ADCP Velocity Computation: Correction and Rotation to East-North-Up Co-ordinate System, Three-beam Solutions and Screening

Rotation of Velocities to East-North-Up Co-ordinate System

Velocity data acquired from RDI and Nortek ADCPs can take on several forms. The original, unaltered velocity data, read from the raw files is normally presented in the **MAT** file product as *adcp.velocity* (RDI) or *data.velocity* (Nortek). (Please note that the data structure names for RDI is *adcp* and for Nortek we use *data*, subsitute *data* for *adcp* when looking at Nortek data). Also in the **MAT** file product is the config struct that contains information on how the data was collected. Based on the configuration, the original data is processed to present the velocities relative to East-North-Up: this is the final data in the **netCDF**, **MAT** files and in the daily current plot **PNG** and **PDF** (see *adcp.u, adcp.v, adcp.w* in the **MAT** file and in the MAT file description below). u,v,w velocities are always relative to geographic North and local gravity (for Up).

In general, the aim of ADCP post-processing is to replicate the functionality in the manufacturer's post-processing software: RDI's *winADCP* and Nortek's *S urge* and *Storm* (which replaced *ExploreP*). RDI provides documentation on their rotations, see: ADCP Coordinate Transformation and the more general guide Acoustic Doppler Current Profiler: Principles of Operation - A Practical Primer (as a background only). Nortek's rotations are documented online, see: http://www.nortek-as.com/lib/forum-attachments/coordinate-transformation/view or the matlab code directly: post-2-71782-Xform.m. See RDI software support login for RDI software and http://www.nortek-as.com/en/support/software for Nortek software.

The coordinate system parameter determines the type of data that is acquired and how it is processed to produce u,v,w velocities. In the **MAT** file product, see *config.coordSys*. If the co-ordinate system is set to 'Beam', the original data contains the radial (along-beam) velocities for each of the 4 or 5 beams, otherwise the velocities are rotated to an orthogonal co-ordinate system: 'Instr' (RDI) or 'XYZ' (Nortek) means the original velocities have been rotated to a basis defined by horizontal plane of the instrument with an azimuth relative to the direction of beam 3, while 'Earth' is relative to the onboard magnetic compass or a fixed direction defined by config.heading_EH (RDI only). Our default setting is 'Beam' so that the most raw form of the data is collected. 'Ship' is not recognized/used. For RDI ADCPs, when the co-ordinate system is either 'Earth' or 'Instr', the 4th row in the *adcp.velocity* matrix is the error velocity (the measurement error on the velocity at that bin and time).

RDI sensor source and sensor availability parameters determine if additional sensors are available to be used for rotation, such sensors include: magnetic compass, tilt sensor (for pitch and roll), depth, conductivity and temperature (the last three are used to determine the sound speed, however we normally fix that value, see *config.soundspeed_EC*). In the **MAT** file, see *config.sensorSource_EZ* and *config.sensorAvail.* If the magnetic compass is available, it is often used for Earth co-ordination rotation. The Nortek ADCPs always have compass and tilt sensors and prescribed sound speeds.

Magnetic compasses are not reliable when in close proximity to varying electromagnetic or magnetic fields (electric power cables or the steel instrument platform can cause the compass to be inaccurate). To improve the data, we normally supply a fixed heading from the site and device metadata. If the device is mobile, deployed on a mooring, cabled profiler or glider, a mobile position sensor is normally supplied. Here is an example of a mobile ADCP site: http://dmas.uvic.ca/Sites?siteId=100206 and here is an example of fixed site: http://dmas.uvic.ca/Sites?siteId=1000359. If the device is mobile and is attached to a device that can supply better orientation information, such as an optical gyro, the data processing code will make use of that device when its sensors are assigned to the heading/pitch/roll of the ADCP's site (we currently do not yet have an example of such a scenario). If neither fixed or mobile heading/pitch/roll are assigned, then the system defaults to devices' internal sensors. When using the onboard magnetic compass, a correction is applied for magnetic declination (if onboard heading is set manually (RDI only), magnetic declination correction is not done). For RDI only, we replicate RDI's gimbal correction for pitch.

Rotation of the input velocities to product East-North-Up velocities (u,v,w) is performed accounting for the incoming co-ordinate system, types and sources of the data. This is somewhat complicated, therefore, all of the processing steps are documented in the **MAT** files, see *adcp.processingComments.* As an example, here is a common scenario: raw data has been collected in the 'Earth' co-ordinate system defined by the onboard magnetic compass and pitch /roll senors, but the instrument is stationary with a known fixed heading, then difference between compass heading and the fixed is calculated and used to rotate the East and North velocities. In that case, *adcp.processingComments* would say (use the matlab command char):

```
>> char(adcp.proccessingComments)

adcp.velocity(1:3,:,:) contains the unaltered velocity data relative to the co-ordination system: Earth. acdp.
velocity(4,:,:) is the RDI error velocity. adcp.u/v/w are processed velocities relative to true East/North/Up,
respectively.
Using a fixed true heading of 228 degrees from metadata.
This device does not have a fixed value for pitch or pitch sensor assigned; using onboard sensor for pitch
(pitch gimbal correction applied).
This device does not have a fixed value for roll or roll sensor assigned; using onboard sensor for roll.
Beam range bins were corrected for tilt by nearest vertical bin-mapping (RDI method) onboard the device (new
pitch/roll values not applied). Average or fixed tilt is 3.14 degrees. adcp.range is the vertical range to the
corrected bin centres.
Rotated Earth co-ordinate, device-supplied adcp.u/v to true North from onboard magnetic North, using a fixed or
variable external source true heading (pitch/roll rotation unchanged).
adcp.backscatter (relative volume backsctter) calculated from 0.45*adcp.intens + 20*log10(adcp.range) +
2*config.soundAbsorptionCoefficient*adcp.range, adcp.meanBackscatter calculated by beam-averaging the volume
backscatter, adcp.meanBackscatter and adcp.backscatter have units of relative dB.
For information on the various processing steps applied, see http://wiki.neptunecanada.ca/display/DP/5.
```

For advanced, interested users, here is the core code that does the rotation:

```matlab
    velocityTrue = nan(3, n*p);
    maxNumPosData = max([length(heading) length(pitch) length(roll)]);  % either 1 or p, the number of time
stamps
    for i = 1:maxNumPosData
        if any(isnan([heading(min(length(heading),i)) pitch(min(length(pitch),i)) roll(min(length(roll),
i))]))
            M = nan(3,3);
        else
            % heading/pitch/roll matrices - this isn't the standard rotations, but it matches what winADCP
does, and this is how it was originally implemented:
            % I considered various ways to get the heading index - pointers, use an eval, etc, but this is
actually faster: heading(min(length(heading), i))
            M1 = [cosd(heading(min(length(heading),i))) sind(heading(min(length(heading),i))) 0; -sind
(heading(min(length(heading),i))) cosd(heading(min(length(heading),i))) 0; 0 0 1];
            M2 = [1 0 0; 0 cosd(pitch(min(length(pitch),i))) -sind(pitch(min(length(pitch),i))); 0 sind(pitch
(min(length(pitch),i))) cosd(pitch(min(length(pitch),i)))];
            M3 = [cosd(roll(min(length(roll),i))) 0 sind(roll(min(length(roll),i))); 0 1 0; -sind(roll(min
(length(roll),i))) 0 cosd(roll(min(length(roll),i)))];
            M = M1 * M2 * M3;
            if strcmp(Config.orient, 'Up') % negate 1st & 3rd column
                M(1:3,1) = -M(1:3,1);
                M(1:3,3) = -M(1:3,3);
            end
        end

        % apply heading/pitch/roll rotations
        if maxNumPosData == 1
            velocityTrue = M * velocityScreenedReshape(1:3,:);
        else
            velocityTrue(:, (i-1)*n+1:1:i*n) = M * velocityScreenedReshape(1:3, (i-1)*n+1:i*n);
        end
    end
    % extract u,v,w,error values - singleton dimensions handled later
    velocityTrue = reshape([velocityTrue; velocityScreenedReshape(4,:)], [m n p]);
    ADCP.u = squeeze(velocityTrue(1,:,:));
    ADCP.v = squeeze(velocityTrue(2,:,:));
    ADCP.w = squeeze(velocityTrue(3,:,:));
    ADCP.velocityError = squeeze(velocityTrue(4,:,:));
```

```
        if useCompassForHeading
            ADCP.uMagnetic = squeeze(velocityInstrOrEarth(1,:,:)); % east velocity relative to magnetic North
            ADCP.vMagnetic = squeeze(velocityInstrOrEarth(2,:,:)); % north velocity relative to magnetic
north
            % rotate to true Earth coords, get magnetic declination at this location
            magdevRadians = -ADCP.magneticDeclination * pi/180; % magnetic declination in radians (negation
is to convert heading to yaw)
            M = [cos(magdevRadians) -sin(magdevRadians); sin(magdevRadians) cos(magdevRadians)]; % standard
2D rotation
            velocityTrue = M * velocityScreenedReshape(1:2,:);
            ADCP.processingComments(end+1,1) = {'Rotated Earth co-ordinate, device-supplied adcp.u/v to true
North from onboard magnetic North, by applying a magnetic declination correction (pitch/roll rotation
unchanged). '};
        else
            % rotate to true Earth coords: get the difference in yaw (transforming heading to yaw first),
apply standard 2D rotation
            % Verification method: grab a bin, calculate the EN vector direction, do the correction,
calculate vector direction, take the
            % difference and it is the same difference as between the heading and the compass. The original
method failed that test.
            velocityTrue = nan(2, n*p);
            for i = 1:p
                yawCorrection = -(heading(min(length(heading),i)) - ADCP.compassHeading(i));
                M = [cosd(yawCorrection) -sind(yawCorrection); sind(yawCorrection) cosd(yawCorrection)]; %
standard 2D rotation
                velocityTrue(:, (i-1)*n+1:i*n) = M * velocityScreenedReshape(1:2, (i-1)*n+1:i*n);
            end
            ADCP.processingComments(end+1,1) = {'Rotated Earth co-ordinate, device-supplied adcp.u/v to true
North from onboard magnetic North, using a fixed or variable external source true heading (pitch/roll
rotation unchanged). '};
        end
        velocityTrue = reshape(velocityTrue, [2 n p]);
        ADCP.u = squeeze(velocityTrue(1,:,:));
        ADCP.v = squeeze(velocityTrue(2,:,:));
```

```
%% get the original beam velocities
if strcmp(Config.coordSys, 'BEAM')
    ADCP.processingComments(end+1,1) =  {'Raw data matrix, data.velocity, is the beam radial velocities.'};
    beamMatrix = ADCP.velocity;
elseif strcmp(Config.coordSys, 'XYZ')
    ADCP.processingComments(end+1,1) = {'Raw data matrix, data.velocity, is the XYZ velocities (instrument
co-ordinates). Used the instrument transform matrix to revert to beam radial velocities. '};
    beamMatrix = T \ ADCP.velocity;
elseif strcmp(Config.coordSys, 'ENU')
    ADCP.processingComments(end+1,1) = {'Raw data matrix, data.velocity, is the ENU velocities (Earth co-
ordinates relative to onboard compass, pitch, roll sensors). Used the instrument transform matrix and raw
onboard sensor data to revert to beam velocities. '};
    beamMatrix = nan(3, n*p);
    for i = 1:p
        hThis = (ADCP.compassHeading(i) - 90) * pi/180;
        pThis = ADCP.pitch(i) * pi/180;
        rThis = ADCP.roll(i) * pi/180;
        H = [cos(hThis) sin(hThis) 0; -sin(hThis) cos(hThis) 0; 0 0 1];
        PR = [cos(pThis) -sin(pThis)*sin(rThis) -cos(rThis)*sin(pThis);...
            0             cos(rThis)            -sin(rThis);  ...
            sin(pThis) sin(rThis)*cos(pThis)  cos(pThis)*cos(rThis)];
        beamMatrix(:, (i-1)*n+1:1:i*n) = (H * PR * T) \ ADCP.velocity(:, (i-1)*n+1:i*n);
    end
    beamMatrix = reshape(beamMatrix, [3 p n]);
else
    error('ONC:nortekENUvelocities:unknownCoodSys', 'Unknown co-ord system');
end
```

```
velocityTrue = nan(3, n*p);
maxNumPosData = max([length(heading) length(pitch) length(roll)]);  % either 1 or p, the number of time
stamps
for i = 1:maxNumPosData
    % heading/pitch/roll matrices
    % this rotation is from Nortek: http://www.nortek-as.com/lib/forum-attachments/coordinate-transformation
/view
    % I considered various ways to get the index - pointers, use an eval, etc, but using 'min(length
(heading),i)' is actually faster
    hThis = (heading(min(length(heading),i)) - 90) * pi/180; % convert heading to yaw, the Nortek code
doesn't have a '-' here, but the sine term is negated below
    pThis = pitch(min(length(pitch),i)) * pi/180;
    rThis = roll(min(length(roll),i)) * pi/180;
    H = [cos(hThis) sin(hThis) 0; -sin(hThis) cos(hThis) 0; 0 0 1];
    PR = [cos(pThis) -sin(pThis)*sin(rThis) -cos(rThis)*sin(pThis);...
        0             cos(rThis)            -sin(rThis);  ...
        sin(pThis) sin(rThis)*cos(pThis)  cos(pThis)*cos(rThis)];
    % apply heading/pitch/roll rotations
    if maxNumPosData == 1
        velocityTrue = H * PR * T * beamReshape;
    else
        velocityTrue(:, (i-1)*n+1:1:i*n) = H * PR * T * beamReshape(:, (i-1)*n+1:i*n);
    end
end
% extract w values (u,v below) - use permute to ensure we don't lose singleton dimensions
velocityTrue = reshape(velocityTrue, [3 p n]);
ADCP.u = permute(velocityTrue(1,:,:), [3 2 1]);
ADCP.v = permute(velocityTrue(2,:,:), [3 2 1]);
ADCP.w = permute(velocityTrue(3,:,:), [3 2 1]);
```

Correction for Tilt: Bin-mapping

In addition to coordinate system rotations, the orientation of the instrument affects the vertical range on which the measurement bins are spaced (depth cell and measurement bin are synonymous). If the instrument is significantly tilted, the bins used to determine the velocities at each range will be a different water depths. Here is Figure 21 from Acoustic Doppler Current Profiler: Principles of Operation - A Practical Primer, notice in (B) that the highlighted depth cells are not the same as in (A):
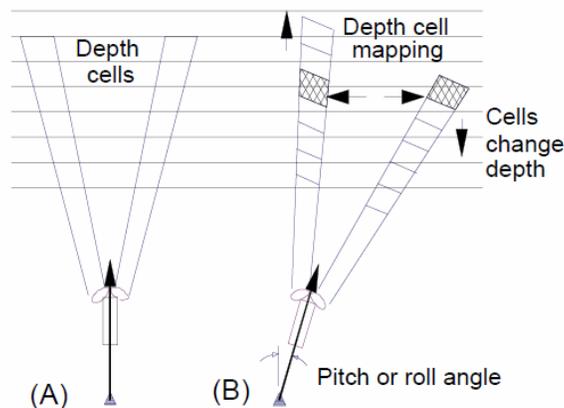
RDI ADCP data collected in Earth co-ordinates normally has had a depth cell mapping algorithm applied to align the depth cells to correct for tilt, while the other modes have not had it applied. Depth cell mapping is applied in post-processing in *winADCP*. Nortek's *Storm* / *Surge* post-processing software also applies bin-mapping when doing co-ordinate system rotations/transforms. (The term 'bin-mapping' is used by Nortek, RDI uses 'depth cell mapping'. We use 'bin-mapping' as it's the most common term in the literature.) Conversely, Nortek ADCP data collected in ENU co-ordinates has not had bin mapping applied. To apply bin-mapping, ENU data is converted back to Beam data, bin-mapping is applied and then converted back to ENU, doing any rotations as needed. The approach originated as described by Pulkkinen (1992). The Pulkkinen / RDI bin-mapping method is based on matching up the nearest vertical bin in each beam for each of the original range steps, then the radial velocities from those bins are used in the rotation to true Earth velocities. For compatibility with original VENUS ADCP data products, an alternative method is offered as a option as well, as described by Ott (2002). In that paper, the Ott method is shown to be an improvement over the discrete nearest vertical bin matching method. The Ott method uses a linear interpolation between the nearest vertical bins for each beam for each of the original range steps. The Ott method also interpolates over a small number of missing bins, otherwise missing data nullifies a greater extent of the data than it does in the nearest vertical bin method. The inner and outer most bins can end up with NaN values when there are less than 4 bins at the same vertical level, see the above graphic. This affects the nearest vertical bin method more so than the Ott method, and more bins are affected with increasing tilts.

Users are offered the option of 'None', 'Nearest vertical bin', or 'Linear interpolation (Ott method)'. In either method, any data where the absolute pitch or roll is greater than or equal to 20 degrees, bin-mapping will not be applied and the data with be returned NaN. If the absolute values of all pitch or all roll data are greater than or equal to 20 degrees, the bin-mapping option will be overridden in the data products to 'None' (warnings will also be issued to the search status and internal logs). When the data is in Earth co-ordinates, bin-mapping cannot be applied/modified. For RDI Earth data, the nearest vertical bin method has normally been applied, while Nortek Earth data does not have bin-mapping applied. In both cases, the users' preference may be overridden.

Bin-mapping is not applied to the backscatter and intensity data. Instead, we offer a vertically corrected range in the mat file products and use this range for the plots when plotting against depth. For example, if the bin spacing is 1 m, but the tilt is 10 degrees, the vertical spacing is 0.98 m.

For advanced, interested users, here is the core code that does bin-mapping:

```
    %% apply nearest vertical bin-mapping (RDI method)
    if isOrientUp
        ZSG = [+1 -1 +1 -1];       % ADCP up/convex
    else
        ZSG = [+1 -1 -1 +1];       % ADCP down/convex
    end
    beamElAbsRad = abs(beamElevation) * pi/180; % calcs below, from beam2uvw.m, take the abs() and convert
to rad, so do it here for speed
    beamElRad = beamElevation * pi/180;
    beamNearestVert = nan(size(beam));
    maxNumPosData = max([length(pitch) length(roll)]);  % either 1 or p, the number of time stamps - gotta
be the same length, error if not!
    for k = 1:maxNumPosData    % loop over time stamps and pitch/roll values
        % NaN any pings that don't have good pitch/roll
        if any([pitchOverLim(k) rollOverLim(k)])
            doMaxPitchRollWarning = true;
            continue
        end
        if any([isnan(pitch(k)) isnan(roll(k))])
            doNaNPitchRollWarning = true;
            continue
        end

        % use the bin-mapping math from beam2uvw.m (not sure what ZSG and SC are supposed to be named for,
but keep them for historical reasons)
        SC  = sin(beamElAbsRad)*cos(pitch(k))*cos(roll(k)) + ...
            cos(beamElRad) .*  ZSG .* [-sin(roll(k)) -sin(roll(k)) sin(pitch(k))*cos(roll(k)) sin(pitch(k))
*cos(roll(k))];
        SC  = abs(sin(beamElAbsRad) ./ SC);  % the abs wasn't in beam2uvw.m but even with nominal conditions
it needs to be there
        vertBinIndex = (round(binIndex.' * SC)).';
        indexOutBoundsFlag = vertBinIndex < 1 | vertBinIndex > n;
        vertBinIndex(indexOutBoundsFlag) = 1;
        for i = 1:m
            beamNearestVert(i, :, k) = beam(i, vertBinIndex(i,:), k);
            beamNearestVert(i, indexOutBoundsFlag(i), k) = NaN;
        end
    end
    beam = beamNearestVert;
```

```
    %% do the Ott method bin mapping, including Ott's filling/smoothing method
    binSpacing = mean(diff(range));
    minRangeExtrap = min(range) - binSpacing;
    maxRangeExtrap = max(range) + binSpacing;

    for i = 1:m  % loop over beams
        % do this calc here since it doesn't change in time
        matrixBeamElevationAzimuth = [-cosd(beamElevation(i))*sind(beamAzimuth(i)); cosd(beamElevation(i))
*cosd(beamAzimuth(i)); -sind(beamElevation(i))] ./ sind(abs(beamElevation(i)));
        for k = 1:p  % loop over time stamps
            iP = min(length(pitch),k);
            iR = min(length(roll),k);
            % NaN any pings that don't have good pitch/roll
            pThis = pitch(iP);
            rThis = roll(iR);
            if any([pitchOverLim(iP) rollOverLim(iR)])
                beam(i,:,k) = NaN;
```

```
                    doMaxPitchRollWarning = true;
                    continue
                end
            if any([isnan(pThis) isnan(rThis)])
                    beam(i,:,k) = NaN;
                    doNaNPitchRollWarning = true;
                    continue
                end
            % skip any pings that are all NaN
            thisBeamTime = squeeze(beam(i,:,k));
            iNaNthisBeamTime = isnan(thisBeamTime);
            numNaNthisBeamTime = sum(iNaNthisBeamTime);
            if numNaNthisBeamTime >= min(n * maxFracNaN, n - 1) % skip this beam time if too much data is
NaN (need at least 2 valid points)
                    beam(i,:,k) = NaN;
                    continue
                end
            % for this time and beam, fill the NaNs with linear interpolation, except where there are too
many NaNs in a row
            % this won't fill on outside of valid data - linear interpolation doesn't extrapolate by default
            if any(iNaNthisBeamTime)
                % find consecutive NaNs for this beam and time, only if there are enough of them to flag
                iTooManyConsecNaNs = false(1, n);
                if numNaNthisBeamTime >= maxConsecutiveNaNs
                    numConsec = 0;
                    for j = find(iNaNthisBeamTime, 1, 'first') : find(iNaNthisBeamTime, 1, 'last')   % loop
over depth bins
                        if iNaNthisBeamTime(j)
                            numConsec = numConsec + 1;
                            if numConsec > maxConsecutiveNaNs
                                iTooManyConsecNaNs((j-numConsec+1):j) = true;
                            end
                        else
                            numConsec = 0;
                        end
                    end
                end
                iNaNsInterp = iNaNthisBeamTime & ~iTooManyConsecNaNs;
                thisBeamTime(iNaNsInterp) = interp1(binIndex(~iNaNthisBeamTime), thisBeamTime
(~iNaNthisBeamTime), binIndex(iNaNsInterp));
            end
            % this is simplicaton of the commented code below
            Mp = [1 0 0;0 cos(pThis) -sin(pThis);0 sin(pThis) cos(pThis)];
            Mr = [cos(rThis) 0 sin(rThis);0 1 0;-sin(rThis) 0 cos(rThis)];
            beamXYZ = Mr * Mp * matrixBeamElevationAzimuth;
            actualVerticalRange = range * abs(beamXYZ(3));

%           % range calculation code from beam2uvw.m
%           Mp = [1 0 0;0 cos(pThis) -sin(pThis);0 sin(pThis) cos(pThis)];
%           Mr = [cos(rThis) 0 sin(rThis);0 1 0;-sin(rThis) 0 cos(rThis)];
%           beamXYZ = Mr * Mp * [-cosd(beamElevation(i))*sind(beamAzimuth(i)); cosd(beamElevation(i))*cosd
(beamAzimuth(i)); -sind(beamElevation(i))];
%           actualVerticalRange = range * abs(beamXYZ(3) ./ sind(abs(beamElevation(i))));
            % map the beam bins to the vertical range - allow some extrapolation - won't extrap through NaNs
though (good).
            % Extrapolation limited to +/- one bin spacing.
            beam(i,:,k) = interp1( actualVerticalRange, thisBeamTime, range, 'linear', 'extrap' );
            beam(i, actualVerticalRange > maxRangeExtrap | actualVerticalRange < minRangeExtrap, k) = NaN;
        end
    end
```

Three-Beam Solutions and Screening (For RDI ADCPs Only)

In systems with four beams, the transformation from radial velocities to Earth co-ordinates velocities has redundant information; only three beams are needed to resolve currents in the three orthogonal directions, while the 4th beam effectively provides error estimation. In the case where exactly one bin out of of the four at any level is flagged and NaN'ed, then currents can still be calculated from the remaining three radial velocities, using the three-beam transformation solution. The screening steps that can flag the data 'NaN' prior to transformation include fish detection and correlation threshold, as well as any bin-mapping (particularly the RDI nearest vertical bin which can set many bins to NaN). The three-beam solution works by assigning the error velocity to be zero and then solving the transformation for the missing bin (for more details see Three Beam Solutions in adcp coordinate transformation_Jan08.pdf) . With the missing data filled in, the normal transform from Beam to Instrument co-ordinate data is applied.

In cases where the data quality team has determined that an entire beam is bad, they can supply a device attribute specifying the bad beam number. That beam is then excluded in the processing from beam or instr co-ords and the values are filled in with the three-beam solution. Only the U,V,W results are affected, the raw velocity data as recorded by the instrument is presented without the exclusion. This forces the 3 beam option to on, regardless of the user input. The excluded beam number is mentioned in a comment on both the currents plot and intensity plots and in the processing comments in the mat files. This is currently only available for RDI ADCPs with 4 beams.

Once in Earth co-ordinates, the data can also be screened by the error velocity. Error velocity screening only applies to data with four valid beams (and non-zero error velocities). For further infromation on the screening steps, including algorithms, see adcp coordinate transformation_Jan08.pdf

In general, all processing steps and the default parameters used replicate the behaviour of winADCP. We test this as part of our regular software deployment process. If users choose any non-default option, the results will not match winADCPs. The Ott bin-mapping option is not available on winADCP at all.

Nortek ADCPs are generally three beam and screening is not applied in the data products, except for a fixed 50% correlation screen on Nortek Signature55 data only, applied to match Nortek's Signature Viewer software.

Here is the code for three-beam solutions and transform from Beam to Instrument (XYZ) co-ordinate data:

```
    %% apply the three beam solution
    % Note, for instr data, this may be a bit redundant unless some data has been screened or bin-mapped out
    if Config.coord_EX(4) == '1'  % only 3-beam if the EX parameter is set, this replicates winADCP's
behaviour
        isnanBM = isnan(beamMatrix);
        is3beam = squeeze(sum(isnanBM, 1)) == 1;  % index of all bins over time that have 1 NaN only (3 beam)
        transformMatrixRow4 = transformMatrix(4,:).';
        for j = find(any(is3beam,2)).'   % loop over the bin indices that have at least one 3 beam instance
            for k = find(is3beam(j,:))   % loop over the times when the current bin has a 3 beam instance
                isnanThis = isnan(beamMatrix(:,j,k));
                temp = beamMatrix(:,j,k) .* transformMatrixRow4;
                beamMatrix(isnanThis,j,k) = -sum(temp(~isnanThis)) / transformMatrix(4, isnanThis);
            end
        end
        ADCP.processingComments(end+1,1) = {['Three beam solution applied to ' num2str(sum(is3beam(:))) '
individual bins (of ' num2str(numel(beamMatrix)) ' total). ']};
    else
        ADCP.processingComments(end+1,1) = {'Three beam solution not applied as the EX command was set to
xxx0x. (This matches the behaviour of winADCP which responds to EX(4). We can override this on request,
contact us. As a side note, winADCP does not respond to EX(5), the command for bin-mapping; winADCP always
does bin-mapping on beam co-ord data.'};
    end

    %% convert bin-mapped, screened and 3-beamed data to XYZ
    vTemp = reshape(beamMatrix, 4, n * p);
    velocityInstrOrEarth = reshape(transformMatrix * vTemp, [m n p]);
```

Overall Processing Flow:

From Instrument or Beam data:

- If Instrument co-ordinate data, convert from Instrument to Beam co-ordinate data
- Exclude bad beams (RDI only, specified by device attribute, forces three-beam solutions to on)
- Apply correlation screening (RDI only, except for a fixed 50% screen on Nortek Signature55 data, applied to match Nortek's Signature Viewer software, option available)
- Apply fish detection screening (RDI only, if requested, tuneable)
- Apply bin-mapping (RDI default is 'nearest vertical bin', Nortek default is none, option available)
- Apply three-beam solutions (RDI only, if requested, option available)
- Transform from Beam co-ordinate data to to Instrument co-ordinate data
- Rotate Instrument data to true Earth (via a combination of fixed heading/pitch/roll, magnetic declination corrected compass heading and onboard pitch/roll data)
- Apply error velocity screening (RDI only, option available)
- Do ensemble averaging if requested (option available)

From Earth data:

- Do nothing if **RDI** source file has been rotated to true Earth (via a combination of fixed heading/pitch/roll, magnetic declination corrected compass heading and onboard pitch/roll data),
  otherwise rotate to from magnetic Earth to True Earth (via a combination of fixed heading/pitch/roll, magnetic declination corrected compass heading and onboard pitch/roll data)
- Apply error velocity screening if the requested threshold is more stringent than the screening applied onboard the device (option available)
- Do ensemble averaging if requested (option available)

The steps above with *option available* are tuneable by the user as detailed in the data product options below. The *ADCP.processingComments* field in both Nortek and RDI **MAT** file data products provides the user with a log of the decisions made in this processing flow. The MAT files (RDI only) also has a structure called *ADCP.processingOptions* that details what options the user choose and what options were actually applied, as some options, such as the error velocity screening threshold are not always applicable. See the data product options and MAT file format sections below for more information. This is complicated, contact us for help.

Data Verification

As noted earlier, the ultimate requirement for the ADCP data products is that they replicate the results of the manufacturer's software. We test against the manufacturers' software to verify the data at every software release (regression testing). The testing suite includes manual and automated testing where the output of the manufacturer's software is compared to the data products and where data products are compared to the captured files from the previous release. The test suite is regularly updated for any new scenarios. Improvements in the test procedure are also made as needed. In addition to testing the software, there has a been a significant effort to insure the metadata, heading in particular, is correct. These steps include careful measurement and documentation at deployment time, verifying and vetting ROV heading sensors, comparing data to nearby ADCPs, comparing results between various processing methods, and perhaps the best check is to compare the currents with the known/modelled tides. Contact us for information on the testing procedures and metadata verification.

A recent simplification in testing came about because of an improvement in the generation of **RDI** file data products, which are we archive to speed up generation of subsequent products, **MAT**/**NC** files and plots. As of November 2015, **RDI** files with Beam or Instr co-ordinate data will incorporate the same heading/pitch/roll values as used for rotation and bin-mapping, that way, users and testers may load and process these files with RDI software, matching the values they can obtain from **MAT** and **NC** file products. For Earth co-ordinate data in **RDI** files, if we update the heading, we also need to rotate the East (u) and North (v) velocities as the new heading will be different from the internal compass heading (either fixed value, mobile position sensor or a magnetic declination is applied) on which the onboard Earth co-ordinate rotation was done. In this case, we don't update the pitch/roll as we cannot update the bin-mapping done for Earth data. Fortunately, the pitch/roll data is generally good and we do not yet have any examples where we do not use the internal pitch/roll data. Once the **RDI** files produced prior to November 2015 are re-generated, all **RDI** files will be made available in Data Search.

In all, the results from RDI winADCP, and Nortek software will match the processed data products. In addition, the original data, particularly the Beam co-ordinate data is always available in the **RDI** and Nortek raw binary files and as *data.velocity* (Nortek) and *adcp.velocity* (RDI) in the **MAT** files. The *processingComments* will completely describe the steps applied to produce the *u,v,w* processed velocities. In addition to the code snippets above, we are happy to supply more code, collaborate on that code, etc. We are interested in collaborating on QAQC processes, data verification, processing, analysis and research. Contact us if you have any questions.