GRIDS User's Guide

Version 7.2

Patrick J. Haley Jr.

January 16, 2001

Table of Contents

List of Tables . . . . **iv**

List of Figures . . . . **v**

1: THE BASICS **1**

1.1: Introduction 1

1.2: Installation 1

1.3: Additional Software/Libraries 1

1.3.1: HOPS Software 2

1.3.2: non–HOPS Software 2

1.4: Conventions 2

1.4.1: Lat–Lon 2

1.4.2: Vertical 2

1.4.3: Overwriting 2

1.5: Auxiliary files 2

1.6: New in version 7.2 3

2: DOMAINS **4**

2.1: Introduction 4

2.2: Horizontal Coordinates 4

2.3: Vertical Coordinates 6

2.4: Discretization 8

2.4.1: Horizontal 8

2.4.2: Vertical 8

2.5: Computational Issues 8

2.5.1: 2–way nesting 8

2.5.2: The hydrostatic consistency factor 9

3: THE GNUMAKEFILE **10**

3.1: Directory Paths 10

3.2: Executable File Names 10

3.3: FORTRAN Compilation Flags (`FFLAGS`) 12

3.4: C Pre–Processing Flags (`CPPFLAGS`) 12

3.4.1: Topography manipulation 12

3.4.2: Step topography manipulation 12

3.4.3: Debugging 12

3.4.4: Time/Date ... 13
3.4.5: Miscellaneous ... 13
3.5: Archive Utility Flags (ARFLAGS) ... 13
3.6: GNUmakefile Installation Issues ... 14
3.7: Compiling with the GNUmakefile ... 14
4: THE INPUT FILE ... **16**
4.I: Grid Output File ... 16
4.II: Horizontal Grid Definition ... 16
4.III: Vertical Levels ... 18
4.IV: Vertical Coordinates ... 18
4.IV.a: Type ... 18
4.IV.b: Additional Discretization Data ... 18
4.V: Plot window & Topography manipulation ... 19
4.VI: Contouring control ... 20
4.VI.a: Topography ... 20
4.VI.b: Hydrostatic Consistency ... 21
4.VII: Plot flags & titles ... 22
4.VIII: Topography manipulation ... 22
4.IX: Additional vertical information ... 23
5: USING GRIDS ... **25**
5.1: Introduction ... 25
5.2: Horizontal Domain Definition ... 25
5.2.1: General Domain Selection ... 25
5.2.2: Nesting ... 25
5.2.3: Land Masking ... 25
5.3: Topographic Conditioning ... 26
5.3.1: Protecting the Terrain Following Coordinates ... 26
5.3.2: Controlling the Slope ... 27
5.3.3: Final Touch–Ups ... 28
5.3.4: Nesting ... 28
5.4: Vertical Coordinate System ... 28
5.4.1: Selecting the Vertical System ... 29
5.4.2: Vertical Resolution ... 29
5.4.3: Definition of Coordinate Interface ... 30
5.4.4: Nested Domains ... 30

6: GRIDS OUTPUT **31**

6.1: Introduction 31

6.2: ASCII Echo 31

6.2.1: Run & Model Identifier 31

6.2.2: Run Input 31

6.2.3: Domain Limits 31

6.2.4: Topography Report 31

6.2.5: Coordinate Interface 31

6.2.6: Representative Model Levels 34

6.2.7: Hydrostatic Consistency Factor 35

6.3: Graphical Output 35

6.4: NetCDF Output 37

6.4.1: Dimensions 37

6.4.2: Run Parameters 42

6.4.3: Vertical Discretization 43

6.4.4: Horizontal Grid 43

6.4.5: Topography 43

6.4.6: PE_mask additions 44

6.4.7: cond_topo additions 44

6.4.8: Global Attributes 44

Bibliography **45**

List of Tables

Dimension Parameters 3
Part of the header of the makefile for compiling GRIDS. 11
Horizontal Grid Parameters 16
Schematic of the input file for the GRIDS program. 17
Vertical Levels Parameters 18
Vertical System Parameters 18
Vertical Discretization Parameters 18
Plot & Topography Parameters 19
Sample Coast File 19
Topography Contouring Parameters 21
User Contours (any interval) 21
User Contours (constant interval) 21
Plot Titles 22
Shapiro Filter Parameters (topography) 22
Clipping Parameters (topography) 23
Median Filter Parameters (topography) 23
Double σ Interface Parameters 23
Median Filter Parameters (step topography) 24
Mean Filter Parameters (step topography) 24
Sample ASCII Echo output produced by GRIDS. 32
GRIDS netCDF dimensions 37
PE_mask netCDF dimensions 37
Sample netCDF output produced by GRIDS. 38
cond_topo netCDF dimensions 42
Input Parameters 42
Terrain Following Input 43

List of Figures

Comparison of horizontal domain types. 5
Comparison of vertical coordinate systems. 7
Grid indexing. 8
Topographic Conditioning. 27
Types of Plots Produced by GRIDS. 36

CHAPTER 1: THE BASICS

1.1: Introduction

The GRIDS package enables the user to design a model domain for the Harvard Ocean Prediction System (HOPS). In particular, the GRIDS package allows the user to

- set–up and view the horizontal extent and resolution of a domain
  - also define horizontally collocated, nested sub–domains
- define the vertical discretization
- extract topography from a gridded, evenly spaced database
  - also extract topography between GRIDS files in nested domain configurations.
- Condition the extracted topography for improved simulations
  - clipping high & low values
  - filtering

The GRIDS package is split–up into three programs

`grids`	The workhorse program. It performs most of the described operations.
`crs2fne`	A utility to help design collocated nested sub-domains.
`extract_grids`	A utility to align topographic & masking data between one GRIDS file and a nested sub-domain GRIDS file.

This document is split–up into six chapters

(1) This chapter. Contains the basic information to get started.
(2) The domains chapter, describes what the GRIDS package is trying to accomplish.
(3) The GNUmakefile chapter, describes how to compile the GRIDS package.
(4) The input chapter, describes the input data required by the GRIDS package.
(5) The usage chapter, describes how to use the GRIDS package.
(6) The output chapter, describes the GRIDS output.

1.2: Installation

This package is available over the INTERNET via anonymous FTP from `maelstrom.harvard.edu` `(128.103.2.50)`. When connected, you will be in the FTP directory. To obtain this package, go to the directory `pub/HOPS/Grids` and get the following files:

`Readme.grids`	ASCII text of basic information.
`grids_7.2.tar.Z`	Compressed (UNIX `compress`) tar file of the GRIDS package.
`grids.ps.gz`	Compressed (GNU `gzip`) current version of this document.
`Ex_grids_7.2.tar.gz`	Compressed (GNU `gzip`) tar file of the GRIDS Examples.

To install this package, simply go to the directory in which you wish to put the GRIDS package and execute the following command:

```
zcat grids_7.2.tar.Z | tar -pxvf -
```

Similarly, to install example(s) associated with this package, simply go to the directory in which you wish to put the examples and execute the following command:

```
gzip -dc Ex_grids_7.2.tar.gz | tar -pxvf -
```

1.3: Additional Software/Libraries

The GRIDS package requires some additional software to function and is helped along by some more. This section describes the software, grouped according to whether it is found in the HOPS distribution site or not.

1.3.1: HOPS Software

PE_mask *required for coastal grids.* This software package creates land masks in GRIDS files. It comes in two versions, a FORTRAN version which is cumbersome to use and a MATLAB version which is easy to use but requires MATLAB.

cond_topo *suggested.* This software provides controlled conditioning of the topography. Only in MATLAB format.

1.3.2: non–HOPS Software

netCDF *required.* This freeware library produces the data format in which GRIDS writes its results. For further information see `http://www.unidata.ucar.edu/`

GNUmake *required.* A freeware alternative to UNIX make, provides standard functionality across platforms. For further information see `http://www.gnu.ai.mit.edu/`

NCAR *suggested.* A proprietary graphics library. Used by GRIDS to display its results. GRIDS can be compiled to not display the results (§3.4.5), but that makes domain designing a blind processes. For further information see `http://www.scd.ucar.edu/`

MATLAB *suggested.* A proprietary programming environment. Used by PE_mask for graphical input and cond_topo for examining intermediate results. For further information see `http://www.mathworks.com/`

NetCDF Toolbox *suggested.* A freeware netCDF interface for MATLAB. Used by PE_mask and cond_topo for input/output. For further information see `http://crusty.er.usgs.gov/~cdenham`

1.4: Conventions

1.4.1: Lat–Lon

GRIDS follows the standard convention in defining the latitude-longitude system.

<u>Latitude</u> Latitudes are referenced from the equator. Northern latitudes are positive and southern latitudes are negative.

<u>Longitude</u> Longitudes are referenced from the prime meridian. Eastern longitudes are positive and Western longitudes are negative.

1.4.2: Vertical

GRIDS follows the standard convention that elevations above sea-level are positive and depths below sea-level are negative.

1.4.3: Overwriting

GRIDS is designed to overwrite files with reckless abandon. Live with it.

1.5: Auxiliary files

Include files

There are a number of include files which contain various parameter definitions and common blocks for GRIDS. Of these, only one is of importance to the casual user:

gridsparm.h This file contains the definitions for the array sizes. Over dimensioning is allowed.

Table 1: **Dimension Parameters**

Parameter	**Description**	**Type**
`dimi`	Dimension of contour integer work space.	Integer
`dimm`	Dimension of graphics area map work space.	Integer
`dimr`	Dimension of contour real work space.	Integer
`ijmx`	Maximum of `imt` and `jmt`.	Integer
`imt`	Number of grid points in x (longitudinal) direction.	Integer
`jmt`	Number of grid points in y (latitudinal) direction.	Integer
`max_cl`	Maximum number of individually specified contour levels.	Integer
`mxcst`	Maximum number of points in a coastline.	Integer
`max_lvl`	Maximum number of vertical levels.	Integer

The parameter `dimm` comes into play when contour lines are blocked from crossing high/low markers (see §4.VI). The parameter `mxcst` is only important when using the netCDF versions of the GSHHS coastal dataset (see §4.V).

1.6: New in version 7.2

- *Standardization* Numerous instances of SUN–tolerated non–ANSI coding have been eliminated. There are 4 categories of non–ANSI codings which are retained.
  - symbolic name(s) longer than 6 characters
  - lower case letters
  - nonstandard form of the DO statement
  - dollar sign or underscore in an identifier

- *Bugs* Some minor coding errors which should not have appeared in the previous version have been eliminated.

CHAPTER 2: DOMAINS

2.1: Introduction

This chapter describes the domain definitions and discretizations. It is written in four parts

(1) The types of horizontal grids supported
(2) The types of vertical grids supported
(3) The types of grids discretizations
(4) Some computational issues

2.2: Horizontal Coordinates

There are three types of horizontal domains currently supported by GRIDS: Cartesian, spherical and rotated spherical (figure 1)

(1) Cartesian: The Cartesian grids are local tangent plane approximations to a sphere. Best used for small regions where spherical effects are relatively unimportant. The Cartesian transformation is defined by three parameters:

- (lat_c,lon_c) The "center" of the transformation. Geographic coordinates are converted to Cartesian based on the distance to this point.
- θ_{rot} The rotation angle. The Cartesian coordinates, (x,y), are rotated from their geographic counterparts, (lon,lat), by the angle θ_{rot}.

The Cartesian domains are defined with six additional parameters

- (nx,ny) The number of grid points in each direction.
- (dx,dy) The grid spacing in each direction.
- (Δx,Δy) The offset (meters) between the center of the domain and the center of the transformation.

(2) Spherical: The spherical grids are geographic latitudes/longitudes. Best used where spherical effects are important and it is more convenient to keep geographic orientation (e.g. If modeling the Southern Ocean, geographic orientation enables periodic boundary conditions). There is no transformation, so lat_c, lon_c and θ_{rot} are all zero. The domains are defined with six parameters:

- (nx,ny) The number of grid points in each direction.
- (dx,dy) The grid spacing in each direction.
- (Δx,Δy) The offset (degrees) between the center of the domain and the center of the transformation. For the spherical grid, (Δx,Δy) become the (lon,lat) of the center of the grid.

(3) Rotated Spherical: A spherical grid in which the equator had been moved to reduce model grid distortions. Best used for regions where spherical effects are important and reducing model grid distortions out weighs any benefit of maintaining geographic orientation. The rotated spherical transformation is defined by three parameters:

- (lat_c,lon_c) The "center" of the transformation. The (0,0) point of the rotated spherical system.
- θ_{rot} The rotation angle. The angle between the rotated coordinates and the geographic coordinates at the "center" of the transformation.

The rotated spherical domains are defined with six additional parameters

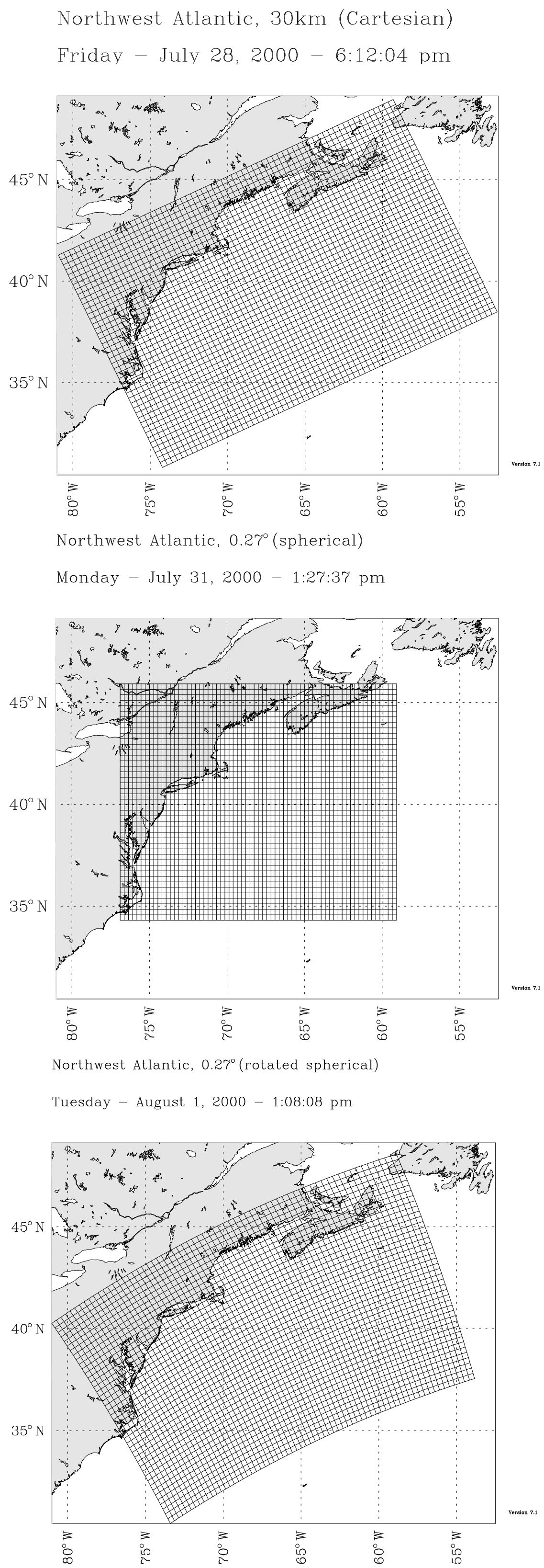

Figure 1: Comparison of horizontal domain types. All grids have the same number of points in each direction and approximately the same grid spacing (0.27° latitudinal spacing $\sim$ 30km)

(nx,ny) The number of grid points in each direction.

(dx,dy) The grid spacing in each direction.

(Δx,Δy) The offset (degrees) between the center of the domain and the center of the transformation.

2.3: Vertical Coordinates

The HOPS dynamical model (Primitive Equation) supports three variants of a terrain following coordinate system in the vertical (figure 2).

(1) σ Coordinates: The depth of a model level is a fixed fraction of the total water column

$$z_{i,j,k} = \sigma_k * h_{i,j}$$

The σ coordinates are most useful in regions with high topography and where maintaining near surface resolution is not, necessarily, critical

(2) Hybrid Coordinates: A generalization of (1). The top k_c levels are flat, the remainder are terrain following.

$$z_{i,j,k} = \begin{cases} z_k & k \le k_c \\ z_c + \sigma_k * (h_{i,j} - z_c) & k > k_c \end{cases}$$

Where z_c is the depth of the flat interface between the two vertical coordinate systems (i.e. the depth of the bottom of the k_c vertical box, §2.4.2) and the convention $0 \le \sigma_k \le 1$ has been used. The hybrid coordinates are most useful in deep ocean where maintaining near surface resolution is important. Recover (1) by $z_c = 0$ and $k_c = 0$.

(3) Double σ Coordinates: A generalization of (2). The constant coordinate interface, z_c, is replaced by one which varies in the horizontal to ride above high topography. The variable coordinate interface, $f_{i,j}$, is given by

$$f_{i,j} = \frac{z_{c_1} + z_{c_2}}{2} + \frac{z_{c_1} - z_{c_2}}{2} \tanh\left\{ \frac{2\alpha}{z_{c_1} - z_{c_2}} \left[h_{i,j} - h_{ref}\right] \right\}$$

where

z_{c_1}: The shallow limit for the coordinate interface.

z_{c_2}: The deep limit for the coordinate interface.

h_{ref}: The "switching" depth. $f_{i,j}$ is halfway between z_{c_1} and z_{c_2} at $h_{i,j} = h_{ref}$.

α: The slope parameter. How fast f responds to changes in the topography, h. Also a limit on the slope of f, $\|\nabla f\| \le \alpha \|\nabla h\|$.

The vertical system is: a σ system referenced between the surface and $f_{i,j}$ above a second σ system referenced between $f_{i,j}$ and the bottom

$$z_{i,j,k} = \begin{cases} \sigma_k * f_{i,j} & k \le k_c \\ f_{i,j} + (\sigma_k - 1) * (h_{i,j} - f_{i,j}) & k > k_c \end{cases}$$

where the (arbitrary) convention $0 \le \sigma_k \le 1$ for $k \le k_c$ and $1 \le \sigma_k \le 2$ for $k > k_c$ has been used. The double σ coordinates are most useful in regions with shallow topography and where maintaining near surface resolution is important. Recover (2) by $z_{c_1} = z_{c_2}$ or $\alpha = 0$.

GRIDS supports all three of these terrain following systems as well as a vertical system with all flat levels. This flat system is of no use as the rest of HOPS does not support it.

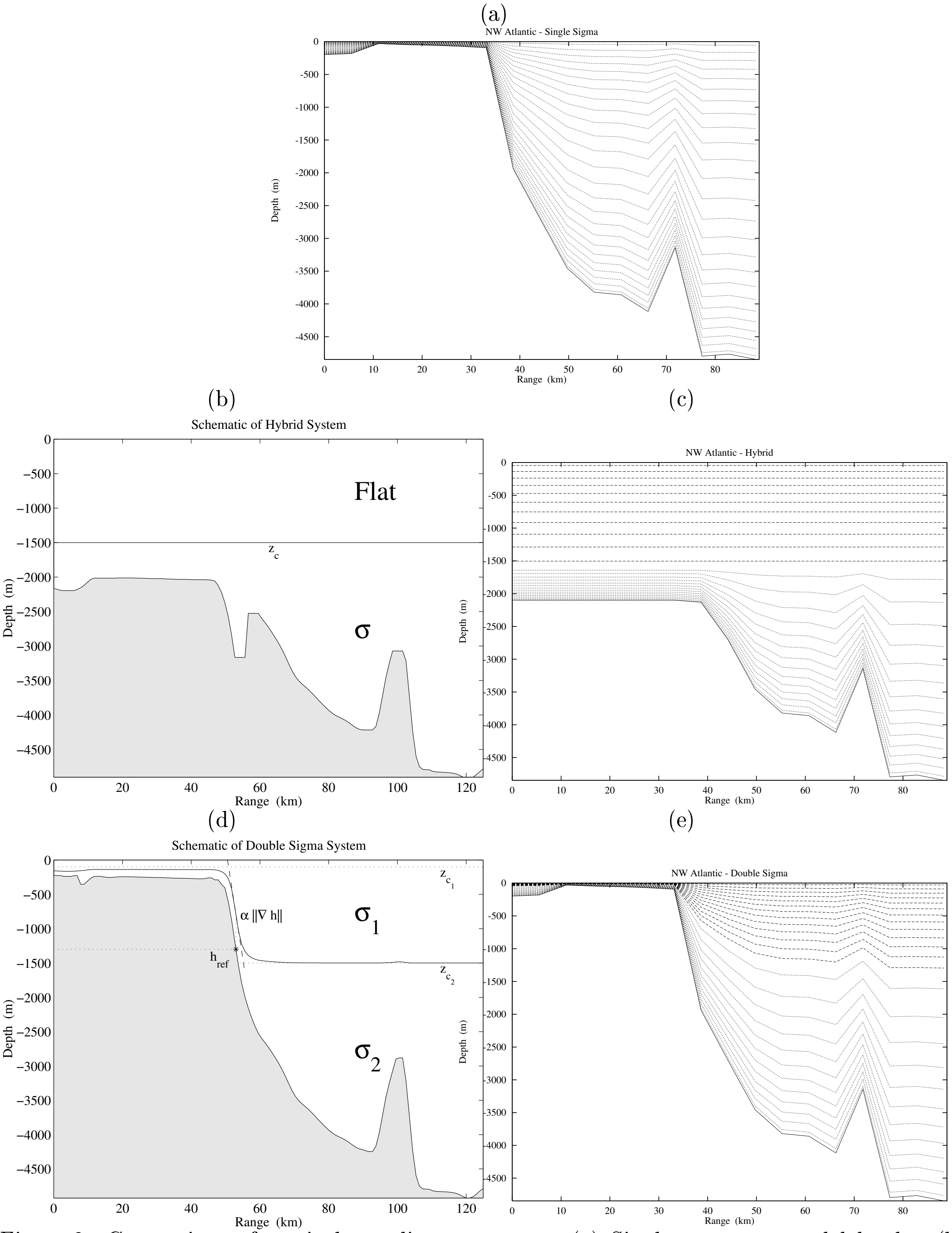

Figure 2: Comparison of vertical coordinate systems. (a) Single σ system model levels. (b–c) Hybrid system, (b) schematic (c) model levels. (d–e) Double σ system, (d) schematic (e) model levels. In both the hybrid and double σ, the upper system is shown in dashed lines and the lower in dotted. Hybrid shields shallow levels from topography, double σ does so to lesser extent but tolerates high topography.

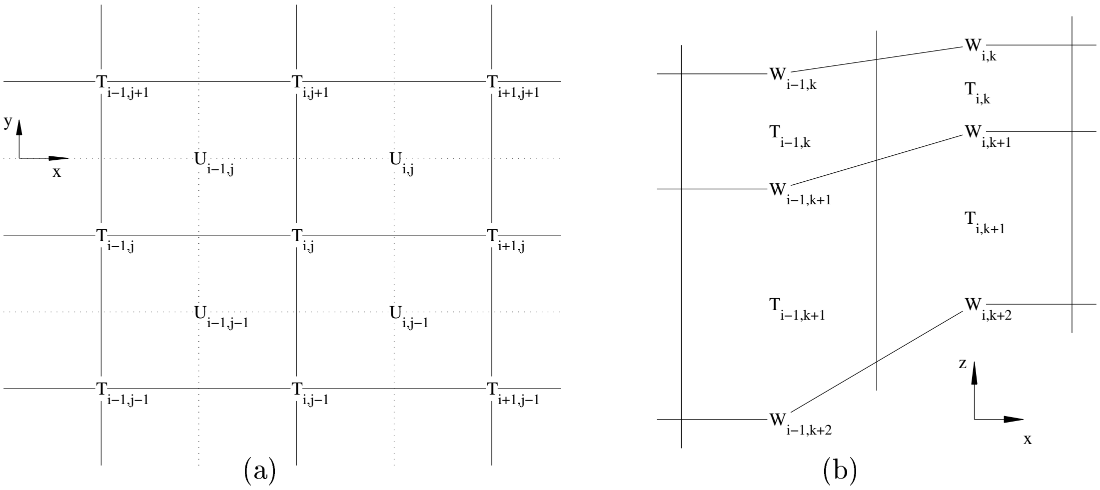

Figure 3: Grid indexing. (a) Horizontal lay–out. Here T stands for tracers (temperature, salinity, other) and streamfunction. U stands for both horizontal velocity components. (b) Vertical lay–out. Tracer vertical boxes are shown, velocity vertical boxes have the same lay–out, merely shifted $1/over2$ grid space horizontally. W stands for vertical velocities.

2.4: Discretization

This section describes how the HOPS–PE model locates data on the model grids.

2.4.1: Horizontal

The HOPS–PE assumes that data is arranged on an Arakawa B–grid. That is, temperature, salinity, streamfunction and other tracers are located on the integer nodes (tracer grid); velocities are located on the $\frac{1}{2}$ integer nodes (velocity grid) (figure 3a). GRIDS defines the model grid and extracts the topography on the tracer grid.

2.4.2: Vertical

Here, the assumption is that all data (except the vertical flux velocities) are located at the center of vertical boxes. The vertical flux velocities are located at the tops of the vertical boxes. (figure 3b) GRIDS defines the vertical grid by specifying the nominal thicknesses of the boxes (§4.III, §4.IV and §4.IX).

2.5: Computational Issues

2.5.1: 2–way nesting

The HOPS–PE can be run in 2–way nested mode. That is, 2 separate PE runs pass information on the state variables during execution. The configuration is:

- one domain (coarse) properly contains the second (fine)
- the fine domain has a grid spacing $\frac{1}{3}$ that of the coarse.
- every third grid point of the fine domain is collocated with the coarse.
- vertical grids are identical.

To properly set–up a collocated nested configuration, the user should:

- Define the coarse domain
- Use `crs2fne` to derive the fine grid parameter values.
- Plug the derived fine grid parameter values into GRIDS to view the results.
- The fine grid can be shifted by integer multiples of the coarse grid spacing through the parameters Δx and Δy. If a smaller shift is required, it can be accomplished by applying that shift to the Δx and Δy of both domains.
- Use `extract_grids` fairly frequently to keep the land masks and topographies aligned. See §5.2.3 & §5.3.4 for more detailed instructions.

2.5.2: The hydrostatic consistency factor

For the terrain following vertical coordinate systems, an important quantity is the hydrostatic consistency factor, hcf, which provides a measure of the vertical grid distortion over topography. The name comes from Haney (1991) who was examining the conditions under which discrete pressure gradient calculations would produce large errors (especially for "static" profiles). The hydrostatic consistency factor is a function of the horizontal grid spacing, δx & δy, the vertical coordinate system, σ, the vertical grid spacing, $\delta\sigma$, the topography, h, the corresponding slope, $\frac{\partial h}{\partial x}$ & $\frac{\partial h}{\partial y}$ and, in the case of a double sigma vertical coordinate system, the slopes of the coordinate interface, $\frac{\partial f}{\partial x}$ & $\frac{\partial f}{\partial y}$. The exact form of the hydrostatic consistency factor is:

<u>hybrid system</u> $\quad (0 \leq \sigma \leq 1)$

$$hcf = \max\left(\frac{\sigma}{\delta\sigma}\frac{\delta x}{|h(x,y)-h_c|}\left|\frac{\partial h}{\partial x}\right| \ , \ \frac{\sigma}{\delta\sigma}\frac{\delta y}{|h(x,y)-h_c|}\left|\frac{\partial h}{\partial y}\right|\right)$$

<u>double sigma system</u> $\quad (1 \leq \sigma \leq 2)$

$$hcf = \max\left(\left|\frac{(2-\sigma)f_x + (\sigma-1)h_x}{h(x,y)-f(x,y)}\right|\frac{\delta x}{\delta\sigma} \ , \ \left|\frac{(2-\sigma)f_y + (\sigma-1)h_y}{h(x,y)-f(x,y)}\right|\frac{\delta y}{\delta\sigma}\right) \quad .$$

The importance of this quantity lies in the observation that for $hcf \leq 2$, the HOPS–PE pressure gradient calculation only extrapolates on the bottom level. The limitation is that for some regions and vertical resolutions, trying to achieve $hcf \leq 2$ alters the topographic too much.

GRIDS reports minimum, maximum and average statistics of the hydrostatic consistency factor and creates contour plots of it. GRIDS also gives the user two easy methods to control the hydrostatic consistency factor. The first is in the selection of the vertical discretization (§4.III, §4.IV.b and §4.IX). The second is in conditioning the topography (§3.4.1 and §4.VIII). Further conditioning of the topography can be done with the aid of the HOPS utility **cond_topo**. This utility is described in its own user's guide, available by anonymous ftp (see §1.2) in the directory `pub/HOPS/Cond_Topo`.

CHAPTER 3: THE GNUMAKEFILE

This chapter describes the GNUmakefile for GRIDS and the options the user has in creating a specific version of GRIDS. The user need only make changes to a small section of the GNUmakefile. An example of the relevant section of the GNUmakefile is included in table 2 (following page).

3.1: Directory Paths

There are four pathways which the user needs to specify:

`SRCDIR` The directory in which the user keeps the source files. These include the `*.F`, `*.FF` and `*.h` files. I tend to use this to point to the directory in which I keep the unmodified source codes. I then put a copy of the GNUmakefile any routines I change in another directory and do my compiling there. The two advantages to this are (1) I always maintain a working copy of the original source code and (2) I've isolated the routines that I've changed, making it easier to pass the changes to the next version.

`PARAMDIR` An alternate directory to search for include files. This is primarily used for the parameter file `gridsparm.h`. Continuing the story begun in the explanation of `SRCDIR`, my usual directory tree has the untouched, original source code in `$HOPSHOME/Grids/Src` and the GNUmakefile & modified files in `$HOPSHOME/Grids/Work`. If I want to change the sizes of items in `gridsparm.h` but not the structure of `gridsparm.h`, then I (1) put a copy of `gridsparm.h` in the directory `$HOPSHOME/Grids`, (2) change the sizes in that copy of `gridsparm.h` and (3) set `PARAMDIR` equal to `$HOPSHOME/Grids`. This separates changes I'll need to port to future versions of GRIDS from a simple modification of maximum array bounds.

`NCDIR` The directory path for NetCDF include files. This enables the user to link to the appropriate include files for the local netCDF library. In the supplied GNUmakefiles, this path is defined with the macro `$(NETCDF_ROOT)`. As written, this is assumed to be defined in the user's `.cshrc` file.

`BINDIR` The directory in which the user would like the executable file.

3.2: Executable File Names

There are three names the user has to set for the GRIDS executables;

`BIN` The name for the GRIDS executable.

`C2FBIN` The name for the nesting utility executable.

`XTRBIN` The name for extraction utility executable.

Table 2: Part of the header of the makefile for compiling GRIDS.

```
#*************************************************************************
#************************ USER's tunable macros **************************
#*************************************************************************
#
# ARFLAGS       flags for the archive utility
# BIN           name of the executable code
# BINDIR        directory path for executable code
# C2FBIN        name for nesting utility executable
# CPPFLAGS      C-preprocessing flags and options
# FFLAGS        flags for the FORTRAN compiler
# NCDIR         directory path for NetCDF include files
# NGLIB         name for an object library containing non-plotting routines
# PARAMDIR      alternate directory path for include files
# SRCDIR        directory path for source codes
# XTRBIN        name for extraction utility executable
#
#*************************************************************************
#
# C-preprocessing options:
#
#  -Daixdate    AIX intrinsic date routine (IBM RS6000).
#  -Dcraydate   Using CRAY time/date intrinsic routines
#  -Ddecdate    Using DEC time/date intrinsic routines
#  -Dfsmed      Filter step topography with integral-weighted median filter
#  -Dfsmen      Filter step topography with integral-weighted mean filter
#  -Dftclip     Filter topography by clipping
#  -Dftmed      Filter topography with integral-weighted median filter
#  -Dftshap     Filter topography with Shapiro filter
#  -Dlnext      Use linear extrapolation of topography to edges.
#  -Dgendbg     Generic debugging.   Prevents deletion of intermediate files.
#  -Dnoplot     Disables NCAR graphics plotting.
#  -Dsundate    Using SUN time/date intrinsic routines
#  -Dsunflush   Flush output buffers on SUN systems.
#  -Dsunfpe     SUN's floating point exception trap
#
#************************************************************************
     SRCDIR   =   .
   PARAMDIR   =   .
      NCDIR   =   $(NETCDF_ROOT)/include
     BINDIR   =   .

        BIN   =   grids
     C2FBIN   =   crs2fne
     XTRBIN   =   extract_grids

      NGLIB   =   nograph_olb.a
   CPPFLAGS   =   -Dftshap -Dftclip -Dftmed -Dsundate -Dsunflush
     FFLAGS   =   -u -O
    ARFLAGS   =   rlv
#************************************************************************
#******************** End of USER's tunable macros **********************
#************************************************************************

#------------------------------------------------------------------------
#-------------------- Internal macro definitions. ----------------------
#------------------------------------------------------------------------
   RMBLKLINES   =   rmblklines
        SHELL   =   /bin/sh
           RM   =   rm -f
         ECHO   =   echo
          LIB   =   -lnetcdf
          CPP   =   cpp -P -I. -I$(PARAMDIR) -I$(SRCDIR) -I$(NCDIR)
           AR   =   ar
       RANLIB   =   ranlib
           FC   =   f77
      NCARGFC   =   ncargf77
```

3.3: FORTRAN Compilation Flags (`FFLAGS`)

These are the installation dependent flags which control the user's FORTRAN compiler. It is *highly* recommended that the user become familiar with the options for his compiler.

One editorial note: many compilers have an option for *optimized* code. It is generally a **bad** idea to mix this option with other compiler options that affect the run-time characteristics of the code. (e.g. Flags for debugging generally don't mix with optimization) Also, it is probably not the best idea to just baldly assume that the optimizer is problem free. A useful check is to run the code both unoptimized and optimized and make sure the results are identical.

One IRIS note: In the supplied `GNUmakefile.iris`, the FORTRAN flags include the macro `$(OSFLAG)`. This is something I define when I'm testing on the IRIS to select the machine precision. The user should feel free to eliminate it.

3.4: C Pre–Processing Flags (`CPPFLAGS`)

These flags determine which optional routines are included in the executable code. The flags are grouped below according to function.

3.4.1: Topography manipulation

These options enable the user to manipulate the topography which has been created from a previous GRIDS run. Any or all of these options may be combined. There are three filter options currently available, listed here in the order in which they request data from section VIII of the input file (§4.VIII). Also listed is a modifier for one of the filters.

- **`-Dftshap`** This version filters the topography with an alternating direction Shapiro filter.
- **`-Dftclip`** This version clips topography both shallower and deeper than the supplied limits.
- **`-Dftmed`** This version filters the topography with repeated applications of a local median filter.
- **`-Dlnext`** This is a sub–option for the median filter (`-Dftmed`). When active, the median filter uses linear extrapolation boundary conditions.

3.4.2: Step topography manipulation

These options enable the user to manipulate the step topography. These options are only active if the run–time variable `hybrid` is *false* (§4.IV). Any or all of these options may be combined. There are two options currently available, listed here in the order in which they request data from section IX of the input file (§4.IX).

- **`-Dfsmed`** This version filters the step topography with a local *median* filter.
- **`-Dfsmen`** This version filters the step topography with a local *mean* filter.

3.4.3: Debugging

These options aid the user in debugging GRIDS. The two options currently available are:

−Dgendbg This option is for "generic debugging". The **only** effect of this option is to prevent GNUmake from deleting intermediate files (*.f, *.o). This, however, can be handy when trying to debug compiler errors.

−Dsunfpe This option enables the SUN's floating point exception flags. This, in turn, enables the SPARCWorks Debugger tool to recognize errors and the lines at which they occur. For the debugger to work the code must be compiled with the `-g` (debugging) option and should be compiled with the `-C` (array bound checking) option.

3.4.4: Time/Date

By default, GRIDS does not produce time and date information. For certain machines GRIDS knows how to produce a time stamp. Which routines it attempts to use depends on which flag is given.

−Daixdate This option uses the AIX (IBM RS6000) routines for time and date.

−Dcraydate This option uses the CRAY routines for time and date.

−Ddecdate This option uses the DEC routines for time and date.

−Dsundate This option uses the SUN routines for time and date.

3.4.5: Miscellaneous

The remaining options are:

−Dnoplot This option inhibits NCAR graphics plotting. This is useful if the user does NOT have access to the NCAR graphics library.

−Dsunflush The default action on the SUN is to have the written update to the log file significantly lag the current code status. This option forces the update to keep pace with the current status.

3.5: Archive Utility Flags (`ARFLAGS`)

These are the installation dependent flags which control the user's archive utility. This utility is used to create an object library of the routines which do not call the NCAR subroutines. This is done to reduce the length of the argument given to the NCAR compilation script. (Only subroutines which do not call NCAR are in the library because the SUN4 compiler gets confused otherwise.) It is *highly* recommended that the user become familiar with the options for his archive utility.

In the example in table 2, *a SUN4 example*, the selected flags are:

r Replace the named files in the archive.

l Local. Place temporary files in the current working directory rather than in the default temporary directory, `/tmp`.

v Verbose. Give a file–by–file description of the creation of a new archive file.

3.6: GNUmakefile Installation Issues

A number of internal macros are defined for the system commands used by the GNUmakefiles. These will generally only have to be defined once, the first time the user installs GRIDS on a new system.

`RMBLKLINES` The name given to the executable code (provided with this package) to remove blank lines from the pre-processed code. This macro is provided to avoid possible naming conflicts.

`SHELL` The shell to be used by the GNUmakefile.

`RM` The remove command.

`ECHO` The echo command.

`LIB` Needed libraries. Usually just the netCDF library.

`CPP` The C Pre-Processor and needed options. The basic options (for the example given in table 2) are:

- `-P` This option inhibits the line control information expected by the C compiler. Very useful when using the FORTRAN compiler instead.
- `-I` Controls the directory search of the C Pre-Processor for the include files. The order used is: first the directory in which the GNUmakefile is located, second the directory specified by the macro `PARAMDIR`, third the source code directory specified by `SRCDIR` and finally the NetCDF include directory specified by `NCDIR`

For some systems (e.g. the DEC Alpha) it is necessary to add an option (`-C`) to *prevent* the removal of C++ style comments. This is necessary since the C++ comment delimiter, `//`, is the FORTRAN string concatenation operator.

`AR` The archive utility.

`RANLIB` Utility to convert archives to random libraries. If not needed by the user's system leave blank (see `GNUmakefile.iris`).

`FC` The FORTRAN compiler.

`NCARGFC` NCAR's script for compiling FORTRAN code with NCAR graphics.

3.7: Compiling with the GNUmakefile

The default action of the GNUmakefile is to compile all three programs in the GRIDS package. For example, assuming that (i) the GNUmake command has been renamed `gmake` to distinguish it from the default `make` command and (ii) the appropriate GNUmakefile.*xxx* has been renamed GNUmakefile, then compiling is simply done by issuing the following command:

```
gmake >& mke.log &
```

Notice that in the above example the screen output of the GNU make command has been redirected to a log file (`mke.log`). This has been done to preserve a record of any problems encountered by GNUmake. It is **highly** recommended that the user examine this log file for error & warning messages. These messages will be dependent on the C Pre-Processing programs & FORTRAN compilers used. Some examples on finding these messages are given for selected systems:

<u>SUN :</u> Use the following 2 commands to check for error messages and warning messages, respectively.

```
grep -i -n error mke.log
grep -i -n warning mke.log
```

Silicon Graphics : Use the following 3 commands to check for (1) certain categories of CPP errors, (2) general errors and (3) general warnings.

```
grep -i -n "unterminated string or character" mke.log
grep -i -n error mke.log
grep -i -n warning mke.log
```

RS6000 : Use the following 2 commands to check for error messages and warning messages, respectively.

```
grep -i -n error mke.log
grep -i -n "(W)" mke.log
```

CHAPTER 4: THE INPUT FILE

An example input file is dissected in table 4. Some of the sections always appear the same. The contents and/or existence of other sections depends upon previous input. The input conventions are that all non–character input is free formatted and hence is column independent. Character data is read with the `(a)` format. Consequently quotation marks are unnecessary. The rest of this chapter is divided into sections which describe the corresponding sections of the input file in detail.

4.I: Grid Output File

This section provides the name of the file to which GRIDS writes its results. The file name is any character string of 80 characters or less. If the variable `new_topo` is *false* (§4.V), this file must already exist as the result of a previous GRIDS run. In this case, it will be the source for the topography (§4.V). This file name must **never** be the same as `topo_infile` (§4.V)

4.II: Horizontal Grid Definition

This section defines the horizontal grid. The necessary information is

Table 3: **Horizontal Grid Parameters**

Variable	Units	Description	Type
`coord`		Type of horizontal grid.	Integer
`dx`	meters or degrees	Mean grid spacing in x-direction.	Real
`dy`	meters or degrees	Mean grid spacing in y-direction.	Real
`lonc`	degrees	The longitude of the center of the grid.	Real
`latc`	degrees	The latitude of the center of the grid.	Real
`delx`	meters or degrees	X-offset between domain & transformation centers.	Real
`dely`	meters or degrees	Y-offset between domain & transformation centers.	Real
`rotang`	degrees	The angle through which the grid is rotated.	Real
`no_x`		The number of grid points in the x direction.	Integer
`no_y`		The number of grid points in the y direction.	Integer

If (`coord` = 0):
- Generate a Cartesian (tangent plane) grid.
- `dx`, `dy`, `delx` and `dely` are specified as *meters*.

Else if (`coord` = 1):
- Generate an unrotated spherical grid.
- `dx`, `dy`, `delx` and `dely` are specified as *degrees*.
- `lonc`, `latc` and `rotang` are zero.

Else if (`coord` = 2):
- Generate a rotated spherical grid.
- `dx`, `dy`, `delx` and `dely` are specified as *degrees*.

Otherwise:
- Invalid value for `coord`.

Table 4: Schematic of the input file for the GRIDS program. The section and sub-section indicators in the table refer to sections and sub-sections of this chapter.

Input file text		sub-section	Section
grids_iff92.nc			I
0	coord		
5000.0	dx (m)		
5000.0	dy (m)		
-10.5	lonc (deg)		
64.25	latc (deg)		II
0.0	delx (m)		
0.0	dely (m)		
0.0	rotang (deg)		
45	no_x		
39	no_y		
5	no_lvl		
50.0	dzt (m)		
100.0			III
100.0			
150.0			
200.0			
t	hybrid	a	
			IV
2	kc		
t	optimum	b	
0.0	tlon (deg)		
0.0	tlat (deg)		
0.0	blon (deg)		
0.0	blat (deg)		V
gap_cst.dat			
t	new_topo		
t	target		
etopo5.nc			
0	no_cl		
t	block_cl	a1	
2	radx		
2	rady		
		a2	VI
0	hno_cl		
t	hblock_cl		
2	hradx	b1	
2	hrady		
		b2	
t	plot_grid		
T-grid, Iceland-Faroe Frontal Region, 5km			
t	cont_top:		VII
Topography, Iceland-Faroe Frontal Region, (raw)			
f	plot_step		
Hydrostatic consistency factor, IFF (raw)			
			VIII
0.0	zc1 (m)		
0.0	zc2 (m)		
0.0	alpha		
-800.0	href (m)		IX
2.0	c		
1.0e-12	zc1_tol		
200	zc1_itmax		
0	opt_iface		

4.III: Vertical Levels

This section defines the vertical levels. The needed information is:

Table 5: **Vertical Levels Parameters**

Variable	Units	Description	Type
`no_lvl`		The number of vertical levels.	Integer
`dzt`	meters	The thicknesses of the T-boxes.	Real

Each thickness (`dzt`) is entered on a separate line. There must be **exactly** `no_lvl` values for `dzt`. If the user specifies more levels than storage allows (i.e. `no_lvl` > `mx_lvl`, see §1.5) GRIDS only uses the first `mx_lvl` values and skips over the rest.

4.IV: Vertical Coordinates

4.IV.a: Type

This section defines the type of vertical coordinate system.

Table 6: **Vertical System Parameters**

Variable	Units	Description	Type
`hybrid`		Flag to indicate the type of vertical coordinate system.	Logical

`hybrid` = true The vertical system is a terrain following system. (See §4.IV.b & §4.IX for selecting between σ, hybrid and double σ systems.)

`hybrid` = false The vertical system is flat model levels.

4.IV.b: Additional Discretization Data

The contents of this subsection depends upon the value of `hybrid`.

`hybrid` = true

This section contains additional information needed for the terrain following systems.

Table 7: **Vertical Discretization Parameters**

Variable	Units	Description	Type
`kc`		The number of levels in the upper system.	Integer
`optimum`		Flag to indicate whether to optimize levels.	Logical

`optimum` = true Change the vertical levels (§4.III) to improve the hydrostatic consistency condition. (See §4.IX for parameters governing the optimization.)

`optimum` = false Leave the model levels (§4.III) alone.

For a single σ–system set `kc` to zero and the interface parameters (§4.IX) for a hybrid (as opposed to double σ) system.

`hybrid` = false

This subsection does not exist, not even a blank line to mark its passing.

4.V: Plot window & Topography manipulation

This section defines the plotting information and the topography input. (Refer to §1.4.1 for lat/lon conventions.) The necessary information is

Table 8: **Plot & Topography Parameters**

Variable	Units	Description	Type
`tlon`	degrees	The longitude of the upper left corner of the plot window.	Real
`tlat`	degrees	The latitude of the upper left corner of the plot window.	Real
`blon`	degrees	The longitude of the lower right corner of the plot window.	Real
`blat`	degrees	The latitude of the lower right corner of the plot window.	Real
`cst_file`		The name of a file containing coastline data.	Character*80
`new_topo`		Flag to indicate whether reading "new" topography.	Logical
`target`		Flag to indicate whether modifying "target" topography.	Logical
`topo_infile`		File from which topography is read.	Character*80

If `blon`=`tlon`, GRIDS automatically adjusts them to just contain both the tracer and velocity grids. A similar readjustment takes place when `blat`=`tlat`.

Coastline data file.

GRIDS accepts 2 formats for coastline data (`cst_file`): ASCII and netCDF. The interpretation of the data is different for each format.

ASCII formatted

The coastline data in an ASCII file are assumed to be simple coastal segments. No segment is assumed to be the full coastline nor is any differentiation made between coastlines separating land/ocean versus coastlines separating lakes/land.

The ASCII formatted coastline data files contain two columns of coastline data points (latitude, longitude). The individual coast lines are delimited with values of 999, as follows

Table 9: **Sample Coast File**

```
999.00000     999.00000
 49.79449    -125.00000
 49.79280    -124.99831
    ⋮             ⋮
 48.70533    -125.00000
999.00000     999.00000
 48.99752    -122.75972
    ⋮             ⋮
 50.00000    -123.95071
999.00000     999.00000
```

This format is provided for both backwards compatibility and to provide an easy format for the user to import his own coastal data.

netCDF formatted

The coastline data in a netCDF file are assumed to be a subset of the Global Self–consistent Hierarchical High–resolution Shorelines (GSHHS) data set (Wessel & Smith, 1996). The coastlines are polygons bounding:

- Land
- Lakes
- Islands in lakes
- Ponds within islands in lakes

Given these hierarchical polygons, GRIDS is able to add shaded regions to its plots.

The netCDF versions of the GSHHS data that GRIDS reads are available via anonymous FTP from `maelstrom.harvard.edu (128.103.2.50)`. Go to the directory `pub/HOPS/Plot` and down–load the file `pltdat.tar.Z`. Included in this file are netCDF versions of the five resolutions of the GSHHS available from NOAA:

`gshhs_f.nc` – Full resolution. Original Data
`gshhs_h.nc` – High resolution. Approximately 0.2 km
`gshhs_i.nc` – Intermediate resolution. Approximately 1 km
`gshhs_l.nc` – Low resolution. Approximately 5 km
`gshhs_c.nc` – Crude resolution. Approximately 25 km

The NCAR polygon filling algorithm seems to get confused with fine resolution coastlines used with large plot windows. I've successfully used the full resolution coastline with a small domain ($< 1°$square). The NWA examples in this document were made with the intermediate resolution coastlines.

If no file exists which contains the relevant coastlines, enter a NONEXISTENT file for `cst_file`.

Topography data.

There are two possibilities for topography input. The one in use is governed by `new_topo`.

<u>`new_topo` = true</u>

The relevant topography is read from a netCDF file, `topo_infile`, containing topography and enough position data to determine the spacing and extent of the topography. The topography is assumed to be on a uniform grid.

<u>`new_topo` = false</u>

The topography is read from a file created by GRIDS. Not just any file, but the output file specified in §4.I. The variable `topo_infile` is read but ignored.

The use of the variable `target` depends on the value of the variable `hybrid` (§4.IV.a).

<u>`hybrid` = true</u>

The variable `target` is used to determine if a second copy of the topography should be written into the netCDF file (§6.4.5). This second topography is the "target" for slope conditioning (§5.3.2). This should only be written for the raw extraction and the first level conditioning (§5.3.1).

<u>`hybrid` = false</u>

The variable `target` is ignored.

4.VI: Contouring control

4.VI.a: Topography

This subsection contains the controls for contouring the topography.

a1) The constant information is

Table 10: **Topography Contouring Parameters**

Variable	Units	Description	Type
`no_cl`		The number of contour levels.	Integer
`block_cl`		Whether to block contour lines through high/low markers.	Logical
`radx`		The high/low search radius in the x-direction.	Integer
`rady`		The high/low search radius in the y-direction.	Integer

The variables `radx` and `rady` are used to place the high/low markers on a contour plot. When the contour lines are blocked from being drawn through these markers (`block_cl`=true), an area map work array is created. If too many high/low markers are drawn, this array will overflow. The symptom of this is a contour plot with many markers and no contour lines. The preferred (quicker) methods of correcting this are either to increase the high/low search radii or to set `block_cl` to false. An alternative method is to increase the workspace size through the parameter `dimm` (see §1.5) and then remake GRIDS.

The method of level selection depends upon the value of `no_cl`.

<u>`no_cl` > 0</u>

`no_cl` contour levels are individually selected by the user.

<u>`no_cl` = 0</u>

GRIDS selects the contour levels automatically.

<u>`no_cl` < 0</u>

|`no_cl`| evenly spaced contour levels are selected by the user.

a2) The information in this section depends upon the value of `no_cl`, always keeping in mind the convention that topographic depths below sea-level are *negative* (§1.4.2).

<u>`no_cl` > 0</u>

This subsection contains **exactly** `no_cl` lines of data:

Table 11: **User Contours (any interval)**

Variable	Units	Description	Type
$\mathtt{clvl}_i$	meters	The i^{th} contour level.	Real

If the user specifies more contour levels than storage allows (`no_cl` > `max_cl`, see §1.5), GRIDS only uses the first `max_cl` values.

<u>`no_cl` = 0</u>

This subsection does not exist, not even a blank line to mark its passing.

<u>`no_cl` < 0</u>

This subsection contains two lines of data:

Table 12: **User Contours (constant interval)**

Variable	Units	Description	Type
`clmin`	meters	The minimum contour level.	Real
`clint`	meters	The contour interval.	Real

4.VI.b: Hydrostatic Consistency

The contents of this section depends upon the value of `hybrid` (§4.IV.a).

<u>`hybrid` = true</u>

This subsection contains contouring information for plots of the hydrostatic consistency factor. The layout is exactly the same as VIa, except that the data are:

1) non–dimensional
2) non–negative.

hybrid = false

This subsection does not exist, not even a blank line to mark its passing.

4.VII: Plot flags & titles

This section determines whether a particular plot is make and what its title will be. The necessary data are:

Table 13: **Plot Titles**

Variable	Units	Description	Type
plot_grid		Whether to plot the grid.	Logical
grd_ttl		Title of the grid plot.	Character*80
cont_top		Whether to contour the topography.	Logical
cont_ttl		Title of the topography contour plot.	Character*80
plot_step		Whether to plot the vertical system information.	Logical
step_ttl		Title of the vertical system plot.	Character*80

The vertical system plot depends upon the value of hybrid (§4.IV.a).

hybrid = true

Contour hydrostatic consistency factor in the upper and lower systems.

hybrid = false

Plot the step topography by drawing lines between adjacent grid points of equal depth.

4.VIII: Topography manipulation

The contents of this section depend in part upon the value of new_topo (§4.V).

new_topo = true

This section does not exist, not even a blank line to mark its passing.

new_topo = false

This section is allotted to the routines to condition the topography. The activation of said routines is governed by the C-Preprocessing flags -Dftshap, -Dftclip and -Dftmed (§3.4.1). If none of these flags is selected, this section does not exist, not even a blank line to mark its passing. The input for each option, and the order in which they appear, are as follows:

–Dftshap This version filters the topography with an alternating direction Shapiro filter. Section VIII of the input file supplies

Table 14: **Shapiro Filter Parameters (topography)**

Variable	Units	Description	Type
filt_ord		Order of the Shapiro filter.	Integer
max_filt		Number of filter applications.	Integer

The Shapiro filter order, `filt_ord`, must be a power of 2 (*i.e.* 2, 4, 8, ...). If this option is selected, the Shapiro filtering can be shut off by setting `max_filt` to zero.

`-Dftclip` This version clips topography both shallower and deeper than the supplied limits. Section VIII of the input file supplies

Table 15: **Clipping Parameters (topography)**

Variable	Units	Description	Type
`hshllw`	meters	Shallowest allowed topography.	Real
`hdeep`	meters	Deepest allowed topography.	Real

If this option is selected, the clipping can be shut off by setting `hshllw` to 10000.0m and `hdeep` to -12000.0m (shallower and deeper than any topography).

`-Dftmed` This version filters the topography with repeated applications of a local median filter. Section VIII of the input file supplies

Table 16: **Median Filter Parameters (topography)**

Variable	Units	Description	Type
`max_filt`		Maximum number of iterations.	Integer

If this option is selected, the median filtering can be shut off by setting `max_filt` to zero.

4.IX: Additional vertical information

The particular use of this section depends upon the value of `hybrid` (§4.IV).

`hybrid` = true

This section contains information to define the surface between the upper and lower coordinate systems (§2.3).

Table 17: **Double σ Interface Parameters**

Variable	Units	Description	Type
`zc1`	meters	Shallowest depth of the coordinate interface.	Real
`zc2`	meters	Deepest depth of the coordinate interface.	Real
`alpha`		Slope parameter of the coordinate interface.	Real
`href`	meters	Reference depth for the coordinate interface.	Real
`C`		Upper bound for $\frac{\sigma}{\delta\sigma}$.	Real
`zc1_tol`		Relative tolerance in resetting `zc1`.	Real
`c1_itmax`		Maximum number of iterations in resetting `zc1`.	Integer
`opt_iface`		Control flag for resetting `zc1`.	Integer

If ((`zc1` = `zc2`) or (`alpha` = 0)):

- The vertical coordinate system is a σ or hybrid system.
- `C` is an upper bound for $\frac{\sigma}{\delta\sigma}$ in the lower system. $\left(C > \frac{3}{2}\right)$

- ∘ GRIDS optimizes the model levels only in the lower coordinate system. (See §4.IV.b)
- ∘ GRIDS automatically resets `zc1` and `zc2` to conform with the first `kc` user specified model levels (`dzt`). (See §4.III & §4.IV.b)
- ∘ Input parameters `zc1_tol`, `zc1_itmax` and `opt_iface` are ignored.

Otherwise

- ∘ The vertical coordinate system is a double sigma system.
- ∘ `C` is an upper bound for $\frac{\sigma}{\delta\sigma}$ in the upper system. $\left(C > \frac{3}{2}\right)$
- ∘ GRIDS optimizes the model levels in both the upper and lower coordinate systems. (See §4.IV.b)
- ∘ Input parameters `zc1_tol`, `zc1_itmax` and `opt_iface` are used to reset `zc1` to a more useful value. `zc1_tol` and `zc1_itmax` are standard parameters to control a Newton iteration. If `opt_iface`=0, `zc1` retains its input value. This is a necessary control for nesting. If `opt_iface`=1, `zc1` is chosen so that coordinate interface is equal to the input value of `zc1` over shallowest topography. If `opt_iface`=2, `zc1` is chosen to keep vertical levels as thick as possible over shallowest topography (aids vertical CFL).

`hybrid` = false

This section is allotted to the routines to condition the step topography. The activation of said routines is governed by the C-Preprocessing flags `-Dfsmed` and `-Dfsmen` (§3.4.2). If neither flag is selected, this section does not exist, not even a blank line to mark its passing. The input for each option, and the order in which they appear, are as follows:

`−Dfsmed` This version filters the step topography with a *median* filter. When used and active, section IX of the input file supplies

Table 18: **Median Filter Parameters (step topography)**

Variable	**Units**	**Description**	**Type**
`max_filt`		Maximum number of iterations.	Integer

If this option is selected, the median filtering can be shut off by setting `max_filt` to zero.

`−Dfsmen` This version filters the step topography with a *mean* filter. When used and active, section IX of the input file supplies

Table 19: **Mean Filter Parameters (step topography)**

Variable	**Units**	**Description**	**Type**
`max_filt`		Maximum number of iterations.	Integer

If this option is selected, the mean filtering can be shut off by setting `max_filt` to zero.

Note: The median filter is generally more faithful to the slope than the mean filter.

CHAPTER 5: USING GRIDS

5.1: Introduction

The GRIDS package was written to be an iterative processes. The user iterates to define the horizontal domain, then iterates to condition the topography. This chapter gives some pointers.

5.2: Horizontal Domain Definition

5.2.1: General Domain Selection

The user makes an initial guess of the horizontal domain parameters. By viewing the resulting grid against coastlines and the extracted topography, the user iterates the choices for horizontal domain parameters until the desired domain is defined.

- Keep it simple. For Cartesian & rotated spherical grids keep the shift parameters (`delx` and `dely`) equal to zero. For these two grid types, the only reason to have nonzero shift parameters is nesting.
- Reduce distortions. For rotated spherical grids, it is a good idea to keep `nx`≥`ny`. This reduces the grid distortions by keeping everything as close as possible to the rotated "equator".
- Ensure that all relevant features are included in the domain. For coastlines and topography, this can be accomplished by trying a grid, viewing the results and making the necessary modifications. For circulation features, this requires comparing plots of the grid and topography against circulation schematics from the literature.
- Make sure enough of the grid on land. Don't waste too many nodes on land, but to have a well supported coast requires two grid points on land with a third point to be the coastline. (The exception, the "northern" boundary, `y`=`ny`. There the last point is "wasted", so three land points are required.)

5.2.2: Nesting

When setting up nest–able domains, the previous general comments still apply. Only now there is the additional constraint that all the domains must be collocated.

- Initially take a top–down approach. Start with the largest domain and work down to the smallest using the utility `crs2fne`. This program reads an input file containing the number of grid points for the fine grid, the desired center for the fine grid and the name of the coarse grid GRIDS netCDF file in which the fine grid will nest.
  - The numbers of grid points in the fine grid are constrained by:

    $n_{x\text{fine}} \bmod 6 = 5$

    $n_{y\text{fine}} \bmod 6 = 0$
- Iterate, iterate, iterate. Collocated, fine grids don't always end up where the user wants. Be prepared to alter the larger domains, if necessary, to get the desired finest domain.
  - For small changes, the shift parameters of all domains can be used to get the desired effect.

5.2.3: Land Masking

For coastal domains, especially coastal domains with nesting, it's generally a good idea to wait until the domain definitions have stabilized to start creating the land masks. Once the domains have settled, use the HOPS package PE_mask to create a land mask and write it to the GRIDS file. (If MATLAB is available, use the MATLAB version, it's much more convenient.)

When nesting with land in the fine domain the basic idea is to allow coastline refinements in the interior of the small domain (naturally) **but** to keep the intersections of the coastlines with the small domain boundaries aligned with the coastlines in the large domain (to ensure correct net transport). To accomplish this use PE_mask and `extract_grids` in the following procedure:

(1) Create the desired mask in large domain with PE_mask.

(2) Create quick and simple mask in small domain with PE_mask. I usually just create a rectangle that fills all land, lakes, bays etc. This step is done to create the data structures in the small domain netCDF file.

(3) Extract the T-grid land mask from the coarse grid to the fine grid using `extract_grids`.

(4) Refine the mask in small domain with PE_mask. To keep the coastlines aligned at the boundary, I don't touch the mask within 3–6 grid points of an inflow/outflow boundary.

(5) Extract the T-grid land mask from the fine grid to the coarse grid using `extract_grids`. This ensures that regions which can be supported by 2–way nesting are active and only those points are active.

(6) Update the coastlines in the large domain with PE_mask. Don't modify the mask, let PE_mask update the auxiliary structures accordingly.

For telescoping, multi–level, 2–way nested domains I repeat steps 1–4 from the coarsest domain to the finest domain. Then I repeat steps 5–6 from the finest domain to the coarsest domain.

5.3: Topographic Conditioning

In the preceding section, as the domains changed the topography was repeatedly extracted from the supplied database. In this section, GRIDS is run again but this time the domain remains the same. The topography of a previous run is modified (`new_topo` is *false*, §4.V). This is done for a couple basic reasons:

- Ensuring valid terrain following coordinated
- Improving HOPS PE mode run characteristics

5.3.1: Protecting the Terrain Following Coordinates

Once the horizontal domains have been defined, the topography will need a bit of work. The first step is to modify the topography to protect the vertical coordinate system from: (1) singularities and (2) two grid point noise. This operation creates the desired ("target") topography, set `target` (§4.V) to *true.*

(1) Singularities. The transformation becomes singular if the terrain following model levels are squeezed into a point. This occurs if

σ: the topography crosses the surface of the ocean.

hybrid: the topography crosses the bottom of the deepest flat box.

2–σ: either (1) the topography crosses the coordinate interface or (2) the coordinate interface crosses the surface of the ocean.

To avoid this, the topography must be clipped (§3.4.1 and §4.VIII). For single or double σ, clip the topography around -10 or -20 meters. (For double σ, the coordinate interface still needs to be defined with care, §4.IX.) For hybrid vertical system, clip the topography below the deepest flat level.

(2) Two Grid Point Noise. Since the vertical system is terrain following, two grid–point noise in the topography can propagate through the model levels into the model fields. To control this a median filter is provided (§3.4.1 and §4.VIII). This filter is designed to remove small scale

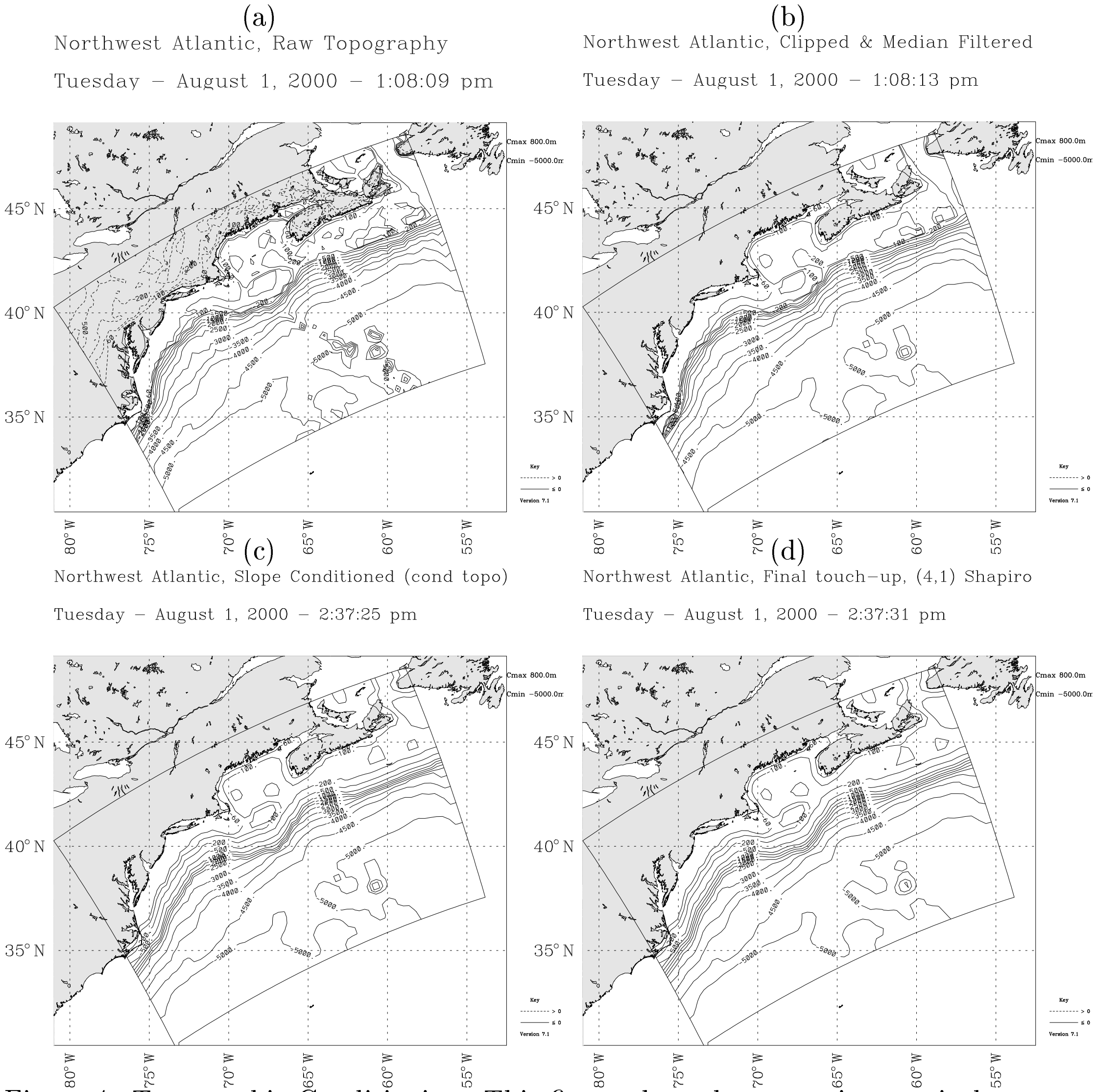

Figure 4: Topographic Conditioning. This figure show the stages in a typical topographic conditioning exercise. (a) Raw topography. The starting point. (b) Clipped and Median Filtered. Small scales removed, slope preserved. (c) Slope conditioned. Steepest slopes reduced. (d) Final touch–up. Only minor changes.

---

noise while preserving gradients. It should be run until it reaches a steady state (usually $<$ 200 iterations).

With these two objectives complete (which can be done in one GRIDS run) the "target" topography is constructed. Any further modifications should be done with `target` set to *false* (§4.V).

5.3.2: Controlling the Slope

Steep slopes can lead to PE pressure gradient errors in terrain following systems. This can be controlled, to an extent, by further conditioning. The preferred method to do this is by using the HOPS utility **cond_topo**. This utility is described in its own user's guide, available by anonymous ftp (see §1.2) in the directory `pub/HOPS/Cond_Topo`.

Currently this utility is only available in MATLAB form. If the user does not have access to MATLAB, then the Shapiro filter (§3.4.1 and §4.VIII) is the only supplied option[1]. This is a numerical filter designed to be tunable for small scale filtering. There are two tuning parameters the filter order and the number of repetitions (§4.VIII). The filter order must be a power of 2 (i.e. 2, 4, 8, 16). The higher the filter order the more the filter concentrates on the smaller scales. For slope conditioning, a lower order is generally advised.

5.3.3: Final Touch–Ups

Once the slopes are under control, there is an optional touch–up procedure. The logic is: the topography will ultimately be used in a PE model run. In that run, the Shapiro filter will be used on the model fields along model levels. In order to reduce the vertical component of this "horizontal" filter, the topography is filtered with the same Shapiro filter as will be used in the model run. There is no formal argument for doing this, but there is some evidence that it helps.

If this touch–up filtering will be done, there are some loose rules of thumb:

- The filter order is the same that will be applied to the tracer variables.
- The number of repetitions is kept small (typically 5 or less).

As always, the Shapiro filter does not respect the topography bounds. Therefore if it is used, the topography must be re–clipped to the desired values to avoid re–introducing singularities.

5.3.4: Nesting

As was mentioned previously (§2.5.1), the HOPS PE 2–way nesting codes assume that the vertical grids are the same. For terrain following coordinate systems this requires that the topographies be aligned. To accomplish this the utility `extract_grids` is supplied. This is an interactive utility which prompts the user for the name of the coarse (large) GRIDS netCDF file, the name of the fine (small) GRIDS netCDF file, the desired direction of extraction (coarse → fine or fine → coarse) and the desired field to extract (topography, land mask or both). Focusing on the topographic extraction, the effect of `extract_grids` is:

<u>coarse → fine</u> The coarse grid topography is bilinearly interpolated onto the fine grid domain. This produces smoother topographies for the run but usually eliminates finer resolutions features which could be important for the fine scale physics. This problem compounds for telescoped, multi–level nests.

<u>fine → coarse</u> The fine grid topography is averaged into the coarse grid topography. This preserves desired fine scales but may be a little more computationally delicate in the Primitive Equation model.

My usual approach is to use the fine → coarse extraction after every set of runs of `cond_topo`. That is, I run `cond_topo` separately in each domain with the same control parameters and then run `extract_grids`.

5.4: Vertical Coordinate System

The definition and refinement of the vertical coordinate system usually proceeds in parallel with the horizontal domain definition and with the topographic conditioning. There are three basic components to defining the vertical coordinate system: (1) selecting the basic type of vertical system (§2.3), (2) selecting the vertical resolution and (3) defining the coordinate interface. Although separated for discussion, these three components are intimately entwined.

[1] Not strictly true. Occasionally a little judicious clipping can help (§3.4.1 & §4.VIII).

5.4.1: Selecting the Vertical System

The final selection of the type of vertical coordinate system should be made by the time of the final determination of the horizontal domain (§5.2.1). The choice of the basic type of vertical system is very much problem dependent, but there are some guiding considerations

high topography "high topography" is topography that varies over a significant portion of the vertical column. In this case, the hybrid coordinate system will probably not be useful.

surface resolution For situations in which maintaining resolution in the upper portion of the water column is important, either the hybrid or double σ is a good choice.

5.4.2: Vertical Resolution

There two parts to the vertical resolution: (1) the total number of model levels and (2) the relative thicknesses of model levels. The total number of model levels must be decided by the time the horizontal domain is settled (§5.2.1). There should be enough levels to resolve the water column but not so many levels that PE model runs are bogged down.

The choice of relative model level thicknesses will be problem driven. Evenly spaced model levels are easy to implement (simply set all the `dzt`'s to the same value, §4.III) and are often a good starting point. For schemes that concentrate model levels where they may most be needed (usually near the surface), there are a couple of different approaches we've used.

controlling the hydrostatic consistency factor For the σ–type levels, the thicknesses may be chosed to reduce the hydrostatic consistency factor. GRIDS will do this automatically (§4.IV & §4.IX). A couple words of caution: (1) This only applies to the σ–type levels, the flat levels in a hybrid system are unaffected. (2) This method tends to repel model levels from the lower surface. (i.e. repelled from the topography for σ, hybrid and the lower system in a double *sigma*; repelled from the coordinate interface for the upper system in a double *sigma*.) This "optimization" is tuned using the upper bound `C` (§4.IX). The bound should always be greater than $\frac{3}{2}$. Larger values of `C` squeeze levels less but results in larger hydrostatic consistency factors. Let the games begin.

constant ratios A completely different school of thought suggests holding the ratio between adjacent model boxes to be a fixed value, close to 1. The basic idea is to keep as close as possible to even spacing to improve the truncation error. This applies to all systems, including the upper levels in a hybrid system or even purely flat (pressure) levels. There is no utility in GRIDS for picking levels this way, so they must be input directly (§4.III). When I try to set–up a discretization this way, I solve the equation

$$\sum_{n=1}^{N} dh_1 r^{n-1} - \Delta H = 0$$

for r, where

r = the constant ratio between model thicknesses.
N = the total number of model levels.
ΔH = the depth of a representative water column to be spanned by these levels.
dh_1 = the desired first thickness for the first (shallowest) model level.

The goal is to find r greater than but close to 1. The game is usually played by trying various combinations of N and dh_1. Bear in mind that for the hybrid and double

σ systems, this ratio won't hold across the coordinate interface for all horizontal locations.

One variation: for an exercise in the Gulf of Cadiz (near the Strait of Gibraltar) we had to resolve both the Atlantic water flowing into the Mediterranean (surface current) and the Mediterranean outflow (bottom current). To set up the vertical discretization I used ΔH and N for the top half of the water column in the above equation. Then I repeated the resulting levels, in reverse order, for the bottom half of the water column.

5.4.3: Definition of Coordinate Interface

This is best discussed individually for the three types of terrain following systems.

σ: There is no coordinate interface. End of story.

Hybrid: The coordinate interface, z_{c_1}, is simply the sum of the first k_c model thicknesses. Just make sure they add up to the desired depth (usually deep enough to contain the main thermocline in the flat levels).

2–σ: This is the hairy case. There are four tunable parameters defining the double σ coordinate interface (§2.3 & figure 2d). In practice, the guidelines for choosing these parameters are:

- **$\mathbf{z_{c_2}}$:** The deepest depth of the interface. This one is pretty free. The coordinate interface will usually get to this depth. Again the usual choice is below the main thermocline to maintain resolution.
- **α:** The slope parameter. Generally want to keep $0 < \alpha \leq \frac{1}{2}$. This shields the upper levels from the steepest topography and helps to prevent "kinks" in the model levels from sudden changes in the interface.
- **$\mathbf{h_{ref}}$:** The topography switching depth. Generally chosen to avoid compacting the model levels. Usually this forces h_{ref} to be a value on the upper slope or even at the shelfbreak.
- **$\mathbf{z_{c_1}}$:** The shallowest depth of the interface. This one is problematic. With typical choices for the other three parameters, the intuitive choices for z_{c_1} fail to produce a coordinate interface which clears the topography. The trouble is that in choosing α and h_{ref} in the suggested manner, the coordinate interface isn't responsive enough to reach z_{c_1} before penetrating the topography. The solution is to set z_{c_1} to an outrageous value, usually above sea level, to "yank" the coordinate interface above the topography. Fortunately, GRIDS provides a mechanism to choose such a value (§4.IX). When setting up a stand–alone domain, I always use this mechanism with `opt_iface` equal to 2.

5.4.4: Nested Domains

The 2–way nesting algorithm in the HOPS–PE assumes that all domains have the same vertical coordinates (i.e. no interpolations are done in the vertical). This requires setting up the domains such that the types of vertical systems, discretizations and coordinate interfaces are identical for all domains. The trickiest part of this is forcing equal coordinate interfaces for double σ systems. The general rule is to set–up the largest domain first and use the automatic tuning of z_{c_1}. Then directly input the resulting value (§6.2.5) into the subdomains (§4.IX). Don't forget to make sure the topographies are aligned (§5.3.4).

CHAPTER 6: GRIDS OUTPUT

6.1: Introduction

GRIDS produces several types of output: an ASCII echo of the state of the run, Graphical representations of the results and a binary (netCDF) file to transmit the results to the rest of HOPS. This chapter provides a brief overview to each of these types of output

6.2: ASCII Echo

GRIDS produces a stream of information describing the state of the run. This section dissects a sample of the echo (table 20).

6.2.1: Run & Model Identifier

This portion of the echo describes when the run occurred and how GRIDS was configured. The specific information is:

- The time and date of the GRIDS run.
- The number of the particular version of GRIDS which was used.
- A list of which options (§3.4) were used when this particular instance of GRIDS was compiled.

6.2.2: Run Input

Simply put, this section echoes the input parameters from sections I-VII of the input file (§4.I–4.VII). A useful starting point to make sure that the desired instructions were correctly read by GRIDS.

6.2.3: Domain Limits

This section reports on the four corners of the tracer (T) and velocity (V) grids (§2.4.1). Any automatic adjustments to the plotting window are also reported here (§4.V).

6.2.4: Topography Report

This section reports on any manipulations performed on the topography (`new_topo`=*false*, §4.V). If any manipulations take place, the input control parameters (§4.VIII) are reported here.

Always reported in this section are the following topography statistics: the shallowest depth, the mean depth, the median depth and the deepest depth. These should always be examined to be sure the topography is "legal" (§5.3.1) and to see how the gross characteristics have been changed by the conditioning.

6.2.5: Coordinate Interface

This section reports on the coordinate interface definition. It first reports on the control parameters which were read in. It then reports on any automatic adjustments made to the coordinate interface.

- For σ and hybrid systems, the interface is automatically set to the sum of the first k_c thicknesses.
- For the double σ system, the shallow limit parameter, z_{c_1}, can be adjusted through a Newton search (§4.IX).

Table 20: Sample ASCII Echo output produced by GRIDS. The echo file has been broken up into sections. The section numbers correspond to the subsection numbers in §6.2.

(1)

```
                  Sunday - June 14, 1998 - 7:54:55 pm
GRIDS,  Version 6.1
Active C Pre-Processing options:
   FSMED          Filter step topography with integral-weighted median filter
   FSMEN          Filter step topography with integral-weighted mean filter
   FTCLIP         Filter topography by clipping
   FTMED          Filter topography with integral-weighted median filter
   LNEXT          Use linear extrapolation boundary conditions on median filter
   FTSHAP         Filter topography with Shapiro filter
   SUNDATE        Using SUN time/date call
   SUNFLUSH       Flush output buffers on SUN systems
```

(2)

```
Grid data file will be:    grids_nwa_rotsph.nc

coord =          2
Generating rotated spherical grid.
spherical rotation parameters:
   dx =  0.2700000107288361     (degrees)
   dy =  0.2700000107288361     (degrees)
   lonc =  -66.69999694824219     (degrees)
   latc =   42.00000000000000     (degrees)
   delx =  -1.417500019073486     (degrees)
   dely =  -2.160000085830688     (degrees)
   rotang =   25.50000000000000     (degrees)
   no_x =   66
   no_y =   42
Number of vertical levels:                    25
Thicknesses of T-boxes:
     3.0000000      (m)
     3.2952533      (m)
     3.6195648      (m)
     3.9757943      (m)
     4.3670831      (m)
     4.7968817      (m)
     5.2689805      (m)
     5.7875414      (m)
     6.3571386      (m)
     6.9827938      (m)
     7.6700249      (m)
     8.4248915      (m)
     9.2540512      (m)
     8.4248915      (m)
     7.6700249      (m)
     6.9827938      (m)
     6.3571386      (m)
     5.7875414      (m)
     5.2689805      (m)
     4.7968817      (m)
     4.3670831      (m)
     3.9757943      (m)
     3.6195648      (m)
     3.2952533      (m)
     3.0000000      (m)
hybrid flag = T
kc =                     13
optimize sigma system?    F
Plot Window:
```

```
                         Longitude     Latitude
                         (degrees)     (degrees)
   Upper left corner:       -81.023       48.386
   Lower right corner:     -53.232       30.446
coastline data file:    /home/mare/haley/Oceanplot/Coast_data/gulf_cst.dat
new_topo = F
Generating topography at T grid points from old file.
   topography input file:    grids_nwa_rotsph.nc
Topography contouring:
   Number of contour levels =        -26
   Block contour lines through high/low labels:    T
   High/low search radius in x =        100
   High/low search radius in y =        100
   User specified regular contours.
      Minimum contour level:      -5000.0
      Contour interval:       200.00
Hydrostatic consistency factor contouring:
   Number of contour levels =         0
   Block contour lines through high/low labels:    T
   High/low search radius in x =        100
   High/low search radius in y =        100
   Contour levels automatically picked.
Plot grid?    F
Contour topography?    T
Contour hydrostatic consistency factor?    T
```

(3)

```
The geographic (longitude,latitude) coordinates of the T-grid corners.  (degrees)
( -80.622    ,  39.822    )--------------------------( -58.970    ,  47.975    )
             |                                                    |
             |                                                    |
             |                                                    |
( -73.403    ,  30.446    )--------------------------( -54.169    ,  37.476    )
The geographic (longitude,latitude) coordinates of the UV-grid corners.  (degrees)
( -80.576    ,  40.009    )--------------------------( -58.851    ,  48.149    )
             |                                                    |
             |                                                    |
             |                                                    |
( -73.348    ,  30.630    )--------------------------( -54.057    ,  37.644    )
```

(4)

```
Topography filtered by alternating direction Shapiro Filter.
   Filter order:         4
   Number of applications:          0
Topography clipped to prescribed maximum & minimum.
   minimum depth =  -10.000000      (m)
   maximum depth =  -12000.000      (m)
Topography filtered by weighted median with nearest neighbors.
   Process repeated until no changes occur.
     30 iterations.
   Maximum allowed iterations:      200
Topography statistics:
   shallowest depth =  -10.000000     (m)
     mean  depth =  -2726.7522     (m)
    median depth =  -3573.5156     (m)
   deepest depth =  -5414.0000     (m)
```

(5)

```
Coordinate interface parameters:
   zc1 =  -5.0000000      (m)
   zc2 =  -1500.0000      (m)
   alpha =  0.40000001
   href =  -2000.0000      (m)
   c (upper) =   2.0000000
   c (lower) =   8.1615868
   zc1_tol =  9.99999996E-13
   zc1_itmax =         200
   opt_iface =           2
Resetting double sigma parameter zc1:
   Shallowest interface depth:      -5.3393593     (m)
                 Required zc1:        232.58028     (m)
            Successful search:    T
```

(6)

```
Depths of sigma levels (m):
   Level         Minimum             Median            Maximum
         1   -0.11001419        -24.140825         -29.443554
         2   -0.34086990        -74.798363         -91.228432
         3   -0.59444594        -130.44150         -159.09404
         4   -0.87297833        -191.56090         -233.63881
         5    -1.1789232        -258.69553         -315.52011
         6    -1.5149786        -332.43744         -405.46002
         7    -1.8841078        -413.43683         -504.25162
         8    -2.2895660        -502.40805         -612.76605
         9    -2.7349284        -600.13556         -731.96021
        10    -3.2241220        -707.48120         -862.88519
        11    -3.7614615        -825.39160         -1006.6956
        12    -4.3516846        -954.90643         -1164.6594
        13    -4.9999962        -1097.1678         -1338.1698
        14    -5.6483088        -1330.8550         -1693.1591
        15    -6.2385325        -1635.0291         -2197.8201
        16    -6.7758722        -1911.9492         -2657.2637
        17    -7.2650666        -2164.0574         -3075.5410
        18    -7.7104301        -2393.5769         -3456.3413
        19    -8.1158886        -2602.5317         -3803.0222
        20    -8.4850187        -2792.7642         -4118.6406
        21    -8.8210745        -2965.9519         -4405.9795
        22    -9.1270199        -3123.6221         -4667.5732
        23    -9.4055529        -3267.1650         -4905.7280
        24    -9.6591301        -3397.8469         -5122.5449
        25    -9.8899860        -3516.8196         -5319.9346
```

(7)

```
Maximal hydrostatic consistency factor (upper) :
   Minimum =          0.
      Mean =   1.1388792
    Median =  5.12853786E-02
   Maximum =   14.364388
Maximal hydrostatic consistency factor (lower) :
   Minimum =          0.
      Mean =   5.1835351
    Median =  0.56589329
   Maximum =   57.243057
Theoretical upper bound for hydrostatic consistency factor =   22.429768
Program successfully terminated.
```

6.2.6: Representative Model Levels

In this section GRIDS writes the actual depths of the model levels at the shallowest, the median and the deepest topography locations. After the topography has undergone the first level conditioning (§5.3.1), these depths should be *negative* and *strictly monotonically* decreasing. If they aren't, then something is wrong either with the topography or with the coordinate interface. This section should **always** be examined as a check on the validity of the vertical system (especially the depths over the shallowest topography).

6.2.7: Hydrostatic Consistency Factor

This section reports on the statistics of the hydrostatic consistency factor, hcf (§2.5.2), in the upper and lower vertical coordinate systems. It also reports on the maximum expected value for hcf if GRIDS is allowed to adjust the vertical discretization to improve hcf. In this section will also appear warning messages about locations where the vertical coordinate systems will break down.

6.3: Graphical Output

If compiled with NCAR graphics (§1.3.2, §3.4.5 & §3.6), GRIDS can produce five types of plots (figure 5) of which four types can appear at any one time. They are:

<u>Horizontal Grid</u>. (figure 5a) This plot shows the horizontal grid lines together with the relevant coastlines (§4.V). Primarily used to position the grid and get a feel for the horizontal resolution. The existence of this plot is controlled by the input parameter `plot_grid` (§4.VII).

<u>Topography Contours</u>. (figure 5b) This plot shows contours of the topography (dashed lines for above sea–level, solid lines for below sea–level) with the relevant coastlines (§4.V). Used to view the topographic features included by the grid and to monitor the changes during the conditioning process (§5.3). The existence of this plot is controlled by the input parameter `cont_top` (§4.VII). The contour lines can be controlled by the user (§4.VI.a).

<u>Hydrostatic Consistency Factor Contours</u>. (figure 5c, terrain following systems only) This plot shows contours of the hydrostatic consistency factor, hcf, (dashed lines for $hcf > 2$, solid lines for $hcf \leq 2$) with the relevant coastlines (§4.V). Used to view the locations with the highest vertical grid distortions. The existence of this plot is controlled by the input parameters `hybrid` (§4.IV.a) and `plot_step` (§4.VII). The contour lines can be controlled by the user (§4.VI.b).

<u>"Step Topography"</u>. (figure 5d, flat vertical systems only) If the user runs GRIDS in flat (pressure) level mode, this plot shows a representation of the "step topography" (ie. the number of model levels at a given horizontal location) along with the relevant coastlines (§4.V). This crude representation merely draws lines between adjacent nodes with equal numbers of model levels. The existence of this plot is controlled by the input parameters `hybrid` (§4.IV.a) and `plot_step` (§4.VII).

<u>Invalid Points</u>. (figure 5e, terrain following systems only) This plot highlights the locations where the vertical coordinate system has a singularity along with the relevant coastlines (§4.V). The locations of the singularities are defined by:

- σ: the topography crosses the surface of the ocean.
- **hybrid:** the topography crosses the bottom of the deepest flat box.
- **2–σ:** either (1) the topography crosses the coordinate interface or (2) the coordinate interface crosses the surface of the ocean.

The points of singularity are highlighted by simply drawing a line between adjacent singular points. The existence of this plot is governed by the value of `hybrid` (§4.IV.a) and the existence of singular points.

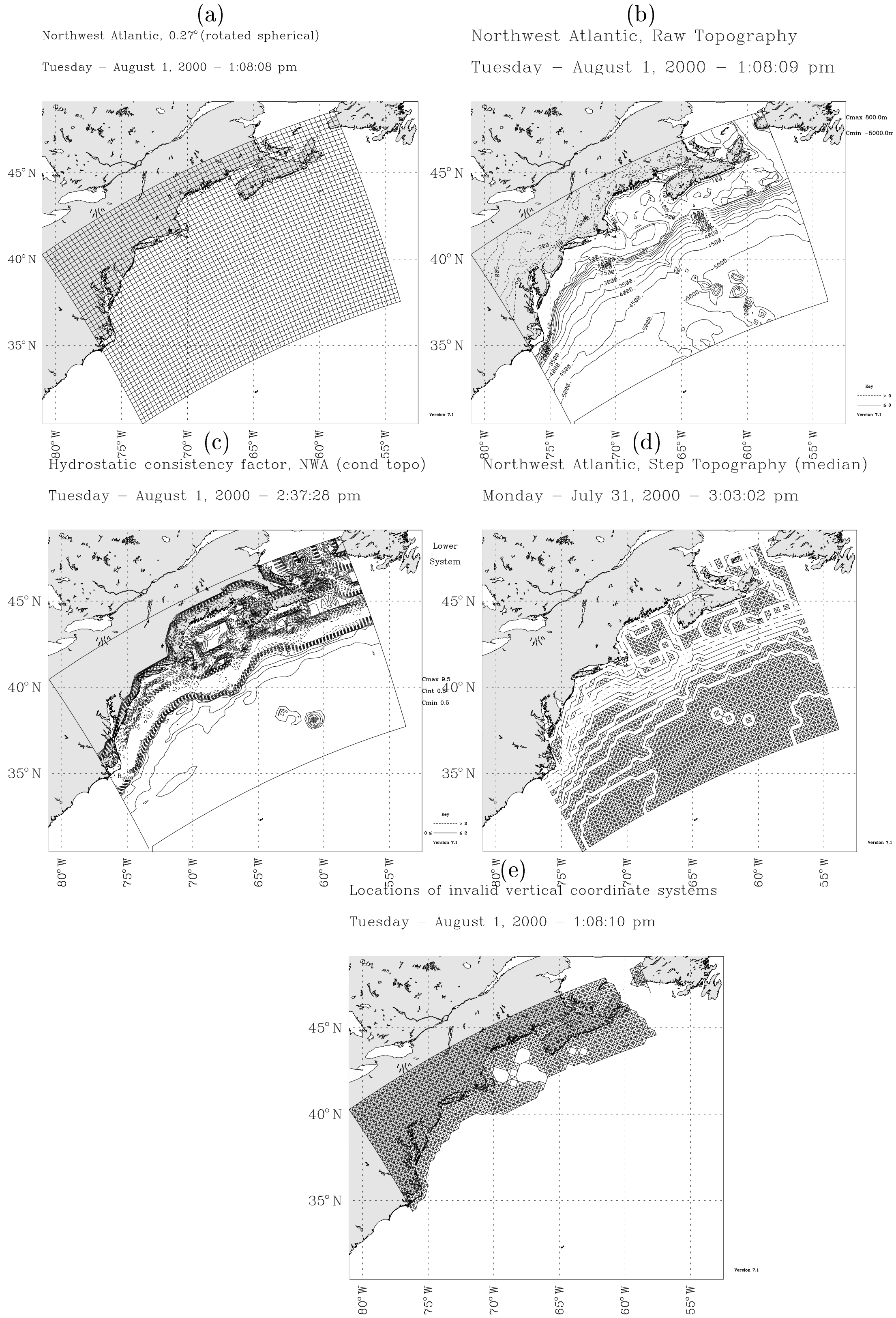

Figure 5: Types of Plots Produced by GRIDS. (a) Model Domain. (b) Topography Contours (c) Hydrostatic Consistency Factor Contours (d) Step Topography. (e) Locations of Invalid Points.

6.4: NetCDF Output

The single most important source for GRIDS output is the netCDF output file (§4.I). This binary data file is the means by which GRIDS transmits its data to the rest of HOPS. The netCDF file has two basic parts: (1) the header section which defines the data & its structure and (2) the data section which contains the actual data values. The netCDF format has three broad categories for the entities it stores.

Dimension An integer "size" used to structure the data. Analogous to an array dimension.

Variable A data "entity", either a single value or a block of data. Analogous to an array of data.

Attribute Additional information about the variables or about the netCDF file as a whole. Usually descriptive in nature, often used to store metadata directly in the netCDF file.

The contents of a netCDF file may be viewed using the `ncdump` utility provided with the netCDF package. A small sample of the results of using `ncdump` is dissected in table 21 and the remainder of this section. Only the header portion of the netCDF file is shown.

6.4.1: Dimensions

As with all netCDF files, the GRIDS output netCDF file begins with its dimensions. These are static entities, in that (with one exception) their values can't be changed. The dimensions GRIDS defines are:

Table 21: **GRIDS netCDF dimensions**

Dimension	Description	Comment
`tlat`	Number of y points tracer grid (§2.4.1)	
`tlon`	Number of x points tracer grid (§2.4.1)	
`vlat`	Number of y points velocity grid (§2.4.1)	
`vlon`	Number of x points velocity grid (§2.4.1)	
`level`	Number of vertical levels	
`axis2`	Number of coordinate axes of 2D fields	for visualization
`len80`	dimension for string of 80 characters	
`system`	Number of vertical coordinate systems.	only present if `hybrid`=*true* (§4.IV.a)

The example in table 23 is taken after the HOPS utilities PE_mask and cond_topo have been applied. Their additions to the dimension section are:

PE_mask:

Table 22: **PE_mask netCDF dimensions**

Dimension	Description	Comment
`ncseg`	maximum number of coastal segments	
`lcseg`	maximum length of coastal segments	
`island`	maximum number of islands	only present if islands were ever present

Table 23: Sample netCDF output produced by GRIDS. An ASCII representation of the netCDF output file header (produced by `ncdump`) has been broken up into sections. The section numbers correspond to the subsection numbers in §6.4.

(1)

```
netcdf grids_nwa_rotsph {
dimensions:
        tlat = 83 ;
        tlon = 131 ;
        vlat = 83 ;
        vlon = 131 ;
        level = 25 ;
        axis2 = 2 ;
        system = 2 ;
        ncseg = 20 ;
        lcseg = 2000 ;
        island = 20 ;
        bnddim = 2 ;
        len256 = 256 ;
        grad_pass = UNLIMITED ; // (1 currently)
```

(2)

```
variables:
        long imt ;
                imt:long_name = "Number of tracer points in the x-direction" ;
        long jmt ;
                jmt:long_name = "Number of tracer points in the y-direction" ;
        long km ;
                km:long_name = "Number of vertical levels" ;
        long coord ;
                coord:long_name = "Flag for type of horizontal grid" ;
                coord:option_0 = "Cartesian (tangent plane) grid" ;
                coord:option_1 = "Unrotated spherical grid" ;
                coord:option_2 = "Rotated spherical grid" ;
        float rlngd ;
                rlngd:long_name = "Transformation centroid longitude" ;
                rlngd:units = "degrees_east" ;
        float rlatd ;
                rlatd:long_name = "Transformation centroid latitude" ;
                rlatd:units = "degrees_north" ;
        float delx ;
                delx:long_name = "X-offset from transformation point to grid center"
;
                delx:units = "degrees" ;
        float dely ;
                dely:long_name = "Y-offset from transformation point to grid center"
;
                dely:units = "degrees" ;
        float thetad ;
                thetad:long_name = "Domain rotation angle" ;
                thetad:units = "degrees" ;
        float meandx ;
                meandx:long_name = "Mean x-grid spacing" ;
                meandx:units = "degrees" ;
        float meandy ;
                meandy:long_name = "Mean y-grid spacing" ;
                meandy:units = "degrees" ;
        char hybrid ;
                hybrid:long_name = "Flag for type of vertical coordinate system"
;
                hybrid:option_T = "Terrain following coordinate system" ;
                hybrid:option_F = "Flat (pressure) level coordinate system" ;
```

```
long kc ;
        kc:long_name = "Number of levels in upper system" ;
char flat ;
        flat:long_name = "Flag for type of terrain following system" ;
        flat:option_T = "Hybrid-sigma levels" ;
        flat:option_F = "Double-sigma levels" ;
char opt_lvl ;
        opt_lvl:long_name = "Flag for optimizing vertical levels" ;
        opt_lvl:option_T = "Optimize vertical levels" ;
        opt_lvl:option_F = "Do not optimize vertical levels" ;
float zc1 ;
        zc1:long_name = "Minimum depth of the coordinate interface" ;
        zc1:units = "meter" ;
float zc2 ;
        zc2:long_name = "Maximum depth of the coordinate interface" ;
        zc2:units = "meter" ;
float zref ;
        zref:long_name = "Reference depth for coordinate interface" ;
        zref:units = "meter" ;
float zslope ;
        zslope:long_name = "Slope parameter of the coordinate interface"
;
float sig_bnd ;
        sig_bnd:long_name = "Upper bound for sigma level optimization" ;
float zc1_tol ;
        zc1_tol:long_name = "Tolerance for correcting zc1" ;
long zc1_itmax ;
        zc1_itmax:long_name = "Maximum no.  of iterations to correct zc1"
;
long opt_iface ;
        opt_iface:long_name = "Flag to reset interface parameter ZC1" ;
        opt_iface:option_0 = "Leave ZC1 as input value" ;
        opt_iface:option_1 = "Minimum interface depth = input ZC1" ;
        opt_iface:option_2 = "Optimize CFL over shallowest topography" ;
```

(3)

```
float hz(level) ;
        hz:long_name = "Thicknesses of the flat grid vertical boxes" ;
        hz:units = "meter" ;
```

(4)

```
float dxt(tlon) ;
        dxt:long_name = "Zonal spacing between tracer points" ;
        dxt:units = "meter" ;
float dyt(tlat) ;
        dyt:long_name = "Meridional spacing between tracer points" ;
        dyt:units = "meter" ;
float dxu(vlon) ;
        dxu:long_name = "Zonal spacing between velocity points" ;
        dxu:units = "meter" ;
float dyu(vlat) ;
        dyu:long_name = "Meridional spacing between velocity points" ;
        dyu:units = "meter" ;
float cs(vlat) ;
        cs:long_name = "Cosine metric factors at velocity points" ;
float tng(vlat) ;
        tng:long_name = "Tangent metric factors at velocity points" ;
float cst(vlat) ;
        cst:long_name = "Cosine metric factors at tracer points" ;
float lrotang(vlat, vlon) ;
        lrotang:long_name = "Local rotation angle at velocity points" ;
        lrotang:units = "degrees" ;
float tgrid2(tlat, tlon, axis2) ;
```

```
        tgrid2:long_name = "2D grid positions at tracer points" ;
        tgrid2:axis = "1:  longitude, 2:  latitude" ;
        tgrid2:units = "degrees_east, degrees_north" ;
float vgrid2(vlat, vlon, axis2) ;
        vgrid2:long_name = "2D grid positions at velocity points" ;
        vgrid2:axis = "1:  longitude, 2:  latitude" ;
        vgrid2:units = "degrees_east, degrees_north" ;
```

(5)

```
float tbath(tlat, tlon) ;
        tbath:long_name = "Bathymetry at tracer points" ;
        tbath:units = "meter" ;
        tbath:field = "bathymetry, scalar" ;
        tbath:positions = "tgrid2" ;
        tbath:filter_0 = "Clipped:    HSHLLW    -10.0,  HDEEP -12000.0" ;
        tbath:filter_1 = "Median filter:    NFILT   31" ;
        tbath:filter_2 = "Gradient conditioning:    both" ;
        tbath:filter_3 = "Shapiro filter:    ORDER    4,  NFILT    1" ;
        tbath:filter_4 = "Clipped:    HSHLLW    -10.0,  HDEEP -12000.0" ;
float vbath(tlat, tlon) ;
        vbath:long_name = "Bathymetry at velocity points" ;
        vbath:units = "meter" ;
        vbath:field = "vel_bathymetry, scalar" ;
        vbath:positions = "vgrid2" ;
float raw_bath(tlat, tlon) ;
        raw_bath:long_name = "Raw bathymetry at tracer points" ;
        raw_bath:units = "meter" ;
        raw_bath:field = "raw_bathymetry, scalar" ;
        raw_bath:positions = "tgrid2" ;
float trg_bath(tlat, tlon) ;
        trg_bath:long_name = "Target bathymetry at tracer points" ;
        trg_bath:units = "meter" ;
        trg_bath:field = "trg_bathymetry, scalar" ;
        trg_bath:positions = "tgrid2" ;
        trg_bath:filter_0 = "Clipped:    HSHLLW    -10.0,  HDEEP -12000.0"
;
        trg_bath:filter_1 = "Median filter:    NFILT   31" ;
float interface(tlat, tlon) ;
        interface:long_name = "Coordinate interface at tracer points" ;
        interface:units = "meter" ;
        interface:field = "interface, scalar" ;
        interface:positions = "tgrid2" ;
float viface(tlat, tlon) ;
        viface:long_name = "Coordinate interface at velocity points" ;
        viface:units = "meter" ;
        viface:field = "vel_interface, scalar" ;
        viface:positions = "vgrid2" ;
float hcf(vlat, vlon, system) ;
        hcf:long_name = "Hydrostatic consistency factor" ;
        hcf:system = "upper; lower" ;
        hcf:field = "hcf, scalar" ;
        hcf:positions = "vgrid2" ;
        hcf:_FillValue = -1.f ;
```

(6)

```
long ncseg ;
        ncseg:long_name = "number of coastal segments" ;
long lcseg ;
        lcseg:long_name = "maximum length of coastal segments" ;
long nisle ;
        nisle:long_name = "number of islands" ;
long lencoast(ncseg) ;
        lencoast:long_name = "length of coastal segments" ;
```

```
        long icoast(ncseg, lcseg) ;
                icoast:long_name = "coastal segment x-index coordinate" ;
                icoast:valid_min = 1 ;
                icoast:valid_max = 131 ;
                icoast:_FillValue = 0 ;
        long jcoast(ncseg, lcseg) ;
                jcoast:long_name = "coastal segment y-index coordinate" ;
                jcoast:valid_min = 1 ;
                jcoast:valid_max = 83 ;
                jcoast:_FillValue = 0 ;
        long landt(tlat, tlon) ;
                landt:long_name = "land/sea mask at tracer points" ;
                landt:option_0 = "land" ;
                landt:option_1 = "sea" ;
                landt:field = "Tmask, scalar" ;
                landt:positions = "tgrid2" ;
                landt:pe_seaseg = 4 ;
        long landv(vlat, vlon) ;
                landv:long_name = "land/sea mask at velocity points" ;
                landv:option_0 = "land" ;
                landv:option_1 = "boundary" ;
                landv:option_2 = "sea" ;
                landv:field = "Vmask, scalar" ;
                landv:positions = "vgrid2" ;
        long isis(island) ;
                isis:long_name = "starting x-index coordinate of island box" ;
                isis:valid_min = 1 ;
                isis:valid_max = 131 ;
        long ieis(island) ;
                ieis:long_name = "ending x-index coordinate of island box" ;
                ieis:valid_min = 1 ;
                ieis:valid_max = 131 ;
        long jsis(island) ;
                jsis:long_name = "starting y-index coordinate of island box" ;
                jsis:valid_min = 1 ;
                jsis:valid_max = 83 ;
        long jeis(island) ;
                jeis:long_name = "ending y-index coordinate of island box" ;
                jeis:valid_min = 1 ;
                jeis:valid_max = 83 ;
```

(7)

```
        long crt_flg(grad_pass) ;
                crt_flg:long_name = "Flag for conditioning criterion." ;
                crt_flg:option_0 = "Control reduced slope." ;
                crt_flg:option_1 = "Control slope." ;
                crt_flg:option_2 = "Control both." ;
                crt_flg:_FillValue = -1 ;
        float slope_bound(grad_pass, bnddim) ;
                slope_bound:long_name = "Maximum allowed value for slope factor."
;
                slope_bound:_FillValue = -1.f ;
        float hshllw(grad_pass) ;
                hshllw:long_name = "Shallowest depth to condition." ;
                hshllw:units = "meter" ;
                hshllw:_FillValue = 99999.f ;
        float hdeep(grad_pass) ;
                hdeep:long_name = "Deepest depth to condition." ;
                hdeep:units = "meter" ;
                hdeep:_FillValue = 99999.f ;
        long itmax(grad_pass) ;
                itmax:long_name = "Maximum number of conditioning iterations." ;
```

```
                itmax:_FillValue = -1 ;
        long medfrq(grad_pass) ;
                medfrq:long_name = "Frequency to apply median filter." ;
                medfrq:_FillValue = -1 ;
        char addcnst(grad_pass, len256) ;
                addcnst:long_name = "Additional constraints on search area." ;
```

(8)

```
    // global attributes:
                :title = "Grid definition for PE model" ;
                :name = "grids_nwa_rotsph.nc" ;
                :creation_date = "Monday - June 22, 1998 - 2:59:24 pm" ;
                :type = "GRIDS" ;
                :version = "6.1" ;
                :topo_infile = "gom_topo.nc" ;
                :longitude_convention = "positive east" ;
                :latitude_convention = "positive north" ;
                :vertical_convention = "positive above sea-level" ;
                :grid_title = "Northwest Atlantic, 0.27$S1PGU1$Z$PRU$ (rotated spherical)"
;
                :topo_title = "Northwest Atlantic, Raw Topography" ;
                :vert_title = "Hydrostatic consistency factor, NWA (Raw)" ;
                :mask = "Land/Sea Mask - PE_MASK (Matlab), Version 3.1 Monday - June
22, 1998 - 3:39:26.3829pm" ;
                :condition = "cond_topo (Matlab) Version 3.3 - Monday - June 22,
1998 - 3:43:55.6971pm" ;
    }
```

cond_topo:

Table 24: **cond_topo netCDF dimensions**

Dimension	Description	Comment
`bnddim`	maximum number of simultaneous bounds	
`len256`	dimension for string of 256 characters	some instances might have larger dimension
`grad_pass`	number of passes through cond_topo	unlimited: this dimension can change

6.4.2: Run Parameters

This section of the GRIDS netCDF output netCDF file contains the parameters that control the GRIDS run. The vast majority of these parameters were input by the user (although some are under different names).

Always Present:

The variables `imt` through `hybrid` are always present in a GRIDS netCDF file. All the variables in this section are echoes of the input file, with the following name changes:

Table 25: **Input Parameters**

netCDF name	Input file name
`imt`	`nx`
`jmt`	`ny`
`km`	`no_lvl`
`rlngd`	`lonc`
`rlatd`	`latc`
`thetad`	`rotang`
`meandx`	`dx`
`meandy`	`dy`

Also note the `option` attributes of the variable `coord`. These attributes give the meaning of `coord` having the indicated value (e.g. `option_0` describes `coord`=0)

Present if `hybrid` = *true*:

The variables `hc` through `opt_iface` are used to describe the terrain following vertical coordinate system. As such, they are only present if `hybrid` was set to *true* (§4.IV.a). They are all input echoes (with the exception of `flat` which is determined internally in GRIDS) with the following name changes:

Table 26: **Terrain Following Input**

netCDF name	Input file name
`opt_lvl`	`optimum`
`zref`	`href`
`zslope`	`alpha`
`sig_bnd`	`C`

6.4.3: Vertical Discretization

This section of the GRIDS netCDF output netCDF file contains the model level thicknesses. For a terrain following system, the thicknesses are taken at a point of typical (median) topography. To recover the nondimensional ($\delta\sigma$) thicknesses, use the following formula:

$$\delta\sigma_k = \begin{cases} \frac{hz_k}{\sum_{n=1}^{kc} hz_n} & k \leq kc \\ \frac{hz_k}{\sum_{n=kc+1}^{km} hz_n} & k > kc \end{cases}$$

6.4.4: Horizontal Grid

This section of the GRIDS netCDF output netCDF file contains additional information on the horizontal grid. The variables `dxt` through `dyu` are the grid spacings. They are stored in 1D vectors, assuming that the grid spacings only vary in their coordinate direction. This feature is not currently supported, only uniform grids spacings are allowed.

The variables `cs` through `cst` are cosine and tangent metric factors. For spherical grids they vary only in the "y" direction, referenced from the center of the transformation. For Cartesian grids, the cosines are identically 1 and the tangents are zero.

The variables `tgrid2` and `vgrid2` are the geographic coordinates of each grid point. They are provided for visualization purposes. Also note the `axis` attributes. These describe the order in which the data appears (*i.e.* when `axis2`=1, the longitude for every point is given.) (Note: these are the FORTRAN values of the indices, for C subtract 1.)

6.4.5: Topography

This section of the GRIDS netCDF output netCDF file contains topography data. Always present are the variables `tbath` & `vbath`, which contain the topography at the tracer & velocity points, respectively. Of special note are the filter attributes of `tbath`. These maintain a record of the modifications made to the original (raw) topography. Also note the `positions` attributes, they indicate which variables describe the positions of the topography points.

Present if `hybrid` = *true*:

The remaining variables in this section of table 23 are only present for terrain following systems.

Present if `hybrid` = *false*:

For flat (pressure) level systems (`hybrid`=*false*, §4.IV.a), the variable `kmt` appears in this section (not shown in table 23). This variable is an integer array containing the number of active model levels at each horizontal grid point.

6.4.6: PE_mask additions

This section of the GRIDS netCDF output netCDF file contains data written by the HOPS utility PE_mask. Of particular interest in the attribute `pe_seaseg` of the variable `landt`. This attribute gives the maximum number of sea segments in any row or column of the current mask. When compiling the HOPS PE model, the parameter `lseg` should never be set less then the value of `pe_seaseg` plus 1. (The plus 1 is due to a question of the counting and should be eliminated in future releases.)

6.4.7: cond_topo additions

This section of the GRIDS netCDF output netCDF file contains data written by the HOPS utility cond_topo. The information content is the controlling input to cond_topo, thereby allowing recreations of the current topography.

6.4.8: Global Attributes

This section of the GRIDS netCDF output netCDF file contains the attributes describing the file. Among these are the time and date the file was first created and by which version of GRIDS. Similar information is written by the utilities pe_mask and cond_topo. An additional piece of information is the name of the original topography archive data file from which the GRIDS file topography was extracted. This file name is stored in the global attribute `topo_infile`.

Bibliography

Arakawa, A. and V.R. Lamb (1977) Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model, *Methods in Comp. Phys.,* **17,** 173–265.

Haney, R.L. (1991) On the pressure gradient force over steep topography in sigma coordinate ocean models, *J. Phys. Oceanog.,* **21,** 610–619.

Wessel, P., and W.H.F. Smith. (1996) A global self-consistent, hierarchical, high-resolution shoreline database, *J. Geophys. Res.,* **101**(B4), 8741–8743.
(`http://www.ngdc.noaa.gov/mgg/shorelines/gshhs.html`)