_lehman

39.51 filter1

39.52 filter2

39.53 filter_

39.54 filteriir

```
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
```

39.55 filterp

39.56 filterp1

39.57 filterstd

39.58 firls_man

39.59 flattopwin

39.60 frequency_response_boxcar

39.61 freqz_boxcar

39.62 gaussfilt1

39.63 hanchangewin

39.64 hanchangewin2

39.65 hanwin

39.66 hanwin_

39.67 highpass

39.68 kaiserwin

39.69 kalman

39.70 lanczoswin

39.71 last

39.72 lowpass

39.73 lowpass2

39.74 lowpass_iir

39.75 lowpass_iir_symmetric

39.76 lowpassfilter2

39.77 maxfilt1

39.78 meanfilt1

39.79 medfilt1_man

39.80 medfilt1_man2

39.81 medfilt1_padded

39.82 medfilt1_reduced

39.83 mid_term_single_sample

39.84 minfilt1

39.85 mu2ar1

```
(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
```

39.86 nanautocorr

39.87 nanmedfilt1

39.88 neper2db

39.89 peaks_man

39.90 polyfilt1

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

39.91 qmedfilt1

39.92 randar1

```
e1 = randar1(sigma,p,n,m)
```

39.93 randar1_dual

39.94 randar2

39.95 randarp

39.96 range_window

39.97 rectwin

39.98 recursive_sum

39.99 select_range

39.100 smooth1d_parametric

```
so that distance to p1 and p2 becomes equal
and the chord length remains the same
```

39.101 smooth2

39.102 smooth_man

39.103 smooth_parametric

```
matvec2x2(R,[dxc;dyc])
```

39.104 smooth_parametric2

39.105 smoothfft

39.106 spectrogram

39.107 std_window

39.108 sum_i_lag

```
sum_i=1^n rho^|i-k|
```

39.109 sum_ii

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

39.110 sum_ii_

39.111 sum_ij

```
sum_{i=1}^n sum_{j=1}^m r^|i-j|
```

39.112 sum_ij_

39.113 sum_ij_partial_

39.114 sum_multivar

39.115 test_acfar1

39.116 test_acfar1_2

39.117 test_acfar1_3

39.118 test_acfar1_4

39.119 test_acfar2

39.120 test_ar1_var_factor

39.121 test_ar1_var_factor_2

39.122 test_ar1_var_mu_single_sample

39.123 test_ar1_var_pop

39.124 test_ar1_var_pop_1

39.125 test_ar1delay

39.126 test_bivariate_covariance_term

39.127 test_convexity

39.128 test_lanczoswin

39.129 test_madcorr

39.130 test_randar1

39.131 test_randar1_multivariate

39.132 test_randar2

39.133 test_sum_ij

39.134 test_sum_multivar

39.135 test_trifilt1

39.136 test_wautocorr

39.137 test_wavelet_transform

39.138 test_wordfilt

39.139 test_xar1_mid_term

39.140 tikhonov_to_ar1

39.141 trapwin

39.142 trifilt1

39.143 triwin

39.144 triwin2

39.145 varar1

```
with respect to the mean, averaged over the population
```

39.146 welch_spectrogram

39.147 wfilt

39.148 winbandpass

39.149 window_make_odd

39.150 winfilt0

39.151 winlength

```
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
```

39.152 wmeanfilt

39.153 wmedfilt

39.154 wordfilt

39.155 wordfilt_edgeworth

39.156 xcorr_man

40 sorting

40.1 sort2

40.2 sort2d

```
returns row and column index of sorted values
```

41 special-functions

41.1 bessel_sphere

41.2 hankel_sphere

```
first kind
```

41.3 hermite

```
input :
n : order
x : value

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

41.4 legendre_man

41.5 neumann_sphere

```
Bessel function of the second kind
```

42 statistics

42.1 atan_s2

42.2 beta_mode_to_parameter

42.3 correlation_confidence_pearson

```
c.f. Fischer 1921
```

43 statistics/distributions

43.1 PDF

43.2 binorm_separation_coefficient

43.3 binormcdf

43.4 binormfit

43.5 binormpdf

43.6 edgeworth_cdf

```
with mean mu, standard deviation sigma, and third and fourth
    cumulants
c.f. Rao 2010
```

43.7 edgeworth_pdf

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

43.8 logn_mode2param

43.9 logn_param2mode

43.10 lognpdf_

43.11 pdfsample

```
Note: better use kernal density estimates
```

43.12 t2cdf

43.13 t2inv

44 statistics

44.1 example_standard_error_of_sample_quantiles

44.2 f_var_finite

```
without replacement
```

44.3 gamma_mode_to_parameter

44.4 hodges_lehmann_correlation

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

44.5 hodges_lehmann_dispersion

```
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
```

45 statistics/information-theory

45.1 akaike_information_criterion

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

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

45.2 bayesian_information_criterion

46 statistics

46.1 kurtosis_bias_corrected

46.2 limit

46.3 logfactorial

46.4 loglogpdf

46.5 logskewcdf

46.6 logskewpdf

47 statistics/logu

47.1 lambertw_numeric

47.2 logtrialtcdf

47.3 logtrialtinv

```
 = (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))))
```

47.4 logtrialtmean

47.5 logtrialtpdf

47.6 logtrialtrnd

47.7 logtricdf

47.8 logtriinv

47.9 logtrimean

47.10 logtripdf

47.11 logtrirnd

47.12 logucdf

47.13 logucm

47.14 loguinv

47.15 logumean

47.16 logupdf

47.17 logurnd

47.18 loguvar

47.19 medlogu

47.20 test_logurnd

47.21 tricdf

47.22 triinv

47.23 trimedian

47.24 tripdf

47.25 trirnd

48 statistics

48.1 maxnnormals

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

48.2 midrange

48.3 minavg

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

48.4 mode_man

49 statistics/moment-statistics

49.1 autocorr_man3

49.2 autocorr_man4

```
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
```

49.3 autocorr_man5

49.4 blockserr

```
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
```

49.5 comoment

```
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
```

49.6 corr_man

49.7 cov_man

49.8 dof

```
for a polynomial of degree order in dim dimensions
```

49.9 edgeworth_quantile

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

49.10 effective_sample_size

```
c.f. Kish
```

49.11 f_correlation

49.12 f_finite

```
without replacement
```

49.13 lmean

49.14 lmoment

49.15 maskmean

49.16 masknanmean

49.17 mean1

49.18 mean_man

49.19 mse

```
this is de-facto the std for an unbiased residual
```

49.20 nanautocorr_man1

49.21 nanautocorr_man2

49.22 nanautocorr_man4

```
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
```

49.23 nancorr

49.24 nancumsum

49.25 nanlmean

49.26 nanr2

49.27 nanrms

49.28 nanrmse

```
this is de-facto the std for an unbiased residual
```

49.29 nanserr

49.30 nanwmean

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

49.31 nanwstd

49.32 nanwvar

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

49.33 nanxcorr

49.34 pearson

49.35 pearson_to_kendall

```
c.f. Kruskal 1958
```

49.36 pool_samples

49.37 qmean

49.38 range_mean

49.39 rmse

```
this is de-facto the std for an unbiased residual
```

49.40 serr

49.41 serr1

49.42 test_qskew

49.43 test_qstd_qskew_optimal_p

49.44 wautocorr

```
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
```

49.45 wcorr

49.46 wcov

49.47 wdof

49.48 wkurt

49.49 wmean

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

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

49.50 wrms

49.51 wserr

49.52 wskew

49.53 wstd

49.54 wvar

```
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
```

50 statistics

50.1 nangeomean

50.2 nangeostd

51 statistics/nonparametric-statistics

51.1 kernel1d

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

51.2 kernel2d

52 statistics

52.1 normmoment

52.2 normpdf2

53 statistics/order-statistics

53.1 hodges_lehmann_location

```
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)
```

53.2 kendall

53.3 kendall_to_pearson

```
correlation coefficient

c.f. Kruska, 1985
```

53.4 mad2sd

```
for normal distributed values
```

53.5 madcorr

53.6 median2_holder

53.7 median_ci

```
se_me = sqrt(1/2 pi) 1.25331 * sd/sqrt(n)
```

53.8 median_man

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

53.9 mediani

53.10 nanmadcorr

53.11 nanwmedian

53.12 nanwquantile

53.13 oja_median

```
note: the multivariate median is not unique

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

53.14 qkurtosis

```
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
```

53.15 qmoments

53.16 qskew

```
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
```

53.17 qskewq

53.18 qstdq

53.19 quantile1_optimisation

53.20 quantile2_breckling

53.21 quantile2_chaudhuri

53.22 quantile2_projected

53.23 quantile2_projected2

53.24 quantile_envelope

53.25 quantile_regression_simple

53.26 ranking

53.27 spatial_median

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

53.28 spatial_quantile

53.29 spatial_quantile2

53.30 spatial_quantile3

53.31 spatial_rank

53.32 spatial_sign

53.33 spatial_signed_rank

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

53.34 spearman

53.35 spearman_rank

53.36 spearman_to_pearson

53.37 wmedian

53.38 wquantile

54 statistics/random-number-generation

54.1 laplacernd

54.2 randc

54.3 skewrnd

54.4 skewrnd2

55 statistics

55.1 range

56 statistics/resampling-statistics/@Jackknife

56.1 Jackknife

```
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
```

56.2 estimated_STATIC

```
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
```

56.3 matrix1_STATIC

56.4 matrix2

57 statistics/resampling-statistics

57.1 block_jacknife

57.2 jackknife_moments

```
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
```

57.3 moving_block_jacknife

```
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
```

57.4 randblockserr

```
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
```

57.5 resample

```
TODO, should be with replacement

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

58 statistics

58.1 scale_quantile_sd

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

58.2 skewpdf

```
c.f. Azzalini 1985
```

58.3 trimmed_mean

58.4 ttest2_man

```
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
```

58.5 ttest_man

```
unequal sample size
equal variance
```

58.6 ttest_paired

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

58.7 wharmean

59 wavelet

59.1 contiuous_wavelet_transform

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

59.2 cwt_man

```
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
```

59.3 example_wavelets

59.4 phasewrap

59.5 test_cwt_man

59.6 test_phasewrap

59.7 test_wavelet

59.8 test_wavelet2

59.9 test_wavelet_analysis

59.10 test_wavelet_reconstruct

59.11 test_wtc

59.12 wavelet

59.13 wavelet_reconstruct

```
(reconstruction of time series)
n : window lengths in multiples of filter period 1/f0
```

59.14 wavelet_transform

```
n : window lengths in multiples of filter period 1/f0
```