Manual for Package: open-channel-flow
Revision 1:8M

Karl Kästner

March 28, 2020

Contents

1 @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 bifurcations-and-weirs/@Lateral_Diversion_Finite_Width

2.1 Jb

2.2 Lateral_Diversion_Finite_Width

2.3 dR

2.4 derive

2.5 evalk

2.6 lateral_outflow_finite_width1

2.7 load_functions

2.8 stagnation_point

```
fdx    = isnan(x);
```

2.9 streamline

2.10 streamline_radius_of_curvature

2.11 u_far

2.12 v_far

2.13 velocity

2.14 velocity_near_bed

3 bifurcations-and-weirs/@Lateral_Diversion_Finite_Width_Gradual

3.1 Jb

3.2 Lateral_Diversion_Finite_Width_Gradual

3.3 coefficients

3.4 condA

3.5 dR

3.6 derive

3.7 evalk

3.8 evalk_

3.9 lateral_outflow_finite_width1

3.10 load_functions

4 bifurcations-and-weirs/@Lateral_Diversion_Finite_Width_Gra

4.1 coefficients_old

5 bifurcations-and-weirs/@Lateral_Diversion_Finite_Width_Gradual

5.1 stagnation_point

```
fdx   = isnan(x);
```

5.2 streamline

5.3 streamline_radius_of_curvature

5.4 u_far

5.5 uv1

5.6 uv_side_branch

5.7 v_far

5.8 velocity

5.9 velocity_linear

5.10 velocity_near_bed

5.11 xp

6 bifurcations-and-weirs/@Lateral_Diversion_Wide_Channel

6.1 Lateral_Diversion_Wide_Channel

6.2 derive_lateral_outflow

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

6.3 derive_lateral_outflow_finite_width

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

6.4 lateral_outflow

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

6.5 lateral_outflow_finite_width

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

7 bifurcations-and-weirs/@Lateral_Diversion_Wide_Channel_Map

7.1 Lateral_Diversion_Wide_Channel_Map

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

7.2 streamline

8 bifurcations-and-weirs/@Side_Weir

8.1 Side_Weir

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

8.2 dzs_dx

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

8.3 surface_elevation

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

9 bifurcations-and-weirs

9.1 Lateral_Diversion_Finite_Width_Map

10 open-channel-flow

```
functions for open channel flow, sub modules:

@Backwater1D
       gradually varied flow in 1D (backwater)
@Potential_Flow
       depth averaged potential flow, numerical solution
@Potential_Flow_Analytic
       depth averaged potential flow, analytical solution
rating-curve
       empirical rating curves
@Side_Weir
       analytical solution to lateral outflow over a side weir
@SWE
       dynamical solution of the shallow water equation (saint-
           venant-equation)
       in 1D
@SWE_2d
       dynamical solution of the shallow water equation (saint-
           venant-equation)
       in 2D
velocity-profile
       vertical and transverse velocity profiles of the streamwise
           velocity
```

10.1 hfilter

11 kinematik-and-diffusion-wave

11.1 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
```

11.2 flood_wave_diffusion_coefficient

11.3 linear_wave

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

12 meander-bend/@Equilibrium_Bend

12.1 Equilibrium_Bend

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

12.2 bed_profile

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

12.3 bed_profile_uniform

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

12.4 calibrate

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

12.5 dD_dr

12.6 dh_dr

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

12.7 dh_dr_uniform

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

12.8 grain_size_profile

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

13 meander-bend

13.1 Kinoshita

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

13.2 bend_transverse_velocity

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

13.3 bend_velocity_near_bed

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

13.4 kinoshita_

13.5 random_meander

```
generate a pseudo random meander
```

13.6 test_rozovskii

14 potential-flow/@Potential_Flow

14.1 Potential_Flow

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

14.2 apply_boundary_potential_old

14.3 assemble_discretization_matrix_rectilinear

```
assemble the discretisation matrix
```

14.4 assemble_potential_matrix

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

14.5 bc_dirichlet

```
apply Dirichlet boundary conditions
```

14.6 boundary_condition_side_outflow

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

14.7 boundary_condition_side_outflow_1

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

14.8 contour

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

14.9 cut_boundary

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

14.10 cut_rectangle

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

14.11 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
```

14.12 infer_bed_level2

```
infer the bed level
```

14.13 infer_bed_level3

14.14 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
```

14.15 objective_bed_level

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

14.16 old

14.17 plot

```
surface plot
```

14.18 quiver

14.19 sediment_transport

compute the sediment transport

14.20 solve_potential

solve for the flow potential

14.21 streamline

compute a streamline

14.22 surface_elevation

compute surface elevation according to Bernoulli's law

14.23 test

14.24 velocity_near_bed

determine the velocity near the bed

14.25 vertical_velocity

determine the vertical velocity from continuity

15 potential-flow/@Potential_Flow_Analytic

15.1 Potential_Flow_Analytic

analytical solutions to various depth-averaged potential flow problems

15.2 streamline

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

16 rating-curve

16.1 ChezyRatingCurve

```
rating curve, Chezy formalism
```

16.2 DynamicKeuleganRC

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

16.3 DynamicManningRC

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

16.4 DynamicPowerRC

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

16.5 KeuleganRatingCurve

16.6 ManningRatingCurve

16.7 PolyRatingCurve

16.8 PowerRatingCurve

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

16.9 PowerRatingCurveOffset

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

16.10 RatingCurve

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

16.11 csarea

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

16.12 csdischarge

```
compute discharge
```

16.13 csperimeter

```
compute wetted perimeter
```

16.14 csradius

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

16.15 cswidth

```
determine cross section width
```

16.16 test_PowerRatingCurve

16.17 wfunc

```
determine channel width
```

17 shallow-water/@SWE

17.1 SWE

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

17.2 bc_incoming_non_reflecting

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

17.3 bc_inflow

```
inflow boundary condition
```

17.4 bc_inflow_low_pass

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

17.5 bc_inflow_non_reflecting

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

17.6 bc_level

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

17.7 bc_level_sommerfeld

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

17.8 bc_nonreflecting

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

17.9 bc_reflecting

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

17.10 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|
```

17.11 dt_cfl

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

17.12 energy

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

17.13 flux

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

17.14 flux_lin

```
linearised st-venant equation
```

17.15 fluxmateig

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

17.16 jacobian

```
Jacobian of the SWE

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

17.17 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
```

17.18 roe_average

```
roe average for the SWE
```

17.19 solve_analytic

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

17.20 solve_stationary

```
stationary solution to the SWE
```

17.21 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
```

17.22 source_friction

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

17.23 source_width

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

17.24 swe_geometry

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

17.25 swe_ic

```
predefined functions of channel geometries
```

18 shallow-water/@SWE_2d

18.1 SWE_2d

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

18.2 apply_boundary_condition_stationary

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

18.3 assemble_stationary

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

18.4 solve_stationary

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

19 shallow-water

19.1 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
```

19.2 sw_reflection_stepwise

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

20 test/test_Backwater1D

20.1 test_bw1d_solve_matrix

21 test

21.1 test_inverse_backwater_curve

21.2 test_normal_flow

21.3 test_nse_nz

22 uniform-stationary-flow

22.1 chezy2drag

22.2 chezy2f

22.3 chezy2manning

22.4 chezy2z0

22.5 critical_flow_depth

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

22.6 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)
```

22.7 f2chezy

22.8 ks2z0

22.9 manning2chezy

22.10 manning2drag

22.11 manning2z0

22.12 normal_flow_depth

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

22.13 normal_flow_depth_

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

22.14 normal_flow_discharge

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

22.15 normal_flow_slope

```
energy slope (surface slope) for uniform stationary flow
normal flow slope in uniform stationary flow
```

22.16 normal_flow_velocity

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

22.17 normal_shear_velocity

22.18 shear_velocity

22.19 z02chezy

22.20 z02ks

22.21 z0tochezy

23 velocity-profile/@Log_profile

23.1 Log_profile

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

23.2 df_dh

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

23.3 df_dh_

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

23.4 df_dln_z0

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

23.5 df_dln_z0_

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

23.6 profile

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

23.7 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)
```

23.8 profile_bias

23.9 regmtx

```
regression matrix
```

23.10 ubar

```
depth averaged velocity
```

24 velocity-profile/@Log_profile_with_bend_correction

24.1 Log_profile_with_bend_correction

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

24.2 df_dc

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

24.3 df_dc_

24.4 du_dz

24.5 fit

```
fit the vertical velocity profile
```

24.6 profile_

```
vertical velocity profile
```

24.7 regmtx

```
regression matrix
```

24.8 u

```
streamwise velocity
```

24.9 u_

```
streamwise velocity
```

25 velocity-profile/@Log_profile_with_cubic_wake

25.1 Log_profile_with_cubic_wake

```
log profile with cubic wake
```

25.2 df_dc

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

25.3 df_dc_

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

25.4 profile_

```
vertical velocity profile
```

25.5 regmtx

```
regression matrix
```

26 velocity-profile/@Log_profile_with_dip

26.1 Log_profile_with_dip

```
Logarithmic profile with dip
```

26.2 fit

```
fit the vertical velocity profile
```

27 velocity-profile/@Log_profile_with_linear_bend_correction

27.1 Log_profile_with_linear_bend_correction

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

27.2 df_dc

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

27.3 df_dc_

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

27.4 du_dz

```
velocity shear along vertical
```

27.5 profile_

```
velocity profile
```

27.6 regmtx

```
regression matrix
```

28 velocity-profile/@Log_profile_with_wake

28.1 Log_profile_with_wake

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

28.2 df_dc

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

28.3 df_dc_

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

28.4 du_dz

```
velocity shear
```

28.5 profile_

```
predict velocity profile
```

28.6 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)
```

29 velocity-profile/@VP

29.1 VP

```
velocity profile
```

29.2 process_joint

29.3 process_transverse_profile

```
process the transverse velocity profile
```

29.4 process_vertical_profile

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

29.5 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
```

30 velocity-profile/@Vertical_profile

30.1 Vertical_profile

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

30.2 fit

```
fit vertical velocity profile parameter

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

30.3 u

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

31 velocity-profile

31.1 fit_displacement_profile

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

31.2 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
```

31.3 test_law_of_the_wall_fit

31.4 transverse_profile_parameter

31.5 transverse_velocity_profile

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

31.6 transverse_velocity_profile_olesen

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

31.7 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
```

31.8 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
```

31.9 transverse_velocity_profile_tidal_channel

31.10 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
```

31.11 vertical_profile_of_velocity_vriend

vertical profile of the streamwise velocity, method of de vriend

31.12 vertical_velocity_profile

vertical profile of the streamwise velocity in non-uniform flow

31.13 z2s_rational

32 wrapper

32.1 discharge2stage

wrapper function

32.2 stage2discharge