Encoding NetCDF Radial Data in the HF-Radar Network

Mark Otero
Coastal Observing Research and Development Center
Scripps Institution of Oceanography
2013/08/26

In order to promote data distribution and interoperability, sea surface velocities measured in near real-time by HF-Radar are being prepared by the HF-Radar Network (HFRNet) in a portable and self-describing format known as Network Common Data Form (NetCDF). NetCDF provides software libraries and conventions that enable creation, access, and sharing of scientific data. A full description of NetCDF, including documentation, conventions and software is available from Unidata. This document describes the NetCDF encoding and conventions used for radar derived sea surface velocities also commonly referred to as radial data. Radial data from individual radar stations are combined to produce what are known as Real-Time Velocities (RTVs) which are already available from HFRNet in NetCDF format. Distribution of radial data will provide access to lower level data for a variety of applications ranging from quality control to model assimilation.

There are currently three types of radars for measuring surface ocean velocity that contribute data to HFRNet; Coastal Ocean Dynamics Application Radar's (CODAR) SeaSonde, Helzel Messtechnik's Wellen Radar (WERA) and the University of Hawaii's Least Expensive Radar (LERA). While all the radars share the same principals of operation, differences in signal transmission, reception and processing yield variations in metadata, quality control metrics and spatial registration. Consequently, NetCDF encoding for all three radars shares a common core of metadata and variables which describe the measured velocity field while customizations have been made to accommodate each radar's unique attributes and data fields (see Appendices A-C for encoding details).

Two metadata conventions are used for organizing and describing radial velocities encoded in NetCDF. The Climate and Forecasting (CF) metadata convention provides the majority of the framework for describing the coordinate system, data and associated metadata through standardized attributes, units and variable names. Of particular importance in this case, the coordinates attribute provides a mechanism for mapping CODAR and LERA data from their native polar coordinate system to latitude and longitude. The Attribute Convention for Dataset Discovery (ACDD) has been adopted to support crosswalks among metadata schemes and, ultimately, improve data discovery through applications such as the Thematic Real-time Environmental Distributed Data Services (THREDDS). Additional global attributes are also included directly from the source file metadata and from the HF-Radar Network for completeness.

Future revisions of NetCDF radial datasets may include a vertical dimension with appropriate qualifiers indicating a nominal depth for the measured velocity field.

Refinements may also be needed in order to optimize aggregation with THREDDS data servers. With some guidance on how keywords should be formatted in the ACDD, AGU index terms will be added including OCEANOGRAPHY: GENERAL > Ocean observing systems, OCEANOGRAPHY: GENERAL > Remote sensing and electromagnetic processes, OCEANOGRAPHY: PHYSICAL > Currents and RADIO SCIENCE > Radio oceanography.

This work has been developed at the Coastal Observing Research and Development Center at Scripps Institution of Oceanography with funding from the Integrated Ocean Observing System. Please direct questions and comments regarding this document to Mark Otero.

Appendix A
CODAR NetCDF Encoding

CODAR SeaSonde NetCDF header in common data language (CDL)

```
netcdf RDL_i_ODU_LISL_2013_03_20_0000 {
dimensions:
        time = UNLIMITED ; // (1 currently)
        bearing = 73 ;
        range = 33 ;
variables:
        int time(time) ;
                time:standard_name = "time" ;
                time:units = "seconds since 1970-01-01" ;
                time:calendar = "gregorian" ;
        float bearing(bearing) ;
                bearing:axis = "Y" ;
                bearing:long_name = "bearing_away_from_instrument" ;
                bearing:units = "degrees_true" ;
        float range(range) ;
                range:axis = "X" ;
                range:long_name = "range_away_from_instrument" ;
                range:units = "km" ;
        float lat(bearing, range) ;
                lat:standard_name = "latitude" ;
                lat:units = "degrees_north" ;
        float lon(bearing, range) ;
                lon:standard_name = "longitude" ;
                lon:units = "degrees_east" ;
        float speed(time, bearing, range) ;
                speed:valid_range = -1000.f, 1000.f ;
                speed:standard_name = "radial_sea_water_velocity_away_from_instrument" ;
                speed:units = "cm s-1" ;
                speed:coordinates = "lon lat" ;
        short direction(time, bearing, range) ;
                direction:valid_range = 0s, 3600s ;
                direction:standard_name = "direction_of_radial_vector_away_from_instrument" ;
                direction:units = "degrees_true" ;
                direction:coordinates = "lon lat" ;
```

```
            direction:scale_factor = 0.1f ;
        float u(time, bearing, range) ;
            u:valid_range = -1000.f, 1000.f ;
            u:standard_name = "surface_eastward_sea_water_velocity" ;
            u:units = "cm s-1" ;
            u:coordinates = "lon lat" ;
        float v(time, bearing, range) ;
            v:valid_range = -1000.f, 1000.f ;
            v:standard_name = "surface_northward_sea_water_velocity" ;
            v:units = "cm s-1" ;
            v:coordinates = "lon lat" ;
        short vflg(time, bearing, range) ;
            vflg:long_name = "vector_flag_masks" ;
            vflg:valid_range = 0s, 2048s ;
            vflg:flag_masks = 1s, 2s, 4s, 8s, 16s, 32s, 64s, 128s, 256s, 512s, 1024s ;
            vflg:flag_meanings = "grid_point_deleted grid_point_near_coast point_measurement
no_radial_solution baseline_interpolation exceeds_max_speed invalid_solution
solution_beyond_valid_spatial_domain insufficient_angular_resolution reserved reserved" ;
            vflg:coordinates = "lon lat" ;
        float espc(time, bearing, range) ;
            espc:long_name = "radial_sea_water_velocity_spatial_quality" ;
            espc:units = "cm s-1" ;
            espc:coordinates = "lon lat" ;
        float etmp(time, bearing, range) ;
            etmp:long_name = "radial_sea_water_velocity_temporal_quality" ;
            etmp:units = "cm s-1" ;
            etmp:coordinates = "lon lat" ;
        float maxv(time, bearing, range) ;
            maxv:long_name = "radial_sea_water_velocity_away_from_instrument_maximum" ;
            maxv:units = "cm s-1" ;
            maxv:coordinates = "lon lat" ;
        float minv(time, bearing, range) ;
            minv:long_name = "radial_sea_water_velocity_away_from_instrument_minimum" ;
            minv:units = "cm s-1" ;
            minv:coordinates = "lon lat" ;
        byte ersc(time, bearing, range) ;
            ersc:long_name = "radial_sea_water_velocity_spatial_quality_count" ;
            ersc:coordinates = "lon lat" ;
        byte ertc(time, bearing, range) ;
            ertc:long_name = "radial_sea_water_velocity_temporal_quality_count" ;
            ertc:coordinates = "lon lat" ;
        float xdst(bearing, range) ;
            xdst:long_name = "eastward_distance_from_instrument" ;
            xdst:units = "km" ;
            xdst:coordinates = "lon lat" ;
        float ydst(bearing, range) ;
            ydst:long_name = "northward_distance_from_instrument" ;
            ydst:units = "km" ;
            ydst:coordinates = "lon lat" ;
        byte sprc(time, bearing, range) ;
            sprc:long_name = "radial_sea_water_velocity_cross_spectal_range_cell" ;
            sprc:coordinates = "lon lat" ;

// global attributes:
            :netcdf_library_version = "4.1.3" ;
            :Conventions = "CF-1.6" ;
```

```
:title = "Near-Real Time Surface Ocean Radial Velocity" ;
:institution = "SIO" ;
:source = "Surface Ocean HF-Radar" ;
:history = "16-Aug-2013 16:32:09: NetCDF file created" ;
:references = "CODAR SeaSonde LonLatUV (LLUV) File Format" ;
:creator_name = "HF-Radar Network Administrators" ;
:creator_email = "hfrnetadm@ucsd.edu" ;
:creator_url = "http://cordc.ucsd.edu/projects/mapping/" ;
:summary = "HF-RADAR measurements of ocean velocity are radial in\n",
      "direction relative to the radar location and representitive \n",
      "of the upper 0.3 - 2.5 meters of the ocean." ;
:geospatial_lat_min = 34.96302f ;
:geospatial_lat_max = 38.42041f ;
:geospatial_lon_min = -78.07812f ;
:geospatial_lon_max = -73.76715f ;
:CTF = "1.00" ;
:FileType = "LLUV rdls RadialMap" ;
:LLUVSpec = "1.17  2011 06 20" ;
:UUID = "6BDBE8F4-E7CD-4ECE-B92C-74BF64148941" ;
:manufacturer = "CODAR Ocean Sensors. SeaSonde" ;
:Network = "ODU" ;
:Site = "LISL" ;
:TimeZone = "UTC +0.000 0 GMT" ;
:TimeCoverage = "180.000 Minutes" ;
:Origin = "36.6917167  -75.9226333" ;
:GreatCircle = "WGS84 6378137.000  298.257223562997" ;
:GeodVersion = "CGEO 1.57  2009 03 10" ;
:LLUVTrustData = "all" ;
:RangeStart = "1" ;
:RangeEnd = "33" ;
:RangeResolutionKMeters = "5.824900" ;
:AntennaBearing = "130.0 True" ;
:ReferenceBearing = "0 True" ;
:AngularResolution = "5 Deg" ;
:SpatialResolution = "5 Deg" ;
:PatternType = "Ideal" ;
:PatternDate = "2009 06 04  15 13 32" ;
:PatternResolution = "1.0 deg" ;
:TransmitCenterFreqMHz = "4.575000" ;
:DopplerResolutionHzPerBin = "0.000488281" ;
:FirstOrderMethod = "0" ;
:BraggSmoothingPoints = "2" ;
:CurrentVelocityLimit = "250.0" ;
:BraggHasSecondOrder = "0" ;
:RadialBraggPeakDropOff = "151.360" ;
:RadialBraggPeakNull = "10.000" ;
:RadialBraggNoiseThreshold = "5.000" ;
:PatternAmplitudeCorrections = "0.9981  0.9981" ;
:PatternPhaseCorrections = "74.00  67.00" ;
:PatternAmplitudeCalculations = "0.0517  0.0641" ;
:PatternPhaseCalculations = "64.50  55.60" ;
:RadialMusicParameters = "40.000 20.000 2.000" ;
:MergedCount = "5" ;
:RadialMinimumMergePoints = "2" ;
:FirstOrderCalc = "1" ;
:MergeMethod = "1 MedianVectors" ;
```

```
        :PatternMethod = "1 PatternVectors" ;
        :TransmitSweepRateHz = "1.000000" ;
        :TransmitBandwidthKHz = "-25.733913" ;
        :SpectraRangeCells = "63" ;
        :SpectraDopplerCells = "2048" ;
        :TableType = "LLUV RDL9" ;
        :TableColumns = "18" ;
        :TableColumnTypes = "LOND LATD VELU VELV VFLG ESPC ETMP MAXV MINV\n",
            "ERSC ERTC XDST YDST RNGE BEAR VELO HEAD SPRC" ;
        :TableRows = "1260" ;
}
```

Sample MATLAB code for encoding CODAR data in NetCDF from a source ASCII file

```
% Example script for formatting CODAR HF-Radar radial files in netCDF.

%% Load sample data
rfile_path = '/home/motero/Documents/Projects/HFRNet/Development/NetCDF Radial/Sample Files/';
rfile = fullfile( rfile_path, 'RDL_i_ODU_LISL_2013_03_20_0000.hfrss10lluv');
r = load( rfile );
R = struct( ...
   'lond', r(:, 1), ...
   'latd', r(:, 2), ...
   'velu', r(:, 3), ...
   'velv', r(:, 4), ...
   'vflg', r(:, 5), ...
   'espc', r(:, 6), ...
   'etmp', r(:, 7), ...
   'maxv', r(:, 8), ...
   'minv', r(:, 9), ...
   'ersc', r(:,10), ...
   'ertc', r(:,11), ...
   'xdst', r(:,12), ...
   'ydst', r(:,13), ...
   'rnge', r(:,14), ...
   'bear', r(:,15), ...
   'velo', r(:,16), .    'head', r(:,17), ...
   'sprc', r(:,18)  ...
);
clear r;

%TimeStamp: 2013 03 20  00 00 00
R.time = datenum(2013,3,20,0,0,0);

%% Generate grid domain
% May want to have clipped to samples w/in a given file or possibly extend
% the full 0-360 range and some nominal range for the given frequency.  The
% latter would work better for aggregation (assuming bearing angles don't
% change over the aggregation period).

%RangeResolutionKMeters: 5.824900
range_dim = min( R.rnge ) : 5.824900 : max( R.rnge );

%AngularResolution: 5 Deg
```

```
bearing_dim = 0 : 5 : 360;

[ bearing, range ] = meshgrid( bearing_dim, range_dim );

% check - see that our grid captures the sample space right
% figure
% plot( range, bearing, 'go' )
% hold on
% plot( R.rnge, R.bear, 'b.')


% Generate lat/lon from bearing & range
% This is the most accurate method of defining lat/lon values for given
% range & bearing.  Other methods of interpolating and extrapolating using
% griddata or otherwise from range, bearing, lon and lat are inconsistent
% and generally not reliable from file to file. The reason we need to do
% this at all is because there cannot be missing values in auxillary
% coordinate variables.

%Origin:  36.6917167  -75.9226333
[latd, lond] = reckon( 36.6917167, -75.9226333, km2deg(range), bearing );

% Check
% figure
% plot(lond, latd, 'ko')
% hold on
% plot(R.lond, R.latd, 'r.')
% daspect([1.2 1 1])


%% Determine the mapping index from sample data to full grid
range_map_idx   = nan( 1, numel(R.rnge) );
bearing_map_idx = nan( 1, numel(R.bear) );
for I = 1 : numel(R.rnge)
   [~, range_map_idx(I)]   = min( abs( range_dim - R.rnge(I) ) );
   [~, bearing_map_idx(I)] = min( abs( bearing_dim - R.bear(I) ) );
end
clear I;
rb_map_idx = sub2ind( size( range ), range_map_idx, bearing_map_idx );


%% Populate gridded matrices

% Eastward Velocity
velu = nan( size( range ) );
velu(rb_map_idx) = R.velu;

% Northward Velocity
velv = nan( size( range ) );
velv(rb_map_idx) = R.velv;

% check
% figure
% quiver(R.rnge, R.bear, R.velu, R.velv, 0, 'k')
% hold on
% quiver(range, bearing, velu, velv, 0, 'r')
```

```
%
% figure
% quiver(R.lond, R.latd, R.velu./100, R.velv./100, 0, 'k')
% daspect([1.2 1 1])
% hold on
% quiver(lond, latd, velu./100, velv./100, 0, 'r')

% Current Speed - flip sign so positive velocity is away from radar
% according to CF standard name
% 'radial_sea_water_velocity_away_from_instrument'. CODAR reports positive
% speeds as toward the radar.
velo = nan( size( range ) );
velo(rb_map_idx) = -R.velo;

% Current Heading
head = nan( size( range ) );
head(rb_map_idx) = R.head;

% Vector Flag
vflg = nan( size( range ) );
vflg(rb_map_idx) = R.vflg;

% Spatial Quality
espc = nan( size( range ) );
espc(rb_map_idx) = R.espc;
espc( espc==999 ) = NaN; % native bad-value

% Temporal Quality
etmp = nan( size( range ) );
etmp(rb_map_idx) = R.etmp;
etmp( etmp==999 ) = NaN; % native bad-value

% Velocity Maximum - flip sign so positive velocity is away from radar
% according to CF standard name
% 'radial_sea_water_velocity_away_from_instrument'. CODAR reports positive
% speeds as toward the radar.
maxv = nan( size( range ) );
maxv(rb_map_idx) = -R.maxv;
maxv( maxv==-999 ) = NaN; % native bad-value

% Velocity Minimum - flip sign so positive velocity is away from radar
% according to CF standard name
% 'radial_sea_water_velocity_away_from_instrument'. CODAR reports positive
% speeds as toward the radar.
minv = nan( size( range ) );
minv(rb_map_idx) = -R.minv;
minv( minv==-999 ) = NaN; % native bad-value

% Spatial Count
ersc = nan( size( range ) );
ersc(rb_map_idx) = R.ersc;
ersc( ersc==999 ) = NaN; % native bad-value

% Temporal Count
ertc = nan( size( range ) );
ertc(rb_map_idx) = R.ertc;
```

```
ertc( ertc==999 ) = NaN; % native bad-value

% X-Distance
xdst = nan( size( range ) );
xdst(rb_map_idx) = R.xdst;

% Y-Distance
ydst = nan( size( range ) );
ydst(rb_map_idx) = R.ydst;

% Spectra Range Cell
sprc = nan( size( range ) );
sprc(rb_map_idx) = R.sprc;


%% Scaling & Fill Values
% Bearing and heading are only reported to 10ths and can be
% reported as short unsigned integers when scaled. However, Bearing is a
% coordinate variable and cannot be scaled by CF metadata convention.
% Also, can only use unsigned when using NetCDF-4 enhansed data model,
% otherwise w/classic data model you're limited ot signed data types.
head = round( head.*10 );

% Type byte fill values
ersc( isnan(ersc) ) = netcdf.getConstant('NC_FILL_BYTE');
ertc( isnan(ertc) ) = netcdf.getConstant('NC_FILL_BYTE');
sprc( isnan(sprc) ) = netcdf.getConstant('NC_FILL_BYTE');

% Type short fill values
head( isnan(head) ) = netcdf.getConstant('NC_FILL_SHORT');
vflg( isnan(vflg) ) = netcdf.getConstant('NC_FILL_SHORT');

% Type float fill values
velo( isnan(velo) ) = netcdf.getConstant('NC_FILL_FLOAT');
velu( isnan(velu) ) = netcdf.getConstant('NC_FILL_FLOAT');
velv( isnan(velv) ) = netcdf.getConstant('NC_FILL_FLOAT');
espc( isnan(espc) ) = netcdf.getConstant('NC_FILL_FLOAT');
etmp( isnan(etmp) ) = netcdf.getConstant('NC_FILL_FLOAT');
maxv( isnan(maxv) ) = netcdf.getConstant('NC_FILL_FLOAT');
minv( isnan(minv) ) = netcdf.getConstant('NC_FILL_FLOAT');
xdst( isnan(xdst) ) = netcdf.getConstant('NC_FILL_FLOAT');
ydst( isnan(ydst) ) = netcdf.getConstant('NC_FILL_FLOAT');



%% Define dimensions, variables and attributes
ncfile = strrep( rfile, 'hfrss10lluv', 'nc' );

% Keep the classic data model to increase compatibility since we
% aren't using features specific to the NetCDF-4.0 model.
mode = netcdf.getConstant( 'NETCDF4' );
mode = bitor(mode, netcdf.getConstant( 'CLASSIC_MODEL' ) );

ncid = netcdf.create( ncfile, mode );
hist_create = [datestr(now) ': NetCDF file created'];
```

```
% Add dimensions (in order of T, Z, Y, X for CF/COARDS compliance)
dimid_t       = netcdf.defDim( ncid, 'time',    netcdf.getConstant('unlimited') );
dimid_bearing = netcdf.defDim( ncid, 'bearing', numel( bearing_dim ) );
dimid_range   = netcdf.defDim( ncid, 'range',   numel( range_dim ) );

%dimid_lat        = netcdf.defDim( ncid, 'lat', length(LatU) );
%dimid_lon        = netcdf.defDim( ncid, 'lon', length(LonU) );

% Add coordinate variables and attributes in the same order as
% the dimensions were written

% For time coordinate variable, when using the standard or gregorian
% calander, units should be in seconds and shifted forward beyond
% 1582/10/15 to avoid Julian -> Gregorian crossover and potential
% problems with the udunits package, if used.  Year length should be
% exactly 365.2425 days long for the Gregorian/standard and proleptic
% Gregorian calander.
%
% MATLAB's datenum uses the proleptic gregorian calander as defined
% by the CF standard.  When using standard_name attribute for time
% coordinate be sure to include the calander and units attributes.
%
% Use int for time data type because need 8 significant figures to
% represent number of seconds in 1 year (31556952 seconds), float
% isn't enough (7 sig. figures).  Would need to use double beyond
% int.  Besides, don't need sub-second accuracy.  Unix time will
% overflow int in 2038.  New epoch will be needed then to keep int
% representation.  Unix time epoch is:
%          1970-01-01 00:00:00Z = 0 seconds, 86,400 sec/day
varid_t = netcdf.defVar( ncid, 'time', 'int', dimid_t );
netcdf.putAtt( ncid, varid_t, 'standard_name', 'time' );
netcdf.putAtt( ncid, varid_t, 'units', 'seconds since 1970-01-01' );
netcdf.putAtt( ncid, varid_t, 'calendar', 'gregorian' );

% Bearing (arbitrary 'y' dimension)
varid_bearing = netcdf.defVar( ncid, 'bearing', 'float', dimid_bearing );
netcdf.putAtt( ncid, varid_bearing, 'axis', 'Y' );
netcdf.putAtt( ncid, varid_bearing, 'long_name', 'bearing_away_from_instrument' );
netcdf.putAtt( ncid, varid_bearing, 'units', 'degrees_true' );

% Range (arbitrary 'x' dimension)
varid_range = netcdf.defVar( ncid, 'range', 'float', dimid_range );
netcdf.putAtt( ncid, varid_range, 'axis', 'X' );
netcdf.putAtt( ncid, varid_range, 'long_name', 'range_away_from_instrument' );
netcdf.putAtt( ncid, varid_range, 'units', 'km' );


% Add global attributes

% Currently using Climate & Forecast Metadata Conventions v1.6
% (CF).  Am also using Attribute Convention for Dataset Discovery
% v1.1 (ACDD) where applicable along with
% a few of my own attributes (X) for things that didn't fit
% well or weren't clearly applicable to features in either
% convention.  The ACDD convention is still developing and may
% eventually support the attributes added.
```

```
varid_global = netcdf.getConstant('global');

% (X) NetCDF library version used
netcdf.putAtt( ncid, varid_global, 'netcdf_library_version', netcdf.inqLibVers );

% (CF)  Conventions
netcdf.putAtt( ncid, varid_global, 'Conventions', 'CF-1.6' );
%netcdf.putAtt( ncid, varid_global, 'Conventions', 'CF-1.5' );

% (CF, ACDD) title
netcdf.putAtt( ncid, varid_global, 'title', 'Near-Real Time Surface Ocean Radial Velocity' );

% (CF, ACDD) institution
netcdf.putAtt( ncid, varid_global, 'institution', 'SIO' );

% (CF)  source
netcdf.putAtt( ncid, varid_global, 'source', 'Surface Ocean HF-Radar' );

% (CF, ACDD) history
netcdf.putAtt( ncid, varid_global, 'history', hist_create );

% (CF)  references
netcdf.putAtt( ncid, varid_global, 'references', 'CODAR SeaSonde LonLatUV (LLUV) File Format' );

% (ACDD)        creator_name
netcdf.putAtt( ncid, varid_global, 'creator_name', 'HF-Radar Network Administrators' );

% (ACDD)        creator_email
netcdf.putAtt( ncid, varid_global, 'creator_email', 'hfrnetadm@ucsd.edu' );

% (ACDD)        creator_url
netcdf.putAtt( ncid, varid_global, 'creator_url', 'http://cordc.ucsd.edu/projects/mapping/' );

% (ACDD)        summary
nc_summary = sprintf( '%s\n%s\n%s', ...
   'HF-RADAR measurements of ocean velocity are radial in', ...
   'direction relative to the radar location and representitive ', ...
   'of the upper 0.3 - 2.5 meters of the ocean.' );
netcdf.putAtt( ncid, varid_global, 'summary', nc_summary);

% (ACDD)        geospatial_[lon|lat]_[max|min]
% For totals, this is computed from the grid, not from actual data
% availability limits. Analogous to remote sensing where availability is
% relative to pass, not necessairly where data is. However, unless we want
% to compute lat/lon for range/bearing where no sample is available, just
% report lat/lon limit of data availability.
netcdf.putAtt( ncid, varid_global, 'geospatial_lat_min', single(min(min(latd))) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lat_max', single(max(max(latd))) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lon_min', single(min(min(lond))) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lon_max', single(max(max(lond))) );

% Globals sourced from radial file metadata
netcdf.putAtt( ncid, varid_global, 'CTF', '1.00');
netcdf.putAtt( ncid, varid_global, 'FileType', 'LLUV rdls RadialMap' );
netcdf.putAtt( ncid, varid_global, 'LLUVSpec', '1.17  2011 06 20' );
```

```
netcdf.putAtt( ncid, varid_global, 'UUID', '6BDBE8F4-E7CD-4ECE-B92C-74BF64148941' );
netcdf.putAtt( ncid, varid_global, 'manufacturer', 'CODAR Ocean Sensors. SeaSonde' );
netcdf.putAtt( ncid, varid_global, 'Network', 'ODU' ); %inserted by HFRNet
netcdf.putAtt( ncid, varid_global, 'Site', 'LISL' );
netcdf.putAtt( ncid, varid_global, 'TimeZone', 'UTC +0.000 0 GMT' );
netcdf.putAtt( ncid, varid_global, 'TimeCoverage', '180.000 Minutes' );
netcdf.putAtt( ncid, varid_global, 'Origin', '36.6917167  -75.9226333' );
netcdf.putAtt( ncid, varid_global, 'GreatCircle', 'WGS84 6378137.000  298.257223562997' );
netcdf.putAtt( ncid, varid_global, 'GeodVersion', 'CGEO 1.57  2009 03 10' );
netcdf.putAtt( ncid, varid_global, 'LLUVTrustData', 'all' );
netcdf.putAtt( ncid, varid_global, 'RangeStart', '1' );
netcdf.putAtt( ncid, varid_global, 'RangeEnd', '33' );
netcdf.putAtt( ncid, varid_global, 'RangeResolutionKMeters', '5.824900' );
netcdf.putAtt( ncid, varid_global, 'AntennaBearing', '130.0 True' );
netcdf.putAtt( ncid, varid_global, 'ReferenceBearing', '0 True' );
netcdf.putAtt( ncid, varid_global, 'AngularResolution', '5 Deg' );
netcdf.putAtt( ncid, varid_global, 'SpatialResolution', '5 Deg' );
netcdf.putAtt( ncid, varid_global, 'PatternType', 'Ideal' );
netcdf.putAtt( ncid, varid_global, 'PatternDate', '2009 06 04  15 13 32' );
netcdf.putAtt( ncid, varid_global, 'PatternResolution', '1.0 deg' );
netcdf.putAtt( ncid, varid_global, 'TransmitCenterFreqMHz', '4.575000' );
netcdf.putAtt( ncid, varid_global, 'DopplerResolutionHzPerBin', '0.000488281' );
netcdf.putAtt( ncid, varid_global, 'FirstOrderMethod', '0' );
netcdf.putAtt( ncid, varid_global, 'BraggSmoothingPoints', '2' );
netcdf.putAtt( ncid, varid_global, 'CurrentVelocityLimit', '250.0' );
netcdf.putAtt( ncid, varid_global, 'BraggHasSecondOrder', '0' );
netcdf.putAtt( ncid, varid_global, 'RadialBraggPeakDropOff', '151.360' );
netcdf.putAtt( ncid, varid_global, 'RadialBraggPeakNull', '10.000' );
netcdf.putAtt( ncid, varid_global, 'RadialBraggNoiseThreshold', '5.000' );
netcdf.putAtt( ncid, varid_global, 'PatternAmplitudeCorrections', '0.9981  0.9981' );
netcdf.putAtt( ncid, varid_global, 'PatternPhaseCorrections', '74.00  67.00' );
netcdf.putAtt( ncid, varid_global, 'PatternAmplitudeCalculations', '0.0517  0.0641' );
netcdf.putAtt( ncid, varid_global, 'PatternPhaseCalculations', '64.50  55.60' );
netcdf.putAtt( ncid, varid_global, 'RadialMusicParameters', '40.000 20.000 2.000' );
netcdf.putAtt( ncid, varid_global, 'MergedCount', '5' );
netcdf.putAtt( ncid, varid_global, 'RadialMinimumMergePoints', '2' );
netcdf.putAtt( ncid, varid_global, 'FirstOrderCalc', '1' );
netcdf.putAtt( ncid, varid_global, 'MergeMethod', '1 MedianVectors' );
netcdf.putAtt( ncid, varid_global, 'PatternMethod', '1 PatternVectors' );
netcdf.putAtt( ncid, varid_global, 'TransmitSweepRateHz', '1.000000' );
netcdf.putAtt( ncid, varid_global, 'TransmitBandwidthKHz', '-25.733913' );
netcdf.putAtt( ncid, varid_global, 'SpectraRangeCells', '63' );
netcdf.putAtt( ncid, varid_global, 'SpectraDopplerCells', '2048' );
netcdf.putAtt( ncid, varid_global, 'TableType', 'LLUV RDL9' );
netcdf.putAtt( ncid, varid_global, 'TableColumns', '18' );
netcdf.putAtt( ncid, varid_global, 'TableColumnTypes', sprintf('%s\n%s', ...
   'LOND LATD VELU VELV VFLG ESPC ETMP MAXV MINV', ...
   'ERSC ERTC XDST YDST RNGE BEAR VELO HEAD SPRC' ) );
netcdf.putAtt( ncid, varid_global, 'TableRows', '1260' );


% Add auxillary coordinate variables to provide mapping from range &
% bearing to lat, lon.

% A minimum of 4 significant figures to the right of the decimal
% place is needed to keep resolution below 10's of meters.  Using
```

```
% float data type for lat yields at least 5 significant digits to
% the right of the decimal giving ~1/2m resolution in latitude.
% Nine significant figures (at least 7 to the right of the
% decimal) could be achieved using int data type but need to
% introduce a scale factor.
%
% Latitude
varid_lat = netcdf.defVar( ncid, 'lat', 'float', [dimid_range dimid_bearing] );
netcdf.defVarDeflate(ncid, varid_lat, true, true, 6);
netcdf.putAtt( ncid, varid_lat, 'standard_name', 'latitude' );
netcdf.putAtt( ncid, varid_lat, 'units', 'degrees_north' );

% Longitude
varid_lon = netcdf.defVar( ncid, 'lon', 'float', [dimid_range dimid_bearing] );
netcdf.defVarDeflate(ncid, varid_lon, true, true, 6);
netcdf.putAtt( ncid, varid_lon, 'standard_name', 'longitude' );
netcdf.putAtt( ncid, varid_lon, 'units', 'degrees_east' );


% Add Data Variables

% radial_sea_water_velocity_away_from_instrument:
% A velocity is a vector quantity. Radial velocity away from instrument
% means the component of the velocity along the line of sight of the
% instrument where positive implies movement away from the instrument (i.e.
% outward). The "instrument" a(examples are radar and lidar) is the device
% used to make an observation.
varid_speed = netcdf.defVar( ncid, 'speed', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_speed, true, true, 6);
netcdf.putAtt( ncid, varid_speed, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_speed, 'standard_name', 'radial_sea_water_velocity_away_from_instrument' );
netcdf.putAtt( ncid, varid_speed, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_speed, 'coordinates', 'lon lat' );

% (radial current) direction
%
% direction_of_radial_vector_away_from_instrument:
% The direction_of_radial_vector_away_from_instrument is the direction in
% which the instrument itself is pointing. The direction is measured
% positive clockwise from due north. The "instrument" (examples are radar
% and lidar) is the device used to make an observation. "direction_of_X"
% means direction of a vector, a bearing.
varid_direction = netcdf.defVar( ncid, 'direction', 'short', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_direction, true, true, 6);
netcdf.putAtt( ncid, varid_direction, 'valid_range', int16( [0 3600] ) );
netcdf.putAtt( ncid, varid_direction, 'standard_name', 'direction_of_radial_vector_away_from_instrument'
);
netcdf.putAtt( ncid, varid_direction, 'units', 'degrees_true' );
netcdf.putAtt( ncid, varid_direction, 'coordinates', 'lon lat' );
netcdf.putAtt( ncid, varid_direction, 'scale_factor', single(0.1) );

% u
varid_u = netcdf.defVar( ncid, 'u', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_u, true, true, 6);
netcdf.putAtt( ncid, varid_u, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_u, 'standard_name', 'surface_eastward_sea_water_velocity' );
```

```
netcdf.putAtt( ncid, varid_u, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_u, 'coordinates', 'lon lat' );

% v
varid_v = netcdf.defVar( ncid, 'v', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_v, true, true, 6);
netcdf.putAtt( ncid, varid_v, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_v, 'standard_name', 'surface_northward_sea_water_velocity' );
netcdf.putAtt( ncid, varid_v, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_v, 'coordinates', 'lon lat' );

% Vector Flag
varid_vflg = netcdf.defVar( ncid, 'vflg', 'short', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_vflg, true, true, 6);
netcdf.putAtt( ncid, varid_vflg, 'long_name', 'vector_flag_masks' );
netcdf.putAtt( ncid, varid_vflg, 'valid_range', int16( [0 2048] ));
netcdf.putAtt( ncid, varid_vflg, 'flag_masks', int16( 2.^[0:10] ));
netcdf.putAtt( ncid, varid_vflg, 'flag_meanings', 'grid_point_deleted grid_point_near_coast
point_measurement no_radial_solution baseline_interpolation exceeds_max_speed invalid_solution
solution_beyond_valid_spatial_domain insufficient_angular_resolution reserved reserved');
netcdf.putAtt( ncid, varid_vflg, 'coordinates', 'lon lat' );

% Spatial Quality
varid_espc = netcdf.defVar( ncid, 'espc', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_espc, true, true, 6);
netcdf.putAtt( ncid, varid_espc, 'long_name', 'radial_sea_water_velocity_spatial_quality' );
netcdf.putAtt( ncid, varid_espc, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_espc, 'coordinates', 'lon lat' );

% Temporal Quality
varid_etmp = netcdf.defVar( ncid, 'etmp', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_etmp, true, true, 6);
netcdf.putAtt( ncid, varid_etmp, 'long_name', 'radial_sea_water_velocity_temporal_quality' );
netcdf.putAtt( ncid, varid_etmp, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_etmp, 'coordinates', 'lon lat' );

% Velocity Maximum
varid_maxv = netcdf.defVar( ncid, 'maxv', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_maxv, true, true, 6);
netcdf.putAtt( ncid, varid_maxv, 'long_name',
'radial_sea_water_velocity_away_from_instrument_maximum' );
netcdf.putAtt( ncid, varid_maxv, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_maxv, 'coordinates', 'lon lat' );

% Velocity Minimum
varid_minv = netcdf.defVar( ncid, 'minv', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_minv, true, true, 6);
netcdf.putAtt( ncid, varid_minv, 'long_name',
'radial_sea_water_velocity_away_from_instrument_minimum' );
netcdf.putAtt( ncid, varid_minv, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_minv, 'coordinates', 'lon lat' );

% Spatial Count
varid_ersc = netcdf.defVar( ncid, 'ersc', 'byte', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_ersc, true, true, 6);
netcdf.putAtt( ncid, varid_ersc, 'long_name', 'radial_sea_water_velocity_spatial_quality_count' );
```

```
netcdf.putAtt( ncid, varid_ersc, 'coordinates', 'lon lat' );

% Temporal Count
varid_ertc = netcdf.defVar( ncid, 'ertc', 'byte', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_ertc, true, true, 6);
netcdf.putAtt( ncid, varid_ertc, 'long_name', 'radial_sea_water_velocity_temporal_quality_count' );
netcdf.putAtt( ncid, varid_ertc, 'coordinates', 'lon lat' );

% X-Distance
varid_xdst = netcdf.defVar( ncid, 'xdst', 'float', [dimid_range dimid_bearing] );
netcdf.defVarDeflate(ncid, varid_xdst, true, true, 6);
netcdf.putAtt( ncid, varid_xdst, 'long_name', 'eastward_distance_from_instrument' );
netcdf.putAtt( ncid, varid_xdst, 'units', 'km' );
netcdf.putAtt( ncid, varid_xdst, 'coordinates', 'lon lat' );

% Y-Distance
varid_ydst = netcdf.defVar( ncid, 'ydst', 'float', [dimid_range dimid_bearing] );
netcdf.defVarDeflate(ncid, varid_ydst, true, true, 6);
netcdf.putAtt( ncid, varid_ydst, 'long_name', 'northward_distance_from_instrument' );
netcdf.putAtt( ncid, varid_ydst, 'units', 'km' );
netcdf.putAtt( ncid, varid_ydst, 'coordinates', 'lon lat' );

% Spectra Range Cell
varid_sprc = netcdf.defVar( ncid, 'sprc', 'byte', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_sprc, true, true, 6);
netcdf.putAtt( ncid, varid_sprc, 'long_name', 'radial_sea_water_velocity_cross_spectal_range_cell' );
netcdf.putAtt( ncid, varid_sprc, 'coordinates', 'lon lat' );

% Exit definition mode
netcdf.endDef( ncid );

%% Write variable data
netcdf.putVar( ncid, varid_bearing, bearing_dim );
netcdf.putVar( ncid, varid_range, range_dim );
netcdf.putVar( ncid, varid_t, 0, round( ( R.time - datenum(1970,1,1) )*86400) );
netcdf.putVar( ncid, varid_lat, latd );
netcdf.putVar( ncid, varid_lon, lond );
netcdf.putVar( ncid, varid_speed, velo );
netcdf.putVar( ncid, varid_direction, head );
netcdf.putVar( ncid, varid_u, velu );
netcdf.putVar( ncid, varid_v, velv );
netcdf.putVar( ncid, varid_vflg, vflg );
netcdf.putVar( ncid, varid_espc, espc );
netcdf.putVar( ncid, varid_etmp, etmp );
netcdf.putVar( ncid, varid_maxv, maxv );
netcdf.putVar( ncid, varid_minv, minv );
netcdf.putVar( ncid, varid_ersc, ersc );
netcdf.putVar( ncid, varid_ertc, ertc );
netcdf.putVar( ncid, varid_xdst, xdst );
netcdf.putVar( ncid, varid_ydst, ydst );
netcdf.putVar( ncid, varid_sprc, sprc );


%% Close file
netcdf.close( ncid );
```

%% Copyright
% Author: Mark Otero, 2013/08/23
% Copyright: 2013 Coastal Observing Research and Development Center,
% Scripps Institution of Oceanography

Example image from Unidata's Integrated Data Viewer (IDV)

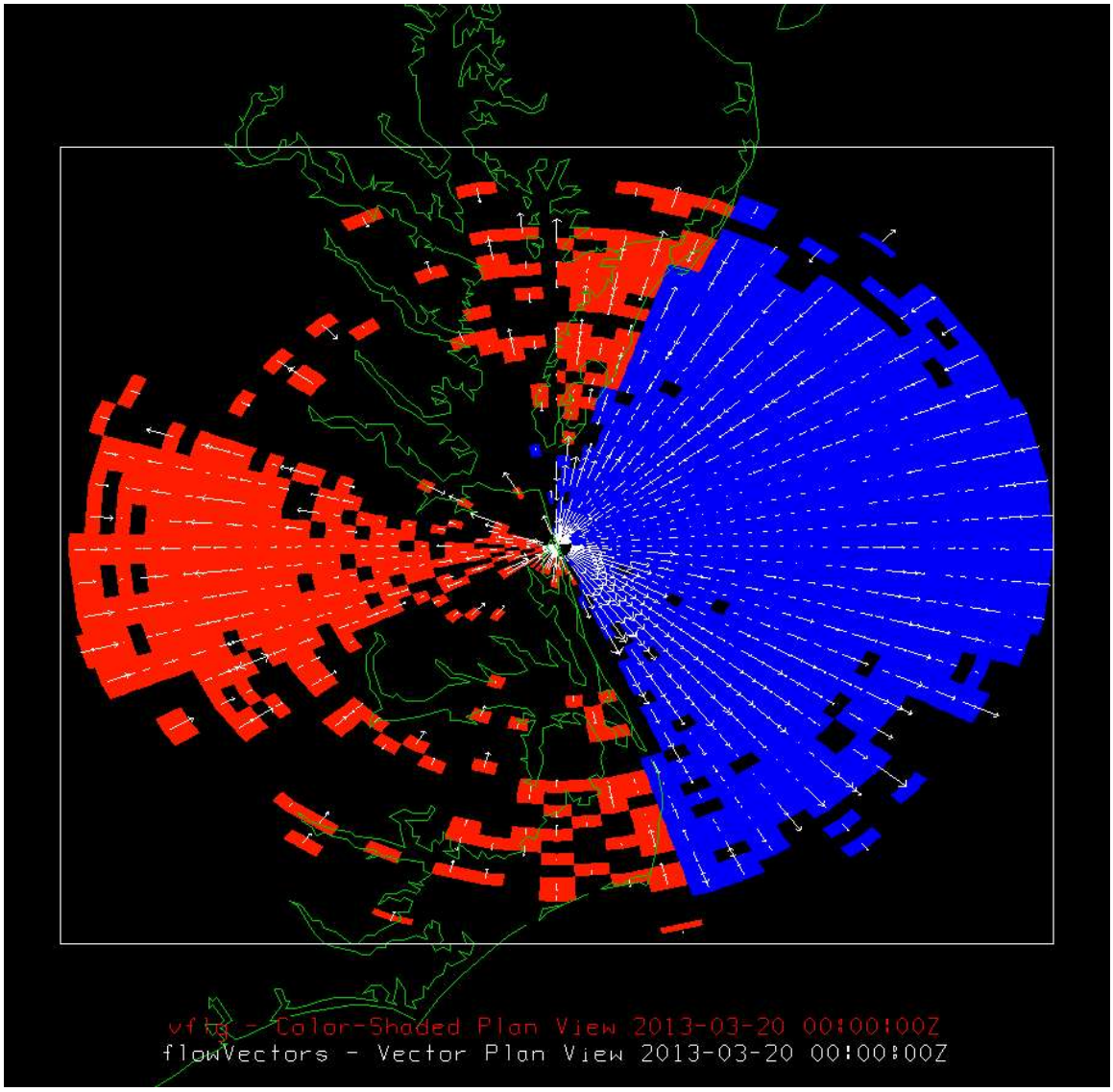

Figure 1. CODAR radial velocities plotted over vector flag values. Red vector flags represent solutions ouside the valid sample domain.

Appendix B

WERA NetCDF Encoding

WERA NetCDF header in common data language (CDL)

```
netcdf RDL_SC_GTN_2013_05_08_1053 {
dimensions:
        time = UNLIMITED ; // (1 currently)
        lat = 95 ;
        lon = 73 ;
variables:
        int time(time) ;
                time:standard_name = "time" ;
                time:units = "seconds since 1970-01-01" ;
                time:calendar = "gregorian" ;
        float lat(lat) ;
                lat:standard_name = "latitude" ;
                lat:units = "degrees_north" ;
        float lon(lon) ;
                lon:standard_name = "longitude" ;
                lon:units = "degrees_east" ;
        short bearing(lat, lon) ;
                bearing:valid_range = 0s, 3600s ;
                bearing:long_name = "bearing_away_from_instrument" ;
                bearing:units = "degrees_true" ;
                bearing:scale_factor = 0.1f ;
        float range(lat, lon) ;
                range:long_name = "range_away_from_instrument" ;
                range:units = "km" ;
        float speed(time, lat, lon) ;
                speed:valid_range = -1000.f, 1000.f ;
                speed:standard_name = "radial_sea_water_velocity_away_from_instrument" ;
                speed:units = "cm s-1" ;
        short direction(time, lat, lon) ;
                direction:valid_range = 0s, 3600s ;
                direction:standard_name = "direction_of_radial_vector_away_from_instrument" ;
                direction:units = "degrees_true" ;
                direction:scale_factor = 0.1f ;
        float u(time, lat, lon) ;
                u:valid_range = -1000.f, 1000.f ;
                u:standard_name = "surface_eastward_sea_water_velocity" ;
                u:units = "cm s-1" ;
        float v(time, lat, lon) ;
                v:valid_range = -1000.f, 1000.f ;
                v:standard_name = "surface_northward_sea_water_velocity" ;
                v:units = "cm s-1" ;
        float evar(time, lat, lon) ;
                evar:long_name = "radial_sea_water_velocity_variance" ;
                evar:units = "cm s-1" ;
        float eacc(time, lat, lon) ;
                eacc:long_name = "radial_sea_water_velocity_accuracy" ;
                eacc:units = "cm s-1" ;
        float xdst(lat, lon) ;
                xdst:long_name = "eastward_distance_from_instrument" ;
                xdst:units = "km" ;
        float ydst(lat, lon) ;
                ydst:long_name = "northward_distance_from_instrument" ;
                ydst:units = "km" ;

// global attributes:
                :netcdf_library_version = "4.1.3" ;
```

```
        :Conventions = "CF-1.6" ;
        :title = "Near-Real Time Surface Ocean Radial Velocity" ;
        :institution = "SIO" ;
        :source = "Surface Ocean HF-Radar" ;
        :history = "23-Aug-2013 11:35:51: NetCDF file created" ;
        :references = "WERA Radial LonLatUV (LLUV) File Format" ;
        :creator_name = "HF-Radar Network Administrators" ;
        :creator_email = "hfrnetadm@ucsd.edu" ;
        :creator_url = "http://cordc.ucsd.edu/projects/mapping/" ;
        :summary = "HF-RADAR measurements of ocean velocity are radial in\n",
            "direction relative to the radar location and representitive \n",
            "of the upper 0.3 - 2.5 meters of the ocean." ;
        :geospatial_lat_min = 31.70518f ;
        :geospatial_lat_max = 34.24284f ;
        :geospatial_lon_min = -79.11958f ;
        :geospatial_lon_max = -76.80283f ;
        :CTF = "1.00" ;
        :FileType = "LLUV rdls" ;
        :Manufacturer = "Helzel Messtechnik GmbH, WERA." ;
        :LLUVSpec = "1.00 2007 12 06" ;
        :Network = "SC" ;
        :Site = "gtn GTN" ;
        :TimeStamp = "2013 05 08 10 53 00" ;
        :TimeZone = "UTC +0.00 0" ;
        :TimeCoverage = "887.46600342 Seconds" ;
        :Origin = "33.356111 -79.152500" ;
        :GreatCircle = "WGS84 6378137.000 298.257223562997" ;
        :GeodVersion = "NGS-Vincenty 2.0 2002 10 01" ;
        :RangeResolutionKMeters = "3.000" ;
        :TransmitCenterFreqMHz = "8.3420" ;
        :DopplerResolutionHzPerBin = "0.001126804" ;
        :CurrentVelocityLimit = "250.0" ;
        :MergedCount = "0001" ;
        :TransmitSweepRateHz = "0.43333" ;
        :TransmitBandwidthKHz = "49.965" ;
        :SpectraRangeCells = "080" ;
        :SpectraDopplerCells = "02048" ;
        :TableType = "LLUV RDL1" ;
        :TableColumns = "12" ;
        :TableColumnTypes = "LOND LATD VELU VELV EVAR EACC XDST YDST RNGE \n",
            "BEAR VELO HEAD" ;
        :TableRows = "2751" ;
}
```

Sample MATLAB code for encoding WERA data in NetCDF from a source ASCII file

```
% Development script to format HF-Radar radial files in netCDF.

%% Load sample data
rfile_path = '/home/motero/Documents/Projects/HFRNet/Development/NetCDF Radial/Sample Files/';
rfile = fullfile( rfile_path, 'RDL_SC_GTN_2013_05_08_1053.hfrweralluv1.0');
r = load( rfile );
R = struct( ...
    'lond', r(:, 1), ...
```

```
    'latd', r(:, 2), ...
    'velu', r(:, 3), ...
    'velv', r(:, 4), ...
    'evar', r(:, 5), ...
    'eacc', r(:, 6), ...
    'xdst', r(:, 7), ...
    'ydst', r(:, 8), ...
    'rnge', r(:, 9), ...
    'bear', r(:,10), ...
    'velo', r(:,11), ...
    'head', r(:,12)  ...
);
clear r;

%TimeStamp: 2013 05 08 10 53 00
R.time = datenum(2013,5,8,10,53,0);

%% Generate grid domain
% Can do this purely based on available data but may not be able
% to aggregate in TDS.  Alternative is to define a constant grid for the
% site on which all data can fall - which is what is attempted below but
% probably wouldn't be done per-file and hard limits on max/mins would be
% imposed.  Would also have implications on if data is interpolated or just
% indexed into the new grid...
lon_dim = unique(R.lond);
lat_dim = unique(R.latd);
dx = mode( unique( diff( lon_dim ) ) );
dy = mode( unique( diff( lat_dim ) ) );
lon_dim = [min(lon_dim) : dx : max(lon_dim)];
lat_dim = [min(lat_dim) : dy : max(lat_dim)];
[lon, lat] = meshgrid(lat_dim, lon_dim);

% check - see that our grid captures the sample space right
% figure
% plot( lon, lat, 'go' )
% hold on
% plot( R.lond, R.latd, 'b.')
%% Determine the mapping index from sample data to full grid
lon_map_idx   = nan( 1, numel(R.lond) );
lat_map_idx = nan( 1, numel(R.latd) );
for I = 1 : numel(R.lond)
    [~, lon_map_idx(I)]   = min( abs( lon_dim - R.lond(I) ) );
    [~, lat_map_idx(I)] = min( abs( lat_dim - R.latd(I) ) );
end
clear I;
ll_map_idx = sub2ind( size( lon ), lon_map_idx, lat_map_idx );


%% Populate gridded matrices

% Eastward Velocity
velu = nan( size( lon ) );
velu(ll_map_idx) = R.velu;

% Northward Velocity
velv = nan( size( lon ) );
```

```
velv(ll_map_idx) = R.velv;

% check
% figure
% quiver(R.lond, R.latd, R.velu./1000, R.velv./1000, 0, 'k')
% daspect([1.2 1 1])
% hold on
% quiver(lon, lat, velu./1000, velv./1000, 0, 'r')

% Current Speed - flip sign so positive velocity is away from radar
% according to CF standard name
% 'radial_sea_water_velocity_away_from_instrument'. CODAR reports positive
% speeds as toward the radar.
velo = nan( size( lon ) );
velo(ll_map_idx) = -R.velo;

% Current Heading
head = nan( size( lon ) );
head(ll_map_idx) = R.head;

% Bearing
bear = nan( size( lon ) );
bear(ll_map_idx) = R.bear;

% Range
rnge = nan( size( lon ) );
rnge(ll_map_idx) = R.rnge;

% Variance
evar = nan( size( lon ) );
evar(ll_map_idx) = R.evar;

% Accuracy
eacc = nan( size( lon ) );
eacc(ll_map_idx) = R.eacc;

% X-Distance
xdst = nan( size( lon ) );
xdst(ll_map_idx) = R.xdst;

% Y-Distance
ydst = nan( size( lon ) );
ydst(ll_map_idx) = R.ydst;

%% Scaling & Fill Values
% Bearing and heading are only reported to 10ths and can be
% reported as short unsigned integers when scaled.
% Also, can only use unsigned when using NetCDF-4 enhanced data model,
% otherwise w/classic data model you're limited ot signed data types.
head = round( head.*10 );
bear = round( bear.*10 );

% Type short fill values
head( isnan(head) ) = netcdf.getConstant('NC_FILL_SHORT');
bear( isnan(bear) ) = netcdf.getConstant('NC_FILL_SHORT');
```

```
% Type float fill values
rnge( isnan(rnge) ) = netcdf.getConstant('NC_FILL_FLOAT');
velo( isnan(velo) ) = netcdf.getConstant('NC_FILL_FLOAT');
velu( isnan(velu) ) = netcdf.getConstant('NC_FILL_FLOAT');
velv( isnan(velv) ) = netcdf.getConstant('NC_FILL_FLOAT');
evar( isnan(evar) ) = netcdf.getConstant('NC_FILL_FLOAT');
eacc( isnan(eacc) ) = netcdf.getConstant('NC_FILL_FLOAT');
xdst( isnan(xdst) ) = netcdf.getConstant('NC_FILL_FLOAT');
ydst( isnan(ydst) ) = netcdf.getConstant('NC_FILL_FLOAT');


%% Define dimensions, variables and attributes
ncfile = strrep( rfile, 'hfrweralluv1.0', 'nc' );

% Keep the classic data model to increase compatibility since we
% aren't using features specific to the NetCDF-4.0 model.
mode = netcdf.getConstant( 'NETCDF4' );
mode = bitor(mode, netcdf.getConstant( 'CLASSIC_MODEL' ) );

ncid = netcdf.create( ncfile, mode );
hist_create = [datestr(now) ': NetCDF file created'];

% Add dimensions (in order of T, Z, Y, X for CF/COARDS compliance)
dimid_t      = netcdf.defDim( ncid, 'time',    netcdf.getConstant('unlimited') );
dimid_lat        = netcdf.defDim( ncid, 'lat', length(lat_dim) );
dimid_lon         = netcdf.defDim( ncid, 'lon', length(lon_dim) );

% Add coordinate variables and attributes in the same order as
% the dimensions were written

% For time coordinate variable, when using the standard or gregorian
% calander, units should be in seconds and shifted forward beyond
% 1582/10/15 to avoid Julian -> Gregorian crossover and potential
% problems with the udunits package, if used.  Year length should be
% exactly 365.2425 days long for the Gregorian/standard and proleptic
% Gregorian calander.
%
% MATLAB's datenum uses the proleptic gregorian calander as defined
% by the CF standard.  When using standard_name attribute for time
% coordinate be sure to include the calander and units attributes.
%
% Use int for time data type because need 8 significant figures to
% represent number of seconds in 1 year (31556952 seconds), float
% isn't enough (7 sig. figures).  Would need to use double beyond
% int.  Besides, don't need sub-second accuracy.  Unix time will
% overflow int in 2038.  New epoch will be needed then to keep int
% representation.  Unix time epoch is:
%          1970-01-01 00:00:00Z = 0 seconds, 86,400 sec/day
varid_t = netcdf.defVar( ncid, 'time', 'int', dimid_t );
netcdf.putAtt( ncid, varid_t, 'standard_name', 'time' );
netcdf.putAtt( ncid, varid_t, 'units', 'seconds since 1970-01-01' );
netcdf.putAtt( ncid, varid_t, 'calendar', 'gregorian' );

% A minimum of 4 significant figures to the right of the decimal
% place is needed to keep resolution below 10's of meters.  Using
% float data type for lat yields at least 5 significant digits to
```

```
% the right of the decimal giving ~1/2m resolution in latitude.
% Nine significant figures (at least 7 to the right of the
% decimal) could be achieved using int data type but need to
% introduce a scale factor.
varid_lat = netcdf.defVar( ncid, 'lat', 'float', dimid_lat );
netcdf.putAtt( ncid, varid_lat, 'standard_name', 'latitude' );
netcdf.putAtt( ncid, varid_lat, 'units', 'degrees_north' );


% A minimum of 4 significant figures to the right of the decimal
% place is needed to keep resolution below 10's of meters.  Using
% float data type for lon yields at least 4 significant digits to
% the right of the decimal giving ~1/2m resolution in longitude.
% Nine significant figures (at least 6 to the right of the
% decimal) could be achieved using int data type but need to
% introduce a scale factor.
varid_lon = netcdf.defVar( ncid, 'lon', 'float', dimid_lon );
netcdf.putAtt( ncid, varid_lon, 'standard_name', 'longitude' );
netcdf.putAtt( ncid, varid_lon, 'units', 'degrees_east' );


% Add global attributes

% Currently using Climate & Forecast Metadata Conventions v1.6
% (CF).  Am also using Attribute Convention for Dataset Discovery
% v1.1 (ACDD) where applicable along with
% a few of my own attributes (X) for things that didn't fit
% well or weren't clearly applicable to features in either
% convention.  The ACDD convention is still developing and may
% eventually support the attributes added.

varid_global = netcdf.getConstant('global');

% (X) NetCDF library version used
netcdf.putAtt( ncid, varid_global, 'netcdf_library_version', netcdf.inqLibVers );

% (CF)  Conventions
netcdf.putAtt( ncid, varid_global, 'Conventions', 'CF-1.6' );

% (CF, ACDD) title
netcdf.putAtt( ncid, varid_global, 'title', 'Near-Real Time Surface Ocean Radial Velocity' );

% (CF, ACDD) institution
netcdf.putAtt( ncid, varid_global, 'institution', 'SIO' );

% (CF)  source
netcdf.putAtt( ncid, varid_global, 'source', 'Surface Ocean HF-Radar' );

% (CF, ACDD) history
netcdf.putAtt( ncid, varid_global, 'history', hist_create );

% (CF)  references
netcdf.putAtt( ncid, varid_global, 'references', 'WERA Radial LonLatUV (LLUV) File Format' );

% (ACDD)        creator_name
netcdf.putAtt( ncid, varid_global, 'creator_name', 'HF-Radar Network Administrators' );
```

```
% (ACDD)          creator_email
netcdf.putAtt( ncid, varid_global, 'creator_email', 'hfrnetadm@ucsd.edu' );

% (ACDD)          creator_url
netcdf.putAtt( ncid, varid_global, 'creator_url', 'http://cordc.ucsd.edu/projects/mapping/' );

% (ACDD)          summary
nc_summary = sprintf( '%s\n%s\n%s', ...
   'HF-RADAR measurements of ocean velocity are radial in', ...
   'direction relative to the radar location and representitive ', ...
   'of the upper 0.3 - 2.5 meters of the ocean.' );
netcdf.putAtt( ncid, varid_global, 'summary', nc_summary);

% (ACDD)          geospatial_[lon|lat]_[max|min]
% For totals, this is computed from the grid, not from actual data
% availability limits. Analogous to remote sensing where availability is
% relative to pass, not necessairly where data is. However, unless we want
% to compute lat/lon for range/bearing where no sample is available, just
% report lat/lon limit of data availability.
netcdf.putAtt( ncid, varid_global, 'geospatial_lat_min', single(min(lat_dim)) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lat_max', single(max(lat_dim)) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lon_min', single(min(lon_dim)) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lon_max', single(max(lon_dim)) );

% Globals sourced from radial file metadata
netcdf.putAtt( ncid, varid_global, 'CTF', '1.00' );
netcdf.putAtt( ncid, varid_global, 'FileType', 'LLUV rdls' );
netcdf.putAtt( ncid, varid_global, 'Manufacturer', 'Helzel Messtechnik GmbH, WERA.' );
netcdf.putAtt( ncid, varid_global, 'LLUVSpec', '1.00 2007 12 06' );
netcdf.putAtt( ncid, varid_global, 'Network', 'SC' ); %inserted by HFRNet
netcdf.putAtt( ncid, varid_global, 'Site', 'gtn GTN' );
netcdf.putAtt( ncid, varid_global, 'TimeStamp', '2013 05 08 10 53 00' );
netcdf.putAtt( ncid, varid_global, 'TimeZone', 'UTC +0.00 0' );
netcdf.putAtt( ncid, varid_global, 'TimeCoverage', '887.46600342 Seconds' );
netcdf.putAtt( ncid, varid_global, 'Origin', '33.356111 -79.152500' );
netcdf.putAtt( ncid, varid_global, 'GreatCircle', 'WGS84 6378137.000 298.257223562997' );
netcdf.putAtt( ncid, varid_global, 'GeodVersion', 'NGS-Vincenty 2.0 2002 10 01' );
netcdf.putAtt( ncid, varid_global, 'RangeResolutionKMeters', '3.000' );
netcdf.putAtt( ncid, varid_global, 'TransmitCenterFreqMHz', '8.3420' );
netcdf.putAtt( ncid, varid_global, 'DopplerResolutionHzPerBin', '0.001126804' );
netcdf.putAtt( ncid, varid_global, 'CurrentVelocityLimit', '250.0' );
netcdf.putAtt( ncid, varid_global, 'MergedCount', '0001' );
netcdf.putAtt( ncid, varid_global, 'TransmitSweepRateHz', '0.43333' );
netcdf.putAtt( ncid, varid_global, 'TransmitBandwidthKHz', '49.965' );
netcdf.putAtt( ncid, varid_global, 'SpectraRangeCells', '080' );
netcdf.putAtt( ncid, varid_global, 'SpectraDopplerCells', '02048' );
netcdf.putAtt( ncid, varid_global, 'TableType', 'LLUV RDL1' );
netcdf.putAtt( ncid, varid_global, 'TableColumns', '12' );
netcdf.putAtt( ncid, varid_global, 'TableColumnTypes', sprintf('%s\n%s', ...
   'LOND LATD VELU VELV EVAR EACC XDST YDST RNGE ', ...
   'BEAR VELO HEAD' ) );
netcdf.putAtt( ncid, varid_global, 'TableRows', '2751' );

% Add Data Variables
```

```
% Range and bearing are not explicitly added as auxillary coordinate
% variables since implementation and usage are unclear at this point.
% Waiting for usage scenario to manifest before defining more explicitly.

% Bearing (arbitrary 'y' dimension)
varid_bearing = netcdf.defVar( ncid, 'bearing', 'short', [dimid_lon dimid_lat] );
netcdf.defVarDeflate(ncid, varid_bearing, true, true, 6);
netcdf.putAtt( ncid, varid_bearing, 'valid_range', int16( [0 3600] ) );
netcdf.putAtt( ncid, varid_bearing, 'long_name', 'bearing_away_from_instrument' );
netcdf.putAtt( ncid, varid_bearing, 'units', 'degrees_true' );
netcdf.putAtt( ncid, varid_bearing, 'scale_factor', single(0.1) );

% Range (arbitrary 'x' dimension)
varid_range = netcdf.defVar( ncid, 'range', 'float', [dimid_lon dimid_lat] );
netcdf.defVarDeflate(ncid, varid_range, true, true, 6);
netcdf.putAtt( ncid, varid_range, 'long_name', 'range_away_from_instrument' );
netcdf.putAtt( ncid, varid_range, 'units', 'km' );

% radial_sea_water_velocity_away_from_instrument:
% A velocity is a vector quantity. Radial velocity away from instrument
% means the component of the velocity along the line of sight of the
% instrument where positive implies movement away from the instrument (i.e.
% outward). The "instrument" (examples are radar and lidar) is the device
% used to make an observation.
varid_speed = netcdf.defVar( ncid, 'speed', 'float', [dimid_lon dimid_lat dimid_t] );
netcdf.defVarDeflate(ncid, varid_speed, true, true, 6);
netcdf.putAtt( ncid, varid_speed, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_speed, 'standard_name', 'radial_sea_water_velocity_away_from_instrument' );
netcdf.putAtt( ncid, varid_speed, 'units', 'cm s-1' );

% (radial current) direction
%
% direction_of_radial_vector_away_from_instrument:
% The direction_of_radial_vector_away_from_instrument is the direction in
% which the instrument itself is pointing. The direction is measured
% positive clockwise from due north. The "instrument" (examples are radar
% and lidar) is the device used to make an observation. "direction_of_X"
% means direction of a vector, a bearing.
varid_direction = netcdf.defVar( ncid, 'direction', 'short', [dimid_lon dimid_lat dimid_t] );
netcdf.defVarDeflate(ncid, varid_direction, true, true, 6);
netcdf.putAtt( ncid, varid_direction, 'valid_range', int16( [0 3600] ) );
netcdf.putAtt( ncid, varid_direction, 'standard_name', 'direction_of_radial_vector_away_from_instrument'
);
netcdf.putAtt( ncid, varid_direction, 'units', 'degrees_true' );
netcdf.putAtt( ncid, varid_direction, 'scale_factor', single(0.1) );

% u
varid_u = netcdf.defVar( ncid, 'u', 'float', [dimid_lon dimid_lat dimid_t] );
netcdf.defVarDeflate(ncid, varid_u, true, true, 6);
netcdf.putAtt( ncid, varid_u, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_u, 'standard_name', 'surface_eastward_sea_water_velocity' );
netcdf.putAtt( ncid, varid_u, 'units', 'cm s-1' );

% v
varid_v = netcdf.defVar( ncid, 'v', 'float', [dimid_lon dimid_lat dimid_t] );
netcdf.defVarDeflate(ncid, varid_v, true, true, 6);
```

```
netcdf.putAtt( ncid, varid_v, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_v, 'standard_name', 'surface_northward_sea_water_velocity' );
netcdf.putAtt( ncid, varid_v, 'units', 'cm s-1' );

% Variance
varid_evar = netcdf.defVar( ncid, 'evar', 'float', [dimid_lon dimid_lat dimid_t] );
netcdf.defVarDeflate(ncid, varid_evar, true, true, 6);
netcdf.putAtt( ncid, varid_evar, 'long_name', 'radial_sea_water_velocity_variance' );
netcdf.putAtt( ncid, varid_evar, 'units', 'cm s-1' );

% Accuracy
varid_eacc = netcdf.defVar( ncid, 'eacc', 'float', [dimid_lon dimid_lat dimid_t] );
netcdf.defVarDeflate(ncid, varid_eacc, true, true, 6);
netcdf.putAtt( ncid, varid_eacc, 'long_name', 'radial_sea_water_velocity_accuracy' );
netcdf.putAtt( ncid, varid_eacc, 'units', 'cm s-1' );

% X-Distance
varid_xdst = netcdf.defVar( ncid, 'xdst', 'float', [dimid_lon dimid_lat] );
netcdf.defVarDeflate(ncid, varid_xdst, true, true, 6);
netcdf.putAtt( ncid, varid_xdst, 'long_name', 'eastward_distance_from_instrument' );
netcdf.putAtt( ncid, varid_xdst, 'units', 'km' );

% Y-Distance
varid_ydst = netcdf.defVar( ncid, 'ydst', 'float', [dimid_lon dimid_lat] );
netcdf.defVarDeflate(ncid, varid_ydst, true, true, 6);
netcdf.putAtt( ncid, varid_ydst, 'long_name', 'northward_distance_from_instrument' );
netcdf.putAtt( ncid, varid_ydst, 'units', 'km' );

% Exit definition mode
netcdf.endDef( ncid );

%% Write variable data
netcdf.putVar( ncid, varid_lat, lat_dim );
netcdf.putVar( ncid, varid_lon, lon_dim );
netcdf.putVar( ncid, varid_t, 0, round( ( R.time - datenum(1970,1,1) )*86400) );
netcdf.putVar( ncid, varid_bearing, bear );
netcdf.putVar( ncid, varid_range, rnge );
netcdf.putVar( ncid, varid_speed, velo );
netcdf.putVar( ncid, varid_direction, head );
netcdf.putVar( ncid, varid_u, velu );
netcdf.putVar( ncid, varid_v, velv );
netcdf.putVar( ncid, varid_evar, evar );
netcdf.putVar( ncid, varid_eacc, eacc );
netcdf.putVar( ncid, varid_xdst, xdst );
netcdf.putVar( ncid, varid_ydst, ydst );

%% Close file
netcdf.close( ncid );

%% Copyright
% Author: Mark Otero, 2013/08/23
% Copyright: 2013 Coastal Observing Research and Development Center,
%            Scripps Institution of Oceanography
```

Example image from Unidata's Integrated Data Viewer (IDV)

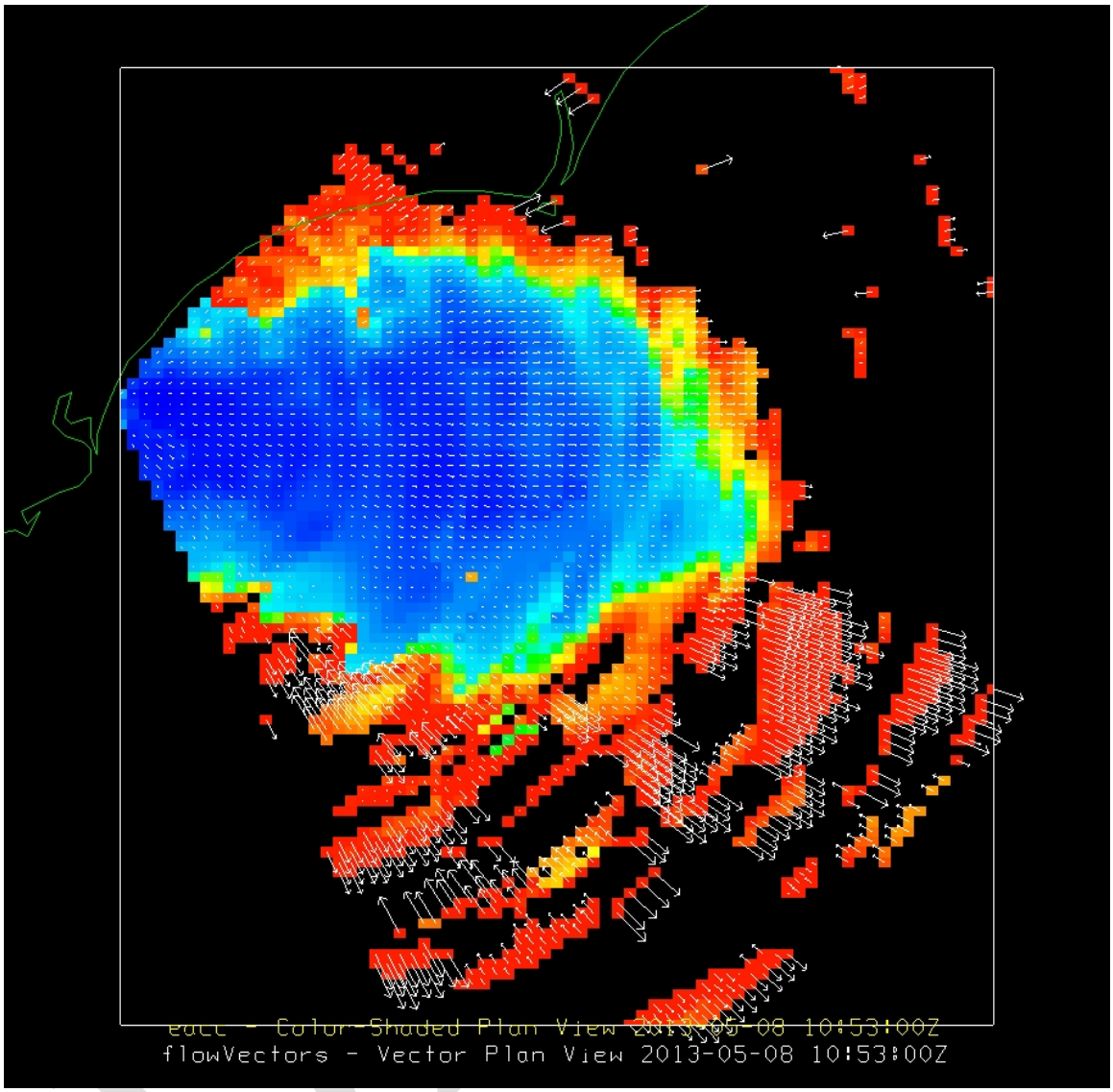

Figure 2. WERA radial velocities plotted over velocity accuracy. Cooler colors indicate higher accuracy (lower error).

Appendix C
LERA NetCDF Encoding

LERA NetCDF header in common data language (CDL)

```
netcdf RDL_UH_KAL_2013_05_08_0400 {
dimensions:
        time = UNLIMITED ; // (1 currently)
        bearing = 361 ;
        range = 61 ;
variables:
        int time(time) ;
```

```
        time:standard_name = "time" ;
        time:units = "seconds since 1970-01-01" ;
        time:calendar = "gregorian" ;
    float bearing(bearing) ;
        bearing:axis = "Y" ;
        bearing:long_name = "bearing_away_from_instrument" ;
        bearing:units = "degrees_true" ;
    float range(range) ;
        range:axis = "X" ;
        range:long_name = "range_away_from_instrument" ;
        range:units = "km" ;
    float lat(bearing, range) ;
        lat:standard_name = "latitude" ;
        lat:units = "degrees_north" ;
    float lon(bearing, range) ;
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
    float speed(time, bearing, range) ;
        speed:valid_range = -1000.f, 1000.f ;
        speed:standard_name = "radial_sea_water_velocity_away_from_instrument" ;
        speed:units = "cm s-1" ;
        speed:coordinates = "lon lat" ;
    short direction(time, bearing, range) ;
        direction:valid_range = 0s, 3600s ;
        direction:standard_name = "direction_of_radial_vector_away_from_instrument" ;
        direction:units = "degrees_true" ;
        direction:coordinates = "lon lat" ;
        direction:scale_factor = 0.1f ;
    float u(time, bearing, range) ;
        u:valid_range = -1000.f, 1000.f ;
        u:standard_name = "surface_eastward_sea_water_velocity" ;
        u:units = "cm s-1" ;
        u:coordinates = "lon lat" ;
    float v(time, bearing, range) ;
        v:valid_range = -1000.f, 1000.f ;
        v:standard_name = "surface_northward_sea_water_velocity" ;
        v:units = "cm s-1" ;
        v:coordinates = "lon lat" ;
    float eacc(time, bearing, range) ;
        eacc:long_name = "radial_sea_water_velocity_accuracy" ;
        eacc:units = "cm s-1" ;
        eacc:coordinates = "lon lat" ;
    float xdst(bearing, range) ;
        xdst:long_name = "eastward_distance_from_instrument" ;
        xdst:units = "km" ;
        xdst:coordinates = "lon lat" ;
    float ydst(bearing, range) ;
        ydst:long_name = "northward_distance_from_instrument" ;
        ydst:units = "km" ;
        ydst:coordinates = "lon lat" ;

// global attributes:
        :netcdf_library_version = "4.1.3" ;
        :Conventions = "CF-1.6" ;
        :title = "Near-Real Time Surface Ocean Radial Velocity" ;
        :institution = "SIO" ;
```

```
        :source = "Surface Ocean HF-Radar" ;
        :history = "23-Aug-2013 08:39:15: NetCDF file created" ;
        :references = "WERA Radial LonLatUV (LLUV) File Format" ;
        :creator_name = "HF-Radar Network Administrators" ;
        :creator_email = "hfrnetadm@ucsd.edu" ;
        :creator_url = "http://cordc.ucsd.edu/projects/mapping/" ;
        :summary = "HF-RADAR measurements of ocean velocity are radial in\n",
            "direction relative to the radar location and representitive \n",
            "of the upper 0.3 - 2.5 meters of the ocean." ;
        :geospatial_lat_min = 20.48137f ;
        :geospatial_lat_max = 22.11363f ;
        :geospatial_lon_min = -158.9595f ;
        :geospatial_lon_max = -157.2077f ;
        :CTF = "1.00" ;
        :FileType = "LLUV rdls" ;
        :Manufacturer = "University of Hawaii SOEST (WERA)" ;
        :LLUVSpec = "1.00 2007 12 06" ;
        :Network = "UH" ;
        :Site = "kal Kalaeloa" ;
        :TimeStamp = "2013 05 08 04 00 00" ;
        :TimeZone = "UTC +0.00 0" ;
        :Origin = "21.2975006  -158.0836029" ;
        :GreatCircle = "WGS84 6378137.000 298.257223562997" ;
        :GeodVersion = "Vincenty (1979) 2.0 2002 10 01" ;
        :RangeResolutionKMeters = "1.500" ;
        :TransmitCenterFreqMHz = "16.05" ;
        :DopplerResolutionHzPerBin = "0.0015" ;
        :CurrentVelocityLimit = "100.0" ;
        :MergedCount = "0001" ;
        :TransmitSweepRateHz = "0.333334" ;
        :TransmitBandwidthKHz = "100.0" ;
        :SpectraDopplerCells = "02048" ;
        :TableType = "LLUV RDL1" ;
        :TableColumns = "11" ;
        :TableColumnTypes = "LOND LATD VELU VELV EACC XDST YDST RNGE BEAR \n",
            "VELO HEAD" ;
        :TableRows = "5638" ;
}
```

Sample MATLAB code for encoding LERA data in NetCDF from a source ASCII file

```
% Development script to format HF-Radar radial files in netCDF.

%% Load sample data
rfile_path = '/home/motero/Documents/Projects/HFRNet/Development/NetCDF Radial/Sample Files/';
rfile = fullfile( rfile_path, 'RDL_UH_KAL_2013_05_08_0400.hfrweralluv1.0');
r = load( rfile );
R = struct( ...
   'lond', r(:, 1), ...
   'latd', r(:, 2), ...
   'velu', r(:, 3), ...
   'velv', r(:, 4), ...
   'eacc', r(:, 5), ...
   'xdst', r(:, 6), ...
```

```
    'ydst', r(:, 7), ...
    'rnge', r(:, 8), ...
    'bear', r(:, 9), ...
    'velo', r(:,10), ...
    'head', r(:,11)  ...
);
clear r;

%TimeStamp: 2013 05 08 04 00 00
R.time = datenum(2013,5,8,4,0,0);

%% Generate grid domain
% May want to have clipped to samples w/in a given file or possibly extend
% the full 0-360 range and some nominal range for the given frequency.  The
% latter would work better for aggregation (assuming bearing angles don't
% change over the aggregation period).

%RangeResolutionKMeters:  1.500
range_dim = min( R.rnge ) : 1.5 : max( R.rnge );

%AngularResolution - inferred from bearing data
bearing_dim = 0 : 1 : 360;

[ bearing, range ] = meshgrid( bearing_dim, range_dim );

% check - see that our grid captures the sample space right
% figure
% plot( range, bearing, 'go' )
% hold on
% plot( R.rnge, R.bear, 'b.')


% Generate lat/lon from bearing & range
% This is the most accurate method of defining lat/lon values for given
% range & bearing.  Other methods of interpolating and extrapolating using
% griddata or otherwise from range, bearing, lon and lat are inconsistent
% and generally not reliable from file to file. The reason we need to do
% this at all is because there cannot be missing values in auxillary
% coordinate variables.

%Origin:    21.2975006  -158.0836029
[latd, lond] = reckon( 21.2975006, -158.0836029, km2deg(range), bearing );

% Check
% figure
% plot(lond, latd, 'ko')
% hold on
% plot(R.lond, R.latd, 'r.')
% daspect([1.2 1 1])


%% Determine the mapping index from sample data to full grid
range_map_idx   = nan( 1, numel(R.rnge) );
bearing_map_idx = nan( 1, numel(R.bear) );
for I = 1 : numel(R.rnge)
    [~, range_map_idx(I)]   = min( abs( range_dim - R.rnge(I) ) );
```

```
    [~, bearing_map_idx(I)] = min( abs( bearing_dim - R.bear(I) ) );
end
clear I;
rb_map_idx = sub2ind( size( range ), range_map_idx, bearing_map_idx );


%% Populate gridded matrices

% Eastward Velocity
velu = nan( size( range ) );
velu(rb_map_idx) = R.velu;

% Northward Velocity
velv = nan( size( range ) );
velv(rb_map_idx) = R.velv;

% check
% figure
% quiver(R.rnge, R.bear, R.velu, R.velv, 0, 'k')
% hold on
% quiver(range, bearing, velu, velv, 0, 'r')
%
% figure
% quiver(R.lond, R.latd, R.velu./100, R.velv./100, 0, 'k')
% daspect([1.2 1 1])
% hold on
% quiver(lond, latd, velu./100, velv./100, 0, 'r')

% Current Speed - flip sign so positive velocity is away from radar
% according to CF standard name
% 'radial_sea_water_velocity_away_from_instrument'. CODAR reports positive
% speeds as toward the radar.
velo = nan( size( range ) );
velo(rb_map_idx) = -R.velo;

% Current Heading
head = nan( size( range ) );
head(rb_map_idx) = R.head;

% Accuracy
eacc = nan( size( range ) );
eacc(rb_map_idx) = R.eacc;

% X-Distance
xdst = nan( size( range ) );
xdst(rb_map_idx) = R.xdst;

% Y-Distance
ydst = nan( size( range ) );
ydst(rb_map_idx) = R.ydst;


%% Scaling & Fill Values
% Bearing and heading are only reported to 10ths and can be
% reported as short unsigned integers when scaled. However, Bearing is a
% coordinate variable and cannot be scaled by CF metadata convention.
```

```
% Also, can only use unsigned when using NetCDF-4 enhansed data model,
% otherwise w/classic data model you're limited ot signed data types.
head = round( head.*10 );

% Type short fill values
head( isnan(head) ) = netcdf.getConstant('NC_FILL_SHORT');

% Type float fill values
velo( isnan(velo) ) = netcdf.getConstant('NC_FILL_FLOAT');
velu( isnan(velu) ) = netcdf.getConstant('NC_FILL_FLOAT');
velv( isnan(velv) ) = netcdf.getConstant('NC_FILL_FLOAT');
eacc( isnan(eacc) ) = netcdf.getConstant('NC_FILL_FLOAT');
xdst( isnan(xdst) ) = netcdf.getConstant('NC_FILL_FLOAT');
ydst( isnan(ydst) ) = netcdf.getConstant('NC_FILL_FLOAT');



%% Define dimensions, variables and attributes
ncfile = strrep( rfile, 'hfrweralluv1.0', 'nc' );

% Keep the classic data model to increase compatibility since we
% aren't using features specific to the NetCDF-4.0 model.
mode = netcdf.getConstant( 'NETCDF4' );
mode = bitor(mode, netcdf.getConstant( 'CLASSIC_MODEL' ) );

ncid = netcdf.create( ncfile, mode );
hist_create = [datestr(now) ': NetCDF file created'];

% Add dimensions (in order of T, Z, Y, X for CF/COARDS compliance)
dimid_t       = netcdf.defDim( ncid, 'time',    netcdf.getConstant('unlimited') );
dimid_bearing = netcdf.defDim( ncid, 'bearing', numel( bearing_dim ) );
dimid_range   = netcdf.defDim( ncid, 'range',   numel( range_dim ) );

% Add coordinate variables and attributes in the same order as
% the dimensions were written

% For time coordinate variable, when using the standard or gregorian
% calander, units should be in seconds and shifted forward beyond
% 1582/10/15 to avoid Julian -> Gregorian crossover and potential
% problems with the udunits package, if used.  Year length should be
% exactly 365.2425 days long for the Gregorian/standard and proleptic
% Gregorian calander.
%
% MATLAB's datenum uses the proleptic gregorian calander as defined
% by the CF standard.  When using standard_name attribute for time
% coordinate be sure to include the calander and units attributes.
%
% Use int for time data type because need 8 significant figures to
% represent number of seconds in 1 year (31556952 seconds), float
% isn't enough (7 sig. figures).  Would need to use double beyond
% int.  Besides, don't need sub-second accuracy.  Unix time will
% overflow int in 2038.  New epoch will be needed then to keep int
% representation.  Unix time epoch is:
%           1970-01-01 00:00:00Z = 0 seconds, 86,400 sec/day
varid_t = netcdf.defVar( ncid, 'time', 'int', dimid_t );
netcdf.putAtt( ncid, varid_t, 'standard_name', 'time' );
```

```
netcdf.putAtt( ncid, varid_t, 'units', 'seconds since 1970-01-01' );
netcdf.putAtt( ncid, varid_t, 'calendar', 'gregorian' );

% Bearing (arbitrary 'y' dimension)
varid_bearing = netcdf.defVar( ncid, 'bearing', 'float', dimid_bearing );
netcdf.putAtt( ncid, varid_bearing, 'axis', 'Y' );
netcdf.putAtt( ncid, varid_bearing, 'long_name', 'bearing_away_from_instrument' );
netcdf.putAtt( ncid, varid_bearing, 'units', 'degrees_true' );

% Range (arbitrary 'x' dimension)
varid_range = netcdf.defVar( ncid, 'range', 'float', dimid_range );
netcdf.putAtt( ncid, varid_range, 'axis', 'X' );
netcdf.putAtt( ncid, varid_range, 'long_name', 'range_away_from_instrument' );
netcdf.putAtt( ncid, varid_range, 'units', 'km' );


% Add global attributes

% Currently using Climate & Forecast Metadata Conventions v1.6
% (CF).  Am also using Attribute Convention for Dataset Discovery
% v1.1 (ACDD) where applicable along with
% a few of my own attributes (X) for things that didn't fit
% well or weren't clearly applicable to features in either
% convention.  The ACDD convention is still developing and may
% eventually support the attributes added.

varid_global = netcdf.getConstant('global');

% (X) NetCDF library version used
netcdf.putAtt( ncid, varid_global, 'netcdf_library_version', netcdf.inqLibVers );

% (CF)  Conventions
netcdf.putAtt( ncid, varid_global, 'Conventions', 'CF-1.6' );

% (CF, ACDD) title
netcdf.putAtt( ncid, varid_global, 'title', 'Near-Real Time Surface Ocean Radial Velocity' );

% (CF, ACDD) institution
netcdf.putAtt( ncid, varid_global, 'institution', 'SIO' );

% (CF)  source
netcdf.putAtt( ncid, varid_global, 'source', 'Surface Ocean HF-Radar' );

% (CF, ACDD) history
netcdf.putAtt( ncid, varid_global, 'history', hist_create );

% (CF)  references
netcdf.putAtt( ncid, varid_global, 'references', 'WERA Radial LonLatUV (LLUV) File Format' );

% (ACDD)        creator_name
netcdf.putAtt( ncid, varid_global, 'creator_name', 'HF-Radar Network Administrators' );

% (ACDD)        creator_email
netcdf.putAtt( ncid, varid_global, 'creator_email', 'hfrnetadm@ucsd.edu' );

% (ACDD)        creator_url
```

```
netcdf.putAtt( ncid, varid_global, 'creator_url', 'http://cordc.ucsd.edu/projects/mapping/' );

% (ACDD)          summary
nc_summary = sprintf( '%s\n%s\n%s', ...
   'HF-RADAR measurements of ocean velocity are radial in', ...
   'direction relative to the radar location and representitive ', ...
   'of the upper 0.3 - 2.5 meters of the ocean.' );
netcdf.putAtt( ncid, varid_global, 'summary', nc_summary);

% (ACDD)          geospatial_[lon|lat]_[max|min]
% For totals, this is computed from the grid, not from actual data
% availability limits. Analogous to remote sensing where availability is
% relative to pass, not necessairly where data is. However, unless we want
% to compute lat/lon for range/bearing where no sample is available, just
% report lat/lon limit of data availability.
netcdf.putAtt( ncid, varid_global, 'geospatial_lat_min', single(min(min(latd))) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lat_max', single(max(max(latd))) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lon_min', single(min(min(lond))) );
netcdf.putAtt( ncid, varid_global, 'geospatial_lon_max', single(max(max(lond))) );

% Globals sourced from radial file metadata
netcdf.putAtt( ncid, varid_global, 'CTF', '1.00');
netcdf.putAtt( ncid, varid_global, 'FileType', 'LLUV rdls' );
netcdf.putAtt( ncid, varid_global, 'Manufacturer', 'University of Hawaii SOEST (WERA)' );
netcdf.putAtt( ncid, varid_global, 'LLUVSpec', '1.00 2007 12 06' );
netcdf.putAtt( ncid, varid_global, 'Network', 'UH' ); %inserted by HFRNet
netcdf.putAtt( ncid, varid_global, 'Site', 'kal Kalaeloa' );
netcdf.putAtt( ncid, varid_global, 'TimeStamp', '2013 05 08 04 00 00' );
netcdf.putAtt( ncid, varid_global, 'TimeZone', 'UTC +0.00 0' );
netcdf.putAtt( ncid, varid_global, 'Origin', '21.2975006  -158.0836029' );
netcdf.putAtt( ncid, varid_global, 'GreatCircle', 'WGS84 6378137.000 298.257223562997' );
netcdf.putAtt( ncid, varid_global, 'GeodVersion', 'Vincenty (1979) 2.0 2002 10 01' );
netcdf.putAtt( ncid, varid_global, 'RangeResolutionKMeters', '1.500' );
netcdf.putAtt( ncid, varid_global, 'TransmitCenterFreqMHz', '16.05' );
netcdf.putAtt( ncid, varid_global, 'DopplerResolutionHzPerBin', '0.0015' );
netcdf.putAtt( ncid, varid_global, 'CurrentVelocityLimit', '100.0' );
netcdf.putAtt( ncid, varid_global, 'MergedCount', '0001' );
netcdf.putAtt( ncid, varid_global, 'TransmitSweepRateHz', '0.333334' );
netcdf.putAtt( ncid, varid_global, 'TransmitBandwidthKHz', '100.0' );
netcdf.putAtt( ncid, varid_global, 'SpectraDopplerCells', '02048' );
netcdf.putAtt( ncid, varid_global, 'TableType', 'LLUV RDL1' );
netcdf.putAtt( ncid, varid_global, 'TableColumns', '11' );
netcdf.putAtt( ncid, varid_global, 'TableColumnTypes', sprintf('%s\n%s', ...
   'LOND LATD VELU VELV EACC XDST YDST RNGE BEAR ', ...
   'VELO HEAD' ) );
netcdf.putAtt( ncid, varid_global, 'TableRows', '5638' );

% Add auxillary coordinate variables to provide mapping from range &
% bearing to lat, lon.

% A minimum of 4 significant figures to the right of the decimal
% place is needed to keep resolution below 10's of meters.  Using
% float data type for lat yields at least 5 significant digits to
% the right of the decimal giving ~1/2m resolution in latitude.
% Nine significant figures (at least 7 to the right of the
% decimal) could be achieved using int data type but need to
```

```
% introduce a scale factor.
%
% Latitude
varid_lat = netcdf.defVar( ncid, 'lat', 'float', [dimid_range dimid_bearing] );
netcdf.defVarDeflate(ncid, varid_lat, true, true, 6);
netcdf.putAtt( ncid, varid_lat, 'standard_name', 'latitude' );
netcdf.putAtt( ncid, varid_lat, 'units', 'degrees_north' );

% Longitude
varid_lon = netcdf.defVar( ncid, 'lon', 'float', [dimid_range dimid_bearing] );
netcdf.defVarDeflate(ncid, varid_lon, true, true, 6);
netcdf.putAtt( ncid, varid_lon, 'standard_name', 'longitude' );
netcdf.putAtt( ncid, varid_lon, 'units', 'degrees_east' );


% Add Data Variables

% radial_sea_water_velocity_away_from_instrument:
% A velocity is a vector quantity. Radial velocity away from instrument
% means the component of the velocity along the line of sight of the
% instrument where positive implies movement away from the instrument (i.e.
% outward). The "instrument" (examples are radar and lidar) is the device
% used to make an observation.
varid_speed = netcdf.defVar( ncid, 'speed', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_speed, true, true, 6);
netcdf.putAtt( ncid, varid_speed, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_speed, 'standard_name', 'radial_sea_water_velocity_away_from_instrument' );
netcdf.putAtt( ncid, varid_speed, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_speed, 'coordinates', 'lon lat' );

% (radial current) direction
%
% direction_of_radial_vector_away_from_instrument:
% The direction_of_radial_vector_away_from_instrument is the direction in
% which the instrument itself is pointing. The direction is measured
% positive clockwise from due north. The "instrument" (examples are radar
% and lidar) is the device used to make an observation. "direction_of_X"
% means direction of a vector, a bearing.
varid_direction = netcdf.defVar( ncid, 'direction', 'short', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_direction, true, true, 6);
netcdf.putAtt( ncid, varid_direction, 'valid_range', int16( [0 3600] ) );
netcdf.putAtt( ncid, varid_direction, 'standard_name', 'direction_of_radial_vector_away_from_instrument'
);
netcdf.putAtt( ncid, varid_direction, 'units', 'degrees_true' );
netcdf.putAtt( ncid, varid_direction, 'coordinates', 'lon lat' );
netcdf.putAtt( ncid, varid_direction, 'scale_factor', single(0.1) );

% u
varid_u = netcdf.defVar( ncid, 'u', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_u, true, true, 6);
netcdf.putAtt( ncid, varid_u, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_u, 'standard_name', 'surface_eastward_sea_water_velocity' );
netcdf.putAtt( ncid, varid_u, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_u, 'coordinates', 'lon lat' );

% v
```

```
varid_v = netcdf.defVar( ncid, 'v', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_v, true, true, 6);
netcdf.putAtt( ncid, varid_v, 'valid_range', single( [-1e3 1e3] ));
netcdf.putAtt( ncid, varid_v, 'standard_name', 'surface_northward_sea_water_velocity' );
netcdf.putAtt( ncid, varid_v, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_v, 'coordinates', 'lon lat' );

% Accuracy
varid_eacc = netcdf.defVar( ncid, 'eacc', 'float', [dimid_range dimid_bearing dimid_t] );
netcdf.defVarDeflate(ncid, varid_eacc, true, true, 6);
netcdf.putAtt( ncid, varid_eacc, 'long_name', 'radial_sea_water_velocity_accuracy' );
netcdf.putAtt( ncid, varid_eacc, 'units', 'cm s-1' );
netcdf.putAtt( ncid, varid_eacc, 'coordinates', 'lon lat' );

% X-Distance
varid_xdst = netcdf.defVar( ncid, 'xdst', 'float', [dimid_range dimid_bearing] );
netcdf.defVarDeflate(ncid, varid_xdst, true, true, 6);
netcdf.putAtt( ncid, varid_xdst, 'long_name', 'eastward_distance_from_instrument' );
netcdf.putAtt( ncid, varid_xdst, 'units', 'km' );
netcdf.putAtt( ncid, varid_xdst, 'coordinates', 'lon lat' );

% Y-Distance
varid_ydst = netcdf.defVar( ncid, 'ydst', 'float', [dimid_range dimid_bearing] );
netcdf.defVarDeflate(ncid, varid_ydst, true, true, 6);
netcdf.putAtt( ncid, varid_ydst, 'long_name', 'northward_distance_from_instrument' );
netcdf.putAtt( ncid, varid_ydst, 'units', 'km' );
netcdf.putAtt( ncid, varid_ydst, 'coordinates', 'lon lat' );

% Exit definition mode
netcdf.endDef( ncid );

%% Write variable data
netcdf.putVar( ncid, varid_bearing, bearing_dim );
netcdf.putVar( ncid, varid_range, range_dim );
netcdf.putVar( ncid, varid_t, 0, round( ( R.time - datenum(1970,1,1) )*86400) );
netcdf.putVar( ncid, varid_lat, latd );
netcdf.putVar( ncid, varid_lon, lond );
netcdf.putVar( ncid, varid_speed, velo );
netcdf.putVar( ncid, varid_direction, head );
netcdf.putVar( ncid, varid_u, velu );
netcdf.putVar( ncid, varid_v, velv );
netcdf.putVar( ncid, varid_eacc, eacc );
netcdf.putVar( ncid, varid_xdst, xdst );
netcdf.putVar( ncid, varid_ydst, ydst );

%% Close file
netcdf.close( ncid );

%% Copyright
% Author: Mark Otero, 2013/08/23
% Copyright: 2013 Coastal Observing Research and Development Center,
%            Scripps Institution of Oceanography
```

Example image from Unidata's Integrated Data Viewer (IDV)

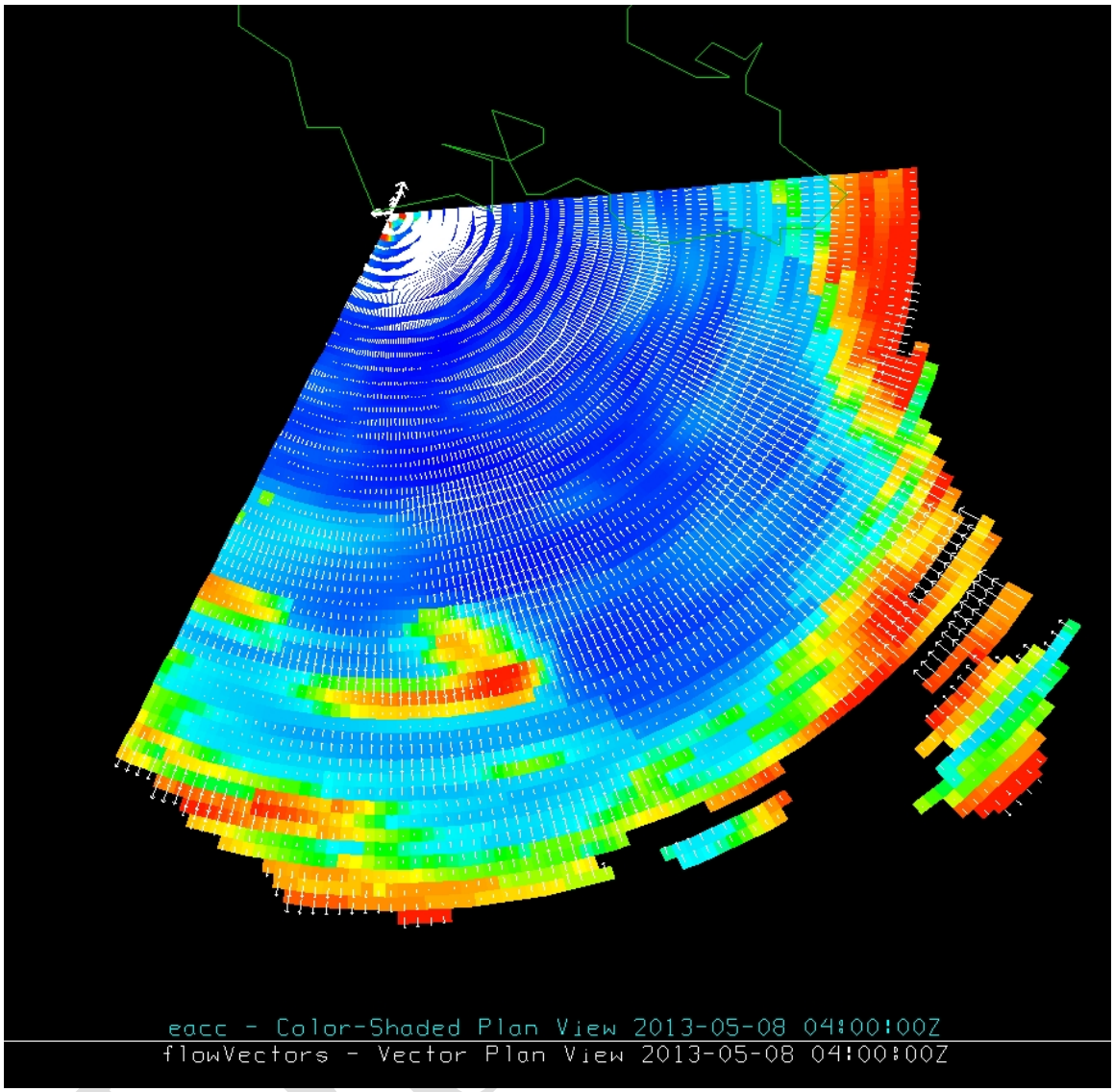

Figure 3. LERA radial velocities plotted over velocity accuracy. Cooler colors indicate higher accuracy (lower error).