Manual for Package: adcp
Revision 4M

Karl Kästner

September 2, 2021

Contents

1 @ADCP

1.1 ADCP

```
ADCP superclass
converts ADCP fixed integer raw data to floats with SI units
provides functions for ADCP data manipulation
calculated from the water temperature and sound frequency
```

1.2 Ds

```
depth of bin, distance between water surface (z_s) and (z_i)

Ds = z_s - z_bin

does not correct for tilts
```

1.3 Dt

```
projected distance from transducer to cell centres
if the instrument is not tilted, this is the vertical distance (
    depth)
between the transducer and cell centres
does not account for transducer depth
```

1.4 R

```
unprojected (slanted) distance between the transducer and cell
    centres
```

1.5 adc_current_slope

```
instrument type specific slope for converting raw current to Ampere
c.f WorkHorse Commands and Output Data Format, March 2014
c.f. XMT Voltage and Current Channels
originally undoccumented by RDI, and taken from Shields 2010
```

1.6 adc_voltage_slope

```
instrument series specific conversion factors for voltage
c.f. WorkHorse Commands and Output Data Format, March 2014
c.f. XMT Voltage and Current Channels
originally undoccumented by RDI, and taken from Shields 2010
```

1.7 assign_file

```
ensemble indices of each file
```

1.8 assign_water_level

```
assign water level to adcp ensembles (combine gauge with boat data)
```

1.9 average_profile

```
average backscatter for each sample within an specific interval
```

1.10 backscatter2ssc

```
wrapper for backscatter conversion
```

1.11 binsize

```
bin size (vertical distance between two bins)
```

1.12 blnk

```
blanking range, range from transduce to centre of first bin
```

1.13 btrange

```
convert raw btrange to vertical distance (projected distance) of
    the bed
level below the transducer, when the transducer is looking
    vertically down
this is the depth less the transducer depth
```

1.14 calc_backscatter

```
backscatter from echo intensity
```

1.15 clock_offset_STATIC

```
dt : median difference between adcp clock and UTC
sd_dt : standard error of dt
```

1.16 convert_raw_binprops_STATIC

```
convert the raw bin properties to si-units
```

1.17 convert_raw_serial_STATIC

```
convert bytes of serial number into single number
big endian system
```

1.18 convert_raw_time_STATIC

```
convert measurement time stamps into matlab internal format
```

1.19 convert_raw_velocity

```
convert scaled integer raw velocity to float SI (m/s)
```

1.20 convert_raw_velocity_STATIC

```
convert raw velocity to SI units (m/s)
```

1.21 copy

```
copy constructor
```

1.22 distmidbin1

```
convert raw distance to first bin centre to SI
```

1.23 file_ensemble_index

```
ensemble index eid_f with respect to file for ensemble eid
```

1.24 file_index

```
first and last ensemble index of of a file
```

1.25 filetime_min

```
start time of each file
```

1.26 fill_coordinate_gaps

```
fill gaps in ensemble coordinates
```

1.27 filter_range

```
filter HADCP velocity by detecting the last valid bin
if the bacscatter does not decreas over 10 bins, than obtstacle or
    intersection
```

1.28 heading_rad

```
convert raw instrument heading angle to [rad]
```

1.29 instrument_depth_m

```
depth of instrument (for submerged deployments)
```

1.30 instrument_to_ship_STATIC

```
transform velocities from instrument reference to ship reference
by correcting for pitch_rad and roll_rad

input
vel   : float [arbitrary unit] instrument reference
btvel : float [arbitrary unit] instrument reference
pitch_rad : float [radians] true pitch_rad, not measured pitch_rad
```

```
roll_rad : float [radians]

output
vel and btvel [input unit] ship reference
```

1.31 lngthtranspulse

```
convert raw transmit pulse length to SI units (m)
```

1.32 load_RSSI_values_STATIC

```
load instrument specific backscatter conversion parameters
```

1.33 nbins

```
number of bins for each file
```

1.34 near_field_correction

```
new fiel correction of the acoustic backscatter
c.f. wall 2006
Psi : (nr,1) near field correction factor
```

1.35 nens

```
number of ensembles
```

1.36 pitch_rad

```
convert raw pitch to radians
```

1.37 pressure_bar

```
convert raw pressure to bar
```

1.38 range2binid

```
convert distance to transducer to bin index
```

1.39 roll_rad

```
convert raw instrument roll angle to [rad]
```

1.40 rotate_velocity

```
rotate the velocity in the horizontal plane with respect to the
    directional
vector dir
dir : direction of the transect
```

1.41 rotate_velocity_sw

```
rotate velocity to local streamwise reference
input velocity can have arbitrary reference
```

1.42 ship_to_earth_STATIC

```
converts velocity from ship to earth coordinate reference
expects input arguments informat:
vel   : float arbitrary unit
btvel : float same unit as vel
heading_rad: float [radians]
```

1.43 sort_STATIC

```
sort files by start time
```

1.44 squeeze_STATIC

```
cut ensembles, skip ensembles or average ensembles in time

adcp : adcp structure
dt   : time between output ensembles in seconds
mode : {'average', 'skip'}
mask : selection of ensembles to keep (computed from dt if not
    provided)
                             fprintf(1,'Progress: %g\n%% %gs\n',idx/
                                 nt,tlast);
```

1.45 temperature_offset_C

```
instrument specific temperature offset
```

1.46 to_abs

```
velocity magnitude
```

1.47 transducer_temperature_C

```
convert raw transducer temperature to SI units [Celsius]
T   : (1,nt) water temperature
```

1.48 verify_pc_time

```
verify the time stored in the data file
```

2 @Ensemble

2.1 Ensemble

```
container for ADCP ensemble data and properties
```

2.2 calc_beamcoords

```
claculate positions in world coordinates where the individual beams
     hit the bottom
```

3 @HADCP

3.1 HADCP

```
coverts raw data of horizontal ADCPs into physical quantities
and provides functions for data processing
```

3.2 beam_to_instrument_STATIC

```
transform the 3 beam velocities into a set of 2 orthogonal
    velocities
and 1 error velocity
This uses always three beams (no two beam solutions)

input
vel       : float [arbitrary unit] beam reference system
btvel     : float [arbitrary unit] beam reference system
beamangle : float [radians]

output
vel and btvel [input unit] instrument reference system
```

3.3 bootstrap_backscatter

```
bootstrap uncertainty of the backscatter parameters
```

3.4 calc_beam_spreading_cone

```
beam spreading
Note: beams spread in the form of bessel functions
      this is the engineering approach as cones, which is however
          not
      a good approximation, it is better to approximate it as a
          gaussian
```

3.5 calc_bin_coordinates

```
get the cartesian (world) coordinates of the HADCP central beam
    bins
```

3.6 calibrate_backscatter

```
calibrate backscatter to sediment concentration by the method of
    Sassi
```

3.7 filter_velocity

```
filter outliers in velocity data
```

3.8 firmware_fix_STATIC

```
correct RDI HADCP firmware bug (2014)
this bug successively invalids data every 4th-bin, which led to 3-
    beam solutions
and consequentially jumps of the transformed velocities
                     vel_s(:,fdx,:) = vel(:,fdx,:);
                     vel_b(:,fdx,:) = vel(:,fdx,:);
```

3.9 fixnan

```
interpolate invalid bin-samples from last and next ensemble
```

3.10 instrument_to_beam_STATIC

```
transform the 3 beam velocities into a set of 2 orthogonal
    velocities
and 1 error velocity
This uses always three beams (no two beam solutions)

input
vel       : float [arbitrary unit] beam reference system
btvel     : float [arbitrary unit] beam reference system
beamangle : float [radians]
```

```
output
vel and btvel [input unit] instrument reference system

mode : beams used for all transformations
      123, 12, 23, 13
```

3.11 reorder_velocity_STATIC

```
reorder the HADCP velocity data into the first three slots, the
    HADCP
has just three beams, but the software stores data for
four beams, similar to the four beam VADCPs
```

3.12 to_beam_STATIC

```
wrapper for conversion to beam velocity
Note that back-conversion to beam velocity is not unique in case of
     3 beam
solutions (as RDI instruments doe not store which beams were used)
    and
if instrument internal bin-mapping is used (whichs precise
    algorithm remains
an RDI secret)
```

3.13 to_earth_STATIC

```
wrappter to transform velocities to world coordinate reference
```

3.14 to_instrument_STATIC

```
wrapper to convert velocity to instrument coordinate reference
```

3.15 to_ship_STATIC

```
wrapper for conversion to ship velocity
```

4 @RDI_mmt

4.1 RDI_mmt

4.2 read

4.3 write

5 @VADCP

5.1 VADCP

```
coverts raw data of vertical ADCPs into physical quantities
```

5.2 assign_transect

```
assign transect index to ensembles
```

5.3 backscatter_report

```
 human readable output of calibration properties
%      fprintf(['Parameters and uncertainty with respect to 95%%
   confidence\n']);
```

5.4 beam_to_instrument_STATIC

```
transform the 4 beam velocities into a set of 3 orthogonal
    velocities
and 1 error velocity

input
vel      : float [arbitrary unit] beam reference system
```

```
btvel     : float [arbitrary unit] beam reference system
beamangle : float [radians]

output
vel and btvel [input unit] instrument reference system

TODO account for NaNs either by three beam solution or
    interpolation
```

5.5 bottom_track_STATIC

```
compute bottom track coordinates
```

5.6 bscalibrate

```
backscatter to sediment calibration

calibtation subroutine
M_ref : sediment concentration calibration values
d_k   : depth of virtual reference value K
        (choose close to receiver, but out of near field, e.g.
            within 2m .. 4m)
TODO : better documentation of input values
TODO : rename nk into ik, bacause it is an index and not a length
TODO rename r_ref and d_k into r_1 and r_2
```

5.7 bsgrid

```
evaluate the objective function at the selected points
```

5.8 bsinvert

```
backscatter inversion
```

5.9 bsjackknife

```
compute the jackknife estimates of the parameters and their
    covariances
```

5.10 bsjointcalibration

```
calibrate backscatter
```

5.11 btvel_from_position

```
determine boat velocity from bottom track, inverse of bottom track
```

5.12 calc_ssc

```
calculate the backscatter
```

5.13 cdf

```
compute and plot cumulative distribution (cdf) of the velocity
    components
```

5.14 convert_nFiles

```
convert coordinates of NMEA-nFiles
```

5.15 correct_coordinates

```
correct the bottom coordinates for pitch and roll
```

5.16 correct_for_platform_velocity_STATIC

```
correct for platform (boat) velocity, this is the negative bed
    velocity
```

5.17 depth_average_velocity

```
average the velocity over depth
```

5.18 depth_integrate

```
depth integrate the velocity to obtain specific discharge
```

5.19 depth_integrate_sediment_discharge

```
depth integrated sediment discharge
```

5.20 filter_velocity

```
filter the velocity data
```

5.21 fit_sediment_concentration_profile

```
fit_suspended_sediment_concentration_profile(obj,profile_cls,
    ensmask,nwin)
```

5.22 fit_velocity_profile

```
fit velocity profile to the streamwise velocity
```

5.23 map_z

```
z-mapping, i.e. correct for roll and pitch of instrument
```

6 @VADCP/old

6.1 assign_crossing

7 @VADCP

7.1 optstr

```
string of arguments, for file name generation
```

7.2 plot_track

```
plot the boat track
```

7.3 plot_velocity_components

```
plot the velocity components
```

7.4 process

```
process VADCP data
```

7.5 range2depth

```
depth below transducer for individual bins of the beams
```

7.6 rangemask

```
mask all bins in range
```

7.7 to

```
transform velocity to given reference
```

7.8 to_beam_STATIC

```
convert velocity data to beam reference
```

7.9 to_cs

```
transform velocity to cross section references
cs-velocity is here defined as the velocity orthogonal to the cs
      % [ 0  1][c -s]=[-s c]
      % [ 1  0][s  c] [ c s]
```

7.10 to_earth_STATIC

```
transform coordinates to cartesian world reference system (earth)
```

7.11 to_sw

```
transform velocity with respect to depth averaged streamwise
    velocity
```

7.12 velocity_near_bed

```
velocity near the bed
```

7.13 xy2nts

```
project coordinates onto a single cross section and assign them nz-
    coordinates at a single cross section
TODO this should be part of transect
```

8 adcp

```
adcp : processing of Acoustic Doppler Current Profiler (ADCP) data

Processing in 3 Levels:

Level 0 : Read in of raw-data (externally provided by ADCPtools,
    Vermeulen et al.)

Level 1 : VADCP, HADCP, SPADCP
       - convert raw data to CI units (m,s,kg)
       - transform velocities to arbitrary coordinate references
```

```
        - depth averaging and integration
        - fit velocity profiles
        - convert backscatter to suspended sediment concentration

Level 2 : CrossSection
        - interpolate and integrate for cross sections

Instruction:
        see and run saggau/sanggau_process_discharge for a working
            example
        to process VADCP discharge at a non-tidal station
```

8.1 ADCP_Bin

```
 ADCP bin (single velocity values)
```

8.2 SPADCP

```
 stream pro acoutic current doppler profiler
```

9 cross-section/@ADCP_Transect

9.1 ADCP_Transect

```
 zero dimensional processing of ADCP data
 no resampling, meshing or gridding
```

9.2 assign_to_transect

```
 assign ensemble to respective transects
 this has a side-effect (writes to) the adcp object,
 but values of induvidial cross sections remain unaffected by each
     other
```

9.3 compare

```
 discharge summary
```

9.4 detect_crossings

```
detect consecutive navigation of transects (channel crossings)
```

9.5 detect_crossings_circling

```
separatate individual navigation of transects,
for cases when the boat goes in circles and crosses the branches
    one after
the other before returning to the original cross section,
thus the boat does not turn at the other bank to return across the
    same section
and always navigates the cross section in the same direction
```

9.6 detect_crossings_returning

```
groups the ensembles into transects,
one transect is defined as all ensembles recorded during the time
    the boat
moved from one bank to the other (return is defined as separate
    transect)
```

9.7 detect_rounds

```
detect rounds, i.e. when boat returns to initial position
```

9.8 export_mmt

```
export RDI mmt
```

9.9 extrapolate_to_bank

```
extrapolate values to bank
```

9.10 fit

9.11 integrate_discharge

```
integrate discharge

Q  = sum q
q  = A_n*u_s = h dn us
   = h * [dx, dy]*[-v; u]
   = h * dt * [-ub, -vb] * [-v; u]

note that uvb * dt is usually more accurate than dx of GPS position
    ,
if uvb determined by doppler shift of ADCP bottom echo,
except when the GPS position (or velocity) is determined from the
    carrier frequency

note that projection can be left out, if cs is defined with
    transect individual end points,
but not recommended, if there are strong secondary currents as
    encountered at
bends or bifurcations
```

9.12 plot

```
plot the transect as a line in cartesian coordinates
```

9.13 plot2d

```
plot transects
```

9.14 plot_rounds

```
plot rounds (consecutiver transects) navigated with the boat
```

9.15 resample

9.16 xy2nts

10 cross-section/@CrossSection

10.1 CrossSection

```
Level-3 ADCP data processing, projection to cross section and
    integration/averaging
```

10.2 calc_auxiliary_quant

```
compute auxiliary quantities
```

10.3 compare

```
interpolate for all cross-sections the values to the same time-slot
for comparison
```

10.4 determine_time_slots

```
split data set into specific time slots
```

10.5 discharge

```
integrate the discharge over all finite elements of the cross
    section
```

10.6 extrapolate_S

```
extrapolate missing values along the vertical
```

10.7 extrapolate_backscatter

```
extrapolate the backscatter
```

10.8 extrapolate_backscatter_2d_STATIC

```
extrapolate backscatter to bed, surface and banks
```

10.9 extrapolate_bed_profile

```
extrapolate bed profile to channel banks
```

10.10 extrapolate_n

```
extrapolate value beyond end of cross section
```

10.11 extrapolate_velocity

```
extrapolate the velocity to the bank, bed, and surface
```

10.12 extrapolate_velocity_1d_STATIC

```
extrapolate depth averaged velocity
```

10.13 extrapolate_velocity_2d_STATIC

```
extrapolate velocity to banks, surface and bottom
TODO, this is only applicable for Grid2
```

10.14 fit_bathymetry_2d

10.15 fit_bed_profile

```
fit the bed profile, has to precede n-z meshing of the cross-
    section
```

10.16 fit_cross_section

```
fit the optimal cross section as the main axis of the transect
by regressing a line through the measurement points in the x-y
    plane

y = c0 + c1 x
```

10.17 fit_vertical_profile_of_velocity

```
fit vertical profile of the streamwise velocity

this function will work with both ensenble data, eg. U_bin taken
    from ensembles,
as well as gridded data, (U_bin taken from the velocity grid)

input
cs    : struct       : cross section averaged data
U_bin : [nrow x ncol] : vertical profiles of stream wise velocity
Z_bin : [nrow x ncol] : positions of bin above bottom for each
    element in U_bin
ens.N : [ncolx1]     : position of each column of U in along the
    cross section
ens.H : [ncolx1]     : depth of each column of U
ens.sH : [ncolx1]     : std of depth at each colum of U
ens.ldx : [ncolx1]   : last valid sample in column of U
dw_z0  : scalar   : grid cell size for grid_n
obj.roughnessmethod        : method to use for the computation
output:
grid_n : struct     : function of u_s and z_0 along cross section
us_ens, ln_z0_ens, U_ens : local estimates for input ensembles/grid
     columns
          not returned by every obj.roughnessmethod
```

10.18 fit_water_level

```
fit water level from depth measurement
this works only if the ADCP is stationary
```

10.19 generate_mesh_tn

```
generate 1+1D mesh over time and across section
```

10.20 generate_mesh_tnz

```
generate a t-n-z mesh
```

10.21 optstr

```
string of options, for file name generation
```

10.22 plot_n_quiver

```
plot quiver across section
```

10.23 plot_nz

```
plot along n and z
```

10.24 plot_nz_quiver

```
quiver plot of velocity across section
```

10.25 plot_tn

```
plot over time and across channel
```

10.26 plot_xyz

```
plot values in "val" in the 2D cross section, where the cartesian
rather than the local coordinates of the cross-section are used
```

10.27 process_backscatter

```
process backscatter, i.e. fit to cross-section grid from bin-values
```

10.28 process_backscatter_tn

```
      process depth integrated backscatter over time t and acrross
          section N
note: backscatter is processed as flux
       due to high concentration and backscatter near the bottom,
      the inner rpoduct of the discharge and concentration
      \bar u \bar c_s is not a good estimate of the
     depth averaged sediment flux \bar{u c_s}
```

10.29 process_backscatter_tnz

```
process the backscatter in 2+1D (time, across channel and along
    vertical)
```

10.30 process_discharge

```
process the discharge
```

10.31 process_velocity_tn

```
process the velocity data
```

10.32 process_velocity_tnz

```
process velocity data in 2+1D (time, across-section and along
    vertical)
```

10.33 summarise

```
summarize discharge of cross section
```

10.34 var_n

```
return value stored in field "fieldname" at position "N" in the
    cross section
```

10.35 var_t

```
return value stored in filed "fieldname" at time t
cross sectionally integrated or averaged value
```

10.36 var_tn

```
return values of field "fieldname" at time t and position N along
    cross section
typically depth integrated or averaged values
```

10.37 var_tnz

```
generically return value stored in field "fieldname" at time t and
    position N
```

11 cross-section

11.1 complete_profiles

```
fill gaps in profiles
assumes profile to be constant in time, this is not true
for tidal flow in compound cross sections and near banks
```

11.2 define_transect

```
gui user selection of cross-section end points
```

11.3 discharge_division

```
discharge division ratio
```

11.4 discharge_summary

```
compute and store discharge summary
             q_tn = cs.q_tn(ti);
             ndx      = abs(N)<=Nlim;
             Qi       = cs.dw*sum(q_tn)';
             Qi_centre = cs.dw*sum(q_tn(ndx,:))';
             Q = [Q; Qi];
             Q_centre = [Q_centre; Qi_centre];
```

11.5 load_vadcp_discharge

```
load previously computed vadcp discharge (auxiliary function for
    plotting)
this function stacks data from several vadcp reference measurements
     into one structure
This assumes that all data sets where processed with the same
    settings
```

11.6 split_transect2

12 hadcp/@HADCP_Discharge

12.1 HADCP_Discharge

```
superclass for HADCP discharge estimation methods
```

12.2 fit

```
fit the model parameter for HADCP discharge prediction,
estimate errors with the Jacknife method
```

13 hadcp/@HDischarge

13.1 Hbin

13.2 calc_specific_discharge_weights

```
calculate unite discharge weights
```

13.3 estimate_discharge

```
integrate and scale specifc discharge to total discharge for each
    ensemble
```

14 hadcp/@HIVM

14.1 HIVM

```
Index velocity method of Horizontal ADCP data
```

15 hadcp/@IVM

15.1 IVM

```
index velocity method
```

16 hadcp

16.1 ESM

16.2 ESM_individual

16.3 SDM

```
Specific Discharge Method
upscale specific discharge to cross sectionally integrate discharge
    ,
 than average
this method is provenly less accurate than averaging before
    upscaling
```

16.4 VPM

```
velocity profile method
correct individual bin velocities for vertical velocity profile
    variation,
then averagem, then upscale to cross sectionally integrated
    discharge
```

16.5 hadcp_homogenize_profile

```
homogenize the hadcp profile
```

16.6 hadcp_homogenize_profile2

```
homogenise the horizontal velocity profile
```

16.7 wavg

```
weighted average ?
```

16.8 wavg_mean

```
weighted average
```

16.9 wopt

```
optimal weights for averaging (lumped) velocities that are each
    associated
with error variance s2
```

17 adcp

```
adcp : processing of Acoustic Doppler Current Profiler (ADCP) data

Processing in 3 Levels:

Level 0 : Read in of raw-data (externally provided by ADCPtools,
    Vermeulen et al.)

Level 1 : VADCP, HADCP, SPADCP
       - convert raw data to CI units (m,s,kg)
       - transform velocities to arbitrary coordinate references
       - depth averaging and integration
       - fit velocity profiles
       - convert backscatter to suspended sediment concentration

Level 2 : CrossSection
       - interpolate and integrate for cross sections

Instruction:
       see and run saggau/sanggau_process_discharge for a working
           example
       to process VADCP discharge at a non-tidal station
```

17.1 smooth_track

```
smooth a repeatedly navigated (circular) track to produce and
    idealized
average track
```

17.2 streawise_velocity

```
rotate ensembles in stream direction (transverse velocity
    integrates to zero)
```

18 test

18.1 example_backscatter_coefficient_2

18.2 test_backscatter_coefficient

18.3 test_bedslope

18.4 test_delta_z_correction

18.5 test_depth_range

18.6 test_linearisation

18.7 test_procTrans_vele

18.8 test_rotvel

18.9 test_sanggau_load_bed_level_2016

18.10 test_sanggau_rc

19 adcp

```
adcp : processing of Acoustic Doppler Current Profiler (ADCP) data

Processing in 3 Levels:

Level 0 : Read in of raw-data (externally provided by ADCPtools,
    Vermeulen et al.)

Level 1 : VADCP, HADCP, SPADCP
       - convert raw data to CI units (m,s,kg)
       - transform velocities to arbitrary coordinate references
       - depth averaging and integration
       - fit velocity profiles
       - convert backscatter to suspended sediment concentration

Level 2 : CrossSection
       - interpolate and integrate for cross sections

Instruction:
       see and run saggau/sanggau_process_discharge for a working
           example
       to process VADCP discharge at a non-tidal station
```

19.1 zztransform

```
 non-linear mapping for bin coordinates when depth averages between
     ensembles
 for avaraging several ensembles

       preserve discharge w int u_avg dz = int int u dz dn = Q
       perserve shear stress is the same (u_avg)^2_s = mean((u_s)^2)
       preserve sediment transport w int u_avg c_avg dz = int int u
           c dz dn
     preserve rouse number

       alternative : correct parameters for effects of averaging

 several approaches :
 s-transform : z_1' = H0/H1 z_1, perserves u_bar
                              does not preserve u_* (du/dz|_0)
 clipping   : z_1' = z_1, z_1 < H0, does not preserve u_bar
                                   unclear if H0>H1
```

```
                                perserves (du/dz)_0 (u_*)
zz-transform : perserve both u_bar and u_
TODO this is non-monotoneous when difference in H0 and H1 is large
```