

**Manual de uso y ejecución del Modelo Numérico  
POM (*Princeton Ocean Model*).**

**ELABORÓ**

**Lic. en Matemáticas M. Alfredo Terrazas Silva**

**RESPONSABLE**

**Dr. Federico Angel Velázquez Muñoz**

**Departamento de Física**

**Universidad de Guadalajara**

**Apoyo PROMEP a la incorporación de NPTC**

**2012-2013.**

**UDG-PTC-1048**

**PROMEP/103.5/12/3418**

## Introducción

El modelo POM es un poderoso código escrito en Fortran77 que es utilizado para simular la dinámica del océano. Tiene la capacidad de simular problemas de amplio rango: circulación y procesos de mezcla en ríos, estuarios, plataforma y talud continental, lagos, mares semi-cerrados y mares abiertos y globales. POM es un modelo en coordenadas sigma con turbulencia integrada y sub-modelos de onda con capacidad de inundación y secado de las zonas costeras. En la actualidad, el uso de modelos numéricos para simulación de procesos costeros, representa una herramienta de suma importancia para evaluar efectos y para estudiar los procesos y fenómenos que afectan la zona costera.

## Especificaciones

Este manual fue diseñado para cuatro versiones de este modelo, las cuales se describen en forma breve a continuación.

Versión	Descripción
pom2k	Versión clásica del modelo tridimensional que resuelve las ecuaciones primitivas de Navier-Stokes en coordenadas rectangulares en la horizontal y coordenadas sigma en la vertical.
pom08	Versión que incorpora la capacidad de inundar y secar regiones costeras.
pomsed	Versión que incorpora la capacidad de transportar sedimentos disuelto en al agua.
pomnh	Versión que elimina la aproximación hidrostática y permite resolver procesos no lineales.

A continuación se mostrarán cada uno de los archivos que conforman estas versiones junto con una breve descripción.

La versión pom2k, está formada con los siguientes archivos:

pom2k.c	Enlista y define la mayoría de las variables utilizadas en el modelo, así como sus dimensiones.
pom2k.f	Este archivo contiene el código principal y las funciones con las que trabaja el modelo.
runpom2k	Compila el modelo y genera un archivo ejecutable, además tiene los parámetros principales con los que se trabaja.

Además de los archivos descritos, existen dos carpetas; una cuenta con los archivos que definen las condiciones iniciales del modelo (**iconds**) y la otra con los archivos que definen las condiciones de frontera (**bconds**). En estas carpetas se pueden añadir archivos con las condiciones particulares de cada configuración o problema.

## Edición del archivo runpom2k.

Se describen los principales cambios a realizar en el archivo runpom2k cuando se desea establecer algún problema en particular.

<pre> # Runscript for pom2k # # ----- PARAMETER FILES ----- # echo '      parameter(im=65, jm=49, kb=21)' &gt; grid echo '      iproblem= 1' &gt; params echo '      days= 10.0' &gt;&gt; params echo '      prtdl= 1./24.' &gt;&gt; params echo '      dte= 1.' &gt;&gt; params # # ----- MAKE LINKS ----- rm bcond.f icond.f  ln -s bconds/bcond.f bcond.f ln -s iconds/seamount.f icond.f  # ----- COMPILE &amp; RUN ----- # ifort pom2k.f -o pom2k.out # </pre>	<p>El programa trabaja con mallas horizontales de <math>IM \times JM</math> nodos y con KB niveles horizontales.</p> <p>Para modificar estos valores, bastará con cambiar los números señalados en la figura.</p>
<pre> # Runscript for pom2k # # ----- PARAMETER FILES ----- # echo '      parameter(im=65, jm=49, kb=21)' &gt; grid echo '      iproblem= 1' &gt; params echo '      days= 10.0' &gt;&gt; params echo '      prtdl= 1./24.' &gt;&gt; params echo '      dte= 1.' &gt;&gt; params # # ----- MAKE LINKS ----- rm bcond.f icond.f  ln -s bconds/bcond.f bcond.f ln -s iconds/seamount.f icond.f  # ----- COMPILE &amp; RUN ----- # ifort pom2k.f -o pom2k.out # </pre>	<p>El parámetro days corresponde al número de días que el modelo simula y prtdl corresponde al tamaño de los intervalos para guardar información.</p> <p>En este ejemplo, el modelo simula 10 días y guarda los datos cada hora (1/24 de día).</p>

<pre> # Runscript for pom2k # # ----- PARAMETER FILES ----- # echo '    parameter(im=65, jm=49, kb=21)' &gt; grid echo '    iproblem= 1' &gt; params echo '    days= 10.0' &gt;&gt; params echo '    prtd1= 1./24.' &gt;&gt; params echo '    dte= 1.' &gt;&gt; params # # ----- MAKE LINKS ----- rm bcond.f icond.f  ln -s bconds/bcond.f bcond.f ln -s icons/seamount.f icond.f  # ----- COMPILE &amp; RUN ----- # ifort pom2k.f -o pom2k.out # </pre>	<p>La variable <i>dte</i> debe ser modificada siempre que se modifique la profundidad (<math>h_{max}</math>) y la resolución espacial <math>\Delta x</math> mediante la relación:</p> $dte \leq \frac{0.2 \Delta x}{\sqrt{9.81 h_{max}}}$
<pre> # Runscript for pom2k # # ----- PARAMETER FILES ----- # echo '    parameter(im=65, jm=49, kb=21)' &gt; grid echo '    iproblem= 1' &gt; params echo '    days= 10.0' &gt;&gt; params echo '    prtd1= 1./24.' &gt;&gt; params echo '    dte= 1.' &gt;&gt; params # # ----- MAKE LINKS ----- rm bcond.f icond.f  ln -s bconds/bcond.f bcond.f ln -s icons/seamount.f icond.f  # ----- COMPILE &amp; RUN ----- # ifort pom2k.f -o pom2k.out # </pre>	<p>Las dos líneas señaladas hacen referencias a los archivos seleccionados como condiciones iniciales y de frontera. Para ligar tu propia condición inicial o de frontera, tendrás que copiar el archivo en la carpeta <i>icons/</i> o <i>bconds/</i> y modificar estas líneas.</p> <p>Por ejemplo, en caso de querer ligar la condición <i>cond.f</i> las líneas serían:</p> <p><code>ln -s icons/cond.f icond.f</code></p>

## Edición del archivo icond.f.

Se describen los principales cambios a realizar en el archivo icond.f cuando se desea establecer algún problema en particular. Se puede tomar como plantilla la subrutina seamount.f para el caso del pom2k y la subrutina wadseamount.f para el pom08.

<pre> subroutine icond C ***** C * C * FUNCTION    : Sets up for seamount problem. C * C ***** C C   implicit none C C   include 'pom2k.c' C C   real delh,delx,elejmid,elwjmid,ra,vel C   integer i,j,k C C   Set delh &gt; 1.0 for an island or delh &lt; 1.0 for a seamount: C C   delh=0.9e0 C C   Grid size: C C   delx=800.e0 C C   Radius island or seamount: C C   ra=2500.e0 C C   Current velocity: C C   vel=1.0e0 C ~ </pre>	<p>En las primeras líneas del código se encuentra la variable <i>delx</i>, ésta cambiará cada que se ajusten las dimensiones de la malla de acuerdo a la siguiente relación:</p> $delx = \frac{L_x}{IM}$ <p>donde <math>L_x</math> es la longitud de la malla en el eje <math>x</math> medida en metros.</p>
<pre> C   Set up grid dimensions, areas of free surface cells, and C   Coriolis parameter: C C   do j=1,jm C     do i=1,im C C     For constant grid size: C C     dx(i,j)=delx C     dy(i,j)=delx C C     For variable grid size: C C     dx(i,j)=delx-delx*sin(pi*float(i)/float(im))/2.e0 C     dy(i,j)=delx-delx*sin(pi*float(j)/float(jm))/2.e0 C C     cor(i,j)=1.e-4 C C     end do C   end do C ~ </pre>	<p>Posteriormente se definen <math>dx</math> y <math>dy</math>, en el ejemplo se definen como constantes (<i>delx</i>), pero podrían ser variables si el modelo así lo requiriera.</p>

<pre> C      Define depth: C       do i=1,im         do j=1,jm C           h(i,j)=500.e0*(1.e0-delh \$              *exp(-((east_c(i,j) \$                  -east_c((im+1)/2,j))**2 \$                  +(north_c(i,j) \$                      -north_c(i,(jm+1)/2))**2) \$                      /ra**2))           if(h(i,j).lt.1.e0) h(i,j)=1.e0 C         end do       end do C ~ </pre>	<p>Líneas después se define la batimetría del modelo <math>h(i,j)</math>, en este caso se define un monte submarino, pero bien se podría hacer un fondo plano o algún otro tipo de relieve sencillo.</p>
<pre> C       open(unit=90,file='bath.dat')       do i=1,im         do j=1,jm C           read(90,*) h(i,j)           if(h(i,j).lt.1.e0) h(i,j)=1.e0 C         end do       end do       close(90) </pre>	<p>Suponiendo que se tiene una batimetría propia en un archivo llamado “<i>bath.dat</i>”, sustituir las líneas de la explicación anterior por las que se muestran del lado izquierdo, hará que el modelo trabaje con la batimetría del archivo.</p>
<pre> C      Set initial conditions: C       do k=1,kbml         do j=1,jm           do i=1,im             tb(i,j,k)=5.e0+15.e0*exp(zz(k)*h(i,j)/1000.e0)-tbias             sb(i,j,k)=35.e0-sbias             tclim(i,j,k)=tb(i,j,k)             sclim(i,j,k)=sb(i,j,k)             ub(i,j,k)=vel*dum(i,j)           end do         end do       end do C </pre>	<p>Los ciclos anidados que se muestran en la figura establecen las condiciones iniciales de temperatura y salinidad, estas podrán ser modificadas según el problema particular que se quiera trabajar.</p>

## Edición del archivo *bcond.f*.

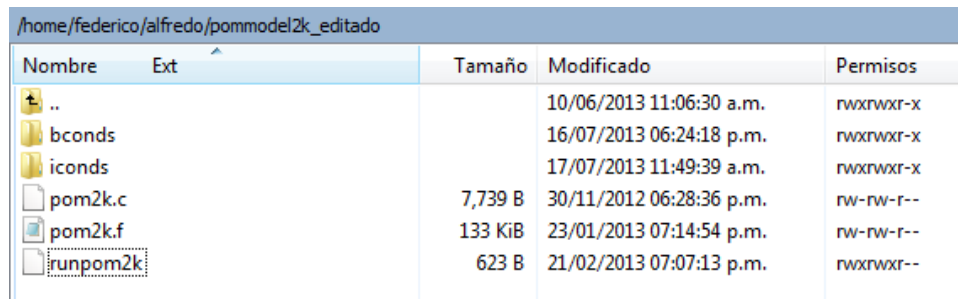
Se describen los principales cambios a realizar en el archivo *bcond.f* cuando se desea establecer algún problema en particular. Tanto en el pom2k, como en el pom08, existe una función llamada *bcond.f* que puede ser utilizada como plantilla.

<pre>etide=-2.e0*sin(2.e0*pi*time/1.e0) etide=-tidamp*sin(2.e0*pi*time/1.e0)</pre>	<p>La variable <i>etide</i> aplica un forzamiento por marea. Cuando la variable <i>tidamp</i> (amplitud medida en metros) es igual a 2, las dos líneas que se muestran del lado izquierdo son equivalentes.</p>
<pre>C      East: C !tne:!wad:- for wad replace H with D       uaf(im,j)=uabe(j)       \$      +rfe*sqrt(grav/d(imml,j))       \$      *(el(imml,j)-ele(j)-etide)       uaf(im,j)=ramp*uaf(im,j)       vaf(im,j)=0.e0  C C      West: C !tne:!wad:- for wad replace H with D add tide       uaf(2,j)=uabw(j)       \$      -rfe*sqrt(grav/d(2,j))       \$      *(el(2,j)-elw(j)-etide)       uaf(2,j)=ramp*uaf(2,j)       uaf(1,j)=uaf(2,j)       vaf(1,j)=0.e0  C       end do  C       do i=2,imml  C C      North: C !tne:!wad:- for wad replace H with D       vaf(i,jm)=vabn(i)       \$      +rfn*sqrt(grav/d(i,jmml))       \$      *(el(i,jmml)-eln(i)-etide)       vaf(i,jm)=ramp*vaf(i,jm)       uaf(i,jm)=0.e0  C C      South: C !tne:!wad:- for wad replace H with D       vaf(i,2)=vabs(i)       \$      -rfs*sqrt(grav/d(i,2))       \$      *(el(i,2)-els(i)-etide)       vaf(i,2)=ramp*vaf(i,2)       vaf(i,1)=vaf(i,2)       uaf(i,1)=0.e0  C       end do</pre>	<p>Se muestra en el ejemplo las condiciones de frontera en los cuatro puntos cardinales, donde además se agrega el mismo forzamiento causado por la variable <i>etide</i>.</p>

## Compilación y ejecución del modelo.

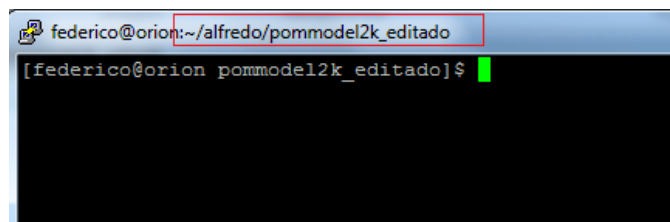
Se explicarán los comandos necesarios para lograr la ejecución del modelo, tomando como ejemplo la versión 2k, para las demás versiones se procedería de manera similar.

1.- Se deben tener ya configurados de forma adecuada los 3 archivos descritos al inicio de este documento y las dos carpetas que contienen los archivos de las condiciones iniciales y de frontera.



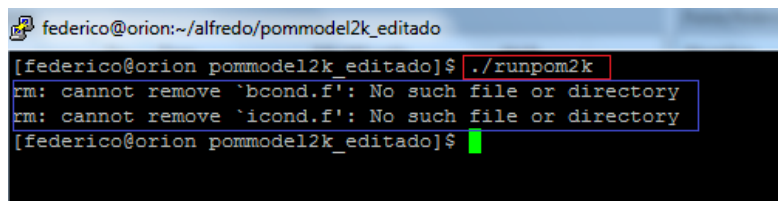
Nombre	Ext	Tamaño	Modificado	Permisos
..			10/06/2013 11:06:30 a.m.	rw-rw-r--
bconds			16/07/2013 06:24:18 p.m.	rw-rw-r--
iconds			17/07/2013 11:49:39 a.m.	rw-rw-r--
pom2k.c		7,739 B	30/11/2012 06:28:36 p.m.	rw-rw-r--
pom2k.f		133 KiB	23/01/2013 07:14:54 p.m.	rw-rw-r--
runpom2k		623 B	21/02/2013 07:07:13 p.m.	rw-rw-r--

2.- En la consola, nos aseguramos de encontrarnos en la carpeta correcta.



```
federico@orion:~/alfredo/pommodel2k_editado
[federico@orion pommodel2k_editado]$
```

3.- Se compilará el modelo tecleando “./runpom2k” y Enter. Generando un archivo ejecutable.



```
federico@orion:~/alfredo/pommodel2k_editado
[federico@orion pommodel2k_editado]$ ./runpom2k
rm: cannot remove `bcond.f`: No such file or directory
rm: cannot remove `icond.f`: No such file or directory
[federico@orion pommodel2k_editado]$
```

Las líneas señaladas en azul aparecerán sólo durante la primera vez que se ejecuta el modelo, puesto que no existen los archivos para ser borrados.

4.- Después de ejecutar la instrucción del punto anterior, la carpeta del modelo tendrá los archivos que se muestran en la figura.



/home/federico/alfredo/pommodel2k_editado			
Nombre	Ext	Tamaño	N
..			10
bconds			10
iconds			10
bcond.f		14 B	10
grid		37 B	10
icond.f		17 B	10
params		148 B	10
pom2k.c		7,739 B	30
pom2k.f		133 KiB	20
pom2k.out		732 KiB	10
runpom2k		620 B	10

El archivo pom2k.out es el ejecutable del modelo, grid y params guardan parámetros de la configuración del modelo. icond.f y bcond.f son ligas a los archivos originales que se encuentran en las carpetas de condiciones.

5.- Para ejecutar el modelo se tecleará “./pom2k.out” y Enter. Deberán aparecer en pantalla algo similar a lo que se muestra a continuación.

```
federico@orion:~/alfredo/pommodel2k_editado
[federico@orion pommodel2k_editado]$ ./pom2k.out

source      =
title       = Run 1

iproblem    =      1
mode        =      3
nadv        =      1
nitera      =      2
sw          =    0.5000
nread       =      0
dte         =    1.00
dti         =    30.0
isplit      =    30
time_start  = 2000-01-01 00:00:00 +00:00
days       =   10.0000
iend        =   28800
prtd1       =    0.0417
iprint      =    120
prtd2       =    1.0000
swtch       =   1000.00
iswtch      =  2880000
iskp, jskp  =    1,    1
lramp       =      F
rhoref      =  1025.000
tbias       =    0.000
sbias       =    0.000
grav        =    9.8060
kappa       =    0.4000
z0b         =    0.010000
cbcmin      =    0.002500
cbcmax      =    1.000000
horcon      =    0.200
tprni       =    0.2000
umol        =    0.0000
hmax        =   4500.00
vmax1       =  100.0000
slmax       =    2.0000
kl1, kl2    =    6,   19
ntp         =      2
nbct        =      1
nbcs        =      1
ispadv      =      5
smoth       =    0.1000
alpha       =    0.2250
```

Y posteriormente se muestran cada uno de los intervalos en los que el modelo guarda las variables. La variable *time* indica el total de días que ha simulado el modelo (5.7917 días en este caso).

```
time = 5.7500, iint = 16560, iext = 31, iprint = 120

vtot = 0.4369663E+12 atot = 0.8916718E+09 eaver = -0.2550119E-01
taver = 0.1642213E+02 saver = 0.3499972E+02 tsalt = 0.1529370E+14
*****
time = 5.7917, iint = 16680, iext = 31, iprint = 120

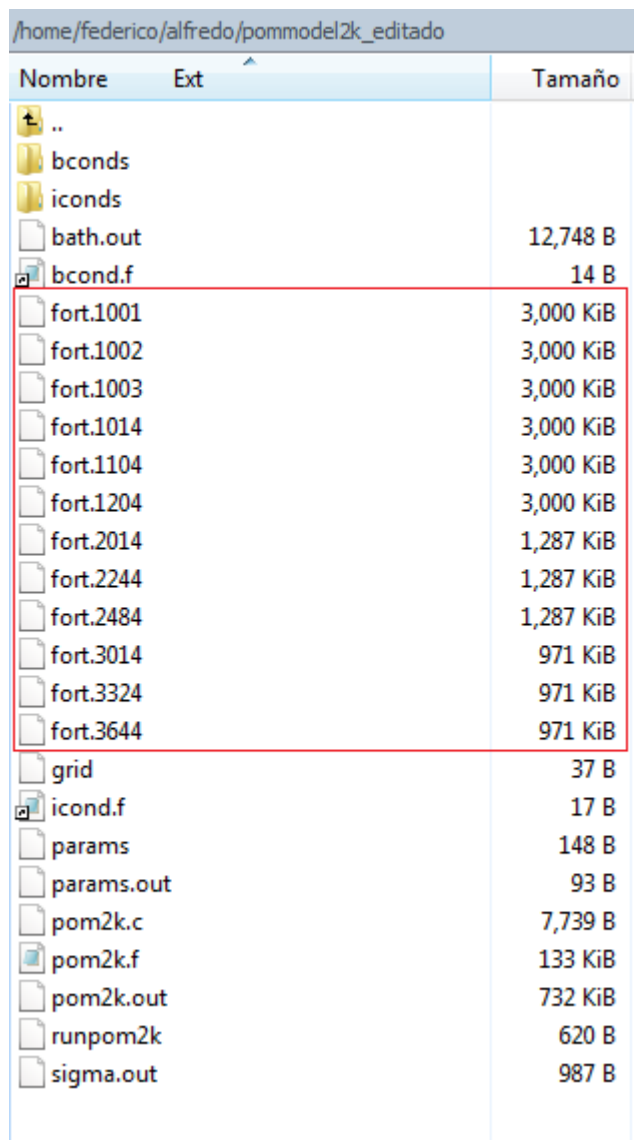
vtot = 0.4369669E+12 atot = 0.8916718E+09 eaver = -0.2506502E-01
taver = 0.1641867E+02 saver = 0.3499972E+02 tsalt = 0.1529372E+14
*****
time = 5.8333, iint = 16800, iext = 31, iprint = 120
```

Cuando el modelo termine de correr, la variable *time* mostrará el tiempo total que se puso en la variable *days* del archivo *runpom2k*. En el ejemplo se usó *days* = 10.

```
vtot = 0.4369668E+12  atot = 0.8916718E+09  eaver = -0.2513672E-01
taver = 0.1638225E+02  saver = 0.3499974E+02  tsalt = 0.1529372E+14
*****
time = 10.0000, iint = 28801, iext = 31, iprint = 120
[federico@orion pommodel2k_editado]$
```

## Sobre la lectura e interpretación de los datos de salida.

Una vez ejecutado el modelo, se obtendrán varios archivos con los datos de salida, éstos archivos son los que tienen un nombre general de la forma “fort.\*\*\*\*”, donde los asteriscos corresponden a números que se especifican en donde se inserta el código para guardar las variables que se desean como datos de salida.



Nombre	Ext	Tamaño
..		
bconds		
icons		
bath.out		12,748 B
bcond.f		14 B
fort.1001		3,000 KiB
fort.1002		3,000 KiB
fort.1003		3,000 KiB
fort.1014		3,000 KiB
fort.1104		3,000 KiB
fort.1204		3,000 KiB
fort.2014		1,287 KiB
fort.2244		1,287 KiB
fort.2484		1,287 KiB
fort.3014		971 KiB
fort.3324		971 KiB
fort.3644		971 KiB
grid		37 B
icond.f		17 B
params		148 B
params.out		93 B
pom2k.c		7,739 B
pom2k.f		133 KiB
pom2k.out		732 KiB
runpom2k		620 B
sigma.out		987 B

A continuación se presenta una tabla explicando el contenido de cada archivo.

fort	Contenido
1001	La variable $e_l$ que es el nivel del mar (medido en metros).
1002	La variable $u_a$ que es la componente $u$ de la velocidad integrada en la vertical (m/s).

1003	La variable $va$ que es la componente $v$ de la velocidad integrada en la vertical (m/s).
------	---

Al resto de los fort les corresponden las siguientes especificaciones:

$$fort.(k * 1000 + i * 10 + n)$$

donde  $k$  e  $i$  pueden tomar los siguientes valores:

k	Significado	i	Significado
1	El fort contiene datos tomados de una sección paralela al plano $xy$ (capas superficiales).	$0 < i < kb$	Índice sobre el eje $z$ de donde fue tomada la capa superficial.
2	El fort contiene datos tomados de una sección paralela al plano $xz$ (secciones zonales).	$0 < i < jm$	Índice sobre el eje $y$ de donde fue tomada la sección zonal.
3	El fort contiene datos tomados de una sección paralela al plano $yz$ (secciones meridionales).	$0 < i < im$	Índice sobre el eje $x$ de donde fue tomada la sección meridional

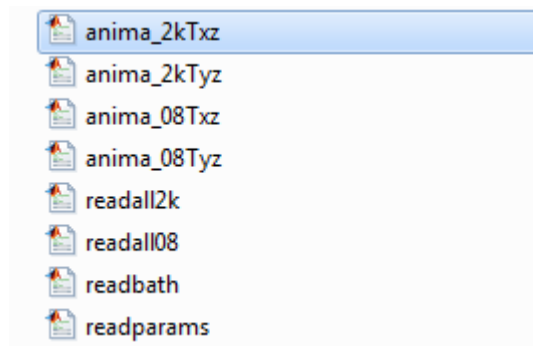
Finalmente,  $n$  indicará la variable que guarda ese fort:

$n$	Variable
4	Temperatura
5	Densidad

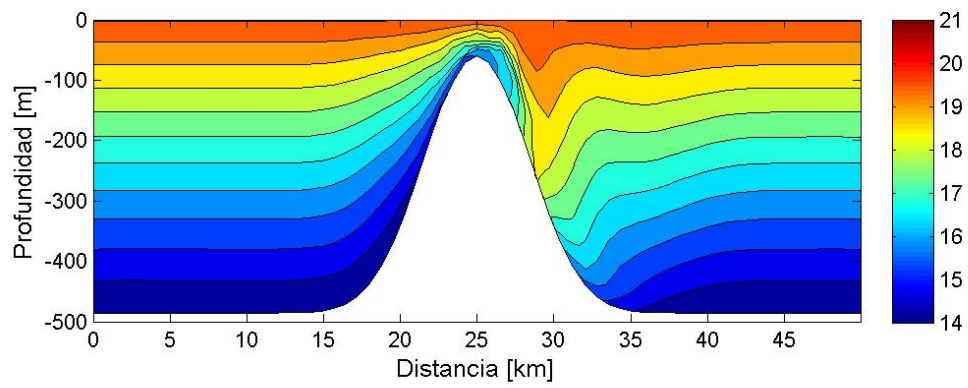
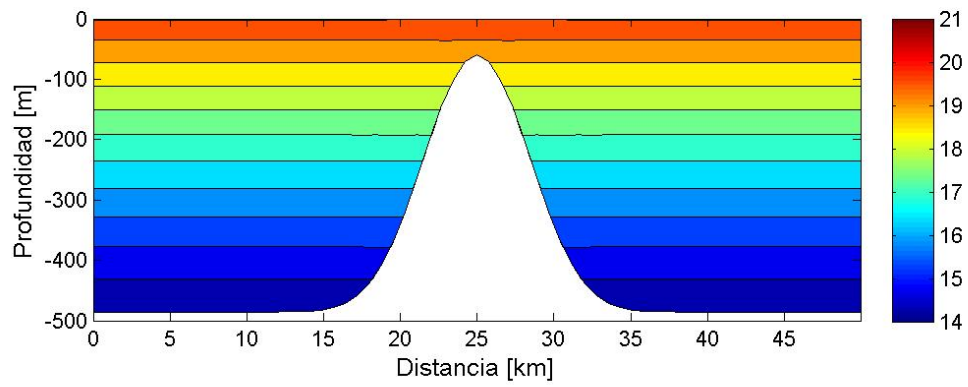
Es posible guardar otras variables y obtenerlas en archivos *fort*, para esto es necesario revisar las funciones *writexy*, *writexz* y *writeyz* que se encuentran en el archivo pom2k.f.

## Visualización de los datos.

Además de los archivos y carpetas del modelo, existen una serie de archivos dentro de una carpeta que se llamada MPROGS/ y que sirven para leer los archivos *fort* para hacer gráficas con los datos obtenidos.

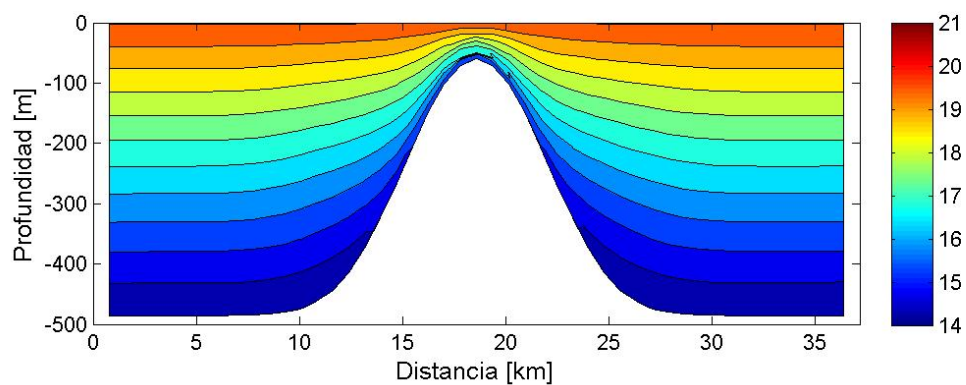
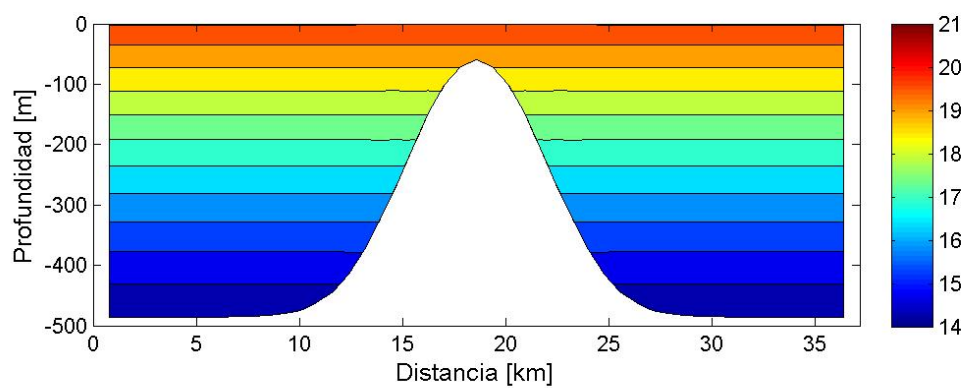


Los programas *anima\_2kTxz* y *anima\_08Txz* generan y guardan una serie de imágenes obtenidas a partir de los datos de los *fort.2\*\*\**, es decir, muestran imágenes de secciones zonales.



La primera figura es el perfil de la condición inicial de temperatura, la segunda es el perfil de 1/24 de día después.

De manera similar, los programas *anima\_2kTyz* y *anima\_08Tyz*, muestran imágenes de secciones meridionales (fort.3\*\*\*).





Los programas *readall08* y *readall2k* utilizan las funciones *readparam* y *readbath* para mostrar el nivel del mar en una gráfica tridimensional.

