Manual for Package: open-channel-flow
Revision 1:6M

Karl Kästner

November 10, 2019

Contents

1 w/@Backwater1D

1.1 Backwater1D

```
solve the gradually varied flow equation (backwater equation)
in one dimension

c.f. Chow, Bresse
```

1.2 backwater_approximation

```
approximation of the backwater curve by an exponential function
note: this is not necessarily a good approximation
in the case of tide, Qt can be given
```

1.3 backwater_curve_iterative

```
analytic solution of the gradually varied flow equation
c.f. Bresse, Chow
```

1.4 backwater_length

```
backwater length
```

1.5 dh_dx

```
change of depth along channel for the backwater equation
beta : momentum coefficient
this is effectively an equation in h^3
```

1.6 dh_dx_

1.7 dzs_dx

```
change of surface elevation along channel
```

1.8 gvf_x_chow

```
analytical solution to the gradually varied flow equation (
    backwater equation)
c.f. Chow, Bresse
```

1.9 invert

```
determine bed level from surface elevation
(inverse backwater equation)
this is ill conditioned, as the surface is smooth for subcritical
    flow,
even if the bed is not smoth

C : chezy
W : width
Q : discharge
S : bed slope
y0 : surface elevation at outflow
lateral inflow
```

1.10 solve

```
solve the gradually varied flow equation (backwater equation)
C : chezy
W : width
Q : discharge
S : bed slope
y0 : surface elevation at outflow
```

1.11 solve_analytic

```
analytical solution to the gradually varied flow equation (bresse
    method)
u_.^(n-m)./(1-u_.^n)
```

1.12 solve_matrix

2 w/@Potential_Flow

2.1 Potential_Flow

```
numerical solution of the potential flow on a curvilinear grid
(not necessarilly curvilinear)
```

2.2 apply_boundary_potential_old

2.3 assemble_discretization_matrix_rectilinear

```
assemble the discretisation matrix
```

2.4 assemble_potential_matrix

```
assemble the discretisation matrix for potential flow
```

2.5 bc_dirichlet

```
apply Dirichlet boundary conditions
```

2.6 boundary_condition_side_outflow

```
apply boundary conditions for side outflow
p*phi + (1-p)*d/db phi = rhs
y : along channel coordinate
```

2.7 boundary_condition_side_outflow_1

```
apply boundary conditions
p*phi + (1-p)*d/db phi = rhs
```

2.8 contour

```
contour plot of the potential flow solution
```

2.9 cut_boundary

```
cut the boundary from the domain
wa : width of inlet to side channel
wb : width of side channel
```

2.10 cut_rectangle

```
cut a rectangle from the domain
TODO, this requires also an adaptation of the derivative matrices
      -> step over to semi-unstructured mesh
```

2.11 infer_bed_level

```
note: this is pretty much a broken function for the inference of
    stationary
      morphology

Missing:
- rolling down of transverse slope to balance secondary flow in
    bends
- quasi time steippong

at stationary state:
- changes of discharge along the streamlines of discharge are
    balanced
  by a change in depth, to keep the velocity and sediment transport
       constant along the streamline

dz_b/dt = dqs/dx + dqs/dn = 0               (i)
TODO this only true for infinite bends, as sediment can also move
    to the side
dqs/ds = d/s(q/h) = 1/h dq/ds - q/h^2 dh/ds = 0
TODO this is only true in an ifinite bend (ikeda)
dqs/dn = 0
streamlines along discharge or velocity -> does not matter eq (i)
    is direction independent
```

2.12 infer_bed_level2

```
infer the bed level
```

2.13 infer_bed_level3

2.14 infer_bed_level_loop

```
the bed level does not completely converge but starts to oscillate,
this is presumably due to the non-compact kernel implementation of
    the laplacian oberator
```

2.15 objective_bed_level

```
objective function for determining the bed level
```

2.16 old

2.17 plot

```
surface plot
```

2.18 quiver

2.19 sediment_transport

```
compute the sediment transport
```

2.20 solve_potential

```
solve for the flow potential
```

2.21 streamline

```
compute a streamline
```

2.22 surface_elevation

```
compute surface elevation according to Bernoulli's law
```

2.23 test

2.24 velocity_near_bed

```
determine the velocity near the bed
```

2.25 vertical_velocity

```
determine the vertical velocity from continuity
```

3 w/@Potential_Flow_Analytic

3.1 Potential_Flow_Analytic

```
analytical solutions to various depth-averaged potential flow
    problems
```

3.2 derive_lateral_outflow

```
derive potential flow solution to lateral outlfow from an
    infinitely
wide main channel
```

3.3 derive_lateral_outflow_finite_width

```
derive coefficients for lateral outflow in the case of potential
    flow
```

3.4 lateral_outflow

```
potential flow solution to the case of lateral outflow from an
    infinitely
wide channel
```

3.5 lateral_outflow_finite_width

```
analytical potential flow solution to lateral outflow from an
    infinitely
wide channel
```

3.6 streamline

```
numerically follow path along streamline by integrating the
    velocity
```

4 w/@SWE

4.1 SWE

```
Class to solve the (cross sectionally averaged) shallow water
    equation
(st venant equation)
```

4.2 bc_incoming_non_reflecting

```
set non-reflecting boundary condition for the 1D SWE
```

4.3 bc_inflow

```
inflow boundary condition
```

4.4 bc_inflow_low_pass

```
set low frequency Dirichlet, high frequency pass boundary condition
```

4.5 bc_inflow_non_reflecting

```
set non-reflecting boundary condition
```

4.6 bc_level

```
set surface level as Dirichlet boundary condition
```

4.7 bc_level_sommerfeld

```
set surface level as boundary condition by sommerfeld method
```

4.8 bc_nonreflecting

```
set non-reflecting boundary condition
extrapolate 0-order
```

4.9 bc_reflecting

```
set reflecting boundary condition
extrapolate 0-order and invert v
```

4.10 dot

```
time derivative
(only for matlab internal ode-solver)
TODO this is not swe specific
continuity
dA/dt + dQ/dx = I

momentum
```

```
dQ/dt + d/dx( Qu + 1/2 gh^2) = gA(S_f - S_b)
S_b = dz_b/dx
S_f = tau_x/rho_w = C_f u|u|
```

4.11 dt_cfl

```
determine time step required by cfl
```

4.12 energy

```
determine total energy as sump of potential and kinetic energy
this is preserved for fricitionless flows
```

4.13 flux

```
st venant's shallow water equation fluw
```

4.14 flux_lin

```
linearised st-venant equation
```

4.15 fluxmateig

```
eigenvalues und vectors of the swe
```

4.16 jacobian

```
Jacobian of the SWE

dq/dt + J dq/dx = sourceterm
note: d/dx(A*q) = J dq/dx
```

4.17 lindot

```
linearised SWE
width variation not included, goes into rhs force term

[      0,  1] [A]   = [   Q]
[ -u^2+gH, 2u] [Q]_dx [Q^2/A+1/2gA^2/w]_dx - 1/2gA^2/w^2 dw/dx
                                          force term
```

4.18 roe_average

```
roe average for the SWE
```

4.19 solve_analytic

```
linearised analytic solution of the swe
```

4.20 solve_stationary

```
stationary solution to the SWE
```

4.21 source_bed_level

```
source term of the SWE caused by a change of the bed level

Note: this term causes splitting and averaging methods to fail to
     give accurate predictions of the smooth surface at steps of
         the bed
```

4.22 source_friction

```
friction source term of the SWE
```

4.23 source_width

```
source term (reaction term) for channels with variable width
```

4.24 swe_geometry

```
predefined functions to set up channel geometry
```

4.25 swe_ic

```
predefined functions of channel geometries
```

5 w/@SWE_2d

5.1 SWE_2d

```
Dynamic solution of the shallow water equation (depth average, 2D)
```

5.2 apply_boundary_condition_stationary

```
apply boundary condition for stationary flow
```

5.3 assemble_stationary

```
TODO, g should be replaced by gx,gy,gz, see chaudhri
assemble discretisation matrix for stationary flow
```

5.4 solve_stationary

```
solve SWE for statinary flow (dU/dt = dQ/dt = 0)
```

6 w/@Side_Weir

6.1 Side_Weir

```
side weir, analytical solution to (critical) lateral outflow
```

6.2 dzs_dx

```
side weir, along channel surface gradient
```

6.3 surface_elevation

```
along-channel surface elevation for (critical) lateral outflow over
     a side-weir
```

7 w

7.1 Potential_Flow_Map

```
wrapper to store precomputed streamlines of potential flows
```

7.2 chezy2manning

7.3 diffusion_wave

```
propagation of a diffusion wave (flood wave), c.f. ponce
advection
diffusion
where is the bed slope?
friction slope
eddy slope
chow 1988
d(A+A0)/dt + dQ/dx = q
dQ/dt + d/dx betaQ^2/A + gA(dh/dx + Sf + Se) - beta q_i v_i + Wf B
    = 0
A0 ignored
inflow and wind shear ignored
```

7.4 friction_slope

```
friction slope (surface slope) for uniform stationary flow
```

7.5 linear_wave

```
linear wave routing (linearised kinematic wave)
```

7.6 manning2chezy

7.7 manning2drag

8 w/meander-bend/@Equilibrium_Bend

8.1 Equilibrium_Bend

```
Transverse profile of the bed level and bed material grain size in
an equilibrium (infintely long) meander bend
```

8.2 bed_profile

```
predict transverse bed profile of an equilibrium meander bend
```

8.3 bed_profile_uniform

```
transverse profile of the bed level of an equilibrium meander bend
with uniform grain size
```

8.4 calibrate

```
calibrate bend geometry to given profile
```

8.5 dD_dr

8.6 dh_dr

```
across channel derivative of flow depth for a meandering river
```

8.7 dh_dr_uniform

```
transverse gradient of the bed level of an equilibrium meander bend
for the case of uniform bed material
```

8.8 grain_size_profile

```
transverse (across channel) profile of the bed material grain size
in a river meander
```

9 w/meander-bend

9.1 Kinoshita

```
% Public properties
% Public get properties
% Private properties
% Constructor
% Setters and getters
% generic methods
```

9.2 bend_transverse_velocity

```
transverse velocity profile in a meander bend
```

9.3 bend_velocity_near_bed

```
near-bed-velocity in a meander bend
```

9.4 kinoshita_

9.5 random_meander

```
generate a pseudo random meander
```

9.6 test_rozovskii

10 w/old

10.1 UniformFlow

11 w/rating-curve

11.1 ChezyRatingCurve

```
rating curve, Chezy formalism
```

11.2 DynamicKeuleganRC

```
Dynamic Rating Curve, Keulegan roughness formulation
(dynamic = correction for hysteresis loop)
```

11.3 DynamicManningRC

```
Dynamic Rating Curve, Manning roughness formulation
(dynamic = correction for hysteresis loop)
```

11.4 DynamicPowerRC

```
Dynamic Power Law Rating curve
(dynamic = correction for hysteresis loop)
```

11.5 KeuleganRatingCurve

11.6 ManningRatingCurve

11.7 PolyRatingCurve

11.8 PowerRatingCurve

```
stationary rating curve, power law
```

11.9 PowerRatingCurveOffset

```
stationary rating curve, stage-discharge follows power law
```

11.10 RatingCurve

```
Fri Feb 13 10:02:52 CET 2015
rating curve superclass
```

11.11 csarea

```
predict cross sectional area from transverse bed level profile
and surface elevation
```

11.12 csdischarge

```
compute discharge
```

11.13 csperimeter

```
compute wetted perimeter
```

11.14 csradius

```
compute hydraulic radius of the cross section
```

11.15 cswidth

```
determine cross section width
```

11.16 test_PowerRatingCurve

11.17 wfunc

```
determine channel width
```

12 w

12.1 surface_slope

```
surface slope for uniform stationary flow
```

12.2 sw_reflection

```
reflection coefficients of shallow water waves at a sudden change
    of the
cross section (sudden change of admittance)
c.f. lighthill, ippen-harleman
```

12.3 sw_reflection_stepwise

```
time passes and phase shifts
transmission and reflection coefficient depend on direction !
iterative (recursive) reflection and transmission
```

13 w/test/test_Backwater1D

13.1 test_bw1d_solve_matrix

14 w

14.1 test_inverse_backwater_curve

14.2 test_nse_nz

15 w/uniform-stationary-flow

15.1 chezy2drag

15.2 critical_flow_depth

```
critical flow depth in uniform stationary flow
```

15.3 drag2chezy

```
convert drag coefficient to chezy coefficient
g dz_s/dx + cd w u^2/h = 0 (swe formalism)
    - S  +  1/C^2 U^2/H = 0 (chezy formalism)
```

15.4 normal_flow_depth

```
normal flow depth for uniform stationary flow
function H = normal_flow_depth(Q,W,C,S)
```

15.5 normal_flow_depth_

```
normal flow depth in uniform stationary flow
```

15.6 normal_flow_discharge

```
normal flow discharge for uniform stationary flow
```

15.7 normal_flow_slope

```
normal flow slope in uniform stationary flow
```

15.8 normal_flow_velocity

```
normal flow velocity in uniform stationary flow
```

16 w/velocity-profile/@Log_profile

16.1 Log_profile

```
logarithmic profile of the streamwise velocity
```

16.2 df_dh

```
sensitivity of profile with respect to depth
```

16.3 df_dh_

```
sensitivity of profile with respect to depth
```

16.4 df_dln_z0

```
sensitivity of velocity profile with respect to roughness length
```

16.5 df_dln_z0_

```
sensitivity of profile with respect to roughness length
```

16.6 profile

```
vertical profile of the streamwise velocity
```

16.7 profile_

```
scale of velocity at instrument depth to depth average velocity
roughness length and associated standard error can change in time,
i.e. may be passed as vectors

zs    : [1xn] water surface level
zb    : [1x1] bottom level
za    : [1xn] or [1x1]
        level of velocity measurement,
        i.e. level of HADCP beam bin centre, coincides with
            instrument level,
        if the HADCP is horizontally aligned
        only needs to be passed as vector if instrument is
            redeployed or
        becomes misaligned
ln_z0 : [1xn] or [1x1]
        natural logarithm of the roughness length
s     : [1xn] or [1x1]
        standard error of ln_z0
function [fz_mu fz_s fz_sp fz_bias fz_eps] = log_profile(zs,zb,za,
    ln_z0,s,sp,e)
```

16.8 profile_bias

16.9 regmtx

```
regression matrix
```

16.10 ubar

```
depth averaged velocity
```

17 w/velocity-profile/@Log_profile_with_bend_correction

17.1 Log_profile_with_bend_correction

```
vertical velocity profile corrected for bend flow
```

17.2 df_dc

```
sensitivity of the velocity profile with respect to the bend
    correction
parameter c
```

17.3 df_dc_

17.4 du_dz

17.5 fit

```
fit the vertical velocity profile
```

17.6 profile_

```
vertical velocity profile
```

17.7 regmtx

```
regression matrix
```

17.8 u

```
streamwise velocity
```

17.9 u_

```
streamwise velocity
```

18 w/velocity-profile/@Log_profile_with_cubic_wake

18.1 Log_profile_with_cubic_wake

```
log profile with cubic wake
```

18.2 df_dc

```
sensitivity of profile with respect to wave parameter
```

18.3 df_dc_

```
sensitivity of profile with respect to wake parameter
```

18.4 profile_

```
vertical velocity profile
```

18.5 regmtx

```
regression matrix
```

19 w/velocity-profile/@Log_profile_with_dip

19.1 Log_profile_with_dip

```
Logarithmic profile with dip
```

19.2 fit

```
fit the vertical velocity profile
```

20 w/velocity-profile/@Log_profile_with_linear_bend_correction

20.1 Log_profile_with_linear_bend_correction

```
log profile with linear bend correction
```

20.2 df_dc

```
sensitivity of profile with respect to wake parameter
```

20.3 df_dc_

```
sensitivity of velocity profile with respect to wave parameter
```

20.4 du_dz

```
velocity shear along vertical
```

20.5 profile_

```
velocity profile
```

20.6 regmtx

```
regression matrix
```

21 w/velocity-profile/@Log_profile_with_wake

21.1 Log_profile_with_wake

```
logarithmic velocity profile with wake correction
c.f. coles
```

21.2 df_dc

```
sensitivity of profile with respect to wake parameter
```

21.3 df_dc_

```
sensitivity of velocity profile with respect to wake parameter
```

21.4 du_dz

```
velocity shear
```

21.5 profile_

```
predict velocity profile
```

21.6 regmtx

```
log law with wake
u = us/k ln(z) - us/k ln(z0) + us/k (2/H^2 z - 3/H^3 z^2)
```

22 w/velocity-profile/@VP

22.1 VP

```
velocity profile
```

22.2 process_joint

22.3 process_transverse_profile

```
process the transverse velocity profile
```

22.4 process_vertical_profile

```
predict vertical profile error distribution parameter for HADCP
    error estimate
```

22.5 profile_prediction_error

```
 input :
 U    : [nbin x nens]
        - values for each bin (or across section) and ensemble (or
            reference measurement)
        this are estimates estimates of the discharge or the cross
            sectional averaged
        velocity from the raw values
        - the profile should be limited to the effective profiling
            range,
        abobj 75-100m for a 600kHz ADCP

 dn   : distance between HADCP bins
 width : cross section width

 objput:
      sd_n : expected standard deviation for increasing profiling
          range
function [s_rel s_err s_dat rho res m2 u_pred fdx] =
    velocity_variation(U)
 hadcp_prediction_error
```

```
TODO take scales and unscaled velocity to do combine with harmmean
    estimate
note: previus versions:
      residual was computed with respect to the predicted local
          velocity
      mse was not upscaled to cs, as profile was expected to cover
          entire cs
      finite width of cs was not considered
parametric estimate from moments, objliers should be filtered
    beforehand
Note that the median absolute deviation is not a good estimate,
because it may excludes rare events like reverse flow of floods
thus, the only acceptible more robust estimate would be mean
    absolute deviation
```

23 w/velocity-profile/@Vertical_profile

23.1 Vertical_profile

```
vertical profile of the streamwise velocity, superclass
```

23.2 fit

```
fit vertical velocity profile parameter

function obj = fit(obj,U,S,h,binmask)
```

23.3 u

```
predict velocity along the vertical based on profile
```

24 w/velocity-profile

24.1 fit_displacement_profile

```
fit the log profile to the vertical profile of the streamwise
    velocity
```

24.2 lateral_division_method

```
transverse (across channel) profile of the streamwise velocity
in a straight channel
numerical solution
the eps seems incorrect, use better stationary_1d_swe

rho g h S - beta q^2 f / (8 h^2) + d/dy(eps_t dq/dy) = 0
rho g h S - beta q^2 g / (C^2 h^2) + d/dy(eps_t dq/dy) = 0
```

24.3 test_law_of_the_wall_fit

24.4 transverse_profile_parameter

24.5 transverse_velocity_profile

```
transverse profile of the streamwise velocity
c.f. shiono knight
```

24.6 transverse_velocity_profile_olesen

```
transverse profile of the streamwise velocity in a meander bend
```

24.7 transverse_velocity_profile_rozovskii

```
transversal velocity distribution in a bend
Rososkii,
as in the book central differences along the radius and euler
    forward in space
are used, note that since the advent of the computer more advanced
    schemes
could be used (see build in solvers)
cfl condition is not explicitely checked
Rosovsky assumes a constant water level, e.g. does not consider
    superelevation
```

```
I_theta = -1/r dz/d_theta   (p. 22)
d_theta = 1/R ds|_R
=> I_theta = -R/r dz/ds = -R/r I_0
It : (1.32) drop of level per unit angle, identical across section
```

24.8 transverse_velocity_profile_shiono_knight

```
transverse profile of the streamwise velocity, determined
    analytically
by the method of shiono and knight
shape of velocity profile only dependent on lambda, f, H, not slope
```

24.9 transverse_velocity_profile_with_slope

```
stationary 1D shallow water equation across a river section
0 = - g h S0 - tau_b/rho + d/dn (nu h du/dn)
0 = - g h S0 + g u^2/C^2 + d/dn (nu h du/dn)
includes tranvese gradient term

note that shiono/knight 1991 provide an _analytic_ solution,
which takes the form of an expontially decaying side wall effect
```

24.10 vertical_profile_of_velocity_vriend

```
vertical profile of the streamwise velocity, method of de vriend
```

24.11 vertical_velocity_profile

```
vertical profile of the streamwise velocity in non-uniform flow
```

24.12 z2s_rational

25 w/wrapper

25.1 discharge2stage

```
wrapper function
```

25.2 stage2discharge