COND_TOPO: Topographic conditioning in MATLAB

Version 3.5

Patrick J. Haley Jr., Carlos J. Lozano

August 14, 2001

`COND_TOPO` is a series of `MATLAB` executable scripts designed to reduce the slope of the topography for use in the HOPS Primitive Equation model. These scripts are designed to modify only the topography in areas where the slope exceeds a user–specified value according a user–specified measure.

This document is divided into four basic sections:

Introduction: This section. Describes what `cond_topo` and what is needed to use it.

Theory: Contains the derivation of the "conditioning problem". Primarily of use to those who wish to modify `cond_topo` to work outside of HOPS.

Implementation: Some notes on the implementation of the "conditioning problem" within `MATLAB`. Primarily of use to those who wish to modify `cond_topo` to work outside of HOPS.

Use: Instructions on running `cond_topo` in `MATLAB`.

Required software

(1) **cond_topo** This package. Obtain by 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/Cond_Topo`

(2) **GRIDS** HOPS software to generate the netCDF file read by `cond_topo`. Obtain by 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`

(3) **NetCDF** The data format for the GRIDS topography file. For further information see:

`http://www.unidata.ucar.edu/`

(4) **MATLAB** Provides the environment in which `cond_topo` is run. For further information see:

`http://www.mathworks.com/`

(5) **NetCDF Toolbox** Interface between MATLAB & netCDF. For further information see:

`http://crusty.er.usgs.gov/~cdenham`

New in version 3.5

- *Nesting.* The input file has been changed to allow nested subdomains to maintain their boundaries consistent with the larger domain.

“Theory”

Conditioning

The extra conditioning of topography in our `MATLAB` scripts represents two types of gradient constraints:

$$\frac{\|\nabla h\|}{\mathcal{G}(h)} \le C_{RS} \tag{1a}$$

$$\|\nabla h\| \le C_S \tag{1b}$$

where h is the topographic depth. (1a) is a restriction on the reduced slope while (1b) is a restriction on the absolute slope. The first question is how to define a discrete gradient of topography. For HOPS this is answered by the following observations:

- The HOPS PE model uses the Arakawa B – grid. That is, tracers and transport streamfunction are located on one horizontal grid (T–grid) while velocities are located on another horizontal grid (V–grid) shifted $\frac{1}{2}$ grid–point to the northeast.
- The constraints are used to help the pressure gradient. The pressure gradient is evaluated on the V–grid.

The upshot is: the discrete gradient is defined as

$$\begin{aligned}(\nabla h)^v_{i,j} =& \frac{\hat{\imath}}{2\Delta x^v_{i,j}\mu^v_{i,j}}\left[\left(h_{(i+1),(j+1)} - h_{i,(j+1)}\right) + \left(h_{(i+1),j} - h_{i,j}\right)\right] + \\ & \frac{\hat{\jmath}}{2\Delta y^v_{i,j}\eta^v_{i,j}}\left[\left(h_{(i+1),(j+1)} - h_{(i+1),j}\right) + \left(h_{i,(j+1)} - h_{i,j}\right)\right]\end{aligned} \tag{2}$$

where $\hat{\imath}$ & $\hat{\jmath}$ are the zonal & meridional unit vectors, Δx & Δy are the zonal & meridional grid spacings and μ & η are the zonal & meridional metric factors. The superscript v denotes a quantity evaluated on the V–grid.

Having defined the gradient, the next question is what action should be taken in areas in which condition (1) is violated? The first part of the answer to this question is to define the areas of violation. Looking at (1) & (2), an area of violation is defined as follows:

> Around points where $(\nabla h)^v_{i,j}$ violates condition (1), the **entire** set of $\{h_{i,j}, h_{(i+1),j}, h_{i,(j+1)}, h_{(i+1),(j+1)}\}$ are considered to be in violation and are included in the set of points to change.[1]

Having identified the points to modify, it's now time to state the method of modification. Over the areas of violation a Poisson equation is set–up to meld the “corrected” gradient data. The Poisson equation is chosen because it gives a well–posed problem to

[1] This is different from the previous versions which attempted to assign a gradient value to each topographic point and assess each point individually

combine the gradient information. To derive the forcing function for this Poisson equation, first look at the Laplacian of the topography:

$$\nabla^2 h_{i,j} = \frac{1}{(2\mu\eta)_{i,j}} \begin{aligned} \Bigg\{ & \frac{1}{\Delta x_{i,j}} \Bigg[\left(\frac{\eta}{\mu}\frac{\partial h}{\partial x}\right)^v_{i,j} - \left(\frac{\eta}{\mu}\frac{\partial h}{\partial x}\right)^v_{(i-1),j} + \\ & \qquad \left(\frac{\eta}{\mu}\frac{\partial h}{\partial x}\right)^v_{i,(j-1)} - \left(\frac{\eta}{\mu}\frac{\partial h}{\partial x}\right)^v_{(i-1),(j-1)} \Bigg] + \\ & \frac{1}{\Delta y_{i,j}} \Bigg[\left(\frac{\mu}{\eta}\frac{\partial h}{\partial y}\right)^v_{i,j} - \left(\frac{\mu}{\eta}\frac{\partial h}{\partial y}\right)^v_{i,(j-1)} + \\ & \qquad \left(\frac{\mu}{\eta}\frac{\partial h}{\partial y}\right)^v_{(i-1),j} - \left(\frac{\mu}{\eta}\frac{\partial h}{\partial y}\right)^v_{(i-1),(j-1)} \Bigg] \Bigg\} \end{aligned} \tag{3}$$

Since this is evaluated in an area of violation at least one of the partial derivatives in the right–hand side of (3) violates (1) or contributes to a violation of (1). Conversely, some of the partial derivatives in the right–hand side of (3) **may** satisfy (1). These two observations drive the definition of the "conditioning" Poisson problem.

Define the "conditioning" Poisson problem as:

$$\nabla^2 h_{i,j} = F(h) \tag{4}$$

where $F(h)$ is defined by the right–hand side of (3). In the forcing function, $F(h)$, replace the partial derivatives of h which are in violation of (1) by values which satisfy (1).[2] The Laplacian operator in (4) is discretized by continuing the discretization of (3) in the manner of (2).

The remaining issue in defining the "conditioning" Poisson problem is determining the replacement values for $F(h)$. These values depend upon which condition is active.

Replacement Procedure for Reduced Slope Conditioning

For the reduced slope constraint (1a) a max norm is used. This requires that both

$$\frac{1}{\mu}\left|\frac{\partial h}{\partial x}\right| \le C_{RS}\mathcal{G}(h) \tag{5a}$$

$$\frac{1}{\eta}\left|\frac{\partial h}{\partial y}\right| \le C_{RS}\mathcal{G}(h) \tag{5b}$$

The function $\mathcal{G}(h)$ depends on the type of vertical system:

$$\mathcal{G}(h) = \begin{cases} |h| & \text{sigma} \\ |h - z_c| & \text{hybrid} \\ |h - f(h)| & \text{double sigma} \end{cases} \tag{6}$$

[2] This is also different from the previous versions which would set $F(h)$ to an arbitrary value, usually zero. This new approach preserves more of the characteristics of the original topography.

The resulting replacement scheme is

Replace	**With**	**If Violate**
$\frac{1}{\mu}\frac{\partial h}{\partial x}$	$C_{RS}\mathcal{G}(h)\,\text{sign}\left(\frac{\partial h}{\partial x}\right)$	(5a)
$\frac{1}{\eta}\frac{\partial h}{\partial y}$	$C_{RS}\mathcal{G}(h)\,\text{sign}\left(\frac{\partial h}{\partial y}\right)$	(5b)

Replacement Procedure for Absolute Slope Conditioning

For the absolute slope constraint (1b) a Euclidean norm is used. This requires that

$$\sqrt{\left(\frac{1}{\mu}\frac{\partial h}{\partial x}\right)^2 + \left(\frac{1}{\eta}\frac{\partial h}{\partial y}\right)^2} \leq C_S \tag{6}$$

Where (6) is violated, the replacement scheme is to scale the corresponding partial derivatives by the factor

$$\frac{C_S}{\sqrt{\left(\frac{1}{\mu}\frac{\partial h}{\partial x}\right)^2 + \left(\frac{1}{\eta}\frac{\partial h}{\partial y}\right)^2}} \quad . \tag{7}$$

Non–conditioning Points

The remainder of the domain falls into two categories: interior and boundary points. The methodology for the boundary points, in turn, is divided into two categories: stand–alone and nested.

Interior Non–conditioning Points

For interior points satisfying (1) the identity

$$h_{i,j} = h_{i,j} \tag{8}$$

is applied.[3]

Stand–alone Boundary Non–conditioning Points

For boundary points in the stand–alone configuration, a *no–normal* condition is applied:

$$\begin{gathered}
\left.\begin{aligned} h_{1,j} - h_{2,j} &= 0 \\ h_{IM,j} - h_{(IM-1),j} &= 0 \end{aligned}\right\} j \in [2,\ JM-1] \\
\left.\begin{aligned} h_{i,1} - h_{i,2} &= 0 \\ h_{i,JM} - h_{i,(JM-1)} &= 0 \end{aligned}\right\} i \in [2,\ IM-1] \\
h_{1,1} - h_{2,2} = 0 \\
h_{IM,1} - h_{(IM-1),2} = 0 \\
h_{1,JM} - h_{2,(JM-1)} = 0 \\
h_{IM,JM} - h_{(IM-1),(JM-1)} = 0
\end{gathered} \tag{9}$$

[3] This is yet another difference from the previous versions. There the identity $\nabla^2 h = \nabla^2 h$ is used. The current identity is both simpler and more direct, which should aid in fidelity.

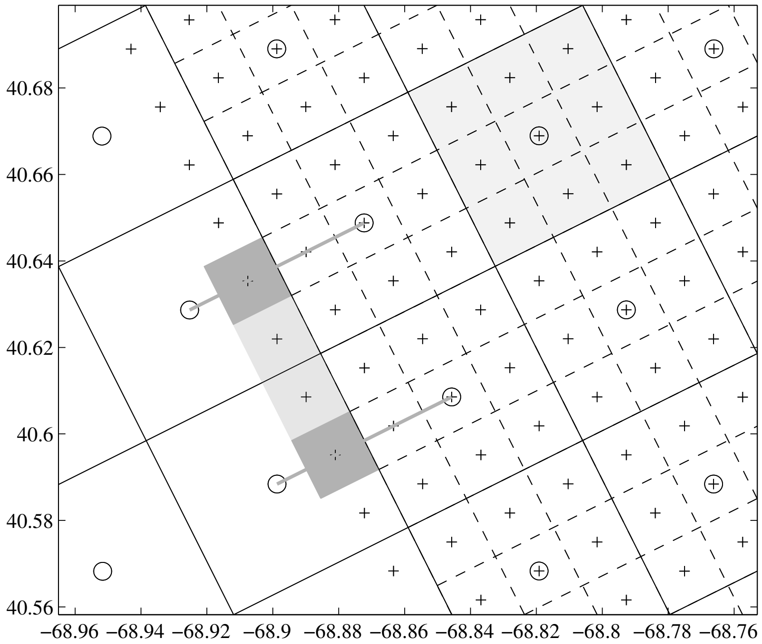

Figure 1: Illustration of simplified nesting. Coarse grid tracer boxes are drawn in solid lines with their centers marked by open circles. Fine grid tracers boxes are drawn in dashed lines with plus signs in their centers. Lightest gray shaded regions shows the 9 fine grid boxes which average to a single coarse grid box. Fine grid boundary boxes aligned with coarse grid centers (darkest gray) are linearly interpolated from the adjacent coarse grid values. Remaining fine grid boundary boxes (medium gray) are interpolated from dark gray values.

This no–normal is incorporated into the conditioning matrix.[4] By applying this condition everywhere along the boundary a couple of benefits are realized:

- The conditioning problem is "made aware" of the floating boundary, rather than trying to work with a static (and possibly poorly conditioned) boundary.
- No special discretization is needed at the boundary.

Nested Boundary Non–conditioning Points

The no–normal condition leads to a pernicious problem for domains to be nested within larger domains. The boundaries of the smaller domain tend to drift away from the corresponding large domain points. This can produce a significant discontinuity in the nesting.

One solution to this problem is based on interpolation. Two simplifying approximations are introduced into the nesting scheme.

(1) Metric factors and variable grid spacings are ignored when averaging fine grid values onto coarse grid points.

[4] The differences keep rolling. In the older versions, the no–normal condition was only applied between conditioning iterations.

(2) Bilinear interpolation is used to obtain fine grid boundary values from coarse grid estimates.

With these assumptions, the nesting can be represented by the schematic in figure 1. In this figure and derivation the boundary points are divided into two classes: (1) those directly coupled to the coarse grid values (dark gray) and (2) those indirectly coupled through the first class (medium gray). We will show the derivation for a typical western boundary point only.

The directly coupled boundary points on the western boundary are identified by the relation $j \bmod 3 = 0$. These values are formally obtained from linear interpolation of coarse grid values as:

$$h_{1,j} = \frac{2}{3} h_{ext}^{coarse} + \frac{1}{3} h_{int}^{coarse} \tag{10}$$

where h_{ext}^{coarse} denotes a coarse grid topography point external to the fine grid and h_{int}^{coarse} denotes a coarse grid topography point internal to the fine grid. Under simplification (1), the interior coarse grid point is a simple average of the underlying 9 fine grid values. Substituting this into (10) and writing as an equation for h yields:

$$h_{1,j} - \frac{1}{27} \left\{ \begin{array}{c} h_{2,j+1} + h_{3,j+1} + h_{4,j+1} + \\ h_{2,j} + h_{3,j} + h_{4,j} + \\ h_{2,j-1} + h_{3,j-1} + h_{4,j-1} \end{array} \right\} = \frac{2}{3} h_{ext}^{coarse} \quad . \tag{11}$$

This produces a boundary value interpolated from the exterior coarse grid value and an updated interior coarse grid value.

The equations for the indirectly coupled boundary values can be derived by inspection:

$$\begin{aligned} h_{1,j} - \frac{2}{3} h_{1,j-1} - \frac{1}{3} h_{1,j+2} &= 0 \qquad (j \bmod 3 = 1) \\ h_{1,j} - \frac{2}{3} h_{1,j+1} - \frac{1}{3} h_{1,j-2} &= 0 \qquad (j \bmod 3 = 2) \end{aligned} \tag{12}$$

Implementation

Like the previous versions, the current conditioning algorithm is coded in `MATLAB` scripts. This is done because setting up the search parameters and checking the results is best done with the interactive viewing capabilities of `MATLAB`. Working blind with these types of conditioners is an **extremely dangerous** idea. These conditioners can do simply awful things to topography if left unsupervised.

One of the chief complaints with the previous incarnation of the conditioning routines was the time involved. The preceding theory attempts to aid speed by better identification of the area to be modified and better guidance in the modification, thereby reducing the required work. The implementation is also designed to improve speed:

(1) Use of `MATLAB`'s matrix manipulators[5] This uses the built–in expertise of `MATLAB`. To save space, the sparse matrix utilities are used. The matrix so created encompasses equations (4), (8) and (9).

[5] The previous incarnation used a home–grown conjugate gradient solver.

Sample Input File

```
1           CRT_FLG     Criterion flag.    Indicates which quantity is bound.
0.0         HSHLLW  (m) Minimum depth above which the search is inactive.
-12000.0    HDEEP   (m) Maximum depth below which the search is inactive.
200         ITMAX       Maximum number of iterations.
10          MEDFRQ      Frequency of applying median filter.
1           IPLOT       Flag for plotting cur.  cond.  fa.  [0] NO plot  [1] plot
0           IPRINT      Flag for printing cur.  cond.  fa.  [0] NO print [1] print
none
grids.nc
/dev/null

    *  Above 3 lines:

       ADDCNST   Character string of additional constraints on active mask.
                 These constraints will be "anded" to the constraints
                 in conditioning.    A true result signifies topography
                 available for conditioning.
                 A blank line or the string "none" deactivates this option

       NETFILE   GRIDS netCDF file.;  input

       NESTFILE  GRIDS netCDF file containing nesting super domain.;  input

Additional descriptions:

    *  CRT_FLG:  [0]  bound reduced slope:  grad(h)/h  where h is the topography.
                 [1]  bound slope:          grad(h)    where h is the topography.
                 [2]  bound both.

    *  HSHLLW, HDEEP:  Search bounds.    Negative below sea-level.
                       General warning:    having both active can lead
                       to stalled searches.

    *  MEDFRQ:  The 2D median filter is applied every MEDFRQ iterations.
                If MEDFRQ=0, the 2D median filter is never applied.    The
                median filter is included so COND_TOPO can reconcile the
                different restrictions imposed by the filter and the
                conditioning.
```

(2) Use of `MATLAB` function scripts The `MATLAB` documentation suggests these function scripts operate faster than non–function scripts.

Usage

Input file

Before executing the topographic conditioning scripts, a number of parameters governing the conditioning need to be set. They are set in an ASCII file, which is read at execution. These parameters are:

CRT_FLG Criterion flag. Indicates which quantity is being bounded by the conditioning.

- [0] Bound reduced slope, equation ($1a$).
- [1] Bound absolute slope, equation ($1b$).
- [2] Bound both, equation (1). At any point the actual constraint applied is that which is in greater violation relative to the desired bounds.

HSHLLW Shallow bound. Indicates the shallowest depth that can be conditioned. The topographic conditioning only acts in the range `HSHLLW` $< h <$ `HDEEP`.

HDEEP Deep bound. Indicates the deepest depth that can be conditioned. The topographic conditioning only acts in the range `HSHLLW` $< h <$ `HDEEP`.

ITMAX Maximum number of iterations. The conditioning script repeatedly solve the conditioning problem described in the previous section. This is necessary because, among other reasons, after one pass through the Poisson problem the boundaries around the active areas may now be in violation.

MEDFRQ Frequency of applying median filter. This parameter controls the frequency of the intermittent median filter. The filtering is done because the conditioning process and the median filter are trying to control two different problems. Filtering significantly slows down a single pass through `cond_topo` but does speed up a repeated cycle of conditioning followed by median filtering in GRIDS. Setting `MEDFRQ = 0` prevents median filtering.

IPLOT Plotting switch. This parameter allows the user to see a continuously updated plot of the current maximum slope factor (i.e. the maximum value of the left hand side of eq. 1). This can be a good monitor of the status of the conditioning. This continuously refreshed plot can also be a damned nuisance on a slow or remote machine.

[0] Do not plot the current maximum slope factor.
[1] Plot the current maximum slope factor

IPRINT Printing switch. This parameter causes the conditioning scripts to print the current maximum slope factor (i.e. the maximum value of the left hand side of eq. 1) after each iteration. Not as useful as the plot and it does fill the screen.

[0] Do not print the current maximum slope factor.
[1] Print the current maximum slope factor

ADDCNST Additional constraints. This is a character string which allows the user to "and" additional constraint to the depth limits (`HSHLLW` & `HDEEP`).

Ex: `~((topo_old>-300) & (lat>=34) & (lat<=36.5) & (lon<=16))`
Prevents topography shallower than 300m in the box 34 $\leq$ latitude $\leq$ 36.5 and longitude $\leq$ 16 from being modified. This constraint was used to preserve an inflow sill.

Some of the variables available to participate in the constraint:

`topo_old` The current topography.
`lon` The longitudes of the topographic positions.
`lat` The latitudes of the topographic positions.

NETFILE GRIDS netCDF file. This is the name of the GRIDS netCDF file containing the topography and the discretization parameters.

NESTFILE GRIDS netCDF file. This is the name of the GRIDS netCDF file containing the topography and the discretization parameters for the larger domain within which the current domain (`NETFILE`) is nested.

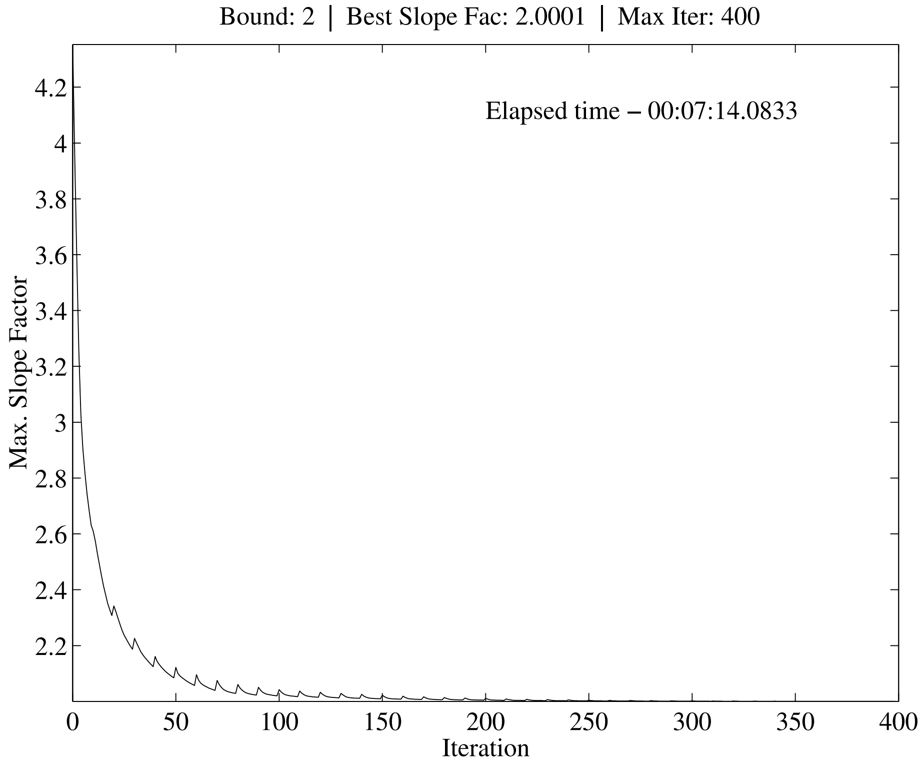

Figure 2: Example of the plot produced when `IPLOT` is set to 1. The jags are produced by the median filter which is applied every 10 steps. The final maximum slope factor was 2.00028 (after the last application of the median filter).

Execution

The topographic conditioning package comes in three basic pieces: initialization, conditioning and writing. The reason for this break–up is to allow the user to examine the input & output. Once the necessary parameters and data are loaded, the user should check to see that they make sense. Once the conditioning is done, the user should check to see that the resultant topography is reasonable. To aid in this, four simple plotting scripts are included:

- `plot_topo` Produces a colored map of the original topography. Usually done with filled contours.
- `plot_topo_new` Produces a colored map of the final, conditioned topography. Usually done with filled contours.
- `plot_curr_fac` Produces a colored map of the current slope factor (left–hand side of eq. 1). Also draws contours of the limiting topography values (`HSHLLW` & `HDEEP`). Usually done with filled contours. (If `crt_flg`=2, then the left–hand sized are normalized by the right–hand sides and the plot displays the larger of the two values on a pointwise basis.)
- `plot_topo_diff` Produces a colored map of the absolute difference between the original and the conditioned topography. Also draws contours of the conditioned topography. Usually done with filled contours.

Initialization (cond_init)

To start, the user must initiate a MATLAB session. At the prompt the user types the command cond_init and is prompted for the name of the parameter input file. To use the default name (in square brackets) simply hit the return key.

```
>> cond_init
Enter name of parameter input file [condition.in] :  cond_dhoh_01.in
Bound for slope factor         = 2.1765
Current maximum slope factor = 4.353
```

The script cond_init then reads the parameter file, opens & extracts the necessary data from the GRIDS netCDF file and computes the default slope factor bounds (right–hand side of eq. 1) & and the slope factor (left–hand side of eq. 1). The default bounds are controlled by the variable crt_flg

- crt_flg=0 The bound is set to $\max\left[2, \frac{1}{2}\max(\text{initial value of reduced slope})\right]$. The value 2 is designed to keep extrapolations from being necessary in all but the top and bottom levels. The half of initial value is a device to prevent accidental over conditioning[6]. Stored in the scalar variable bound.
- crt_flg=1 The bound is set to 0.06. This bound is purely heuristic. Past experience has shown it to be useful. Stored in the scalar variable bound
- crt_flg=2 Both bounds are used. Stored in the vector variable bound, bound(1) is the bound for the reduced slope & bound(2) is the bound for the slope.

At this point a little inspection is necessary. First, consider the bound for the slope factor and the current maximum slope factor. Is the bound already satisfied? If it is then either the bound is too big or the topography doesn't need conditioning. Is the bound much smaller than the current maximum value? If it is, then maybe the bound is too small.

Next, use the script plot_curr_fac to view the slope factor and the topographic bounds on conditioning. Will the bounds prevent the conditioning routine from smoothing the areas in violation? Is there enough space between bounding contours for the areas in violation to be smoothed? Are any important topographic features in danger of being eliminated? The answers to these questions often dictate changes in HSHLLW, HDEEP and/or ADDCNST.[7]

Conditioning (cond_topo)

Once the parameters have been determined to be acceptable, the conditioning can proceed. Simply type the command cond_topo. This script needs no more input, everything has been set–up by cond_init. When this script is finished, it creates a file called

[6] Author's note: In one domain (in the Ionian Sea) I was trying to apply a bound of 2 for the reduced slope when the maximum starting value was 50. I eventually had to increase the bound to 30. Too much topographic information was being lost.

[7] Author's note: When I'm first attempting to condition an area I usually spend several iterations just between cond_init and the input file, tuning these parameters. Any time I actually use ADDCNST, it's been after playing with various zooms of the plot of the current slope factor with additional contour lines to highlight important features.

`cond_topo.mat`. This file save the necessary working space for the writing step. It's basically a precaution for cases where there could be a long delay between the end of the conditioning and the writing step.

Once the conditioning is complete examine the results. Does the final topography preserve enough of the features from the original? Has the conditioning script introduced false features? If either too many features were lost or erroneous features were added, then don't save the results of the search and start over from scratch. Is the final topography smooth enough? If not, then the results can be saved but the conditioner may have to be run again. Figure 3 shows examples of the minimum set of comparisons that should be made. The provided plotting scripts were used to generate these figures.

Saving the Results (`cond_write`)

Once the changes have been determined to be acceptable, the GRIDS file needs to be updated. This is simply accomplished by using the script `cond_write`.[8] Again, this script needs no additional input. Everything has been set–up in `cond_init` and `cond_topo`.

After `cond_write` is finished, the `MATLAB` session can be terminated. The conditioning is done.

Nesting considerations

Conditioning in nested configurations is usually approached as an iterative problem. The goal is to gradually condition the entire set of nested domains so that the larger domains provide good driving boundary conditions during the conditioning of the smaller domains without introducing shocks. As an example of this, consider a triply nested set–up we did for the western North Atlantic. In that exercise we had 3 domains, a Northwest Atlantic domain (NWA) (largest), a Gulf of Maine domain (GOM) and a Massachusetts Bay domain (MB) (smallest). In that region we had the additional constraint the the 100m isobath should have minimal distortions. The conditioning procedure there was:

(1) Use the program **extract_grids** to update the interior topography values of GOM from MB. Similarly, update NWA from the newly updated GOM.
(2) Use **cond_topo** to condition everything below 100m. First condition the NWA domain. Then condition the GOM domain with boundary conditions from NWA. Then condition MB with boundary conditions from GOM.
(3) Repeat (1).
(4) Use **cond_topo** to condition everything above 100m. First condition the NWA domain. Then condition the GOM domain with boundary conditions from NWA. Then condition MB with boundary conditions from GOM.
(5) Repeat (1).
(6) Use **cond_topo** to condition everything. First condition the NWA domain. Then condition the GOM domain with boundary conditions from NWA. Then condition MB with boundary conditions from GOM.
(7) Repeat (1).

[8] If the `MATLAB` session was interrupted after `cond_topo` was finished but before `cond_write` was run, the necessary data will be in the file `cond_topo.mat`. Simply type `load cond_topo` before running `cond_write`.

Acknowledgments

We thank Reiner Onken and Wolfgang Erasmi for their work in creating standard plotting routines.