

Producing an Animation with the BCS Universal File Interface (UFI), the Informix Spatial DataBlade, and gnuplot

Contents

Producing an Animation with the BCS Universal File Interface (UFI), the Informix Spatial DataBlade, and gnuplot.....	1
Introduction.....	2
The Steps in Brief	3
Getting the Data	3
Converting the GRIB data to NetCDF.....	4
Using UFI to Access the NetCDF Files.....	4
Create the Template Tables	6
Create the Virtual Tables	6
Perform the Query.....	9
For more information.....	11
Appendix – One of the NetCDF Files.....	12

Introduction

Many “technologists” (e.g., scientists, engineers, and other researchers) regularly deal with quite large data files that they consider too cumbersome/difficult/slow/costly or too transient in nature to warrant loading into a database. Yet they typically want to extract information from these files, and they sometimes have a requirement to analyze these files in the context of other data that might reside in a database. To carry out these analyses they may have to write fairly complex code to read in all these files, extract other data from databases, and so on.

At BCS we have developed another approach – a solution that avoids technologists having to write complex, custom applications to analyze their files; instead they can use much simpler and more powerful SQL without loading their files into a database!

The solution is the *Universal File Interface (UFI)*, a database extension based on the IBM Informix Virtual Table Interface (VTI). VTI is a technology that supports making external datasets appear as tables to SQL queries and statements¹. UFI is a BCS database extension for delivering the contents of external data files as though they were rows in a database table. UFI makes a file look like a set of database tables, so “UFI-managed tables” are actually *virtual* database tables. Thus, users are able to perform analyses, using SQL, between these files and database objects as if the files had actually been loaded into the database.

In order to demonstrate how UFI works in concert with other database elements we’ve set up a process to generate and display an animated GIF image of weather forecast information for three areas:

- 1) the area around Vancouver Island and the Puget Sound,
- 2) the (newly named) Salish Sea, and
- 3) the Great Lakes region

These animated GIFs are updated twice daily from data that we download from [Environment Canada](#).

This document describes the steps involved in producing the Great Lakes Region animated GIF and, in particular, explains how the BCS Universal File Interface (UFI) is used in the process. (Analogous steps were employed for the other two regions.)

¹ It addresses the same needs as the SQL/MED, or Management of External Data, extension to the SQL standard as defined by ISO/IEC 9075-9:2003.

The Steps in Brief

- 1) Download the data from Environment Canada. This data is in the form of GRIB (GRIdded Binary) files.
- 2) Convert the GRIB files to NetCDF.
- 3) Define the UFI tables.
- 4) Perform the queries to generate the GIF files.
- 5) Combine the GIF files into an animated GIF and update our website.

Getting the Data

The first step is to download the latest forecast data from the Environment Canada website http://dd.weatheroffice.ec.gc.ca/model_gem_regional/high_resolution/grib2/00.

This website/directory contains gridded weather forecast data produced using a [GEM Regional Model](#) and a [regional polar stereographic grid](#). The directory contains data for a multitude of parameters and for each of 17 timesteps (every 3 hours from the time of the forecast until 48 hours in the future). Each file contains data for one parameter and one timestep. The two parameters that we are interested in are the forecast U and V components of wind velocity at a height of 10 meters above ground. From these two parameters we can determine the forecast wind speed. Here is a portion of the C shell script we use to download the 34 (2x17) files we need:

```
foreach t (000 003 006 009 012 015 018 021 024 027 030 033 036 039 042 045 048)
  foreach dir (U V)
    wget --tries=5 http://dd.weatheroffice.ec.gc.ca/
    model\_gem\_regional/high\_resolution/grib2/00/
    CMC\_reg\_\${dir}GRD\_TGL\_10\_ps15km\_\${DATE}\_P\${t}.grib2
  end
end
```

The `_${DATE}` string parameter in these commands represents the year, month, day, and hour when the forecast was made; this string forms part of the filename.

Note that the `wget` argument has been split onto multiple lines for readability; in the actual script there is no such split.

Note also that we allow up to 5 attempts for each file, since the `wget` connection can sometimes time out.

Converting the GRIB data to NetCDF

We will be using UFI to read the weather forecast data files, which we've just downloaded in [GRIB](#) format. Although UFI is able to read GRIB files (through its GDAL adapter), the GRIB files are in a gridded polar stereographic representation and we will require the data to be in geographic coordinates. Hence, we convert each of the 34 GRIB files to [NetCDF](#) using the [National Center for Atmospheric Research](#) (NCAR) program [ncl_convert2nc](#).

```
setenv NCARG_ROOT /usr/local
foreach t (000 003 006 009 012 015 018 021 024 027 030 033 036 039 042 045 048)
  foreach dir (U V)
    ncl_convert2nc CMC_reg_${dir}GRD_TGL_10_ps15km_${DATE}_P${t}.grib2
  end
end
end
```

Using UFI to Access the NetCDF Files

A partial listing of one of the NetCDF files produced is provided in the [Appendix](#).

The following points about these files should be noted:

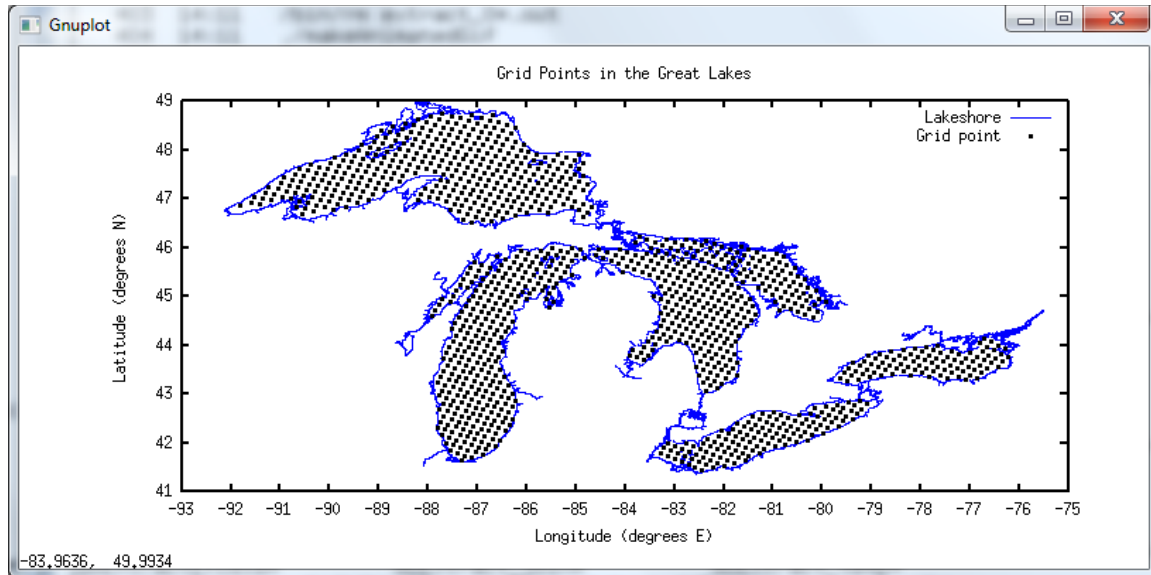
- 1) each file pertains to just one of the two parameters and a single one of the 17 forecast times, meaning that we somehow have to consolidate 34 (2x17) files in order to do our analysis, and
- 2) each file contains forecasts for much of North America, an area much larger than our particular area of interest (i.e., the Great Lakes).

The following image, which appears on the Environment Canada [website](#) listed above, illustrates the second point.



Each NetCDF file is 3.2 MB in size and contains data for each of 199,899 grid points (399x501), covering the area shown above.

However, the number of grid points that lie in one of the Great Lakes is just 1264, as shown in the following diagram²:



The Universal File Interface provides an easy means of consolidating multiple files (34, in this case) and zeroing in on a region of interest.

UFI is used to map each of the 34 NetCDF files onto its own virtual table³. This is accomplished using the sequence of UFI / SQL commands described in the following sections:

² The shoreline geometry for the Great Lakes waterbodies was taken from the website [Maps and GIS of the Great Lakes Region](http://gis.glin.net/geoserver/wfs?request=getfeature&service=wfs&version=1.0.0&typename=glin:glgis_gl_shore_noaa_70k&outputformat=shape-zip), specifically http://gis.glin.net/geoserver/wfs?request=getfeature&service=wfs&version=1.0.0&typename=glin:glgis_gl_shore_noaa_70k&outputformat=shape-zip

³ Note that with UFI it would be possible to map all 17 u files onto a single UFI virtual table and all 17 v files onto a second UFI virtual table. These virtual tables would have an additional column (representing the forecast time), mapped from the `forecast_time` attribute of the `10u_p0_L103_GST0` (or `10v_p0_L103_GST0`) variable. However, it's more efficient to "store" each forecast time in its own table (since there will be fewer rows to search, and in each query we're interested in data for a single forecast time).

Create the Template Tables

The template tables are real database tables whose columns and column types reflect the columns and column types that will appear in our UFI virtual tables. The contents don't matter (in this application the template tables will remain empty), but the structure tells UFI what the virtual tables are going to look like. We create a *u* template table and a *v* template table for each of the 17 forecast times:

```
> CREATE TABLE windsu_template_0004(
  ix INTEGER,
  iy INTEGER,
  rot DOUBLE PRECISION,
  lon DOUBLE PRECISION,
  lat DOUBLE PRECISION,
  u DOUBLE PRECISION
);

> CREATE TABLE windsv_template_000(
  ix INTEGER,
  iy INTEGER,
  rot DOUBLE PRECISION,
  lon DOUBLE PRECISION,
  lat DOUBLE PRECISION,
  v DOUBLE PRECISION
);
```

Create the Virtual Tables

The next step is to create a UFI virtual *u* table and a virtual *v* table for each of the time steps. The label 'netcdf' tells UFI to use its external program designed for handling NetCDF files.

```
> EXECUTE PROCEDURE ufi_make_managed('windsu_000', 'netcdf',
  'windsu_template_000');
> EXECUTE PROCEDURE ufi_make_managed('windsv_000', 'netcdf',
  'windsv_template_000');
```

Next we must tell UFI where to get the data for each of the columns in the *u* and *v* virtual tables.

Table	Column	NetCDF variable	Comments
windsu_* windsv_*	ix	xgrid_0	x index position of this datum.
windsu_* windsv_*	iy	ygrid_0	y index position of this datum.
windsu_* windsv_*	lon	gridlon_0	The longitude corresponding to this datum (i.e., at the (ix, iy) grid point).
windsu_* windsv_*	lat	gridlat_0	The longitude corresponding to this datum.
windsu_*	rot	gridrot_0	The rotation angle to apply at this point

⁴ "000" is the first forecast time, subsequent values as "003", "006", ..., "048".

windsv_*			in the grid in order to determine the velocity relative to the earth ⁵ .
windsu_*	u	10u_P0_L103_GST0	The u component of the wind velocity at lat(ix,iy),lon(ix,iy).
windsv_*	v	10v_P0_L103_GST0	The v component of the wind velocity at lat(ix,iy),lon(ix,iy).

The following commands perform this mapping:

```
> EXECUTE PROCEDURE ufi_add_column('windsu_000', 'lon', 'gridlon_0');
> EXECUTE PROCEDURE ufi_add_column('windsu_000', 'lat', 'gridlat_0');
> EXECUTE PROCEDURE ufi_add_column('windsu_000', 'u',
    '10u_P0_L103_GST0');
> EXECUTE PROCEDURE ufi_add_column('windsu_000', 'ix', 'xgrid_0');
> EXECUTE PROCEDURE ufi_add_column('windsu_000', 'iy', 'ygrid_0');
> EXECUTE PROCEDURE ufi_add_column('windsu_000', 'rot', 'gridrot_0');

> EXECUTE PROCEDURE ufi_add_column('windsv_000', 'lon', 'gridlon_0');
> EXECUTE PROCEDURE ufi_add_column('windsv_000', 'lat', 'gridlat_0');
> EXECUTE PROCEDURE ufi_add_column('windsv_000', 'v',
    '10v_P0_L103_GST0');
> EXECUTE PROCEDURE ufi_add_column('windsv_000', 'ix', 'xgrid_0');
> EXECUTE PROCEDURE ufi_add_column('windsv_000', 'iy', 'ygrid_0');
> EXECUTE PROCEDURE ufi_add_column('windsv_000', 'rot', 'gridrot_0');

> EXECUTE PROCEDURE ufi_add_file('windsu_000', 'windsu_000',
    '/opt/data/CdnMetService/CMC_reg_UGRD_TGL_10_ps15km_${DATE}_P000.nc');
> EXECUTE PROCEDURE ufi_add_file('windsv_000', 'windsv_000',
    '/opt/data/CdnMetService/CMC_reg_VGRD_TGL_10_ps15km_${DATE}_P000.nc');

> EXECUTE PROCEDURE ufi_validate('windsu_000');
> EXECUTE PROCEDURE ufi_validate('windsv_000');

> SELECT count(*) FROM windsu_000;

      (count(*))

      199899

1 row(s) retrieved.
```

(As expected, 399 longitude x 501 latitude = 199899 values are selected.)

Since we're interested in wind values inside a Great Lake, we make use of lake shoreline polygons that have been stored inside an Informix database using the Spatial DataBlade.

```
> CREATE TABLE lakes(shoreline st_polygon, lakename VARCHAR(100));
```

⁵ Specifically,

$$\begin{aligned} V_{Earth} &= \cos(\text{rot}) * V_{grid} - \sin(\text{rot}) * U_{grid} \\ U_{Earth} &= \sin(\text{rot}) * V_{grid} + \cos(\text{rot}) * U_{grid} \end{aligned}$$

We can then isolate the grid points that are inside one of the lakes using the following simple SQL, which references the shoreline geometry in the `lakes` table and the grid points in the `UFI_windsu_000` table created earlier.

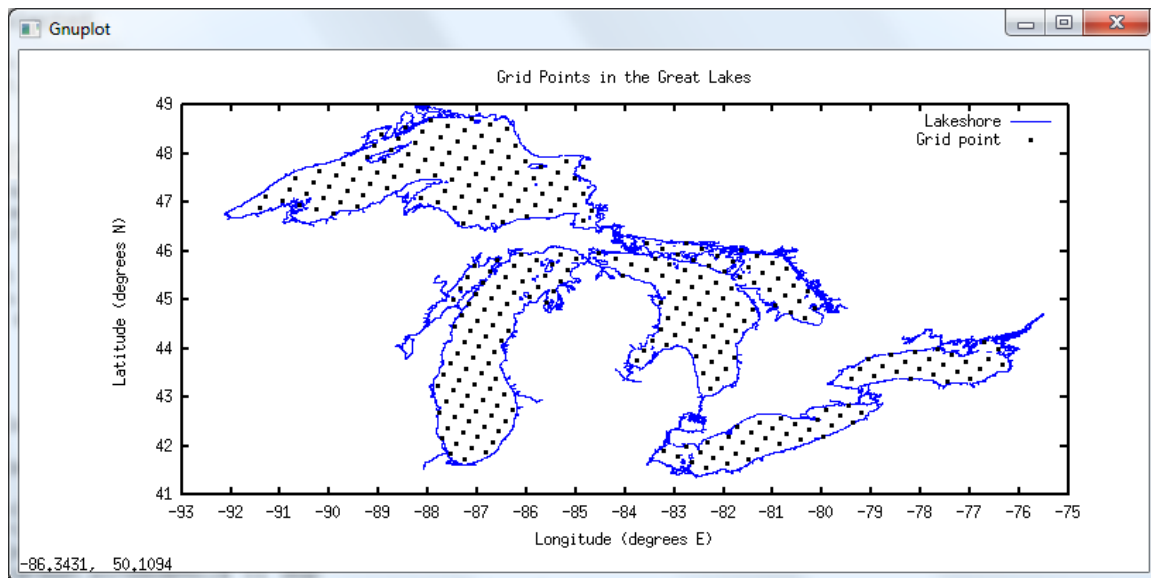
```
> UNLOAD TO gridusedpointinlake.plot DELIMITER ' '
SELECT {+ORDERED, FULL(U)} u.lon,u.lat
FROM windsu_000 u, lakes b WHERE
    lakename IN ('Lake Ontario','Lake Erie','Lake Huron',
                'Lake Superior','Lake Michigan') AND
    ST_Within(('5 POINT(' || u.lon || ' ' || u.lat || ')')::ST_Point,
             B.shoreline) AND
    u.lon BETWEEN -93 AND -75 AND u.lat BETWEEN 40 AND 50 AND
    MOD(u.ix,2) = 0 AND MOD(u.iy,2) = 0;
```

318 row(s) unloaded.

For efficiency, we apply the `ST_WITHIN` test to just those grid points that are known to lie inside a bounding rectangle of the Great Lakes region (i.e., points having a latitude between 40 and 50 degrees N and a longitude between -93 and -75 degrees E). UFI can use the NetCDF dimension variables to very efficiently apply this test.

To prevent the plot from becoming too cluttered, we use the final clause to pick just every second point each direction.

When plotted, the result of this SQL looks like the following:



Perform the Query

We are now ready to perform the query. Specifically, we want to:

- 1) get the u velocity component for each grid point from the “ u ” table,
- 2) get the v velocity component for each grid point from the “ v ” table,
- 3) restrict the grid points to those inside the Great Lakes (using the logic [described above](#)),
- 4) reproject the u and v components from the polar stereographic grid to “over the earth” values (see the [earlier footnote](#)), and
- 5) classify the winds as being “lighter” or “heavier”⁶, with one plot file for each class.

Lighter Winds Query

```
> UNLOAD TO slowWinds.plot DELIMITER ' '
  SELECT {+ORDERED, FULL(U), FULL(V)} u.lon,u.lat,
    (sin(u.rot)*v.v + cos(u.rot)*u.u)/10,
    (cos(u.rot)*v.v - sin(u.rot)*u.u)/10
FROM windsu_${T} u, windsv_${T} v, lakes b
WHERE sqrt((sin(u.rot)*v.v + cos(u.rot)*u.u)*
  (sin(u.rot)*v.v + cos(u.rot)*u.u)+
  (cos(u.rot)*v.v - sin(u.rot)*u.u)*
  (cos(u.rot)*v.v - sin(u.rot)*u.u)) <= 5 AND
  lakename IN ('Lake Ontario','Lake Erie','Lake Huron',
    'Lake Superior','Lake Michigan') AND
  ST_Within(('5 POINT(' || u.lon || ' ' || u.lat || ')')::ST_Point,
    b.shoreline) AND
  u.lat = v.lat AND u.lon = v.lon AND
  u.lon BETWEEN -93 AND -75 AND u.lat BETWEEN 40 AND 50 AND
  MOD(u.ix,2) = 0 AND MOD(u.iy,2) = 0;
```

Faster Winds Query

```
> UNLOAD TO fastWinds.plot DELIMITER ' '
  SELECT {+ORDERED, FULL(U), FULL(V)} u.lon,u.lat,
    (sin(u.rot)*v.v + cos(u.rot)*u.u)/10,
    (cos(u.rot)*v.v - sin(u.rot)*u.u)/10
FROM windsu_${T} u, windsv_${T} v, lakes b
WHERE sqrt((sin(u.rot)*v.v + cos(u.rot)*u.u)*
  (sin(u.rot)*v.v + cos(u.rot)*u.u)+
  (cos(u.rot)*v.v - sin(u.rot)*u.u)*
  (cos(u.rot)*v.v - sin(u.rot)*u.u)) > 5 AND
  lakename IN ('Lake Ontario','Lake Erie','Lake Huron',
    'Lake Superior','Lake Michigan') AND
  ST_Within(('5 POINT(' || u.lon || ' ' || u.lat || ')')::ST_Point,
    b.shoreline) AND
  u.lat = v.lat AND u.lon = v.lon AND
  u.lon BETWEEN -93 AND -75 AND u.lat BETWEEN 40 AND 50 AND
  MOD(u.ix,2) = 0 AND MOD(u.iy,2) = 0;
```

⁶ The winds on the Great Lakes are generally not particularly heavy, so for illustrative purposes we’ve picked 13.5 knots (25 kph or 15 mph – a moderate breeze) as the dividing value.

In each query there are four values output. These values represent the 2 (2D) endpoints of the wind speed vector to be plotted. The vector extends from the associated gridpoint to a point offset by amounts proportional to the UEarth and VEarth values (the scaling factor 10 is chosen in order to produce reasonable length vectors).

The queries are run for forecast times from *now*+0 hours to *now*+48 hours (in steps of 3 hours), and each set of vectors, along with the lake shore boundaries and grid points, are plotted to a gif file using the free and open source program [gnuplot](#). The resulting 17 gif files are then turned into a single animated gif file using the free and open source program [gifsicle](#).

For more information

For more information on UFI, visit our website at
<http://www.barrodale.com/bcs/universal-file-interface-ufi>.

To discuss other trial options or to purchase UFI, please contact us at
BCSInfo@barrodale.com.

To learn more about BCS and our other products, please visit our website at
<http://www.barrodale.com>.

Appendix – One of the NetCDF Files

The following is the lead portion of the `ncdump` output applied to one of the NetCDF files used in this demo⁷. All the other files have the same format and structure, except that the VGRD files have a `V_GRD_GDS5_HTGL_10` variable instead of a `U_GRD_GDS5_HTGL_10` variable.

```
netcdf CMC_reg_UGRD_TGL_10_ps15km_2010070900_P021 {
dimensions:
    ygrid_0 = 399 ;
    xgrid_0 = 493 ;
variables:
    float gridlon_0(ygrid_0, xgrid_0) ;
        gridlon_0:La1 = 32.54898f ;
        gridlon_0:Lo1 = 225.3804f ;
        gridlon_0:Lov = 249.f ;
        gridlon_0:Dx = 15.f ;
        gridlon_0:Dy = 15.f ;
        gridlon_0:units = "degrees_east" ;
        gridlon_0:grid_type = "Polar Sterographic Projection (North or South)" ;
        gridlon_0:long_name = "longitude" ;
        gridlon_0:corners = -134.6196f, -72.37911f, -20.99989f, 158.9998f ;
    float gridlat_0(ygrid_0, xgrid_0) ;
        gridlat_0:La1 = 32.54898f ;
        gridlat_0:Lo1 = 225.3804f ;
        gridlat_0:Lov = 249.f ;
        gridlat_0:Dx = 15.f ;
        gridlat_0:Dy = 15.f ;
        gridlat_0:units = "degrees_north" ;
        gridlat_0:grid_type = "Polar Sterographic Projection (North or South)" ;
        gridlat_0:long_name = "latitude" ;
        gridlat_0:corners = 32.54898f, 24.54101f, 46.2826f, 65.23019f ;
    float gridrot_0(ygrid_0, xgrid_0) ;
        gridrot_0:note2 = "apply formulas to derive u and v components relative
            to earth" ;
        gridrot_0:note1 = "u and v components of vector quantities are resolved
            relative to grid" ;
        gridrot_0:formula_v = "Vearth = cos(rot)*Vgrid - sin(rot)*Ugrid" ;
        gridrot_0:formula_u = "Uearth = sin(rot)*Vgrid + cos(rot)*Ugrid" ;
        gridrot_0:units = "radians" ;
        gridrot_0:GridType = "Polar Sterographic Projection (North or South)" ;
        gridrot_0:long_name = "vector rotation angle" ;
    float \10u_P0_L103_GST0(ygrid_0, xgrid_0) ;
        \10u_P0_L103_GST0:initial_time = "06/29/2011 (12:00)" ;
        \10u_P0_L103_GST0:forecast_time_units = "hours" ;
        \10u_P0_L103_GST0:forecast_time = 21 ;
        \10u_P0_L103_GST0:level = 10.f ;
        \10u_P0_L103_GST0:level_type =
            "Specified height level above ground (m)" ;

        \10u_P0_L103_GST0:parameter_template_discipline_category_number =
            0, 0, 2, 2 ;
        \10u_P0_L103_GST0:parameter_discipline_and_category =
```

⁷ The spacing has been altered slightly in order to improve readability.

```
        "Meteorological products, Momentum" ;
\10u_P0_L103_GST0:grid_type =
        "Polar stereographic can be south or north" ;
\10u_P0_L103_GST0:coordinates = "gridlat_0 gridlon_0" ;
\10u_P0_L103_GST0:_FillValue = 1.e+20f ;
\10u_P0_L103_GST0:units = "m/s" ;
\10u_P0_L103_GST0:long_name = "10 meter u velocity" ;
\10u_P0_L103_GST0:production_status = "TIGGE test products" ;
\10u_P0_L103_GST0:center =
        "Canadian Meteorological Service - Montreal (RSMC)" ;

// global attributes:
:creation_date = "Wed Jun 29 08:59:26 PDT 2011" ;
:NCL_Version = "5.2.0" ;
:system = "Linux tungsten.barrodale.com 2.6.18-238.9.1.el5xen #1 SMP
        Tue Apr 12 19:32:47 EDT 2011 i686 i686 i386 GNU/Linux" ;
:conventions = "None" ;
:grib_source = "CMC_reg_UGRD_TGL_10_ps15km_2011062912_P021.grib2" ;
:title = "NCL: convert-GRIB-to-netCDF" ;

data:

gridlon_0 =
-134.6196,-134.4986,-134.3774,-134.256,-134.1344,-134.0126,-133.8905,
-133.7682,-133.6457,-133.523,-133.4,-133.2769,-133.1535,-133.0299,
-132.9061,-132.782,-132.6578,-132.5333,-132.4087,-132.2838,
```

...