Manual for Package: tide
Revision 24M

Karl Kästner

May 2, 2021

Contents

1 @T_Tide

1.1 T_Tide

```
wrapper for TPXO generated tidal time series
```

1.2 build_index

```
build a structure whose field names contain the index
```

1.3 from_tpxo

```
read TPXO output into tidetable object
```

1.4 get_constituents

```
extract constituents of tpxo object
```

1.5 reorder

```
order constituents as specified by "name"
```

1.6 select

```
select a subsect of constituents
```

1.7 shift_time_zone

```
shift phase according to time zone
```

2 @Tidal_Envelope

2.1 Tidal_Envelope

```
process tidal data to extrac the tidal envelope
```

2.2 init

```
initialize with data
```

3 @Tide_wft

wavelet analysis of tidal data

3.1 Tide_wft

```
wavelet transform of tidal time series
```

3.2 transform

```
wavelet transform tidal time series
input:
time   : [1xn] abszissa of input vector, for example time, must be
    equally spaced
val    : [1xn] signal, input data series (e.g water level or
    velocity)
F      : [1xm] base frequencies, 1, 1, 2, ... for mean level,
    diurnal, semidirunal ...
              base periods from base frequencies T=1/F
n      : [1xm] wavelet window length in multiple of periods
fc, nc : [scalar] low frequency cutoff and window length in periods
winstr : [char] fourier windows (kaiser (recommended), hanning, box
    , etc)
dt_max : [scalar] maximum time to fill gaps in input data series (
    recommended 3/24 for tide)
output:
tide   : struct with fields
        w_coeff  : [1xn] wavelet coefficients (complex)
        amplitude : amplitude
        phase    : phase
        range    :
        h_tide   :
        h_low    :
        h
```

4 @Tidetable

class for generating tidetable data

4.1 Tidetable

```
Tide table
```

4.2 analyze

```
extract tidal envelope from time series
```

4.3 export_csv

```
export tide table to csv file
```

4.4 generate

```
run TPXO to generate time series
```

4.5 generate_tpxo_input

```
generate tpxo input table
Note: superseeded by perl script
```

4.6 import_tpxo

```
import TPXO data into tidetable object
```

4.7 plot_neap_spring

```
plot average neap and spring tide
```

5 tide

```
analysis prediction of tides in rivers and estuaries by empirical
    and theoretical methods
```

5.1 constituents

5.2 doodson

```
frequency of tidal constituents
method of doodson
source: wikipedia
```

5.3 envelope_amplitude

```
compute envelopes of hw and low water
```

5.4 envelope_slack_water

```
slack water envelope of the tide
```

5.5 interval_extrema

```
times and evelations for high and low water
```

5.6 interval_extrema2

```
mimimum and maximum within intervals of constant length,
intended for periodic functions
```

5.7 interval_zeros

```
times of slack water determined frim velocity u
```

5.8 lunar_phase

```
lunar phase
```

5.9 rayleigh_criterion

```
raleigh criterion for resolving tidal constituents
T > 1/|f1-f2|
```

6 river-tide/@Bifurcation

6.1 Bifurcation

6.2 confluence_rule

6.3 sediment_division

6.4 sediment_division_geometric

7 river-tide/@River_Tide

```
predict tide in a backwater affected river with a sloping/varying
    bed

Assumptions and capabilities:
       - tidal dynamics follow the 1D-Shallow-Water-Equation
         (depth and cross-sectionally averaged Navier-Stokes-
            Equation)
       - rectangular cross section
       - width can vary along the channel
       - friction coefficient (cd) constant along channel and over
           time (Chezy)
       - advective accelleration term is considered, but can be
           deactivated
       - vertical profile of streamwise velocity is constant
         (Boussinesq coefficient is unity (1))
```

```
Limitations / TODO list:
	- single channel dynamics only (no tidal networks)
	- no wind-shear stress (no storm surges)
	- no tidal flats / intertidal areas (width constant in time)
	- no flood-plain during high-river flow
	- no stratification or along-channel salinty gradient
	- negligible head loss in channel bends
	- negligible feed-back of the sediment concentration on the
	   propagation of the tide
	- low Froude Number (no hydraulic jumps due to cataracts or
	   tidal bores)

- At present, only two tidal components are supported
  (either D1 with D2 or D2 with D4, in addition to the mean water
      level z0),
  for mixed diurnal-semidiurnal cases with dominant semidiurnal
      component,
  the class has to be extended to support three components (D1, D2
      and D4)

- At present, the tripel overtide is not computed (D3 for diurnal,
    D6 for semindiurnal tide),
  note that this is the main overtide for the case of low river flow

- At present, the 1/h non-linearity is only included in the
    approximations of
  the backwater curve, but not it's influence on the tidal frequency
       components

Method:

This class calls numerical solvers for second order ordinary
    differential
equation boundary value problems

Tides is represented as exponential series in form of total
    discharge Q = sum Q_i = Q_0 + Q_1 + Q_2,
as discharge is conserved (balanced), the equations are simpler than
     for level z and velocity u,
and the frequency components of z are straight forward determined by
     differentiation of Q

Class and function structure:
	River_Tide :
		computes river tide, provides the ode coefficients to
		    the boundary value solver
	bvp2c, bvp2fdm :
		solve the underlying second order boundary value
		    problem
```

```
        River_Tide_Map :
                provides convenient batch runs and processing of
                    River_Tide instances

Minimum working example, c.f. example_rive_tide.m and
    example_river_tide_map.m

        input:

        Q0    : scalar, river discharge (m^3/s)
        omega : scalar, angular frequency main tidal species in (1/
            seconds)
        x     : 2x1 vector, left and right end of computational
            domain of the river (m)
        w(x)  : function of width along the river (m)
        cd(x) : function of drag coefficient along the river (1)
        zb(x) : function of bed level along the river (m)

        opt   : structure with options
        opt.model_str = 'wave' (other solver are not supported at the
             moment)
        opt.solver = @bvp2c or @bvp2fdm
        opt.nx : number of grid points along channel
        opt.ns : base for logarithmic spacing of grid points, 1 :
            linear spacing

        bc  : structure array of boundary conditions
                r, row 1..2 : left and right end, respectively
                c, column 1 : mean (river) component
                          2..n : condition form column-1 frequency
                              component

                (       q(1)*(p(1) y^-(x0) + p(2) dy^-/dx(x0) ...
                     +  q(2)*(p(1) y^+(x0) + p(2) dy^+/dx(x0) ) = bc(c
                          ,r).val
                           = val(0)

        bc(c,r).var : Quantity, either 'z' or 'Q'
        bc(c,r).val : complex amplitude of chosen variable
                     (c.f. (1 + 0i) [m] for surface elevation
                         amplitude of 1m)
                     (value has to be real for mean component)
                     mean component requires z and Q to be specified
                          at opposit ends
        bc(c,r).p  : factor for Dirichlet p(1) or Neumann p(2)
            condition
                     p = [1,0] : pure Dirichlet
                     p = [0,1] : pure Neumann
                     sum of abs(p) must be nonzero for each end and
```

```
                  each frequency component
       bc(c,r).q  : factor for left and right going wave, only
           available for bvp2c
                  q = [1,1] : total water level / discharge
                  q = [1,0] : only left going wave
                  q = [0,1] : only right going wave
                  q has no meaning for the mean component and is
                      ignored
                  q is only supported by bvp2c,
                  bvpfdm uses default q = [1,1]
                  sum of abs(q) for each frequency component must
                      be zero
```

7.1 River_Tide

```
physical functions for computation of river tides in a single 1D
    channel
combined with BVP-solver in child-classes to determine the
    hydrodynamics
```

7.2 check_continuity

7.3 check_momentum

7.4 coefficient_frequency_components

7.5 d2au1_dx2

```
second derivative of the tidal velocity magnitude

note: this is for finding zeros,
      the true derivative has to be scaled up by z
```

7.6 d2az1_dx2

```
second derivative of the tidal surface elevation

note: this is for finding zeros,
     the true derivative has to be scaled up by z
```

7.7 decompose

```
decompose the tide into a right and left travelling wave,
i.e. into incoming and reflected wave
TODO subtract forcing term
```

7.8 derive_lorentz

7.9 discharge2level

```
determines tidal water surface amplitude (non-zero freqyency
    components of surface elevation)
from tidal discharge (non-zero freqyency components of the
    discharge)

by continuity :

dz/dt + dq/dx = 0
=> i o z = - dq/dx
=>    z = -1/(io) dq/dx
=>    z = 1i/o dq/dx
```

7.10 dkq_dx

```
along-channel derivative of the wave number of the discharge
neglects width variation

TODO, rederive with g as variable
```

7.11 dkz_dx

```
along channel derivative of the wave number of the tidal surface
    elevation
ignores width variation dh/dx and second order depth variation (d^2
    h/dx^2)
TODO rederive with g symbolic
```

7.12 energy

7.13 even_overtide_analytic

7.14 fourier_derivative

7.15 friction_coefficient

```
function cf = friction_coefficient(obj,Qmid,Qhr)
```

7.16 friction_coefficient_dronkers

```
friction coefficient according to Dronkers

the coefficients are semi-autogenerated

c.f. dronkers 1964
c.f. Cai 2016

p = [p0,p1,p2,p3];
alpha = Ur/Ut = river velocity / tidal velocity amplitude = (umax+
    umin)/(umax-umin)

function p = friction_coefficient_dronkers(alpha,order)
```

7.17 friction_coefficient_godin

```
friction coefficient according to Godin
these coefficients are identical to Dronker's for U_R = phi = 0

function G = friction_coefficient_godin(obj,phi)
```

7.18 friction_coefficient_lorentz

```
coefficients of the Fourier expansion of the signed square of the |
    Q|Q
of the friction term

Lorent'z used this first for the case of no river flow

identical to Dronker's coefficient for zero river flow
and a single frequency component
c.f. Cai
c.f. Dronkers (gamma = alpha)

note difference in coefficients due to different definitions:
definition used here:
      Q = Q0 + 1/2*(sum_k Q_k e(k iwt) + conj(Q_k e(k iwt)))
but Dronkers defines
      Q = Q + sum_k Q_k e(k iwt)

function L = friction_coefficient_lorentz(obj,phi)
```

7.19 friction_dronkers

```
friction determined by Dronker's method

input :
       u    : velocity time series
         Umid : arithmetic mean of mininmum and maximum velocity
              (not the mean of the velocity, usually non-zero even
                  without river flow)
       Uhr  : half-range of the velocity, less than the sum of
              the frequency amplitudes, except at perigean spring
                  tides

function [uau_sum uau p] = friction_dronkers(u,Umid,Uhr,order)
```

7.20 friction_exponential_dronkers

```
friction coefficicients for the frequency components computed by
    Dronkers method
c.f. Dronker's 1964 eq 8.2 and 8.4
Note: Cai dennominates alpha as phi

function [c uau uau_ p] = friction_trigonometric_dronkers(u,dp,Umid
    ,Uhr,order,psym)
```

7.21 friction_godin

```
compute friction with the method of Godin
```

7.22 friction_lorentz

7.23 friction_quadratic

```
friction determined by Dronker's method
```

7.24 friction_trigonometric_dronkers

```
friction computed by the method of Dronkers
expressed as coefficients for the frequency components
c.f. dronkers 1964 eq 8.2 and 8.4
Note: Cai dennominates alpha as phi
```

7.25 friction_trigonometric_godin

```
friction computed by the method of Godin
expressed as coefficients of the frequency components (
    trigonometric form)

Chebycheff coeffcients for zero river flow
(albeit applied by Godin to cases with river flow)
c.f. godin 1990, table 1, column Ch
Note: the coefficients do indeed not (exactly) sum up to 1
```

```
Note: Godin tries several slightly different sets of coefficients,
      of which the Chebysheff set is best
```

7.26 friction_trigonometric_lorentz

```
friction computed by the method of Lorent'z
expressed as coefficients of the frequency components (
    trigonometric form)
```

7.27 mwl_offset

```
offset of the tidally averaged surface elevation caused by tidal
    friction
Linear estimate of the mean water level offset (ignoring feed-back
    of tide)
```

7.28 mwl_offset_2

7.29 mwl_offset_analytic

7.30 odefun

```
coefficients of the wave equation for river-tides decomposed in
    frequency components
zero frequency component corresponds to backwater equation with
    tidal influence
```

7.31 odefunQ0

7.32 odefun_advective_acceleration

7.33 odefun_friction

7.34 odefun_ghof

7.35 odefun_swe_jacobian

```
Jacobian matrix of the Shallow-Water Equation
d(A,Q)/dt + J(A,Q) d(A,Q)/dx = forcing terms

dA/dt + [      0,      1][dA/dx] = [f_c]   (c)
dQ/dt   [-Q^2/A^2, 2 Q/A ][dQ/dx] [f_m]   (m)

dm/dt
d^2Q/dt^2 - d/dt Q^2/A^2 - 2 d/dt (Q/A) dQ/dx = d/dt f_m
d^2Q/dt^2 - d/dt Q^2/A^2 - 2 (1/A dQ/dt - Q/A^2 dA/dt) dQ/dx = d/dt
     f_m
d^2Q/dt^2 - d/dt Q^2/A^2 - 2 (1/A dQ/dt + Q/A^2 dQ/dx) dQ/dx = d/dt
     f_m

ode = - 1/(g h w) d^2/dt^2 Q + 1/w d^2/dx^2 Q = 0
    = -1 (iko)^2/(g h w) Q + 1/w Q''
    =    (k^2 o^2)/(g h w) Q + 1/w Q''
```

7.36 odefun_width

```
forcing by along-channel width-variation
```

7.37 odefunk_1

7.38 odefunk_1_

```
coefficients of the ordinary differential equation of the k-th
    freqeuncy
component of the tide
```

```
f1 Q'' + f2 Q' + f3 Q + f4 = 0

function [f, F3] = odefunk(obj, k, Q, QQ, Qhr, h0, dh0_dx, dz0_dx,
    w0, dw0_dx, Cd, c, D1_dx)
```

7.39 odefunk_2

7.40 odefunk_3

7.41 odefunz0

```
coefficients of the backwater equation for the river tide
TODO merge with backwater class
```

7.42 wave_number_analytic

```
analytic expression of the wave number of river tides

valid for both tidally, river dominated and low friction conditions
and converging channels

k10   : complex wave number for k and z in a reach with constant
    width and bed slope
im(k) : damping modulus (rate of amplitude change)
re(k) : actual wave number (rate of phase change)

kq    : wave number for Q for a reach with changing width and depth
kz    : wave number for z for a reach with changing width and depth

c.f. derive_wave_number
```

7.43 wave_number_approximation

```
approximate wave number of the left and right traveling wave for
    variable coefficients

TODO merge with wave_number_analytic

function [k, k0, dk0_dx_rel, obj] = wave_numer_aproximation(obj)
```

8 river-tide/@River_Tide_Cai

```
Prediction of river tide by the method of Cai
c.f. Cai 2013, Cai 2015
```

8.1 Gamma

```
Gamma parameter for tidal propagation
c.f. Cai 2014
```

8.2 River_Tide_Cai

```
prediction of river tide by the method of Cai (2014)
```

8.3 river_tide_cai_

```
determine the surface amplitude of the river-tide
c.f. Cai
```

8.4 rt_quantities

```
determine the quantities that determine the tidal propagation
c.f. Cai

Note: this computes 4 unknowns following Cai, however,
lambda, mu and epsilon can be substituted
making it an equation in one unknown (delta) only
```

9 river-tide/@River_Tide_Channel

9.1 River_Tide_Channel

9.2 bcfun

```
Robin (mixed) boundary conditions for the river tide,
supplied for each frequency component,
wrapper that copies values are from the member struct "bc"

     q*(p*Q_1^- + (1-p)*dQ_1^-/dx
input :
      cid : channel index
      bif : 1,2 : index for letft/right end of channel
      fid : frequency component index
             (1 = 0 omega (mean), 2 : 1 omega, 3 : 2 omega, ... )
columns of bc : frequency
rows of bc, left, right boundary
output :
      p : [2x1] linear combination of Dirichlet and Neumann
          boundary condition
        p(1) -> weight Dirichlet boundary condition
          p(2) -> weight Neumann boundary condition
     q linear combination of left and right travelling (incoming and
         outgoing) wave
          q(1) weight left going wave
        q(2) weight right going wave
      rhs = 0 -> homogeneous boundary condition


function [rhs, p, q, obj] = bcfun(obj,cid,bid,fid)
```

9.3 check_continuity

```
compute residual for the continuity equation
dA/dt + dQ/dx = Q_in
```

9.4 decompose

```
decompose the tide into a right and left travelling wave,
i.e. into incoming and reflected wave
TODO subtract forcing term
```

9.5 extract

```
extract values of individual variables from BVP-solver result
    vector
```

9.6 initial_value

9.7 odefun

```
coefficients of the backwater and wave equation for river-tides
```

9.8 postprocess

```
postprocess hydrodynamic solver output
```

9.9 sediment_transport

```
compute sediment transport for a single channel
```

9.10 transform_bc

```
transform arbitrary to cs-integrated discharge boundary condition
```

10 river-tide/@River_Tide_Empirical

```
Empirical fit to measurement and prediction (from tide at sea and
    river discharge)
of the river tide
```

10.1 River_Tide_Empirical

```
class for fitting models to at-a-station time series of tidal
    elevation
```

10.2 fit_amplitude

```
fit the oscillatory components
```

10.3 fit_mwl

```
fit the tidally averaged water level
```

10.4 fit_phase

```
fit the phase of the oscillatory components
```

10.5 fit_range

```
fit the tidal range
```

10.6 predict_amplitude

```
predict the oscillatory components
```

10.7 predict_mwl

```
predict the mean water level
```

10.8 predict_phase

```
predict tidal phase
```

10.9 predict_range

```
predict the tidal range
```

10.10 rt_model

```
select the model for fitting
```

11 river-tide/@River_Tide_Hydrodynamics_Map

```
hash container for a set of River_Tide predictions for different
    boundary conditions
```

11.1 River_Tide_Hydrodynamics_Map

```
container class to store multiple river-tide scenarios
```

11.2 fun

```
compute river tide for a scenario with specific boundary conditions
     and store it in the hash,
or retrive the scenario, if it was already computed
```

11.3 plot

```
quick plot of scenario result

function obj = plot(obj,Xi,Q0,W0,S0,z1_downstream,cd,zb_downstream,
    omega,q,opt)
```

12 river-tide/@River_Tide_IVP

12.1 solve

13 river-tide/@River_Tide_JK

```
empirical analysis and prediction of river tides by the method of
    Jay and Kukulka
```

13.1 River_Tide_JK

13.2 damping_modulus

```
damping modulus of the river tide
c.f. Jay and Kukula
function r = damping_modulus(obj,h0,b,Qr)
```

13.3 mean_level

```
tidally averaged surface elevation
c.f. Jay and Kukulka
```

13.4 rivertide_predict

```
predict river tide by the method of jay and kukulka
TODO rename
```

13.5 rivertide_regress

```
Regression of tidal coefficients according to Jay & Kulkulka

coefficients of the r-regression factor 2 apart for specis (jay C7)
this can be repeated for each tidal species (diurnal, semidiurnal)
```

13.6 tidal_discharge

```
tidal discharge
c.f. Jay and Kukulka
function Qt = tidal_discharge(obj,x,R0,h0,b,Qr)
```

13.7 tidal_range

```
predict tidal range
```

14 river-tide/@River_Tide_Morphodynamics_Map

14.1 River_Tide_Morphodynamics_Map

```
container class to store multiple river-tide morphodyanics
    scenarios
```

14.2 fun

```
morphodynamics of a tidal river
either retrive a precomputed scenario or compute and store a new
    scenario
```

15 river-tide/@River_Tide_Network

15.1 River_Tide_Network

```
hydrodynamics and morphodynamics of 1D tidal channel networks
```

15.2 dzb_dt

```
change of bed level over time, when width constant over time
dzb/dt + 1/(p rho w) dQs/dx = 0
```

15.3 evolve_bed_level

```
evolve the bed level of the tidal river network over time
```

15.4 evolve_bed_level_scenario

```
shortcut function for batch simulation runs
```

15.5 generate_delft3d

```
generate a Delft3D 4 model for the channel network
```

15.6 init

```
initial condition
function obj = init(obj)
```

15.7 mg_interpolate

15.8 mg_prepare

15.9 mg_restrict

15.10 mg_step

15.11 read_cfg

15.12 sediment_transport

```
compute sediment transport for the channel network, including
    routing at
junctions
```

15.13 solve

```
determine hydrodynamics
```

16 river-tide/@River_Tide_Network_Simple

16.1 River_Tide_Network_Simple

```
tide in a fluvial delta channel network, extension of 1D river tide
the network is a directed graph
TODO convert from trig-to exponential form
```

16.2 discharge_amplitude

```
discharge amplitude
```

16.3 mean_water_level

```
predict the mean water level
```

16.4 plot_mean_water_level

```
plot tidally averaged water level
```

16.5 plot_water_level_amplitude

```
plot surface elevation amplitude
```

16.6 solve

```
solve for the tide in a fluvial chanel network

boundary condition at end points not connected to junctions
	[ channel 1 id, endpoint id (1 or 2), s0, c0
	...
	 channel n id, endpoint id (1 or 2), s0, c0]

conditions at junctions are specified as cells
	 each cell contains an nx2 array
	 n : number of connecting channels
	 [channel id1, endpoint id (1 or 2), ...
```

```
       channel idn, endpoint id (1 or 2)]

every tidal species for each channel has 4 unknowns
these are 2x2 unknowns for the sin + cos of left and right going
    wave
```

16.7 water_level_amplitude

```
predict the surface elevation amplitude
```

17 river-tide/@River_Tide_SWE

17.1 solve

```
determine river tide by the fully non-stationary FVM and then
    extract the tide
 this is experimental and not yet fully working
```

18 river-tide

```
analysis and prediction of river tides

Sub-Classes:

@River_Tide
       - prediction of river tide in a backwater affected river with
            a sloping bed
@River_Tide_Cai
       - prediction of river tide, method of Cai
@River_Tide_Empirical
       - prediction of river tide, empirical
@River_Tide_JK
       - prediction of river tide, empirical after Jay and Kukulka
@River_Tide_Map
       - mulitple-scenaria container for River_Tide
@River_Tide_Network
       - extension of River_Tide to networks
```

18.1 damped_wave_bvp

```
solved damped wave equation
z'' + a z = 0
z(0) = z0, z(L) = 0
```

18.2 damped_wave_ivp

```
linearly damped wave in rectangular channel
solve tide as initial value problem
damped wave approximation

z'' + a z = 0

x_t = Ax + b
```

18.3 damping_modulus_river

```
damping modulus of the tidal wave for river flow only
```

18.4 rdamping_to_cdrag_tide

```
converts damping rate to drag coefficient
c.f. friedrichs, ippen harleman
```

18.5 river_tide_godin

```
analytic solution to the river tide formulated as boundary value
    problem
in a river with finite length

c.f. Godin 1986
```

18.6 river_tide_transport_scale

18.7 river_tide_transport_scale_5

18.8 rt_celerity

```
celerity of the tidal wave
```

18.9 rt_quasi_stationary_complex

```
quasi-stationary solution of the SWE
TODO staggered grid does not help: q1' needed
```

18.10 rt_quasi_stationary_trigonometric

```
quasi statinary form of the SWE
```

18.11 rt_reflection_coefficient_gradual

```
reflection coefficient for gradual varying cross section geometry
without damping
```

18.12 rt_transport

18.13 rt_wave_equation

```
solve river tide as boundary value problem

input:
omega : [nfx1] angluar frequency of tidal component, zero for mean
    flow
reach : [nrx1] struct
.L    : [1x1] length of reaches
        .width(x,h)   width
        .bed(x,h)     bed level
        .surface(x,h) surface elevation
        .Cd(x,h)      drag coefficient
.bc   : [nd,nf] boundary/junction conditions
         bc(id,if).type : {surface, velocity, discharge} (dirichlet)
         bc(id,if).val : value
opt   : [1x1] struct
       - constant surface elevation
       - deactivative advective acceleration
       .dx : spatial resolution

dimensions:
       nr : nurmber or reaches
```

```
nd : upstream/downstream index
nf : frequency index
```

18.14 rt_z2q

```
determine tidal discharge from water level for tidal wave
in contrast to the inverse, discharge to level,
this is not unique, due to the integration constant
```

18.15 tidal_ellipse

```
tidal ellipse, numerical ode solution
```

18.16 tide_slack_exp

18.17 wave_number_tide

```
wave number of the tide without river flow
c.f. friedrichs, ippen harleman
output :
k : wave number, such that
               z(t,x) = z1(t,0) exp(1i*(omega*t-k*x))

 re(k) : rate of phase change
-im(k) : damping rate

function [k k_low k_high] = damping_modulus_tide(omega,cd,h0,az1)
```

18.18 wavetrainz

```
determine river tide by iterated integration of the surface
    elevation
```

18.19 wavetwopassz

```
two pass solution for the linearised wave equation, for surface
    elevation
```

19 test/river-tide-hydrodynamics

19.1 example_river_tide

19.2 example_river_tide_map

19.3 example_river_tide_read_cfg

19.4 hydrodynamic_scenario

19.5 river_tide_test_plot

19.6 test_bvp2c2

19.7 test_bvp2c_sym

19.8 test_celerity

19.9 test_characteristic_rate_of_change

19.10 test_complex_even_overtide

19.11 test_dronkers_compound

19.12 test_fourier_power_exp

19.13 test_friction

19.14 test_friction_dronkers

19.15 test_friction_dronkers2

19.16 test_fv_compare_schemes

19.17 test_fv_convergence

19.18 test_power_series

19.19 test_reflection

19.20 test_reflection_coefficient_gradual

19.21 test_ricatti

19.22 test_river_tide_hydrodynamics_01

19.23 test_river_tide_hydrodynamics_02

19.24 test_river_tide_hydrodynamics_03

19.25 test_river_tide_hydrodynamics_04

19.26 test_river_tide_hydrodynamics_05

19.27 test_river_tide_hydrodynamics_06

19.28 test_river_tide_hydrodynamics_07

19.29 test_river_tide_hydrodynamics_08

```
hold on;
plot(x,abs(z),'--');
hold on;
plot(x,angle(z),'--');
```

19.30 test_river_tide_hydrodynamics_09

19.31 test_river_tide_hydrodynamics_10

19.32 test_river_tide_hydrodynamics_11

19.33 test_river_tide_hydrodynamics_12

19.34 test_river_tide_hydrodynamics_13

19.35 test_river_tide_hydrodynamics_14

19.36 test_river_tide_hydrodynamics_15

19.37 test_river_tide_hydrodynamics_50

19.38 test_river_tide_hydrodynamics_60

19.39 test_river_tide_hydrodynamics_90

19.40 test_river_tide_hydrodynamics_batch

```
river tide test case batch run
```

19.41 test_river_tide_metadata

19.42 test_river_tide_models

19.43 test_rt_d3d_evaluate

19.44 test_rt_reflection

19.45 test_rt_wave_number

19.46 test_rt_zs0

19.47 test_swe

19.48 test_tidal_river_network

19.49 test_tidal_river_network_z0

19.50 test_tide_slack_exp

19.51 test_wave_number_godin

19.52 test_wave_numer_aproximation

19.53 test_wave_twopass

20 test/river-tide-morphodynamics

20.1 rtm_plot

20.2 test_river_tide_morphodynamics_01

20.3 test_river_tide_morphodynamics_02

20.4 test_river_tide_morphodynamics_03

20.5 test_river_tide_morphodynamics_04

20.6 test_river_tide_morphodynamics_16

20.7 test_river_tide_morphodynamics_17

20.8 test_river_tide_morphodynamics_18

20.9 test_river_tide_transport_scale

21 test/river-tide-network

21.1 test_river_tide_network_01

21.2 test_river_tide_network_02

21.3 test_river_tide_network_03

21.4 test_river_tide_network_04

21.5 test_river_tide_network_05

22 test

22.1 test_rt_transport

22.2 test_stokes_transport

22.3 test_tidal_harmonic_analysis

23 tide

```
analysis prediction of tides in rivers and estuaries by empirical
    and theoretical methods
```

23.1 tidal_constituents

23.2 tidal_energy_transport_1d

```
 energy transport of a tidal wave
```

23.3 tidal_envelope

```
envelope of the tide

input : t time in days
       f surface elevation
ouput : tl time of low water
       vl surface elevation at low water
       ldx index of low water
       th time of high water
       vh surface elevation at high water
       hdx index of high water
       ndx neap index
       sdx spring index
       dmax:
       drange: range per day
```

23.4 tidal_envelope2

```
surface levelation envelope of the tide
low water, high water and tidal range for lunar each day


input:
      time  :
      L     : surface elevation
      order : interpolation order (default 2)
ouput:
      timei : vector eqispaced
      lmini : minimum level
      lmaxi : maximum level
      rangei : range
      midrangei : (min + max)/2, usually different from mean
      phii  : pseudo phase

Note: the pseudo phase phi jumps, this is because if the tide is
    semidiurnal,
     sometimes the lower hw becomes the next day higher then than
        the
      current high water, e.g. there is no smooth transition by
     51min but a jump by 12h
```

23.5 tidal_harmonic_analysis

```
tidal_harmonic analysis
```

23.6 tidal_range_exp

23.7 tidal_range_tri

24 tide-savenije

24.1 savenije_phase_lag

```
phase lag of high and low water

phi : u_river/u_tide < 1

delta_eps_hw = omega*(t_hws - t_hw)
delta_eps_hw = omega*(t_lws - t_lw)

c.f. savenije
```

24.2 savenije_tidal_range

```
tidal range

based on Savenije 2012

x    : distance to river mouth
eta  : range
eta0 : range at river mouth
hbar : mean water depth
phi  : velocity ratio u_tide/u_river
       note: this varies in strongly convergent estuaries
K    : mannings coefficient
I    : residual surface slope I
```

24.3 savenije_tidal_range1

```
tidal range

based on Horrevoets/Savenije, 2004

H0   : tidal range at river mouth
h0   : initial water depth
v    : velocity scale
b    : convergence length
sine : phase lag
K    : Mannings coefficient
Q_r   : river discharge
```

24.4 savenije_timing_hw_lw

```
time of high water and low water
c.f. savenije 2012
```

24.5 tide-savenije

25 tide

```
analysis prediction of tides in rivers and estuaries by empirical
    and theoretical methods
```

25.1 tide_low_high_exp

25.2 tide_low_high_tri