Manual of the matlab scripts of `LP Bathymetry`

Mathieu Dutout Sikirić

April 26, 2011

When one uses the ROMS model, one needs to smooth the bathymetry in order to get realistic results. Two roughness factors are involved: the rx_0 factor of Beckman and Haidvogel:

$$rx_0 = \max_{e \equiv e'} \frac{|h(e) - h(e')|}{h(e) + h(e')}$$

which should not go above 0.2 and the rx_1 factor of Haney which should not be above 6 [1]. (both rx_0 and rx_1 are shown up at the beginning of a ROMS run).

The original physical bathymetry as computed by interpolation and sampling is often too rough for the models and a smoothing operation is needed. The programs exposed here try given a roughness factor to find the bathymetry that is nearest to the real one. More details are given in [2].

The factor that matters is actually the rx_1 number which is required to be small. The problem is that it is quite difficult to optimize with respect to rx_1. The idea is to assume that there is a multiplicating factor between rx_0 and rx_1, i.e. $rx_1 = Crx_0$ and to optimize rx_0 instead of rx_1. This works quite well for `Vtransform=1` but not for the other transformations that were introduced later. Then a possible solution is to optimize with respect to a varying factor rx_0. The appropriate functions are provided.

1 Availability

The source of the program is available from http://www.liga.ens.fr/~dutour/Bathymetry/index.html

The linear programs are solved by the program `lpsolve` (see [6] for the installation). Note that we do not use the mex facility but the standalone program. The scripts are `matlab`© scripts and so you need to have `matlab`© installed.

2 How to use it

First of all, you need your bathymetry in the form of an array of the form `Hobs(eta_rho, xi_rho)` and a mask `MSK(eta_rho, xi_rho)`.

2.1 Using GRID_LinProgHeuristic

The command to do the filtering is then

```
>> Hfielt=GRID_LinProgHeuristic(MSK, Hobs, rx0max);
```

with

1. **MSK(eta_rho,xi_rho)** the mask.
2. **Hobs(eta_rho,xi_rho)** the bathymetry.
3. **rx0max** the chosen maximal rx_0 factor.

The program uses a divide and conqueer strategy for reducing the time of the run, that is it uses as subroutine **GRID_LinearProgrammingSmoothing_rx0_simple**, which may be used separately if desired. If some additional constraint are needed, have a look at **GRID_LinearProgrammingSmoothing_rx0**.

2.2 Using GRID_LinearProgrammingSmoothing_rx0_volume

If you want to preserve the total volume, then a variation of the above is:

```
>> Hfilt=GRID_LinearProgrammingSmoothing_rx0_volume(MSK, Hobs, rx0max, AreaMatrix);
```

with

1. **MSK(eta_rho,xi_rho)** the mask of the grid
2. **Hobs(eta_rho,xi_rho)** the observed bathymetry of the grid
3. **AreaMatrix(eta_rho,xi_rho)** the areas of the wet and land ρ-points of the grid.
4. **rx0max, rx1max** are roughness factors.

2.3 Using GRID_LinProgSmoothVertVert_rx0

Sometimes, you want to smooth the bathymetry but preserve the total volume. Here the method is significantly different: We increase the bathymetry at one cell e by say, $\delta_{e,e'}$ and decrease it at an adjacent cell e' by $\delta_{e,e'}$. We minimize the quantity

$$\sum_{e \equiv e'} |\delta_{e,e'}|$$

This method obviously preserve the volume and tend to preserve the volume of structures like basin and seamounts.

This method is used in the following way.

```
>> Hfilt=GRID_LinProgSmoothVertVert_rx0(MSK, Hobs, r);
```

with r the roughness factor you want to achieve. The problem of this method is its high computational cost since the number variable is higher.

2.4 Using GRID_LinProgHeuristic_rx0_fixed

This command corrects the bathymetry (if possible) and leaves the bathymetry of a set of points invariant

```
>> Hfilt=GRID_LinProgHeuristic_rx0_fixed(MSK, Hobs, PRS, r);
Hfilt=
```

with

1. **MSK(eta_rho,xi_rho)** the mask of the grid.
2. **Hobs(eta_rho,xi_rho)** the original bathymetry of the grid.
3. **PRS(eta_rho,xi_rho)** the list of grid point for which we want to preserve the bathymetry (PRS(iEta, iXi) == 1 if we want to preserve it).
4. **rx0max** is the maximum rx_0 factor.

The program uses a divide and conqueer strategy for reducing the time of the run, that is it uses as subroutine **GRID_LinearProgrammingSmoothing_rx0_fixed**, which may be used separately if desired.

2.5 Using GRID_LinearProgrammingSmoothing_rx0_blockconstraint

This command corrects the bathymetry (if possible) and returns a bathymetry satisfying a number of block condition:

```
>> Hfilt=GRID_LinearProgrammingSmoothing_rx0_blockconstraint(...
     MSK, Hobs, r, ListVal, ListBlock);
```

with

1. **MSK(eta_rho, xi_rho)** the mask of the grid.
2. **Hobs(eta_rho, xi_rho)** the original bathymetry of the grid.
3. **ListVal(nbBlock,1)** the list of values of constraints.
4. **ListBlock(nbBlock,eta_rho,xi_rho)** the list of arrays of constraints. We should have for all $1 \leq i \leq nbBlock$ the constraints

$$\sum_{iEta,iXi} ListBlock(iEta, iXi)(h(iEta, iXi) - h^{obs}(iEta, iXi)) \leq ListVal(i, 1)$$

2.6 Using GRID_SmoothPositive_*

This command makes the bathymetry correct by increasing it.

```
>> Hfilt=GRID_SmoothPositive_rx0(MSK, Hobs, rx0max);
>> Hfilt=GRID_SmoothPositive_ROMS_rx1(...
     MSK, Hobs, rx1max, ARVD);
```

with

1. **MSK(eta_rho,xi_rho)** the mask of the grid
2. **Hobs(eta_rho,xi_rho)** the observed bathymetry of the grid
3. **rx0max, rx1max** are roughness factors.
4. **ARVD** is the record of vertical parameterization the S-coordinates parameters.

```
ARVD.Vtransform=2;
ARVD.Vstretching=1;
ARVD.ThetaS=4; % named THETA_S in the roms.in file
ARVD.ThetaB=0.35; % named THETA_B in the roms.in file
ARVD.hc=10; % named TCLINE in the roms.in file
ARVD.N=30;
```

2.7 Using GRID_PlusMinusScheme_rx0

This command makes the bathymetry correct by doing a sequence of increase/decrease at adjacent cells (see [4]).

```
>> [RetBathy, HmodifVal]=GRID_PlusMinusScheme_rx0(...
       MSK, Hobs, rx0max, AreaMatrix);
```

with

1. **MSK(eta_rho,xi_rho)** the mask of the grid
2. **Hobs(eta_rho,xi_rho)** the observed bathymetry of the grid
3. **AreaMatrix(eta_rho,xi_rho)** the areas of the wet and dry ρ-points.
4. **rx0max, rx1max** are roughness factors.

2.8 Using GRID_LaplacianSelectSmooth_rx0

This command makes the bathymetry correct by doing an iterated sequence of laplacian filterings

```
>> Hfilt=GRID_LaplacianSelectSmooth_rx0(MSK, Hobs, rx0max);
```

with

1. **MSK(eta_rho,xi_rho)** the mask of the grid
2. **Hobs(eta_rho,xi_rho)** the observed bathymetry of the grid
3. **rx0max** the maximal roughness factor.

2.9 Using heuristic functions

This command makes the bathymetry correct by first applying linear programming and then doing bathymetry increases.

```
>> Hfilt=GRID_LinearProgrammingSmoothing_rx1_heuristic(...
     MSK, Hobs, rx1max, ARVD);
```

3 Notes and Recommendations

- The smoothing with respect to rx_0 is best done with **GRID_LinProgHeuristic** which uses a linear programming approach and should be fast even in very large and not pathological grids.
- The smoothing with respect to rx_1 is problematic since the number of constraint is much larger. Also for `Vtransform=2` those constraints are nonlinear. A variant of the Martinho Batteen [5] is implemented in `GRID_SmoothPositive_ROMS_rx1` and deals with all the vertical parametrization available in ROMS.
- The function `GRID_LaplacianSelectSmooth` has several advantages over the function `smth_bath.m` of the ROMS matlab package:
  - It respects the mask
  - It is guaranteed to terminate
  - It creates a perturbation to the bathymetry of smaller amplitude.

  Still our recommendation is not to use Laplacian/Shapiro filtering as they produce worse solution than other methods and have a very tenuous justification as an adequate method.
- If preserving the volume is important, you can use the function `GRID_PlusMinusScheme_rx0`. It will always produce a larger perturbation than `GRID_LinearProgrammingSmoothing_rx0_volum` or `GRID_LinProgSmoothVertVert` but it is much faster.

References

[1] R.L. Haney, *On the pressure gradient force over steep bathymetry in sigma coordinates ocean models*, Journal of Physical Oceanography **21** (1991) 610–619.

[2] M. Dutour Sikiric, I. Janekovic, M. Kuzmic, *A new approach to bathymetry smoothing in sigma-coordinate ocean models*, Ocean Modelling, Volume 29, Issue 2, 2009, Pages 128-136

[3] V. Chvátal, *Linear Programming*, W.H. Freeman and Company, 1983.

[4] G.L. Mellor, T. Ezer and L.-Y. Oey, *The pressure gradient conundrum of Sigma coordinate Ocean models*, Journal of atmospheric and oceanic technology **11** (1994) 1126–1134.

[5] A.S. Martinho and M.L. Batteen, *On reducing the slope parameter in terrain following numerical ocean models*, Ocean Modelling **13** (2006) 166–175.

[6] P. Notebaert and K. Eikland, http://lpsolve.sourceforge.net/5.5/