```
setenv FRCGRD $LNHOME/data/frc/frc.t21l5zm.topo.grd       # topo.  forcing
```

After you edit the script as appropriate, just run it.

```
%> cd $LNHOME/model/sh/wvfrc.topo
%> wvfrc.t21l5.topo.csh
```

The procedure to convert the GrADS formatted forcing file to the spectral coefficients, you can refer to section 3.3.3. The SWM response to the orographic forcing inthe above example is shown in Fig. 3.4, which shows the 300 hPa wind and 500 hPa height response quite similar to the results in Nigam et al. (1988).

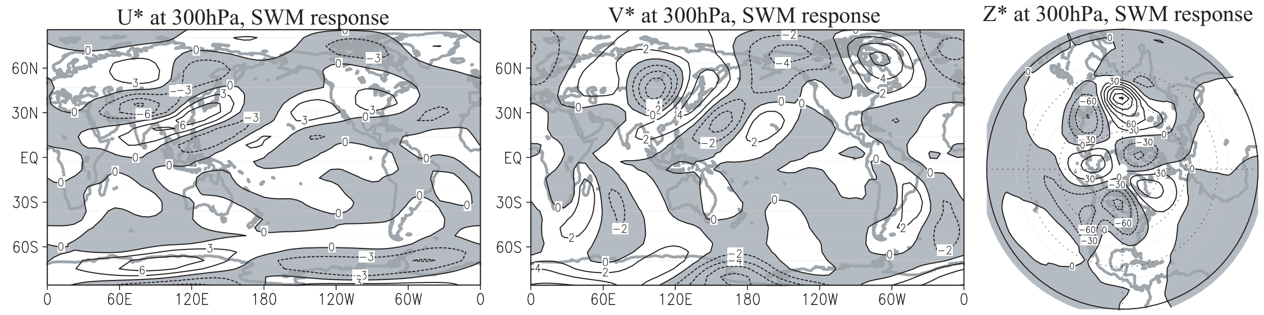
U* at 300hPa, SWM response
V* at 300hPa, SWM response
Z* at 300hPa, SWM response
60N
30N
EQ
30S
60S
0
60E
120E
180
120W
60W

Figure 3.4 Example of the T21L5 steady stationary wave response to the orographic forcing. Shown are the zonal (left) and meridional (middle) winds at 300hPa and geopotential height at 500 hPa (right) all associated with the stationary eddies. Negative values are shaded.

3.4 Accelerated iterative method (AIM)

A simple way for solving the steady response under zonally varying basic state is the time integration (section 3.1). But the time integration is not quite efficient, so that some attempts have been made to solve the steady problems with help of advanced algorithms: for example, the so-called out-of-core solver (Branstator 1992), and parallelization of the LBM code (DeWeaver and Nigam 2000b). We have recently proposed another efficient way, referred to as the accelerated iterative method (AIM), which is shown to be one-order faster than the time integration approach. The mathematical principle is described in Watanabe et al. (2005), so in this document only the practical flows of computation is explained.

AIM is a kind of combined scheme of the relaxation and direct methods. The first part is almost the same as the SWM described in section 3.3, namely, the block matrices for each zonal wavenumber are calculated. After they are inverted by the direct method, the second part of AIM is to correct the first guess of the SWM solution by referring to the non-zonal part of the basic state. The control parameter of the AIM scheme is γ, which is denoted as the acceleration factor. The forced solution is iteratively calculated, and the solution is thought to be converged when the measure of convergence, the normalized differential norm λ, becomes smaller than a specified value.

3.4.1 Making model binary

Choose `'PROJECT = aim'` in `$LNHOME/Lmake.inc`. Although now you can choose higher resolution, let's consider to solve the sample matrix as in 3.2.1, namely, a T21L5 model. For AIM, model option is the conventional one:

```
###### options for model ###################

### dry AIM
MODELOPT = -DOPT_CLASSIC
```

Then, make the model executable file.

```
%> cd $LNHOME/model/src
%> make clean.special
%> make lbm
```

This will create an executable file `lbm2.t21ml5caim`.

3.4.2 Computing linear operator matrix

A sample script for the AIM is `$LNHOME/model/sh/aim/aimbc.l5.classic.csh`. As in the previous sections, let us copy it to `aimbc.l5.test.csh`, and edit lines if necessary. Unlike the previous sections, this shell script is to carry out all the procedures, so that the file names of the linear operator matrix and its inverse are specified here:

```
setenv MATFILE $DIR/mat/MATPWM.t21l5.zm-r.dat         # Linear matrix
setenv MATIFILE $DIR/mat/MATINV.t21l5.zm-r.dat        # Linear inv.  matrix
```

The input data for making the matrix are the 3D basic state, which will be internally truncated at $m = 0$, and topography (only requested for $\sigma \rightarrow$ pressure transform). In preparing the block matrices, set parameters for horizontal diffusion and linear drag in the first block of the script (a part beginning from `=== Making Matrix Ls ===`), and furthermore set

```
setenv OMKMATL   TRUE
setenv OMKMATR   FALSE
setenv OMKINV    FALSE
setenv OAIM      FALSE
```

then run the script

```
%> cd $LNHOME/model/sh/aim
%> aimbc.l5.test.csh
```

It should be noted that the linear operator matrix is not created at this step, instead a group of column vectors is generated.

3.4.3 Acceleration matrix and inverse

To ensure besides achieve faster convergence, we need to add an acceleration matrix to the (SWM) linear operator matrix. This procedure is fast and may need to be repeated several time for tuning the acceleration factor, so separated from the previous step. Namely, modify `aimbc.l5.test.csh` as

```
setenv OMKMATL   FALSE
setenv OMKMATR   TRUE
setenv OMKINV    TRUE
setenv OAIM      FALSE
```

and set the acceleration factor γ

```
setenv RFACT 2000     #
```

where the value of γ should be changed following the diffusion parameters employed in the LBM. For the current example, $\gamma = 2000$ is found to be appropriate. After compiling the routines `redist` and `inv`, run the script again.

```
%> cd $LNHOME/solver/util
%> make clean
%> make
%> cd $LNHOME/solver/custom
%> make clean
%> make
%> cd $LNHOME/model/sh/aim
%> aimbc.l5.test.csh
```

which will make an inverse matrix file spacified by `MATIFILE` after a while.

3.4.4 Preparing forcing

A procedure to prepare forcing is exactly the same as in 3.1.2. Note that the GrADS file of the forcing is only used in the AIM script.

3.4.5 Iteration

The final procedure of AIM is the iterative solver using the inverse matrices and steady forcing. Again, set the environmental variables in `aimbc.l5.test.csh`

```
setenv OMKMATL   FALSE
setenv OMKMATR   FALSE
setenv OMKINV    FALSE
setenv OAIM      TRUE
```

and set the maximum number of iteration and threshold for the differential norm λ,

```
setenv MAXITE 3000
setenv ERRMN 1.d-8
```

Before starting iteration, it is suggested to check whether the parameters `&nmdelt` , `&nmhdif` , `&nmdamp` , `&nmzmfct` , and `&nmvdif` specified in the last block in the script (beginning from `AIM`) are exactly the same as those given in the first block, i.e., used in section 3.4.2. Otherwise, AIM does not lead to the correct solution. The input forcing file and output file of the solution are given as

```
setenv FRCFILE   $FRDIR/frc.t21l5.tst.grd     # forcing
setenv RSPFILT   $FRDIR/aim.t21l5.tst.grd     # response
```

Also, if you would like to compare the AIM solution with the true solution that should be calculated by the direct method in advance, you may specify the files of the true response (input) and the RMS error (output),

```
setenv TRUEFILE   $FRDIR/rsp.t21l5.tst.grd     # true response
setenv RMSEFILE   $FRDIR/rmse.t21l5.tst.grd    # RMS error
```

then set the namelist parameter

```
&nmerr oerr=t ...
```

The default output contains the first guess and the steady soloution (obtained at the last iteration step). If you do not need to have the first guess,

```
&nmrsp frsp='$RSPFILT', oinit=f ...
```

or alternatively if you need to have all the solutions at every iteration step,

```
&nmrsp frsp='$RSPFILT', ofdump=t
```

The output file `$RSPFILT` contains 8 variables (see sample `.ctl` file `sample/aim.t42l20.ctl`):

· stream function	[m^2 s^{-1}]
· velocity potential	[m^2 s^{-1}]
· zonal wind	[m s^{-1}]
· meridional wind	[m s^{-1}]
· pressure vertical velocity (ω)	[hPa s^{-1}]
· temperature	[K]
· geopotential height	[m]
· surface pressure	[hPa]

all of which are transformed into pressure corrdinate if you set

```
&nms2p os2p=t
```

in the script `aimbc.l5.tst.csh`.

Now, you are ready to compute the iterative solutions, by running the script again.

```
   %> cd $LNHOME/model/sh/aim
%> aimbc.l5.test.csh
```

The resultant log file, `$FRDIR/SYSOUT.aim`, stores the extent to which the solution approaches convergence,

```
===== ITE: 0 DNORM= 845058.153954282 LAMBDA= 9.99E+35
===== ITE: 1 DNORM= 349007.2598010577 LAMBDA= 1.0
===== ITE: 2 DNORM= 247727.48374276334 LAMBDA= 0.7098061051336692
===== ITE: 3 DNORM= 196303.15673701846 LAMBDA= 0.5624615283043564
........
===== ITE: 1413 DNORM= 698.3571292681396 LAMBDA= 2.0009816691670526E-3
===== ITE: 1414 DNORM= 697.9744875512099 LAMBDA= 1.999885297369091E-3
```

where the number at the right indicates λ. If you set the threshold for λ at larger value (`setenv ERRMN  2.d-3` in the above example), you will finally have the message

```
@@@ AIM CONVERGED AT: 1414
```

otherwise,

```
### AIM NOT CONVERGED AT: 3000
```

It should be emphasized, while AIM is much efficient than other iterative methods, that you need to know the best value of γ before starting iteration; in particular, the solution diverges for γ smaller than the appropriate value. Moreover, the system has to be stable or near neutral. When the AIM solution blows up due to either of the above causes, you can step back to section 3.4.3 (NOT section 3.4.2!), and then change γ or damping parameters to repeat the calculation.

One of the most useful application of AIM is to solve a number of steady response to different forcing, with the identical basic state. An example of such a computation is the hindcast of the wintertime circulation anomalies by solving the steady responses to diabatic forcing and transient eddy forcing provided for individual winter. See `aimbc.t42l20.hindcast.csh` is the sample shell script for the hindcast diagnosis, in which additional namelist parameters at the last block are modified:

```
&nmamat ofmult=t,
        nffst=1,
        nflst=43, ...
```