NADCON5-ng

0.0.2

Generated by Doxygen 1.8.11

Contents

Chapter 1

NADCON5-ng

Tweaks and Updates to US National Geodetic Survey `NADCON5` Tool. Used to convert Geodetic Data between various US Datums, including: US Standard Datum (`USSD`) used prior to `NAD27`, North American Datum of 1927 (`NAD27`), and various realizations of the North American Datum of 1983 `NAD83`

`Link To Doxygen Documentation Website`

The intent of this fork is to adapt the existing tool to be accessible to more users, developers, and data scientists.↩ Through the implementation of additional interfaces and workflows on top of existing NADCON5 Code Base.

> **NOTE**: This project is a personal project that is not in any way affiliated with the US Government, NOAA, or the National Geodetic Survey

Derivative Work: Additions and Modifications to this software are released explicitly under Public Domain.

As a product of the United States Government NADCON5 Source Code is considered a work under public domain.

Build the daset with one command

```
make
```

Project Status

This project is new, feature requests and development will be driven through issues filed in github.

At the time of this README was update, the following was true

1. The existing processing pipeline has been offloaded to GNU Make to eliminate in-source builds
2. Doxygen was strapped on top of the project to create documentation , source files were modified, superficially, to export documentation in doxygen
3. Documentation and website live, hosted on `github-pages`, at url: `https://docs.nc5ng.↩ org/latest`
4. Initial Framework for a python glue library, with several functioning submodules and functions
    - install with `pip install nc5ng`

On the Immediate Roadmap

1. Remove dependence on proprietary Oracle Fortran `f95`
    - Requires mapping build options to `gfortran` and correcting where necessary
    - Biggest issue is compiler specific handling of I/O and certain convenience extensions, not the math
2. Take over the "batch generator" programs (e.g. makework() , makeplotfiles01 , etc.) so that individual conversions can be done as needed, through Make or otherwise
3. Create an `install` target - install existing fortran programs onto system as a distribution
    - Some tweaks to programs to make this doable (path dependencies)
    - Pruning of applications to core install package

What is NADCON5?

`NGS NADCON5 Front Page`

`NGS NADCON5 Website`

The Following Information is Reproduced from the NADCON5 Webpage from NGS

What is NADCON 5.0?

NADCON 5.0 performs three-dimensional (latitude, longitude, ellipsoid height) coordinate transformations for a wide range of datums and regions in the National Spatial Reference System. NADCON 5.0 is the replacement for all previous versions of the following tools:

- NADCON, which transformed coordinates between the North American Datum of 1927 (NAD 27) and early realizations of the North American Datum of 1983 (NAD 83), and
- GEOCON, which transformed coordinates between various latter realizations of NAD 83.

How do I use NADCON 5.0?

NADCON 5.0 is functionally implemented in NGS's Coordinate Conversion and Transformation Tool. Unlike earlier versions of NADCON and GEOCON, NADCON 5.0 is not a stand-alone tool.

Visit the NADCON 5.0 Digital Archive to access raw transformation data that make up NADCON 5.0 (e.g., grids, images, software).

How can I learn more about NADCON 5.0?

`NOAA Technical Report NOS NGS 63 (PDF, 17 MB)` provides detailed information on NADCON 5.0, and the digital archive includes plots and data.

Building `NADCON5-ng`

Build simply with

```
make
```

Which will build the initial tools and generate conversion output and images for the configured conversion

Dependencies

1. `Generic Mapping Tools` `**GMT**`
   - Tested with 5.2.1
   - Install on Debian Systems with `sudo apt-get install gmt gmt-dcw gmt-gshhg`
2. Oracle Fortran (`f95`) available for free (as in money, but not freedom) in `Oracle Developer Studio`
   - Set `f95` path with environment variable `FC` (Per `GNU Conventions`)
3. GNU Make

Build Options

The configurable options for the build steps are

1. `OLD_DATUM` - source datum (default: `ussd`)
2. `NEW_DATUM` - target datum (default: `nad27`)
3. `REGION` - geographical region (default: `conus`)
4. `GRIDSPACING` - Grid Spacing in arc-seconds (default: `900`)
5. `MAP_LEVEL` - Map Resolution Flag (default: `0`)

These can be set as environment variables or directly on the command line

```
export OLD_DATUM=nad27
export NEW_DATUM=nad83
make
# Equivalent
OLD_DATUM=ussd NEW_DATUM=nad27 make
# Third Option
make OLD_DATUM=ussd NEW_DATUM=nad27
```

Targets

The Upstream build sequence can be simulated by using the targets `doit` `doit2` `doit3` `doit4`, as in

```
make doit
make doit2
make doit3
make doit4
```

This can be useful to compare results from the vanilla `NADCON`

Additionally, for the intermediate scripts `gmtbat0X` convenience targets are provided to manually step through the asset compilation

```
make gmtbat01
make gmtbat02
make gmtbat03
make gmtbat04
make gmtbat05
make gmtbat06
make gmtbat07
```

Cleaning up is easy

Delete only the current configured build

```
make clean
```

Delete all compiled output (deletes build directory)

```
make mrclean
```

Chapter 2

NADCON5-ng Manual

`NADCON5-ng` has been updated from the original development to use GNU Make as the target build system

2.1 NADCON5.0 Data Pipeline

The toplevel Makefile defines the rules to construct the NADCON5-ng dataset in the traditional NADCON5.0 fashion, using the same fortran programs, simply managed by Make

To generate the target NADCON data, from the command line

```
$ cd nadcon5-ng
$ make OLD_DATUM=ussd NEW_DATUM=nad27 REGION=conus GRIDSPACING=900 MAPLEVEL=0
$ cd build/out.ussd.nad27.conus.900.0
```

Where the variables are specified and indicate

- `OLD_DATUM` = Source Datum
- `NEW_DATUM` = Target Datum
- `REGION` = Geographic Region to Compute
- `GRIDSPACING` = The Grid Spacing in arcsec
- `MAPLEVEL` = The zoom level of images to generate (`0,1,2`)

For more Information on the meaning of thes parameter see the documentation of functions in: NADCON5 Build Programs

Note

Conversion between datums is restricted in a strictly one-step chronoligical direction. Not all Regions are possible with all conversions

2.1.1 Step-by-step

With no arguments Make will execute the default target, in this case `all`, which is defined to run the data pipeline.

Additional targets are defined to step through the data pipeline.

These targets approximately mimic the steps taken in the upstream build scritpts `doitX.bat`

```
$ make doit
$ make doit2
$ make doit3
$ make doit4
```

, will execute the build system in the same step-by step fashion as the upstream batch files. Generating a portion of the output each time.

These `doit` targets use programs defined in NADCON5 Build Programs to generated Generic Mapping Tools (GMT) batch files in the output directory. Additionally, as a first step a "work" file is constructed (see makework)

Dataset construction can be done by stepping through these GMT scripts

```
$ make workfile
$ make gmtbat01
$ make gmtbat02
...
$ make gmtbat07
```

Allowing one to manually execute the batch file in the build directory

Note

These scripts call GMT functions without explicitly calling `gmt` which may not work on all distributions, shell wrappers for gmt are provided in `gmt_wrappers/` and can be added to path for convenience.

2.1.2 Data Archive

An archive file, with a unix timestamp is constructed with the `archive` target

That is,

```
$ make archive
```

This creates a file

```
build/nadcon5-TIMESTAMP.OLD_DATUM.NEW_DATUM.REGION.GRIDSPACING.MAPLEVEL.tgz
```

2.1.3 Output Files

Output files are generated in the folder

```
build/out.OLD_DATUM.NEW_DATUM.REGION.GRIDSPACING.MAPLEVEL
```

2.2 Source Compilation

The required files are compiled by the Data Pipeline automatically, but if you need to do this manually it can be done from the `src/` directory

```
$ cd nadcon5-ng/src
$ make
```

Binaries are placed in the directory

```
build/bin
```

2.3 Documentation Compilation

This page, as well as all the documentation on this page is also generated using Make

2.3.1 HTML

To produce html documentation suitable for browsing or hosting

```
$ cd nadcon5-ng/docs
$ make full_docs
```

Additionally, other output forms are available

All forms of documentation can be compiled at once by omitting the target

```
$ cd nadcon5-ng/docs
$ make
```

2.3.2 Latex and PDF:

Latex sources and compiled PDF can be created by calling (in the docs folder)

```
$ make latex_docs
```

2.3.3 manpage

∗NIX style manual pages, suitable for use by the `man` command can be generated with the `bin_manual` and `lib_manual` targets

Generate manual pages for the Compiled Programs (`man.1`)

```
$ make bin_manual
```

Generate Documentation for subroutines and functions

```
$ make lib_manual
```

2.4 Dependencies

- Oracle Fortran
- Generic Mapping Tools GMT
- GNU Make
- Doxygen
-

2.5 Makefile Primer

Note: This primer has been adapted from other sources and is not specific to NADCON5-ng

Makefiles define **rules** to make **targets**.

A Makefile may look something like this.

```
# Target to Compile a single file.
#  The first line defines the target
#  further lines define the commands
#  that are run
object.o:
  cc -o object.o object.c

# Target to link executable with external releaselib.
#  object.o is a dependency of release_exec
release_exec: object.o
  ld  -r -o release_exec  releaselib object.o

# Convenience Target
release: release_exec

# Since "release" does not actually produce a file
#  this is required boiler plate
.PHONY: release
```

This Makefile is actually equivalent to a simple compiler one-liner.

```
cc -o release_exec -lreleaselib object.c
```

However, it already provides power behind the scenes.

For example Calling

```
make release
```

Will not recompile the executable if `object.c` has not been changed.

The real power of `Make` comes from "Pattern Rules" constructed using `Automatic variables` and `Pattern Matching` which result in a more general rule.

```
# The target name is special
#  and the variable $@
#  refers to the target name
test.o:
  cc -o $@ object.c

# The dependency name is special
#  and the variable $< refers to the
#  first (only) dependency
better_test.o: object.c
  cc -o $@ $<

# Multiple Dependencies
#  can be refered to by the variable
#  $^
other_test.a: test.o better_test.o
  ld -r -o $@ $^

# Pattern matching makes general rules
#  The following  rule compiles all .c files to .o files
%.o:%.c
  cc -o $@ $<

# PHONY rules can be used to build multiple
#  dependencies
build_some_libs: pattern_lib.o \
                 pattern_lib2.o \
                 pattern_lib3.o \
                 other_test.a
.PHONY: build_all_libs
```

By default `Make` assumes that the target is a real file that is created, and will track the state of dependency based on the file and if it exists.

As such when dealing with build directories, a target must be specified with its path for make to track its status.

The `.PHONY` target is a special target that indicates that it should not be tracked and will be rebuilt every time, often this is good for things like printing debug output or a convenience target that builds multiple targets (e.g. `all`).

In our Makefiles you will see things like the following, this example is our default rule for building fortran programs

```
$(BIN_DIR)/%:$(notdir %).f | $(BIN_DIR)
  $(FC) $(FFLAGS) $< -o $@
```

In this case, `FC`, `FFLAGS`, are variables define in our Makefile which can be overridden at the command line, this is described below. `BIN_DIR` is a Makefile Variable and is the directory where binary programs artifacts are to be placed after building.

This default rule maps all programs like `/path/to/build/bin/myprogram` to compile from the file `myprogram.f` using the Fortran Compile `FC` and the compilation flags `FFLAGS`

Chapter 3

NADCON5.0 Reference Manual

Reference Manual for the NADCON5.0 Codebase

This document serves as a high level reference for the filetypes and processing programs in NADCON5.0 .

A detailed Documentation of the Source Code is available in `Code Documentation` chapter. Where the primary build programs are described in NADCON5 Build Programs and the auxiliarly library functions and subroutines are described in NADCON5 Core Library

3.1 Input Files

This Section Defines all the Files used by NADCON5.0 programs in the NADCON5 Core Library to generate the `GMT Batch Files` and some of the `Output Files`.

3.1.1 Control File

The Control File describes the `In Files` which contain data for a given transformation.

They are named

```
control.OLD_DATUM.NEW_DATUM.REGION
```

Control Files exist in the folder `data/Control` . The existence of a control file means that the transformation is defined.

Note

Conversely. If a control file does not exist, then the transformation is not defined

The Structure of the Control File is a list of valid input files with formatted header information.

We show this structure by using an example.

```
HEADER: Master File for creating a NADCON5 work file
REGION: CONUS
DATUM1: USSD
DATUM2: NAD27
REJMET: 10000
NFILES: 49
NADCON5.USSD.NAD27.AL.in
NADCON5.USSD.NAD27.AR.in
...
49 Total Files Listed
...
```

The Control file is used by makework to generate the initial `Work File`

Note

It is critical that the header labels (HEADER, REGION, DATUM1, etc.) are not changed, as the program makework verifies each line against these strings.

Parameter Details

- HEADER - Header line. Can contain anything.
- REGION - REGION Parameter from Build Pipeline. Must conform to the following list:
  - conus
  - alaska
  - prvi
  - hawaii
  - guamcnmi
  - as
  - pribilof
  - stlawrence
- DATUM1 - The older datum, chronologically. OLD_DATUM Parameter from Build Pipeline.
- DATUM2 - The newer datum, chronologically. NEW_DATUM Parameter from Build Pipeline.
- REJMET The rejection criteria in meters. Basically if any latitude shift or longitude shift or horizontal shift exceeds this value (in absolute value), then all shifts for this point are set to zero (to avoid asterisks in the output file) but the whole line is labeled with a triple reject criteria, effectively eliminating the pair from use.
- NFILES - The number of ∗.in files which connect the old and new datums in the region being addressed.

3.1.2 workedits File

Used by makework and located in data/Work , the workedits is a database of rejected datum transformation points. That is, the workedits file provides a list of specific points in the InFiles which should be rejected for various, manually identified, reasons.

The file format is fixed column delminted. The formatting is hardcoded in makework with index records, the format each individual read is a complete string. Strings are not "stripped" so left justification is required.

The columns are:

```
01- 10  : olddtm : lower case, left justified
    11  : "|"    : vertcal spacer just for ease of reading
12- 21  : newdtm : lower case, left justified
    22  : "|"    : vertcal spacer just for ease of reading
23- 32  : region : lower case, left justified (conus, alaska, hawaii, prvi, guamcnmi, as)
    33  : "|"    : vertcal spacer just for ease of reading
34- 39  : PID    : upper case, left justified
    40  : "|"    : vertcal spacer just for ease of reading
41- 43  : rejects: Three digits (0's or 1's only) to reject lat, lon, eht, in that order.  '1' = reject, '0' =
    44  : "|"    : vertcal spacer just for ease of reading
45-200  : reason : Upper/lower case, giving first your name then reason for the line to exist
```

Or, in other Words:

```
olddtm    |newdtm     |region     |PID    |rej|Reason
lwr case  |lwr case   |lwr case   |uprcas|0's|Give your name or initials first, then reason
left      |left       |left       |6 char|or |Upper or lower case or both
justified|justified  |justified  |      |1's|
10 char   |10 char    |10 char    |      |   |
```

An Example:

```
ussd       |nad27      |conus      |KG0640|110|DAS: USSD and NAD 27 coordinates are impossibly identical.
ussd       |nad27      |conus      |DK3691|110|DAS: USSD and NAD 27 coordinates are impossibly identical.
XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX-XXXXXX-XXX-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  oldtm        newtm       region    pid    rej                                 comment
```

Parameters:

- `oldtm` - Source Datum
- `newtm` - Target Datum
- `region` - Geographical Region (e.g. conus)
- `pid` - Point ID
- `rej` - Bit Field to reject lat, lon, eht
- `Reason` - Comment Field

3.1.3 In Files

The In Files are located in data/InFiles and contain the actual reference and transformation data for the construction of the NADCON5.0 dataset.

These files are created per state/territory and for every transformation vector.

The files are named per the convention

```
NADCON5.DATUM1.DATUM2.STATE.in
```

The format of these files is a fixed format file. The format strings are defined in makework

The first row of the file identifies the converted datums in comment header, with a Fortran Format specifier of

```
100 format(27x,a15,26x,a15)
```

For example: (with format offsets and variable name indicated underneath)

```
                         US Standard Datum           |                  NAD 27
---------------------------XXXXXXXXXXXXXXX--------------------------XXXXXXXXXXXXXXX
                             nameh                                     namef
```

That is, 15 Character Strings with offsets `27` and `68` (`27+15+26`).

Note

Characters outside the defined 15 Character Region are simply ignored, in the above example, the String `US Standard Datum` would be truncated and trimed by Fortran to `Standard Datum`. However, this header line is not currently used for anything by makework

The remaining rows define the actual datapoints, with a fortran format specifier of

```
101 format(a6,1x,a2,5x,a13,1x,a14,1x,a9,3x,a13,1x,a14,1x,a9)
```

For Example:

```
AA5496 CA 085 N372639.93300 W1220955.92000       N/A | N372639.71836 W1220959.79579       N/A
XXXXXX-XX-----XXXXXXXXXXXXX-XXXXXXXXXXXXXX-XXXXXXXXX---XXXXXXXXXXXXX-XXXXXXXXXXXXXX-XXXXXXXXX
pid    state       clath         clonh         cehth       clatf           clonf         cehtf
```

Parameters:

- `pid` - Point ID (NGS Internal Unique Designator)
- `state` - State this point belongs to
- `clath` - Source Datum Lat. (Decimal Degrees with Cardinal Direction)
- `clonh` - Source Datum Lon. (Decimal Degrees with Cardinal Direction)
- `cehth` - Source Datum Height (Meters)
- `clatf` - Target Datum Lat.
- `clonf` - Target Datum Lon.
- `cehtf` - Target Datum Height

Note

Partial Transformation points (e.g. no Height data) are specified by using `N/A` specifier to exclude the lat. , lon. , and/or height from the calculations.

The file is read until the last record indicated by an end of file (`EOF`), there is no footer or record counter.

3.2 GMT Batch Files

Todo Write This Section

3.3 Output Files

3.3.1 Work File

The `workfile` is the first file created in the build pipeline. It serves as a local database of all the points and associated transformation data which is later used to construct images, grids, and other outputs.

The workfile is generated with the name

```
work.DATUM1.DATUM2.REGION
```

And is placed (hardcoded) in the local `Work/` directory

The new file has the following format:

```
 Cols  Format Description
 1-  6   a6    PID
     7   1x    - blank -
 8-  9   a2    State
    10   a1    Reject code for missing latitude pair (blank for good)
    11   a1    Reject code for missing longitude pair (blank for good)
    12   a1    Reject code for missing ellip ht pair (blank for good)
    13   1x    - blank -
14- 27 f14.10  Latitude (HARN), decimal degrees (-90 to +90)
    28   1x    - blank -
29- 42 f14.10  Lonitude (HARN), decimal degrees (0 to 360)
    43   1x    - blank -
44- 51   f8.3  Ellipsoid Height (HARN), meters
    52   1x    - blank -
53- 61   f9.5  Delta Lat (FBN-HARN), arcseconds
    62   1x    - blank -
63- 71   f9.5  Delta Lon (FBN-HARN), arcseconds
    72   1x    - blank -
73- 81   f9.3  Delta Ell Ht (FBN-HARN), meters
    82   1x    - blank -
83- 91   f9.5  Delta Horizontal (absolute value), arcseconds
    92   1x    - blank -
93-101   f9.5  Azimuth of Delta Horizontal (0-360), degrees
   102   1x    - blank -
103-111   f9.3  Delta Lat (FBN-HARN), meters
   112   1x    - blank -
113-121   f9.3  Delta Lon (FBN-HARN), meters
   122   1x    - blank -
123-131   f9.3  Delta Horizontal (absolute value), meters
```

Which is defined by fortran `format` Identifier

```
  104     format(a6,1x,a2,a1,a1,a1,1x,f14.10,1x,f14.10,1x,f8.3,1x,
 *    f9.5,1x,f9.5,1x,f9.3,1x,f9.5,1x,f9.5,1x,f9.3,1x,f9.3,1x,f9.3,
 *    1x,a10,1x,a10)
```

An example of a workfile record:

```
BG3971 AL  1  30.3435430556 272.5144786111    0.000  -0.24400  -0.43700     0.000   0.50050 237.09789    -7.53
XXXXXX-XXxXx-XXXXXXXXXXXXXX-XXXXXXXXXXXXXX-XXXXXXXX-XXXXXXXXX-XXXXXXXXX-XXXXXXXXX-XXXXXXXXX-XXXXXXXXX-XXXXXXXX
pid state|rej    xlath          xlonh         xehth    dlatsec   dlonsec    dehtm    dhorsec  azhor      dlatm
```

Chapter 4

LICENSE

Chapter 5

Todo List

Page NADCON5.0 Reference Manual

Write This Section

Chapter 6

Module Index

6.1 Modules

Here is a list of all modules:

Chapter 7

Module Documentation

7.1 NADCON5 Build Programs

Programs which perform the generation of `GMT` batch scripts and the creation of output files using NADCON5 Core Library and helpers.

Directories

- directorysrc

  Folder containing the NADCON5 Build Programs and subfolders Code/Subs and Code/BinSource.

Functions

- program checkgrid

  Part of the NADCON5 build process, generates `gmtbat04`
- program makeplotfiles01

  Part of the NADCON5 process, generates gmtbat01.
- program makeplotfiles02

  Part of the NADCON5 process, generates `gmtbat03`
- program makeplotfiles03

  Part of the NADCON5 process, generates `gmtbat06`
- program makework

  ***Program** to create a work file which will serve as the primary information needed to analyze and create NADCON v5.0 grids.*
- program mymedian5

  Program to filter Map Data for GMT Plotting.
- program myrms

  Part of the NADCON5 build process, generates `gmtbat05`

7.1.1 Detailed Description

Programs which perform the generation of `GMT` batch scripts and the creation of output files using NADCON5 Core Library and helpers.

The elements described here are the "doers", programs that construct the output using elements of NADCON5 Core Library

7.1.2 Function Documentation

7.1.2.1 program checkgrid ( )

Part of the NADCON5 build process, generates `gmtbat04`

Creates a batch file called

```
  gmtbat04.(olddtm).(newdtm).(region).(igridsec)
```

This Program:

1) Compare grids of dlat, dlon and deht to vectors of dlat, dlon and deht. 2) Spit out interpolated (from grid) vectors 3) Spit out differential (interpolated minus original) vectors. 4) Create a GMT batch file to plot said vectors.

The input vectors:

```
Represent *all* (outlier removed)
vectors of dlat/dlon/deht for the
olddatum/newdatum/region combination
```

However, for the sake of understanding, the vectors will be read in from their "thinned" and "dropped" files, so that we can generate statistics of:

thinned-versus-gridded dropped-versus-gridded all-versus-gridded

This is important, since ONLY the thinned vectors went into the grid and it seems that their statistics should be better against the grid than the dropped vectors. Additionally, one might argue that the only independent check on the grid is the dropped agreement. We'll see.

The input grids:

```
dlat/dlon/deht grids based on thinning
all of the vectors (see above) using
a median thinning at some block spacing
in arcseconds, and gridding to that same
block spacing.
```

Program arguments

Arguments are newline terminated and read from standard input

They are enumerated here

Parameters

oldtm	Source Datum
newdtm	Target Datum,region
region	Conversion Region
agridsec	Grid Spacing in arcsec

Example:

```
olddatum = 'ussd'
newdatum = 'nad27'
region = 'conus'
agridsec = '900'
```

Program Inputs:

Changelog

2016 08 26:

Changed "getmapbounds" to bring in a better way of computing the reference vector location and added a new variable for its label

Also changing the call to "getmapbounds" to give it "olddatum" and "newdatum" to aide in filtering out things like the Saint regions in Alaska for unsupported transformations.

2016 07 29:

Scrapped personal placement of vectors and just let them sit outside/below the map

2016 07 21:

Added code to allow for optional placement of reference vectors, coming from "map.parameters" as read in subroutine "getmapbounds"

2015 10 08:

Added HOR output in M and S for Lat/Lon to both "gi" and "dd" vector output.

Combined: v(m/s)(a/t/d)(gi/dd)lat... v(m/s)(a/t/d)(gi/dd)lon... into : v(m/s)(a/t/d)(gi/dd)hor...

2015 9 10:

Initial Release For use in creating NADCON5 Built by Dru Smith

References bicubic(), bilin(), biquad(), bwplotvc(), getgridbounds(), getmapbounds(), and onzd2().

7.1.2.2 program makeplotfiles01 ( )

Part of the NADCON5 process, generates gmtbat01.

Program to take a "work" file and create a variety of GMT-ready data files of the following

1. Coverage in latitude
2. Coverage in longitude
3. Coverage in ellipsoid height
4. Vectors in latitude
5. Vectors in longitude
6. Vectors in ellipdoid height
7. Vectors in horizontal (properly azimuthed)

It furthermore will create batch file to run the GMT scripts:

```
gmtbat01.(olddtm).(newdtm).(region).(mapflag)
```

Program arguments

Arguments are newline terminated and read from standard input

They are enumerated here

Parameters

oldtm	Source Datum
newdtm	Target Datum,region
region	Conversion Region
mapflag	Map Detail Level

Example:

```
olddatum = 'ussd'
newdatum = 'nad27'
region = 'conus'
mapflag = 0
```

Program Inputs:

Changelog

2016 08 26

Added new code to do reference vectors consistently See DRU-12, p. 56-57 Also changing the call to "getmapbounds" to give it "olddatum" and "newdatum" to aide in filtering out things like the Saint regions in Alaska for unsupported transformations.

2016 07 29:

Scrapped the code for personalized reference vector location. Just put all ref vectors outside/below plot.

2016 07 21:

Added code to allow for optional placement of reference vectors, coming from "map.parameters" as read in subroutine "getmapbounds"

References bwplotcv(), bwplotvc(), getmapbounds(), and onzd2().

7.1.2.3 program makeplotfiles02 ( )

Part of the NADCON5 process, generates `gmtbat03`

Built upon the skeleton of "makeplotem.f" for GEOCON v2.0 But built specifically for NADCON v5.0. So different in file names and expanded plot creation that it was given the new name "makeplotfiles02.f" to align with another NADCON5 program "makeplotfiles01.f"
Creates a batch file called

```
  gmtbat03.(olddtm).(newdtm).(region).(igridsec)
```

That batch file will create JPGs of:

1. Color Plots of the dlat/dlon/deht grids at T=0.4
2. Color Plots of the "method noise" grids (the "d3" grids, see DRU-11, p. 150) with thinned coverage overlaid
3. B/W plots of thinned vectors that went into the T=0.4 transformation grid
4. B/W plots of dropped vectors that did not go into the T=0.4 transformation grid
5. B/W plots of thinned coverage of points that went into the T=0.4 transformation grid
6. B/W plots of dropped coverage of points that did not go into the T=0.4 transformation grid

Program arguments

Arguments are newline terminated and read from standard input

They are enumerated here

Parameters

oldtm	Source Datum
newdtm	Target Datum,region
region	Conversion Region
agridsec	Grid Spacing in arcsec

Example:

```
olddatum = 'ussd'
newdatum = 'nad27'
region = 'conus'
agridsec = '900'
```

Program Inputs:

Changelog

2016 08 26

Added new code to do reference vectors consistently See DRU-12, p. 56-57 Also fixed a typo in reference vector length for vmtcdeht plots Also changing the call to getmapbounds to give it `olddatum` and `newdatum` to aide in filtering out things like the Saint regions in Alaska for unsupported transformations.

2016 07 29:

Scrapped code about personalized reference vectors. Just put all reference vectors outside/below plot Also moved gridstats and vecstats out into the `/Subs` directory to be used by other programs (like makeplotfiles03.f)

2016 07 28:

Changed code to build the color palette of the `d3` grids around the median, and not ave or std. See DRU-12, p. 48

2016 07 21:

Added code to allow for optional placement of reference vectors, coming from `map.parameters` as read in subroutine getmapbounds

2016 01 21:

Updated to get the CPT values fixed in `d3` grids, so that (cpthi - cptlo) is exactly divisible by `cptin` at (2 x csm)

2015 10 27:

Updated to work with the new naming scheme (see DRU-11, p. 150)

2015 10 05:

Updated to work with the new naming scheme (see DRU-11, p. 139)

References bwplotcv(), bwplotvc(), coplot(), coplotwcv(), cpt(), cpt2(), getmapbounds(), gridstats(), onzd2(), and vecstats().

7.1.2.4 program makeplotfiles03 ( )

Part of the NADCON5 process, generates `gmtbat06`

Creates a batch file called

```
gmtbat06.(olddtm).(newdtm).(region).(igridsec).(mapflag)
```

That batch file will create JPGs of:

1. Color Plots of the rddlat/rddlon/rddeht grids
2. B/W plots of coverage of RMS'd differential vectors that went into the grid
3. B/W plots of RMS'd differential vectors that went into the grid

Program arguments

Arguments are newline terminated and read from standard input

They are enumerated here

Parameters

oldtm	Source Datum
newdtm	Target Datum,region
region	Conversion Region
agridsec	Grid Spacing in arcsec
mapflag	Map Generation Level

Example:

```
olddatum = 'ussd'
newdatum = 'nad27'
region = 'conus'
agridsec = '900'
mapflag = '0'
```

Program Inputs:

Changelog

2016 10 19:

FIX for the HARN/FBN transformation. As it stood, the "gridstats" was returning "0.0" as the median of the post-masked "ete" grid (which is true, but unfortunate.) I've changed the call to "gridstats" to send it the "PREMASKED" version of the "ete" grid, so the median won't be zero.

2016 08 26

Added new code to do reference vectors consistently See DRU-12, p. 56-57

Fixed an error where "lorvogehtm" was declared real∗8 but past 72 column, so defaulting to integer∗4 and coming out as "0.000" on plots

Also changing the call to "getmapbounds" to give it "olddatum" and "newdatum" to aide in filtering out things like the Saint regions in Alaska for unsupported transformations.

2016 08 02:

Changed the color palette for "09" grids from 2xMedian to 3xMedian

2016 07 29:

Dropped code about personalized reference vectors and just let them be below/outside map

2016 08 01:

Moved "gridstats" to subroutines Also, completely removed the in-program computations of the color palette, and instead relied on "cpt" and "cpt2" as per "makeplotfiles02"

2016 07 21:

Added code to allow for optional placement of reference vectors, coming from `map.parameters` as read in subroutine getmapbounds

2016 01 21:

Updated to fix the CPT for the "09" (data noise) grids so that (cpthi - cptin) is exactly divisible by cptin

2015 11 09:

Updated to adopt new naming scheme (yes, again) (See DRU-11, p. 150), as well as creating plots of the total error grid.

2015 10 07:

Latest Version which had new naming scheme and dual-computations of arcseconds and meters for lat/lon See DRU-11, pl. 139.

References bwplotcv(), bwplotvc(), coplot(), cpt2(), getmapbounds(), gridstats(), and onzd2().

7.1.2.5 program makework ( )

Program to create a *work* file which will serve as the primary information needed to analyze and create NADCON v5.0 grids.

This program is based on previous programs that were created by Dru Smith during the GEOCON v2.0 process. It has been modified specifically to be a tool for NADCON v5.0.

Rather than have multiple programs (1, 2, 3, 4 as was the case for GEOCON v2.0), It was decided make ONE working "makework.f" program (this one) and have it feed off of an input file which can be modified.

The input file will reflect all that is necessary to create a work file.

Program arguments

Arguments are newline terminated and read from standard input

They are enumerated here

Parameters

oldtm	Source Datum
newdtm	Target Datum,region
region	Conversion Region

Program Inputs:

- A *control file* in directory `Control/`, the name is generated from input arguments
  Known control file names are:

  ```
  cfname = control.ussd.nad27.conus
  cfname = control.nad27.nad83_1986.conus
  ```

- A *manual edits* file, called `workedits` in directory `Work/`

By way of example...

If the input file controlling the creation of the work file is:

```
Control/control.ussd.nad27.conus
```

then the output data file is:

```
Work/work.ussd.nad27.conus
```

The work file has the following format:

```
Cols  Format Description
  1-  6   a6    PID
      7   1x    - blank -
  8-  9   a2    State
     10   a1    Reject code for missing latitude pair (blank for good)
     11   a1    Reject code for missing longitude pair (blank for good)
     12   a1    Reject code for missing ellip ht pair (blank for good)
     13   1x    - blank -
 14- 27 f14.10  Latitude (Old Datum), decimal degrees (-90 to +90)
     28   1x    - blank -
 29- 42 f14.10  Lonitude (Old Datum), decimal degrees (0 to 360)
     43   1x    - blank -
 44- 51   f8.3  Ellipsoid Height (Old datum), meters
     52   1x    - blank -
 53- 61   f9.5  Delta Lat (New Datum minus Old Datum), arcseconds
     62   1x    - blank -
 63- 71   f9.5  Delta Lon (New Datum minus Old Datum), arcseconds
     72   1x    - blank -
 73- 81   f9.3  Delta Ell Ht (New Datum minus Old Datum), meters
     82   1x    - blank -
 83- 91   f9.3  Delta Horizontal (absolute value), arcseconds
     92   1x    - blank -
 93-101   f9.5  Azimuth of Delta Horizontal (0-360), degrees
    102   1x    - blank -
103-111   f9.3  Delta Lat (New Datum minus Old Datum), meters
    112   1x    - blank -
113-121   f9.3  Delta Lon (New Datum minus Old Datum), meters
    122   1x    - blank -
123-131   f9.3  Delta Horizontal (absolute value), meters
    132   1x    - blank -
133-142   a10   Old Datum Name
    143   1x    - blank -
144-153   a10   New Datum Name
    format(a6,1x,a2,a1,a1,a1,1x,f14.10,1x,f14.10,1x,f8.3,1x,
   *f9.5,1x,f9.5,1x,f9.3,1x,f9.3,1x,f9.5,1x,f9.3,1x,f9.3,1x,f9.3,
    1x,a10,1x,a10)
```

This differs from Dennis's GEOCON v1.0 in that:

- 3 reject codes
- 10 decimal places in latitude (See DRU-10, p. 123)
- 10 decimal places in longitude (See DRU-10, p. 123)
- Lat, Lon and Horizontal in both arcseconds and meters each
- Azimuth of Horizontal
- Identification of which datums are transformed

References

NADCON5:

See:

- DRU-11, p. 124

GEOCON v2.0:

See:

- DRU-10, p. 143
- DRU-11, p. 10
- DRU-11, p. 26
- DRU-11, p. 56

Changelog

2017 11 19 (NG)

Formated Comments to be compatible with Doxygen Due to deprecation of various arguments for GMT tools, invocation of `xyz2grd` and `grd2xyz` have arguments options changed in generated files

- `-bos` -> `-bo3f` with equivalent meaning of a 3 column single precision output
- `-bis` -> `-bi3f` with equivalent meaning of a 3 column single precision input

2016 09 14:

Due to some complications in mymedian5.f which happen if data is in the *work* file that is not to be sorted and used, I've decided to put the *out of grid* point removal code here, so that such points will go into the *work* file but will all get a `111` set of reject codes so they don't go forward in the processing.

2016 09 13:

Fixed a bug that sends an incoming `0` reject flag as a **zero**. All later programs expect a BLANK for a "good" reject code. The incoming `0` is from the `workedits` file and is fine to come in, but must go OUT as a BLANK.

Also, put in code to correct situations where an entry is in `workedits`, but the PID for that entry isn't actually in the incoming data. This relies on a new vector `EditTracker`

2016 02 26:

Change (see: DRU-12, p. 18) to reflect the decision that "manual edits" should ONLY edit data OUT and **not** add data back in.

2016 01 07:

Changed to split "Relevant Edits" into three counts: lat, lon and eht

References getgridbounds().

7.1.2.6 program mymedian5 ( )

Program to filter Map Data for GMT Plotting.

1. Run a customized block-median thinning algorithm on coordinate differences (horizontal and ellipsoid height)
2. Save the thinned data to GMT-ready plottable files
3. Save the removed data to GMT-ready plottable files
4. Create a GMT batch file (gmtbat02) to grid the thinned data at T=0.4 (which becomes the final transformation grid) and also at T=1.0 and T=0.0, (whose difference becomes the bases for the "method noise" grid)
5. If, and only if, we are doing the combination of HARN/FBN/CONUS, insert commands into gmtbat02 to apply a mask to the "04.b" grids (See DRU-12, p. 36-37) See DRU-11, p. 127

unlike `mymedian.f`, this program is set up to filter/process all data at once in one run. Also, significant philosophical changes occurred, including:

1. Median filter on absolute horizontal length, and then, when we have our "kept" points, we use the lat and lon of those kept points to grid lat and lon separately. (mymedian.f actually sorted lat medians and lon medians separately, raising the very real possibility that separate points would go into each grid.)
2. Nothing RANDOM! No coin flipping, etc. It was viewed, for NADCON 5.0, as scientifically improper for the final grids to be reliant upon a filtering mechanism that could be different each time it was run.

Program arguments

Arguments are newline terminated and read from standard input

They are enumerated here

Parameters

oldtm	Source Datum
newdtm	Target Datum,region
region	Conversion Region
agridsec	Grid Spacing in Arc Seconds

Program Inputs:

Changelog

2016 09 14

Bug found when running Hawaii with points outside grid – some mixup between "ikt" and "ipid". Changed this program as follows: It now REQUIRES that the incoming data has NO points outside the grid boundary. This has been forced by giving such points a "444" reject code in "makework" when creating the work file. By going through "makeplotfiles01" with a "444" reject, those points won't even make it into the "all" file, which is our input for median filtering here...

2016 06 29

Updated to insert masking commands for the "...04.b" (transformation grid) into gmtbat02 when working ONLY in the HARN/FBN/CONUS combination.

2015 10 27

For use in creating NADCON5 Built by Dru Smith Built from scratch, scrapping all previous "mymedian" programs used in making GEOCON

References getgridbounds(), indexxd(), and indexxi().

7.1.2.7 program myrms ( )

Part of the NADCON5 build process, generates `gmtbat05`

Creates a batch file called

```
gmtbat05.(olddtm).(newdtm).(region).(igridsec)
```

Program to

1. Run a customized RMS-computing algorithm on vector differences (gridded minus true)
2. Save the RMS data to GMT-ready plottable files
3. Create a GMT batch file to plot both the thinned data and removed data in both coverage and vectors

Unlike "mymedian5.f", this program is set up to compute the RMS of vector differences in a cell-by-cell basis (aka NOT a median filter at all, but a true RMS representation of of disagreements)

A "value" is the differential vector of:

```
interpolated-from-grid minus true
```

For any cell with at least ONE point with a value, the following is done:

1. Compute the average latitude of all points in the cell
2. Compute the average longitude of all points in the cell
3. Compute the RMS of all values in the cell

The output vector will then reflect these three values.

For latitude and ellipsoid height, the azimuth of the vector will ALWAYS be 0.0 (pointing up) while for longitude it will always be 90.0 (pointing right). However, these are mere conventions as they are not directional vectors anyway, but rather quanta which will be gridded and it is the grid which is of utmost importance.

No PIDS will be in the output files, as the output RMS vectors are not reflective of any one point, but rather a cell-wide conglomeration of information.

See DRU-11, p. 130

Program arguments

Arguments are newline terminated and read from standard input

They are enumerated here

Parameters

oldtm	Source Datum
newdtm	Target Datum,region
region	Conversion Region
agridsec	Grid Spacing in arcsec

Example:

```
olddatum = 'ussd'
newdatum = 'nad27'
region = 'conus'
agridsec = '900'
```

Program Inputs:

Changelog

2016 01 21:

Updated to RETURN to an old way of registering RMS vectors at AVE lat/lon rather than center of cell.

2015 10 28

Updated to work with new naming scheme (see DRU-11, p. 150) and to adopt the central lat/lon for the RMS vectors, rather than ave lat/ave lon (see DRU-11, p.145), and to also set up the gridding of RMS vectors at the T=0.9 level (see DRU-11, p. 148)

Also added "donzd" functionality to help control the magnitude of the Length of Reference Vector on Ground variables.

Also, added a section at the end to create the TOTAL error grids. by RMS-combining the "method noise grid" (the "d3" grid) with the "data noise grid" (the "rdd...09" grid) into one single "transformation error grid"

2016 08 25:

For reasons that are difficult to describe, "donzd" is now "onzd2.f" in /home/dru/Subs. Change and recompile...

2015 10 09:

Updated to add HOR vectors

2015 10 06:

Updated

2015 09 16:

Initial Release, For use in creating NADCON5 Built by Dru Smith

References getgridbounds(), indexxi(), and onzd2().

7.2 NADCON5 Core Library

Programs and Functions which are responsible for computing the grid transformations used to build NADCON5.

Directories

- directoryBinSource

  Folder containing NADCON5 Core Library Programs.
- directorySubs

  Folder containing NADCON5 Core Library Subroutines.

Functions

- program addem

  Part of the NADCON5 NADCON5 Core Library , adds one grid to another.
- program b2xyz

  Part of the NADCON5 NADCON5 Core Library , converts `*.b` *grid to* `xyz`
- program convlv

  Part of the NADCON5 NADCON5 Core Library , Convolves two grids.
- program decimate

  Part of the NADCON5 NADCON5 Core Library , Extract a reduced (1 of n) dataset.
- program gabs

  Part of the NADCON5 NADCON5 Core Library , Convert values in a `*.b` *grid to absolute value.*
- program gscale

  Part of the NADCON5 NADCON5 Core Library , Scales a grid by a factor.
- program gsqr

  Part of the NADCON5 NADCON5 Core Library , Squares values in a `*.b` *grid.*
- program gsqrt

  Part of the NADCON5 NADCON5 Core Library , Square Root of values in a `*.b` *grid.*
- program regrd2

  Part of the NADCON5 NADCON5 Core Library , regrid data.
- program subtrc

  Part of the NADCON5 NADCON5 Core Library , Subtract one grid from another.
- program xyz2b

  Part of the NADCON5 NADCON5 Core Library , Converts GMT `*.grd` *to a* `*.b` *NADCON style grid file.*
- subroutine bicubic (z, glamn, glomn, dla, dlo, nla, nlo, maxla, maxlo, xla, xlo, val)

  Subroutine to perform a 2-D cubic ("bicubic") interpolation.
- subroutine bilin (data, glamn, glomn, dla, dlo, nla, nlo, maxla, maxlo, xla, xlo, val)

  Subroutine to perform bilinear interpolation.
- subroutine biquad (z, glamn, glomn, dla, dlo, nla, nlo, maxla, maxlo, xla, xlo, val)

  Subroutine to perform a 2-D quadratic ("biquadratic") interpolation.
- subroutine bwplotcv (ele, fname, bw, be, bs, bn, jm, b1, b2, maxplots, olddtm, newdtm, region, elecap, ij, igridsec, fn)

  Subroutine to make GMT calls to do a B/W coverage plot.
- subroutine bwplotvc (ele, fname, bw, be, bs, bn, jm, b1, b2, maxplots, olddtm, newdtm, region, elecap, ij, xvlon, xvlat, xllon, xllat, lorvog, lorvopc, igridsec, fn)

  Subroutine to make GMT calls to do a B/W vector plot.
- subroutine coplot (ele, fname, bw, be, bs, bn, jm, b1, b2, maxplots, olddtm, newdtm, region, elecap, ij, cptlo, cpthi, cptin6, suffixused, igridsec, fn)

Subroutine to make GMT calls to do Color Raster Rendering of Gridded Data.

- subroutine coplotwcv (ele, fname, bw, be, bs, bn, jm, b1, b2, maxplots, olddtm, newdtm, region, elecap, ij, cptlo, cpthi, cptin6, suffixused, igridsec, fn, cvfname)

  Subroutine to make GMT calls to do a color raster rendering of gridded data, with coverage overlaid.
- subroutine cpt (ave, std, csm, xlo, xhi, xin)

  This subroutine generates the color pallette variables for a GMT color plot.
- subroutine cpt2 (med, csm, xlo, xhi, xin)

  This subroutine generates the color pallette variables for a GMT color plot.
- real function cubterp (x, f0, f1, f2, f3)

  This function fits a cubic function through four points.
- subroutine getgridbounds (region, xn, xs, xw, xe)

  Subroutine to collect up the GRID boundaries for use in creating NADCON 5.
- subroutine getmag (x, ix)

  Subroutine to return the magnitude of a double precision value.
- subroutine getmapbounds (mapflag, maxplots, region, nplots, olddtm, newdtm, bw, be, bs, bn, jm, b1, b2, fn, lrv, rv0x, rv0y, rl0y)

  Subroutine to collect up the MAP boundaries for use in creating NADCON 5.
- subroutine gridstats (fname, ave, std, med)

  Subroutine to print grid statistics to stdout.
- subroutine indexxd (n, nd, arr, indx)

  Subroutine to perform ?? indexing on floating point data (double precision)
- subroutine indexxi (n, nd, arr, indx)

  Subroutine to perform ?? indexing on integer data.
- integer ∗2 function iselect2 (k, n, arr, nmax)

  Function to select an element of a partially filled, but packed multi dimensional array, `integer*2`
- real ∗4 function onzd (x)

  Function to round a digit to one significant figure (one non zero digit), single precision.
- real ∗8 function onzd2 (x)

  Function to round a digit to one significant figure (one non zero digit), double precision.
- subroutine plotcoast (region, ifnum)

  Subroutine to write GMT-based commands to create a shoreline Write GMT-based commands to create a shoreline based on region.
- real function qterp (x, f0, f1, f2)

  This function fits a quadratic function through 3 points.
- real ∗8 function select2 (k, n, arr, nmax)

  Function to select an element of a partially filled, but packed multi dimensional array, double precision.
- real ∗4 function select2 (k, n, arr, nmax)

  Function to select an element of a partially filled, but packed multi dimensional array, single precision.
- subroutine vecstats (fname, n)

  Subroutine to tell us how many thinned vectors were used to make a grid.

7.2.1 Detailed Description

Programs and Functions which are responsible for computing the grid transformations used to build NADCON5.

The programs here can be considered "Library" Programs, or functions, which do the raw work of computing various transformations and conversions from `*.b` NADCON binary grid to a `GMT` style `*.grd`

7.2.2 Function Documentation

7.2.2.1 program addem ( )

Part of the NADCON5 NADCON5 Core Library , adds one grid to another.

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infileA	First Input File Name
infileB	Second Input File Name
outfile	Output File Name of A+B

Program Inputs:

- `lin1` Input File A
- `lin2` Input File B
- `lout` Output File to Write A+B

7.2.2.2 program b2xyz ( )

Part of the NADCON5 NADCON5 Core Library , converts `*.b` grid to `xyz`

Program to convert standard "*.b" grid formatted data to a binary xyz (lon, lat, value) list, which can then be used by GMT for various things (like running the GMT routine "xyz2grd", to get a "*.grd" file, which is useful for plotting, etc)

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infile	Input File Name

Program Inputs:

- Input File defined by `infile`

Program Outputs:

- `temp.xyz`

7.2.2.3 subroutine bicubic (real∗4, dimension(maxla,maxlo) *z*, *glamn*, *glomn*, *dla*, *dlo*, *nla*, *nlo*, *maxla*, *maxlo*, *xla*, *xlo*, *val*)

Subroutine to perform a 2-D cubic ("bicubic") interpolation.

Performs interpolation at location "xla,xlo" off of grid "z", whose header information is the standard ".b" header information with additional inputs

Parameters

`in`	*z*	Input Grid
`in`	*glamn*	minimum latitude (real∗8 decimal degrees)
`in`	*glomn*	minimum longitude (real∗8 decimal degrees)
`in`	*dla*	latitude spacing (real∗8 decimal degrees)
`in`	*dlo*	longitude spacing (real∗8 decimal degrees)
`in`	*nla*	number of lat rows (integer∗4)
`in`	*nlo*	number of lon cols (integer∗4)
`in`	*maxla*	actual dimensioned size of "z" in rows
`in`	*maxlo*	actual dimensioned size of "z" in cols
`in`	*xla*	lat of pt for interpolation (real∗8 dec. deg)
`in`	*xlo*	lon of pt for interpolation (real∗8 dec. deg)
`out`	*val*	interpolated value (real∗8)

Method:

Fit a 4x4 window over the random point. Unless the point is less than one grid spacing from the edge of the grid, it will fall in the inner 2x2 cell, and the 4x4 cell will be centered upon that.

Thus, if our point of interest is the asterisk, the window will look like this:

```
.   .   .   .
.   .   .   .
.   .*  .   .
.   .   .   .
```

Referenced by checkgrid().

7.2.2.4 subroutine bilin (real∗4, dimension(maxla,maxlo) *data*, *glamn*, *glomn*, *dla*, *dlo*, *nla*, *nlo*, *maxla*, *maxlo*, *xla*, *xlo*, *val*)

Subroutine to perform bilinear interpolation.

Performs a bilinear interpolation at location `xla,xlo` off of grid `data`, whose header information is the standard `.b` header information

Parameters

in	*data*	Input Data assumed to be real∗4
in	*glamn*	minimum latitude (real∗8 decimal degrees) `*.b`
in	*glomn*	minimum longitude (real∗8 decimal degrees) `*.b`
in	*dla*	latitude spacing (real∗8 decimal degrees) `*.b`
in	*dlo*	longitude spacing (real∗8 decimal degrees) `*.b`
in	*nla*	number of lat rows (integer∗4) `*.b`
in	*nlo*	number of lon cols (integer∗4) `*.b`
in	*maxla*	actual dimensioned size of "data" in rows `*.b`
in	*maxlo*	actual dimensioned size of "data" in cols `*.b`
in	*xla*	lat of pt for interpolation (real∗8 dec. def)
in	*xlo*	lon of pt for interpolation (real∗8 dec. def)
out	*val*	interpolated value (real∗8)

Referenced by checkgrid().

7.2.2.5 subroutine biquad (real∗4, dimension(maxla,maxlo) *z*, *glamn*, *glomn*, *dla*, *dlo*, *nla*, *nlo*, *maxla*, *maxlo*, *xla*, *xlo*, *val*)

Subroutine to perform a 2-D quadratic ("biquadratic") interpolation.

Performs a biquadratic interpolation at location `xla,xlo` off of grid `z`, whose header information is the standard ".b" header information

Parameters

in	*z*	Input Grid
in	*glamn*	minimum latitude (real∗8 decimal degrees)
in	*glomn*	minimum longitude (real∗8 decimal degrees)
in	*dla*	latitude spacing (real∗8 decimal degrees)
in	*dlo*	longitude spacing (real∗8 decimal degrees)
in	*nla*	number of lat rows (integer∗4)
in	*nlo*	number of lon cols (integer∗4)
in	*maxla*	actual dimensioned size of "z" in rows
in	*maxlo*	actual dimensioned size of "z" in cols
in	*xla*	lat of pt for interpolation (real∗8 dec. deg)
in	*xlo*	lon of pt for interpolation (real∗8 dec. deg)
out	*val*	interpolated value (real∗8)

Method:

Fit a 3x3 window over the random point. The closest 2x2 points will surround the point. But based on which quadrant of that 2x2 cell in which the point falls, the 3x3 window could extend NW, NE, SW or SE from the 2x2 cell.

Referenced by checkgrid().

7.2.2.6 subroutine bwplotcv (character∗3 *ele,* character∗200 *fname,* real∗8, dimension(maxplots) *bw,* real∗8, dimension(maxplots) *be,* real∗8, dimension(maxplots) *bs,* real∗8, dimension(maxplots) *bn,* real∗4, dimension(maxplots) *jm,* real∗4, dimension(maxplots) *b1,* real∗4, dimension(maxplots) *b2,* integer∗4 *maxplots,* character∗10 *olddtm,* character∗10 *newdtm,* character∗10 *region,* character∗3 *elecap,* *ij,* *igridsec,* character∗10, dimension(maxplots) *fn*)

Subroutine to make GMT calls to do a B/W coverage plot.

Changelog

2016 08 29:

Updated the `-R` and `-B` initial calls to 6 decimal places

2016 08 25:

`.gmtdefaults4` has been changed so X_ORIGIN is equal to 0.0 Center the plot with "-Xc" at first "psxy" call Remove all "-JM∗∗i+" references, and just use the actual width (jm) that came out of the "getmapbounds" routine and was sent here.

2016 07 21:

Modified use of JM command based on new forced sizes.

2015 02 15:

Updated to allow this subroutine to work earlier (in makeplotfiles01()), before `igridsec` was defined. See DRU-11, p. 139

References plotcoast().

Referenced by makeplotfiles01(), makeplotfiles02(), and makeplotfiles03().

7.2.2.7 subroutine bwplotvc (character∗3 *ele,* character∗200 *fname,* real∗8, dimension(maxplots) *bw,* real∗8, dimension(maxplots) *be,* real∗8, dimension(maxplots) *bs,* real∗8, dimension(maxplots) *bn,* real∗4, dimension(maxplots) *jm,* real∗4, dimension(maxplots) *b1,* real∗4, dimension(maxplots) *b2,* integer∗4 *maxplots,* character∗10 *olddtm,* character∗10 *newdtm,* character∗10 *region,* character∗3 *elecap,* *ij,* *xvlon,* *xvlat,* *xllon,* *xllat,* real∗8 *lorvog,* real∗8 *lorvopc,* *igridsec,* character∗10, dimension(maxplots) *fn*)

Subroutine to make GMT calls to do a B/W vector plot.

Changelog

2016 08 29:

Expanded the refence vector calls to be 6 decimal places, as well as the initial -R call for S/N/W/E and also the -B part of that call

2016 08 25:

`.gmtdefaults4` has been changed so X_ORIGIN is equal to 0.0 Center the plot with "-Xc" at first "psxy" call Remove all "-JM∗∗i+" references, and just use the actual width (jm) that came out of the "getmapbounds" routine and was sent here.

2016 07 29:

Updated the reference vector call to have the "-N" option, so it'll plot outside the map

2016 07 21:

Modified use of JM command based on new forced sizes.

2015 02 15:

Updated to allow this subroutine to work earlier (in makeplotfiles01()), before `igridsec` was defined. See DRU-11, p. 139

References plotcoast().

Referenced by checkgrid(), makeplotfiles01(), makeplotfiles02(), and makeplotfiles03().

7.2.2.8 program convlv ( )

Part of the NADCON5 NADCON5 Core Library , Convolves two grids.

Convolves one grid against another

$$c(i,j) = a(i,j) * b(i,j)$$

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infileA	First Input File Name
infileB	Second Input File Name
outfile	Output File Name of A∗B

Program Inputs:

- `lin1` Input File A
- `lin2` Input File B
- `lout` Output File to Write A+B

7.2.2.9 subroutine coplot (character∗3 *ele,* character∗200 *fname,* real∗8, dimension(maxplots) *bw,* real∗8, dimension(maxplots) *be,* real∗8, dimension(maxplots) *bs,* real∗8, dimension(maxplots) *bn,* real∗4, dimension(maxplots) *jm,* real∗4, dimension(maxplots) *b1,* real∗4, dimension(maxplots) *b2,* integer∗4 *maxplots,* character∗10 *olddtm,* character∗10 *newdtm,* character∗10 *region,* character∗3 *elecap,* *ij,* *cptlo,* *cpthi,* *cptin6,* character∗200 *suffixused,* *igridsec,* character∗10, dimension(maxplots) *fn*)

Subroutine to make GMT calls to do Color Raster Rendering of Gridded Data.

Changelog

2016 09 08:

Had to up the D_FORMAT default to %.3G because tight scalebar ranges with the newly allowed "more free" average values were showing repeating values when only 2 digits could be shown.

2016 09 07:

Had to add lines pre/post "makecpt" to change the D_FORMAT. This is because I *had* been forcing the "scave" in cpt.f to be ONZD. But that yielded bad values sometimes, so I switched it. With that switch, the scave could have lots of digits. Well, that means the newly adopted "D_FORMAT" of %.2G was insufficient for the CPT table. Who knew that D_FORMAT affected that! Anyway, so change "D_FORMAT" pre/post all makecpt calls.

2016 08 30:

See item #39 in Google ToDo list Changed "grdcontour" to have a blank "-R" call so it'll mimic whatever decimal places are in the "grdimage" call that came before it.

2016 08 29:

Updated the initial -R and -B calls to 6 decimal places

2016 08 25:

-.gmtdefaults4 has been changed so X_ORIGIN is equal to 0.0

- Center the plot with "-Xc" at grdimage
- Center the scalebar by setting its Xcoordinate, which runs in "plot frame" coordinates (0/0 at lower left) , to be equal to "jm/2"
- Force the scale bar to be exactly 4 inches wide, always
- Change the format for "makecpt" from 0.6 to 0.10

2016 07 29:

Update to put more data into comment/echo

Also,

- forced the -Ctemp.cpt option in "grdcontour" to make the contours line up with the color palette
- For d3 plots, to drop all contours
- For d3 plots to only use "coverage" part *only* if "ij" is not "1"
- Same for "09" and "ete" grids

2016 07 21:

- Set the "JM" code in "grdcontour" to just be "-JM" and let it therefore run with whatever JM size is used in "grdimage"
- Set the "A" code in "grdcontour" to be "-A-" which should turn off the labels on all contours
- Fixed size of scale bar

2016 03 01:

1. Changed to a continuous color plot
    - Get rid of "-Z" in "makecpt"
2. Changed to an 8 color, from 6 color, plot (without changing the RANGE yet...see DRU-12, p. 19)
    - Change varible that comes in from "cptin" to "cptin6"
    - Compute cptin = cptin6 ∗ 0.75d0 immediately

Update 2016 02 29:

1. Removed all shading from color plots
    - Get rid of "grdgradient" call
    - Remove from "grdimage" the "-Itempi.grd" part
    - Remove the "rm -f tempi.grd" line

References plotcoast().

Referenced by makeplotfiles02(), and makeplotfiles03().

7.2.2.10 subroutine coplotwcv (character∗3 *ele,* character∗200 *fname,* real∗8, dimension(maxplots) *bw,* real∗8, dimension(maxplots) *be,* real∗8, dimension(maxplots) *bs,* real∗8, dimension(maxplots) *bn,* real∗4, dimension(maxplots) *jm,* real∗4, dimension(maxplots) *b1,* real∗4, dimension(maxplots) *b2,* integer∗4 *maxplots,* character∗10 *olddtm,* character∗10 *newdtm,* character∗10 *region,* character∗3 *elecap,* *ij,* *cptlo,* *cpthi,* *cptin6,* character∗200 *suffixused,* *igridsec,* character∗10, dimension(maxplots) *fn,* character∗200 *cvfname*)

Subroutine to make GMT calls to do a color raster rendering of gridded data, with coverage overlaid.

Changelog

2016 09 08:

Had to up the D_FORMAT default to %.3G because tight scalebar ranges with the newly allowed "more free" average values were showing repeating values when only 2 digits could be shown.

2016 09 07:

Had to add lines pre/post "makecpt" to change the D_FORMAT. This is because I *had* been forcing the "scave" in cpt.f to be ONZD. But that yielded bad values sometimes, so I switched it. With that switch, the scave could have lots of digits. Well, that means the newly adopted "D_FORMAT" of %.2G was insufficient for the CPT table. Who knew that D_FORMAT affected that! Anyway, so change "D_FORMAT" pre/post all makecpt calls.

2016 08 30:

See item #39 in Google ToDo list Changed "grdcontour" to have a blank "-R" call so it'll mimic whatever decimal places are in the "grdimage" call that came before it.

2016 08 29:

Updated the initial -R and -B calls to 6 decimal places

2016 08 25:

- .gmtdefaults4 has been changed so X_ORIGIN is equal to 0.0
- Center the plot with "-Xc" at grdimage
- Center the scalebar by setting its Xcoordinate, which runs in "plot frame" coordinates (0/0 at lower left) , to be equal to "jm/2"
- Force the scale bar to be exactly 4 inches wide, always
- Change the format for "makecpt" from 0.6 to 0.10

2016 07 29:

- Update to put more data into comment/echo
- Forced grdcontour with -Ctemp.cpt to align contours with color palette

2016 07 28:

- For d3 plots, to drop all contours
- For d3 plots to only use "coverage" part *only* if "ij" is not "1"
- Same for "09" and "ete" grids

2016 07 21:

- Set the "JM" code in "grdcontour" to just be "-JM" and let it therefore run with whatever JM size is used in "grdimage"
- Set the "A" code in "grdcontour" to be "-A-" which should turn off the labels on all contours
- Fixed the size of the scale bar

2016 03 01:

1. Changed to a continuous color plot
    - Get rid of "-Z" in "makecpt"
2. Changed to an 8 color, from 6 color, plot (without changing the RANGE yet...see DRU-12, p. 19)
    - Change varible that comes in from "cptin" to "cptin6"
    - Compute cptin = cptin6 $*$ 0.75d0 immediately

2016 02 29:

1. Removed all shading from color plots
    - Get rid of "grdgradient" call
    - Remove from "grdimage" the "-Itempi.grd" part
    - Remove the "rm -f tempi.grd" line

References plotcoast().

Referenced by makeplotfiles02().

7.2.2.11 subroutine cpt (real∗8 *ave,* real∗8 *std,* real∗8 *csm,* real∗8 *xlo,* real∗8 *xhi,* real∗8 *xin*)

This subroutine generates the color pallette variables for a GMT color plot.

Parameters

`in`	*ave*	Average of the gridded data
`in`	*std*	Standard deviation of the gridded data
`in`	*csm*	Color Sigma Multiplier (how many sigmas on each side of the average do you want the colors to range?)
`out`	*xlo*	Low value
`out`	*xhi*	High value
`out`	*xin*	Interval

Changelog

2016 09 06:

Modified because the forcing of "scave" to be one non zero digit was throwing off the scalebar so far in Guam that the data in Guam wasn't even plotting. Change to make the interval still be one non zero digit, but then a simpler formula for the scaled average was put in.

2016 07 29:

Modified from original version to reflect "new math" invented this week that helps shrink the color bar and/or widen the color bar (see issues 14 and 15 in DRU-12, p. 48)

Referenced by makeplotfiles02().

7.2.2.12 subroutine cpt2 (real∗8 *med,* real∗8 *csm,* real∗8 *xlo,* real∗8 *xhi,* real∗8 *xin*)

This subroutine generates the color pallette variables for a GMT color plot.

This particular routine is best for data that are all positive, but cluster near a small value while having a lot of outliers to the high-side. The color plot uses the MEDIAN (and a multiplier) to set the upper limit, while forcing the lower limit to be ZERO.

Parameters

`in`	*med*	Median of the gridded data
`in`	*csm*	Color Sigma Multiplier (The maximum color range will be based on csm∗med. The minimum color range will be zero)
`out`	*xlo*	Low value
`out`	*xhi*	High value
`out`	*xin*	Interval

Referenced by makeplotfiles02(), and makeplotfiles03().

7.2.2.13 real function cubterp (*x, f0, f1, f2, f3*)

This function fits a cubic function through four points.

This function fits a cubic function through four *equally* spaced points along the x-axis at indices 0, 1, 2 and 3. The spacing along the x-axis is "dx"

Thus:

$$\begin{aligned} f0 = f_0 &= y(x_0) \\ f1 = f_1 &= y(x_1) \\ f2 = f_2 &= y(x_2) \\ f3 = f_3 &= y(x_3) \end{aligned}$$

Where:

$$\begin{aligned} x_1 &= x_0 + dx \\ x_2 &= x_1 + dx \\ x_3 &= x_2 + dx \end{aligned}$$

The input value is some value of "x" that falls between 0 and 3. The output value (cubterp) is the cubic function at x.

Parameters

`in`	*x*	Compute Interpolation at this positon, a value between 0 and 3 it is scaled relative to `x_0` `x_3` and `dx`. For example, a value of 1.5 is `x_0 + 1.5*dx` which falls between `x1` and `x2`
`in`	*f0*	`y` value at `x_0`
`in`	*f1*	`y` value at `x_1 = x_0 + dx`
`in`	*f2*	`y` value at `x_2 = x_0 + dx`
`in`	*f3*	`y` value at `x_3 = x_0 + dx‘`

This function uses Newton-Gregory forward polynomial

$$\begin{aligned} \nabla f_0 &= f_1 - f_0 \\ \nabla f_1 &= f_2 - f_1 \\ \nabla^2 f_0 &= \nabla f_1 - \nabla f_0 \\ cubterp(x, f_0, f_1, f_2, f_3) &= f_0 + x \nabla f_0 + 0.5x\,(x - 1.0)\,\nabla^2 f_0 \end{aligned}$$

7.2.2.14 program decimate ( )

Part of the NADCON5 NADCON5 Core Library , Extract a reduced (1 of n) dataset.

Decimate - Extract `1` of every `n` points

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infile	Input File Name
outfile	Output File Name
ny	Latitude Decimation Ratio `1:ny`
nx	Longitude Decimation Ratio `1:nx`

Program Inputs:

- `lin` Input File A

Program Outputs:

- `lout` Output File to Write Decimated File

7.2.2.15 program gabs ( )

Part of the NADCON5 NADCON5 Core Library , Convert values in a `*.b` grid to absolute value.

Belongs to the suite of ".b" file manipulators

This program will convert every value in a ".b" grid to its absolute value.

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infile	Input File Name
outfile	Output File Name

Program Inputs:

- `lin` Input File (`*.b` grid)

Program Outputs:

- `lout` Output File (`*.b` grid)

7.2.2.16 subroutine getgridbounds (character∗10 *region,* real∗8 *xn,* real∗8 *xs,* real∗8 *xw,* real∗8 *xe*)

Subroutine to collect up the GRID boundaries for use in creating NADCON 5.

This CAN BE different than the MAP boundaries as such:

GRID boundaries will be just four values (n/s/w/e) for any region

MAP boundaries will allow multiple maps to be made and may or may not align with the GRID boundaries. Used to allow for more "close up" maps and such, without the need to screw up the MAP boundaries.

Parameters

in	*region*	Region Name
out	*xn*	north boundary for this region
out	*xs*	south boundary for this region
out	*xw*	west boundary for this region
out	*xe*	east boundary for this region

Subroutine Input Files:

- ‘Data/grid.parameters

Referenced by checkgrid(), makework(), mymedian5(), and myrms().

7.2.2.17 subroutine getmag (*x,* *ix*)

Subroutine to return the magnitude of a double precision value.

Parameters

out	*x*	result, magnitude of ix
in	*ix*	input douple precision

7.2.2.18 subroutine getmapbounds (character∗1 *mapflag,* *maxplots,* character∗10 *region,* *nplots,* character∗10 *olddtm,* character∗10 *newdtm,* real∗8, dimension(maxplots) *bw,* real∗8, dimension(maxplots) *be,* real∗8, dimension(maxplots) *bs,* real∗8, dimension(maxplots) *bn,* real∗4, dimension(maxplots) *jm,* real∗4, dimension(maxplots) *b1,* real∗4, dimension(maxplots) *b2,* character∗10, dimension(maxplots) *fn,* logical, dimension(maxplots) *lrv,* real∗8, dimension(maxplots) *rv0x,* real∗8, dimension(maxplots) *rv0y,* real∗8, dimension(maxplots) *rl0y*)

Subroutine to collect up the MAP boundaries for use in creating NADCON 5.

This CAN BE different than the GRID boundaries as such:

GRID boundaries will be just four values (n/s/w/e) for any region

MAP boundaries will allow multiple maps to be made and may or may not align with the GRID boundaries. Used to allow for more "close up" maps and such, without the need to screw up the MAP boundaries.

Parameters

`in`	*mapflag*	Map Generation Flag
`in`	*maxplots*	
`in`	*region*	region to get map bounds
`out`	*nplots*	number of plots generated
`in`	*olddtm*	source datum
`in`	*newdtm*	target datum
`out`	*bw*	western bound of plot (Array of length `maxplots`)
`out`	*be*	eastern bound of plot (Array of length `maxplots`)
`out`	*bs*	southern bound of plot (Array of length `maxplots`)
`out`	*bn*	northern bound of plot (Array of length `maxplots`)
`out`	*jm*	(Array of length `maxplots`)
`out`	*b1*	(Array of length `maxplots`)
`out`	*b2*	(Array of length `maxplots`)
`out`	*fn*	(Array of length `maxplots`)
`out`	*lrv*	(Array of length `maxplots`)
`out`	*rv0x*	(Array of length `maxplots`)
`out`	*rv0y*	(Array of length `maxplots`)
`out`	*rl0y*	(Array of length `maxplots`)

Version for NADCON 5 Built upon the original version used in GEOCON v2.0 Do not use with GEOCON v2.0

Broken down into sub-subroutines to make it easier to swap out when I make different choices.

Changelog

2016 08 29:

Taking in olddtm and newdtm now, and adding code to use that to filter out "Saint" regions in Alaska when plotting transformations not supported in those regions.

2016 08 26:

Used actual mercator projection math to compute the exact reference vector and label locations 1/2 inch and 3/4 inch respectively below the S/W corner of the plot.

2016 07 21:

Two new columns added to "map.parameters", which have the location of the reference vector. Return a logical "lrv" as true if there is an optional special location for the reverence vector. Return as false if not. If true, return lon/lat coords of ref vector origin in rv0x/rv0y. If false, return zeros in those fields.

Also, compute "jm" on the fly, ignoring what is in the table. All plots will now be forced PORTRAIT and forced no wider than 6" and no taller than 8", while maintaining proper X/Y ratios in a Mercator projection. That means, make the biggest plot possible, with the right ratio, that is neither wider than 6" nor taller than 8" and then, whatever the width of that largest plot is – return that width in the "jm" field.

Referenced by checkgrid(), makeplotfiles01(), makeplotfiles02(), and makeplotfiles03().

7.2.2.19 subroutine gridstats (character∗200 *fname*, *ave*, *std*, real∗8 *med*)

Subroutine to print grid statistics to stdout.

Parameters

`in`	*fname*	name of grid stat file
`out`	*ave*	average
`out`	*std*	standard deviatio
`out`	*median*	

Referenced by makeplotfiles02(), and makeplotfiles03().

7.2.2.20 program gscale ( )

Part of the NADCON5 NADCON5 Core Library , Scales a grid by a factor.

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infile	Input File Name
factor	Scaling Factor
outfile	Output File Name

Program Inputs:

- `lin1` Input File

Program Outputs:

- `lout` Output File

7.2.2.21 program gsqr ( )

Part of the NADCON5 NADCON5 Core Library , Squares values in a `*.b` grid.

Belongs to the suite of ".b" file manipulators

This program will convert every value in a ".b" grid to its squared value.

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infile	Input File Name
outfile	Output File Name

Program Inputs:

- `lin` Input File (`*.b` grid)

Program Outputs:

- `lout` Output File (`*.b` grid)

7.2.2.22 program gsqrt ( )

Part of the NADCON5 NADCON5 Core Library , Square Root of values in a `*.b` grid.

Belongs to the suite of ".b" file manipulators

This program will convert every value in a ".b" grid to its square-root value.

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infile	Input File Name
outfile	Output File Name

Program Inputs:

- `lin` Input File (`*.b` grid)

Program Outputs:

- `lout` Output File (`*.b` grid)

7.2.2.23 subroutine indexxd (integer *n*, integer *nd*, real*8, dimension(nd) *arr*, integer, dimension(nd) *indx*)

Subroutine to perform ?? indexing on floating point data (double precision)

Parameters

`in`	*n*	number of iterations (rows?)
`in`	*nd*	array and index dimensions
`in`	*arr*	input data array
`out`	*indx*	index out

Changelog

1/8/2004:

Modified to allow `indx` and `arr` to be DIMENSIONED differently than the number of good values they contain

11/7/2003

Modified to REAL∗8 by D. Smith,

Referenced by mymedian5().

7.2.2.24 subroutine indexxi (integer *n,* integer *nd,* integer∗4, dimension(nd) *arr,* integer, dimension(nd) *indx*)

Subroutine to perform ?? indexing on integer data.

Parameters

`in`	*n*	number of iterations (rows?)
`in`	*nd*	array and index dimensions
`in`	*arr*	input data array
`out`	*indx*	index out

Changelog

2/5/2013:

Modified by D. Smith. Arr has been changed to integer∗4. And like other versions of "indexx" which I've modified, I allow indx and arr to be DIMENSIONED differently than the number of good values they contain

Parameters

nd	Modified by D. Smith, 2/5/2013 . Arr has been changed to integer∗4. And like other versions of "indexx" which I've modified, I allow indx and arr to be DIMENSIONED differently than the number of good values they contain

Referenced by mymedian5(), and myrms().

7.2.2.25 integer∗2 function iselect2 (integer *k,* integer *n,* integer∗2, dimension(nmax) *arr,* integer *nmax*)

Function to select an element of a partially filled, but packed multi dimensional array, `integer*2`

Finds the "kth" element of an array, "arr", which is dimensioned to be "nmax" values long, but which only has data in the first "n" cells.

Changelog

1/14/2016:

Like "select2" but modified by D. Smith to allow an "nmax" array given, but which only has values in elements 1-n, and to have "arr" be Integer∗2

7.2.2.26 real∗4 function onzd (real∗4 *x*)

Function to round a digit to one significant figure (one non zero digit), single precision.

Function "onzd" stands for "One Non Zero Digit"

It takes a Real∗4 number as input, and rounds that number to the closest number containing only 1 non-zero digit. The list of such numbers is infifinite, but contain these, in order:

```
0.7 , 0.8 , 0.9 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9, 10 , 20 , 30 , etc etc
```

Parameters

`in`	*x*	input value

Returns

`real*4` rounded value of x to one non zero digit

Examples of input/output are:

```
   0.000019      =>    0.000020
   0.007432      =>    0.007000
   1.7           =>    2.000000
   9.143         =>    9.000000
  17.4           =>   20.000000
 947.3           =>  900.000000
 987.432         => 1000.000000
1014.8           => 1000.000000
1502.7           => 2000.000000
```

7.2.2.27 real∗8 function onzd2 (real∗8 *x*)

Function to round a digit to one significant figure (one non zero digit), double precision.

Function "onzd" stands for "One Non Zero Digit"

Version 2 operates just like version 1 (onzd()), only the input and output will be real∗8 values, not real∗4.

It takes a Real∗8 number as input, and rounds that number to the closest number containing only 1 non-zero digit. The list of such numbers is infifinite, but contain these, in order:

```
0.7 , 0.8 , 0.9 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9, 10 , 20 , 30 , etc etc
```

Parameters

`in`	*x*	input value

Returns

`real*8` rounded value of x to one non zero digit

Examples of input/output are:

```
   0.000019      =>    0.000020
   0.007432      =>    0.007000
   1.7           =>    2.000000
   9.143         =>    9.000000
  17.4           =>   20.000000
 947.3           =>  900.000000
 987.432         => 1000.000000
1014.8           => 1000.000000
1502.7           => 2000.000000
```

Referenced by checkgrid(), makeplotfiles01(), makeplotfiles02(), makeplotfiles03(), and myrms().

7.2.2.28 subroutine plotcoast (character∗10 *region*, *ifnum*)

Subroutine to write GMT-based commands to create a shoreline Write GMT-based commands to create a shoreline based on region.

Use GMT-default coastline for:

- `conus`
- `alaska`

Use Dru's custome coastline for:

- hawaii
- prvi
- as
- guamcnmi
- stlawrence
- stmatthew
- stgeorge
- stpaul

Parameters

`in`	*region*	The Region to create coastline for
`in`	*ifnum*	the file descriptor of the output file to write `GMT` commands to

Changelog

2016 01 07:

Forced the Alaska region to plot the islands of St. George, St. Matthew and St. Paul (St. Lawrence is already plotted), as well as 35 missing Aleutian Islands

2015 09 23:

Added four new regions:

- St. Lawrence Island, Alaska
- St. Matthew Island, Alaska
- St. George Island, Alaska
- St. Paul Island, Alaska

Referenced by bwplotcv(), bwplotvc(), coplot(), and coplotwcv().

7.2.2.29 real function qterp (*x,* *f0,* *f1,* *f2*)

This function fits a quadratic function through 3 points.

This function fits a parabola (quadratic) function through three *equally* spaced points along the x-axis at indices 0, 1, and 2. The spacing along the x-axis is "dx"

Thus:

$$\begin{aligned} f0 = f_0 &= y(x_0) \\ f1 = f_1 &= y(x_1) \\ f2 = f_2 &= y(x_2) \end{aligned}$$

Where:

$$\begin{aligned} x_1 &= x_0 + dx \\ x_2 &= x_1 + dx \\ x_3 &= x_2 + dx \end{aligned}$$

The input value is some value of "x" that falls between 0 and 2. The output value (qterp) is the quadratic function at x.

Parameters

in	*x*	Compute Interpolation at this positon, a value between 0 and 3 it is scaled relative to `x_0` `x_2` and `dx`. For example, the value of 1.5 is `x_0 + 1.5*dx` which falls between `x1` and `x2`
in	*f0*	`y` value at `x_0`
in	*f1*	`y` value at `x_1 = x_0 + dx`
in	*f2*	`y` value at `x_2 = x_0 + dx`

Returns

`real` quadratically interpolated value of `f(x*)` where `x* = x_0 + x*dxx`

This function uses Newton-Gregory forward polynomial

$$\begin{aligned}\nabla f_0 &= f_1 - f_0\\ \nabla f_1 &= f_2 - f_1\\ \nabla^2 f_0 &= \nabla f_1 - \nabla f_0\\ qterp(x, f_0, f_1, f_2) &= f_0 + x\nabla f_0 + 0.5x\,(x - 1.0)\,\nabla^2 f_0\end{aligned}$$

7.2.2.30 program regrd2 ( )

Part of the NADCON5 NADCON5 Core Library , regrid data.

Regrid gridded data using biquadratic interpolation

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infile	Input Master Grid File Name
outfile	Output Regrid File Name
nrow	Number of rows in new Grid (Lat)
ncol	Number of cols in new Grid (Lon)

Program Inputs:

- `lin` Input File

Program Outputs:

- `lout` Output File

7.2.2.31 real∗8 function select2 (integer *k,* integer *n,* real∗8, dimension(nmax) *arr,* integer *nmax*)

Function to select an element of a partially filled, but packed multi dimensional array, double precision.

Finds the "kth" element of an array, "arr", which is dimensioned to be "nmax" values long, but which only has data in the first "n" cells.

Changelog

7/17/2008:

Like "select2" but modified by D. Smith to allow an "nmax" array given, but which only has values in elements 1-n, and to have "arr" be Integer∗2

7.2.2.32 real∗4 function select2 (integer *k,* integer *n,* real∗4, dimension(nmax) *arr,* integer *nmax*)

Function to select an element of a partially filled, but packed multi dimensional array, single precision.

Finds the "kth" element of an array, "arr", which is dimensioned to be "nmax" values long, but which only has data in the first "n" cells.

Changelog

7/17/2008:

Like "select2" but modified by D. Smith to allow an "nmax" array given, but which only has values in elements 1-n, and to have "arr" be Integer∗2

7.2.2.33 program subtrc ( )

Part of the NADCON5 NADCON5 Core Library , Subtract one grid from another.

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infileA	First Input File Name
infileB	Second Input File Name
outfile	Output File Name of A∗B

Program Inputs:

- `lin1` Input File A
- `lin2` Input File B
- `lout` Output File to Write A-B

7.2.2.34 subroutine vecstats (character∗200 *fname*, integer∗4 *n*)

Subroutine to tell us how many thinned vectors were used to make a grid.

Parameters

`in`	*fname*	vector filename to read
`out`	*n*	number of thinned vectors

Referenced by makeplotfiles02().

7.2.2.35 program xyz2b ( )

Part of the NADCON5 NADCON5 Core Library , Converts GMT `∗.grd` to a `∗.b` NADCON style grid file.

Turn gmt/netcdf grd dump into my grid file (real number version) assumes grd dump is longitude/latitude/real (binary s.p.)

Program arguments

Arguments are newline terminated and read from standard input

When run from the command line, the program prints a prompt for each argument

They are enumerated here

Parameters

infile	Input File Name
outfile	Output File Name

Program Inputs:

- `lin` Input File (`∗.grd`)

Program Outputs:

- `lout` Output File (`∗.b`)

Index