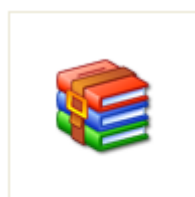


MITgcm 菜鸟攻略

by 张正光

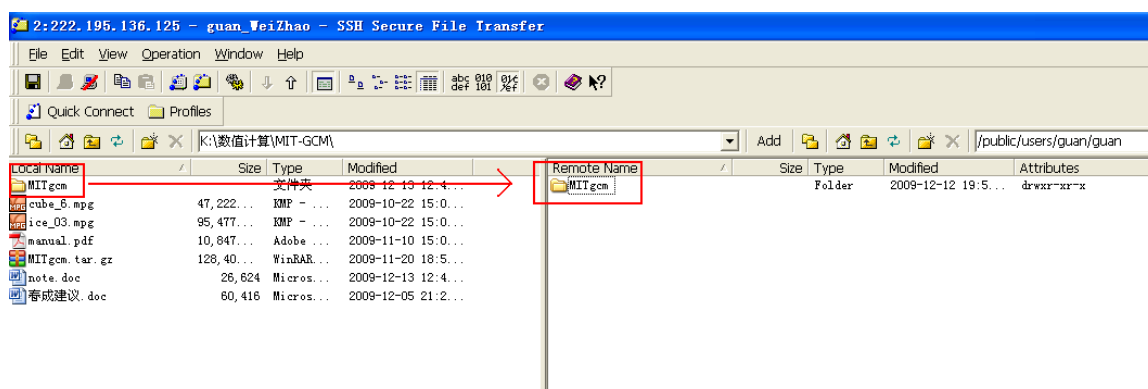
让我们开始吧！

这是我们上手的第一个例子，看来我们要先安装一下，文件夹中我应该给大家准备了一个压缩文件：

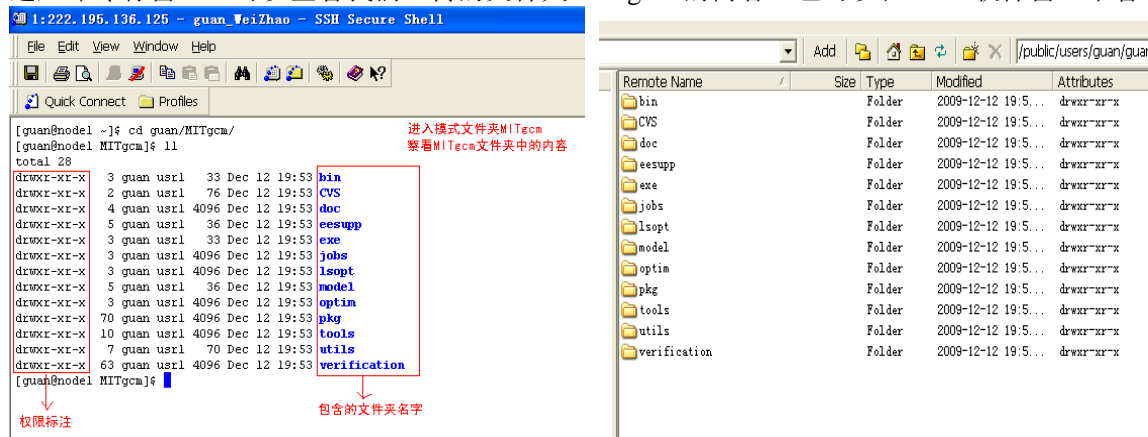


MITgcm.tar.gz

嘿嘿，大家把它解压缩后会产生一个名字为 MITgcm 的文件夹，我们需要的模式文件就在这个文件夹当中，我们把这个文件夹上传到机群上：

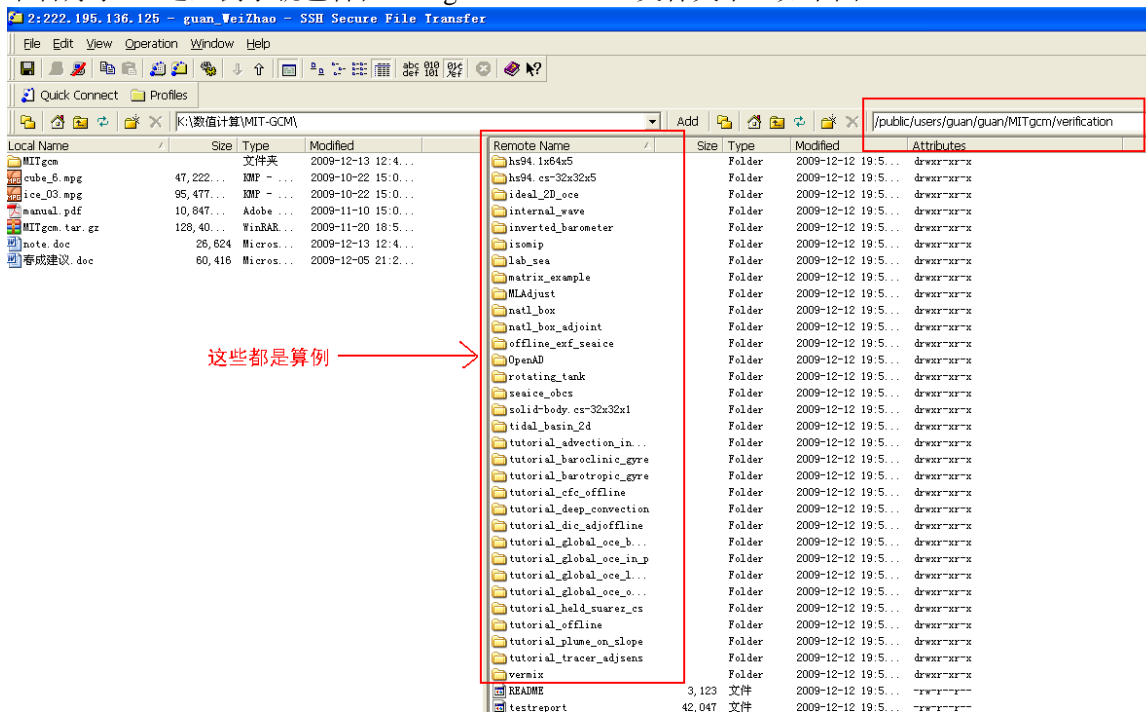


进入命令行窗口，可以查看我们上传的文件夹 MITgcm 的内容，也可以在 SSH 软件窗口中看：

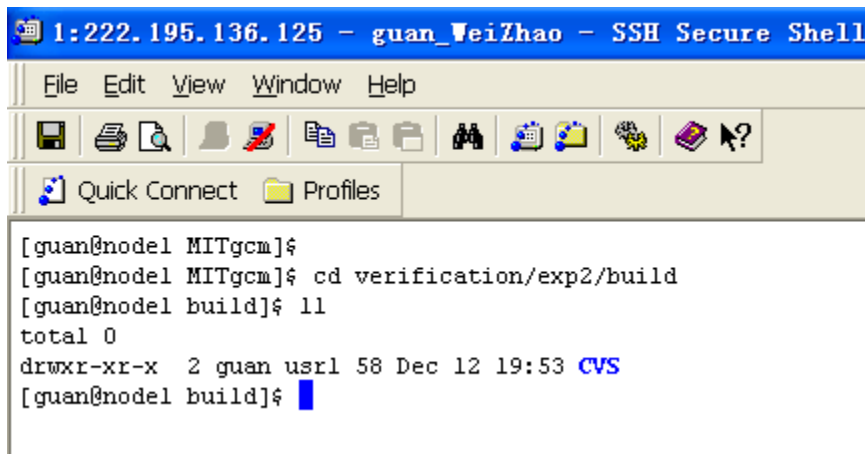


对这些文件夹包含内容的介绍参照 manual 中的 3.3 节

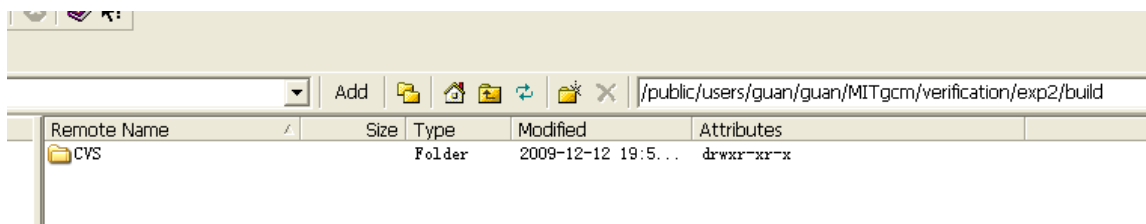
对我们比较有利的事情是，MITgcm 自己包含了许多例子（当然里面的各种条件人家都给准备好了），这些例子就包含在 MITgcm/verification/文件夹下，如下图：



我们一上来当然是想找软柿子捏啦，为了与 manual 保持一致，我们将首先拿算例 exp2 开刀：



用命令 cd 进入到 MITgcm/verification/exp2/build/当中，可以看到现在 build 文件夹中还空空如也（Cv 文件夹是模式用于自我版本更新用的，不参与计算），用 ssh 软件看的结果一样。



现在进入正题，第一步，我们需要在这个文件夹中创建一个叫Makefile的文件，而生成这个文件的工具叫genmake2，genmake2是一个Linux的shell文件（其实就是一个批处理文件），这个文件存放在 MITgcm/tools/ 文件夹中，我们现在进入这个文件夹：

```
1:222.195.136.125 - guan_WeiZhao - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

[guan@model build]$
[guan@model build]$ cd ../../tools
[guan@model tools]$ ll
total 216
drwxr-xr-x 3 guan usr1 4096 Dec 12 19:53 adjoint_options
-rw-r--r-- 1 guan usr1 239 Dec 12 19:53 adjoint_sed
-rw-r--r-- 1 guan usr1 2053 Dec 12 19:53 ad_taf_output.f.adtrick.diff
-rw-r--r-- 1 guan usr1 2360 Dec 12 19:53 ad_taf_output.f.diva.diff_dell
-rw-r--r-- 1 guan usr1 2322 Dec 12 19:53 ad_taf_output.f.diva.diff_full
-rw-r--r-- 1 guan usr1 657 Dec 12 19:53 ad_taf_output.f.onlyfwd.diff
-rw-r--r-- 1 guan usr1 376 Dec 12 19:53 as
drwxr-xr-x 4 guan usr1 8192 Dec 12 19:53 build_options
-rw-r--r-- 1 guan usr1 3848 Dec 12 19:53 calc_diagnostics_dims
-rw-r--r-- 1 guan usr1 961 Dec 12 19:53 convert_cpp_cmd2defines
drwxr-xr-x 2 guan usr1 76 Dec 12 19:53 CVS
drwxr-xr-x 3 guan usr1 4096 Dec 12 19:53 cyrus-inapd-nakedepend
-rw-r--r-- 1 guan usr1 6938 Dec 12 19:53 do_tst_2+2
drwxr-xr-x 3 guan usr1 54 Dec 12 19:53 embed_encode
drwxr-xr-x 7 guan usr1 82 Dec 12 19:53 example_scripts
-rw-r--r-- 1 guan usr1 1174 Dec 12 19:53 f90mkdepend
-rw-r--r-- 1 guan usr1 91525 Dec 12 19:53 genmake2
-rw-r--r-- 1 guan usr1 297 Dec 12 19:53 guess_mpi_include_dir.sh
-rw-r--r-- 1 guan usr1 335 Dec 12 19:53 icard
drwxr-xr-x 3 guan usr1 4096 Dec 12 19:53 mpack-1.6
-rw-r--r-- 1 guan usr1 667 Dec 12 19:53 nc.sh
drwxr-xr-x 3 guan usr1 66 Dec 12 19:53 netcdf_notes
-rw-r--r-- 1 guan usr1 529 Dec 12 19:53 remove_comments_sed
-rw-r--r-- 1 guan usr1 160 Dec 12 19:53 set64bitConst.csh
-rw-r--r-- 1 guan usr1 145 Dec 12 19:53 set64bitConst.sh
-rw-r--r-- 1 guan usr1 7068 Dec 12 19:53 suggest_optfile_names
-rw-r--r-- 1 guan usr1 12890 Dec 12 19:53 tst_2+2
-rw-r--r-- 1 guan usr1 4319 Dec 12 19:53 xmkdepend
[guan@model tools]$
```

我们需要执行使用的genmake2没有可执行的权限，这是个大问题

注意这些权限，有很多都没有可执行权限

再之后的操作中我们将经常调用tools中的文件，但都需要他们是可执行的，但现在情况好像不太行，所以我们做了一个大胆的决定，把tools中所有的文件都给予可执行权限！

```
1:222.195.136.125 - guan_WeiZhao - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

[guan@model tools]$
[guan@model tools]$ chmod a+x *
[guan@model tools]$ ll
total 216
drwxr-xr-x 3 guan usr1 4096 Dec 12 19:53 adjoint_options
-rw-r-xr-x 1 guan usr1 239 Dec 12 19:53 adjoint_sed
-rw-r-xr-x 1 guan usr1 2053 Dec 12 19:53 ad_taf_output.f.adtrick.diff
-rw-r-xr-x 1 guan usr1 2360 Dec 12 19:53 ad_taf_output.f.diva.diff_dell
-rw-r-xr-x 1 guan usr1 2322 Dec 12 19:53 ad_taf_output.f.diva.diff_full
-rw-r-xr-x 1 guan usr1 657 Dec 12 19:53 ad_taf_output.f.onlyfwd.diff
-rw-r-xr-x 1 guan usr1 376 Dec 12 19:53 as
drwxr-xr-x 4 guan usr1 8192 Dec 12 19:53 build_options
-rw-r-xr-x 1 guan usr1 3848 Dec 12 19:53 calc_diagnostics_dims
-rw-r-xr-x 1 guan usr1 961 Dec 12 19:53 convert_cpp_cmd2defines
drwxr-xr-x 2 guan usr1 76 Dec 12 19:53 CVS
drwxr-xr-x 3 guan usr1 4096 Dec 12 19:53 cyrus-inapd-nakedepend
-rw-r-xr-x 1 guan usr1 6938 Dec 12 19:53 do_tst_2+2
drwxr-xr-x 3 guan usr1 54 Dec 12 19:53 embed_encode
drwxr-xr-x 7 guan usr1 82 Dec 12 19:53 example_scripts
-rw-r-xr-x 1 guan usr1 1174 Dec 12 19:53 f90mkdepend
-rw-r-xr-x 1 guan usr1 91525 Dec 12 19:53 genmake2
-rw-r-xr-x 1 guan usr1 297 Dec 12 19:53 guess_mpi_include_dir.sh
-rw-r-xr-x 1 guan usr1 335 Dec 12 19:53 icard
drwxr-xr-x 3 guan usr1 4096 Dec 12 19:53 mpack-1.6
-rw-r-xr-x 1 guan usr1 667 Dec 12 19:53 nc.sh
drwxr-xr-x 3 guan usr1 66 Dec 12 19:53 netcdf_notes
-rw-r-xr-x 1 guan usr1 529 Dec 12 19:53 remove_comments_sed
-rw-r-xr-x 1 guan usr1 160 Dec 12 19:53 set64bitConst.csh
-rw-r-xr-x 1 guan usr1 145 Dec 12 19:53 set64bitConst.sh
-rw-r-xr-x 1 guan usr1 7068 Dec 12 19:53 suggest_optfile_names
-rw-r-xr-x 1 guan usr1 12890 Dec 12 19:53 tst_2+2
-rw-r-xr-x 1 guan usr1 4319 Dec 12 19:53 xmkdepend
[guan@model tools]$
```

用这个语句就可以给文件夹中所有文件以可执行权限，其中的a代表对所有用户，*代表多有文件

我们可以看到多有文件都有了可执行权限，它们的名字也都变绿了

好了，我们回到 MITgcm/verification/exp2/build/文件夹，在其中用下述命令生成 Makefile 文件：

```
1:222.195.136.125 - guan_WeiZhao - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

[guan@node1 build]$ clear
[guan@node1 build]$ ../../../../tools/genmake2 -mods=../code

GENMAKE :

A program for GENERating MAKEfiles for the MITgcm project.  For a
quick list of options, use "genmake -h" or for more detail see:
```

然后会扑拉扑拉的出现好多信息提示，其中我们要注意一下的是：

```
=== Processing options files and arguments ===
getting local config information: none found
Warning: ROOTDIR was not specified but there appears to be a copy of MITgcm at "../../.." so we'll try it.
getting OPTFILE information:
Warning: no OPTFILE specified so we'll look for possible settings

=== Searching for possible settings for OPTFILE ===
The platform appears to be: linux_amd64
The possible C compilers found in your path are:
gcc c89 cc c99 mpicc icc
Setting C compiler to: gcc
The possible FORTRAN compilers found in your path are:
g77 f77 pgf77 pgf95 ifort mpif77 gfortran
Setting OPTFILE to: ../../../../tools/build_options/linux_amd64_g77
using OPTFILE="../../../../tools/build_options/linux_amd64_g77"
getting AD_OPTFILE information:
using AD_OPTFILE="../../../../tools/adjoint_options/adjoint_default"
```

— 检测出系统是64位的 Linux
— 系统已经安装的C语言的编译器有这些
— 系统已经按章的fortran编译器有这些
— MITgcm默认的选择了g77为使用的编译器

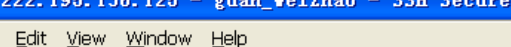
最后出来 “=== Done ===” 就可以了，我们用命令“ll”进行察看，可以发现文件夹中现在多了几个文件，其中就有我们需要的 Makefile：

```
Making list of "exceptions" that need ".p" files
Making list of NOOPTFILES
Add rules for links
Adding makedepend marker

=== Done ===
[guan@node1 build]$ ll
total 88
-rw-r--r-- 1 guan usrl 470 Dec 13 13:23 AD_CONFIG.h
drwxr-xr-x 2 guan usrl 58 Dec 12 19:53 CVS
-rw-r--r-- 1 guan usrl 3145 Dec 13 13:23 genmake_ad_optfile
-rw-r--r-- 1 guan usrl 2176 Dec 13 13:23 genmake_optfile
-rw-r--r-- 1 guan usrl 13436 Dec 13 13:23 genmake_state
-rw-r--r-- 1 guan usrl 3705 Dec 13 13:23 genmake_warnings
-rw-r--r-- 1 guan usrl 2048 Dec 13 13:23 hello.o
-rw-r--r-- 1 guan usrl 43396 Dec 13 13:23 Makefile
-rw-r--r-- 1 guan usrl 212 Dec 13 13:23 Makefile.bak
-rw-r--r-- 1 guan usrl 3065 Dec 13 13:23 PACKAGES_CONFIG.h
[guan@node1 build]$
```

```
[guan@node1 build]$ make depend
```

```
F ini_p_ground.F ini_pnh.F ini_pressure.F ini_tegrate_for_w.F integr_continuity.F load_fields_init_fixed.F packages_init_variables.F pack
et_defaults.F set_grid_factors.F set_parms.F s
taueddy_init_varia.F the_main_loop.F the_mode
te_masks_etc.F update_r_star.F update_surf_dr.
../../../../tools/f90mkdepend >> Makefile
rm -f makedepend.out
[guan@model build]$
```

[illegible]

```
1:222.195.136.125 - guan_VeiZhao - SSH Secure Shell
File Edit View Window Help
[guan@model build]$ export FC=ifort
[guan@model build]$ export CC=icc
[guan@model build]$
```

然后我们使用下面的命令生成可执行文件“mitgcmuv”:

```
1:222.195.136.125 - guan_WeiZhao - SSH Secure Shell
File Edit View Window Help
[guan@node1 build]$
[guan@node1 build]$ make
```

我们可以看到“mitgcmuv”已经被创建了:

/public/users/guan/guan/MITgcm/verification/exp2/build					
Remote Name	Size	Type	Modified	Attributes	
memsync.f	11,488	Fortra...	2009-12-13 14:0...	-rw-r--r--	
memsync.o	888	0 文件	2009-12-13 14:0...	-rw-r--r--	
mitgcmuv	2,112,357	文件	2009-12-13 14:0...	-rwxr-xr-x	
modeldata_example.F	24,829	Symbol...	2009-12-12 19:5...	lrw-r--r--	
modeldata_example.f	97,637	Fortra...	2009-12-13 14:0...	-rw-r--r--	

现在可以运行模式了,但是我们需要把模式需要输入的数据文件关联过来,以便模式调用,使用下述语句:

```
[guan@node1 build]$ ln -s ../input/* .
ln: `./CVS': cannot overwrite directory
ln: `./data': File exists
ln: `./data.pkg': File exists
ln: `./eedata': File exists
ln: `./eedata.mth': File exists
ln: `./salt.bin': File exists
ln: `./SSS.bin': File exists
ln: `./SST.bin': File exists
ln: `./theta.bin': File exists
ln: `./topog.bin': File exists
ln: `./windx.bin': File exists
ln: `./windy.bin': File exists
[guan@node1 build]$
```

然后就可以运转模式了,使用语句:

```
[guan@node1 build]$ ./mitgcmuv > output.txt
```

在 build 文件夹中也多了一个 output.txt 来存放运转信息:

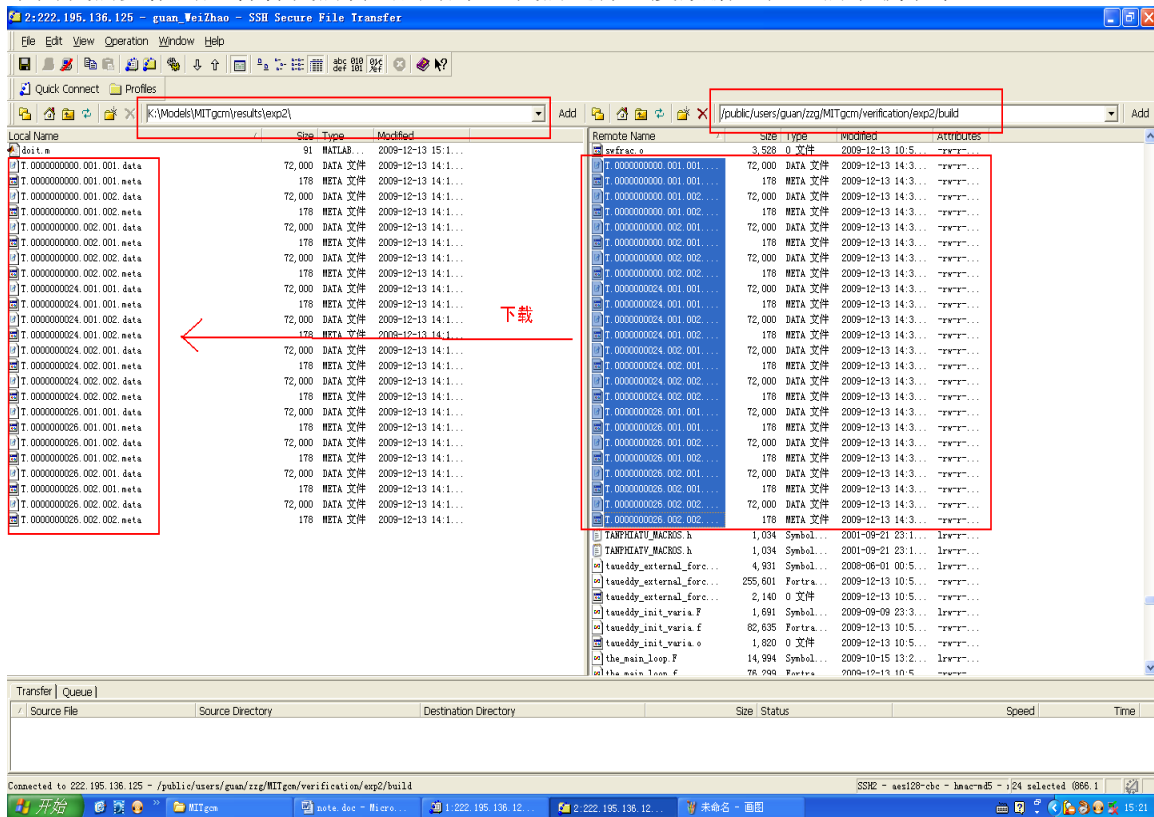
/public/users/guan/guan/MITgcm/verification/exp2/build					
Remote Name	Size	Type	Modified	Attributes	
nml_set_terminator.f	24,399	Fortra...	2009-12-13 14:0...	-rw-r--r--	
nml_set_terminator.o	1,440	0 文件	2009-12-13 14:0...	-rw-r--r--	
open_copy_data_file.F	4,410	Symbol...	2009-12-12 19:5...	lrw-r--r--	
open_copy_data_file.f	25,867	Fortra...	2009-12-13 14:0...	-rw-r--r--	
open_copy_data_file.o	12,652	0 文件	2009-12-13 14:0...	-rw-r--r--	
output.txt	266,812	文本文档	2009-12-13 14:1...	-rw-r--r--	
packages_boot.F	4,951	Symbol...	2009-12-12 19:5...	lrw-r--r--	
packages_boot.f	77,652	Fortra...	2009-12-13 14:0...	-rw-r--r--	

这是 output.txt 中的内容:

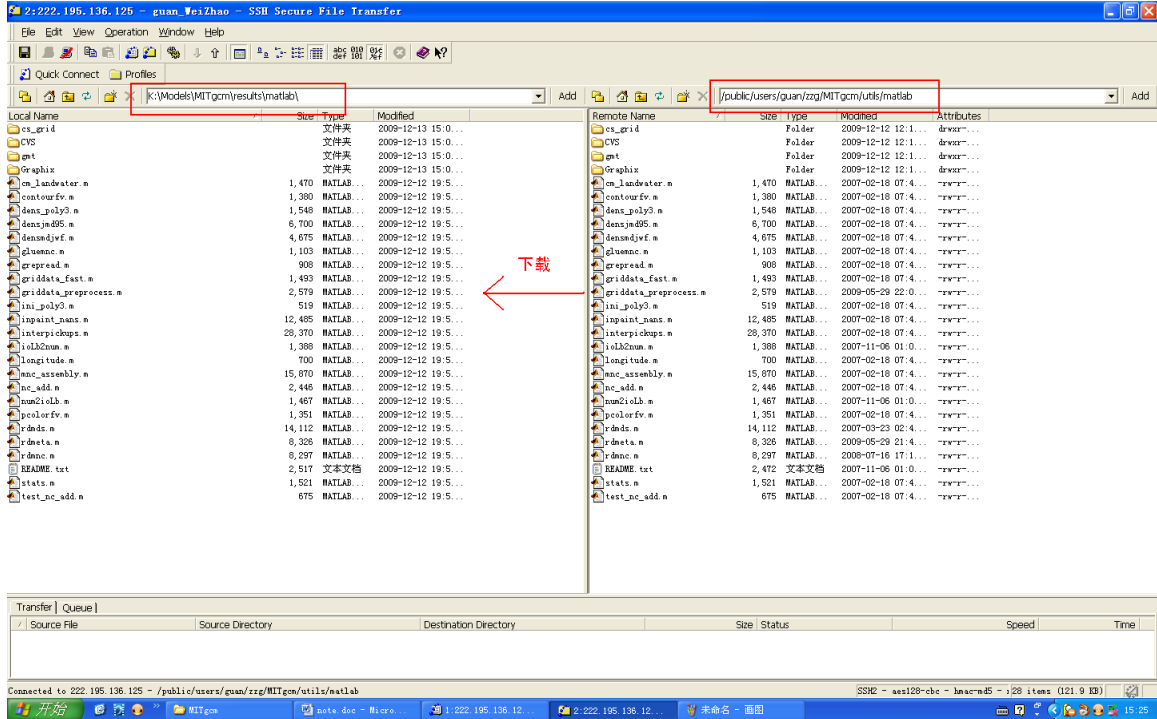
```
output (3).txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

(PID.TID 0000.0001)
(PID.TID 0000.0001) // =====
(PID.TID 0000.0001) // MITgcm UV
(PID.TID 0000.0001) // =====
(PID.TID 0000.0001) // execution environment starting up...
(PID.TID 0000.0001)
(PID.TID 0000.0001) // MITgcmUV version: checkpoint61x
(PID.TID 0000.0001) // Build user:      guan
(PID.TID 0000.0001) // Build host:      node1
(PID.TID 0000.0001) // Build date:      Sun Dec 13 13:58:03 CST 2009
(PID.TID 0000.0001)
(PID.TID 0000.0001) // =====
(PID.TID 0000.0001) // Execution Environment parameter file "eedata"
(PID.TID 0000.0001) // =====
(PID.TID 0000.0001) ># Example "eedata" file
(PID.TID 0000.0001) ># Lines beginning "#" are comments
(PID.TID 0000.0001) ># nTx - No. threads per process in X
(PID.TID 0000.0001) ># nTy - No. threads per process in Y
(PID.TID 0000.0001) > &EEPARMS
(PID.TID 0000.0001) > nTx=1,
(PID.TID 0000.0001) > nTy=1,
(PID.TID 0000.0001) > &
(PID.TID 0000.0001) ># Note: Some systems use & as the
(PID.TID 0000.0001) ># namelist terminator. Other systems
(PID.TID 0000.0001) ># use a / character (as shown here).
```

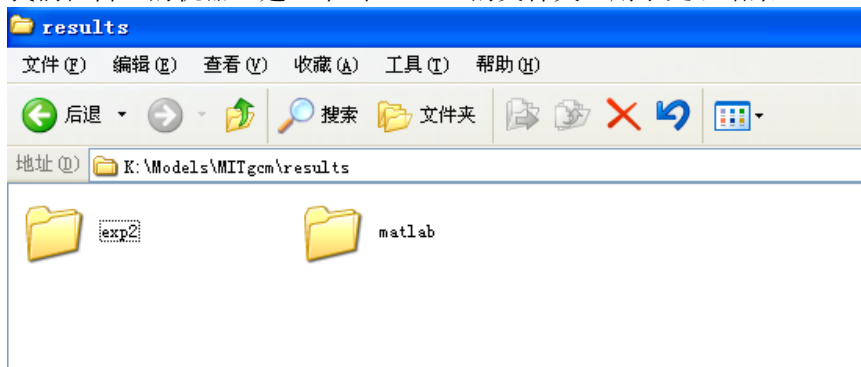
下面我们要做的是看看我们转出的结果，我们选择温度数据，把它们下载下来:



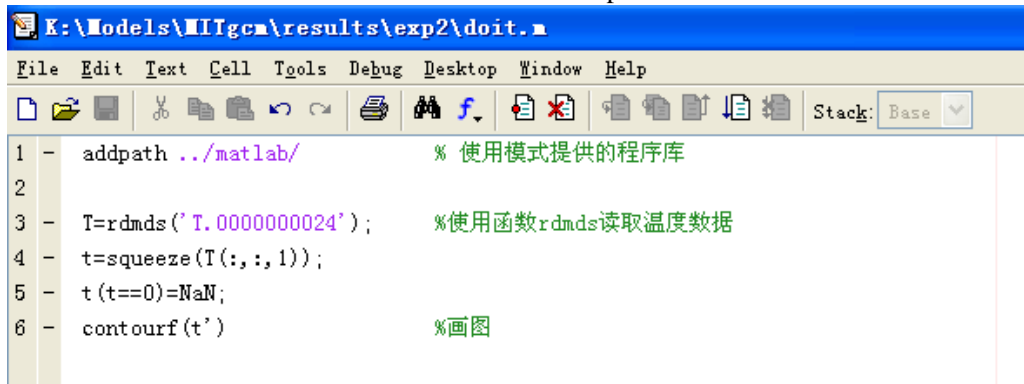
同时，如果我们想读取数据文件，我们需要模式自带的 matlab 程序，故我们将模式自带的程序下载下来：



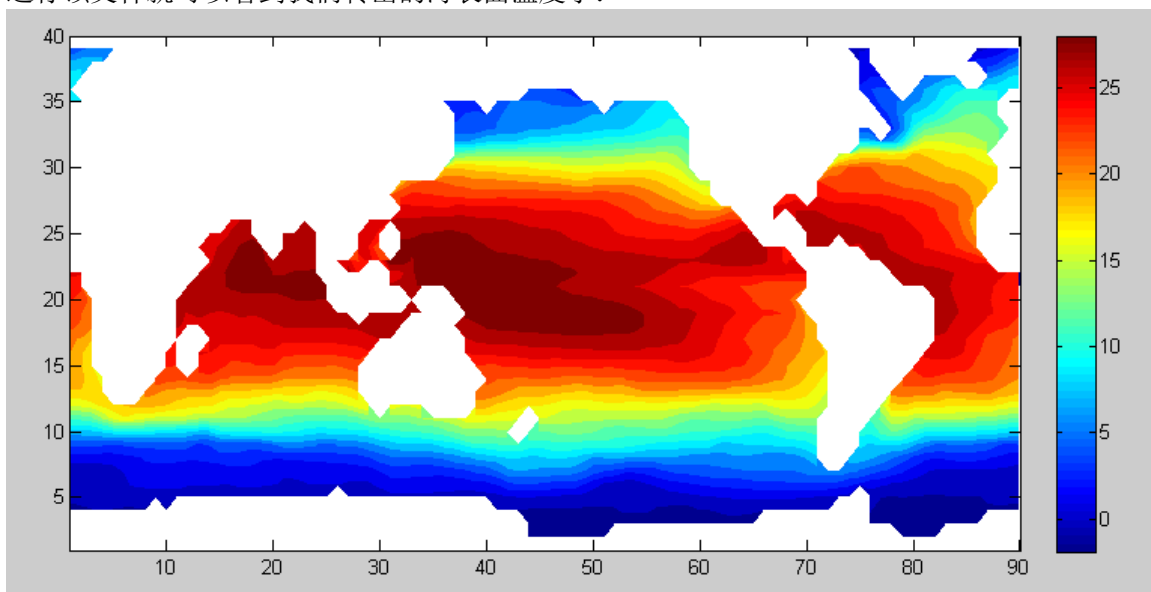
我们在自己的机器上建一个叫”results”的文件夹，用于处理结果：



现在其中有连个文件夹，exp2 存放我们刚刚下载的 exp2 运转结果中的温度数据，matlab 存放我们刚刚下载的模式自带的 matlab 程序库。进入 exp2，建立一个叫 doit.m 的 m 文件：



运行该文件就可以看到我们转出的海表面温度了：



至此我们的首次测试运转成功了！

陆架上的重力流动算例小试牛刀

3.16 Gravity Plume On a Continental Slope

(in directory: *verification/tutorial_plume_on_slope/*)

呵呵，这个算例要想转起来很简单，只需按照 manual 上的指示做就行：

3.16.5 Build and run the model

Build the model per usual. For example:

```
% cd verification/plume_on_slope
% mkdir build
% cd build
% ../../tools/genmake -mods=../code -disable=gredi,kpp,zonal_filt
,shap_filt
% make depend
% make
```

When compilation is complete, run the model as usual, for example:

```
% cd ../
% mkdir run
% cp input/* build/mitgcmuv run/
% cd run
% ./mitgcmuv > output.txt
```

但是有一个地方我们需要注意，重力里流动的成长需要几天的时间，而模式默认的积分时间是很短的，下面是控制参数的 `input/data` 文件的原始形式

UltraEdit - [K:\Models\IMPACT\verification\tutorial_plume_on_slope\input_data] 文件的原始形式

文件(F) 编辑(E) 搜索(S) 插入(I) 项目(O) 视图(V) 格式(O) 列(L) 宏(M) 脚本(S) 高级(A) 窗口(W) 帮助(H)

data

打开 资源管理器 列

筛选: *

C: D: E: F: G: H: K: FTP 帐号

```

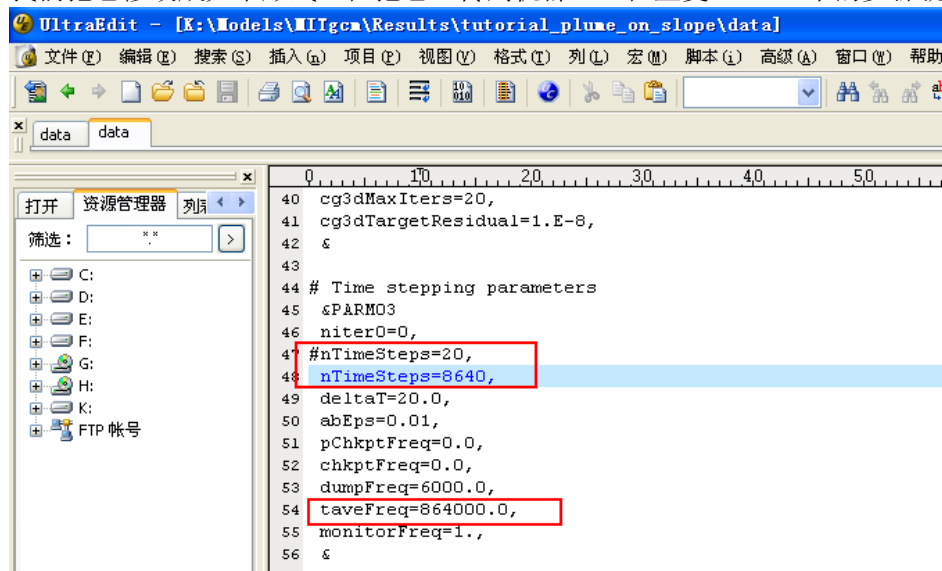
40 cg3dMaxIters=20,
41 cg3dTargetResidual=1.E-8,
42 &
43
44 # Time stepping parameters
45 cPARM03
46 niter0=0,
47 nTimeSteps=20,
48 !cTimeSteps=8640,
49 deltaT=20.0,
50 abEps=0.01,
51 pChkptFreq=0.0,
52 chkptFreq=0.0,
53 dumpFreq=6000.0,
54 taveFreq=864000.0,
55 monitorFreq=1.,
56 &
57
58 # Gridding parameters
59 cPARM04
60 usingCartesianGrid=.TRUE.,
61 usingSphericalPolarGrid=.FALSE.,
62 delXfile='dx.bin',
63 dYspacing=200.,
64 delZ=60*3.33333333333333333333333333333333,
65 &

```

这里只积分20次，显然不够

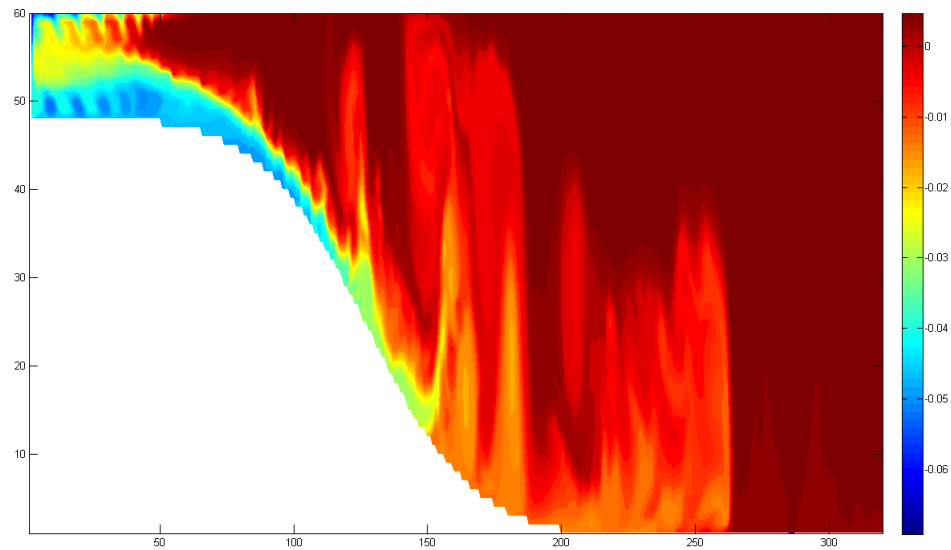
注意到下面一行积分86400次的被注释掉了，我们将启用它 (86400次积分相当于2天)

我们把它修改成如下形式，在把它上传到机群上，在重复 manual 中的步骤就 OK 啦：



```
40 cg3dMaxIters=20,  
41 cg3dTargetResidual=1.E-8,  
42 &  
43  
44 # Time stepping parameters  
45 &PARM03  
46 niter0=0,  
47 nTimeSteps=20,  
48 nTimeSteps=8640,  
49 deltaT=20.0,  
50 abEps=0.01,  
51 pChkptFreq=0.0,  
52 chkptFreq=0.0,  
53 dumpFreq=6000.0,  
54 taveFreq=864000.0,  
55 monitorFreq=1.,  
56 &
```

这样我们就能得到充分发展的重力下坡流动图像了，下图是温度场：

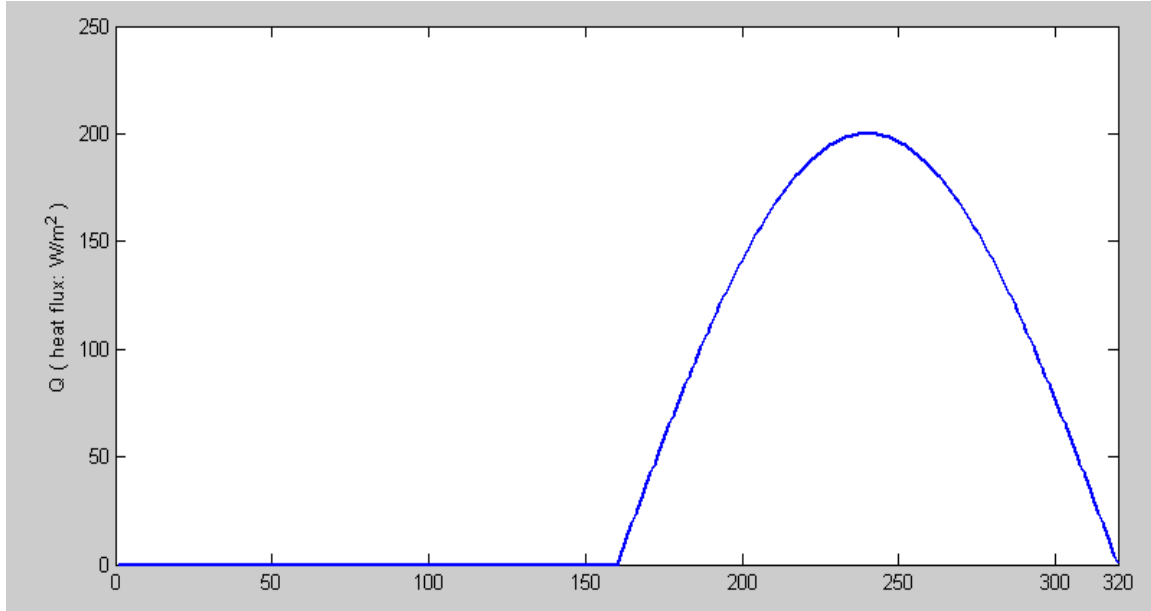


下雨一样的冷对流

在上一个算例的基础上我们将对模式作一些修改，弄一个比较有趣的算例，但是需要我们对模式的水平网格、地形、上边界的流量通量、时间步进作一些调整，首先我们需要调整的是生成网格和边界条件的文件：**gendata_zzg.m**，它在 [\Results\tutorial_plume_on_slope\test02\input](#) 文件夹中：

```
5 - prec='real*8'; % 这两个变量与输出的数据文件格式有关
6 - ieee='b';
7
8 % Dimensions of grid
9 - nx=320; % x方向的网格数
10 - ny=1; % y方向的网格数
11 - nz=60; % z方向的网格数
12 % Nominal depth of model (meters)
13 - H=200.0; % z方向上的区域厚度
14 % Size of domain
15 - Lx=6.40e3; % x方向上的区域长度
16
17 % Spacial resolution (m)
18
19 - for i=1:nx
20 -     dx(i) = Lx/nx; % x方向上的网格步长，我们这里采用的是均匀网格，也被不均匀
21 - end
22 - clear i
23 - fid=fopen('dx.bin','w',ieee); fwrite(fid,dx,prec); fclose(fid); % 这是输出水平网格的语句
24
25 - dy = Lx/nx; % y方向上的网格步长
26 - dz=H/nz; % z方向上的网格步长
27
28 - x=zeros(nx,1);
29 - x(1) = dx(1);
30 - for i=2:nx
31 -     x(i)=x(i-1) + dx(i); % x方向的坐标序列
32 - end
33 - clear i
34
35 - z=-dz/2:-dz:-H; % z方向的坐标序列
36
37 % Heat Flux
38 - Qo=200; % 热通量的参考值, 正值为cooling
39 - Q=0.0*rand([nx,ny]);
40 - for i=1:nx
41 -     Q(i,:) = Q(i,:) - Qo*sin( 2*pi*x(i)/Lx );
42 - end
43 - clear i
44 - Q(Q<0)=0; % 取消加热
45 - fid=fopen('Qnet.forcing','w',ieee); fwrite(fid,Q,prec); fclose(fid); % 将热通量数据写入文件
```

这里使用的热通量，如果大于 0，代表对水体的冷却，我们这里使用的冷却是一个半波长的正弦冷却：

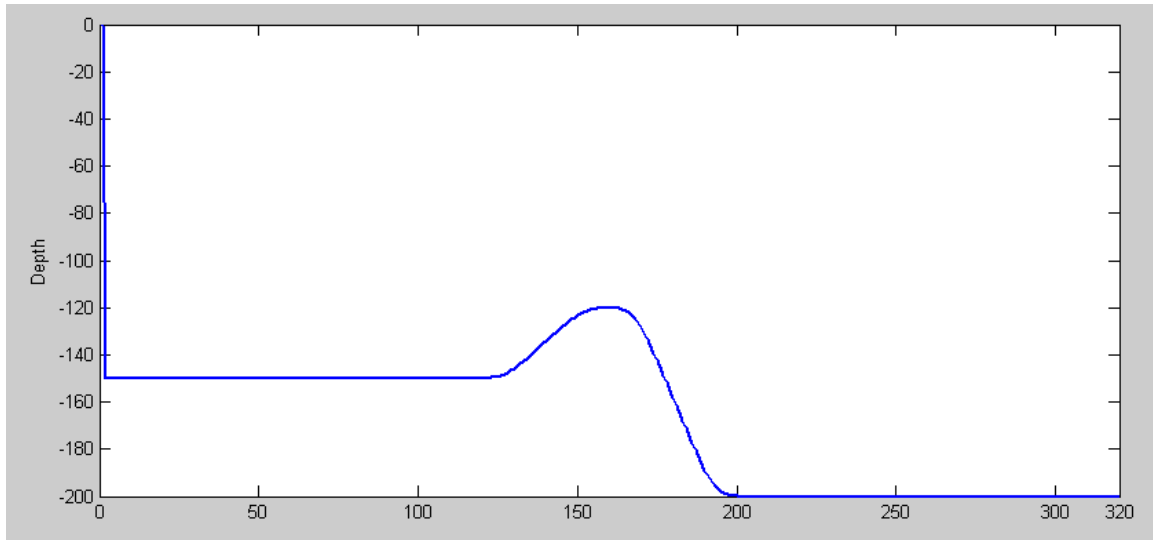


设置初始层结为随机温度场，有利于对流发展：

```
48 % Initial Stratification
49 - t=0.01*rand([nx,ny,nz]);
50 - fid=fopen('T.init','w',ieee); fwrite(fid,t,prec); fclose(fid);
```

最后是构造地形数据：

```
53 % Sloping channel
54 % tanh function for slope
55
56 - d=0.0*rand([nx,ny]);
57 - h=0.0*rand([nx,ny]);
58 - h(1:160)=50;
59 - h(140:180)=80;
60 - h=smooth(h,10);
61 - h=smooth(h,10);
62 - h=smooth(h,25);
63 for i=1:nx
64     for j=1:ny
65         d(i,j) = h(i,j) - H ;
66     end
67 end
68 clear i j
69 d(1,:)=0.0; %这个地方太麻烦了，看看data.obcs（边界设置文件），里面只把东边界设为开边界了，所以西边界要闭合
70 % d(320,:)=0.0;
71 - fid=fopen('topog.slope','w',ieee); fwrite(fid,d,prec); fclose(fid);
```



改完了 **gendata_zzg.m**, 运行之, 会生成四个数据文件:

gendata_zzg.m	3 KB	MATLAB M-file	2009-12-22 13:54
dx.bin	3 KB	BIN 文件	2009-12-22 14:01
Qnet.forcing	3 KB	FORCING 文件	2009-12-22 14:01
T.init	150 KB	INIT 文件	2009-12-22 14:01
topog.slope	3 KB	SLOPE 文件	2009-12-22 14:01

分别是: 水平网格、热量通量、初始温度场、底地形

之后我们需要的干的事情还有把 **input** 中的 **data** 文件动一动手脚:

```

44 # Time stepping parameters
45 &PARM03
46 niter0=0,
47 #nTimeSteps=600,
48 nTimeSteps=160000,
49 deltaT=2.0,
50 abEps=0.01,
51 pChkptFreq=0.0,
52 chkptFreq=0.0,
53 dumpFreq=200.0,
54 taveFreq=86400.0,
55 monitorFreq=1.,
56 # staggerTimeStep=.TRUE.,
57 &

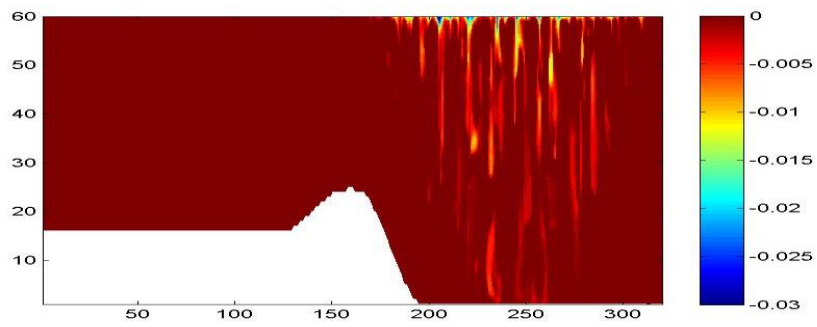
```

修改积分次数为160000次
修改时间步长为 2 秒

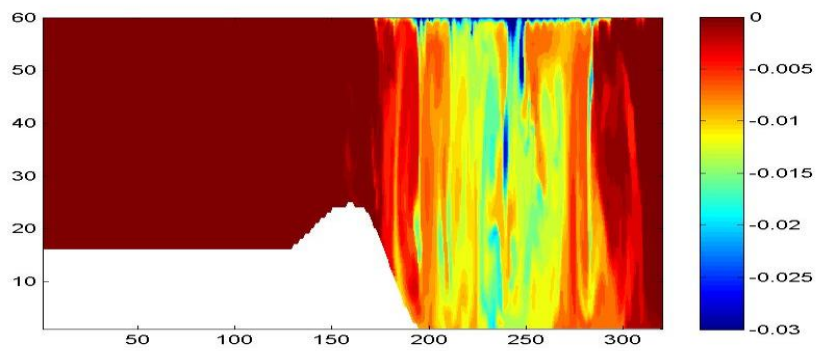
两百秒输出一次

然后把这些修改过的文件上传到机群上算例的 **input** 文件夹中, 运行模式就行了。

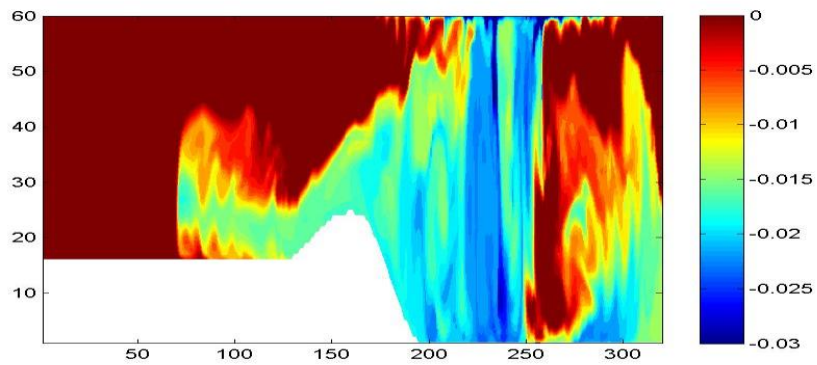
给两张成功运算的图，温度场：
积分 10000 次：



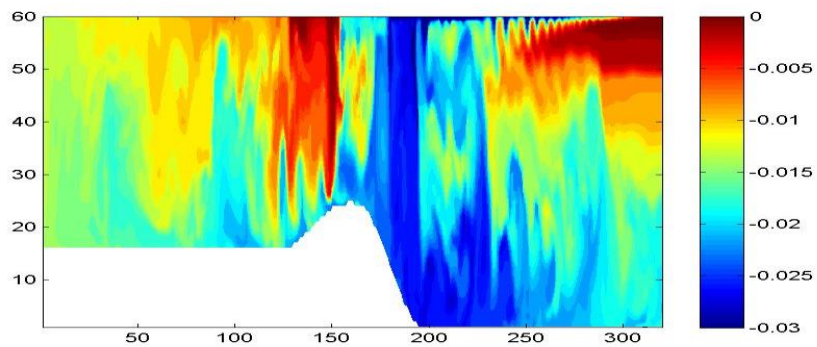
积分 37200 次



积分 84100 次

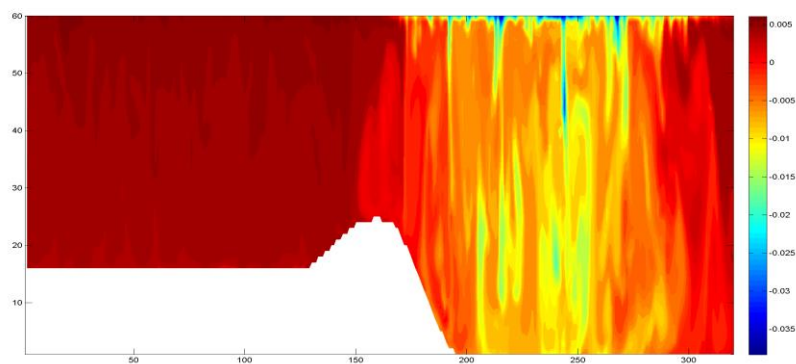


积分 160000 次

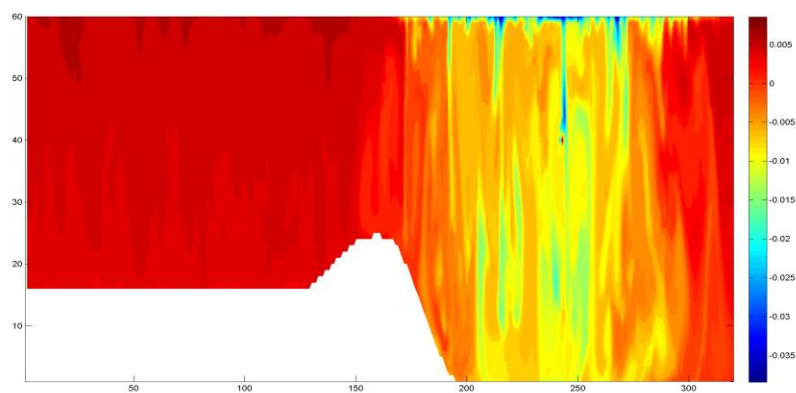


但在此之前，我们曾经取过时间步长为 20 秒，失败了，取了积分失败前几步的温度场，发现是计算不稳定，应该降低积分步长，我们调成 4 秒，还是不稳定，又调成 2 秒才可以的，下面看一下 20s 试验中积分失败前几步的图像：

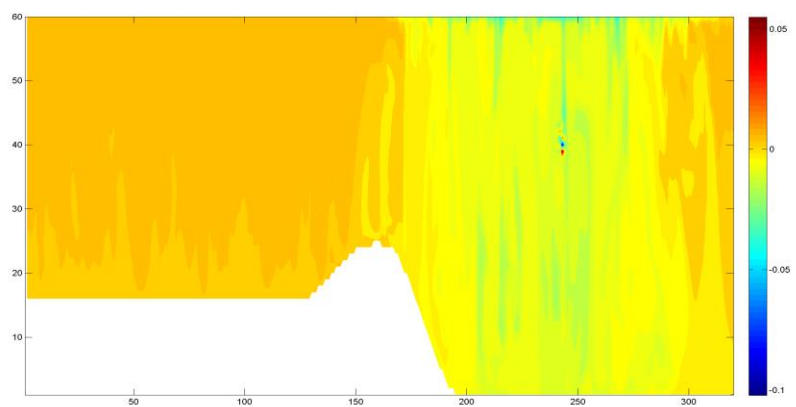
前 5 步



前 4 步



前 3 步



从 2 维到三维——碰钉子以后

这次的经验再次证明，古人说“心急吃不了热豆腐”是非常有道理的！
做模式要一次修改一小点，决不能一次修改一大片！

目录“\Results\01_tutorial_plume_on_slope\test03”记录着我的失败经验，在那个算例中，我对上一个成功的算例进行了大刀阔斧的改造，最终模式就是不点我，运行 mitgcmuv 时告诉我 data 中有错，我费了很大劲还是不知道问题处在哪，最后我决定还是一点一点的改，所以我们下面这个算例中，我们将在上一个成功算例的基础上做最小改动，让它三维化：

事实上，只有两个文件我们需要改动：**code\SIZE..h** 和 **input\gendata_zzg.m**：

对 **SIZE.h** 的修改：

```
0 10 20 30 40 50 60 70
34 C Nr :: No. points in Z for full process domain.
35     INTEGER sNx
36     INTEGER sNy
37     INTEGER OLx
38     INTEGER OLy
39     INTEGER nSx
40     INTEGER nSy
41     INTEGER nPx
42     INTEGER nPy
43     INTEGER Nx
44     INTEGER Ny
45     INTEGER Nr
46     PARAMETER (
47         sNx = 80, ← 从 1 改到 5
48         sNy = 5, ←
49         OLx = 3,
50         OLy = 3,
51         nSx = 4, ← 从 1 改到 4
52         nSy = 4, ←
53         nPx = 1,
54         nPy = 1,
55         Nx = sNx*nSx*nPx, ← y方向上的网格由
56         Ny = sNy*nSy*nPy, 1变成了40
57         Nr = 60)
58
```

对 **gendata_zzg.m** 的修改：

```
7
8 % Dimensions of grid
9 nx=320; % x方向的网格数
10 ny=20; % y方向的网格数 从1改为20
11 nz=60; % z方向的网格数

58 h(1:160,:)=50;
59 h(140:180,:)=80;
60 for i=1:ny
61     h(:,i)=smooth(h(:,i),10);
62     h(:,i)=smooth(h(:,i),10);
63     h(:,i)=smooth(h(:,i),25);
64 end
65 for i=1:nx
66     for j=1:ny
67         d(i,j) = h(i,j) - H ;
68     end
69 end
70 clear i j
71 d(1,:)=0.0; %这个地方太麻烦了，看看data.obcs（边界设置文件），
72 d(320,:)=0.0;
73 d(:,1)=0.0; 加上南北的固壁边界
74 d(:,ny)=0.0;
75 fid=fopen('topog.slope','w',ieee); fwrite(fid,d,prec); fclose(fid);
76
```

嘿嘿，兄弟姐妹们，之后就是转啦！（上述修改的历史保存在 K:\Models\MITgcm\Results\01_tutorial_plume_on_slope\test04\01_history\01_从 2 维到三维\configure 目录中）

当我们转的时候就可以发现，速度明显变慢了，这是可以理解的，二维时网格是 320×60 ，现在变成三维的啦， $320 \times 40 \times 60$ ，足足大了 40 倍，所以我们要关心一下计算时间：

STDERR.0000	193	0000	文件	2009-12-25 20:50:41	
T.0000000000.001.001...	96,000	DATA	文件	2009-12-25 20:05:49	这个时间是模式开始运转的时间
T.0000000000.001.001...	178	META	文件	2009-12-25 20:05:49	
T.0000001000.004.003...	110	META	文件	2009-12-25 20:50:40	
T.0000001600.004.004...	96,000	DATA	文件	2009-12-25 20:50:40	这时间是模式运行完成的时间
T.0000001600.004.004...	178	META	文件	2009-12-25 20:50:40	
T.init	3,072,000	INIT	文件	2009-12-25 20:05:34	
topog.slope	51,200	SLOPE	...	2009-12-25 20:05:34	

可见模式用了 45 分钟运算完成 1600 步的积分，大约一分钟 35 步，故我们对运算效率有了一个认识：

1 个 CPU 运算 $320 \times 40 \times 60$ 的网格，一分钟大约能运算 35 步

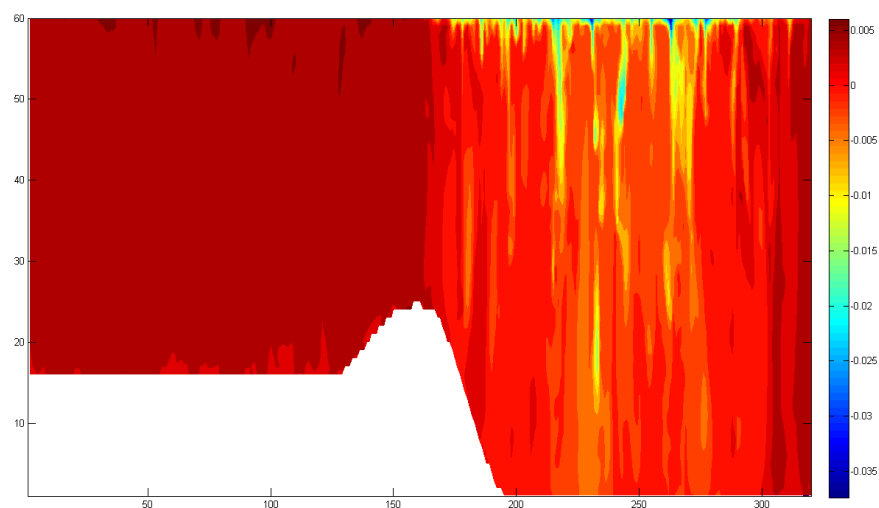
呵呵，最后我们看一下我们战果：

```

1 - close all
2 - clear
3 - addpath ../../../../matlab/ % 使用模式提供的程序库
4
5
6 - T=rdmds('T.0000001600'); %使用函数rdmds读取温度数据
7
8 - t_EtW=squeeze( T(:,19,:) );
9 - t_EtW=t_EtW';
10 - t_EtW=flipud(t_EtW);
11 - t_EtW(t_EtW==0)=NaN;
12 - contourf(t_EtW,20,'linestyle','none')

```

作图 16000 步的温度：



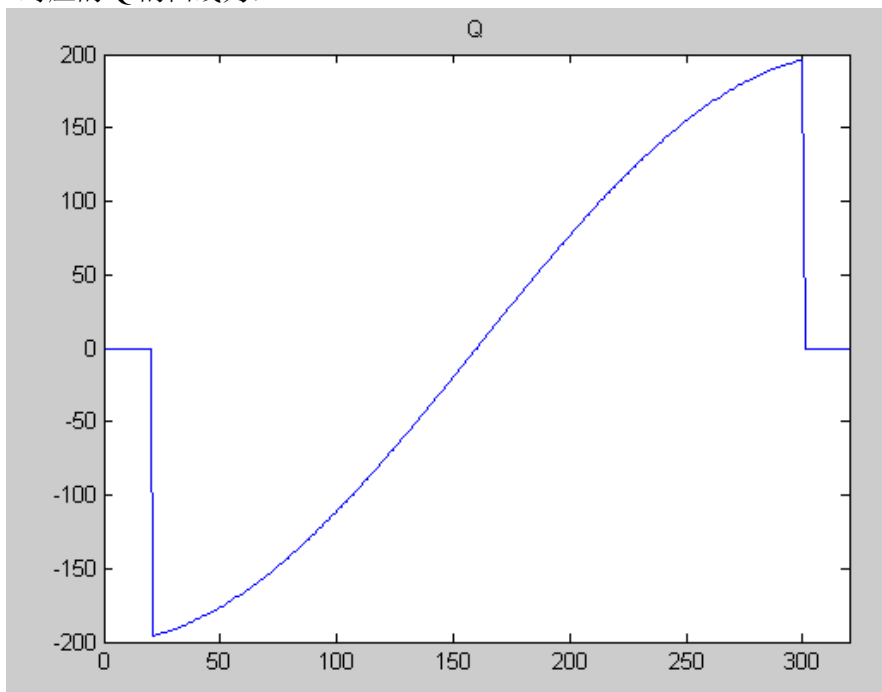
下面让我们一点一点的改模式吧

我们发现不稳定性出现在离开边界很近的地方，所以我们想是不是边界有问题，这时赵玮老师天才的指出，在边界附近 20 点内不能有加热和冷却，即：

gendata_zzg.m:

```
37 % Heat Flux
38 - Qo=200; % 热通量的参考值, 正值为cooling
39 - Q=0.0*rand([nx,ny]);
40 - for i=1:nx
41 -   Q(i,:) = Q(i,:) - Qo*cos( pi*x(i)/Lx );
42 - end
43 - clear i
44 - Q(1:20)=0;
45 - Q(301:320)=0;
46 - %Q(Q<0)=0; % 取消加热
47 - fid=fopen('Qnet.forcing','w',ieee); fwrite(fid,Q,prec); fclose(fid); % 将热通量数据写入文件
```

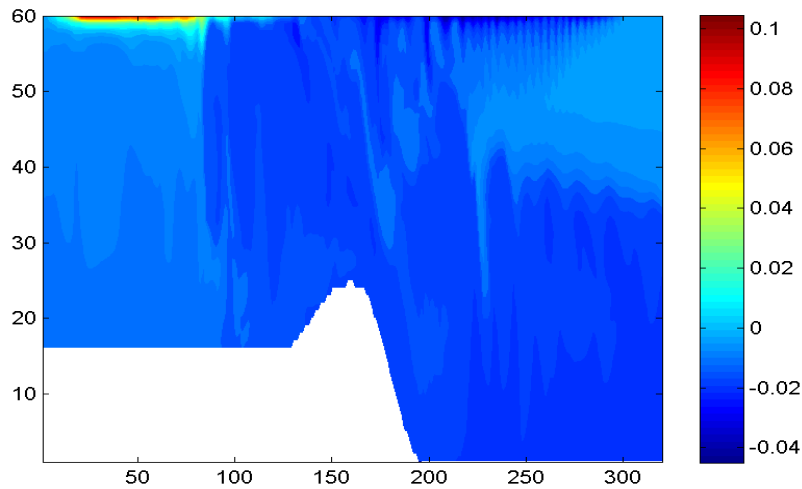
对应的 Q 的曲线为：



另外，我们还在 data 中动了点手脚，把铅直混合率调成了 10^{-3}

```
5 # Continuous equation parameters
6 &PARM01
7 tRef=60*1.,
8 sRef=60*35.,
9 viscA4=0.0E4,
10 viscAh=1.E-2,
11 viscAz=1.E-3,
12 no_slip_sides=.TRUE.,
13 no_slip_bottom=.TRUE.,
14 diffK4T=0.E4,
15 diffKhT=0.E-2,
16 diffKzT=1.E-3,
17 diffK4S=0.E4,
18 diffKhS=0.E-2,
19 diffKzS=1.E-3,
```

嗯，稳定了，来看看结果吧 (K:\Models\MITgcm\Results\02_BOX Model\test01\T.0000860000.tif):



显然有问题呀，唉，你看看，右侧那个暖水明显是从开边界进来捣乱的，开边界呀开边界，给我闭上吧：

控制开边界的文件 **data.obcs** 中的兄弟，都休息吧（全部注释掉）

```
K:\Models\MITgcm\Results\02_BOX Model\test02\input\data.obcs
1 # Open-boundaries
2 &OBCS_PARM01
3 # OB_Ieast=1*-1,
4
5 3 useOrlanskiEast=.TRUE.,
6 &
7
8 # Orlanski parameters
9 &OBCS_PARM02
10 # Cmax=0.45,
11 # cVelTimeScale=1000.,
12 &
```

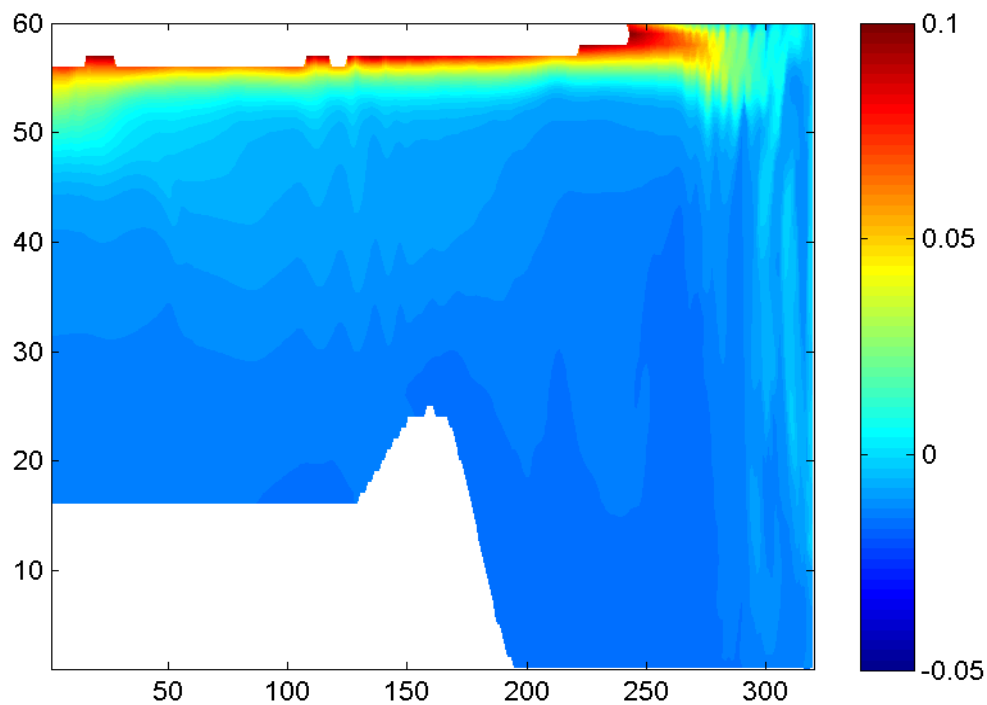
控制使用开边界包的 **data.pkg** 中开边界给我 false 吧

```
K:\Models\MITgcm\Results\02_BOX Model\test02\input\data.pkg
1 # Packages
2 &PACKAGES
3 # useOBCS=.TRUE.,
4 useOBCS=.FALSE.,
5 &
```

最后，**gendata_zzg.m** 中东边界的高墙竖起来吧：

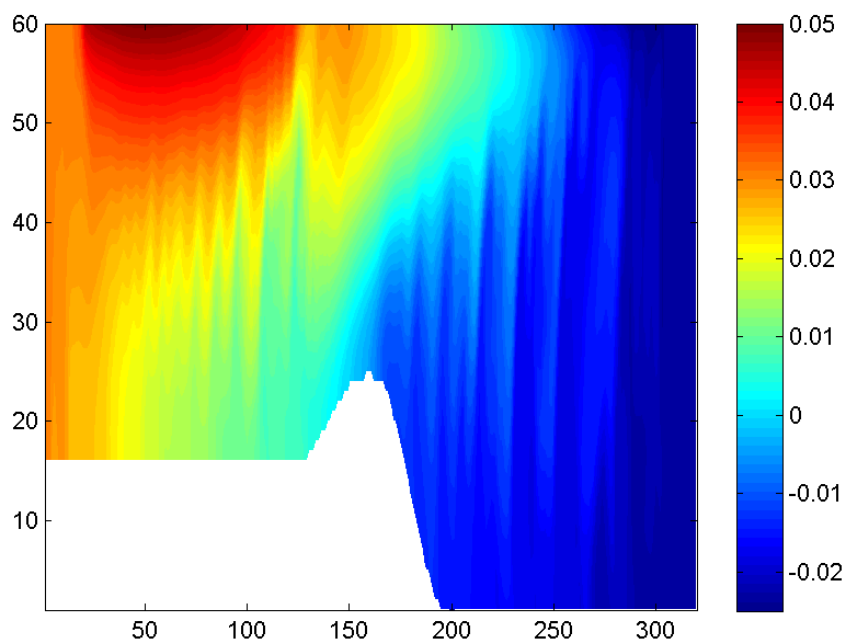
```
72 - d(1,:)=0.0; %这个地方太猥琐了，看看data.obcs（边界设
73 - d(320,:)=0.0;
74 - fid=fopen('topog.slope','w',ieee); fwrite(fid,d,prec); fclose(fid);
75
```

来来来，show 一下结果吧（这个算例的结果忠实地记录在 [Results\02_BOX Model\test02\](#) 文件夹中），下面是 1160000 步的积分结果，我们为了画图好看吧温度特别高的区域设成 NaN 了，我们看到了温跃层的成长，同时看到温跃层东侧与下降对流区的地方出现了不稳定性，总是产生波动：

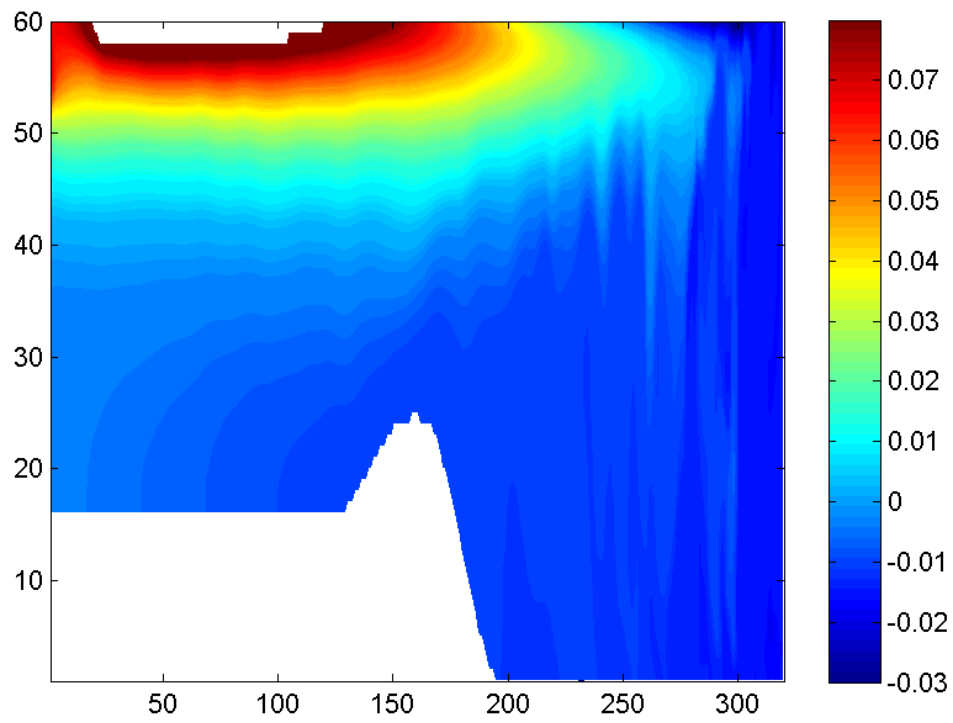


同时我们还调了不同的混合率做了相关试验：

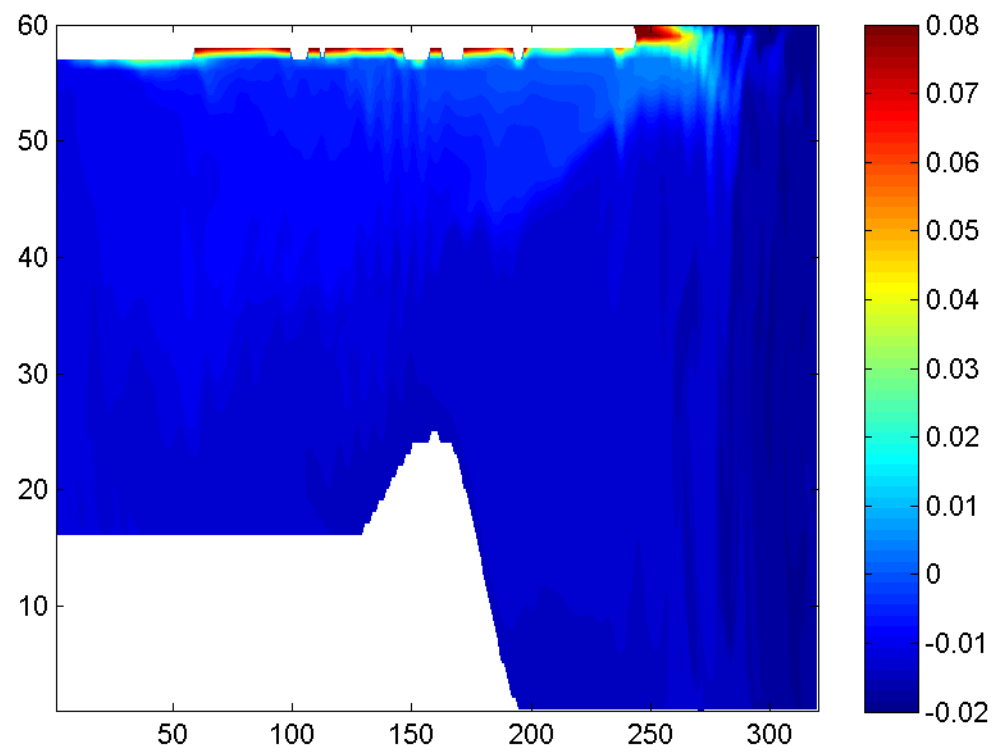
Diff=10⁻¹:



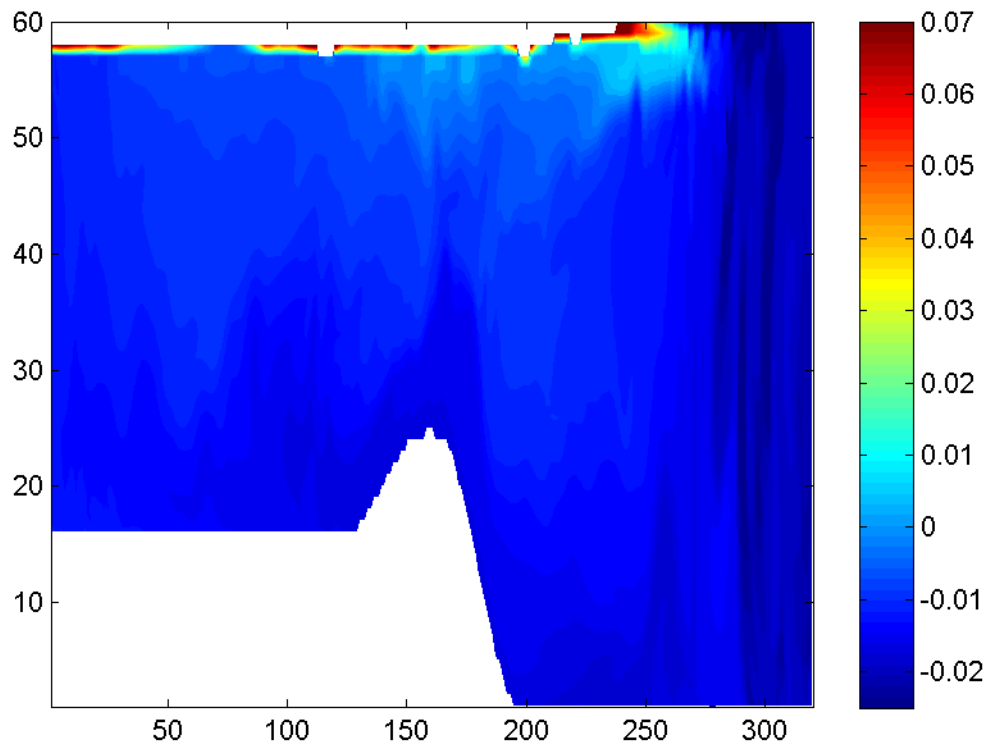
Diff=10⁻²:



Diff=10⁻⁴:



Diff=10⁻⁵:



嗯，果然有点意思，“Results\02_BOX Model”中 test02-test06 忠实地记录这些试验
不过为了达到我们加入不均匀混合的终极目的，还有一些工作需要我们来进行，启动 test07：

修改“code”文件夹中的 CPP_OPTION 文件,加入一行程序，告诉 gcm 我们要制定每一点的铅直混合率：

```
16 C o Include/exclude call to S/R CONVECT
17 #define INCLUDE_CONVECT_CALL
18
19 C o Allow full 3D specification of vertical diffusivity
20 #define ALLOW_3D_DIFFKR
21
22 C o Include/exclude call to S/R CALC_DIFFUSIVITY
23 #define INCLUDE_CALC_DIFFUSIVITY_CALL
```

在“input”的 data 中，要做两处修改：

```
12 no_slip_sides=.TRUE.,
13 no_slip_bottom=.TRUE.,
14 diffK4T=0.E4,
15 diffKhT=0.E-2,
16 diffKzT=0.E-1,
17 diffK4S=0.E4,
18 diffKhS=0.E-2,
19 diffKzS=0.E-3,
20 f0=0.e-4,
```

这些混合率都设为 0，为了不影响我们后面用文件指定的混合率：


```

68 # Input datasets
69 &PARM05
70 bathyFile='topog.slope',
71 hydrogThetaFile='T.init',
72 surfQfile='Qnet.forcing',
73 diffKrFile='diffK.bin'
74 #hydrogThetaFile='T.pickup',
75 #uVelInitFile='U.pickup',
76 #pSurfInitFile='Eta.pickup',
77 &

```

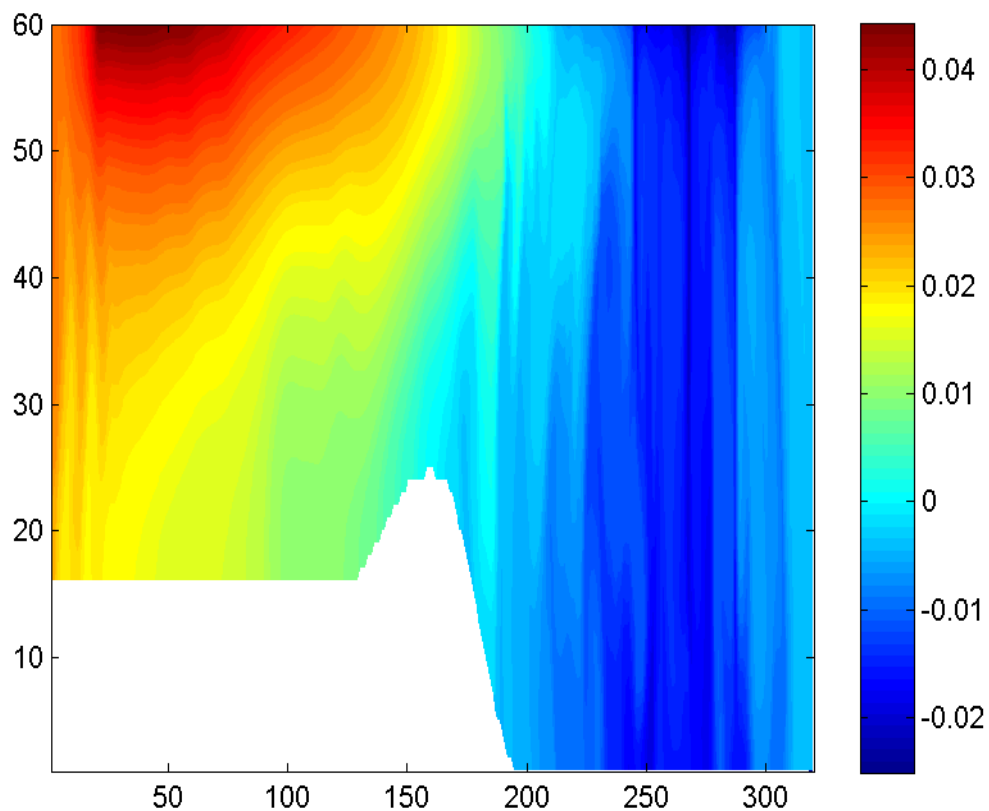
在这里，我们告诉 gcm 制定混合率的文件名字叫“**diffK.bin**”，最后我们要在 **gendata_zzg.m** 中加入生成这个混合率文件的程序（将混合率设为均匀的 10^{-1} ）：

```

53
54 % Specify 3D vertical diffusivity
55 - diff=0.1*ones([nx,ny,nz]);
56 - fid=fopen('diffK.bin','w',ieee); fwrite(fid,diff,prec); fclose(fid);
57

```

我们来看一下运行了 100000 步的结果，可以的！

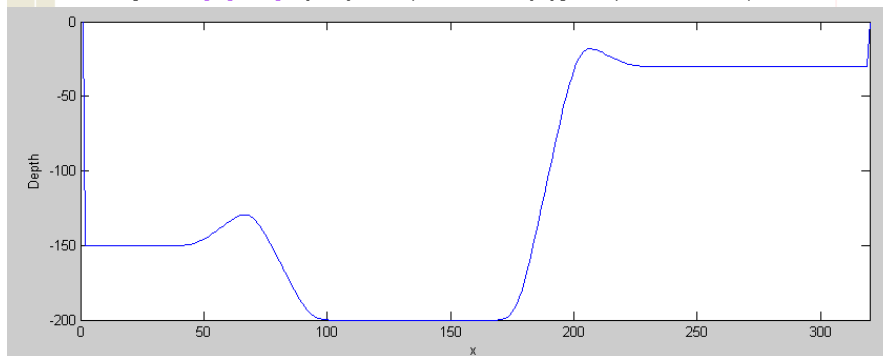


下面我们要做的就是加入不均匀的混合，不过在此之前我们要动一些其他的手脚，我们在前面几个算例中发现，在加热和冷却交接的地方，出现了强烈的斜压不稳定性，我想的方法就是学习真实海洋，在冷却下降的地方给它弄个海盆，修改 **gendata_zzg.m**:

```

67
68 - d=0.0*rand([nx,ny]);
69 - h=0.0*rand([nx,ny]);
70 - h(1:59)=50;
71 - h(60:80)=80;
72 - h(200:320)=170;
73 - h(190:210)=190;
74 - % h(300:320)=[0:7.5:150]; % 在右侧加斜坡
75 - h=smooth(h,10);
76 - h=smooth(h,10);
77 - h=smooth(h,25);
78 - for i=1:nx
79 -     for j=1:ny
80 -         d(i,j) = h(i,j) - H ;
81 -     end
82 - end
83 - clear i j
84 - d(1,:)=0.0; %这个地方太麻烦了，看看data.obcs（边界设置文件）
85 - d(320,:)=0.0;
86 - fid=fopen('topog.slope','w',ieee); fwrite(fid,d,prec); fclose(fid);

```

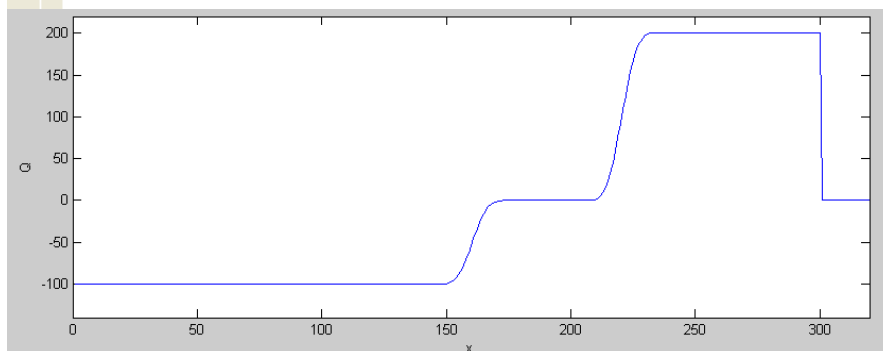


当然了，加热冷却也得改：

```

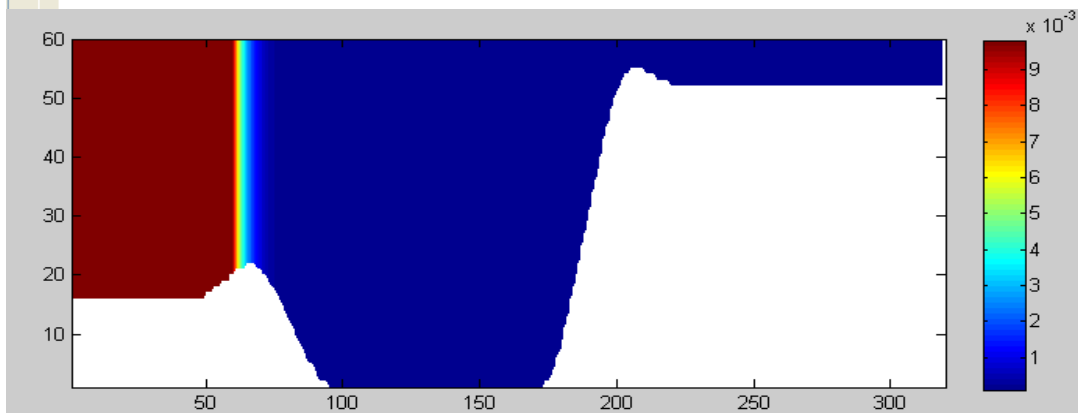
37 - % Heat Flux
38 - Q0=200; % 热通量的参考值, 正值为cooling
39 - Q=zeros([nx,ny]);
40 - Q(221:300)=Q0;
41 - Q(1:160)=-Q0/2;
42 - Q(21:300)=smooth(Q(21:300),10);
43 - Q(21:300)=smooth(Q(21:300),10);
44 - Q(21:300)=smooth(Q(21:300),10);
45 - fid=fopen('Qnet.forcing','w',ieee); fwrite(fid,Q,prec); fclose(fid); % 将热通量数据写入文件

```



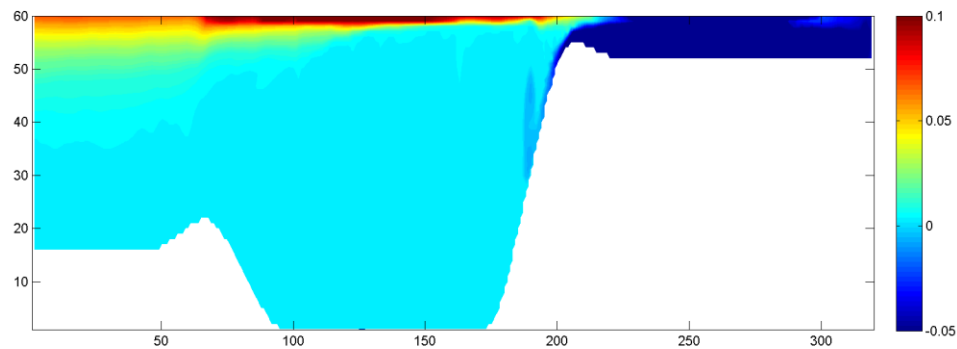
最后，当然是要设置不均匀的铅直混合率，在左侧小海盆中为 10^{-2} ，在右侧所有地方都为 10^{-4} ，在海槛处均匀过渡：

```
52 % Specify 3D vertical diffusivity
53 - diff=0.0001*ones([nx,ny,nz]);
54 - diff(1:60,:)=0.01;
55 - for k=1:60
56 -     for j=1:1
57 -         for i=61:80
58 -             diff(i,j,k)=10^( -2 -(i-60)*(2)/20 );
59 -         end
60 -     end
61 - end
62 - fid=fopen('diffK.bin','w',ieee); fwrite(fid,diff,prec); fclose(fid);
```

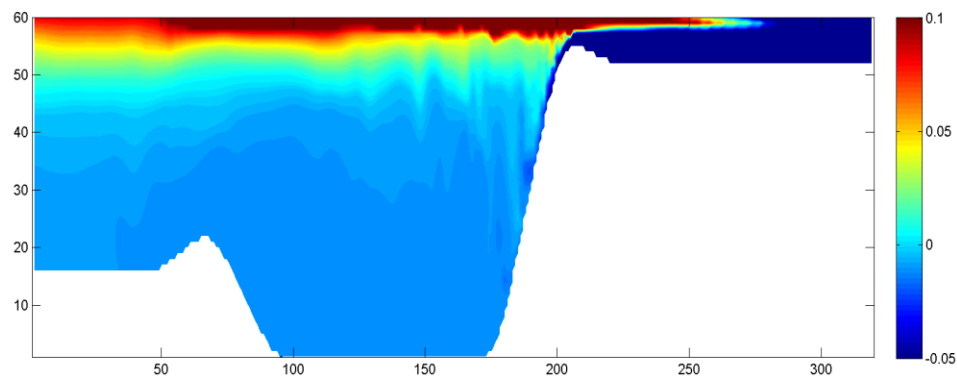


这个例子忠实地记录在文件夹：[MITgcm\Results\02_BOX Model\test08](#)

下图为 70000 步的温度分布图，可以明显的看到小海盆局地加强混合的效果：



下图为 890000 步的温度分布图

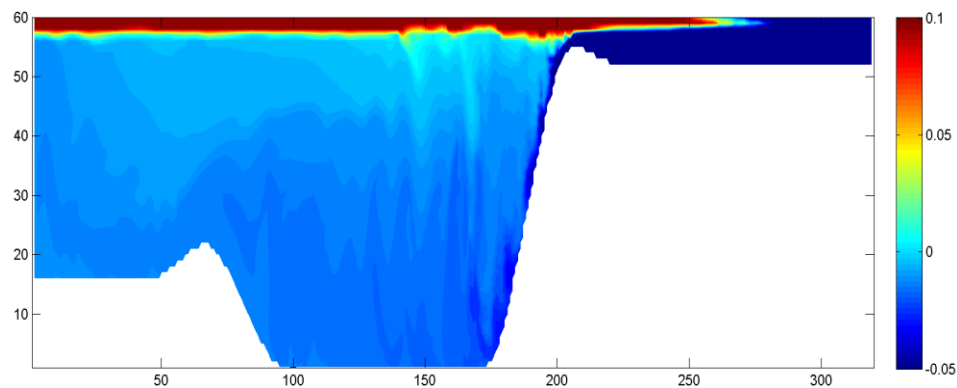


同时，我们还作了一个对比试验，就是将上一个例子 test08 中不均匀的混合全改为 10^{-4} ，其他设置不变，修改 **gendata_zzg.m**:

```
52 % Specify 3D vertical diffusivity
53 - diff=0.0001*ones([nx,ny,nz]);
54 % diff(1:60,:,:) = 0.01;
55 % for k=1:60
56 %     for j=1:1
57 %         for i=61:80
58 %             diff(i,j,k)=10^(-2-(i-60)*(2)/20);
59 %         end
60 %     end
61 % end
62 - fid=fopen('diffK.bin','w',ieee); fwrite(fid,diff,prec); fclose(fid);
```

这个实验的结果忠实地记录在文件夹: MITgcm\Results\02_BOX Model\test09

我们看一下它第 420000 步的结果，嘻嘻，不太一样吧:

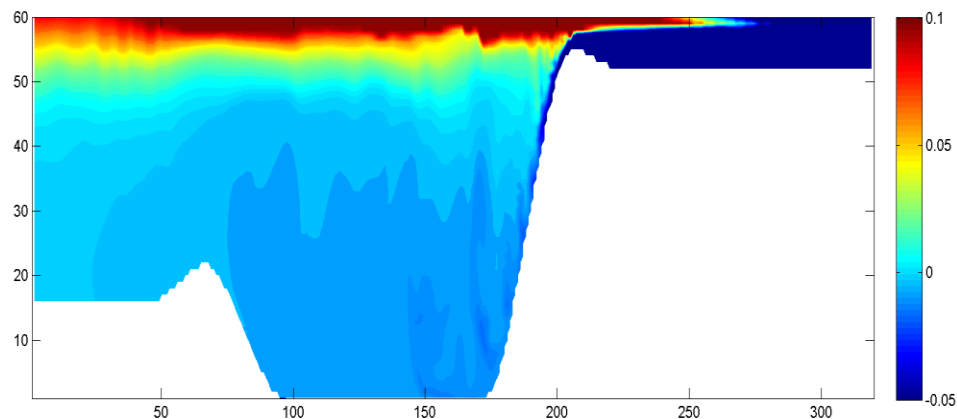


我们还弄了一个试验，看看水平混合的作用，在 test08 的基础上，改 **data**:

```
14 diffK4T=0.E4,
15 diffKhT=1.E-3,
16 diffKzT=0.E-1,
17 diffK4S=0.E4,
18 diffKhS=1.E-3,
19 diffKzS=0.E-3,
```

这个实验的结果忠实地记录在文件夹: MITgcm\Results\02_BOX Model\test10

我们看一下它第 420000 步的结果:



大家注意到，我们之前的实验都是在一个小水池中进行的，它长不过 6km，深不过 200m，这跟实际的海洋在尺度上有很大区别，所以我们考虑着至少将海盆尺度修改为实际海洋的尺度，修改 **gendata_zzg.m**:

修改网格

```
8 % Dimensions of grid
9 - rx=320; % x方向的网格数
10 - ny=1; % y方向的网格数
11 - nz=60; % z方向的网格数
12 % Nominal depth of model (meters)
13 - H=4200.0; % z方向上的区域厚度
14 % Size of domain
15 - Lx=30000*320; % x方向上的区域长度
```

修改地形

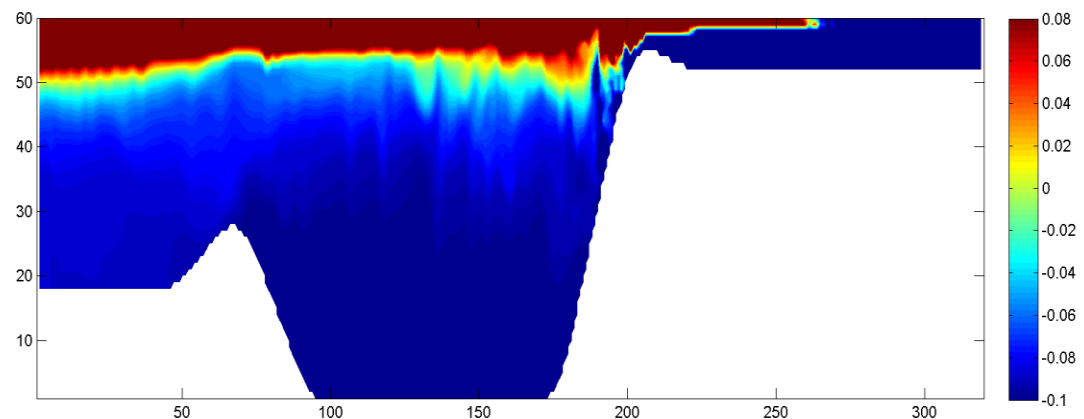
```
67 - d=0.0*rand([nx,ny]);
68 - h=0.0*rand([nx,ny]);
69 - h(1:59)=1200;
70 - h(60:80)=2200;
71 - h(200:320)=3570;
72 - h(190:210)=3990;
73 % h(300:320)=[0:7.5:150]; % 在右侧加斜坡
74 - h=smooth(h,10);
75 - h=smooth(h,10);
76 - h=smooth(h,25);
```

然后我们要做的就是调整时间步长，我们进行了系列测试，最后发现步长 3600s 时将不稳定，但 1200s 时仍能保持稳定，为稳妥起见我们选择时间步长为 600s，修改 **data**:

```
44 # Time stepping parameters
45 &PARM03
46 niter0=0,
47 #nTimeSteps=600,
48 nTimeSteps=4000000,
49 deltaT=600.0,
50 # deltaT=1200.0,
51 abEps=0.01,
52 pChkptFreq=0.0,
53 chkptFreq=0.0,
54 dumpFreq=12000000.0,
55 # taveFreq=86400.0,
56 monitorFreq=6000.,
57 # staggerTimeStep=.TRUE.,
58 &
```

```
# Gridding parameters
&PARM04
usingCartesianGrid=.TRUE.,
usingSphericalPolarGrid=.FALSE.,
delXfile='dx.bin',
dYspacing=2000.,
delZ=60*70.0,
&
```

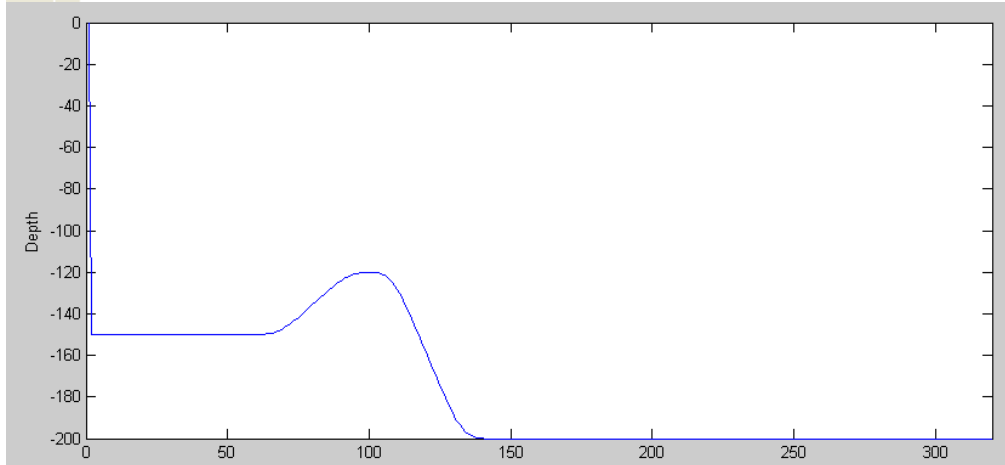
这个实验的结果忠实地记录在文件夹: MITgcm\Results\02_BOX Model\test11
看一下 880000 步的结果:



呼呼，出现吧！开边界！

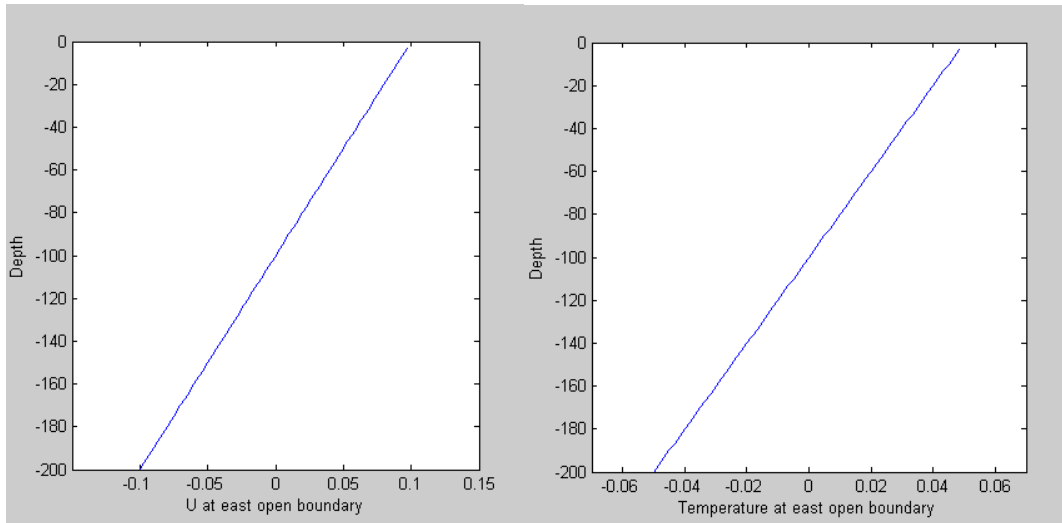
在这个实验中，我们从 test08 的基础上改，首先去掉右侧的冷却海盆，改 **gendata_zzg.m**:

```
71 - h(1:80)=50;
72 - h(81:120)=80;
73 - % h(300:320)=[0:7.5:150]; % 在右侧加斜坡
74 - h=smooth(h,10);
75 - h=smooth(h,10);
76 - h=smooth(h,25);
将右侧的开边界打开:
83 - d(1,:)=0.0; %这个地方太麻烦了，看看data.obcs（边界设置文件）
84 - % d(320,:)=0.0;
85 - fid=fopen('topog.slope','w',ieee); fwrite(fid,d,prec); fclose(fid);
```



加入控制东开边界温度和东西向流速的部分，其中控制温度的文件叫 **tEast.bin**，控制东西向流速的文件叫 **uEast.bin**，下面程序中有一个变量 **cir**，是设置开边界上重复的周期的，一般用于加潮汐信号用，我们这里是定常的，故 **cir = 1**:

```
87 - % get the open boudary
88 - uEast=0.0*ones([ny,nz,cir]);
89 - for i=1:nz
90 -     uEast(i)=0.1-0.2/60*i;
91 - end
92 - clear i
93 - fid=fopen('uEast.bin','w',ieee); fwrite(fid,uEast,prec); fclose(fid);
94
95 - tEast=0.0*ones([ny,nz,cir]);
96 - for i=1:nz
97 -     tEast(i)=0.05-0.1/60*i;
98 - end
99 - clear i
100 - fid=fopen('tEast.bin','w',ieee); fwrite(fid,tEast,prec); fclose(fid);
```



嗯，似乎热通量也得改改，现在边界处的平流相当于冷却了，所以需要我们在上边界只加加热就行了：

```
38 % Heat Flux
39 - Q0=200; % 热通量的参考值, 正值为cooling
40 - Q=-Q0.*ones([nx,ny]);
41 % Q(221:300)=Q0;
42 % Q(21:180)=-Q0/2;
43 % Q(21:300)=smooth(Q(21:300),10);
44 % Q(21:300)=smooth(Q(21:300),10);
45 % Q(21:300)=smooth(Q(21:300),10);
46 - fid=fopen('Qnet.forcing','w', 'ieee'); fwrite(fid,Q,prec); fclose(fid); % 将热通量数据写入文件
```

在 **data.pkg** 文件中要激活开边界程序包：

```
0 10 20 30
1 # Packages
2 &PACKAGES
3 useOBCS=.TRUE.,
4 &
5
```

在 **data.obcs** 文件中设置开边界：

```
1 # Open-boundaries
2 &OBCS_PARM01
3 OB_Ieast=1*-1, ← 这个 -1 代表最后一个网格, 由于是二维, 边界在经向没有宽度, 所以是1*-1
4 useOBCSbalance=.TRUE., ← 用于平衡开边界的质量通量
5 useOBCSsprescribe=.TRUE., ← 告诉模式开边界上的各种数值我要能直接给定
6 useOBCSsponge=.TRUE. ← 使用海绵边界条件
7 OBEuFile='uEast.bin' ← 指定 uEast.bin 文件来控制东边界上的纬向流速
8 OBETFile='tEast.bin' ← 指定 tEast.bin 文件来控制东边界上的温度
9 &
10
11 # *****
12 # Sponge Layer Parameters.
13 # *****
14 &OBCS_PARM03
15 Urelaxobcsinner = 200000.0,
16 Urelaxobcsbound = 90000.0,
17 #Vrelaxobcsinner = 270000.0,
18 #Vrelaxobcsbound = 90000.0,
19 spongeThickness = 20, ← 海绵边界的厚度
20 &
21 # end of the parameters.
```

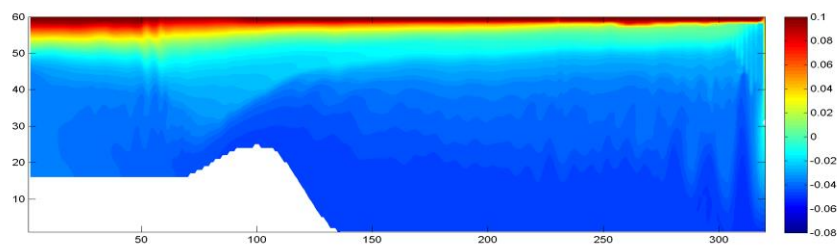
最后，我们在 `code` 文件夹中还需要添加一个专门控制开边界程序包的文件 **OBCS_OPTION.h**，这个文件可以从 `MITgcm\verification\exp4\code` 中直接拷贝过来，不需要做任何改动，其内容主要是激活我们需要的功能模块：

```

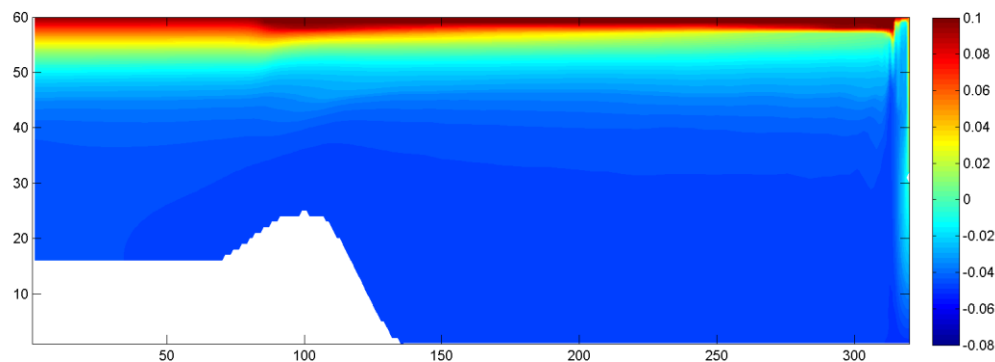
1  # $Header: /u/gcm-pack/MITgcm/verification/exp4/code/OBCS_OPTIONS.h,v 1.1 2005/10/11 13:00:56 mlosch Exp $
2  C $Name: $
3
4  C CPP options file for OBCS package
5  C
6  C Use this file for selecting options within the OBCS package
7  C
8  C OBCS is enabled with ALLOW_OBCS in CPP_OPTIONS.h
9
10 #ifndef OBCS_OPTIONS_H
11 #define OBCS_OPTIONS_H
12 #include "PACKAGES_CONFIG.h"
13 #include "CPP_OPTIONS.h"
14
15 #ifdef ALLOW_OBCS
16
17 C Enable individual open boundaries
18 #define ALLOW_OBCS_NORTH
19 #define ALLOW_OBCS_SOUTH
20 #define ALLOW_OBCS_EAST
21 #define ALLOW_OBCS_WEST
22
23 C This include hooks to the Orlanski Open Boundary Radiation code
24 #define ALLOW_ORLANSKI
25
26 C Enable OB values to be prescribed via external fields that are read
27 C from a file
28 #define ALLOW_OBCS_PRESCRIBE
29
30 C This includes hooks to sponge layer treatment of uvel, vvel
31 #undef ALLOW_OBCS_SPONGE
32
33 C balance barotropic velocity
34 #undef ALLOW_OBCS_BALANCE
35
36 #endif /* ALLOW_OBCS */
37 #endif /* OBCS_OPTIONS_H */

```

这个实验的结果忠实地记录在文件夹：`MITgcm\Results\02_BOX Model\test12`
我们来看一下这个实验 330000 步的结果：



1468800 前后 86400 步平均的结果：



底摩擦出现吧，加长版中央海盆抑制惯性上冲

我们回忆一下 test11 中的例子，发现在左侧海盆中有等密度面上扬的趋势，这暗示我们即使不加局部强混合，由于中央海盆太短，底层的重流体凭借下坡积累的动能，仍能冲过左侧海槛，进入左侧海盆，为了搞定这一问题，我们想通过改网格宽度来加长中央海盆，以及加底摩擦的方法来去掉惯性上冲。我们在 test11 的基础上改

加底摩擦需要对 **data** 做改动，设 **Cd** 为 0.003:

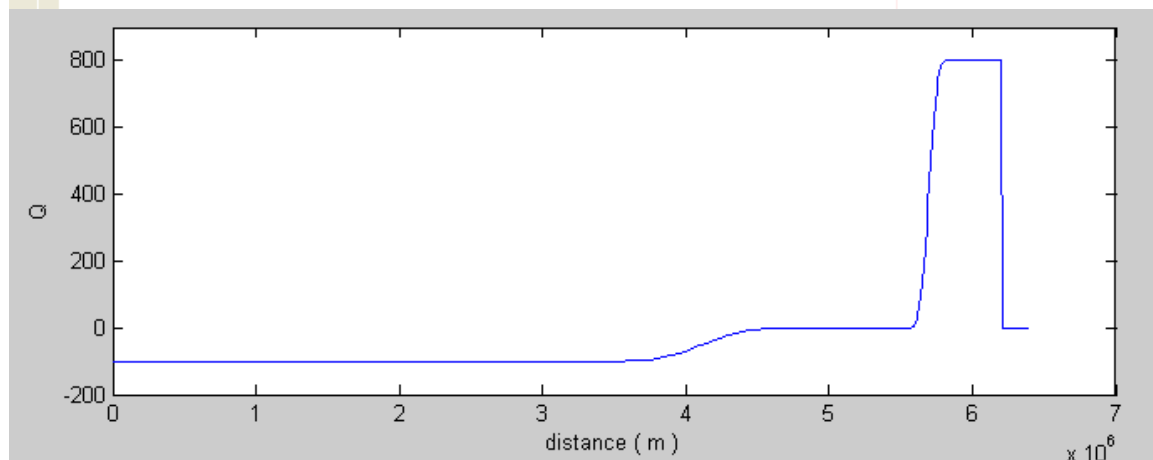
```
32 bottomDragLinear=0.E-4,  
33 bottomDragQuadratic=0.003,  
34 tempAdvScheme=33,
```

加长中央海盆，改 **gendata_zzg.m**:

```
23 - dx=10000.*ones([nx,ny]);  
24 - dx(121:200)=50000;  
25 - dx=smooth(dx,15);  
26 - dx=smooth(dx,15);  
27 - dx=smooth(dx,15);  
28  
29 - fid=fopen('dx.bin','w',ieee); fwrite(fid,dx,prec); fclose(fid); % 这是输出水平网格的语句
```

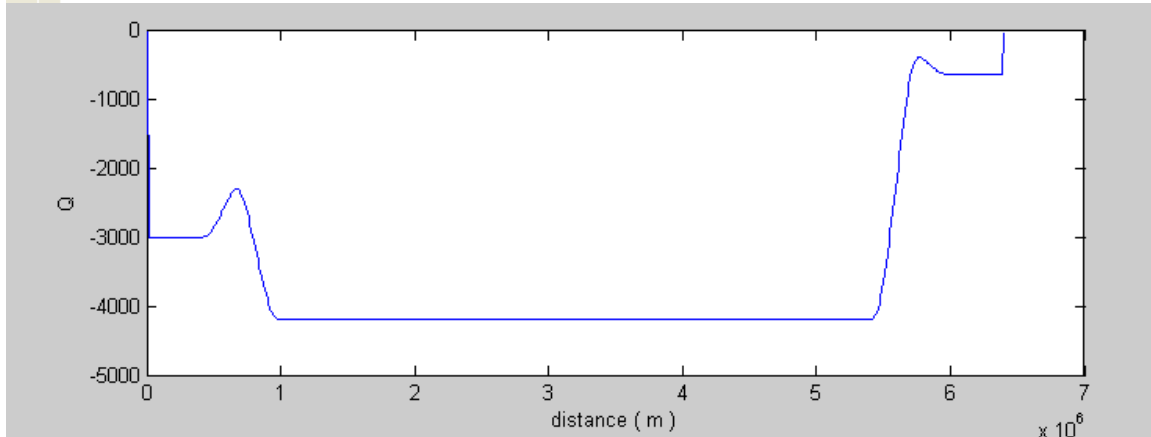
改加热:

```
43 % Heat Flux  
44 - Q0=200; % 热通量的参考值,正值为cooling  
45 - Q=zeros([nx,ny]);  
46 - Q(1:178)=-Q0/2;  
47 - sum_Q_positive=sum(Q.*dx);  
48 - sum_dx=sum(dx(250:300));  
49 - Q(250:300)=-sum_Q_positive/sum_dx;  
50 - Q_sum=sum(Q.*dx);  
51 - Q(21:300)=smooth(Q(21:300),10);  
52 - Q(21:300)=smooth(Q(21:300),10);  
53 - Q(21:300)=smooth(Q(21:300),10);  
54 - fid=fopen('Qnet.forcing','w',ieee); fwrite(fid,Q,prec); fclose(fid); % 将热通量数据写入文件
```

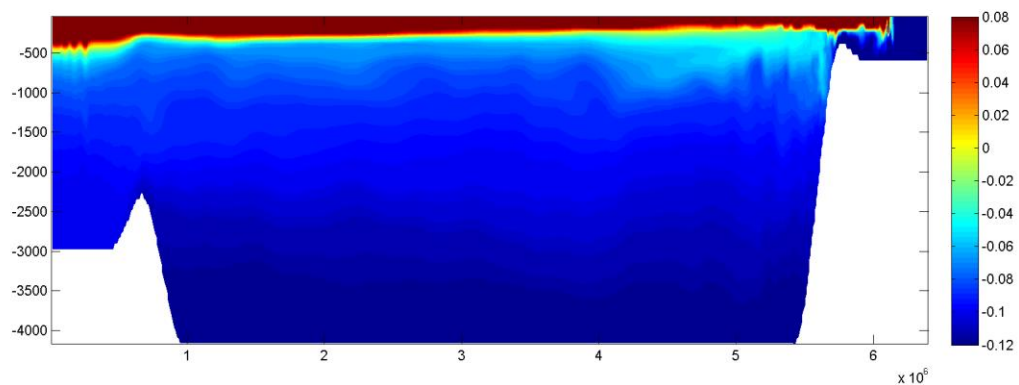


改地形:

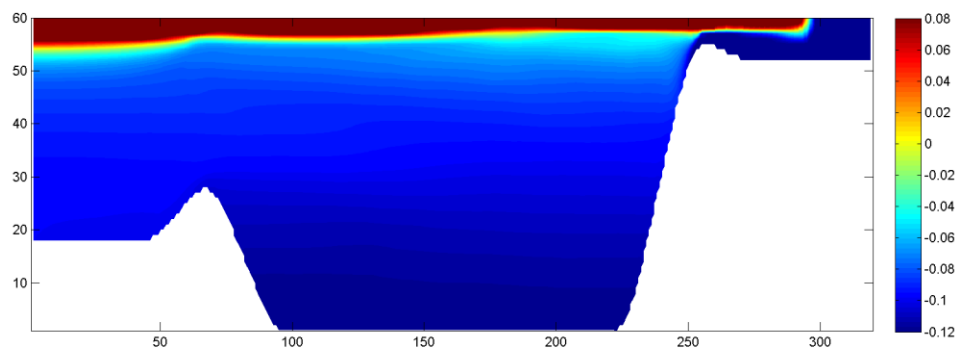
```
76 - d=0.0*rand([nx,ny]);
77 - h=0.0*rand([nx,ny]);
78 - h(1:59)=1200;
79 - h(60:80)=2200;
80 - h(260:320)=3570;
81 - h(240:260)=3990;
82 - % h(300:320)=[0:7.5:150]; % 在右侧加斜坡
83 - h=smooth(h, 10);
84 - h=smooth(h, 10);
85 - h=smooth(h, 25);
```



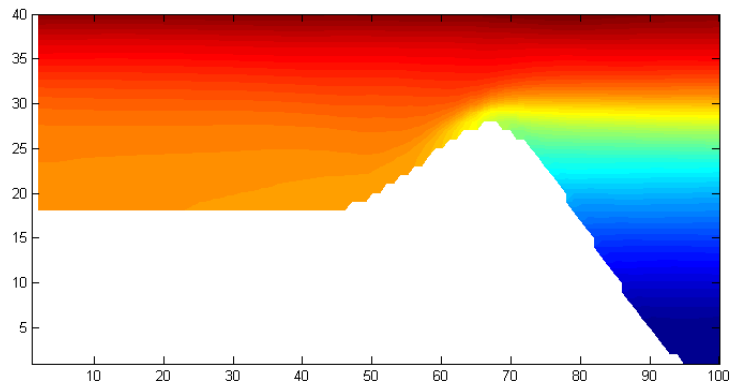
这个实验的结果忠实地记录在文件夹: [MITgcm\Results\02_BOX Model\test13](#)
我们来看一下这个实验 1060000 步的结果:



1100000 前后 100000 步平均的结果:



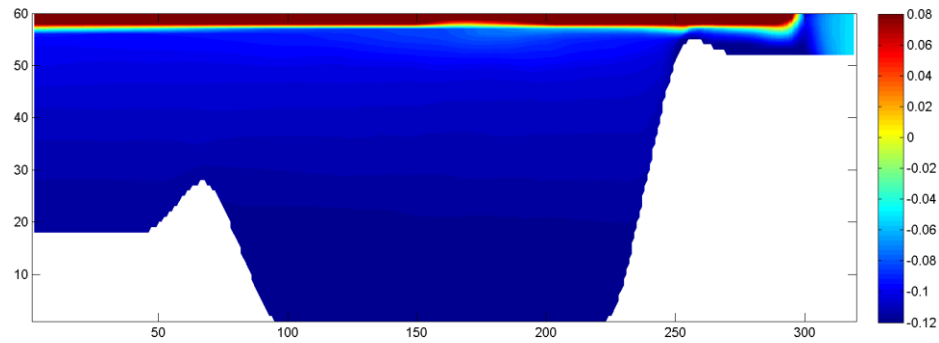
我们将右侧局部强混合海盆放大看看：



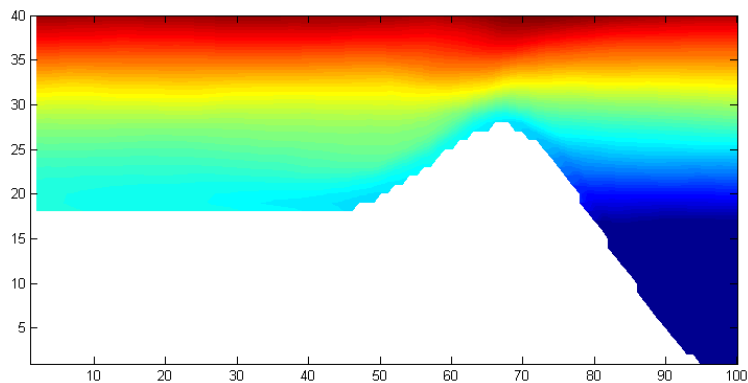
为了对比，我们还需要一个不在左侧海盆加强混合的对比试验，改 **gendata_zzg**:

```
61 % Specify 3D vertical diffusivity
62 - diff=0.0001*ones([nx,ny,nz]);
63 % diff(1:60,:)=0.01;
64 % for k=1:60
65 %     for j=1:1
66 %         for i=61:80
67 %             diff(i,j,k)=10^(-2-(i-60)*(2)/20);
68 %         end
69 %     end
70 % end
71 - fid=fopen('diffK.bin','w',ieee); fwrite(fid,diff,prec); fclose(fid);
```

这个实验的结果忠实地记录在文件夹：**MITgcm\Results\02_BOX Model\test14**
800000 前后 100000 步平均的结果：



我们将右侧局部强混合海盆放大看看：



千呼万唤始出来，并行计算！

算得太慢啦！

嗯，我们将在“从二维到三维”的例子：[Results\01_tutorial_plume_on_slope\test04](#) 的基础上进行并行运算：

对 **SIZE.h** 和 **SIZE.h.mpi** 进行修改：

```
47      &          sNx = 80,
48      &          sNy = 5,
49      &          OLx = 3,
50      &          OLy = 3,
51      &          nSx = 2,
52      &          nSy = 2,
53      &          nPx = 2,
54      &          nPy = 2,
55      &          Nx = sNx*nSx*nPx,
56      &          Ny = sNy*nSy*nPy,
57      &          Nr = 60)
```

这样 $nPx*nPy = 4$ ，即使用 4 个 cpu

首先我们在使用 **genmake2** 时发现：

```
The platform appears to be: linux_amd64
The possible C compilers found in your path are:
gcc c89 cc c99 mpicc icc
Setting C compiler to: gcc
The possible FORTRAN compilers found in your path are:
g77 f77 pgf77 pgf95 ifort mpif77 gfortran
Setting OPTFILE to: ../../tools/build_options/linux_amd64_g77
using OPTFILE="../../tools/build_options/linux_amd64_g77"
getting AD_OPTFILE information:
using AD_OPTFILE="../../tools/adjoint_options/adjoint_default"
```

即系统是 **linux**，机器的类型是 **amd64**，默认的编译器是 **g77**

于是我们在机群上的 `/public/users/zzg/MITgcm/ZZG_test_mpi/test01/build` 文件夹里兴冲冲的进行并行的变异过程，输入了一下命令：

```
[zzg@model build]$ ../../tools/genmake2 -mods=../code -mpi -of=../../tools/build_options/linux_amd64_ifort_generic_mpi
```

其中 `linux_amd64_ifort_generic_mpi` 是一个我们瞎蒙的控制并行程序平台的文件，然后我们就 `make depend`，不过马上报错：

```
makedepend: warning: mdsio_gl.F (reading EESUPPORT.h, line 175): cannot find include file "mpif.h"
not in mpif.h
not in mpif.h
not in /usr/local/netcdf/include/mpif.h
not in chksum_tiled.F/mpif.h
not in /usr/local/lib/gcc-include/mpif.h
not in /usr/include/mpif.h
not in /usr/lib/gcc/x86_64-redhat-linux/3.4.6/include/mpif.h
../../tools/f90mkdepend >> Makefile
rm -f makedepend.out
```

系统说我找不到 `mpif.h` 这个文件啦，你看怎么办吧，于是我们请来了苗春葆大哥，苗大哥潇洒的输入了如下命令：

```
[zzg@model build]$ which mpif77
/data/dawning/soft/ompil40_intel/bin/mpif77
[zzg@model build]$ mpif77 --show
ifort -I/data/dawning/soft/ompil40_intel/include -pthread -L/data/dawning/soft/ompil40_intel/lib -lmpi_f77 -lmpi -lopen-rte -lopen-pal -Wl,--export-dynamic -lnsl -lutil
[zzg@model build]$
```

确定了 **mpi77** 所在的位置是 `/data/dawning/soft/mpi_140_intel/bin/mpif77`，确定了 **mpi77** 调用的编译器是 `ifort` 编译器，确定了我们需要的 **mpif.h** 文件的位置 `-I/data/dawning/soft/mpi_140_intel/include`

下面呢，我们把 `MITgcm/tools/build_option/linux_amd64_ifort` 这个文件拿过来用一用，放在：

/public/users/zzg/MITgcm/ZZG_test_mpi/test01			
Remote Name	Size	Type	Modified
build		Folder	2010-01-26 10:01:59
code		Folder	2010-01-26 09:22:03
input		Folder	2010-01-27 17:00:09
run		Folder	2010-01-27 16:52:46
linux_amd64_ifort	3,138	文件	2010-01-26 09:56:39

在 **linux_amd64_ifort** 里面我们要做两处改动：第一，将编译器名称全都改为 **mpi** 的编译器：

```
16 # impact appears to be minimal.
17 FC=mpif77
18 F90C=mpif77
19 CC=mpicc
20 LINK='mpif77 -i-dynamic -no-ipo'
```

第二，在程序的最后告诉模式 **mpih.f** 上哪去找：

```
85 fi
86
87 INCLUDES="-I/data/dawning/soft/mpi140_intel/include ${INCLUDES}"
```

改完了，进 **build** 来编译一下：

```
[zzg@node1 build]$ ../../tools/genmake2 -mods=../code -mpi -of=../linux_amd64_ifort
```

然后 **make depend**，**make** 就搞定啦！

剩下的就是运行啦，不过这里也有讲究，需要我们用一个 **linux** 的 **scrip** 文件进行控制，这个文件叫 **dopbs**，我们姑且把它放在 **input** 之中，到时候运行了一并拷贝到 **run** 里面，现在看一下它的内容：

```
1 #!/bin/bash
2
3 #PBS -N mitgcmzzg
4 #PBS -l nodes=node2:ppn=2+node3:ppn=2
5 #PBS -j oe
6
7 NP='wc -l < $PBS_NODEFILE'
8
9 cd $PBS_O_WORKDIR
10 #setenv P4_GLOMEMSIZE 80000000
11 #mpiexec mitgcm
12 mpirun -np $NP -machinefile $PBS_NODEFILE ./mitgcmuv
13
```

到时候使用的进程的名字

这个表示使用第二个节点2个cpu，使用第三个节点2个cpu

这个地方写要运行的可执行文件的名字

运行时，在 **run** 里输入：

```
[zzg@node1 ~]$ qsub dopbs
```

就把计算任务提交上去了，察看运行情况用：

```
[zzg@node1 ~]$ qstat
```

Job id	Name	User	Time Use	S	Queue
368.node1	mitgcmzzg	zzg	05:10:58	R	batch

这个例子的结果忠实地记录在文件夹：**MITgcm/Results/03_mpi_test/test01**

我们观察一下时间使用情况：

T.0000000000.001.001...	96,000	DATA 文件	2010-01-27 16:52:46
T.0000000000.001.001...	178	META 文件	2010-01-27 16:52:46
T.0000000000.001.002...	96,000	DATA 文件	2010-01-27 16:52:46
T.0000000000.001.002...	178	META 文件	2010-01-27 16:52:46
T.0000016000.004.003...	96,000	DATA 文件	2010-01-27 18:31:23
T.0000016000.004.003...	178	META 文件	2010-01-27 18:31:23
T.0000016000.004.004...	96,000	DATA 文件	2010-01-27 18:31:23
T.0000016000.004.004...	178	META 文件	2010-01-27 18:31:23

我们可以得出：

4 个 CPU 运算 320*40*60 的网格，一分钟大约能运算 160 步

比原来 1 个 cpu 的计算效率提高了 4 倍多一点点

在这个算例中我们每个 cpu 使用了四个线程，即 $nSx * nSy = 4$ ，我们再展开一个算例，将线程数缩小为 1 个，看一看有什么影响，修改 **SIZE.h** 和 **SIZE.h.mpi**：

```
47      &          sNx = 160,
48      &          sNy = 10,
49      &          OLx = 3,
50      &          OLy = 3,
51      &          nSx = 1,
52      &          nSy = 1,
53      &          nPx = 2,
54      &          nPy = 2,
55      &          Nx = sNx*nSx*nPx,
56      &          Ny = sNy*nSy*nPy,
57      &          Nr = 60)
```

这个例子的结果忠实地记录在文件夹：**MITgcm\Results\03_mpi_test\test02**

我们观察一下时间使用情况：

```
T.0000016000.002.002...      178  META 文件   2010-01-31 12:11:08
T.init                        3,072,000  INIT 文件   2010-01-31 10:27:19
```

我们发现这对计算效率几乎没有影响。

察看节点使用情况：

```
[xzy@node1 ~]$ pbsnodes 察看节点使用
node2
state = job-exclusive,busy
np = 8
ntype = cluster
jobs = 0/450.model, 1/450.model, 2/450.model, 3/450.model, 4/450.model, 5/450.model, 6/450.model, 7/450.model
status = opsys=linux,uname=linux node2 2.6.9-67.ELsmp #1 SMP Wed Nov 7 13:56:44 EST 2007 x86_64,sessions=13959,nsessions=1,musers=1,idletime=3685352,totmem=26591656kb
165112kb,ncpus=8,loadave=10.07,netload=41412506088557,state=busy,jobs=450.model,varattr=,rectime=1267496207

node3
state = job-exclusive
np = 8
ntype = cluster
jobs = 0/448.model, 1/448.model, 2/448.model, 3/448.model, 4/448.model, 5/448.model, 6/448.model, 7/448.model
status = opsys=linux,uname=linux node3 2.6.9-67.ELsmp #1 SMP Wed Nov 7 13:56:44 EST 2007 x86_64,sessions=30779,31358,31361,31385,31409,31433,31457,31481,31505,nsessio
,totmem=26591656kb,availmem=23855356kb,physmem=8165112kb,ncpus=8,loadave=5.61,netload=45631383234010,state=free,jobs=448.model,varattr=,rectime=1267496223
```

删除进程：

qdel 进程号

怎么维持一个海盆的层结？——回转寿司！

其实，做这个模式要解决的问题就是南海混合驱动 overflow 的问题，需要在模式中维持太平洋海盆层结不变，如果我们用一般模拟热盐环流的方法生成大洋层结，即模拟大洋水团生成，将需要积分 1000 年以上，我们受不了。

所以我就想，能不能在一个海盆中（先是二维）两侧的开边界上设定上我们需要的层结，然后让一个边界进水，一个边界出水，两个边界的流速和温盐剖面都一样，这样在中间的海盆中应该能形成我们需要的层结，就像回转寿司一样，吃完了又转过来了，就补充上了，这样能保证顾客一直有寿司吃。下面的这个实验就是要实施这个想法：

在开边界算例 `test12` 的基础上进行修改，这个例子的信息记录在 `Results\02_BOX Model\test31`

首先我们准备舍弃上边界加热冷却的强迫，改用上边界温度边界条件，即使用舍弃第二类温度边界条件，改用第一类温度边界条件，首先对 `data` 进行改动，加入温度上边界条件：

```
# Input datasets
&PARMOS
  bathyFile='topog.slope',
  hydrogThetaFile='T.init',
  surfQfile='Qnet.forcing',
  thetaClimFile='SST.bin', 海表面温度边界条件由文件SST.bin指定
  diffKrFile='diffK.bin'
#hydrogThetaFile='T.pickup',
#uVelInitFile='U.pickup',
#pSurfInitFile='Eta.pickup',
&

dumpFreq=20000.0,
taveFreq=100000.0,
tauThetaClimRelax=100000.0, 指定温度上边界条件的松弛时间
monitorFreq=1000.,
```

而后对 `gendata_zzg.m` 进行修改：

取消加热

```
38 % Heat Flux
39 - Q0=0;
40 - Q=-Q0.*ones([nx,ny]);
```

引入海表面温度边界条件：

```
48 % Surface Temperature
49 - t0=0.9666666666666667;
50 - sst=t0.*ones([nx,ny]);
51 - fid=fopen('SST.bin','w',ieee); fwrite(fid,sst,prec); fclose(fid);
```

将铅直混合设为 0：

```
53 % Specify 3D vertical diffusivity
54 - diff=0.0000*ones([nx,ny,nz]);
```

地形设为平底:

```
66 % Sloping channel
67 % tanh function for slope
68
69 - d=0.0*rand([nx,ny]);
70 - h=0.0*rand([nx,ny]);
```

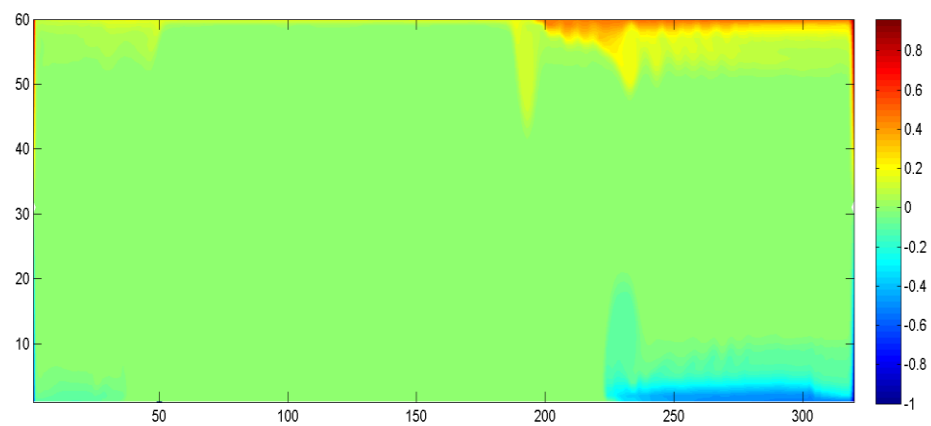
东西边界条件设为一样的, 从东边界进水, 从西边界出水:

```
87 % get the open boudary
88 - uEast=-0.01*ones([ny,nz,cir]);
89 - fid=fopen('uEast.bin','w',ieee); fwrite(fid,uEast,prec); fclose(fid);
90
91 - tEast=0.0*ones([ny,nz,cir]);
92 - for i=1:nz
93 -     tEast(i)=1.0-2.0/60*i;
94 - end
95 - clear i
96 - fid=fopen('tEast.bin','w',ieee); fwrite(fid,tEast,prec); fclose(fid);
97
98 - tWest=tEast;
99 - fid=fopen('tWest.bin','w',ieee); fwrite(fid,tEast,prec); fclose(fid);
100
101 - uWest=-0.01*ones([ny,nz,cir]);
102 - fid=fopen('uWest.bin','w',ieee); fwrite(fid,uEast,prec); fclose(fid);
```

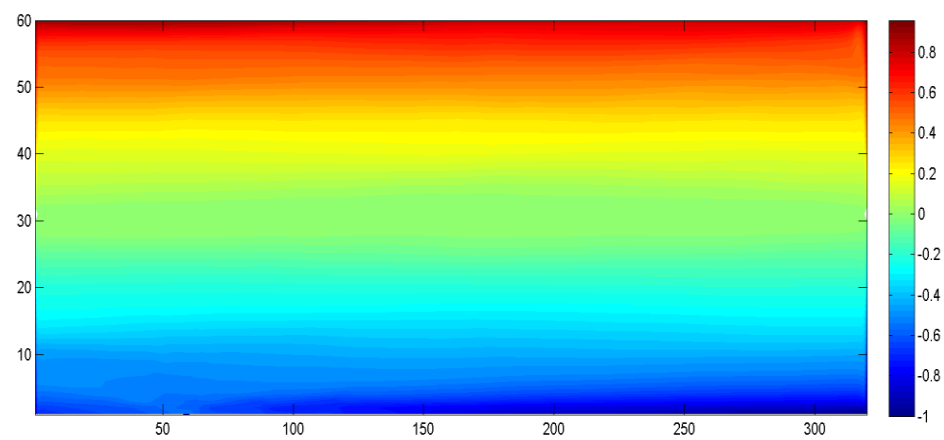
对应的修改 **data.obcs** 文件:

```
1 # Open-boundaries
2 &OBCS_PARM01
3 OB_Ieast=1*-1,
4 OB_Iwest=1*1,
5 useOBCSbalance=.TRUE.,
6 useOBCSprescribe=.TRUE.,
7 useOBCSsponge=.TRUE.,
8 OBEuFile='uEast.bin',
9 OBETFile='tEast.bin',
10 OBWuFile='uWest.bin',
11 OBWTFile='tWest.bin',
12 &
13
14 # *****
15 # Sponge Layer Parameters.
16 # *****
17 &OBCS_PARM03
18 Urelaxobcsinner = 100000.0,
19 Urelaxobcsbound = 100000.0,
20 #Vrelaxobcsinner = 270000.0,
21 #Vrelaxobcsbound = 90000.0,
22 spongeThickness = 20,
23 &
24 # end of the parameters.
25
```

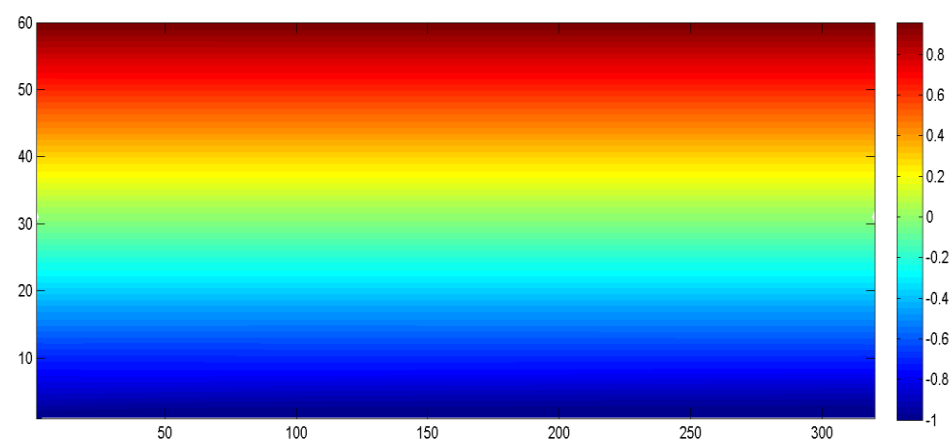

看一下结果：
2000 步的温度



360000 步的温度



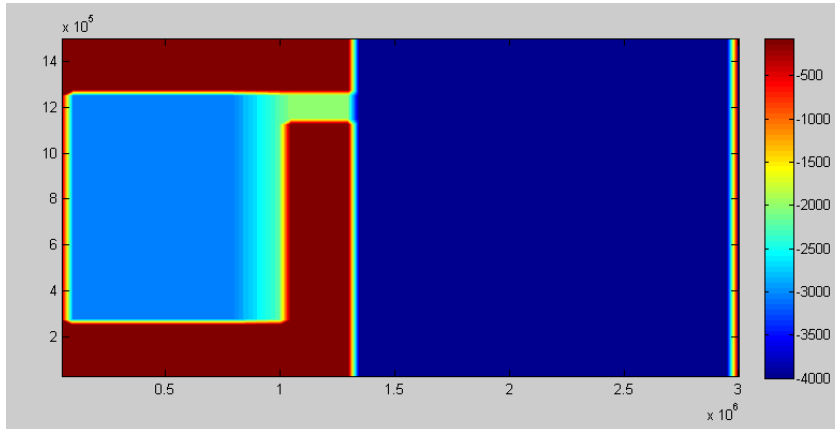
2000000 步的温度（已经平衡了）



看来我们的回转寿司法对于 2 维情况是适用的

终于出现了，第一关关底！——三维盒箱！

我们至此终于获得了所需的所有技术：并行技术、局地加强混合的技术、维持层结不变的技术，所以准备开始迎接第一关关底——三维盒箱模型的计算。区域设置大致如下图：



左侧代表南海（3000m）；右侧代表太平洋海盆（4000m），太平洋海盆中的层结应该保持不变，实现这一点使用回转寿司法（这里使用南北边界，南进北出），中间的通道代表吕宋海峡（2000m），在南海一侧还有一个地形过渡的斜坡，用 5 个格点从 2000m 变到 3000m。

在第一个算例中我们先不加强混合，只测试层结能不能得到维持，这个算例的结果保存在 [Results\02_BOX Model\test32](#)。

首先，我们需要的吕宋海峡以东太平洋的层结（在这里贯彻田老师：宜粗不宜细的方针，不考虑盐度），所以下载 Levitus annual potential temperature （注意到 MITgcm 使用的是位温），数据文件为 [Results\02_BOX Model\test32\prepare\levitus_annual_potential_temperature.cdf](#)，我编写了一个程序，将数据提取出来，并使用 124 E 21N 的位温剖面，保存在 [PonTem.mat](#) 文件中。将 [PonTem.mat](#) 拷贝到 [Results\02_BOX Model\test32\input](#) 文件夹中。

我们准备在 [test25](#) 算例的基础上进行修改，获得这个算例。

1、修改 [code](#) 中的 [SIZE.h](#) 和 [SIZE.h.mpi](#) 文件，两个文件内容修改为一样的。

```
47      &      sNx = 10,
48      &      sNy = 15,
49      &      OLx = 3,
50      &      OLy = 3,
51      &      nSx = 1,
52      &      nSy = 1,
53      &      nPx = 6,
54      &      nPy = 4,
55      &      Nx = sNx*nSx*nPx,
56      &      Ny = sNy*nSy*nPy,
57      &      Nr = 40)
```

可以看到我们使用了 24 个 cpu，在 [input](#) 文件夹中 [dopbs](#) 文件作对应的修改。

```
#!/bin/bash

#PBS -N 3D_Box_01
#PBS -l nodes=node3:ppn=8+node8:ppn=16
#PBS -j oe

NP=`wc -l < $PBS_NODEFILE`

cd $PBS_O_WORKDIR
#setenv P4_GLOBMEMSIZE 80000000
#mpiexec mitgcm
mpirun -np $NP -machinefile $PBS_NODEFILE ./mitgcmuv
```

2、修改 data

取消旋转效应:

```
f0=0.e-5,  
beta=0.E-11,
```

将状态方程设定为真实的状态方程:

```
tRef=40*20.,  
sRef=40*35.,
```

 修改参考温度、参考盐度

```
rhonil=1035.,  
rhoConstFresh=1000.,  
eosType = 'JMD95Z',
```

 设定参考密度、淡水的参考密度、状态方程类型

去掉 tAlpha、sBeta 两项

铅直分辨率为 100 m:

```
delZ=40*100.0,
```

加入控制温度上边界条件的参数:

```
thetaClimFile='SST.bin',  
tauThetaClimRelax=12000000.0,
```

为保持计算稳定, 将时间步长降到 120 s:

```
deltaT=120.0,
```

3、加入边界条件的控制

MITgcm 自己带了好多算例, 其中有一个叫 exp4, 将其中 code/OBCS_OPTION.h 文件拷贝到 test32/code 中, 并对 OBCS_OPTION.h 作如下修改:

```
C Enable individual open boundaries  
#define ALLOW_OBCS_NORTH  
#define ALLOW_OBCS_SOUTH  
#undef ALLOW_OBCS_EAST  
#undef ALLOW_OBCS_WEST  
  
C This includes hooks to sponge layer treatment of uvel, vvel  
#define ALLOW_OBCS_SPONGE
```

同时, 将 test31\input 中的 data.obcs 和 data.pkg 拷贝过来, 放在 input 中, 修改 data.obcs:

```
1 # Open-boundaries  
2 &OBCS_PARM01  
3 OB_Jnorth=60*-1,  
4 OB_Jsouth=60*1,  
5 useOBCSsprescribe=.TRUE.,  
6 useOBCSsponge=.TRUE.,  
7 OBNvFile = 'vNorth.bin',  
8 OBNTFile = 'tNorth.bin',  
9 OBNsFile = 'sNorth.bin',  
10 OBSvFile = 'vSouth.bin',  
11 OBStFile = 'tSouth.bin',  
12 OBSsFile = 'sSouth.bin',  
13 &  
14  
15 # *****  
16 # Sponge Layer Parameters.  
17 # *****  
18 &OBCS_PARM03  
19 #Urelaxobcsinner = 100000.0,  
20 #Urelaxobcsbound = 100000.0,  
21 Vrelaxobcsinner = 100000.0,  
22 Vrelaxobcsbound = 100000.0,  
23 spongeThickness = 6,  
24 &  
25 # end of the parameters.
```

注意这里海绵层的厚度为 6, 为的是在南海的南北两侧, 海绵层不落在海盆中。

4、修改 **gendata_zzg.m** :

读入太平洋一侧的位温剖面:

```
1 - clear
2 - load PonTem.mat
```

设置网格数:

```
11 - nx=60; % x方向的网格数
12 - ny=60; % y方向的网格数
13 - nz=40; % z方向的网格数
14 - cir=1; % 边界层循环周期
```

设置 x 方向网格宽度 50km, y 方向上网格宽度 25km:

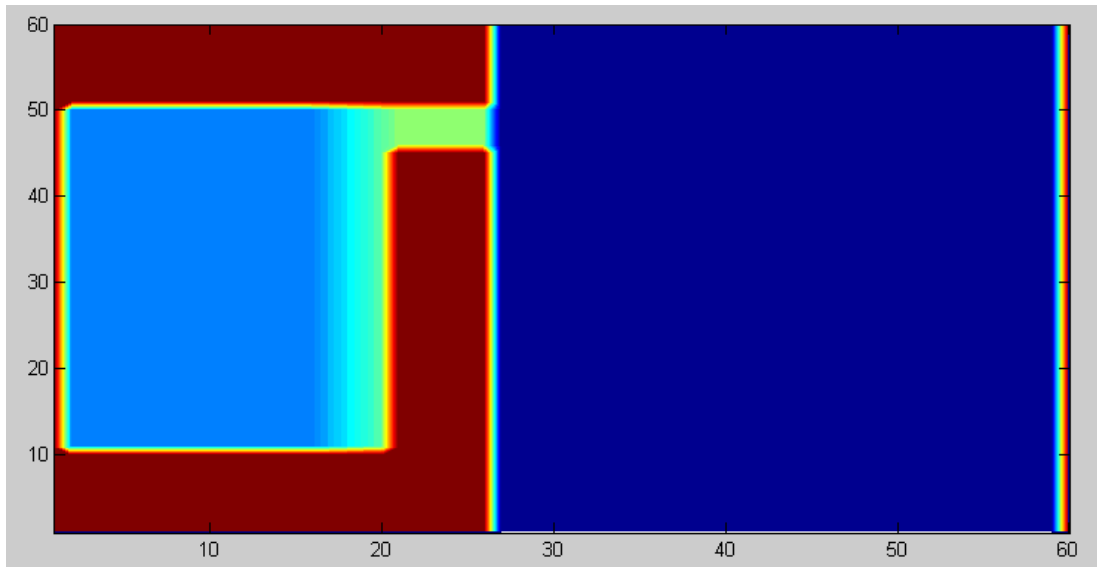
```
18 % x-grid
19 - dx=50000.0.*ones(nx,1);
20 - fid=fopen('dx.bin','w',ieee); fwrite(fid,dx,prec); fclose(fid);
21 % y-grid
22 - dy=25000.0.*ones(ny,1);
23 - fid=fopen('dy.bin','w',ieee); fwrite(fid,dy,prec); fclose(fid);
```

设置 z 方向上的网格:

```
14 % Nominal depth of model (meters)
15 - H=4000.0; % z方向上的区域厚度
25 - dz=H/nz; % z方向上的网格步长
```

设置地形

```
44 %-----get topography-----
45
46 - d=0.0*ones([nx,ny]);
47
48 - d(1:16,:)= -3000.0;
49 - d(17,:)= -2800.0;
50 - d(18,:)= -2600.0;
51 - d(19,:)= -2400.0;
52 - d(20,:)= -2200.0;
53 - d(21:26,46:50)= -2000.0;
54 - d(27:60,:)= -4000.0;
55
56 - d(1,:)=0.0;
57 - d(nx,:)=0.0;
58 - d(1:20,1:10)=0.0;
59 - d(1:20,(ny-9):ny)=0.0;
60 - fid=fopen('topog.slope','w',ieee); fwrite(fid,d,prec); fclose(fid);
```



取消加热:

```

61 %----- Heat Flux-----
62
63 - Q0=0; % 热通量的参考值, 正值为cooling
64 - Q=-Q0.*ones([nx,ny]);
65 - fid=fopen('Qnet.forcing','w',ieee); fwrite(fid,Q,prec); fclose(fid); % 将热通量数据写入文件

```

设置初始层结为均匀 10 度。

```

73 %----- Initial Stratification -----
74 - t=10.0+0.01*rand([nx,ny,nz]);
75 % for i=1:nx
76 %     for j=1:ny
77 %         t(i,j,:)=PonTem;
78 %     end
79 % end
80 % clear i j
81 % t(1:20,:,21:40)=PonTem(20);
82 - fid=fopen('T.init','w',ieee); fwrite(fid,t,prec); fclose(fid);

```

将铅直混合系数到处设为 0 :

```

83 %----- Specify 3D vertical diffusivity-----
84 - diff=0.0*ones([nx,ny,nz]);

```

设置太平洋开边界条件，位温剖面为 levitus，盐度为均匀 35，速度为均匀 5cm/s 向北：

我们设计开边界条件为：

```
98 %----- get the open boudary -----
99 - vNorth=0.0*ones([nx,nz,cir]);
100 - vNorth(27:59,:)=0.05;
101 - fid=fopen('vNorth.bin','w',ieee); fwrite(fid,vNorth,prec); fclose(fid);
102
103 - tNorth=0.0*ones([nx,nz,cir]);
104 - for i=1:nx
105 -     tNorth(i,:)=PonTem;
106 - end
107 - clear i
108 - fid=fopen('tNorth.bin','w',ieee); fwrite(fid,tNorth,prec); fclose(fid);
109
110 - sNorth=35.0*ones([nx,nz,cir]);
111 - fid=fopen('sNorth.bin','w',ieee); fwrite(fid,sNorth,prec); fclose(fid);
112
113 - vSouth=0.0*ones([nx,nz,cir]);
114 - vSouth(27:59,:)=0.05;
115 - fid=fopen('vSouth.bin','w',ieee); fwrite(fid,vSouth,prec); fclose(fid);
116
117 - tSouth=0.0*ones([nx,nz,cir]);
118 - for i=1:nx
119 -     tSouth(i,:)=PonTem;
120 - end
121 - clear i
122 - fid=fopen('tSouth.bin','w',ieee); fwrite(fid,tSouth,prec); fclose(fid);
123
```

事实上，最终证明在边界上加流速是不成功的，故我们，并且初始层结我们将采用太平洋一侧的层结为 levitus,在南海一侧层结为 2000 米以上与太平洋一致，在 2000 米以下为均匀层结：

设置层结：

```
73 %----- Initial Stratification -----
74 - t=0.0.*rand([nx,ny,nz]);
75 - for i=1:nx
76 -     for j=1:ny
77 -         t(i,j,:)=PonTem;
78 -     end
79 - end
80 - clear i j
81 - t(1:20,:,21:40)=PonTem(20);
82 - fid=fopen('T.init','w',ieee); fwrite(fid,t,prec); fclose(fid);
```

设置混合一无混合

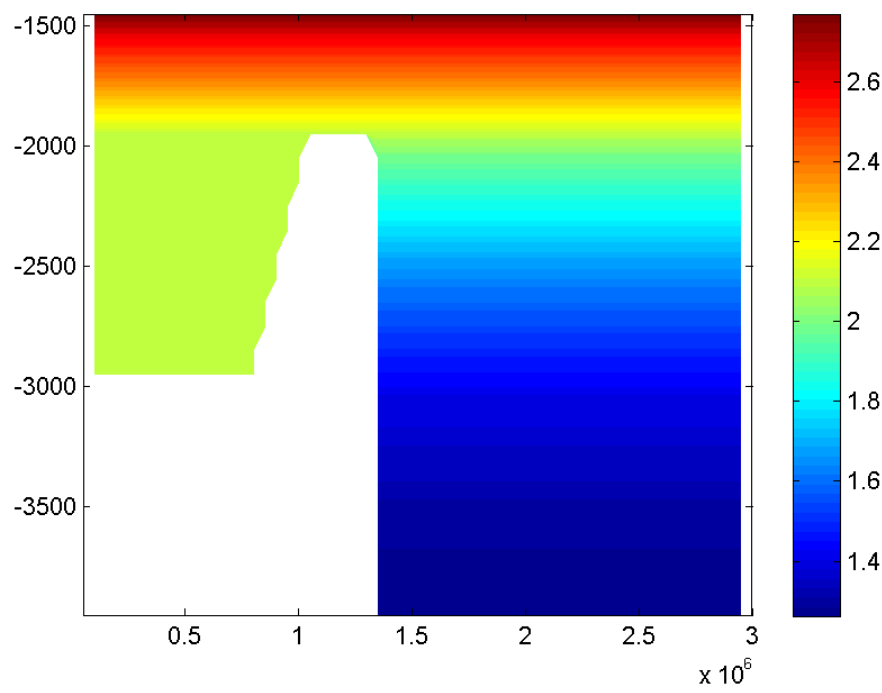
```
84 %----- Specify 3D vertical diffusivity-----
85 - diff=0.0*ones([nx,ny,nz]);
86 - fid=fopen('diffK.bin','w',ieee); fwrite(fid,diff,prec); fclose(fid);
```

设置边界条件

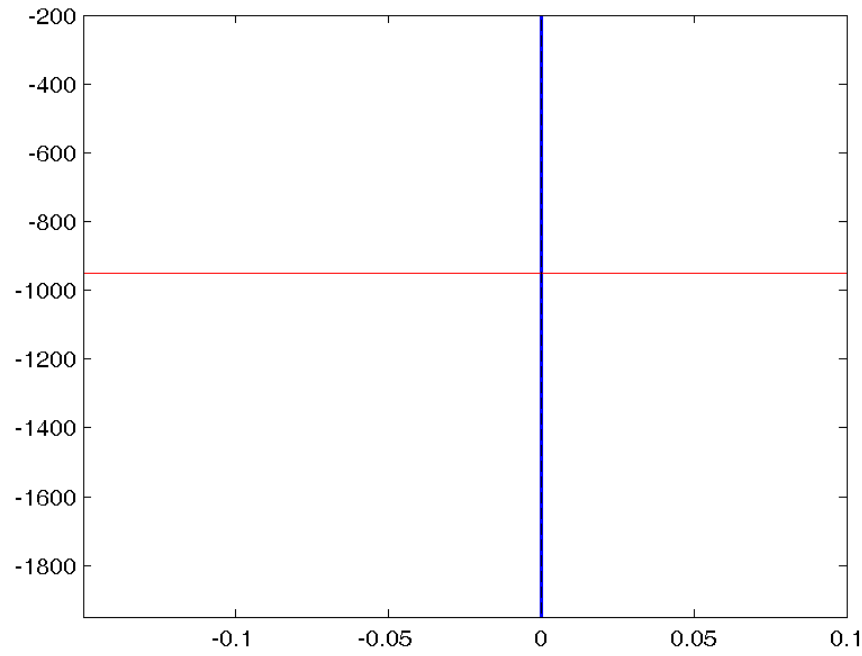
```
88 %----- get the open boudary -----
89 - vNorth=0.0*ones([nx,nz,cir]);
90 %vNorth(27:59,:)=0.05;
91 - fid=fopen('vNorth.bin','w',ieee); fwrite(fid,vNorth,prec); fclose(fid);
92
93 - tNorth=0.0*ones([nx,nz,cir]);
94 - for i=1:nx
95 -     tNorth(i,:)=PonTem;
96 - end
97 - clear i
98 - fid=fopen('tNorth.bin','w',ieee); fwrite(fid,tNorth,prec); fclose(fid);
99
100 - sNorth=35.0*ones([nx,nz,cir]);
101 - fid=fopen('sNorth.bin','w',ieee); fwrite(fid,sNorth,prec); fclose(fid);
102
103 - vSouth=0.0*ones([nx,nz,cir]);
104 %vSouth(27:59,:)=0.05;
105 - fid=fopen('vSouth.bin','w',ieee); fwrite(fid,vSouth,prec); fclose(fid);
106
107 - tSouth=0.0*ones([nx,nz,cir]);
108 - for i=1:nx
109 -     tSouth(i,:)=PonTem;
110 - end
111 - clear i
112 - fid=fopen('tSouth.bin','w',ieee); fwrite(fid,tSouth,prec); fclose(fid);
113
114 - sSouth=35.0*ones([nx,nz,cir]);
115 - fid=fopen('sSouth.bin','w',ieee); fwrite(fid,sSouth,prec); fclose(fid);
```

我们看到这种设置时间步进之后是稳定的：

东西向温度分布：



海峡处的速度剖面一无速度：



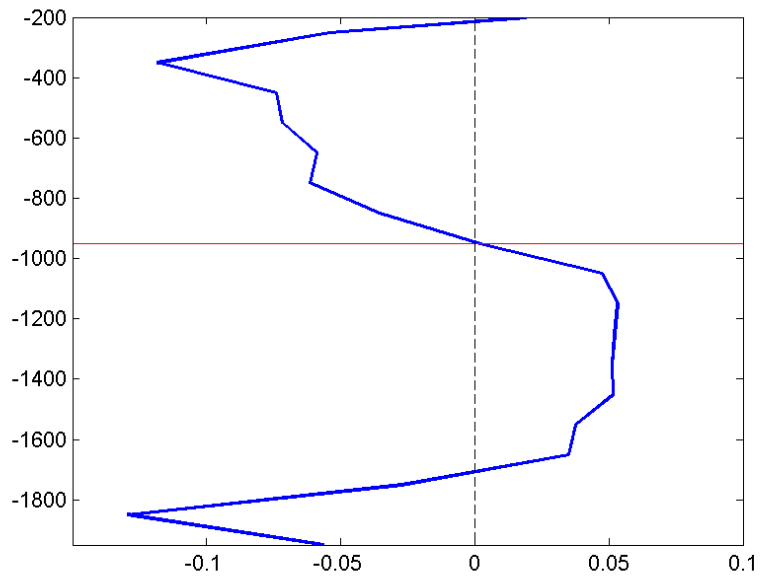
我们现在看一个将南海 1000 米以下混合加到 10^{-3}ms^{-2} 强度的实验结果：

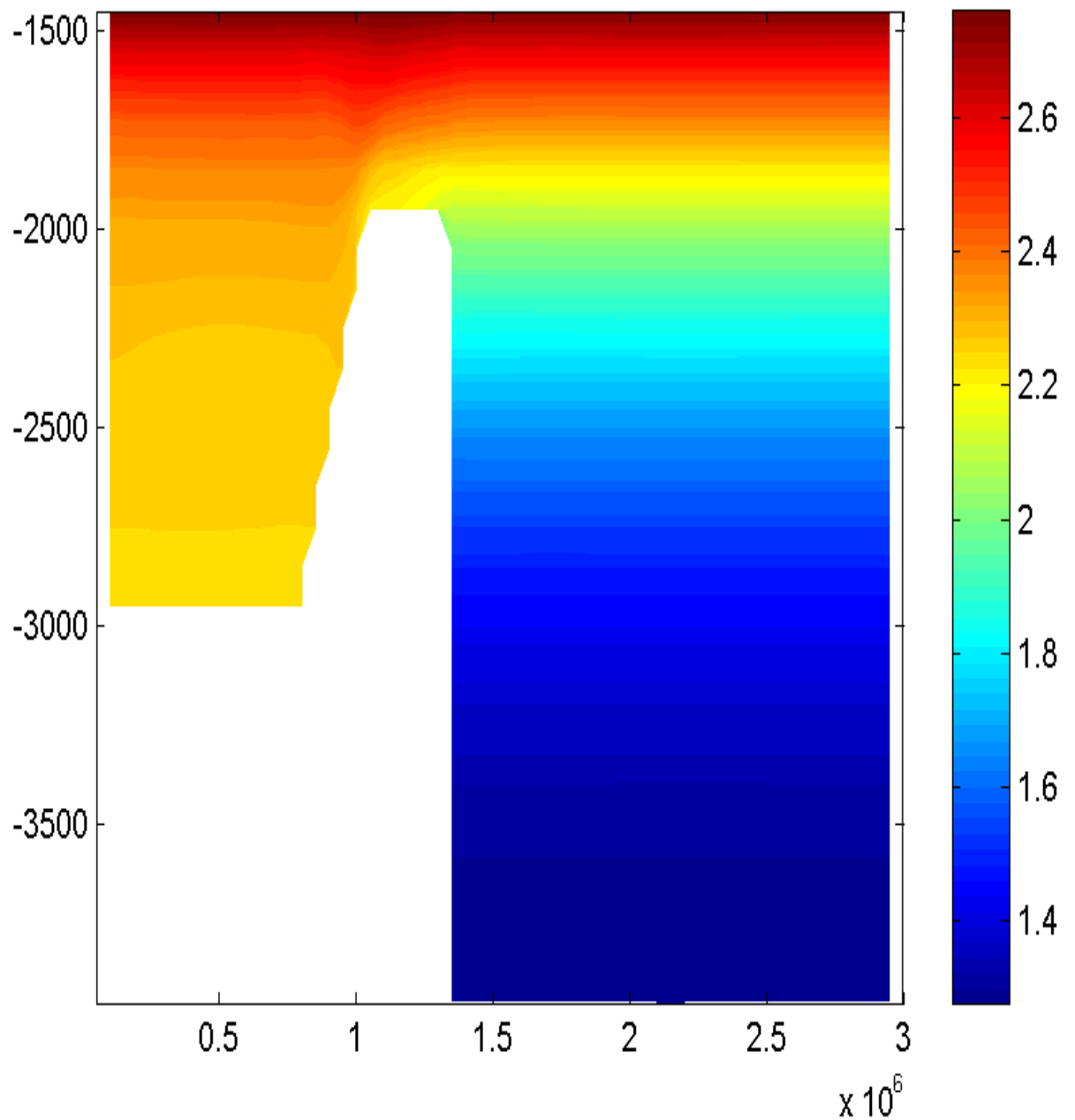
这个例子的信息记录在 [Results\02_BOX Model\test33](#)

首先看修改 **gendata_zzg.m**：

```
84 %----- Specify 3D vertical diffusivity-----  
85 - diff=0.0*ones([nx,ny,nz]);  
86 - diff(1:20,10:50,10:40)=0.001;  
87 - fid=fopen('diffK.bin','w',ieee); fwrite(fid,diff,prec); fclose(fid);
```

再看结果：





但是我们现在其实十分尴尬，我们的时间步长是 120 秒，对于需要几份上百年的我们是很悲剧的事情，向他人求助的结果：

- 1、 打开非静力模块以后时间步长的要求就是会变得很短。
- 2、 在 `data` 文件中有一个和非静力模块下时间步进有关的参数需要调节，在我们原来的算例中：

```

37 # Elliptic solver parameters
38 &PARM02
39 cg2dMaxIters=300,
40 cg2dTargetResidual=1.E-13,
41 cg3dMaxIters=20,
42 cg3dTargetResidual=1.E-8,

```

但经过他人测试，**cg3dMaxIters** 这个参数在使用非静力模块时应该调到 400 左右。这样虽然会延长一次时间步进所耗的时间，但会使稳定性有所增加，但时间步长可以调长仍保持稳定。仍需要测试！

（这个方法证明无效）

下面我介绍一个管用的方法：

把 data 中的

```

viscAh=0.,
viscA4=0.,

```

替换成：

```

viscAh=0.,
viscAhGrid=0.,
viscC2Leith=0.,
viscA4=0.,
useFullLeith=.TRUE.
viscC4Leith=2.0,
viscC4Leithd=2.0,

```

（原理：For horizontal viscosity, you can also try the Leith scheme, which attempts to choose a viscosity that is just right for damping grid scale noise while maintaining a realistic energy cascade that extends as close as possible to your grid's Nyquist wavenumber:）

成功地将时间步长由 120s 提高到 600s，计算 10000 步用时 768 秒

test33 计算 100000 用时 7200 秒，故计算一步的时间基本没变，计算效率提高约 5 倍。

这个算例纪录在：MITgcm\Results\02_BOX Model\test34\z001_speed test