Reporte WRF

Joao Fabián

Instalación

La instalación del modelo WRF, del sistema de procesamiento WPS, y de los softwares de post-procesamiento, son realizados en distribuciones de Linux. Todas las secuencias y comandos mostrados en este reporte deben realizarse en el terminal de Linux, a no ser que se especifique lo contrario.

Se deben crear dos directorios en la *carpeta de usuario* o *home directory*. Un directorio será llamado *Build_WRF*, en este se montarán los archivos necesarios del modelo *WRF*. El otro directorio será llamado *TESTS*, y en este se montarán archivos que permitirán realizar pruebas para confirmar que el software adicional requerido funciona correctamente.

Software adicional requerido

Es necesario instalar: *gfortran*, *csh*, *m4* y *build-essential.*

```
$ sudo apt-get install gfortran csh m4 build-essential
```

Prueba del entorno del sistema

Se deben tener instalados compiladores de *Fortran* (gfortran), *C++* (gpp) y *C* (gcc), actualizados en sus versiones más recientes.

Es necesario descargar los archivos que servirán para hacer las pruebas.

```
$ cd ~/TESTS
```

```
$ wget http://www2.mmm.ucar.edu/wrf/OnLineTutorial/
   compile_tutorial/tar_files/Fortran_C_tests.tar
$ tar -xvf Fortran_C_tests.tar
```

Son siete pruebas que se encuentran disponibles. Estas se correrán una por una.

1. Prueba para *Fortran* de formato fijo:

```
$ gfortran TEST_1_fortran_only_fixed.f
$ ./a.out
```

   En caso de que la prueba sea exitosa, en el terminal se mostrará:

```
SUCCESS test 1 fortran only fixed format
```

2. Prueba para *Fortran* de formato libre:

```
$ gfortran TEST_2_fortran_only_free.f90
$ ./a.out
```

   En caso de que la prueba sea exitosa, en el terminal se mostrará:

```
SUCCESS test 2 fortran only free format
```

3. Prueba de *C*:

```
$ gcc TEST_3_c_only.c
$ ./a.out
```

   En caso de que la prueba sea exitosa, en el terminal se mostrará:

```
SUCCESS test 3 C only
```

4. Prueba de *Fortran* llamando a una función en *C*:

```
$ gcc -c -m64 TEST_4_fortran+c_c.c
$ gfortran -c -m64 TEST_4_fortran+c_f.f90
$ gfortran -m64 TEST_4_fortran+c_f.o TEST_4_fortran+
   c_c.o
$ ./a.out
```

En caso de que la prueba sea exitosa, en el terminal se mostrará:

```
C function called by Fortran
Values are xx = 2.00 and ii= 1
SUCCESS test 4 fortran calling c
```

5. Prueba de *csh*:

```
$ csh TEST_csh.csh
```

En caso de que la prueba sea exitosa, en el terminal se mostrará:

```
SUCCESS csh test
```

6. Prueba de *Perl*:

```
$ ./TEST_perl.pl
```

En caso de que la prueba sea exitosa, en el terminal se mostrará:

```
SUCCESS perl test
```

7. Prueba de *sh*:

```
$ ./TEST_sh.sh
```

En caso de que la prueba sea exitosa, en el terminal se mostrará:

```
SUCCESS sh test
```

Construcción de librerías

Se crea el directorio *LIBRARIES*:

```
$ mkdir ~/Build_WRF/LIBRARIES
```

Se descargan los archivos comprimidos de las librerías: *MPICH*, *NetCDF*, *JasPer*, *libpng* y *zlib*:

```
$ cd ~/Build_WRF/LIBRARIES
$ wget http://www2.mmm.ucar.edu/wrf/OnLineTutorial/
   compile_tutorial/tar_files/mpich-3.0.4.tar.gz
$ wget http://www2.mmm.ucar.edu/wrf/OnLineTutorial/
   compile_tutorial/tar_files/netcdf-4.1.3.tar.gz
$ http://www2.mmm.ucar.edu/wrf/OnLineTutorial/
   compile_tutorial/tar_files/jasper-1.900.1.tar.gz
$ http://www2.mmm.ucar.edu/wrf/OnLineTutorial/
   compile_tutorial/tar_files/libpng-1.2.50.tar.gz
$ http://www2.mmm.ucar.edu/wrf/OnLineTutorial/
   compile_tutorial/tar_files/zlib-1.2.7.tar.gz
```

Configuración de NetCDF

Se abre el archivo *.bashrc* que se encuentra en el *home directory*:

```
$ sudo nano ~/.bashrc
```

Así, al final del archivo *.bashrc* se deben agregar las líneas:

```
# WRF environment variables
export DIR=/home/lmc/Build_WRF/LIBRARIES
export CC=gcc
export CXX=g++
export FC=gfortran
export CFLAGS=-m64
export F77=gfortran
export FFLAGS=-m64
```

Se corre *.bashrc* y se descomprime el archivo correspondiente a *NetCDF*:

```
$ source ~/.bashrc
$ cd ~/Build_WRF/LIBRARIES
$ tar -zxvf netcdf-4.1.3.tar.gz
```

Se corre el archivo de configuración de *NetCDF*:

```
$ cd ~/Build_WRF/LIBRARIES/netcdf-4.1.3
$ ./configure --prefix=$DIR/netcdf --disable-dap --
   disable-netcdf-4 --disable-shared
$ make
$ make install
```

Se debe modificar otra vez el archivo *.bashrc*:

```
$ sudo nano ~/.bashrc
```

Al final del archivo *bashrc* se deben agregar las siguientes líneas:

```
export PATH=$DIR/netcdf/bin:$PATH
export NETCDF=$DIR/netcdf
```

Se corre *.bashrc*:

```
$ source ~/.bashrc
```

Configuración de MPICH

Se descomprime el archivo correspondiente a *MPICH*:

```
$ cd ~/Build_WRF/LIBRARIES
$ tar -zxvf mpich-3.0.4.tar.gz
```

Se corre el archivo de configuración de MPICH:

```
$ cd ~/Build_WRF/LIBRARIES/mpich-3.0.4
$ ./configure --prefix=$DIR/mpich
$ make
$ make install
```

Se abre el archivo *.bashrc* que se encuentra en el *home directory*:

```
$ sudo nano ~/.bashrc
```

Al final del archivo *.bashrc*, se agrega la línea:

```
export PATH=$DIR/mpich/bin:$PATH
```

Se corre *.bashrc*:

```
$ source ~/.bashrc
```

Configuración de zlib

Se descomprime el archivo correspondiente a *zlib*:

```
$ cd ~/Build_WRF/LIBRARIES
$ tar -zxvf zlib-1.2.7.tar.gz
```

Se abre el archivo *.bashrc* que se encuentra en el *home directory*:

```
$ sudo nano ~/.bashrc
```

Al final del archivo *.bashrc* se agregan las lneas:

```
export LDFLAGS=-L$DIR/grib2/lib
export CPPFLAGS=-I$DIR/grib2/include
```

Se corre *.bashrc*:

```
$ source ~/.bashrc
```

Se corre el archivo de configuración de *zlib*:

```
$ cd ~/Build_WRF/LIBRARIES/zlib-1.2.7
$ ./configure --prefix=$DIR/grib2
$ make
$ make install
```

Configuración de libpng

Se descomprime el archivo correspondiente a *libpng*:

```
$ cd ~/Build_WRF/libraries
$ tar -zxvf libpng-1.2.50.tar.gz
```

Se corre el archivo de configuración de *libpng*:

```
$ cd ~/Build_WRF/LIBRARIES/libpng-1.2.50
$ ./configure --prefix=$DIR/grib2
$ make
$ make install
```

Configuración de JasPer

Se descomprime el archivo correspondiente a *Jasper*

```
$ cd ~/Build_WRF/LIBRARIES
$ tar -zxvf jasper-1.900.1.tar.gz
```

Se corre el archivo de configuración de *JasPer*:

```
$ cd ~/Build_WRF/LIBRARIES/jasper-1.900.1
$ ./configure --prefix=$DIR/grib2
$ make
$ make install
```

Prueba de compatibilidad de librerías con WRF y WPS

Se descargan los archivos de pruebas:

```
$ cd ~/TESTS
$ wget http://www2.mmm.ucar.edu/wrf/OnLineTutorial/
   compile_tutorial/tar_files/Fortran_C_NETCDF_MPI_tests
   .tar
$ tar -xvf Fortran_C_NETCDF_MPI_tests.tar
```

Son dos pruebas, y se realizarán una por una:

1. Prueba de Fortran + C + NetCDF:

```
$ cd ~/TESTS
$ cp ${NETCDF}/include/netcdf.inc .
$ gfortran -c 01_fortran+c+netcdf_f.f
$ gcc -c 01_fortran+c+netcdf_c.c
$ gfortran 01_fortran+c+netcdf_f.o 01_fortran+c+
   netcdf_c.o -L${NETCDF}/lib -lnetcdff -lnetcdf
$ ./a.out
```

   En caso de que la prueba sea exitosa, el terminal mostrará:

```
C function called by Fortran
Values are xx = 2.00 and ii = 1
SUCCESS test 1 fortran + c + netcdf
```

2. Prueba de Fortran + C + NetCDF + MPI:

```
$ cd ~/TESTS
$ cp ${NETCDF}/include/netcdf.inc .
$ mpif90 -c 02_fortran+c+netcdf+mpi_f.f
$ mpicc -c 02_fortran+c+netcdf+mpi_c.c
$ mpif90 02_fortran+c+netcdf+mpi_f.o 02_fortran+c+
   netcdf+mpi_c.o -L${NETCDF}/lib -lnetcdff -lnetcdf
$ mpirun ./a.out
```

   En caso de que la prueba sea exitosa, el terminal mostrará:

```
C function called by Fortran
Values are xx = 2.00 and ii = 1
status =                2
SUCCESS test 2 fortran + c + netcdf + mpi
```

Construcción de WRF V4

Se descarga WRF:

```
$ cd ~/Build_WRF
$ wget http://www2.mmm.ucar.edu/wrf/src/WRFV4.0.TAR.gz
$ tar -zxvf WRFV4.0.TAR.gz
```

Se corre el archivo de configuración de WRF:

```
$ cd ~/Build_WRF/WRF
$ ./configure
$ 34
```

Se escoge la opción **34** debido a que la computadora presenta varios núcleos, y permite trabajar en paralelo, además, se trabajará con *gfortran/gcc*.

Una vez que la configuración se complete, debe haberse generado un archivo llamado *configure.wrf* en el mismo directorio.

Puesto que la configuración está completa, se pasa a compilar, seleccionando el caso de acuerdo al propósito para el que será usado el modelo *WRF*. En este caso, se compila *WRF* para casos reales:

```
$ cd ~/Build_WRF/WRF
$ ./compile em_real >& compile.log &
$ tail -f compile.log
```

Para corroborar que la compilación fue exitosa, se revisa el directorio *WRF*:

```
$ cd ~/Build_WRF/WRF
$ ls -las main/*.exe
```

La compilación se habrá completado correctamente si en el terminal se observan los ejecutables:

```
main/ndown.exe
main/real.exe
main/tc.exe
main/wrf.exe
```

Construcción de WPS V4

El programa *WPS* debe construirse debido a que se busca trabajar con casos reales.

Se descarga WPS:

```
$ cd ~/Build_WRF
$ wget http://www2.mmm.ucar.edu/wrf/src/WPSV4.0.TAR.gz
$ tar -zxvf WPSV4.0.TAR.gz
$ cd WPS
$ ./clean
```

Se abre el archivo *.bashrc*:

```
$ sudo nano ~/.bashrc
```

Al final del archivo se agregan las líneas:

```
export JASPERLIB=$DIR/grib2/lib
export JASPERINC=$DIR/grib2/include
```

Se corre *.bashrc*:

```
$ source ~/.bashrc
```

Se corre el archivo de configuración de *WPS*:

```
$ cd ~/Build_WRF/WPS
$ ./configure
$ 1
```

Se selecciona la opción **1** debido a las características de la computadora, además de estar usando *gfortran*. Si la configuración se completa correctamente, debe aparecer un mensaje en el terminal que terminará con:

```
This installation NetCDF is 64-bit
C compiler is 64-bit
Fortran compiler is 64-bit
```

El sistema *WPS* debe direccionarse hacia las librerías *I/O* de *WRF*, donde se encuentran los ejecutables *metgrid.exe* y *geogrid.exe*:

```
$ cd ~/Build_WRS/WPS
$ sudo nano configure.wps
```

En el archivo, especificar la ruta hacia el directorio WRF:

```
   WRF_DIR          =        ../WRF
```

Se compila *WPS*:

```
$ cd ~/Build_WRS/WPS
$ ./compile >& compile.log&
$ tail -f compile.log
```

Para corroborar que la compilación fue exitosa, revisar el directorio *WPS*:

```
$ cd ~/Build_WRS/WPS
$ ls -las *.exe
```

La compilación será correcta si en el terminal se muestran los ejecutables:

```
geogrid.exe
metgrid.exe
ungrib.exe
```

Para inicializar casos reales, se deben crear la locación física del dominio en el goblo, y la información estática para tal locación. Esta información debe ser descargada:

```
$ cd ~/Build_WRF
$ wget http://www2.mmm.ucar.edu/wrf/src/wps_files/
   geog_10m.tar.gz
$ tar -zxvf geog_10m.tar.gz
$ mv geog WPS_GEOG
```

Se abre el archivo *namelist.WPS*:

```
$ cd ~/Build_WRF/WPS
```

```
$ nano namelist.wps
```

Se modifica el directorio correcto en el archivo *namelist.wps*:

```
geog_data_path=          '/home/lmc/Build_WRF/WPS_GEOG/'
```

Post - Procesamiento

Para poder leer los archivos generados por *WRF* usando el core *ARW*, se usa el programa *ARWPost*, además, se usa *GrADS* para poder hacer un plot de los resultados obtenidos.

Descargar ARWPost:

```
$ cd ~/Build_WRF
$ wget http://www2.mmm.ucar.edu/wrf/src/ARWpost_V3.tar.
   gz
$ tar -zxvf ARWpost_V3.tar.gz
```

Asumiento que ya se configuró NetCDF, se configura ARWpost:

```
$ cd ~/Build_WRF/ARWpost
$ ./configure
$ 3
$ cd src
```

Se abre el archivo *Makefile*:

```
$ cd ~/Build_WRF/ARWpost
$ sudo nano Makefile
```

En el archivo *Makefile* se modifica la línea correspondiente a *ARWpost.exe*:

```
ARWpost.exe: $(OBJS)
                $(FC) $(FFLAGS) $(LDFLAGS) -o $@ $(OBJS)
                     \
                                -L$(NETCDF)/lib -lnetcdf
                                    -lnetcdff -I$(NETCDF
                                   )/include -lnetcdf
```

Además, se abre el archivo *configure.arwp*:

```
$ cd ~/Build_WRF/ARWpost
$ sudo nano configure.arwp
```

En el archivo *configure.awps* se modifican las líneas correspondientes a *CFLAGS* y *CPP*:

```
CFLAGS          =               -fPIC -m64
CPP             =               /lib/cpp -P -traditional
```

Se compila *ARWpost*:

```
$ cd ~/Build_WRF/ARWpost
$ ./compile
```

Se revisan los archivos generados, para comprobar si la compilación se realizó de manera correcta:

```
$ cd ~/Build_WRF/ARWpost
$ ls -ls *.exe
```

La compilación será correcta si el terminal muestra el ejecutable:

```
ARWpost.exe
```

Para poder hacer un plot de los archivos que serán generados a través de ARWpost, es necesario instalar el programa GrADS:

```
$ sudo apt-get install grads
```

Pruebas

Test 1

Geogrid.exe

En el procesamiento con WPS, se modifica el archivo *namelist.wps*:

```
$ cd ~/Build_WRF/WPS
$ sudo nano namelist.wps
```

Así, los parámetros en *namelist.wps* son modificados de acuerdo a:

```
wrf_core = 'ARW',
&share
 max_dom = 1,
 start_date = '2011-09-05_12:00:00','2006-08-16_12
    :00:00',
 end_date   = '2011-09-10_12:00:00','2006-08-16_12
    :00:00',
 interval_seconds = 10800
/

&geogrid
 parent_id         =   1,   1,
 parent_grid_ratio =   1,   3,
 i_parent_start    =   1,  31,
 j_parent_start    =   1,  17,
 e_we              =  15, 112,
 e_sn              =  15,  97,
 geog_data_res     = '2m','2m',
 dx = 3333.33,
 dy = 3333.33,
 map_proj = 'lambert',
 ref_lat   =  45.68,
 ref_lon   = -111.06,
 truelat1  =  30.0,
 truelat2  =  60.0,
 stand_lon = -111.06,
 geog_data_path = '/home/marv/Build_WRF/WPS_GEOG/'
/

&ungrib
 out_format = 'WPS',
 prefix = 'FILE',
/

```

```
&metgrid
 fg_name = 'FILE'
 io_form_metgrid = 2,
/
```

Se genera el ejecutable *geogrid.exe*:

```
$ cd ~/Build_WRF/WPS
$ ./geogrid.exe
```

El terminal indicará el mensaje de confirmación siguiente:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Successful completion of geogrid.        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

ungrib.exe

El dataset usado es el **ds609.2**. Se realiza la descarga de este:

```
$ mkdir ~/Build_WRF/data
$ mkdir ~/Build_WRF/data/AWIP_datasets
$ mkdir ~/Build_WRF/data/AWIP_datasets/ds609.2
$ cd ~/Build_WRF/data/AWIP_datasets/ds609.2
$ wget https://rda.ucar.edu/data/ds609.2/SFanal/2011/
   G43091.SFanal.201109
$ wget https://rda.ucar.edu/data/ds609.2/3Danal/2011/
   G43084.3Danal.201109.16-30
$ wget https://rda.ucar.edu/data/ds609.2/3Danal/2011/
   G43083.3Danal.201109.01-15
$ mv G43091.SFanal.201109 G43091.SFanal.201109.tar
$ mv G43084.3Danal.201109.16-30 G43084.3Danal
   .201109.16-30.tar
$ mv G43083.3Danal.201109.01-15 G43083.3Danal
   .201109.01-15.tar
$ tar -zxvf G43091.SFanal.201109.tar
$ tar -zxvf G43084.3Danal.201109.16-30.tar
$ tar -zxvf G43083.3Danal.201109.01-15.tar
```

Se genera el ejecutable *ungrib.exe*:

```
$ cd ~/Build_WRF/WPS
$ ln -sf ungrib/Variable_Tables/Vtable.AWIP Vtable
$ ./link_grib.csh ~/Build_WRF/AWIP_datasets/ds609
   .2/201109
$ ./ungrib.exe >& ungrib.log
```

metgrid.exe

Se corre el ejecutable *metgrid.exe*:

```
$ cd ~/Build_WRF/WPS
$ ./metgrid.exe
```

El terminal indicará el mensaje de confirmación siguiente:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Successful completion of metgrid.        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

real.exe y wrf.exe

Se entra al directorio *test*:

```
$ cd ~/Build_WRF/WRF/test/em_real
$ ln -sf ~/Build_WRF/WPS/met_em.d01.* .
$ sudo nano namelist.input
```

Los parámetros dentro del archivo *namelist.input* se modifica de la forma:

```
 &time_control
 run_days                            = 3,
 run_hours                           = 0,
 run_minutes                         = 0,
 run_seconds                         = 0,
 start_year                          = 2011, 2000, 2000,
 start_month                         = 09,   01,   01,
```

```
 start_day                           = 06,   24,   24,
 start_hour                          = 12,   12,   12,
 start_minute                        = 00,   00,   00,
 start_second                        = 00,   00,   00,
 end_year                            = 2011, 2000, 2000,
 end_month                           = 09,   01,   01,
 end_day                             = 09,   25,   25,
 end_hour                            = 12,   12,   12,
 end_minute                          = 00,   00,   00,
 end_second                          = 00,   00,   00,
 interval_seconds                    = 10800,
 input_from_file                     = .true.,.true.,.
    true.,
 history_interval                    = 180,  60,   60,
 frames_per_outfile                  = 1000, 1000, 1000,
 restart                             = .false.,
 restart_interval                    = 21600,
 io_form_history                     = 2,
 io_form_restart                     = 2,
 io_form_input                       = 2,
 io_form_boundary                    = 2,
 debug_level                         = 0,
 /

 &domains
 time_step                           = 30,
 time_step_fract_num                 = 0,
 time_step_fract_den                 = 1,
 max_dom                             = 1,
 s_we                                = 1,
 e_we                                = 15,    112,   94,
 s_sn                                = 1,
 e_sn                                = 15,    97,    91,
 s_vert                              = 1,
 e_vert                              = 60,    28,    28,
 p_top_requested                     = 5000,
 num_metgrid_levels                  = 27,
 num_metgrid_soil_levels             = 4,
 dx                                  = 3333.33, 10000,
    3333.33,
```

```
 dy                                  = 3333.33, 10000,
    3333.33,
 grid_id                             = 1,     2,     3,
 parent_id                           = 0,     1,     2,
 i_parent_start                      = 1,     31,    30,
 j_parent_start                      = 1,     17,    30,
 parent_grid_ratio                   = 1,     3,     3,
 parent_time_step_ratio              = 1,     3,     3,
 feedback                            = 1,
 smooth_option                       = 0,
 sfcp_to_sfcp = .true.
 /

 &physics
 physics_suite                       = 'CONUS'
 mp_physics                          = -1,    -1,    -1,
 cu_physics                          = -1,    -1,     0,
 ra_lw_physics                       = -1,    -1,    -1,
 ra_sw_physics                       = -1,    -1,    -1,
 bl_pbl_physics                      = -1,    -1,    -1,
 sf_sfclay_physics                   = -1,    -1,    -1,
 sf_surface_physics                  = -1,    -1,    -1,
 radt                                = 30,    30,    30,
 bldt                                = 0,     0,     0,
 cudt                                = 5,     5,     5,
 icloud                              = 1,
 num_land_cat                        = 21,
 sf_urban_physics                    = 0,     0,     0,
 /

 &fdda
 /

 &dynamics
 hybrid_opt                    = 2,
 w_damping                     = 0,
 diff_opt                      = 1,       1,     1,
 km_opt                        = 4,       4,     4,
 diff_6th_opt                  = 0,       0,     0,
 diff_6th_factor               = 0.12,    0.12,  0.12,
```

```
 base_temp                           = 290.
 damp_opt                            = 3,
 zdamp                               = 5000.,  5000.,  5000.,
 dampcoef                            = 0.2,    0.2,    0.2
 khdif                               = 0,      0,      0,
 kvdif                               = 0,      0,      0,
 non_hydrostatic                     = .true., .true., .true.,
 moist_adv_opt                       = 1,      1,      1,
 scalar_adv_opt                      = 1,      1,      1,
 gwd_opt                             = 1,
 /

 &bdy_control
 spec_bdy_width                            = 5,
 specified                                 = .true.
 /

 &grib2
 /

 &namelist_quilt
 nio_tasks_per_group = 0,
 nio_groups = 1,
 /
```

Para obtener los ejecutables *real.exe* y *wrf.exe*:

```
$ ./real.exe
$ ./wrf.exe >& wrfoutput.log
```

Se generan archivos de salida correspondientes al procesamiento realizado haciendo uso de *WRF*.

ARWpost

Se ingresa al directorio *ARWpost* y se modifica el archivo *namelist.ARWpost*:

```
$ cd ~/Build_WRF/WRF/ARWpost
$ sudo nano namelist.ARWpost
```

El archivo debe modificarse de la siguiente forma:

```
&datetime
 start_date = '2000-01-24_12:00:00',
 end_date   = '2000-01-25_00:00:00',
 interval_seconds = 10800,
 tacc = 0,
 debug_level = 0,
/

&io
 input_root_name = '~/Build_WRF/WRF/test/em_real/
    wrfout_d01'
 output_root_name = './test'
 plot = 'all_list'
 fields = 'height,pressure,tk,tc'
 mercator_defs = .true.
/
 split_output = .true.
 frames_per_outfile = 2


 plot = 'all'
 plot = 'list'
 plot = 'all_list'
! Below is a list of all available diagnostics
 fields = 'height,geopt,theta,tc,tk,td,td2,rh,rh2,umet,
    vmet,pressure,u10m,v10m,wdir,wspd,wd10,ws10,slp,
    mcape,mcin,lcl,lfc,cape,cin,dbz,max_dbz,clfr'


&interp
 interp_method = 0,
 interp_levels =
    1000.,950.,900.,850.,800.,750.,700.,650.,600.,550.,500.,450.,400.,350

/
extrapolate = .true.

 interp_method = 0,     ! 0 is model levels, -1 is nice
```

```
    height levels, 1 is user specified pressure/height
34
35  interp_levels =
       1000.,950.,900.,850.,800.,750.,700.,650.,600.,550.,
36  500.,450.,400.,350.,300.,250.,200.,150.,100.,
37  interp_levels = 0.25, 0.50, 0.75, 1.00, 2.00, 3.00,
       4.00, 5.00, 6.00, 7.00, 8.00, 9.00, 10.0, 11.0,
       12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0,
       20.0,
```

Se corre *ARWpost.exe* para generar un archivo con extensión *.ctl* en el directorio especificado en *namelist.ARWpost*:

```
1  $ cd ~/Build_WRF/WRF/ARWpost
2  $ ./ARWpost.exe
```

Se va al directorio donde se generó el archivo *.ctl* para generar un gráfico a partir de este:

```
1  $ ~/Build_WRF/ARWpost
2  $ grads
3  $ open test.ctl
4  $ d t2
```

El gráfico obtenido a partir de la variable **t2**, se muestra en la Figura 1.
El gráfico obtenido a partir de la variable **tk**, se muestra en la Figura 2.
El gráfico obtenido a partir de la variable **p** , se muestra en la Figura 3.
El gráfico obtenido a partir de la variable **v** , se muestra en la Figura 4.

Figura 1: Variable t2.

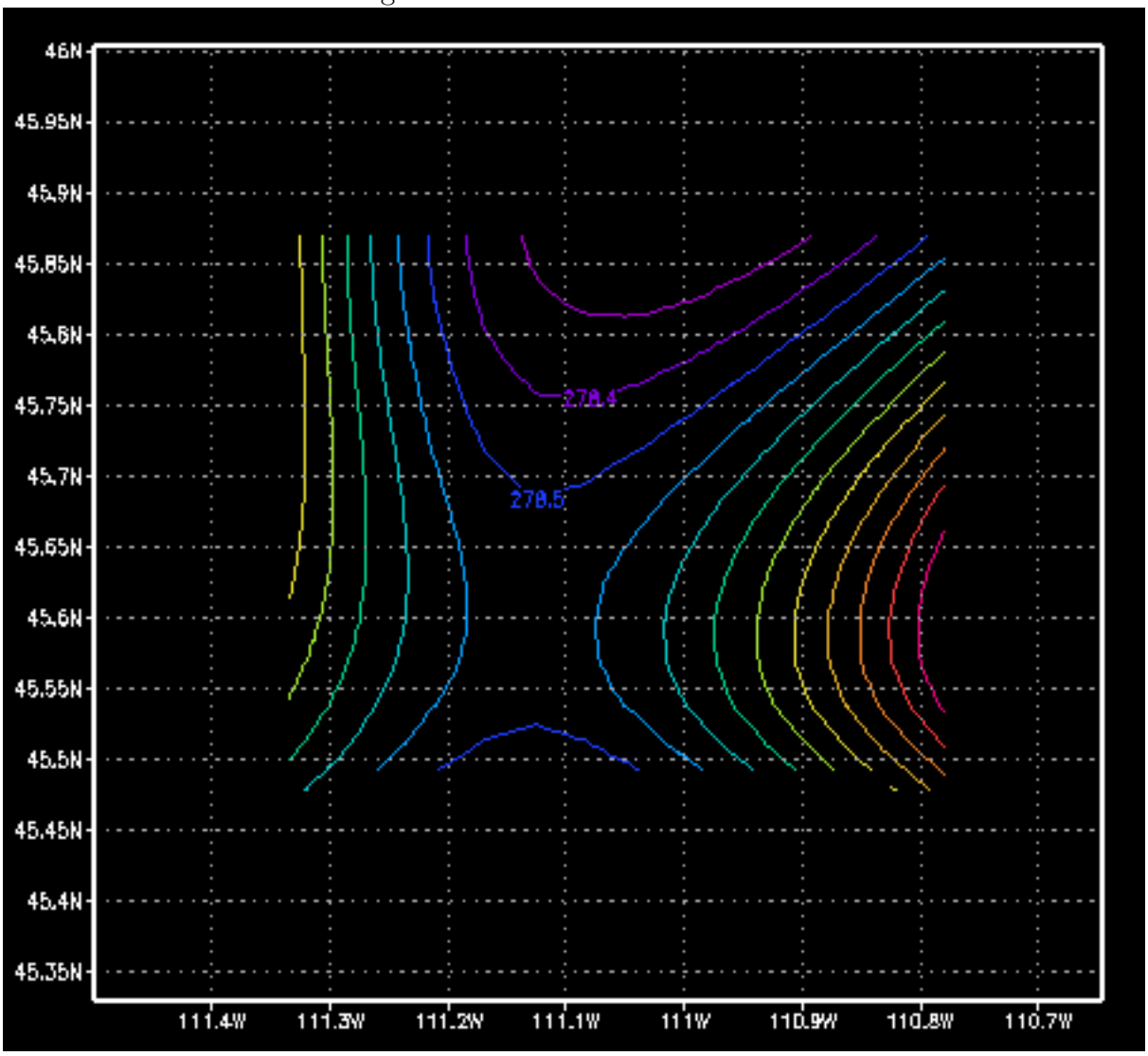


Figura 2: Variable tk.

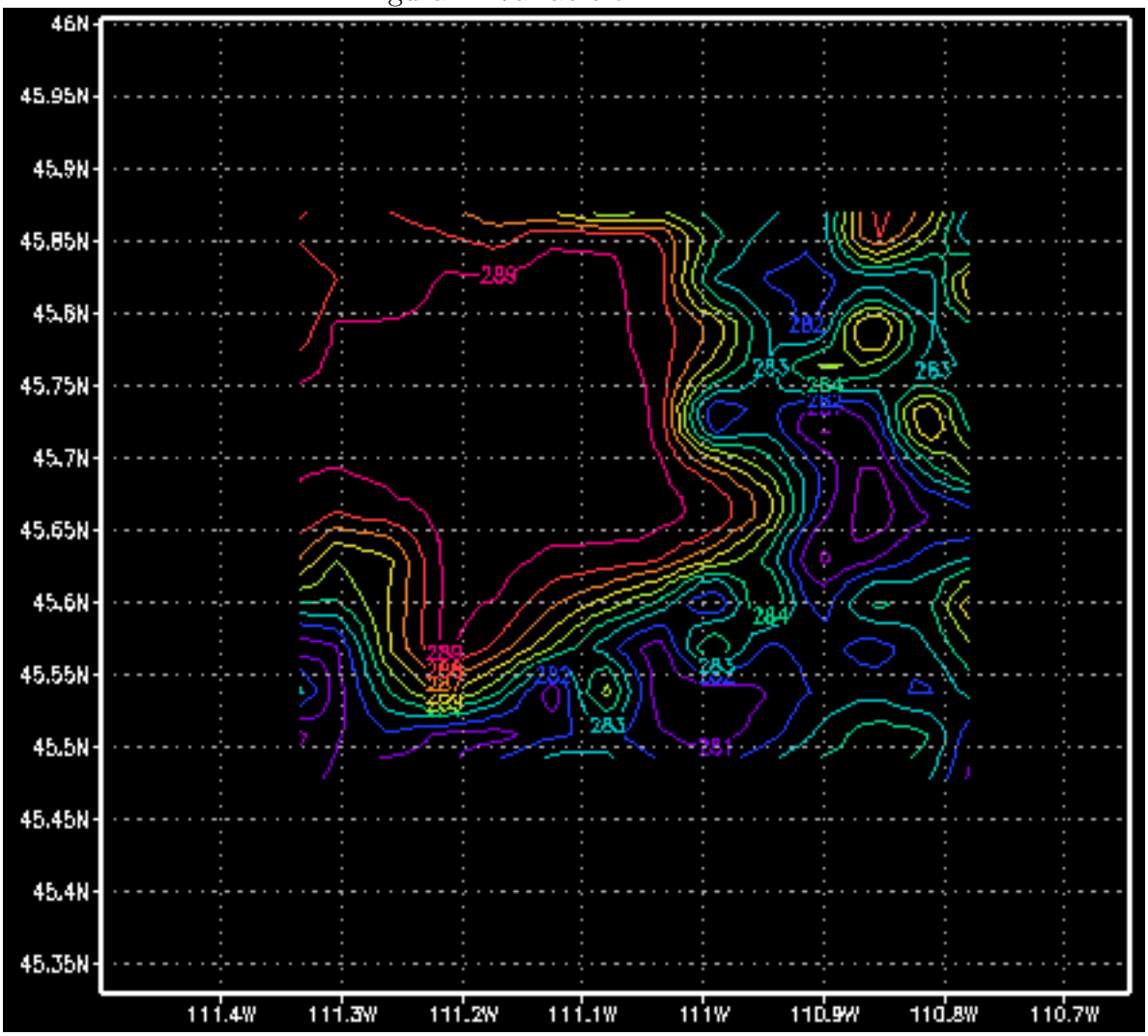


Figura 3: Variable p.

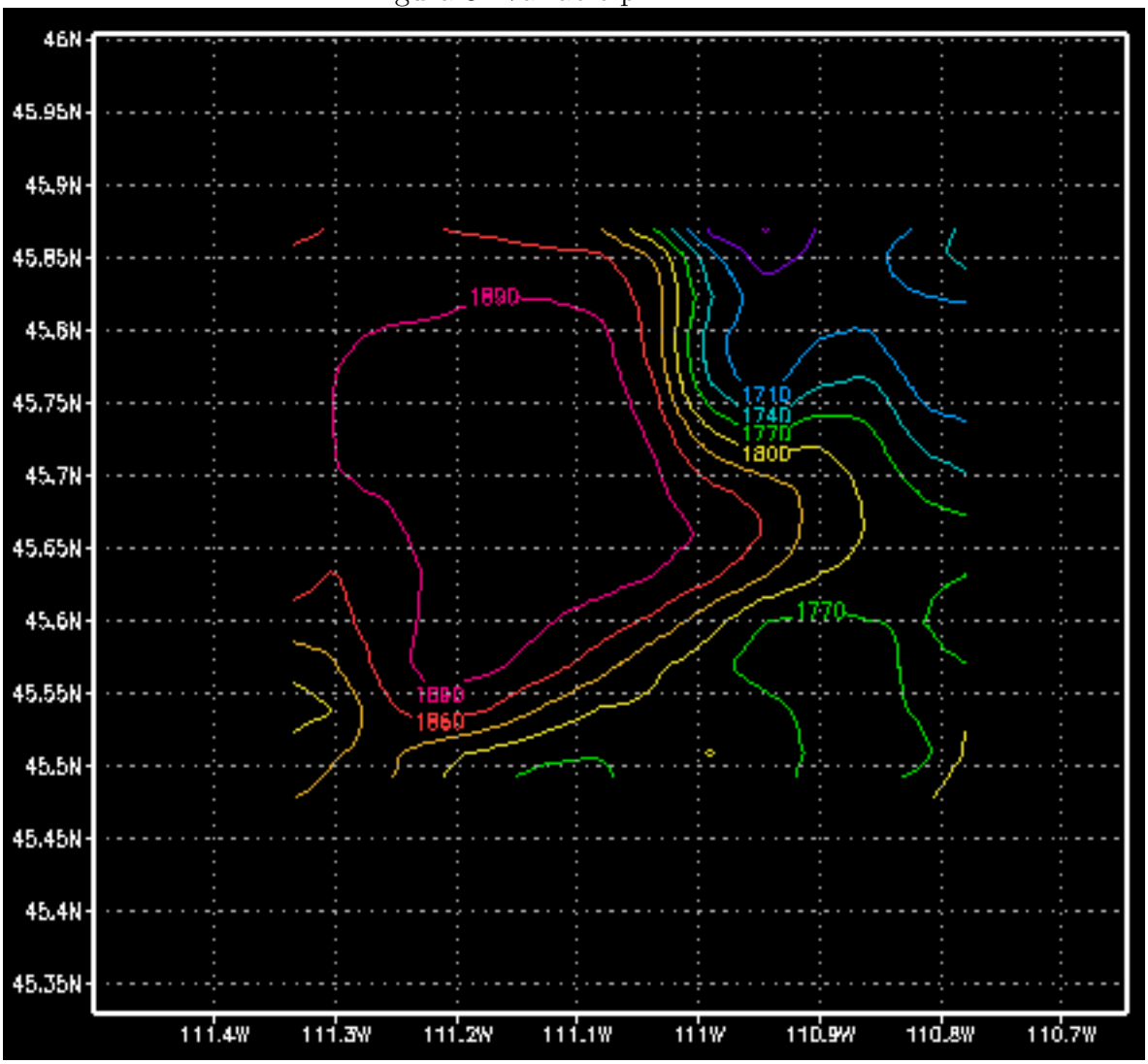
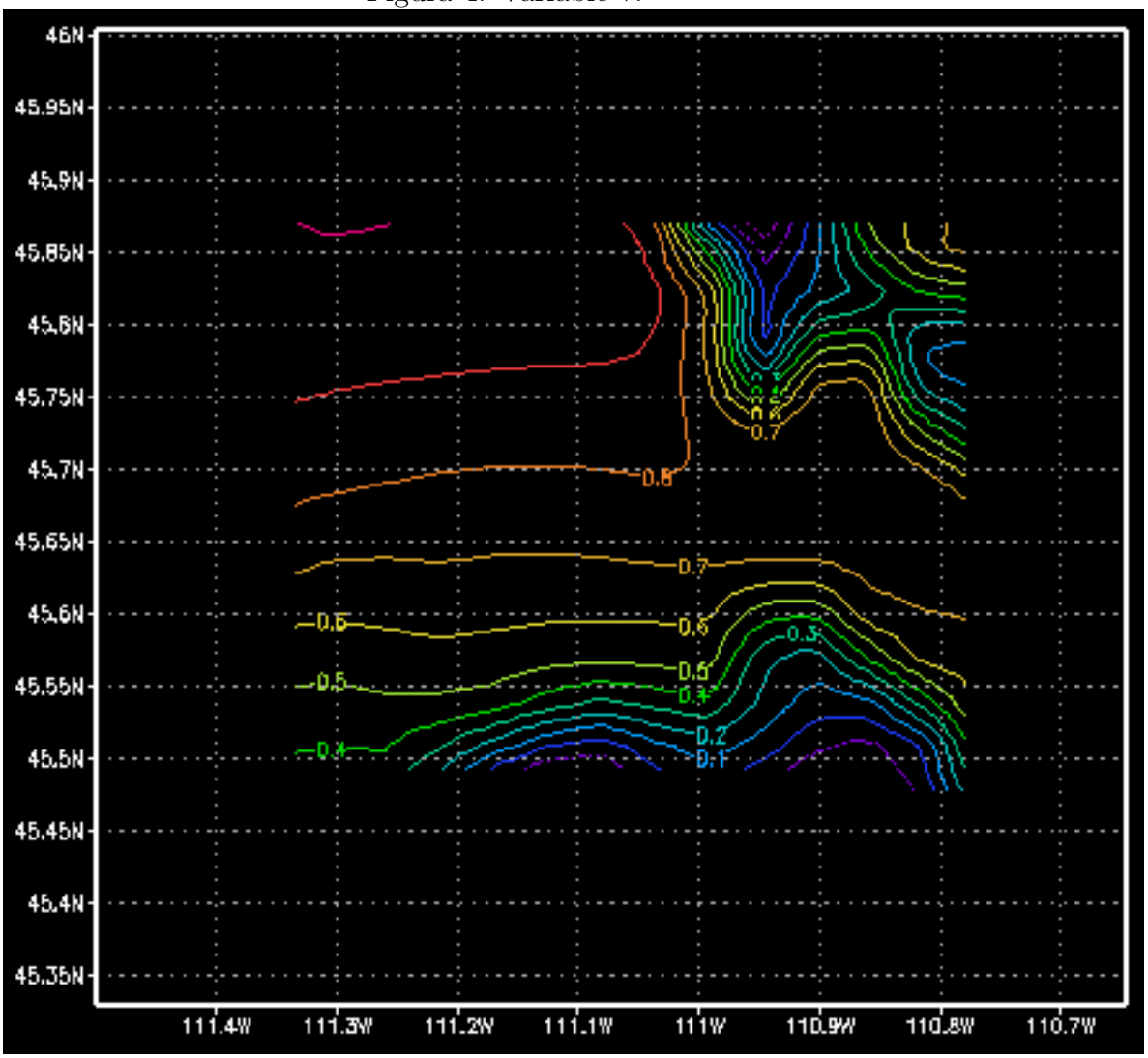


Figura 4: Variable v.



Observaciones

- De acuerdo a los permisos del usuario que instale WRF, WPS y ARWpost; puede ser necesario tener permisos de "superusuario" para algunas tareas. Esta situación se encontró principlamente al momento de correr ejecutables.

- Al momento de ejecutar *geogrid.exe*, dependiendo de la configuración del archivo *namelist.wps*, pueden existir errores que indiquen la carencia de algunos datos. En este caso, los archivos pueden descargarse manualmente desde: `http://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html`

- Al momento de ejecutar *metgrid.exe*, pueden existir errores que indiquen inconsistencias con la configuración de *geogrid.exe* y los parámetros en el archivo *namelist.wps*. Se debe tener en cuenta que la generación de los ejecutables es secuencial, por lo tanto *metgrid.exe* depende directamente de *geogrid.exe*.

- Algunos parámetros elegidos para compilar *WRF* y *WPS* dependen directamente de el equipo en el que se va a trabajar. El listado de opciones es mostrado directamente en el terminal, o puede ser visto en: `http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3.9/users_guide_chap2.htm`

- Algunos comandos y opciones de ploteo con *GrADS* pueden encontrarse en: `http://www.jamstec.go.jp/frsgc/research/iprc/nona/GrADS/plot-contour.html`

- Las guías que provee la web de *GrADS*, hasta el 15/08 se encuentran inactivas: `http://grads.iges.org/grads/`

- La instalación fue una adaptación de distintos archivos y videos encontrados online:

  - `http://www2.mmm.ucar.edu/wrf/OnLineTutorial/`
  - `https://www.youtube.com/channel/UCzzX_r13xOoubpyk_0oJ0bg`
  - `http://www2.mmm.ucar.edu/wrf/users/wrfv2/runwrf.html`
  - `http://forum.wrfforum.com/`