Zygmunt Kowalik
Institute of Marine Science
University of Alaska, Fairbanks

Workbook on Numerical Modeling

Fairbanks, 2003

Table of Contents

This workbook is complementary to the course based on the book by Z. Kowalik and T.S. Murty entitled: *Numerical Modeling of Ocean Dynamics*, Published by: World Scientific Publishing Co., Singapore, New Jersey, London, 1993.

The purpose of this workbook is to apply fundamentals of computer simulation described in *Numerical Modeling of Ocean Dynamics* (NM) to the short practical projects for the real hands–on experience in applying numerical methods. We assume that the reader possesses experience with FORTRAN. The workbook will follow the material described in NM and solve simple problems through FORTRAN programs. It contains short descriptions of the programs. All programs are copied onto diskette, which is also included. We will confine ourselves to transport equations (diffusion and advection), shallow water phenomena – tides, storm surges and tsunamis, and three-dimensional time dependent oceanic motion.

CHAPTER I

PROBLEMS IN TRANSPORT EQUATIONS

1. Vertical coordinate

We shall start by considering a simple one dimensional problem for the advection of concentration described in the first part of Ch.II of NM. Ch.II considers one dimensional horizontal advection. For such problem the construction of the finite difference formulas is straightforward. Procedure along the vertical direction is not so obvious, therefore we take this direction as a basic direction for all problems related to one directional advection (along the vertical direction this transport is called convection). To facilitate further consideration Fig.4.1 from NM is taken and is renamed here as Fig.1.

It is important to notice that the concentration (salinity) and the vertical velocity (w) are not located at the same grid point. Therefore index (l) describes two different points separated by half of the grid distance ($h_{z,l}/2$). The beginning of the system of coordinate is taken at the free surface with z axis pointing upward. Such geometry introduces inconvenience of being at the negative values of z when processes in the ocean are considered. Therefore, the following assumptions are made: depth is positive from the surface to the bottom, index l is positive and increases from the surface towards the bottom, but the positive z direction is still upward.

Convective transport along z is expressed by the equation,

$$\frac{\partial S}{\partial t} + w\frac{\partial S}{\partial z} = 0 \tag{I.1}$$

Using Taylor series given by formulas (2.1) and (2.2) and grid notation (2.25) from NM we shall construct a few difference schemes to be applied later on.

Forward difference:

$$S^m_{l-1} = S^m_l + h\frac{\partial S^m_l}{\partial z} + \frac{h^2}{2}\frac{\partial^2 S^m_l}{\partial z^2} + \tag{I.2}$$

Notice somewhat unusual notation. The forward direction from the point l is upward but since the index is diminishing in this direction, therefore upward is the point $l-1$.

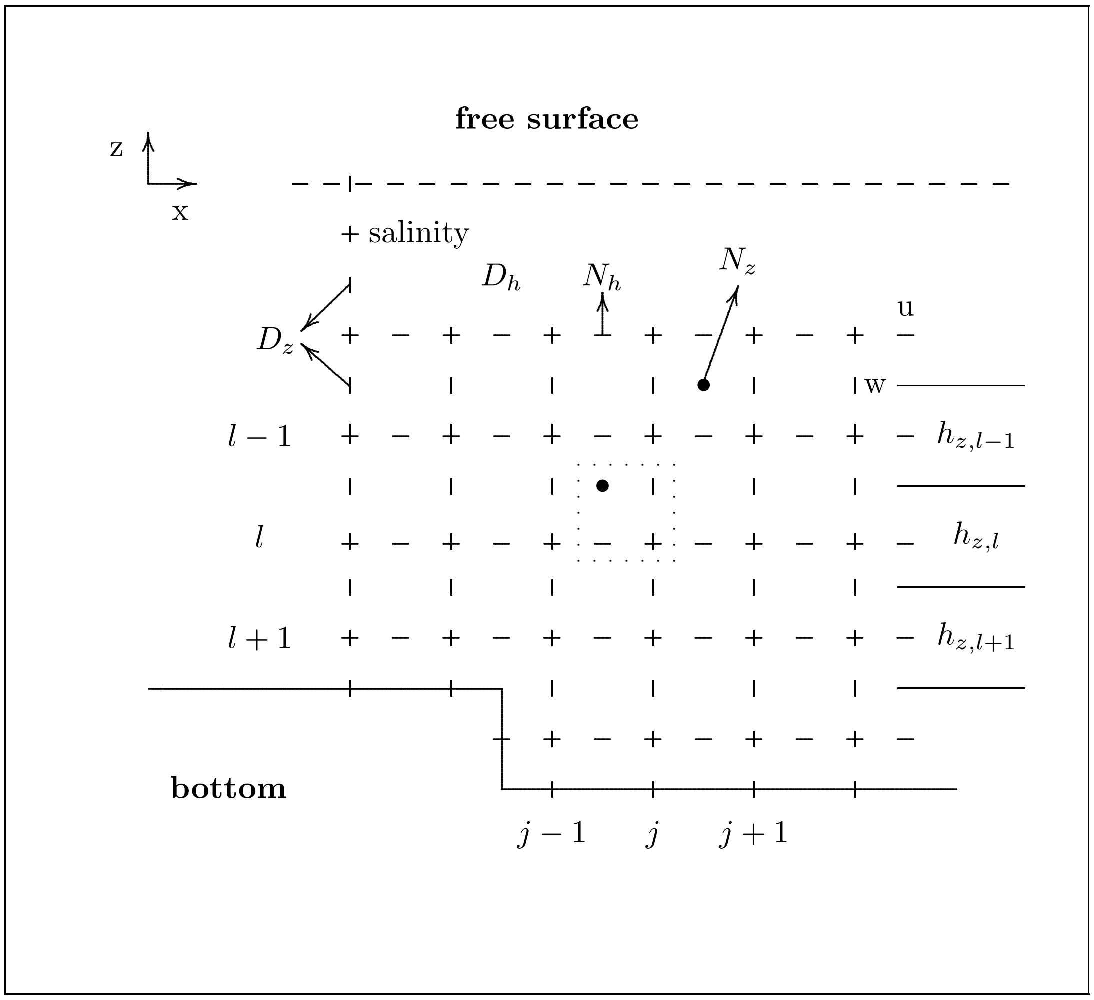


Fig. 1
Numerical grid distribution in the x–z plane.
Dotted rectangle contains variables with the same j, l indices.

Backward difference:

$$S_{l+1}^{m} = S_l^m - h\frac{\partial S_l^m}{\partial z} + \frac{h^2}{2}\frac{\partial^2 S_l^m}{\partial z^2} + \tag{I.3}$$

Similar formulas can be written for the time derivative, but here both index and time increase in the same direction, thus;

$$S_l^{m+1} = S_l^m + T\frac{\partial S_l^m}{\partial t} + \frac{T^2}{2}\frac{\partial^2 S_l^m}{\partial t^2} + \tag{I.4}$$

$$S_l^{m-1} = S_l^m - T\frac{\partial S_l^m}{\partial t} + \frac{T^2}{2}\frac{\partial^2 S_l^m}{\partial t^2} + \tag{I.5}$$

Problem No. 1, explicit time marching with directional derivatives in space

We start by considering an upwind-downwind formulation for the space derivative and first order approximation for the time marching. We are looking for the formulas similar to (2.54) given in NM. With the help of formula (I.4) the time derivative can be expressed as

$$\frac{\partial S}{\partial t} \simeq \frac{S_l^{m+1} - S_l^m}{T} \tag{I.6}$$

Space derivative will be constructed according to the direction of velocity at the grid point l. If velocity is positive (upward) then

$$\frac{\partial S}{\partial z} \simeq \frac{S_l^m - S_{l+1}^m}{h}, \tag{I.7}$$

and, if current direction is negative (downward) then

$$\frac{\partial S}{\partial z} \simeq \frac{S_{l-1}^m - S_l^m}{h} \tag{I.8}$$

Velocity in the concentration grid point l can be expressed as an average $wa = 0.5(w_l + w_{l+1})$. Velocities along the positive and negative directions in the grid point l are defined as;

$$wp = 0.5(wa + abs(wa)) \qquad wn = 0.5(wa - abs(wa)) \tag{I.9}$$

Using above formulas the following numerical analog of (I.1) can be constructed;

$$\frac{S_l^{m+1} - S_l^m}{T} + wp\frac{S_l^m - S_{l+1}^m}{h} + wn\frac{S_{l-1}^m - S_l^m}{h} = 0 \tag{I.10}$$

Introducing notation used below in the program fluxdir.f,

$$h = H = HS, \qquad TOH = T/HS$$

$$FLP = WP(SO(L) - SO(L+1)), \qquad FLN = WN(SO(L-1) - SO(L))$$

one can write formula to update salinity value,

$$SN(L) = SO(L) - TOH(FLP + FLN) \tag{I.11}$$

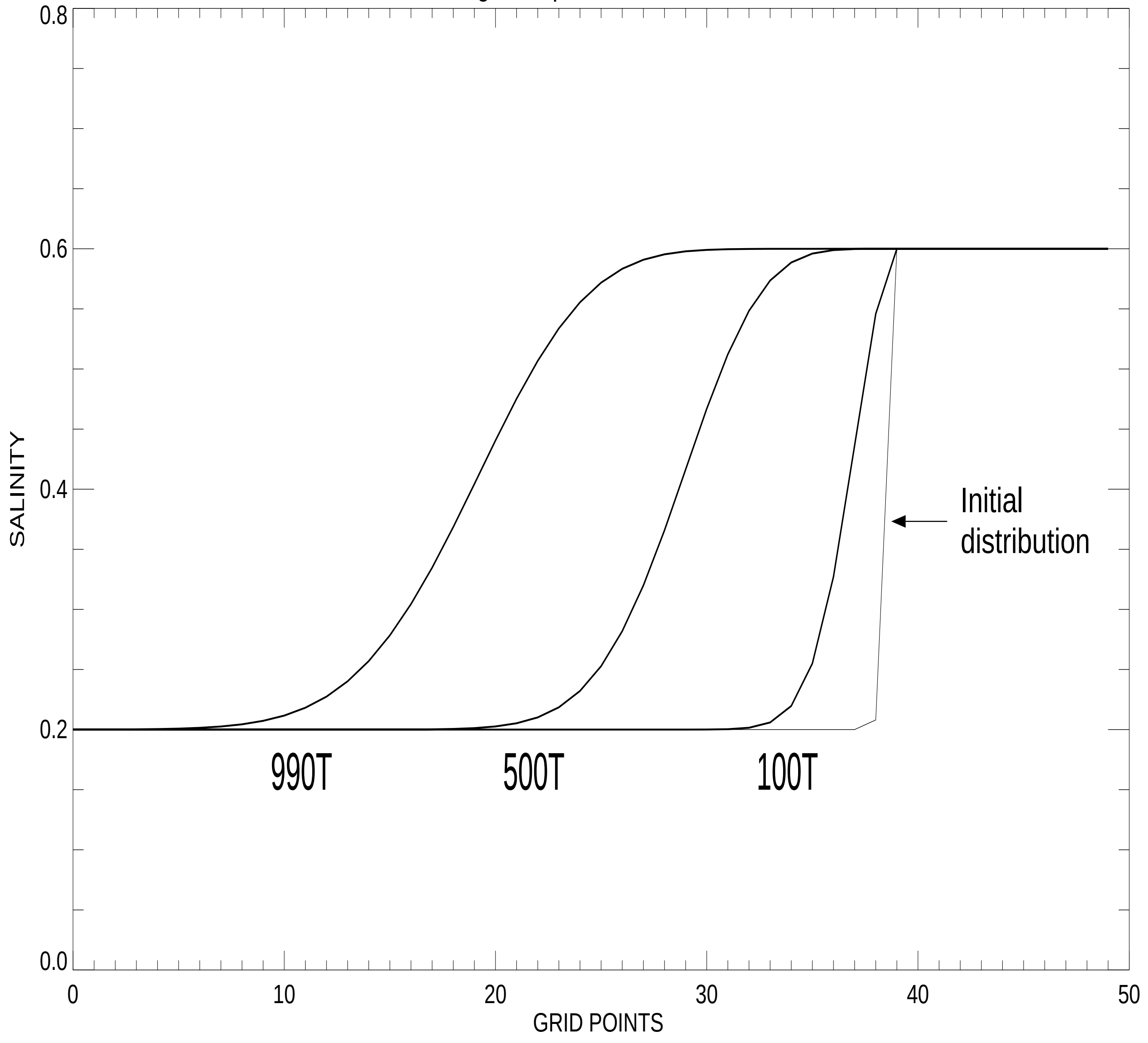
Even though the FORTRAN program is written to account for the variable velocity in space (w_l) and the variable space step $h = hz(l)$, in this computation the space step is taken as constant $h = hz(l) = hs = 5$cm and the vertical velocity is also set constant $w_l = 10^{-4}$cm/s. The time step is $T = 1000$s. The domain is 2.5m long in the vertical direction, therefore there are 50 grid points in this domain. Initial salinity distribution is 0.2 from 1 to 39 grid point and 0.6 from 40 to 50 grid point. The old value of salinity S_l^m is called in the program SO and the new value S_l^{m+1} is named SN.

The results of computation are depicted in Fig.2. The initial jump of salinity propagates downstream and slowly changes its shape due to numerical diffusion. (see explanation in NM given on page 52).

```
==============================================================
C PROGRAM TO CALCULATE CONVECTION BY upwind DERIVATIVES
C CALCULATIONS ARE MADE IN CGS UNITS
C PROGRAM FLUXDIR.F
PARAMETER (LZZ=50)
REAL W(LZZ),SO(LZZ),SN(LZZ),HZ(LZZ)
T=1000. ! TIME STEP
MON=1000 !TOTAL NUMBER OF TIME STEPS
DO L=1,LZZ-1
HZ(L)=5. ! SPACE STEP
END DO
C BECAUSE HZ(L)=HS=CONST
HS=5.
C INITIAL DISTRIBUTION
DO L=1,39
SO(L)=0.2
END DO
DO L=40,LZZ
SO(L)=0.6
END DO
OPEN(UNIT=3,NAME='SOZ1.VER',STATUS='UNKNOWN')
C SET VERTICAL VELOCITY
DO L=1,LZZ
W(L)=1.0E-4
END DO
C START TIME LOOP
50 N=N+1
C WRITE SALINITY AT TIME STEP 2, 100, 500, 990.
IF(N.EQ.2.OR.N.EQ.100.OR.N.EQ.500.OR.N.EQ.990)THEN
WRITE(3,*)(SO(L),L=1,LZZ)
```

```
END IF
TOH=T/HS
C CALCULATE NEW VALUE OF SALINITY SN
DO L=2,LZZ-1
wa=0.5*(w(l)+w(l+1))
wp=0.5*(wa+abs(wa))
wn=0.5*(wa-abs(wa))
FLP=WP*(SO(L)-SO(L+1))
FLN=WN*(SO(L-1)-SO(L))
SN(L)=SO(L)-(FLP+FLN)*TOH
END DO
C KEEP BOUNDARY CONDITIONS
SN(LZZ)=0.6
SN(1)=0.2
C CHANGE VARIABLES
DO L=1,LZZ
SO(L)=SN(L)
END DO
C STOP COMPUTATION AT TIME STEP MON
IF(N.EQ.MON) GO TO 18
C RETURN BACK TO 50 TO REPEAT TIME LOOP
GO TO 50
18 CONTINUE
END
```

Fig.I.2: Upwind-downwind



Problem No. 2, explicit time marching and central derivatives in space

In this problem we shall march in time by (I.6) but instead of the directional space derivatives let's use a central form;

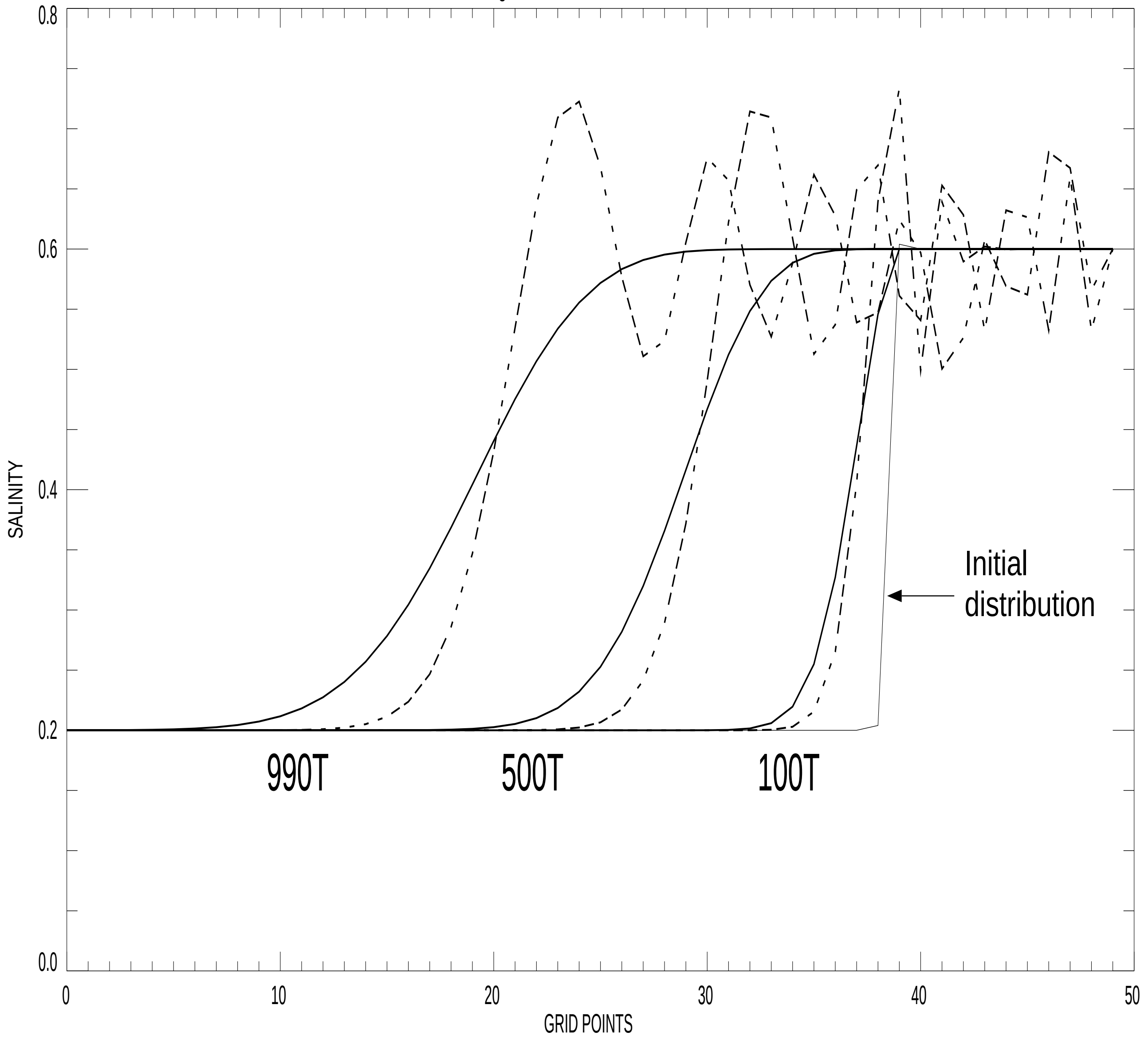
$$\frac{\partial S}{\partial z} \simeq \frac{S^m_{l-1} - S^m_{l+1}}{2h} \tag{I.12}$$

This formula is of the second order approximation and should better describe convective processes. We shall not write a new FORTRAN program for this problem, it is adequate to substitute formula (I.11) in the old program for

$$SN(L) = SO(L) - 0.5TOH \times W(L)(SO(L-1) - SO(L+1)) \tag{I.13}$$

The results of computation are depicted in Fig. 3 (broken lines) together with former results obtained by the upwind differencing (continuous lines). The new results are much better when compared against the upwind method because salinity jump is reproduced better. The new source of errors are short wave wiggles in the upper right-hand corner. These are so called short dispersive waves and their origin was explained in Ch.II, Sec. 6 of NM.

Fig.I.3: Central derivative



Problem No. 3, simple correction schemes

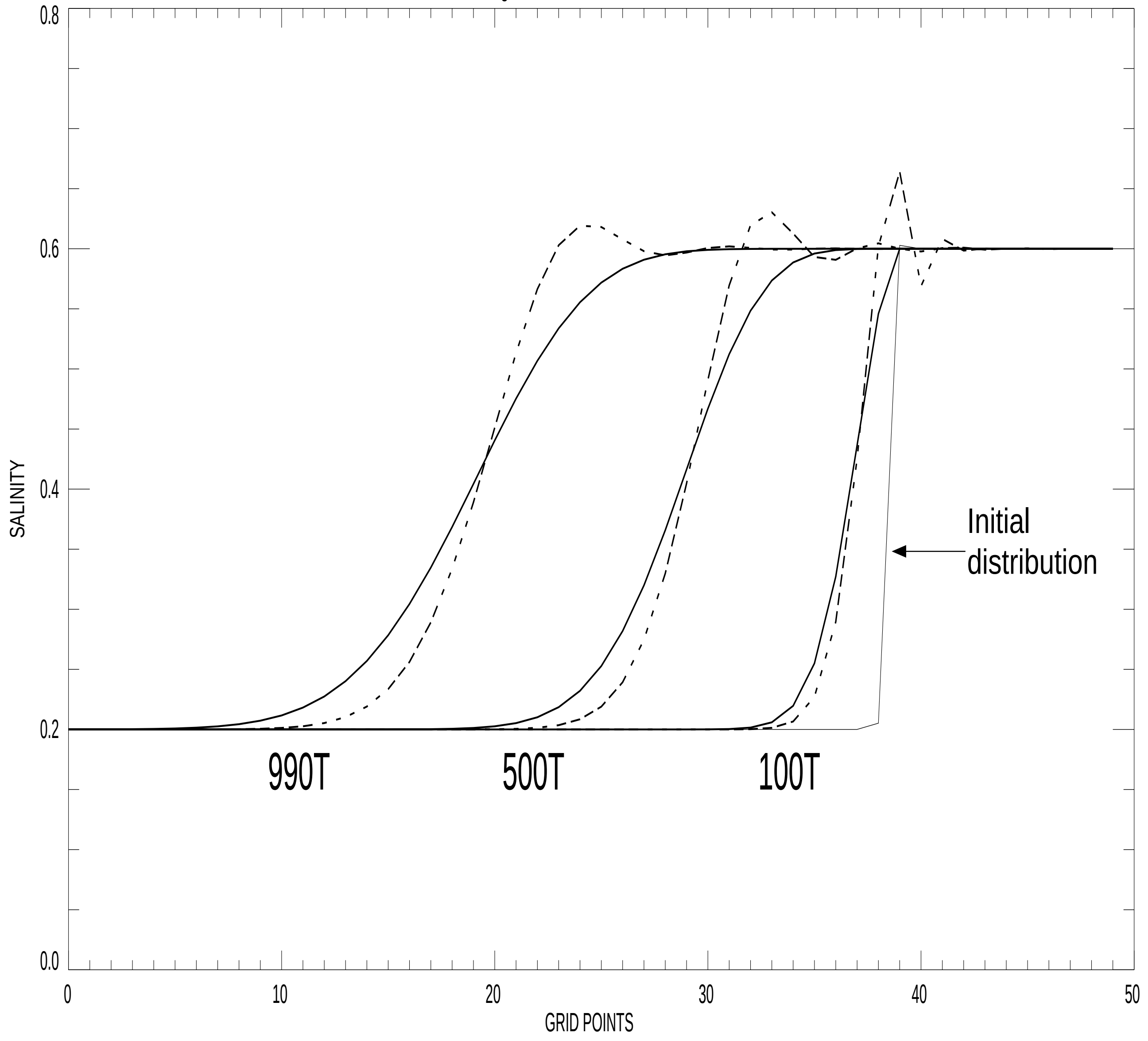
Two numerical schemes used for calculation of the convection depict severe limitations. Directional derivatives due to numerical diffusion smooth salinity jump, while central derivatives generate parasite short wave spikes which may influence physics of propagation. For example, these numerical spikes in the vertical distribution of salinity may lead to an unstable density stratification and generate motion related rather to numerical than physical causes. In the ensuing consideration the numerical schemes with better properties will be constructed, but now we demonstrate a very simple scheme for suppressing some of the short wave oscillations. In NM several simple ways of improving performance of the numerical schemes has been delineated. To suppress numerical friction a positive definite scheme (Ch.II, Sec. 9.2) can be used. To partly suppress short wave oscillations numerical scheme (2.257) from NM can be used,

$$\frac{S_l^{m+1} - S_l^m}{T} + w\frac{S_{l-1}^m - S_{l+1}^m}{2h}$$

$$-A\frac{w^2T}{2h^2}(S_{l+1}^m + S_{l-1}^m - 2S_l^m) = 0 \tag{I.14}$$

The last term in this equation serves to suppress short waves. If one compares formula (I.14) with the original expression (2.257), one finds the coefficient A. This coefficient serve to derive better results and usually A is ranging from 10 to 20. Results of applications of the above scheme with $A = 15$ are given in Fig.4. Again for comparison, results of this exercise are depicted together with the results from the upwind scheme.

Fig.I.4: Central corrected



Problem No. 4, QUICKEST method

Computational algorithm (I.14) has been constructed with the help of Taylor series and basic diffusive equation. Derivation is outlined in (2.177) through (2.195). The algorithm is not totally effective but shows improvement. Basic idea is to bring second derivative for correction of numerical errors. The QUICKEST (Ch.II, Sec. 9.4b of NM) unlike previous methods increases number of grid points around main computational point. By using two points upstream and downstream from the computational grid point the higher order derivatives are constructed and better approximation is achieved. We shall use here formula (2.195) from NM in the specific geometry related to the vertical coordinate and for the positive vertical velocity only.

$$SN(L) = SO(L) - Q*(0.5*(SO(L-1) - SO(L+1))$$

$$-0.5*Q*(SO(L+1) + SO(L-1) - SO(L)*2.)$$

$$+(SO(L+2) - 3.*SO(L+1) + 3.*SO(L) - SO(L-1))*(1 - Q*Q)/6.) \quad \text{(I.15)}$$

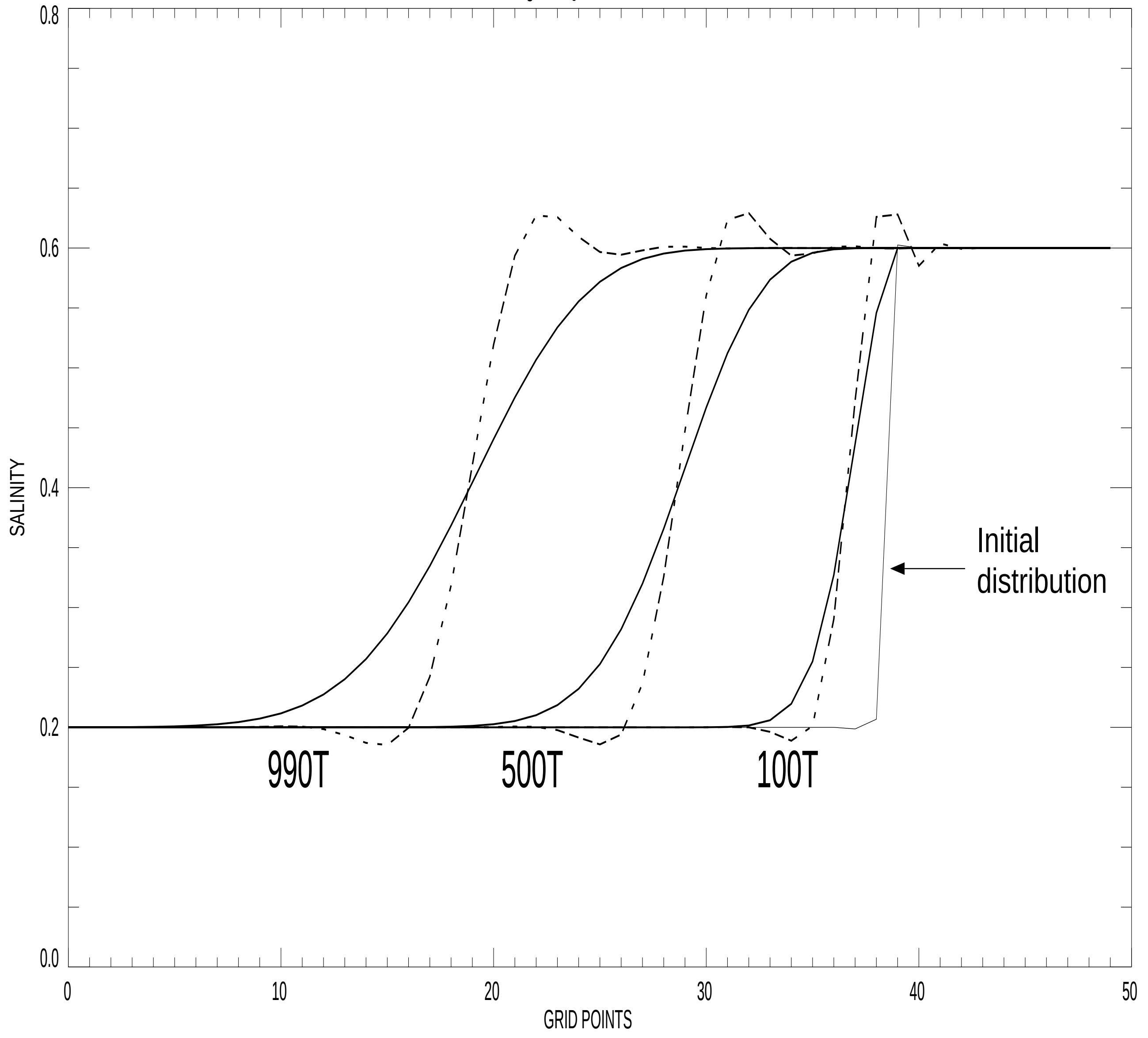
Here $Q = wT/H$. This nondimensional number is often called Courant number.

The program given below (quickest.f) applies the same geometry and parameters as Problem No.1. The result of computation depicted in Fig. 5 (broken lines) is compared against directional derivatives result (continuous lines).

```
==================================================
C PROGRAM TO CALCULATE CONVECTION BY QUICKEST
c program quickest.f
c program originated by Z.K. at IMS
PARAMETER (LZZ=50)
REAL W(LZZ),SO(LZZ),SN(LZZ),HZ(LZZ)
T=1000. ! time step
MON=1000 ! total number of time steps
DO L=1,LZZ
HZ(L)=5.
END DO
DO L=1,39
SO(L)=0.2
END DO
DO L=40,LZZ
SO(L)=0.6
```

```
END DO
OPEN(UNIT=3,NAME='SOZ.VER',STATUS='UNKNOWN')
C SET VERTICAL VELOCITY
DO L=1,LZZ-1
W(L)=1.0E-4
END DO
c q parameter =w*T/hz
Q=(1.0E-4)*T/5
c start time loop
50 N=N+1
c write salinity at time step 2,100,500,990.
IF(N.EQ.2.OR.N.EQ.100.OR.N.EQ.500.OR.N.EQ.990)THEN
WRITE(3,*)(SO(L),L=1,LZZ)
END IF
c calculate new salinity
DO L=3,LZZ-2
SN(L)=SO(L)-Q*(0.5*(SO(L-1)-SO(L+1))-0.5*Q*(SO(L+1)+SO(L-1)-
2SO(L)*2.)+(SO(L+2)-3.*SO(L+1)+3.*SO(L)-SO(L-1))*(1-Q*Q)/6.)
END DO
c keep two boundary points as known values
SN(1)=0.2
SN(2)=0.2
SN(LZZ)=0.6
SN(LZZ-1)=0.6
C CHANGE VARIABLES
DO L=1,LZZ
SO(L)=SN(L)
END DO
IF(N.EQ.MON) GO TO 18
c return to repeat loop
GO TO 50
18 CONTINUE
END
```

Fig.I.5: QUICKEST



Problem No. 5, stability parameters.

Among various properties of an numerical algorithm we discuss stability parameter because it helps to grasp quickly the basic properties of the applied algorithm.

Upwind scheme

Starting from the upwind equation in the form (I.10), and searching for stability of this equation with the help of (2.48) from NM, we obtain stability parameter λ in the complex form,

$$\lambda = 1 - \frac{wT}{h} + \frac{wT}{h}(\cos\kappa h + i\sin\kappa h) \tag{I.16}$$

Amplitude or absolute value of this parameter defines stability of an upwind numerical scheme and for the positive velocity it is given by formula (2.52) from NM,

$$|\lambda|^2 = 1 + 2q(q-1)(1-a) \tag{I.17}$$

Here: $q = wT/h$ and $a = \cos\kappa h$. As we know $|\lambda| \leq 1$ only if q is less than 1. Behavior of the stability parameter depends on two nondimensional numbers q (nondimensional Courant number) and κh. It is obvious that the Courant number describes ratio of convective transport velocity (w) and numerical cell velocity (h/T). The nondimensional number

$$\kappa h = \frac{2\pi h}{L} \tag{I.18}$$

defines ratio of given wavelength L to the space step h. The shortest possible wavelength is $L = 2h$. The wave is usually well resolved in space when $L/h \geq 10$.

Stability parameter is a complex number and together with the amplitude of stability parameter (which we usually call stability parameter), the second information is available, i.e., phase. Stability parameter answers important question, namely how signal is changed due to numerical scheme. The amplitude tells whether a signal over one time step was amplified or damped, the phase tells whether signal was accelerated or deccelerated. We shall briefly describe this dependence through eq.(I.17) for the stability and for the phase we use the following approach. According to formula (2.49) from NM

$$\lambda = c^* e^{i\omega T} = c^*(\cos\omega T + i\sin\omega T) \tag{I.19}$$

The phase lag α can be defined from this formula as

$$\tan\alpha = \frac{\sin\omega T}{\cos\omega T} \tag{I.20}$$

Comparing (I.16) to the right hand side of (I.19) we arrive at formula for the phase lag of the upwind differencing scheme,

$$\tan\alpha = \frac{q\sin\kappa h}{1-q(1-\cos\kappa h)} \tag{I.21}$$

In the ensuing program (stability.f) both stability amplitude and phase lag are calculated. The result of computations are depicted in Figs. 6 and 7. Errors in the stability and phase are diminishing with the higher spatial resolution. The role of the Courant number is less obvious.

Before embarking on the rough road of calculations it is well advised to look into properties of (I.17) and (I.21) by considering various values of nondimensional parameters. For example, behavior of these formulas at the shortest wave resolved by the numerical grid (L=2h) should always be scrutinized. Thus if L=2h, coskh=-1 and sinkh=0. Therefore $|\lambda|$=1, when q=1 or q=0. Clearly $\tan\alpha$=0 at L=2h. Behavior of $\tan\alpha$ is also very specific at q=0.5, q=$\simeq$ 0.67 and q=1. At these values $\tan\alpha$ changes sign and the phase in Fig.7 depicts discontinuity. Assume for example L=3h, hence κh=2π/3 and (I.21) will change sign at q$\simeq$ 0.67.

```
================================================================
C PROGRAM TO CALCULATE stability of the upwind method
c originated by z.k. at ims
c program stability.f
c program originated by z.k. at ims
PARAMETER (LX=50,LY=100)
REAL LAMBDA(LX,LY),alpha(lx,ly)
c Lambda is stability, alpha is angle
pi = 4.*atan(1.)
art=180./pi
c to express alpha from radians to degree multiply by art
c stability is calculated as function of two variables (two coordinates)
c variable Q=W*T/H and variable HK=2*PI*H/L. Here coordinate HK
C changes from
c small number to 2*PI. L/H sets space resolution of numerical scheme.
c L=2*H is the shortest wave. FLOAT(J)=L/H changes from 2 to LY.
c Q changes along LX from 1/50 to 1.
DO I=1,LX
DO J=2,LY
HK=2*PI/(FLOAT(J))
Q=FLOAT(I)/50. !Q=WT/H
A2=1.-COS(HK)
LAMBDA(I,J)=sqrt(1.+2.*Q*(Q-1.)*A2)
a3=q*sin(hk)
```

```
a4=-q*a2+1.
a5=a3/(a4+1.0E-6)
alpha(i,j)=art*(atan(a5))
END DO
END DO
open (unit=12,file='lambda.dat',status='unknown')
open (unit=13,file='phase.dat',status='unknown')
write(12,*)((lambda(j,k),j=1,lx),k=1,ly)
write(13,*)((alpha(j,k),j=1,lx),k=1,ly)
END
```

Central space derivative and forward time derivative (CF).

Let us check stability properties of the numerical scheme used in the problem No.2 and based on (I.6) and (I.12). Again using (2.48) from NM, we arrive at

$$\frac{\lambda - 1}{T} = \frac{iw}{h} \sin \kappa h \tag{I.22}$$

Stability parameter is

$$|\lambda| = \left[1 + (q \sin \kappa h)^2\right]^{1/2} \tag{I.23}$$

In this formula $q = wT/h$. From the above formula follows that the stability parameter is always greater than one, and therefore numerical scheme is unstable. If we check again Problem No.2 we can see that indeed the amplitude of the short waves in Fig.3 is increasing in time.

Central space derivative and central time derivative (CC).

Instead using forward derivative in time we shall use central derivative in the following scheme

$$\frac{S_l^{m+1} - S_l^{m-1}}{2T} = -w \frac{S_{l-1}^m - S_{l+1}^m}{2h} \tag{I.24}$$

This is scheme is also given in NM as (2.60). Before we check stability properties of (I.24) a few comments are needed on organization of computation by this formula. To march in time we need to know S on two previous time step (l and $l-1$). If only one initial condition is given the best way to start is to use forward derivative in time. After this first step the time marching can be done by the leapfrog explained in Fig. 3.5 of NM.

Searching stability of (I.24) we obtain two roots for the stability parameter (this result is close to (2.61) from NM),

$$\lambda_{1,2} = \frac{iq_1 \pm \sqrt{4 - q_1^2}}{2} \tag{I.25}$$

Here $q = \frac{2wT}{h}$, $\quad q_1 = q \sin \kappa h$. When $q_1^2 \le 4$ or $\frac{wT}{h} \le 1$, stability parameter $|\lambda| = 1$, and numerical scheme is stable. Although amplitude of the signal is not altered by numerical scheme the phase of the signal undergoes change defined by,

$$\tan \alpha = \frac{q \sin \kappa h}{\sqrt{4 - (q \sin \kappa h)^2}} \tag{I.26}$$

The resulting phase lag is given in Fig.8.

Fig.I.**6**:STABILITY PARAMETER LAMBDA--UPWIND SCHEME

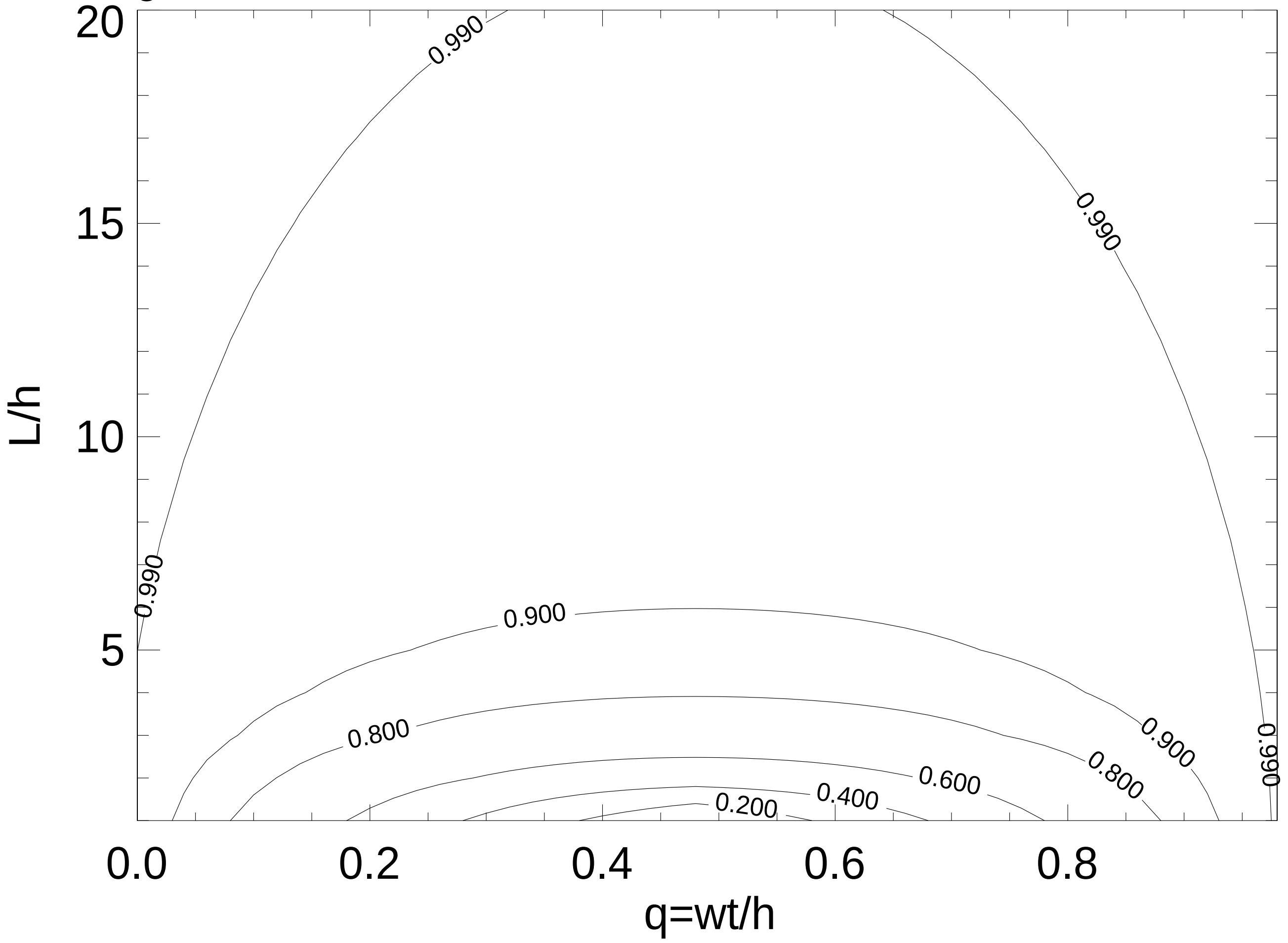


Fig.7: PHASE LAG--UPWIND SCHEME

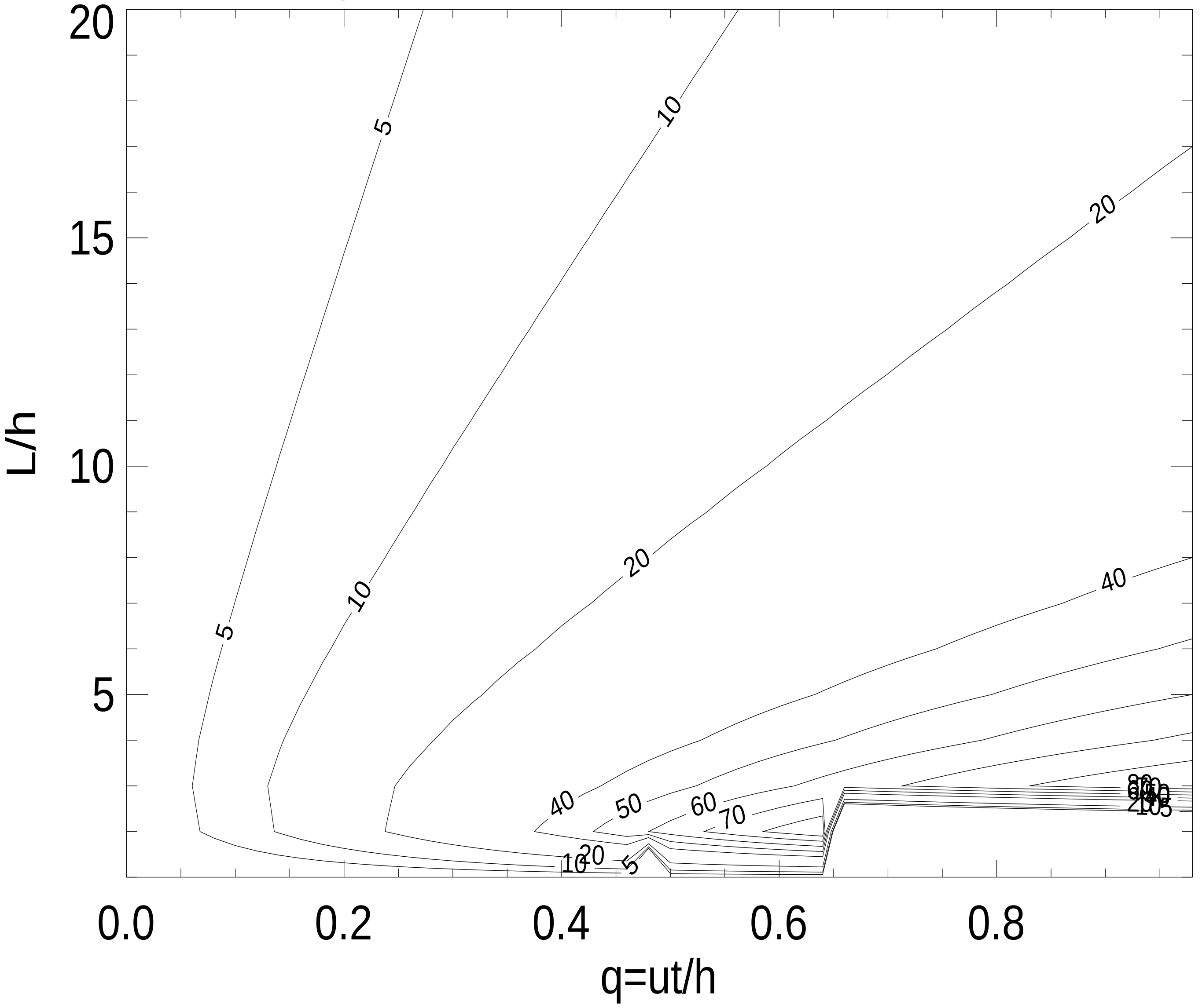
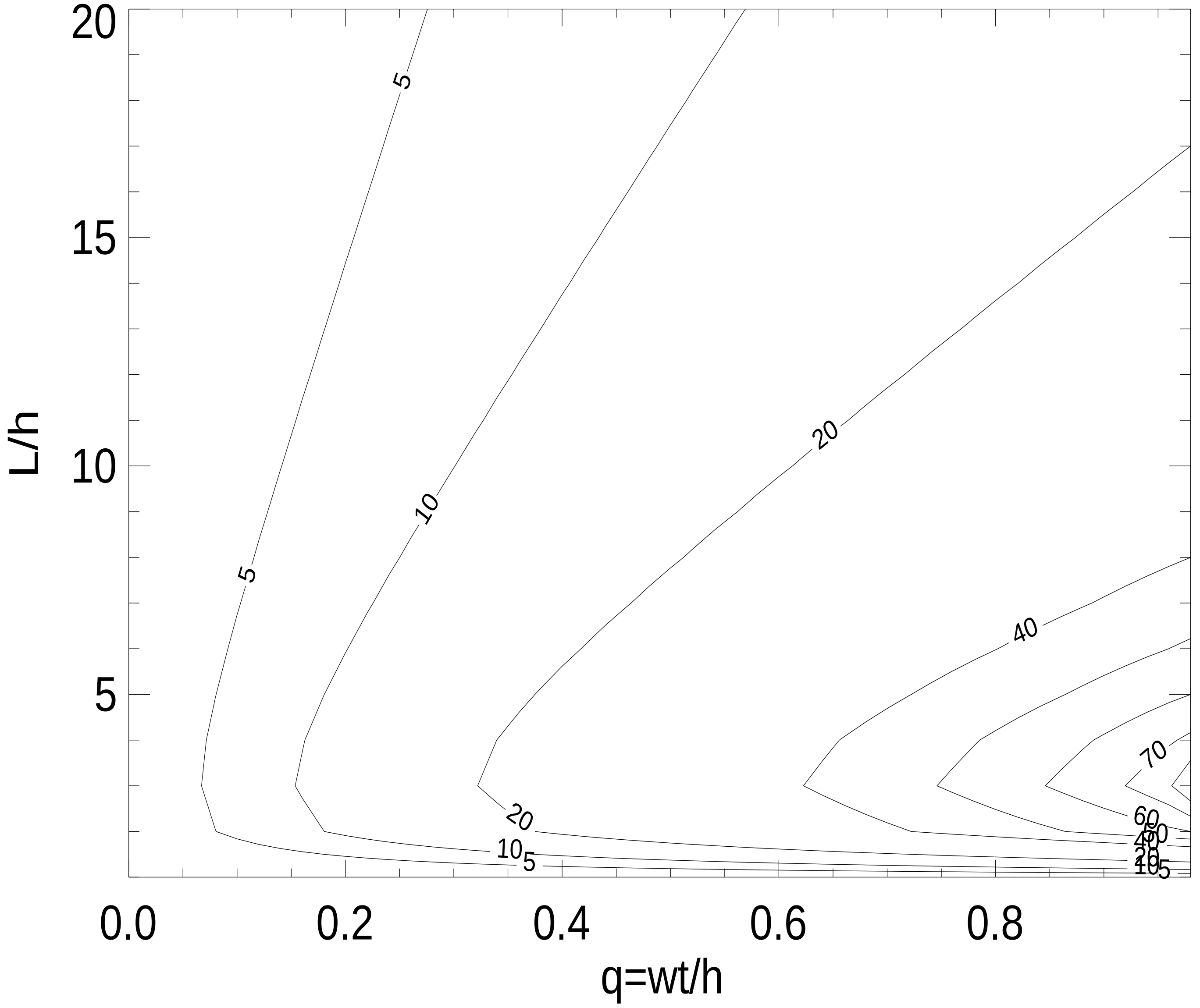


Fig.I.8: PHASE LAG--CENTRAL SCHEME



2. Flux form of convective/advective transport

Here we shortly describe an upstream-downstream numerical schemes employing equation of transport in the flux form. Such an equation is based on the principle of flux conservation. The difference equations constructed from the differential equations should as well display the same conservative properties. We shall search for the schemes which will possess good accuracy, so that they will introduce small amplitude and phase errors. Also we shall search for the schemes which are positive definite to avoid instabilities due to parasitic short waves.

Let's start from upstream-downstream scheme. Consider a control surface spanned around concentration grid point S(L) – Fig.9

Square box (red dash lines) denotes control surface. Crosses are located at the concentration points S, and vertical bars denote w component of velocity. The grid distance HZ is variable and is measured between the vertical velocity grid points.

Let us integrate equation of transport for salinity

$$\frac{\partial S}{\partial t} + \frac{\partial wS}{\partial z} = 0 \tag{I.27}$$

over a region dz and time dt, then

$$\int_0^T \left(\int_0^{\Delta z} \frac{\partial S}{\partial t} dz\right) dt = -\int_0^T \left(\int_0^{\Delta z} \frac{\partial wS}{\partial z} dz\right) dt \tag{I.28}$$

The left hand side term is easily transformed into a finite difference form. The integral on the right hand side is transformed into an integral along the boundary of the control surface. Thus

$$(S_l^{m+1} - S_l^m)\Delta z = -[(wS)_U^m - (wS)_D^m]T \tag{I.29}$$

or dividing by Δz and T we arrive at the familiar finite difference form

$$\frac{S_l^{m+1} - S_l^m}{T} = -\frac{(wS)_U^m - (wS)_D^m}{\Delta z} \tag{I.30}$$

It remains to express fluxes through the surfaces as functions of velocity on the surfaces. Thus defining the positive and negative velocity at the upper surface of the control volume which spans the grid point S(L) (see Fig.9),

$$w_U^p = 0.5(|w_l^m| + w_l^m); \quad w_U^n = 0.5(-|w_l^m| + w_l^m), \tag{I.31}$$

the flux at the upper surface can be expressed in the upstream- downstream form

$$(wS)_U^m = w_U^p S_l^m + w_U^n S_{l-1}^m \tag{I.32}$$

Similar approach can be used to define fluxes at the lower surface. Again the positive and negative velocity can be written as,

$$w_D^p = 0.5(|w_{l+1}^m| + w_{l+1}^m); \quad w_D^n = 0.5(-|w_{l+1}^m| + w_{l+1}^m), \tag{I.33}$$

and afterwards flux through the surface follows,

$$(wS)_D^m = w_D^p S_{l+1}^m + w_D^n S_l^m \tag{I.34}$$

Eqs. (I.32) and (I.34) shows that the formulas for the flux at the upper and lower surface are very similar, and it is sufficient to change index in eqs. (I.31) and (I.32) from l to l+1 to derive eq. (I.34). Therefore, in ensuing consideration we shall only develop formulas for the upper surface of the control volume.

Now, from eq. (I.30) the updated value of salinity can be calculated by the following finite difference form

$$\frac{S_l^{m+1} - S_l^m}{T} = -\frac{(wS)_U^m - (wS)_D^m}{\Delta z} = -\frac{(wS)_U^m - (wS)_D^m}{HZ(l)}$$
$$= -\frac{FL(l) - FL(l+1)}{HZ(l)} \tag{I.35}$$

Here $FL(l)$ denotes the flux through the upper surface and $FL(l+1)$ is the flux through the lower surface (see Fig. 9).

It is important to notice again, that concentration (salinity) and vertical velocity (w) are not located at the same grid point. Therefore index (l) describes two different points separated by half of the grid distance ($HZ(L)/2$). As usually, the beginning of the system of coordinate is taken at the free surface with z axis pointing upward. Space integration index l is positive and increases from the surface towards the bottom, but the positive z direction is still upward. The index starts from 1 and usually in computations runs up to LZZ. To account for the boundary conditions at the ocean surface and at the bottom the flux is set to zero. The first grid point is located in the atmosphere and the last one is in the bottom, therefore the computations actually proceeds from L=2 to L=LZZ-1.

Problem No. 6, computation of the convective transport by the first order scheme using fluxes and upstream downstream spatial discretization

From eq. (I.35) the formula to update salinity in time follows,

$$SN(L) = SO(L) - (FL(L) - FL(L+1)) * TOH(L) \tag{I.36}$$

This formulation is related to the problem No.1, but above formulation conserves flux.

Beneath the FORTRAN program (FLUXDDIR.F) is written to account for the variable velocity in space (w_l) and the variable space step $HZ(L)$. The time step is $T = 1000$s. There are 100 grid points in this domain. Initial salinity distribution is 0.6 from 1 to 49 grid point and 0.2 from 50 to 100 grid point. The old value of salinity S_l^m is called in the program SO and the new value S_l^{m+1} is named SN. The velocity is constant and is set to -1.0×10^{-4}.

```
C PROGRAM TO CALCULATE CONVECTION BY Fluxes using upstream
C DERIVATIVES PROGRAM FLUXDDIR.F
PARAMETER (LZZ=100)
REAL W(LZZ),SO(LZZ),SN(LZZ),HZ(LZZ),TOH(LZZ)
REAL FL(LZZ)
T=1000. ! TIME STEP
MON=1000 !TOTAL NUMBER OF TIME STEPS
DO L=1,LZZ
HZ(L)=5.-2.5*L/lzz ! SPACE STEP is variable
TOH(L)=T/HZ(L) ! COMBINATION OF TIME STEP AND SPACE STEP
END DO
C INITIAL DISTRIBUTION of salinity
DO L=1,49
SO(L)=0.6
END DO
DO L=50,LZZ
SO(L)=0.2
END DO
OPEN(UNIT=3,NAME='SOZ1.VER',STATUS='UKNOWN')
C SET VERTICAL VELOCITY
DO L=1,LZZ
W(L)=-1.0E-4
END DO
C START TIME LOOP
50 N=N+1
C WRITE SALINITY AT TIME STEP 1, 100, 500, 990.
IF(N.EQ.1.OR.N.EQ.100.OR.N.EQ.500.OR.N.EQ.990)THEN
WRITE(3,*)(SO(L),L=1,LZZ)
END IF
C CALCULATE NEW VALUE OF flux FL
FL(2)=0. ! FLUX AT SURFACE
FL(LZZ)=0. ! FLUX AT BOTTOM
DO L=3,LZZ-1
wpL=0.5*(w(l)+abs(w(l))) ! POSITIVE VELOCITY
wnL=0.5*(w(l)-abs(w(l))) ! NEGATIVE VELOCITY
```

```
FL(L)=WPL*SO(L)+WNL*SO(L-1) ! FLUX THROUGH UPPER SURFACE
END DO
C NEW SALINITY SN
DO L=2,LZZ-1
SN(L)=SO(L)-(FL(L)-FL(L+1))*TOH(L)
END DO
C KEEP BOUNDARY VALUES
SN(LZZ-1)=0.2
SN(2)=0.6
C BOUNDARY CONDITIONS SET EQUAL SALINITY AT TWO POINTS
C SO THAT THE FLUX IS ZERO AT THE BOUNDARIES.
sN(LZZ)=SN(LZZ-1)
SN(1)=SN(2)
C CHANGE VARIABLES
DO L=1,LZZ
SO(L)=SN(L)
END DO
C STOP COMPUTATION AT TIME STEP MON
IF(N.EQ.MON) GO TO 18
C RETURN BACK TO 50 TO REPEAT TIME LOOP
GO TO 50
18 CONTINUE
END
```

The results of computation are depicted are exactly the same as in Fig. 2. Initial jump of salinity propagates downstream and slowly changes its shape due to numerical diffusion. (see explanation in NM given on page 52).

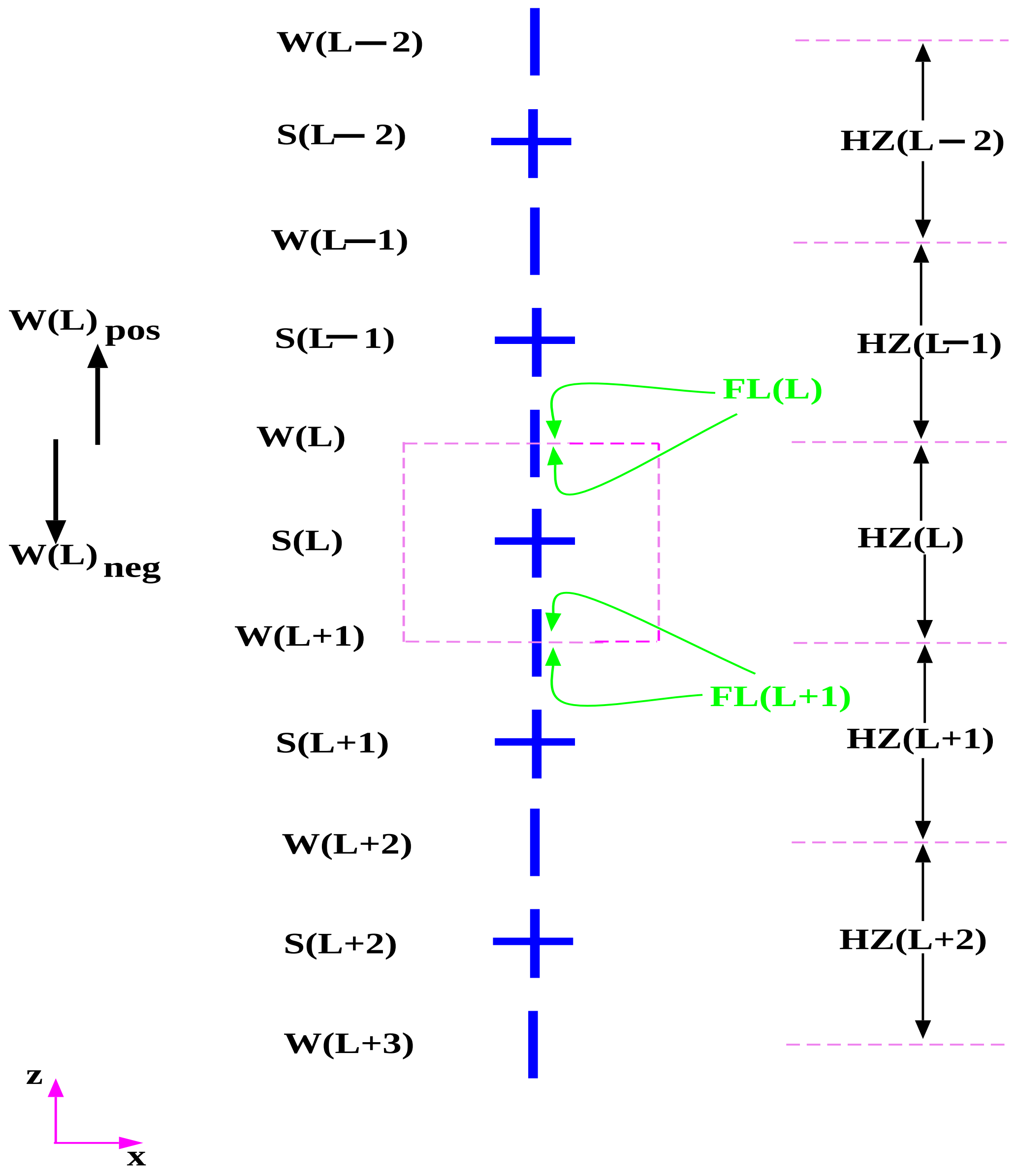


Fig. 9: Grid point distribution for integration of the transport equation along vertical direction. Square box denotes control volume.

Problem No. 7, computation of the convection by second order scheme using fluxes and upstream downstream spatial discretization

A forward in time upstream convection/advection scheme described above is based on conservative principle and well preserves positivity. Unfortunately, it also display strong numerical dissipation, hence the sharp front in the salinity distribution at the initial time becomes smoothed in the process of computation. To derive higher order schemes we continue to use the upstream approach. The flux at the cell's boundary will be defined by the salinity which will be derived through linear and polynomial fitting. In the upstream formulation of eq. (I.27) for the positive velocity it is assumed that flux of salinity through the upper surface carries the value of salinity from the nearest point S(L). For the negative velocity the flux is based on salinity S(L-1). To increase order of approximation one needs to know these values at the future time preferably at time step m+1/2. For this purposethe method of characteristics will be used. Equation (I.27) has such property that S is constant along any line defined by $z-wt=const$. Let $w(L)$ is positive velocity and time step is T. The value of salinity which is at the distance Tw from the upper surface of the cell will be the future value of salinity at the upper surface after time step T elapses. To predict value of salinity for the half time step the distance should be 0.5Tw. Thus the approach is straightforward: approximate linearly the value of salinity at the distance 0.5T$(w(L)+|w(L)|)$ from the upper surface for the positive velocity and at the distance 0.5T$(w(L)-|w(L)|)$ for the negative velocity (Fig. 10). Salinity at the upper surface for the positive velocity ($S_{UP}^{m+1/2}$) is

$$S_{UP}^{m+1/2}=\frac{S(L)[0.5HZ(L-1)+0.5Tw(L)]+S(L-1)[0.5HZ(L)-0.5Tw(L)]}{0.5[HZ(L-1)+HZ(L)]} \tag{I.37}$$

Similar approach can be used to define salinity at the upper surface for the negative velocity ($S_{UN}^{m+1/2}$)

$$S_{UN}^{m+1/2}=\frac{S(L)[0.5HZ(L-1)+0.5Tw(L)]+S(L-1)[0.5HZ(L)-0.5Tw(L)]}{0.5[HZ(L-1)+HZ(L)]} \tag{I.38}$$

The salt flux through the upper surface can be expressed similarly to eq. (I.32)

$$FL^{m+1/2}(L)=(wS)_U^{m+1/2}=w_U^pS_{UP}^{m+1/2}+w_U^nS_{UN}^{m+1/2} \tag{I.39}$$

To obtain flux at the lower surface of the control volume it is sufficient to change index L in the above formula to $L+1$ and define velocities at the lower surface accordingly to eq. (I.33). New value of salinity at the L grid point can now be defined based on the new flux formulation

$$\frac{S_l^{m+1}-S_l^m}{T}=-\frac{(wS)_U^{m+1/2}-(wS)_D^{m+1/2}}{\Delta z}=-\frac{(wS)_U^{m+1/2}-(wS)_D^{m+1/2}}{HZ(l)}$$

$$= -\frac{FL^{m+1/2}(l) - FL^{m+1/2}(l+1)}{HZ(l)} \tag{I.40}$$

From eq. (I.40) the formula to update salinity in time follows,

$$SN(L) = SO(L) - (FL(L) - FL(L+1)) * TOH(L) \tag{I.41}$$

Beneath the FORTRAN program (FLUXNEW.F) is written to account for the variable velocity in space (w_l) and the variable space step $HZ(L)$. The time step is $T = 1000$s. There are 100 grid points in this domain. Initial salinity distribution is 0.6 from 1 to 49 grid point and 0.2 from 50 to 100 grid point. The old value of salinity S_l^m is called in the program SO and the new value S_l^{m+1} is named SN. Linearly fitted value of salinity at the upper surface is called SLU The velocity is constant and is set to -1.0×10^{-4}.

```
C PROGRAM TO CALCULATE CONVECTION BY Fluxes using upstream
c DERIVATIVES SECOND ORDER OF APPROXIMATION
C PROGRAM FLUXNEW.F
PARAMETER (LZZ=100)
REAL W(LZZ),SO(LZZ),SN(LZZ),HZ(LZZ),TOH(LZZ)
REAL FL(LZZ)
T=1000. ! TIME STEP
MON=1000 !TOTAL NUMBER OF TIME STEPS
DO L=1,LZZ
HZ(L)=5.-2.5*L/lzz ! SPACE STEP
TOH(L)=T/HZ(L)
END DO
C INITIAL DISTRIBUTION
DO L=1,49
SO(L)=0.6
END DO
DO L=50,LZZ
SO(L)=0.2
END DO
OPEN(UNIT=3,NAME='SOZ.VER',STATUS='UKNOWN')
C SET VERTICAL VELOCITY
DO L=1,LZZ
W(L)=-1.0E-4
END DO
C START TIME LOOP
50 N=N+1
C WRITE SALINITY AT TIME STEP 2, 100, 500, 990.
```

```
IF(N.EQ.1.OR.N.EQ.100.OR.N.EQ.500.OR.N.EQ.990)THEN
WRITE(3,*)(SO(L),L=1,LZZ)
END IF
c TOH=T/HS
C CALCULATE NEW VALUE OF SALINITY SN
FL(2)=0.
FL(LZZ)=0.
DO L=3,LZZ-1
C LINEAR APPROXIMATION OF SALINITY AT THE UPPER SURFACE
OF A CONTROL VOLUME
HSU=0.5*(HZ(L)+HZ(L-1))
wpL=0.5*(w(l)+abs(w(l)))
wnL=0.5*(w(l)-abs(w(l)))
SLU1=0.5*(HZ(L-1)+W(L)*T)*SO(L)
SLU2=0.5*(HZ(L)-W(L)*T)*SO(L-1)
SLU=(SLU1+SLU2)/HSU
FL(L)=(WPL*SLU+WNL*SLU)
END DO
DO L=2,LZZ-1
SN(L)=SO(L)-(FL(L)-FL(L+1))*TOH(L)
END DO
C KEEP BOUNDARY CONDITIONS
SN(LZZ-1)=0.2
SN(2)=0.6
SN(LZZ)=0.2
SN(1)=0.6
C CHANGE VARIABLES
DO L=1,LZZ
SO(L)=SN(L)
END DO
C STOP COMPUTATION AT TIME STEP MON
IF(N.EQ.MON) GO TO 18
C RETURN BACK TO 50 TO REPEAT TIME LOOP
GO TO 50
18 CONTINUE
END
```

The computed salinity distribution is depicted in Fig.11. Broken lines depict solution by the first order method and continuous lines by the second order method. Although the second order method reproduces front much better in comparison to the first order method it does generate parasitic short waves which appear to grow in time.

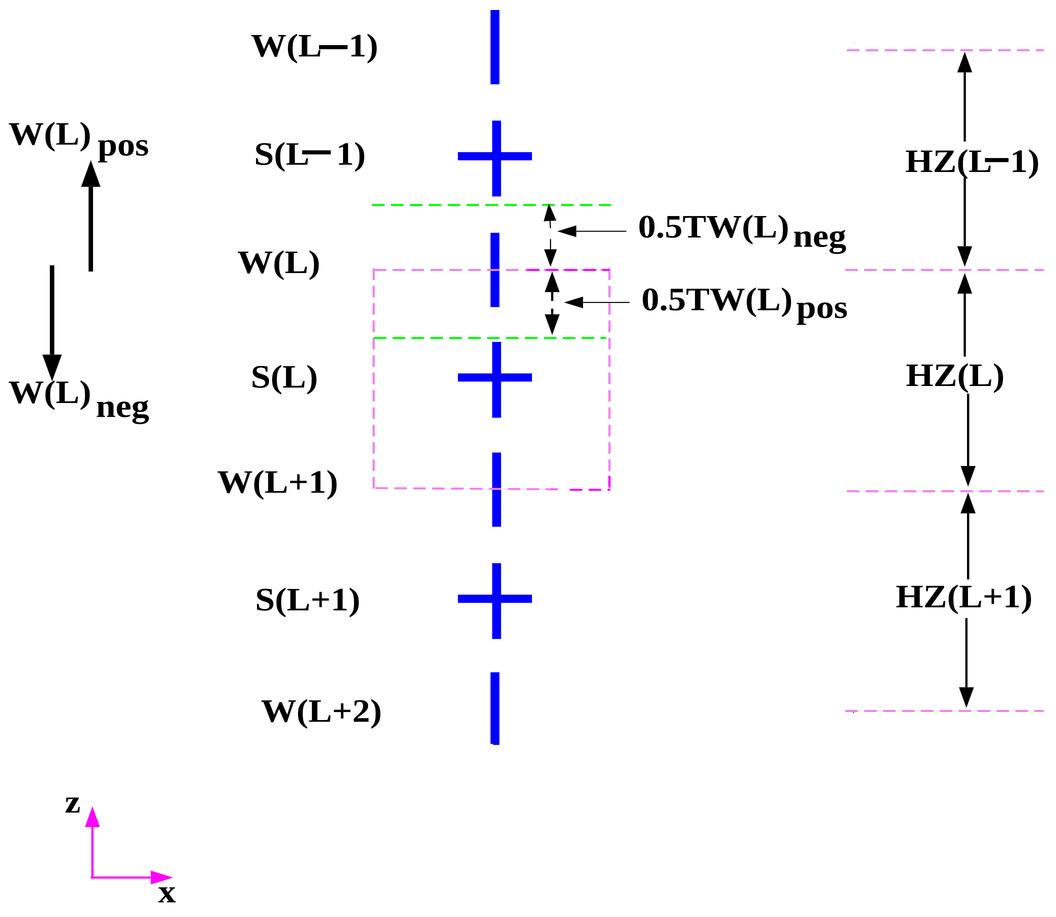
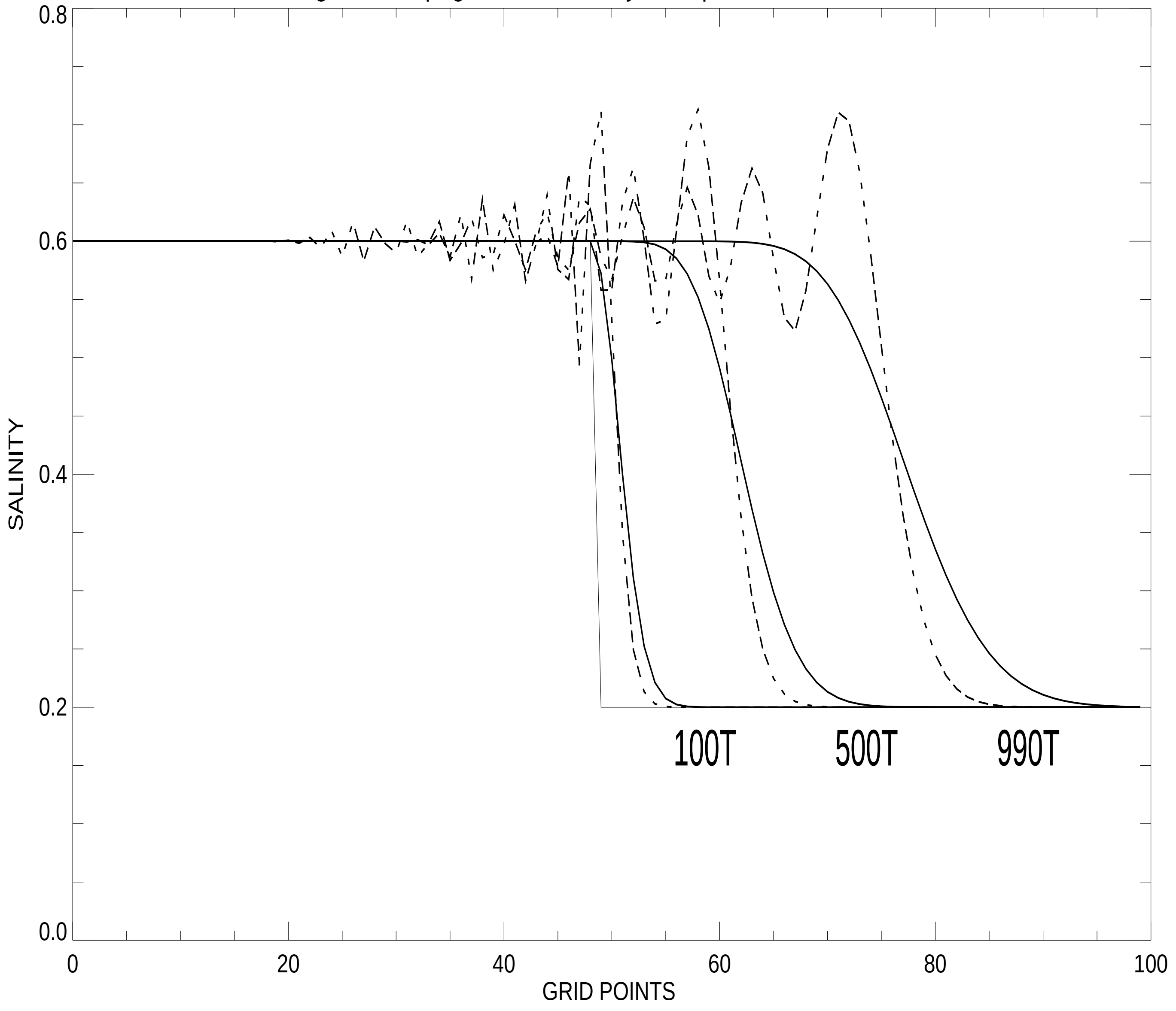


Fig. 10: Grid point distribution for integration of the transport equation by the second order scheme using upstream/downstream spatial discretization.

Fig.I.11: Propagation of Front by the Upstream Methods



Problem No. 8, computation of the convection by third order scheme using fluxes and upstream downstream spatial discretization

Second order scheme developed above shows improvement over the first order scheme, therefore we shall continue along this approach by fitting better approximation to the concentration at the surface of the control volume. For this purpose an upstream-biased stencil of three concentration points will be used. For the positive velocity at the upper surface three salinity points, namely: S(l-1), S(l) and S(l+1) are used to predict salinity for the half time step ahead at the upper surface (Fig. 9). Since this salinity at time step m is located at the distance 0.5Tw(l) from the upper surface of the control volume (Fig. 10), one can use the three above mentioned grid points to fit salinity at this location. A Lagrangian polynomial will be fit to this discrete set of data. The general form of three point polynomial can be written in the following way,

$$S_{UP}^{m+1/2} = P_{2,0}S_{l+1} + P_{2,1}S_l + P_{2,2}S_{l-1} \tag{I.42}$$

Here

$$P_{2,0} = \frac{(z-z_1)(z-z_2)}{(z_0-z_1)(z_0-z_2)}$$

$$P_{2,1} = \frac{(z-z_0)(z-z_2)}{(z_1-z_0)(z_1-z_2)}$$

$$P_{2,2} = \frac{(z-z_0)(z-z_1)}{(z_2-z_0)(z_2-z_1)} \tag{I.43}$$

To relate this general formula to our geometry a local system of coordinate is introduced. Beginning of this system is set at the S(l+1) grid point. Thus $z_0 = 0$, $z_1 = 0.5(HZ(l+1) + HZ(l))$ and $z_2 = z_1 + 0.5(HZ(l) + HZ(l-1))$. Variable distance $z = z_1 + 0.5(HZ(l) - w(l)T)$.

For the negative velocity at the upper surface the upstream-biased stencil is based on S(l), S(l-1) and S(l-2) grid points (Figs. 9 and 10). Thus salinity due to the negative velocity at the upper surface is defined as,

$$S_{UN}^{m+1/2} = P_{2,0}S_l + P_{2,1}S_{l-1} + P_{2,2}S_{l-2} \tag{I.44}$$

Here Lagrangian polynomial is again defined by the general formula eq. (I.42), but the coefficients will follow from the new (local) coordinate system. This time the beginning is chosen at the point S(l) and $z_0 = 0$, $z_1 = 0.5(HZ(l) + HZ(l-1))$ and $z_2 = z_1 + 0.5(HZ(l-1) + HZ(l-2))$. Variable distance $z = 0.5(HZ(l) - w(l)T)$.

Using these new expressions the salt flux through the upper surface can be stipulated similarly to eq. (I.32),

$$FL^{m+1/2}(L) = (wS)_U^{m+1/2} = w_U^p S_{UP}^{m+1/2} + w_U^n S_{UN}^{m+1/2} \tag{I.45}$$

To obtain flux at the lower surface of the control volume it is sufficient to change index L in the formulas for the upper surface to $L+1$ and define velocities at the lower surface accordingly to eq. (I.33). Temporal change of salinity at the L grid point can now be defined based on this new flux formulation

$$\frac{S_l^{m+1} - S_l^m}{T} = -\frac{(wS)_U^{m+1/2} - (wS)_D^{m+1/2}}{\Delta z} = -\frac{(wS)_U^{m+1/2} - (wS)_D^{m+1/2}}{HZ(l)}$$

$$= -\frac{FL^{m+1/2}(l) - FL^{m+1/2}(l+1)}{HZ(l)} \tag{I.46}$$

From eq. (I.46) the updated salinity follows,

$$SN(L) = SO(L) - (FL(L) - FL(L+1)) * TOH(L) \tag{I.47}$$

Beneath the FORTRAN program (FLUX-LAGRA.F) is written to account for the variable velocity in space (w_l) and the variable space step $HZ(L)$. The time step is $T = 1000$s. There are 100 grid points in this domain. Initial salinity distribution is 0.6 from 1 to 49 grid point and 0.2 from 50 to 100 grid point. The old value of salinity S_l^m is called in the program SO and the new value S_l^{m+1} is named SN. Fitted value of salinity at the upper surface for the positive velocity is called SLUP and for the negative velocity is called SLUN. Local coordinates are as follows: $z = ZU$, $z_1 = Z1U$ and $z_2 = Z2U$.

Velocity is constant and is set to -1.0×10^{-4}. Three terms of the polynomial in eqs.(I.42) and (I.44) are expressed in the program as,

$$P_{2,0} = A1{*}B1/(Z1U{*}Z2U); P_{2,1} = ZU{*}B1/(Z1U{*}C1); P_{2,2} = ZU{*}A1/(Z2U{*}C1)$$

Here

$$A1 = ZU - Z1U; \qquad B1 = ZU - Z2U; \qquad C1 = Z1U - Z2U$$

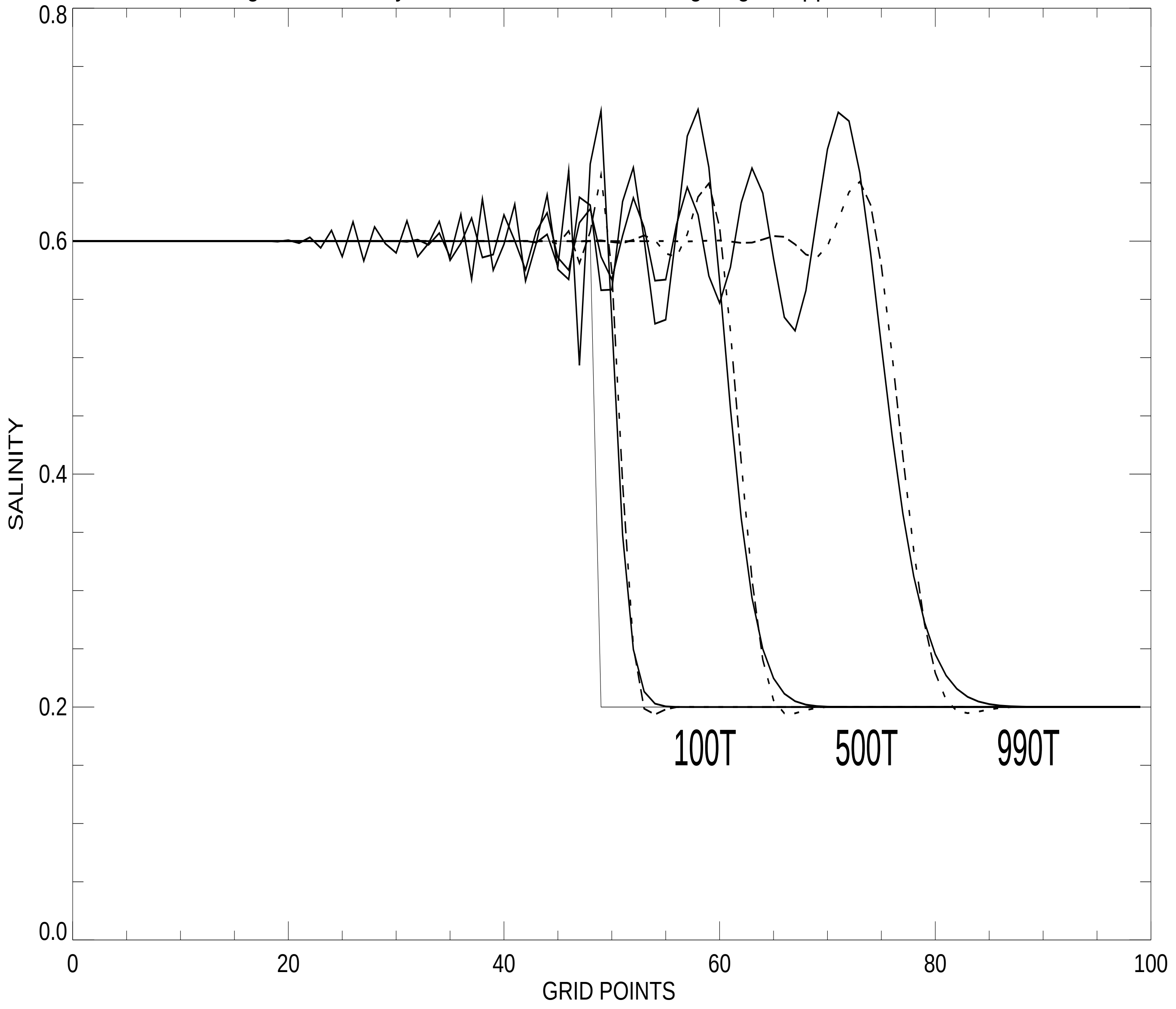
```
C PROGRAM TO CALCULATE CONVECTION BY upwind DERIVATIVES
C AND BY FITTING SALINITY AT CONTROL SURFACE BY
C LAGRANGE'S POLYNOMIAL
C PROGRAM FLUX-LAGRA.F
PARAMETER (LZZ=100)
REAL W(LZZ+1),SO(LZZ),SN(LZZ),HZ(LZZ),TOH(LZZ)
REAL FL(LZZ+1)
T=1000. ! TIME STEP
MON=1000 !TOTAL NUMBER OF TIME STEPS
```

```
DO L=1,LZZ
HZ(L)=5.-2.5*L/lzz ! SPACE STEP
TOH(L)=T/HZ(L)
END DO
C BECAUSE HZ(L)=HS=CONST
c HS=5.
C INITIAL DISTRIBUTION
DO L=1,49
SO(L)=0.6
END DO
DO L=50,LZZ
SO(L)=0.2
END DO
OPEN(UNIT=3,NAME='SOZ1.VER',STATUS='UKNOWN')
C SET VERTICAL VELOCITY
DO L=1,LZZ
W(L)=-1.0E-4
END DO
C START TIME LOOP
50 N=N+1
C WRITE SALINITY AT TIME STEP 2, 100, 500, 990.
IF(N.EQ.1.OR.N.EQ.100.OR.N.EQ.500.OR.N.EQ.990)THEN
WRITE(3,*)(SO(L),L=1,LZZ)
END IF
C CALCULATE NEW VALUE OF SALINITY SN.
C FLUX AT L=1 AND LZZ+1 IS 0.
FL(2)=0
FL(LZZ)=0
DO L=3,LZZ-1
WPU=.5*(W(L)+ABS(W(L)))
WNU=.5*(W(L)-ABS(W(L)))
c positive velocity at upper boundary
c beggining of local coordinate for lagrange polynomial
c at z=hz(l+1)/2=0
C Z=ZU, Z1=Z1U; Z2=Z2U
z1u=0.5*(HZ(L)+HZ(L+1))
Z2U=0.5*(HZ(L)+HZ(L+1))+HZ(L)
ZU=HZ(L)+0.5*(HZ(L+1)-WPU*T)
A1=ZU-Z1U
B1=ZU-Z2U
C1=Z1U-Z2U
SLU1=SO(L+1)*A1*B1/(Z1U*Z2U)
```

```
SLU2=SO(L)*ZU*B1/(Z1U*C1)
SLU3=-SO(L-1)*ZU*A1/(Z2U*C1)
SLUP=SLU1+SLU2+SLU3
z1u=0.5*(HZ(L)+HZ(L-1))
Z2U=0.5*(HZ(L)+HZ(L-2))+HZ(L-1)
ZU=0.5*(HZ(L)-WNU*T)
A1=ZU-Z1U
B1=ZU-Z2U
C1=Z1U-Z2U
SLU1=SO(L)*A1*B1/(Z1U*Z2U)
SLU2=SO(L-1)*ZU*B1/(Z1U*C1)
SLU3=-SO(L-2)*ZU*A1/(Z2U*C1)
SLUN=SLU1+SLU2+SLU3
c positive velocity at lower boundary
c beggining of local coordinate for lagrange polynomial
c at z=hz(l+2)/2 =0
FL(L)=WPU*SLUP+WNU*SLUN
END DO
do l=2,LZZ-1
SN(L)=SO(L)-(FL(L)-FL(L+1))*TOH(L)
end do
C KEEP BOUNDARY CONDITIONS
SN(LZZ-1)=0.2 ! this conditions to keep value 0.2 at the right
c side otherwise concentration at the right side will grow because flux
c through the boundary is zero. (this comment for negative velocity)
SN(2)=0.6 ! otherwise 0.6 will travel away and 0 will start to
c generate.
C BELOW NEED TO KEEP OUTSIDE SALINITY EQUAL TO INSIDE
SALINITY
sN(LZZ)=SN(LZZ-1)
SN(1)=SN(2)
C CHANGE VARIABLES
DO L=1,LZZ
SO(L)=SN(L)
END DO
C STOP COMPUTATION AT TIME STEP MON
IF(N.EQ.MON) GO TO 18
C RETURN BACK TO 50 TO REPEAT TIME LOOP
GO TO 50
18 CONTINUE
END
```

The result of computations are given in Fig.12 for the second-order (continuous line) and for the third order (broken lines). Two improvements can be noticed, the front is better reproduced and short wave oscillations have been limited to one spike whose amplitude is constant in time.

Fig.I.12: Front by third order method and Lagrangian approximation



Problem No. 9, computation of advection by the third order scheme using fluxes and the upstream/downstream spatial discretization (geometry along x direction).

A second order scheme are written for the geometry along the vertical direction (index of integration is diminishing along positive z direction). In the regular geometry an index of enumeration is increased along the positive direction. Above formulas we will rewrite along x axis. Let consider the flux through the right surface of a control volume. For the positive velocity at the right-hand surface the three salinity points, namely: S(j-1), S(j) and S(j+1) are used to predict salinity which will occur half time step later at the right-hand surface. Since this salinity at the time step m is located at the distance 0.5Tu(j+1) from the surface, the three above mentioned grid points can be used to fit salinity at this location. A Lagrangian polynomial will be fit to this discrete set of data.

$$S_{RP}^{m+1/2} = P_{2,0}S_{j-1} + P_{2,1}S_j + P_{2,2}S_{j+1} \tag{I.48}$$

Here

$$P_{2,0} = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}$$

$$P_{2,1} = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$$

$$P_{2,2} = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \tag{I.49}$$

To relate this general formula to our geometry a local system of coordinate is introduced. Beginning of this system is set at the S(j-1) grid point. Thus $x_0 = 0$, $x_1 = HX$ and $x_2 = 2HX$. Variable distance $x = x_1 + 0.5(HX - u(j+1)T)$. Here we assume that the space step of numerical integration along x direction is constant and equal to HX.

For the negative velocity at the right-hand surface the upstream-biased stencil is based on S(j), S(j+1) and S(J+2) grid points. Thus salinity due to negative velocity at the right hand surface is defined as,

$$S_{RN}^{m+1/2} = P_{2,0}S_j + P_{2,1}S_{j+1} + P_{2,2}S_{j+2} \tag{I.50}$$

Here Lagrangian polynomial is again defined by the general formula eq.(I.42), but the coefficients will follow from the new local coordinate system. This time, the beginning is chosen at the point S(j) and $x_0 = 0$, $x_1 = HX$ and $x_2 = 2HX$. Variable distance $x = 0.5(HX - u(j+1)T)$. Using these new expressions the salt flux through the right-hand surface can be stipulated similarly to eq. (I.32),

$$FL^{m+1/2}(j+1) = (uS)_R^{m+1/2} = u_R^p S_{RP}^{m+1/2} + u_R^n S_{RN}^{m+1/2} \tag{I.51}$$

To obtain flux at the left-hand surface of the control volume it is sufficient to change index j in the formulas for the righ-hand surface to $j-1$ and define velocities at the left-hand surface.

New value of salinity at the j grid point can now be defined based on the new flux formulation

$$\frac{S_j^{m+1} - S_j^m}{T} = -\frac{(uS)_R^{m+1/2} - (uS)_L^{m+1/2}}{\Delta x} = -\frac{(uS)_R^{m+1/2} - (uS)_L^{m+1/2}}{HX}$$

$$= -\frac{FL^{m+1/2}(j+1) - FL^{m+1/2}(j)}{HX} \tag{I.52}$$

From eq. (I.52) formula to update salinity follows,

$$SN(j) = SO(j) - (FL(j+1) - FL(j)) * T/HX \tag{I.53}$$

We will not construct FORTRAN program to compute advection because program (FLUX-LAGRA.F) can be easily adapted for this task.

Problem No. 10, computation of convection by Flux-Corrected Method (FCT)

The flux–corrected method is well described in the Ch.II, Sec. 9.5. It is applied infrequently in the large oceanic models because it needs a few step to achieve result. These steps when repeated many times in the large models make big difference in the computer time required for solving problems.

The main portion of the FCT is limiter or filter which produces ripple-free solution. The limiter is given by eq.(2.213) from NM. It works by limiting an antidifusive flux, i.e., flux which is a difference of the two fluxes, obtained by two different method. One method is the low order method (usually related to directional derivatives), and second method is the high order method. The latter reproduces well salinity jump but produces also dispersive short waves. In the NM the high order numerical schemes in time are based on the symmetrical central derivative. Here we use the high accuracy the third order scheme.

The FCT algorithm constructs the net fluxes point by point as a weighted average of a low-order scheme and a high-order scheme. This weighting is done in such manner as to maximize the use of the high order scheme without overshooting or undershooting. The FCT algorithm can be broken down into six steps.

1. Compute F_l^L – flux by the low order scheme. The purpose here is to derive a positive definite, or sometimes called monotonic, solution. For our purpose it is only important that the monotonic solution is ripple-free. Solution is obtained by the method of upstream discretization described by eqs. (I.31) –(I.35)

2. Compute F_l^H – flux by the high order scheme. Here the third order approximation given by eqs.(I.37)–(I.40) is used.

3. Introduce an antidiffusive flux as

$$A_l = F_l^H - F_l^L \tag{I.54}$$

4. Compute low order updated solution (SL)

$$SL_l^{m+1} = S_l^m - \frac{T}{h}(F_l^L - F_{l+1}^L) \tag{I.55}$$

5. Limit antidiffusion fluxes in such a way that the new concentration (S^{m+1}) is ripple-free. This is accomplished through limiter L_l as

$$A_l^L = L_l A_l, \quad 0 \le L_l \le 1 \tag{I.56}$$

6. Use the limited antidiffusive flux to derive concentration S_l^{m+1}

$$S_l^{m+1} = SL_l^{m+1} - \frac{T}{h}(A_l^L - A_{l+1}^L) \tag{I.57}$$

In the construction of the finite-difference formulas we use the staggered grid depicted in Fig. 9.

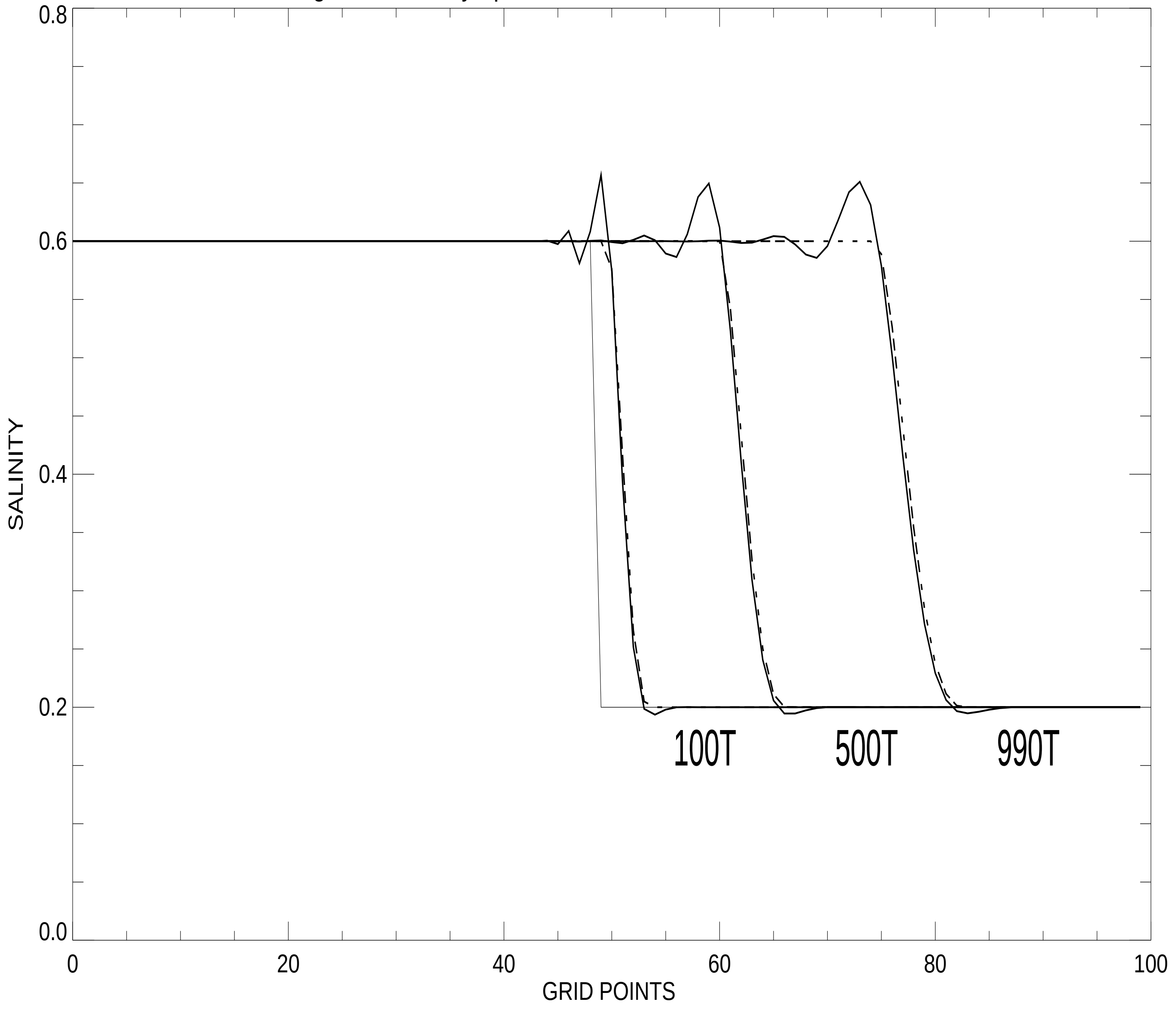
Basic formulas for the low order and high order solution were constructed above. Below is given Fortran program fluxall.f. New value of salinity calculated by the low order scheme is called SNL. Flux obtained by the low order scheme is denoted as FLL and flux calculated by the high order scheme is FL. Antidiffusive flux is AUN.

```
C PROGRAM TO CALCULATE CONVECTION BY Flux Corrected Method
C PROGRAM FLUXALL.F
PARAMETER (LZZ=100)
REAL W(LZZ),SO(LZZ),SN(LZZ),SNL(LZZ),HZ(LZZ),TOH(LZZ)
REAL FLL(LZZ),FL(LZZ),AU(LZZ)
REAL TOH1(LZZ)
T=1000. ! TIME STEP
MON=1000 !TOTAL NUMBER OF TIME STEPS
DO L=1,LZZ
HZ(L)=5.-2.5*L/lzz ! SPACE STEP
TOH(L)=T/HZ(L)
TOH1(L)=0.5*(HZ(L)+HZ(L-1))/T
END DO
C INITIAL DISTRIBUTION
DO L=1,49
```

```
SO(L)=0.6
END DO
DO L=50,LZZ
SO(L)=0.2
END DO
OPEN(UNIT=3,NAME='SOZ1.VER',STATUS='UKNOWN')
C SET VERTICAL VELOCITY
DO L=1,LZZ
W(L)=-1.0E-4
END DO
C START TIME LOOP
50 N=N+1
C WRITE SALINITY AT TIME STEP 1, 100, 500, 990.
IF(N.EQ.1.OR.N.EQ.100.OR.N.EQ.500.OR.N.EQ.990)THEN
WRITE(3,*)(SO(L),L=1,LZZ)
END IF
c TOH=T/HS
C CALCULATE NEW VALUE OF SALINITY SN BY THE LOW ORDER
SCHEME
FLL(LZZ)=0.
DO L=2,LZZ-1
WPU=.5*(W(L)+ABS(W(L)))
WNU=.5*(W(L)-ABS(W(L)))
FLL(L)=WPU*SO(L)+WNU*SO(L-1)
END DO
DO L=2,LZZ-1
SNL(L)=SO(L)-(FLL(L)-FLL(L+1))*TOH(L)
END DO
FL(2)=0.
FL(LZZ)=0.
DO L=3,LZZ-1
WPU=.5*(W(L)+ABS(W(L)))
WNU=.5*(W(L)-ABS(W(L)))
c positive velocity at upper boundary
c beggining of local coordinate for lagrange polynomial
c at z=hz(l+1)/2=0
z1u=0.5*(HZ(L)+HZ(L+1))
Z2U=0.5*(HZ(L)+HZ(L+1))+HZ(L)
ZU=HZ(L)+0.5*(HZ(L+1)-WPU*T)
A1=ZU-Z1U
B1=ZU-Z2U
C1=Z1U-Z2U
```

```
SLU1=SO(L+1)*A1*B1/(Z1U*Z2U)
SLU2=SO(L)*ZU*B1/(Z1U*C1)
SLU3=-SO(L-1)*ZU*A1/(Z2U*C1)
SLUP=SLU1+SLU2+SLU3
C NEGATIVE VELOCITY AT UPPER BOUNDARY
c beggining of local coordinate for lagrange polynomial
C AT THE POINT C(L),Z=0
z1u=0.5*(HZ(L)+HZ(L-1))
Z2U=0.5*(HZ(L)+HZ(L-2))+HZ(L-1)
ZU=0.5*(HZ(L)-WNU*T)
A1=ZU-Z1U
B1=ZU-Z2U
C1=Z1U-Z2U
SLU1=SO(L)*A1*B1/(Z1U*Z2U)
SLU2=SO(L-1)*ZU*B1/(Z1U*C1)
SLU3=-SO(L-2)*ZU*A1/(Z2U*C1)
SLUN=SLU1+SLU2+SLU3
FL(L)=WPU*SLUP+WNU*SLUN
C INTRODUCE ANTIDIFFUSIVE FLUX
C AT UPPER SURFACE
AUN=FL(L)-FLL(L)
C FILTERING FIKU-MIKU
A0=AUN
A1=SIGN(1.0,A0)
A2=A1*(SNL(L-2)-SNL(L-1))*TOH1(L-1)
A3=A1*(SNL(L)-SNL(L+1))*TOH1(L+1)
A4=ABS(AUN)
A5=AMIN1(A2,A3,A4)
AU(L)=A1*AMAX1(0.,A5)
END DO
AU(2)=0.
AU(LZZ)=0.
C NEW SALINITY
DO L=2,LZZ-1
SN(L)=SNL(L)-(AU(L)-AU(L+1))*TOH(L)
END DO
C KEEP BOUNDARY CONDITIONS
SN(LZZ)=0.2
SN(LZZ-1)=0.2
SN(1)=0.6
SN(2)=0.6
```

Fig.I.13: Front by upstream third order method and FCT



```
C CHANGE VARIABLES
DO L=1,LZZ
SO(L)=SN(L)
END DO
C STOP COMPUTATION AT TIME STEP MON
IF(N.EQ.MON) GO TO 18
C RETURN BACK TO 50 TO REPEAT TIME LOOP
GO TO 50
18 CONTINUE
END
```

The result of calculation by FCT (broken lines) and by third order scheme (continuous line) is given in Fig. 12. The FCT produced output without any parasitic waves.

CHAPTER II

TWO-DIMENSIONAL MODELS

1. Propagation in channel

We shall construct numerical algorithms for two dimensional problems described in the first part of Ch.III of NM. In order to identify the important steps in the constructing and analysis of a numerical scheme we shall consider first propagation of the long wave in a channel. These experiments are described in the Ch.III, Sec.3 of NM.

We shall consider the numerical solution of the equations of motion and continuity

$$\frac{\partial u}{\partial t} = -g\frac{\partial \zeta}{\partial x} \tag{II.1}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial}{\partial x}(Hu) \tag{II.2}$$

Solution of this system is usually searched by the two-time-level or by the three-time-level numerical schemes. For construction of the space derivatives in the equations (II.1) and (II.2), a space staggered grid (Figure 3.2 from NM) is usually used (Arakawa C grid). The two-time-level numerical scheme (3.46 from NM)

$$\frac{(u_j^{m+1} - u_j^m)}{T} = -g\frac{\zeta_j^m - \zeta_{j-1}^m}{h} \tag{II.3}$$

$$\frac{\zeta_j^{m+1} - \zeta_j^m}{T} = -\frac{(u_{j+1}^{m+1}H_{j+1} - u_j^{m+1}H_j)}{h} \tag{II.4}$$

is of the second order of approximation in space and only the first order in time. For the quick reference we shall call above the Fisher's scheme. The leapfrog (three-time-level scheme)

$$\frac{(u_j^{m+1} - u_j^{m-1})}{2T} = -g\frac{\zeta_j^m - \zeta_{j-1}^m}{h} \tag{II.5}$$

$$\frac{\zeta_j^{m+1} - \zeta_j^{m-1}}{2T} = -\frac{(u_{j+1}^{m}H_{j+1} - u_j^{m}H_j)}{h} \tag{II.6}$$

is of the second order of approximation in time and space.

The space-staggered grid given in Figure 3.2 from NM is used to construct the space derivatives in the above equations. The variables u and ζ are located in such a way that the second order of approximation in space is achieved. The depth is taken in the sea level points. The space step along the x direction is h. Index m stands for the time stepping. In the Fisher scheme the time step is T and in the leapfrog scheme the time step is $2T$. To demonstrate a distortion generated by the finite difference equations due to an approximation error, let us consider a simple case of a sinusoidal wave propagating over a long distance in the channel of constant depth. At the left end of the channel a sinusoidal wave is given as

$$\zeta = \zeta_0 \sin(\frac{2\pi t}{T_p}) \qquad \text{(II.7)}$$

Here the amplitude is ζ_0=100 cm, and the period is T_p=10 min. Propagation of this monochromatic wave toward the right end of the channel will be reproduced. The right end is open, and a radiating condition will be used so that the wave can propagate beyond the channel without reflection (eq. 3.45 from NM). At the left end of the channel (II.7) is applied for one period only; after that, the radiating condition is used as well. The channel is 2100 km long and 4077 m deep. The wave period under consideration is 10 min, which results in a 120-km wavelength. The time step of numerical integration will be taken in all experiments equal to 10 s. The space step is chosen equal to 12 km. The 12-km space grid sets 10 steps per wavelength (SPW). Such a resolution introduces numerical errors resulting in the wave dispersion.

Problem No. 1, solution by Fisher method

We start by solving above problem through the method given by the set of equations (II.3) and (II.4). The name of program is FISHER.F. Time step (T) is 10s and space step (DX) is equal to 12km. Therefore in the channel of 2100 km length there are 175 grid points. Two variables are considered namely sea level (Z - old value; ZN - new value), and velocity (U old value; UN new value). Space index is j, it runs from j=1 to j=je. At the left-hand open boundary a sinusoidal wave of 100 cm amplitude is given as: zn(1)=100.*sin(0.1047197*float(mo-1)). In this formula mo denotes index of time stepping. At the right-hand end of channel a radiating condition is used to calculate velocity. First, the new velocity UN is computed. After computing the new value of velocity, this value is introduced into the sea level computations because of stability considerations. The change of variables in loop 14 actually performs time stepping in which the new values become the old ones. Command GO TO 50 brings the whole process to the beginning of the time loop and the computations can be repeated again until mo=mon.

```
C Program fisher.f
```

```
C PROGRAM COMPUTE propagation in 1D channel
C by fisher method (eq. 3.46 from NM)
C program originated by Z.K. at IMS, Alaska
PARAMETER (je=175)
REAL U(je),UN(je),Z(je),ZN(je) ! Velocity and sea level
REAL H(je),H1(JE) ! Depth and total depth
open (unit=10,file='euler.dat',status='unknown')
c mon is total number of time steps
mon=1001
c if B=1 total depth is computed H1=H+Z
B=0.
G=982.
c time step and space step
t=10.
DX=12.e5 ! grid step in cm
C COEFFICIENTS FOR EQUATIONS
PL=2.0*T/DX
PLG=G*PL
c depth
do j=1,je
h1(j)=4.077e5
h(j)=h1(j)
end do
c start time loop
50 MO=MO+1
leu=leu+1
c input boundary for one period and afterwards radiating condition
zn(1)=-u(2)*sqrt(h(2)/g) ! Sea level from radiation condition
if(mo.gt.60)go to 22
zn(1)=100.*sin(0.1047197*float(mo-1)) ! Sea levl given for the 60
C time steps
22 continue
c radiating boundary at the very end of the channel
Un(je)=Z(je-1)*SQRT(G/H(je-1))
C compute velocity
DO80 J=2,je-1
UN(J)=U(J)-.5*PLG*(Z(J)-Z(J-1))
80 CONTINUE
c compute sea level
DO 120 J=2,je-1
ZN(J)=Z(J)+.5*PL*(UN(J)*(H1(J)+H1(J-1))/2.0-UN(J+1
2)*(H1(J+1)+H1(J))/2.)
```

```
120 CONTINUE
C CHANGE OLD VARIABLES TO NEW
DO 14 J=1,je
U(J)=UN(J)
H1(J)=H(J)+B*(ZN(J))
Z(J)=ZN(J)
14 CONTINUE
if(mo.eq.60.or.mo.eq.360.or.mo.eq.670.or.mo.eq.980)then
print *, mo
write (10,*)(zn(j),j=1,je-1)
end if
IF(MO.EQ.MON)GO TO 23 ! Finish computations
GO TO 50
23 CONTINUE
END
```

Problem No. 2, solution by leap-frog method

Now we solve the same problem by the leapfrog method described by eqs. (II.5) and (II.6). First we note that the time stepping is done through three time levels, therefore we introduce additional time level called respectively UO for the velocity, and ZO for the sea level. To begin computations in time by the leapfrog method two initial conditions are needed. One way to resolve this problem is to use a two-time-level numerical scheme to derive one additional initial value. Therefore, in the program we have constructed below, both leapfrog and Fisher method are used. Fisher scheme is used not only for generating an additional initial condition. Leapfrog scheme may generate two solutions. One of these is parasite solution which can obfuscate physical solution (see Ch.II. Sec.7 in NM). In a simple case of the constant depth, we consider here, the parasite solution is not present. It can be damped by calling Fisher scheme every 20 time steps or so. This option is built in the program LEAPFROG.F. All other notations are from the program fisher.f. One thing to remember is to scrutinize Fig. 3.5 from NM before starting computations. Although the leapfrog method requires double time step, due to change of the variables, the time stepping in the leapfrog and in the Fisher method is the same (see how old and intermediate variables are upgraded to the new ones in the loop 14 of the below program).

```
c program: leapfrog.f
C PROGRAM COMPUTES propagation in 1D channel
c by leap-frog method
C program originated by Z.K. at IMS, Alaska
PARAMETER (je=175)
C
```

```
REAL U(je),UN(je),Z(je),ZN(je)
REAL H(je),UO(je),ZO(je),H1(JE)
open (unit=10,file='lp10c5.dat',status='unknown')
C B SERVE TO CONTROL TOTAL DEPTH
C IF B=1.,TOTAL DEPTH H1=H+B*(Z+TE)
B=0.
mon=1001
c basic data
G=982.
c time step and space step
t=10.
DX=12.e5
c depth
do j=1,je
h1(j)=4.077e5
h(j)=h1(j)
end do
PL=2.0*T/DX ! notice double time step
c start time loop
50 MO=MO+1
leu=leu+1
c input boundary only for one period and then switch to
c radiating condirion
zn(1)=-u(2)*sqrt(h(2)/g)
if(mo.gt.60)go to 22
zn(1)=100.*sin(0.1047197*float(mo-1))
22 continue
c radiating boundary at the very end of the channel
Un(je)=Z(je-1)*SQRT(G/H(je-1))
C TO INITIALIZE COMPUTATION USE FISHER SCHEME
C OR MAY BE YOU WISH TO APPLY
C IT EVRY 20 STEPS TO REMOVE NON-PHYSICAL SOLUTION.
C FOR THIS PURPOSE COUNTER LEU IS INTRODUCED
c if(leu.eq.20)go to 99 ! For Fisher's restart
c IF(MO.EQ.1.or.mo.gt.1)GO TO 99
IF(MO.EQ.1)GO TO 99
C LEAP-FROG
DO8 J=2,je-1
c calculate velocity
UN(J)=UO(J)-PL*G*(Z(J)-Z(J-1))
8 CONTINUE
DO 12 J=2,je-1
```

```
ump=0.5*u(j)*(h1(j)+h1(j-1))
up1=0.5*u(j+1)*(h1(j)+h1(j+1))
C CALCULATE SEA LEVEL
ZN(J)=ZO(J)+PL*(ump-up1)
12 CONTINUE
C=========================================
C FISHER RESTART
GO TO 98 ! Skip Fisher
99 LEU=0
C CALCULATE VELOCITY
DO80 J=2,je-1
UN(J)=U(J)-.5*PL*G*(Z(J)-Z(J-1))
80 CONTINUE
C CALCULATE SEA LEVEL
DO 120 J=2,je-1
ZN(J)=Z(J)+.5*PL*(UN(J)*(H1(J)+H1(J-1))/2.0-UN(J+1
2)*(H1(J+1)+H1(J))/2.)
120 CONTINUE
98 CONTINUE
C CHANGE OLD VARIABLES TO NEW
DO 14 J=1,je
UO(J)=U(J)
U(J)=UN(J)
H1(J)=H(J)+B*(ZN(J))
ZO(J)=Z(J)
Z(J)=ZN(J)
14 CONTINUE
if(mo.eq.60.or.mo.eq.360.or.mo.eq.670.or.mo.eq.980)then
print *, mo
write (10,*)(zn(j),j=1,je-1)
end if
c stop calculation when mo=mon
IF(MO.EQ.MON)GO TO 23
GO TO 50
23 CONTINUE
END
```

The results of computations derived by the Fisher and leapfrog algorithms are depicted in Fig.II.1. In both solutions the wave propagating from the left end displays diminishing amplitude along the channel. It has also a tail of secondary waves following the main wave. These distortions are due to the numerical approximations we applied. In the governing equations (II.1) and (II.2) there is no friction or dispersion responsible for the results given in Fig.II.1. Comparing the upper and lower part of Fig.II.1 one can see that solutions are quite similar. Because leapfrog has second order approximation in time and Fisher's scheme is only of the first order, one can conclude that distortions seen in Fig.II.1 are due mainly to the space resolution.

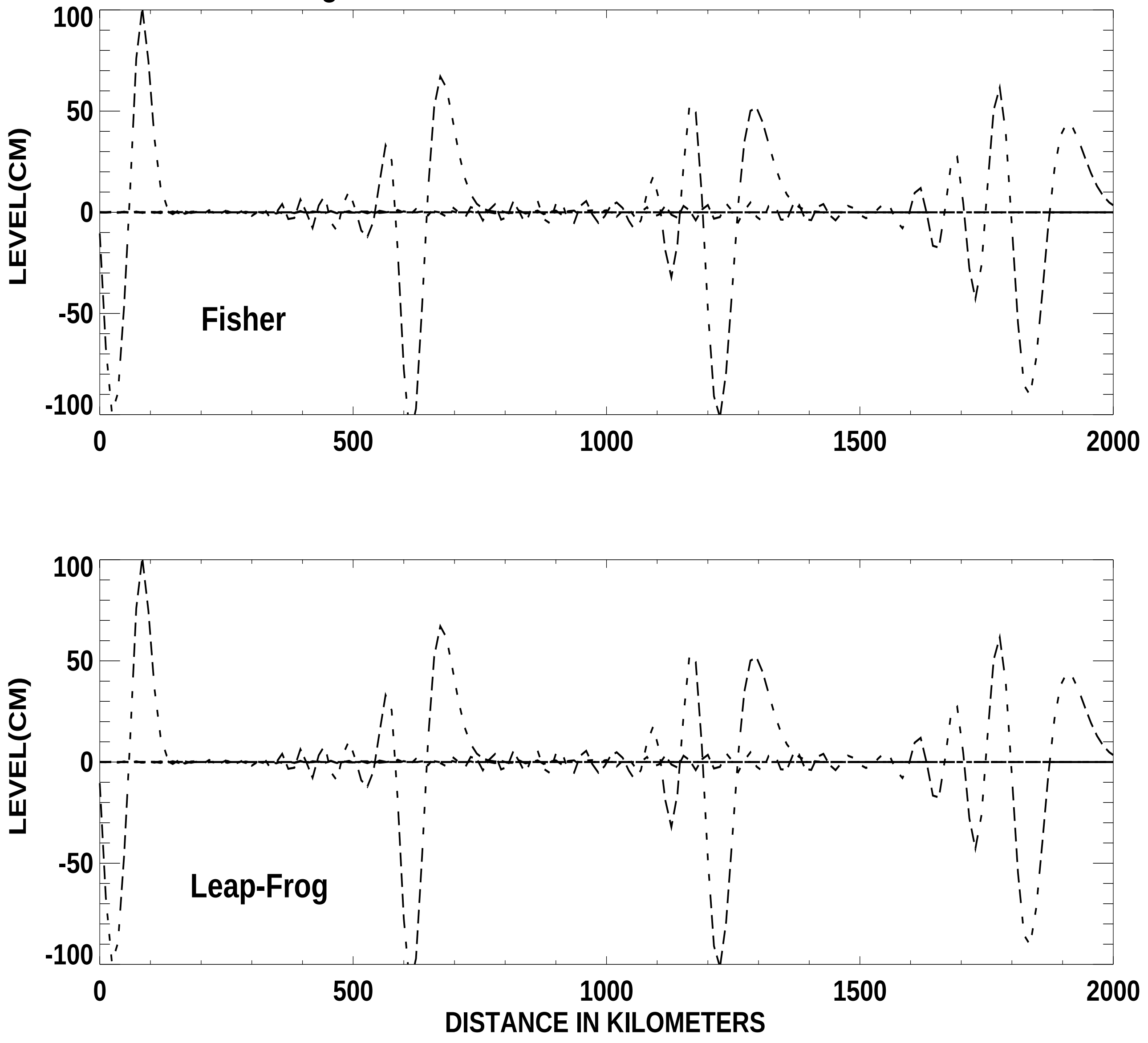


Fig.II.1: WAVE PROPAGATION IN CHANNEL

Problem No. 3, solution by corrected leap-frog method

To find the terms responsible for wave distortion in the leapfrog method, we shall introduce the Taylor series (see Ch.II of NM) into (II.5)and (II.6), to obtain

$$\frac{\partial u}{\partial t} + \frac{T^2}{6}\frac{\partial^3 u}{\partial t^3} = -g(\frac{\partial \zeta}{\partial x} + \frac{1}{24}\frac{\partial^3 \zeta}{\partial x^3}h^2) + O(h^4, T^4) \tag{II.8}$$

$$\frac{\partial \zeta}{\partial t} + \frac{T^2}{6}\frac{\partial^3 \zeta}{\partial t^3} = -\frac{\partial (uH)}{\partial x} - \frac{1}{24}\frac{\partial^3 (uH)}{\partial x^3}h^2 + O(h^4, T^4) \tag{II.9}$$

Thus the terms that introduce wave distortion are dependent on the third derivatives in space and time. Having written explicitly the error of approximation, we may correct this error by introducing its value into (II.5) and (II.6) with an opposite sign. Here we briefly describe the way to calculate space derivatives by defining a numerical form of a third derivative of the sea level in the u point,

$$\frac{\partial^3 \zeta}{\partial x^3} \simeq \frac{-\zeta_{j-2} + 3\zeta_{j-1} - 3\zeta_j + \zeta_{j+1}}{h^3} \tag{II.10}$$

The time derivatives are calculated somewhat differently as

$$\frac{\partial^3 \zeta}{\partial t^3} \simeq \frac{0.5\zeta_j^{m+2} - \zeta_j^{m+1} + \zeta_j^{m-1} - 0.5\zeta_j^{m-2}}{T^3} \tag{II.11}$$

The following correction will be introduced into the leapfrog scheme of (II.5) and (II.6).

$$Ctu = \frac{T^2}{6}\frac{\partial^3 u}{\partial t^3} \quad Cxu = \frac{1}{24}\frac{\partial^3 \zeta}{\partial x^3}h^2 \tag{II.12}$$

$$Ct\zeta = \frac{T^2}{6}\frac{\partial^3 \zeta}{\partial t^3} \quad Cx\zeta = \frac{1}{24}\frac{\partial^3 (uH)}{\partial x^3}h^2 \tag{II.13}$$

These corrections written in the numerical form can be introduced into system (II.5) and (II.6) with the opposite signs,

$$\frac{(u_j^{m+1} - u_j^{m-1})}{2T} - Ctu^m = -g\frac{\zeta_j^m - \zeta_{j-1}^m}{h} + gCx\zeta^m \tag{II.14}$$

$$\frac{\zeta_j^{m+1} - \zeta_j^{m-1}}{2T} - Ct\zeta^m = -\frac{(u_{j+1}^m H_{j+1} - u_j^m H_j)}{h} + Cxu^m \tag{II.15}$$

Direct application of this approach leads to mixed results: the spatial correction works well, while the temporal correction leads to the unstable numerical scheme. This time behavior of the leapfrog numerical scheme is caused by the presence of the so-called false solution. Therefore we direct our attention to redefining the time derivative in such a way that it will not influence overall stability of the numerical scheme. For construction of the third derivative in time, we shall change time derivative into the space derivative. From (II.1) and (II.2)

$$\frac{\partial^3 u}{\partial t^3} = -g^2 \left[\frac{\partial^2 H}{\partial x^2}\frac{\partial \zeta}{\partial x} + 2\frac{\partial H}{\partial x}\frac{\partial^2 \zeta}{\partial x^2} + H\frac{\partial^3 \zeta}{\partial x^3} \right] \tag{II.16}$$

$$\frac{\partial^3 \zeta}{\partial t^3} = -g \left[\frac{\partial H}{\partial x}\frac{\partial uH}{\partial x} + H\frac{\partial^3 uH}{\partial x^3} \right] \tag{II.17}$$

At this point, we introduce further simplification by assuming a flat bottom so the depth derivative in the above equations will vanish. Thus (II.8) and (II.9) by using (II.16)) and (II.17) changes to:

$$\frac{\partial u}{\partial t} = -g\frac{\partial \zeta}{\partial x} - g(\frac{h_x^2}{24} - gH\frac{T^2}{6})\frac{\partial^3 \zeta}{\partial x^3}h^2 + O(h^4, T^4) \tag{II.18}$$

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial uH}{\partial x}$$

$$-(\frac{h_x^2}{24} - gH\frac{T^2}{6})\frac{\partial^3 uH}{\partial x^3}h^2 + O(h^4, T^4) \tag{II.19}$$

Having written explicitly the error of approximation, we may correct this error as in the system (II.14)-(II.15).

The leapfrog program with these corrections is called SIMPDISP.F. Notation is similar to the programs given above and the new addition is related to to the higher order terms. Because it is difficult to define third derivative in the end points of the channel the coefficients ai, and aii are introduced to account for this problem

```
c program: simpdisp.f
C PROGRAM COMPUTES propagation in 1D channel
c by leap-frog method
c additional corrections are put on/off by commenting or
c uncommenting d2 in loop 8 and d4 in loop 12.
C program originated by Z.K. at IMS, Alaska
PARAMETER (je=175)
C
REAL U(je),UN(je),Z(je),ZN(je)
REAL H(je),UO(je),ai(je),aii(je)
REAL ZO(je),H1(JE)
open (unit=10,file='lp10c5.dat',status='unknown')
C B SERVE TO CONTROL TOTAL DEPTH
C IF B=1.,TOTAL DEPTH H1=H+B*(Z+TE)
B=0.
```

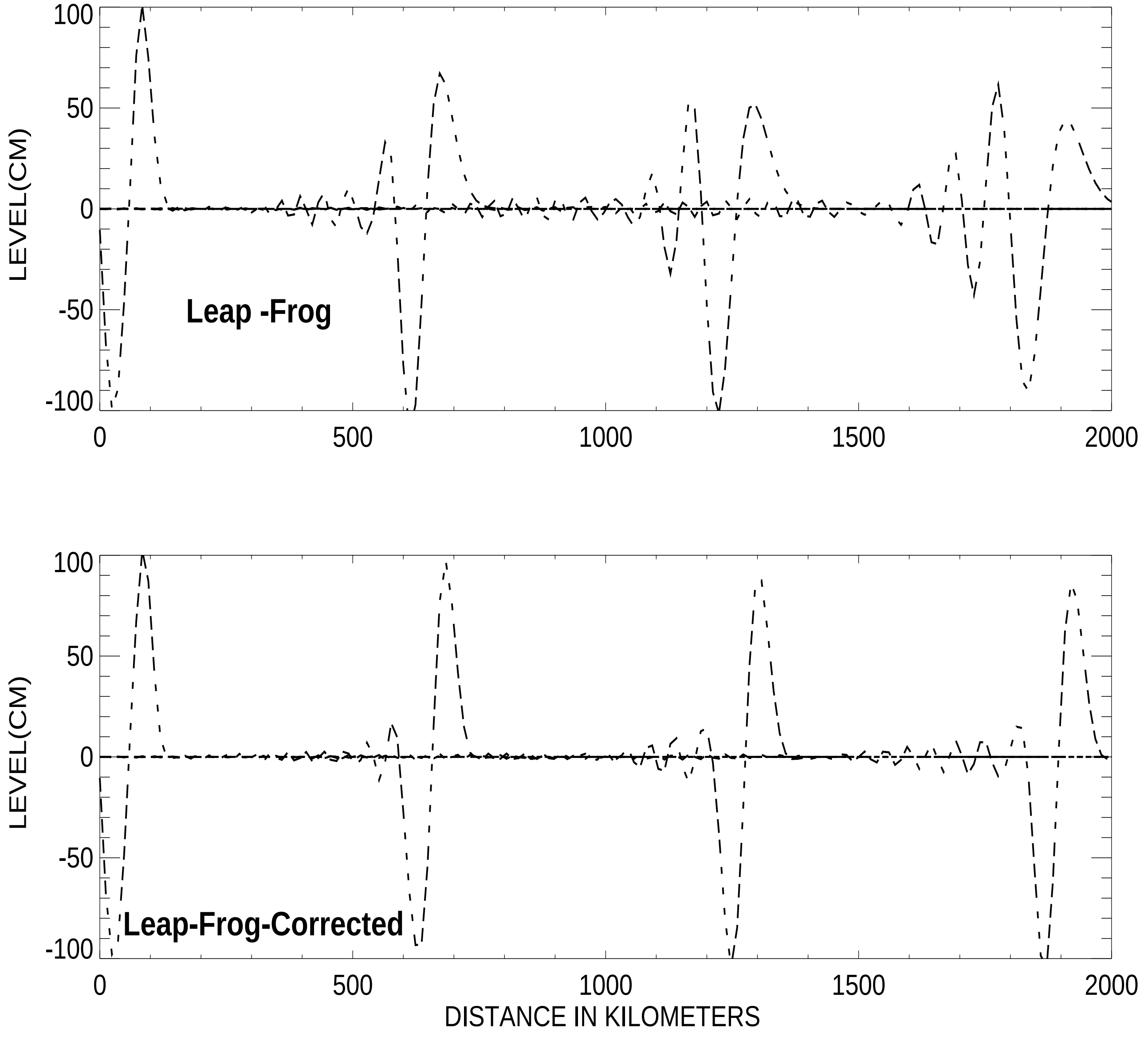
```
mon=1001
c basic data
G=982.
c time step and space step
t=10.
DX=12.e5
c depth
do j=1,je
h1(j)=4.077e5
h(j)=h1(j)
end do
PL=2.0*T/DX
DX2=DX*DX
DX3=DX*DX*DX
pd=2.*T/(DX3)
c this is done to for correction in proximity to the boundary
do j=1,je
ai(j)=0.
aii(j)=0.
end do
do j=3,je-3
ai(j)=1
end do
do j=5,je-5
aii(j)=1
end do
c start time loop
50 MO=MO+1
leu=leu+1
c input boundary only for one period and then switch to
c radiating condirion
zn(1)=-u(2)*sqrt(h(2)/g)
if(mo.gt.60)go to 22
zn(1)=100.*sin(0.1047197*float(mo-1))
22 continue
c radiating boundary at the very end of the channel
Un(je)=Z(je-1)*SQRT(G/H(je-1))
C TO INITIALIZE COMPUTATION USE FISHER SCHEME OR MAY
C BE YOU WISH TO APPLY
C IT EVRY 20 STEPS TO REMOVE NON-PHYSICAL SOLUTION.
C FOR THIS PURPOSE COUNTER LEU IS INTRODUCED
c if(leu.eq.20)go to 99
```

```
c IF(MO.EQ.1.or.mo.gt.1)GO TO 99
IF(MO.EQ.1)GO TO 99
C LEAP-FROG
DO8 J=2,je-1
HS=.5*(H1(J)+H1(J-1))
c numerical dispersion correction
c comment line d2=0 for correction
d1=-ai(j)*pd*G*(T*T*g*hs/6.-DX2/24.)
d2=d1*(z(j+1)-3.*z(j)+3.*z(j-1)-z(j-2))
d11=aii(j)*pL*G/(1920.)
d21=d11*(z(j+2)-5.*z(j+1)+10.*z(j)-10*z(j-1)+5.*z(j-2)-z(j-3))
d2=d2+d21
d2=0.
c calculate velocity
UN(J)=UO(J)-PL*G*(Z(J)-Z(J-1))+d2
8 CONTINUE
DO 12 J=2,je-1
um2=0.5*u(j-2)*(h1(j-3)+h1(j-2))
um1=0.5*u(j-1)*(h1(j-2)+h1(j-1))
ump=0.5*u(j)*(h1(j)+h1(j-1))
up1=0.5*u(j+1)*(h1(j)+h1(j+1))
up2=0.5*u(j+2)*(h1(j+2)+h1(j+1))
up3=0.5*u(j+3)*(h1(j+3)+h1(j+2))
HS=.5*(H1(J)+H1(J-1))
c numerical dispersion correction,
c comment line d4=0 for correction
d3=-ai(j)*pd*(T*T*g*hs/6.-DX2/24.)
d4=d3*(up2-3.*up1+3.*ump-um1)
d33=aii(j)*pl/(1920.)
d31=d33*(up3-5.*up2+10.*up1-10*ump+5.*um1-um2)
d4=d4+d31
d4=0.
C CALCULATE SEA LEVEL
ZN(J)=ZO(J)+PL*(ump-up1)+d4
12 CONTINUE
C==================================
C FISHER RESTART
GO TO 98
99 LEU=0
C CALCULATE VELOCITY
DO80 J=2,je-1
UN(J)=U(J)-.5*PL*G*(Z(J)-Z(J-1))
```

```
80 CONTINUE
C CALCULATE SEA LEVEL
DO 120 J=2,je-1
ZN(J)=Z(J)+.5*PL*(UN(J)*(H1(J)+H1(J-1))/2.0-UN(J+1
2)*(H1(J+1)+H1(J))/2.)
120 CONTINUE
98 CONTINUE
C CHANGE OLD VARIABLES TO NEW
DO 14 J=1,je
UO(J)=U(J)
U(J)=UN(J)
H1(J)=H(J)+B*(ZN(J))
ZO(J)=Z(J)
Z(J)=ZN(J)
14 CONTINUE
if(mo.eq.60.or.mo.eq.360.or.mo.eq.670.or.mo.eq.980)then
print *, mo
write (10,*)(zn(j),j=1,je-1)
end if
c stop calculation when mo=mon
IF(MO.EQ.MON)GO TO 23
GO TO 50
23 CONTINUE
END
```

The experiments are performed with a 12-km space step. In every experiment the equations are first solved without the third derivatives, and subsequently the correction is introduced by reversing the sign at the third derivative in (II.18) and (II.19). In Figure II.2 (top part) the results are given for the 12-km grid step (SPW=10). The leapfrog grossly distorts the wave 2000 km from the entrance. The leading wave is no longer dominating the wave pattern. The second wave has the largest amplitude, and the trailing waves are significantly modified both in phase and amplitude owing to energy flow toward the trailing edge. The correction (lower part of Fig. II.2) improves the wave parameters and reestablishes the first wave as the leading wave with 80% original amplitude.

Fig.II.2: WAVE PROPAGATION IN CHANNEL



Problem No. 4, numerical stability

Here, we shortly discuss stability properties of the numerical scheme given by (3.47) from NM, (in this chapter equations are numbered as (II.3 and II.4)). Stability parameter of this numerical scheme is always equal to one if Courant-Friedrichs-Lewy condition,

$$\sqrt{gH} \le \frac{h}{T} \tag{II.20}$$

is fulfilled. To estimate phase lag error introduced by the numerical scheme the formula (3.63) from NM is used.

$$\phi = \arctan \frac{\sqrt{(4Q - Q^2)}}{(2 - Q)} \tag{II.21}$$

Here Q is nondimensional parameter,

$$Q = 4gH(\frac{T}{h} \sin \frac{\kappa h}{2})^2 \tag{II.22}$$

which is expressed through the two nondimensional numbers, namely Courant number and spatial resolution number,

$$\frac{T}{h}\sqrt{gH} \qquad \text{and} \qquad \frac{L}{h} \tag{II.23}$$

We also introduce celerity ratio, as a parameter for the verification of the numerical scheme approximation (and also its stability). It compares numerically derived velocity against analytical expression for the phase velocity. The purpose is to retain this parameter close to unity so that the long wave phase velocity in model and in nature will have the same value. Celerity ratio is defined by (3.64) from NM, as,

$$\text{Celerity} \quad \text{ratio} = c_n/c_a \tag{II.24}$$

Here, numerical and analytical celerity is

$$c_n = \frac{\omega}{\kappa} = \frac{\phi}{\kappa T}; \qquad c_a = \sqrt{gH} \tag{II.25}$$

In the ensuing program (stabilitychannel.f) both the phase and celerity ratio are computed. To differentiate between notation for the depth and space step, the horizontal space step is called DX in the program. The results of computations are depicted in Figs. II.3 and II.4. The basic parameter which can rapidly influence the phase and celerity ratio is the space resolution. Increasing space resolution will diminish phase lag and bring celerity ration to unity. Decreasing of the Courant parameter influences phase lag but practically has no influence over celerity ratio.

Small domain at the right lower corner of the figures is influenced by negative values of the phase lag and celerity ratio greater than one.

```
    ==========================================================
    C PROGRAM TO CALCULATE stability of the Fisher method
    c program stabilitychannel.f (stabilt.f0)
    c program originated by z.k. at ims
    PARAMETER (LX=50,LY=100)
    REAL alpha(lx,ly),Celerity(lx,ly)
    c alpha is angle , celerity defines ratio of numerical and analytical
    c velocities
    pi = 4.*atan(1.)
    art=180./pi
    c to express alpha from radians to degree multiply by art
    c stability parameter of the Fisher method is equal to unity
    c ( formula 3.53 from NM) under
    c condition (3.54a) from NM. Here we calculate phase lag (LAMBDA)
    c by (3.63) from NM and celerity ratio by (3.64) from NM. Both are
    c calculated as function of two variables (two coordinates)
    c variable Q1=sqrt(gH)*T/DX and variable HK=2*PI*DX/L. Here coordinate
    c HK changes from small number to PI. Here DX denotes space step
    c usually in book given as h. L/DX sets horizontal space resolution
    c of numerical scheme. L=2*DX is the shortest wave.
    c FLOAT(J)=L/DX changes from 2 to LY. Q1 changes along LX from 1/50 to
1.
    DO I=2,LX
    DO J=1,LY
    HK=2.*PI/(FLOAT(J))
    Q1=FLOAT(I)/50. !Q1=sqrt(gH)*T/DX
    A2=4.*Q1*Q1
    hk1=hk/2.
    A3=sin(HK1)*sin(HK1)
    q=A2*A3
    a3=4*q-q*q
    a4=2.-q
    a5=sqrt(a3)/a4
    alpha(i,j)=art*(atan(a5))
    c this is for negative angles
    if(alpha(i,j).lt.0.)then
    alpha(i,j)=90+abs(alpha(i,j))
    end if
    Celerity(i,j)=alpha(i,j)/(hk*q1*art)
    END DO
    END DO
    open (unit=12,file='celerity.dat',status='unknown')
    open (unit=13,file='phasechan.dat',status='unknown')
    write(12,*)((Celerity(j,k),j=1,lx),k=1,ly)
    write(13,*)((alpha(j,k),j=1,lx),k=1,ly)
    stop
    end
```

Fig.II.3 Phase lag - propagation in channel

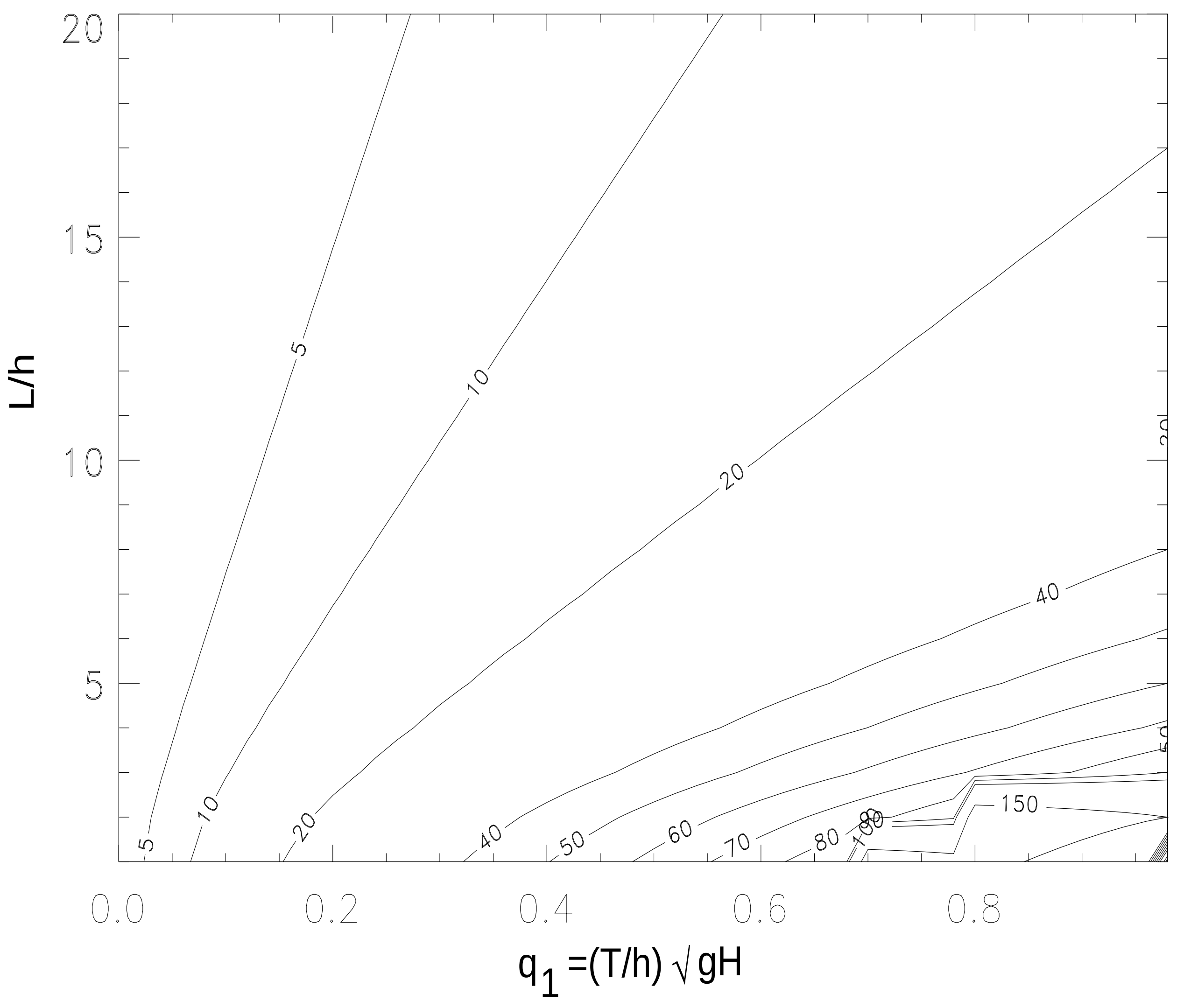
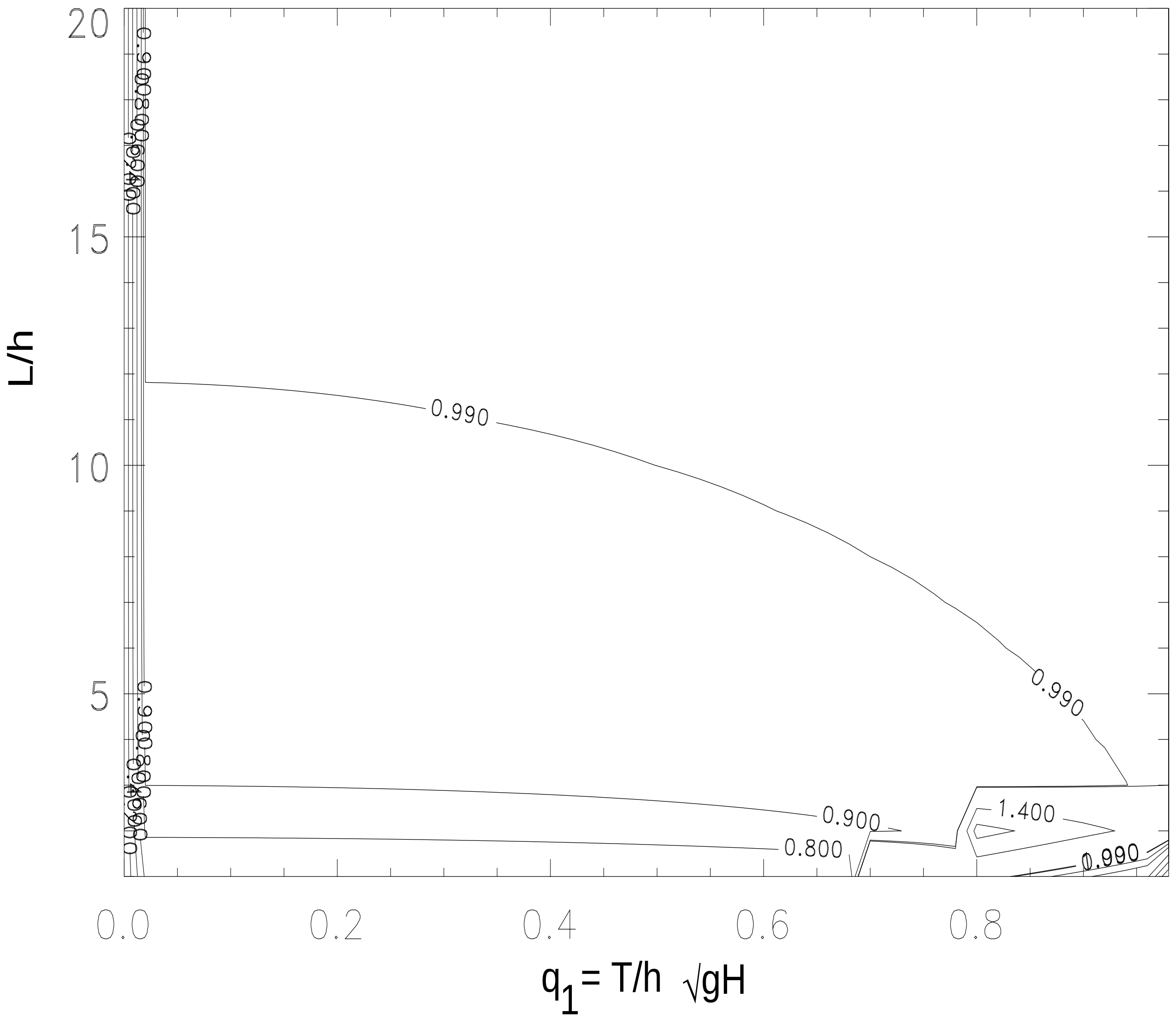


Fig.II.4 Celerity ratio - propagation in channel



2. Run-up in channel

We shall proceed to construct a simple algorithm for the run-up in the channel. This tool can be easily applied to the 2-D problems with the help of material from Ch.III, Sec.7.1 and 7.2 of NM. Consider equation of motion and continuity along x direction:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -g\frac{\partial \zeta}{\partial x} - \frac{ru|u|}{D} \tag{II.26}$$

$$\frac{\partial \zeta}{\partial t} = \frac{\partial}{\partial x}(Du) \tag{II.27}$$

Solution of this system will be searched through the two-time-level numerical scheme. The nonlinear (advective) term will be approximated by the upwind/downwind scheme. D in the above equations denotes total depth $D = H + \zeta$. The major problem arises when wave starts to move into and out of the dry domain. Obviously none of the second order symmetrical space schemes can not be used, because the movement is defined only in the wet domain. One of the major new assumptions is that equation of continuity can be approximated by the upwind/downwind approach as well. This approach makes equation of continuity quite stable at the boundary between wet and dry domains. To simulate run-up and run-down, the variable domain of integration is established after every time step by checking whether total depth is positive (condition 3.160 from NM). The following numerical scheme is used to march in time:

$$\begin{aligned}\frac{(u_j^{m+1} - u_j^m)}{T} &+ up\frac{(u_j^m - u_{j-1}^m)}{h} + un\frac{(u_{j+1}^m - u_j^m)}{h} \\ &= -g\frac{(\zeta_j^m - \zeta_{j-1}^m)}{h} + \frac{ru_j^m|u_j^m|}{0.5(D_j^m + D_{j-1}^m)}\end{aligned} \tag{II.28}$$

Here: $up = 0.5(u_j^m + |u_j^m|)$, and $un = 0.5(u_j^m - |u_j^m|)$

$$\frac{\zeta_j^{m+1} - \zeta_j^m}{T} = -(upj1 \times D_j^m + unj1 \times D_{j+1}^m - upj \times D_{j-1}^m - unj \times D_j^m) \tag{II.29}$$

In the above equation:

$$upj1 = 0.5(u_{j+1}^{m+1} + |u_{j+1}^{m+1}|) \quad \text{and} \quad unj1 = 0.5(u_{j+1}^{m+1} - |u_{j+1}^{m+1}|)$$

$$upj = 0.5(u_j^{m+1} + |u_j^{m+1}|) \quad \text{and} \quad unj = 0.5(u_j^{m+1} - |u_j^{m+1}|)$$

The space-staggered grid given in Figure 3.18 from NM is used to construct the space derivatives and to visualize procedures at the wet/dry boundary. At the left end of the channel a sinusoidal wave is given:

$$\zeta = \zeta_0 \sin(\frac{2\pi t}{T_p}) \tag{II.30}$$

Here the wave period $T_p = 1000s$. The wave propagates in the channel of 2km long. The bottom of the channel is sloping into the beach. The depth changes linearly from 10m at the left end of the channel to -10m at the right end of the channel. In tsunami problem the positive value of depth is related to the wet domain. The fine space grid of 5m is used; the time step is 0.5s.

Problem No. 5, calculation of the run-up in channel

The loop 50 is time stepping loop for the run-up. We start by defining sinusoidal signal at the left end of the channel. Next the total depth is calculated. Because part of the total depth related to the sea level is changing in time, the total depth is updated at each time step. The question of the boundary between the wet and dry domains is resolved by by checking the total depth in the loop 10. To better identify the boundary between dry and wet domains a small negative value p=0.1cm is added to the total depth. The extent of the wet domain is defined by index jm. Do we need to include point jm+1 and when we include this point into the computational domain is decided by simple condition whether the sea level at the point jm is greater than the depth at the point jm+1. Velocity is computed in the loop 11. After this loop is finished the velocity at the point jm+1 is evaluated by the linear extrapolation. The same approach is used for the sea level. When calculational loop for the sea level is finished a new sea level at the point jm+1 is evaluated through the linear extrapolation. The change of variable from the new UN, SLN to the old ones UO, SLO is done in the loop 13. The computation run in the time loop 50 until the number of time steps m reach the number ME. The name of the program is runupo.f. I have got basic ideas for this program from C. Mader, Los Alamos.

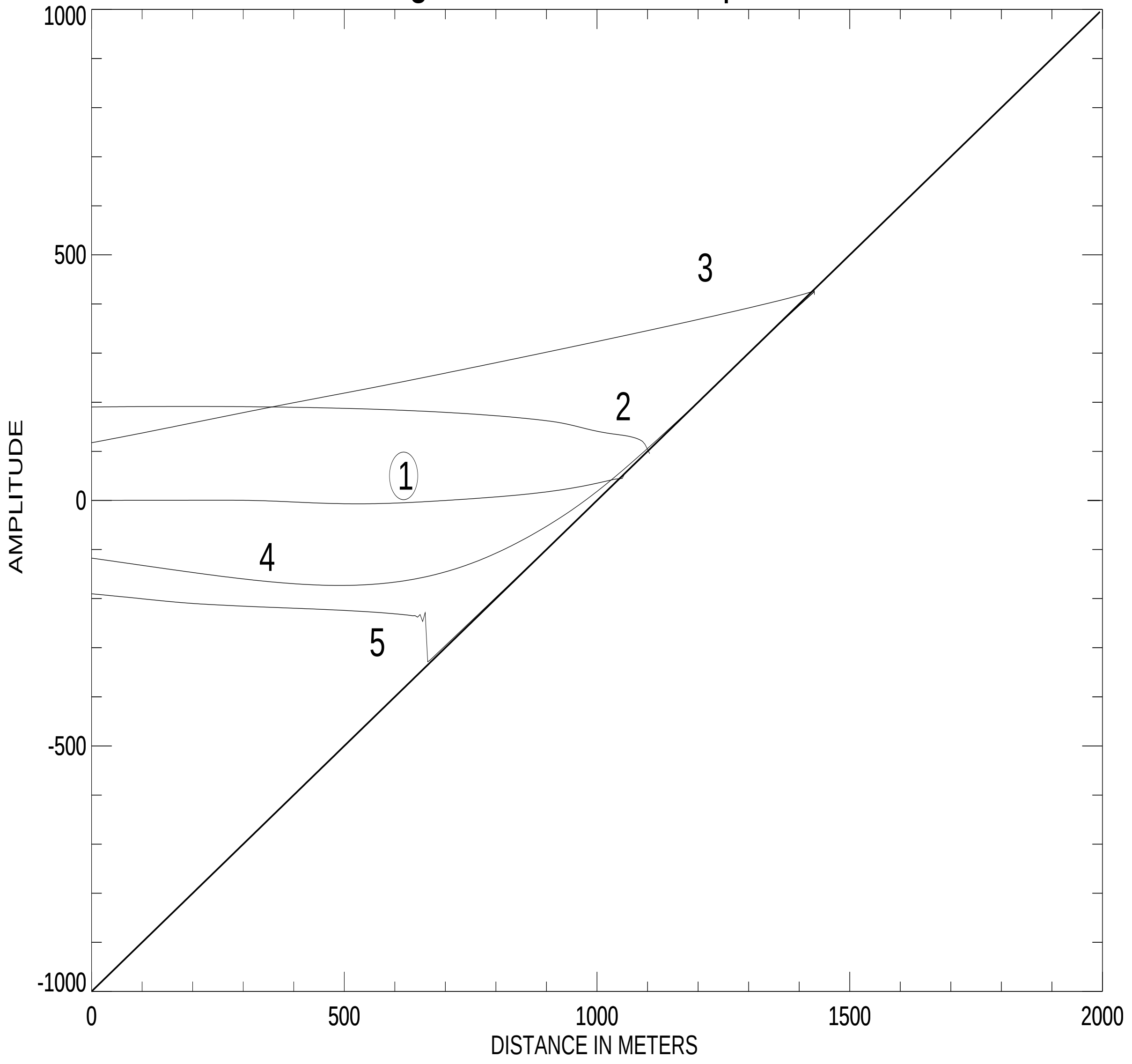
```
      ==========================================================■
      PROGRAM runup0.f
      ! BASIC IDEAS IN THIS PROGRAM WERE SUGGESTED BY C. MADER,
LOS ALAMOS
      parameter(js=1,je=401)
      dimension UO(je),UN(je),SLO(je),SLN(je),hm(je),d(je),du(je)
      dimension hu(je)
      open(unit=20,file='run20.dat',status='unknown')
      c-----------------------------------------
      T=0.1 ! time step in sec
      h=500. ! space step in cm
      g=981.
      aa=g*T/h
      bb=T/h
      am=200.
      tp=1000.
```

```
p=0.1
pi=3.1415927
w=2*pi/tp
r=0.003
c r=0.
adv=1.
c——————————————
do j=js,je
hm(j)=1005.-float(j)*5. ! depth distribution
end do
m=0
ME=20000 ! number of time steps
c——————————————
50 m=m+1
c mm=mm+1
c————————boundary——————–-
SLO(js)=am*sin(w*m*T) ! sinusoidal signal at the left
c————————————depths————–-
do j=js,je
d(j)=hm(j)+SLO(j) ! total depth
end do
do j=js+1,je
du(j)=0.5*(d(j)+d(j-1)) ! total depth in the u points
hu(j)=0.5*(SLO(j)+SLO(j-1))
end do
c——————————UN————— ! wet domain
c———wet or dry?————————————-
do 10 j=js,je
if(0.5*(d(j-1)+d(j)-p).gt.0.) jm=j ! variable index jm
! denotes the right boundary of the wet domain
10 continue
jmm=jm
if(SLO(jmm).gt.-hm(jmm+1))then
c only when sea level at the point jmm is greater than the depth
c in the point jmm+1. The latter is included into wet domain.
du(jmm+1)=0.5*(hm(jmm)+SLO(jmm))
jm=jmm+1
end if
c———————————————————
if(MOD(m,10).eq.0)then
write(*,*) jm
end if
```

```
c———————————————————
UN(2)=UO(2)-aa*(SLO(2)-SLO(1))-r*T*UO(2)*abs(UO(2))/du(2)
! first point is always special and it needs special treatment
do11 j=js+2,jm
u2n=0.5*(UO(j)-abs(UO(j)))
u2p=0.5*(UO(j)+abs(UO(j)))
UN(j)=UO(j)-adv*bb*(u2p*(UO(j)-UO(j-1))+u2n*(UO(j+1)-UO(j)))-
* aa*(SLO(j)-SLO(j-1))-r*T*UO(j)*abs(UO(j))/(du(j)+5.)
11 continue
c—-extrapolation to the first dry point——————————
UN(jm+1)=2.*UN(jm)-UN(jm-1)
c———————————SLN——————-
do 12 j=js+1,jm
td11=d(j+1)
td=d(j)
td22=d(j-1)
u1n=0.5*(UN(j+1)-abs(UN(j+1)))
u1p=0.5*(UN(j+1)+abs(UN(j+1)))
u2n=0.5*(UN(j)-abs(UN(j)))
u2p=0.5*(UN(j)+abs(UN(j)))
SLN(j)=SLO(j)-bb*(u1n*td11+u1p*td-u2n*td-u2p*td22)
12 continue
SLN(jm+1)=2*SLN(jm)-SLN(jm-1)
c—————————change variables—————————
do 13 j=js+1,je
UO(j)=UN(j)
SLO(j)=SLN(j)
13 continue
if(mod(m,2000).eq.0)then
write (20,*) (SLO(j),j=1,je)
end if
57 IF (m.ge.ME) go to 55
go to 50
55 continue
stop
end
```

The results of computations are given in the Fig.II.5. It depicts the sea level distribution at the different stages of the run-up and run-down of the wave. Sea level curves 1 and 2 are for the initial run-up. Curves 3 and 4 depict run-down. Curve 5 depicts an initial stage of the run-up of the second wave.

Fig.II.5: Tsunami runup



Problem No. 6, propagation in sloping channel – nonlinear terms

We shall proceed to construct a simple algorithm for the propagation along the up-sloping channel. This numerical scheme allows us to investigate processes occurring across the shelf. We will be able to pinpoint the influence of friction and nonlinear terms on the process of propagation and dissipation and hopefully understand how numerical schemes change tsunami physics. Consider equation of motion and continuity II.26 and II 27 solved by the numerical algorithms II.28 and II.29.

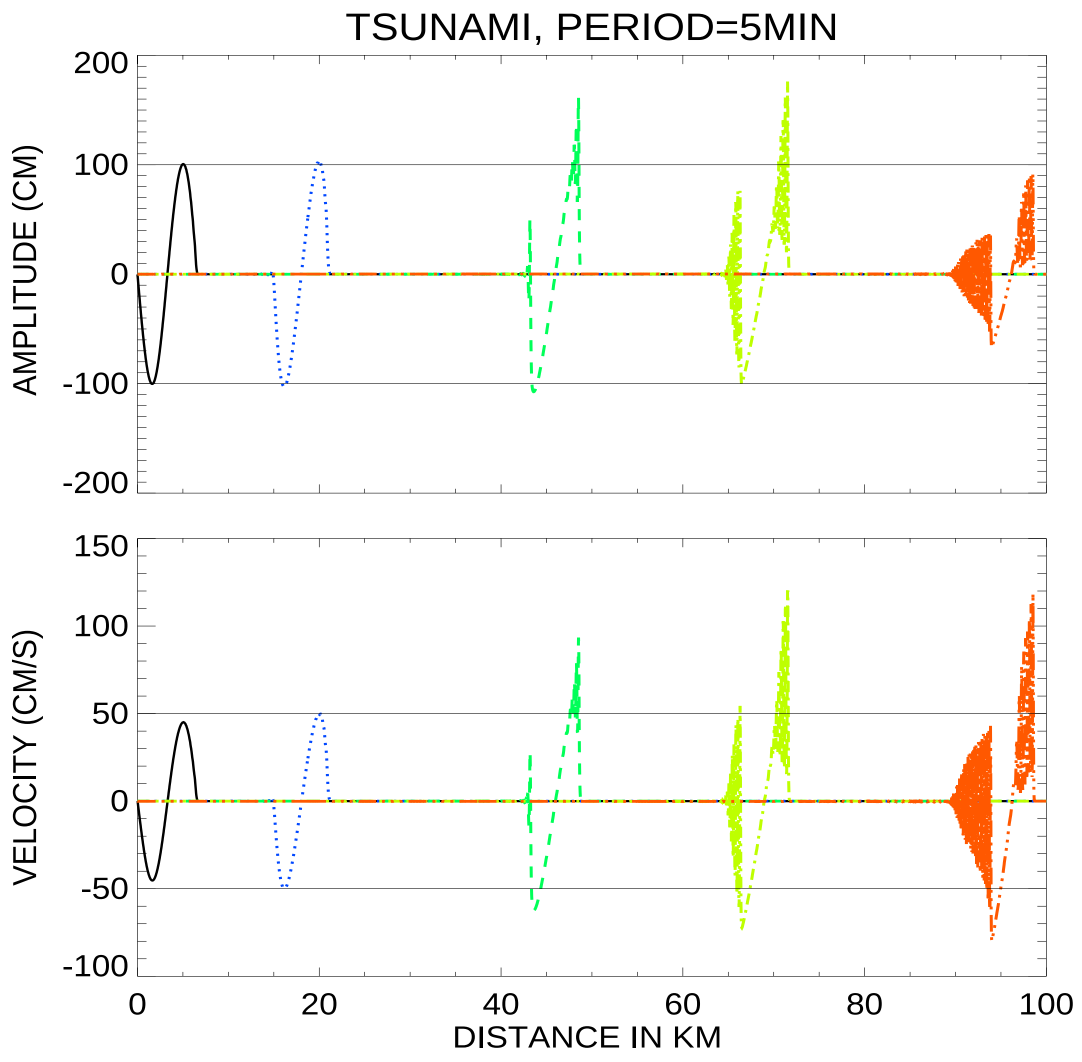


Figure II.6
Amplitude (upper panel) and velocity (lower panel) of 5 min period wave. Advective term and bottom friction are included.

In this experiment the upsloping channel of 100 km length is considered. Depth is changing from 50 m at the entrance to 5 m at the end of the channel. We start by computing propagation of a 5 min period tsunami wave from the deep water towards the shallow water. Results are depicted in Fig.II.6. It is important to notice the large disparity in the sea level and velocity. While the sea level amplitude changes over a small range along the channel, velocity, on the other hand, displays much greater variations and is less prone to the dissipation. Here we investigate the wave breaking process in the upsloping channel.

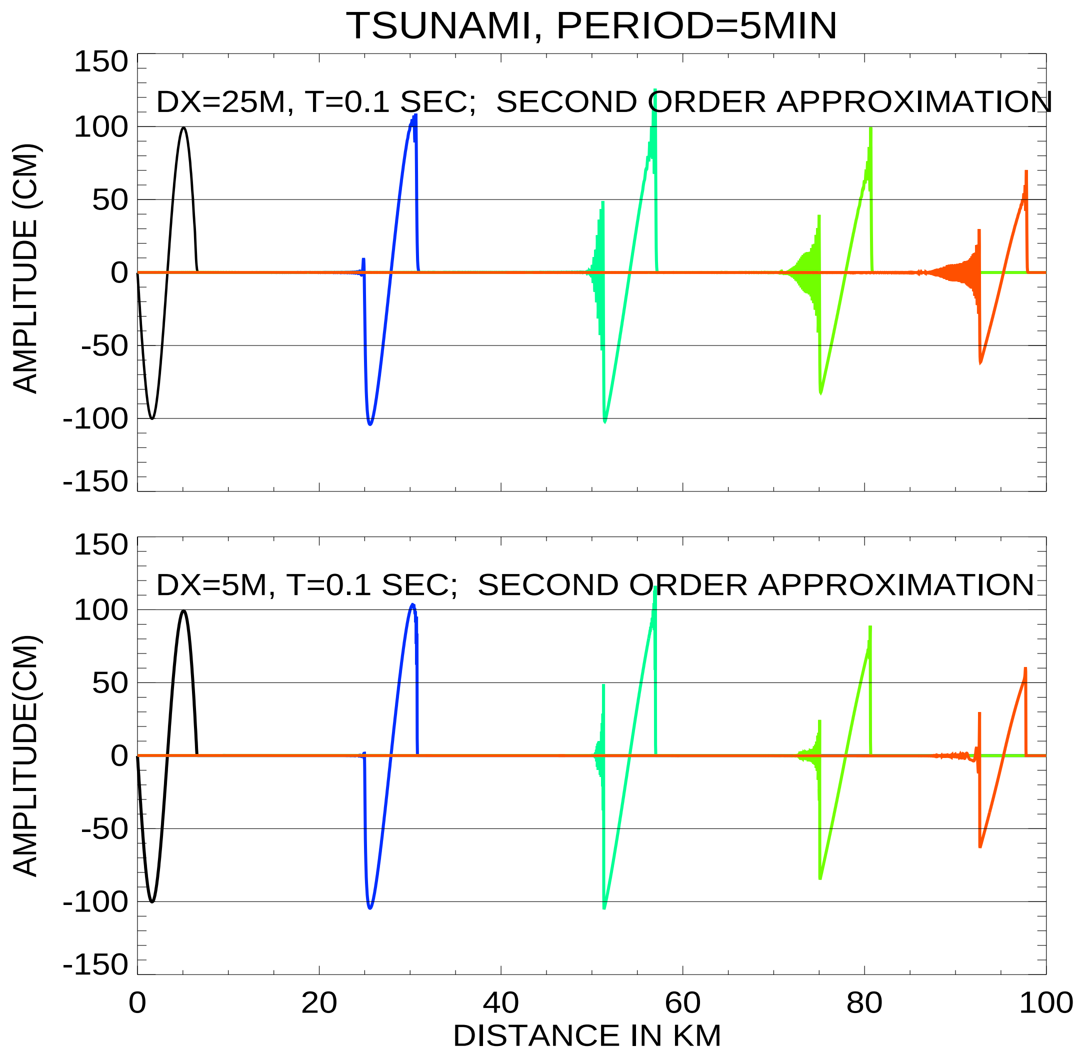


Figure II.7

Wave traveling in upsloping channel. Solution obtained by eqs. (II.28) and (II.29). Upper panel: space step 25 m, lower panel: space step 5 m.

Is this true physical process of the long wave breaking or this is a numerical artifact? The period is 600 s and the time step 0.1 s, thus the temporal resolution is 6000SPP (step per period). The wavelength is changing from approximately 7 km at the 50 m depth to 2 km at the 5 m depth. The latter is resolved with 25 m grid resulting in 80SPW, which is still an excellent resolution. Unfortunately, a simple notion of the spatial and temporal resolution needs to be reexamined since in the shallow water a strong nonlinear interaction occurs. This phenomenon should change our approach to analyzing the numerical stability of the basic set of equations, because previously we relied on the linear stability analysis only. Strong nonlinearities are often source of computational instabilities. To test whether the short period waves, occuring at the head and tail of the main wave, are the part of physical phenomenon we carry out a simple experiment with an increased spatial resolution by taking step h=5m. The result of calculation is given in Fig. II.7. In the upper panel for comparison the result with h=25 m is also shown. The improvement in resolution (lower panel) leads to decreasing of the short wave oscillations. Therefore, we may conclude that the short wave oscillations is an computational artifact caused by the poor space resolution.

The main source of nonlinear effects is the advective term. To the advective term in eq. (II.26) the upwind method of the first order approximation in space is applied. For the stability reason we keep the upstream approach even with the higher order of approximation for the first derivatives (see Kowalik and Bang, 1987). Construction of the first derivative can be carried out on the three-point or four-point stencil. For the three-point stencil the following construction can be used for the advective term,

$$u\frac{\partial u}{\partial x} \simeq up\frac{(3u_j^m - 4u_{j-1}^m + u_{j-2}^m)}{2h} + un\frac{(-u_{j+2}^m + 4u_{j+1}^m - 3u_j^m)}{2h} + O(h^3) \quad \text{(II.31)}$$

To bring the higher order of approximation to the entire set of equations together with the advective term the higher order space derivatives for the sea level and velocity in eqs. II.26 and II.27 has been used.

We start by constructing the space derivative for the sea level in the equation of motion II.26. The central point (u point) located in the j grid point is surrounded by the two sea level grid points at the distance $h/2$ from the velocity point. Considering two Taylor series for the sea level, one on the lattice $h/2$ and the second one on the lattice $3h/2$ by combining the two formulas the higher order formula can be constructed. The new formula for the first derivative of the sea level in the u point reads,

$$\frac{\partial \zeta}{\partial x} = [27(\zeta_j - \zeta_{j-1}) - (\zeta_{j+1} - \zeta_{j-2})]/24h + O(h^4) \quad \text{(II.32)}$$

Space derivative for the velocity in the continuity equation II.27 is constructed in the similar way by noticing that the central point for such derivative is the sea level and the space index should be moved to the right so that j is substituted by $j+1$,

$$\frac{\partial}{\partial x}(Hu) = \{27[u_{j+1}(h_j + h_{j+1})/2 - u_j(h_j + h_{j-1})/2] -$$

$$[u_{j+2}(h_{j+2} + h_{j+1})/2 - u_{j-1}(h_{j-2} + h_{j-1})/2]\}/24h + O(h^4) \qquad \text{(II.33)}$$

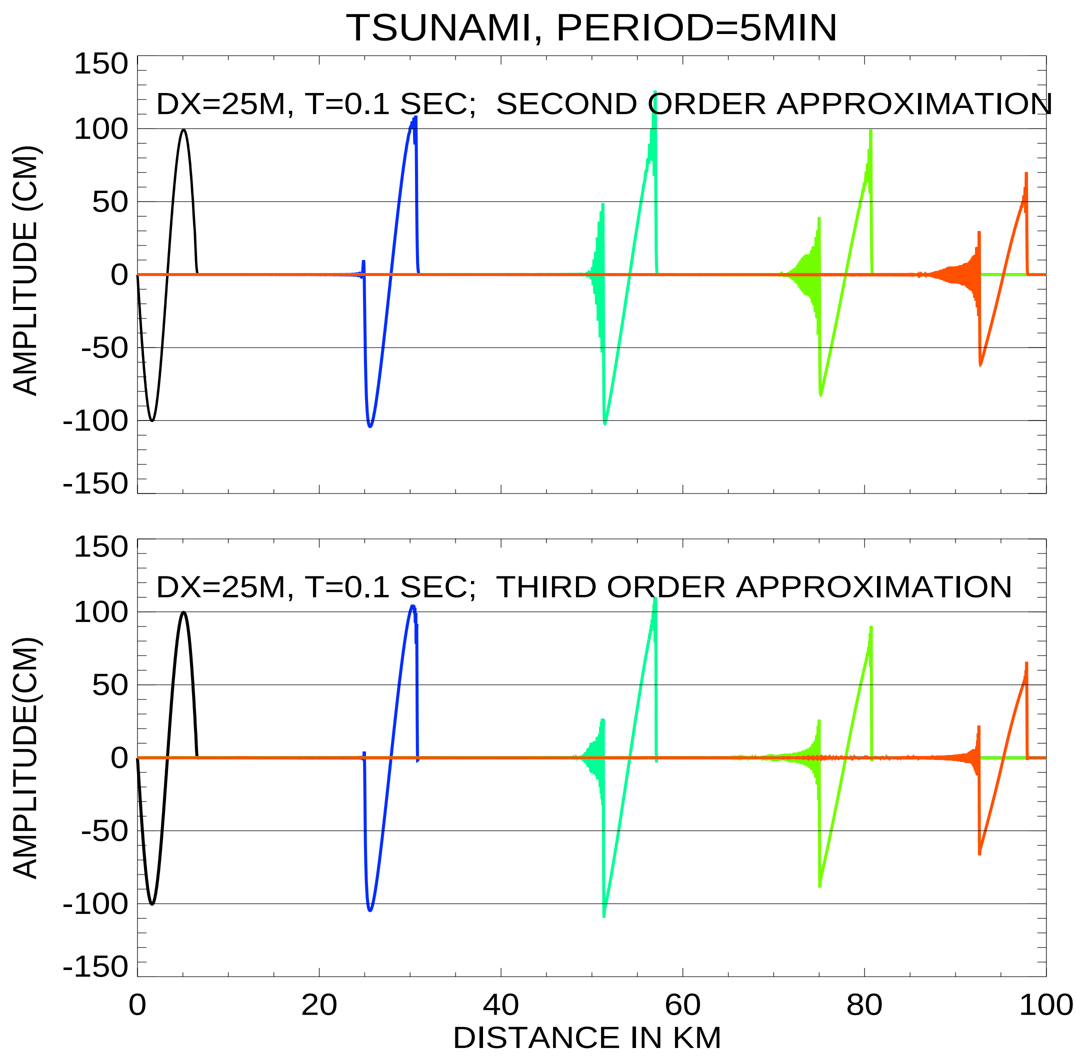


Figure II.8
Wave traveling in upsloping channel. Upper panel: solution by eqs. (II.28) and (II.29), lower panel: solution by the third order approximation

Application of this approach to the advective term together with the higher order derivatives (II.32) and (II.37) for the remaining space derivatives leads again to the improved results shown in the lower panel of Fig. II.8.

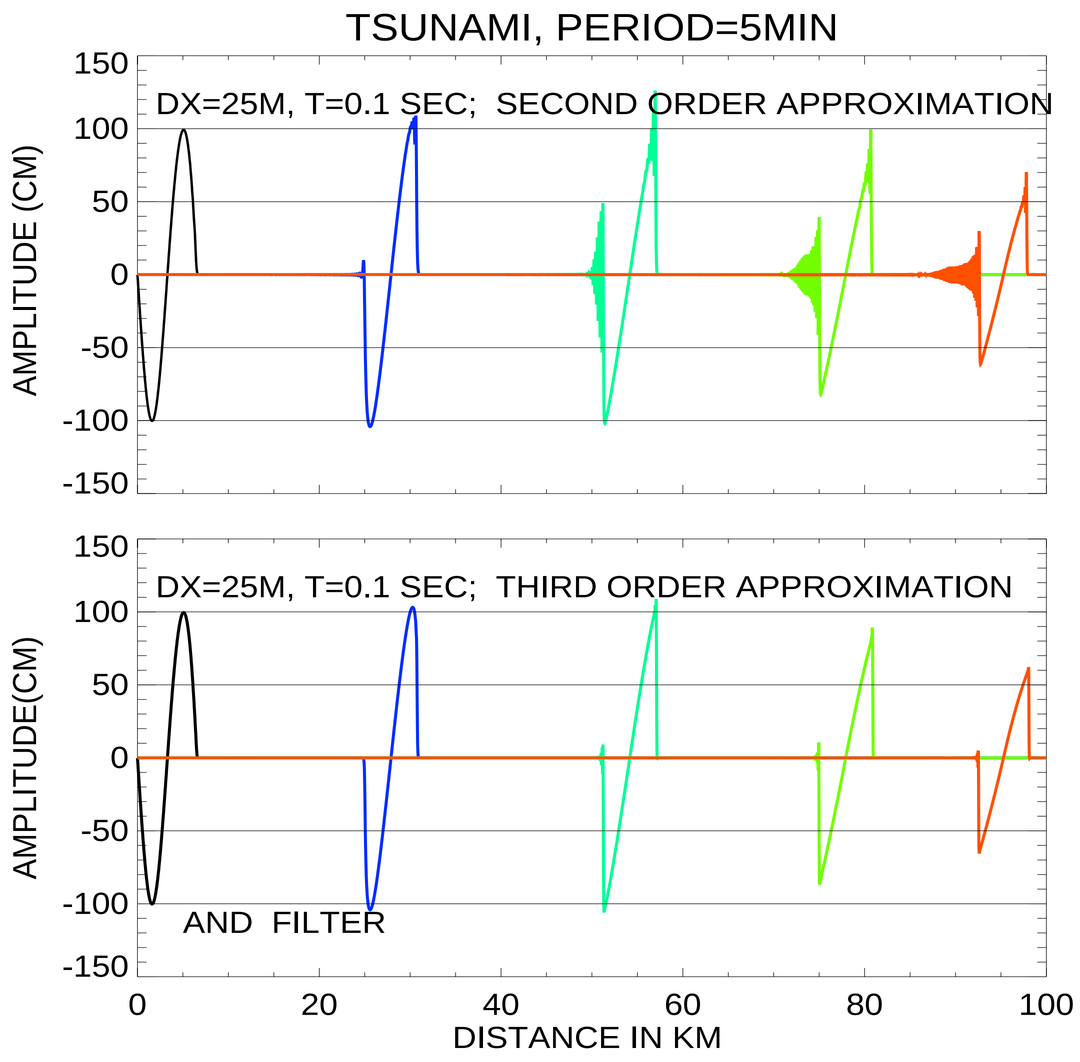


Figure II.9

Wave traveling in upsloping channel. Upper panel: solution by eqs. (II.28) and (II.29), lower panel: solution by the third order approximation and space filter

Conclusion from the above experiments is that the parasitic short wave oscillations can be deleted through an application of the high spatial and temporal resolutions. A somewhat different and easier solution is application of a simple space filter. It is applied only to the computed velocity. The new velocity u_j^{m+1} is filtered in the following manner,

$$UN(J) = u_j^{m+1} * (1 - ALP) + 0.25 * (u_{j-1}^{m+1} + 2. * u_j^{m+1} + u_{j+1}^{m+1}) * ALP \quad \text{(II.34)}$$

The filter parameter $ALP = 0.005$. The computation carried out with the high order derivatives and the above filter are shown in the lower panel of Fig. II.9

Program to perform above calculations is constructed in very similar way as the previous programs. Here the variable depth from 50m to 5m is considered.

```
    !Program upslope
    Program channel
    ! PROGRAM COMPUTE propagation in 1D upward sloping channel
    ! by the higher order methods
    ! program originated by Z.K. at IMS, Alaska
    INTEGER,PARAMETER:: je=4001,JS=1,MON= 65001
    INTEGER::NS,MON1
    REAL,DIMENSION(1:JE)::U,UN,Z,ZN,
H,H1,DU,U2N,U2P,AN,UF,UN1,UN2
    REAL,PARAMETER::HL=5000.,HR=500.,ADV=1.,B=1.,R=0.0033
    ! REAL,PARAMETER::HL=5000.,HR=500.,ADV=.0,B=0.,R=0.0
    open (unit=10,file='amp5.dat',status='unknown')
    open (unit=12,file='vel3.dat',status='unknown')
    OPEN(UNIT=11,FILE='depth.dat',STATUS='unknown')
    ! MON is total number of time steps
    ! if B=1 total depth is computed H1=H+Z*B
    ! IF ADV=1, NONLINEAR TERMS ARE IN
    ! R BOTTOfM FRICTION COEFFICIENT
    MON1=MON/200
    G=982.
    TM2=44714.1353 ! M2 TIDE PERIOD
    TP=300.
    PI=3.141592653589793238462643383279502884197
    ! time step and space step
    t=0.1 !0.1S
    DX=2500. ! 25 meters
    !NUMBER OF TIME STEPS IN TSUNAMI PERIOD
    ns=INT(TP/T)
    ! PRINT *,NS
    ! COEFFICIENTS FOR EQUATIONS
    PL=2.0*T/DX
    PLG=G*PL
    ! depth from HL= 50M AT J=1 to HR= 5M AT J=JE
```

```
do j=JS,je
hb=(HR-HL)/(Float(je-JS))
ha=(HL*FLOAT(JE)-HR*FLOAT(JS))/(FLOAT(JE)-FLOAT(JS))
h1(j)=ha+hb*float(j)
h(j)=h1(j)
end do
! write (11,*)(h1(j),j=1,je-1)
! start time loop
50 MO=MO+1
leu=leu+1
! DEPTH IN THE VELOCITY POINTS
DO J=2,JE
DU(J)=(H1(J-1)+H1(J))*0.5
IF(DU(J).LE.0.)DU(J)=10. ! JUST IN CASE TOTAL DEPTH IS ZERO
END DO ! OR NEGATIVE
! input boundary for one period and afterwards radiating condition
zn(1)=-u(2)*sqrt(h(2)/g)
! ZN(1)=0.
if(mo.gt.NS)go to 22
! if(mo.le.20000)go to 22
! if(mo.gt.20000+NS)go to 22
zn(1)=100.*sin(2.*pi*t*float(mo-1)/TP)
! zn(1)=ZN(1)+100.*sin(2.*pi*T*float(mo-1)/tm2)
! GO TO 24
22 continue
! zn(1)=ZN(1)+100.*sin(2.*pi*T*float(mo-1)/tm2)
24 CONTINUE
! radiating boundary at the very end of the channel
Un(je)=Z(je-1)*SQRT(G/H1(je-1))
! compute velocity
u2n(2:je-1)=0.5*(U(2:je-1)-abs(U(2:je-1))) ! ALWAYS NEGATIVE
u2p(2:je-1)=0.5*(U(2:je-1)+abs(U(2:je-1))) ! ALWAYS POSITIVE
!FOR THE POINTS CLOSE TO THE BOUNDARY THE LOWER ORDER
!OF APPROXIMATION HAS BEEN
!RETAINED SINCE WE ARE LACKING THE NUMBER OF GRID
!POINTS TO CONSTRUCT THE HIGH ORDER
AN(2)=U2N(2)*(U(3)-U(2))+&
U2P(2)*(U(2)-U(1))
AN(3)=U2N(3)*(U(4)-U(3))+&
U2P(3)*(U(3)-U(2))
UN(2)=U(2)-.5*PLG*(Z(2)-Z(1))&
-adv*0.5*PL*AN(2)&
```

```
-r*T*U(2)*abs(U(2))/DU(2)
UN2(2)=UN(2)
UN1(2)=UN(2)
AN(JE-1)=U2N(JE-1)*(U(JE)-U(JE-1))+&
U2P(JE-1)*(U(JE-1)-U(JE-2))
AN(JE-2)=U2N(JE-2)*(U(JE-1)-U(JE-2))+&
U2P(JE-2)*(U(JE-2)-U(JE-3))
! HIGHER ORDER FOR ADVECTIVE TERM
AN(3:JE-2)= U2N(3:JE-2)*(-U(5:JE)+4.*U(4:JE-1)-3.*U(3:JE-2))/2.+ &
U2P(3:JE-2)*(3.*U(3:JE-2)-4.*U(2:JE-3)+U(1:JE-4))/2.   !THIRD DERIVA-
TIVE
! AN(3:JE-2)=U2N(3:JE-2)*(-U(5:JE)+6.*U(4:JE-1)-
!3.*U(3:JE-2)-2.*U(2:JE-3))/6.+&
! U2P(3:JE-2)*(2.*U(4:JE-1)+3.*U(3:JE-2)
!-6.*U(2:JE-3)+U(1:JE-4))/6. ! fOURTH DERIVATIVE
! AN(4:JE-3)= U2N(4:JE-3)*(2.*U(7:JE)-9.*U(6:JE-1)
!+18*U(5:JE-2)-11.*U(4:JE-3))/6.+&
! U2P(4:JE-3)*(-2.*U(1:JE-6)+9.*U(2:JE-5)
!-18*U(3:JE-4)+11.*U(4:JE-3))/6.
DO 80 J=3,je-2
!LOWER ORDER APPROXIMATION
!UN(J)=U(J)-.5*PLG*(Z(J)-Z(J-1)) &
!-adv*0.5*PL*(u2p(J)*(U(j)-U(j-1))+u2n(J)*(U(j+1)-U(j)))&
!-r*T*U(j)*abs(U(j))/DU(J)
! HIGHER ORDER OF APPROXIMATION
A1=27.*(Z(J)-Z(J-1))
A2=Z(J+1)-Z(J-2)
UN(J)=U(J)-.5*PLG*(A1-A2)/24.&
-adv*0.5*PL*AN(J)&
-r*T*U(j)*abs(U(j))/DU(J)
80 CONTINUE
! USE FILTER - FOR THIS PURPOSE CHANGE IN THE ABOVE
!FORMULA UN TO UN1 SO THAT UN COMES OUT OF FILTER
! DO J=3,JE-2
! UN2(J)=UN1(J)-adv*0.5*PL*AN(J)
! END DO
! DO J=3,JE-2
! ALP=0.005
! UN(J)= UN1(J)*(1-ALP)+0.25*(UN1(J-1)+2.*UN1(J)+UN1(J+1))*ALP
! END DO
! compute sea level
DO 120 J=2,je-1
```

```
! lOWER ORDER APPROXIMATION
! ZN(J)=Z(J)+.5*PL*(UN(J)*(H1(J)+H1(J-1))/2.0 &
! -UN(J+1)*(H1(J+1)+H1(J))/2.)
! LOWER ORDER OF APPROXIMATION USING
!UPWIND/DOWNWIND (SIGN OF UN)
! td11=H1(j+1)
! td=H1(j)
! td22=H1(j-1)
! u1n=0.5*(UN(j+1)-abs(UN(j+1)))
! u1p=0.5*(UN(j+1)+abs(UN(j+1)))
! u2n=0.5*(UN(j)-abs(UN(j)))
! u2p=0.5*(UN(j)+abs(UN(j)))
! ZN(j)=Z(j)-0.5*PL*(u1n*td11+u1p*td-u2n*td-u2p*td22)!
! HIGHER ORDER (FOURTH ORDER) OF APPROXIMATION
AA1=27.*(UN(J+1)*(H1(J)+H1(J+1))/2.0-UN(J)*(H1(J)+H1(J-1))/2.0)
AA2=UN(J+2)*(H1(J+2)+H1(J+1))/2.0-UN(J-1)*(H1(J-2)+H1(J-1))/2.0
ZN(J)=Z(J)-.5*PL*(AA1-AA2)/24.
120 CONTINUE
! CHANGE OLD VARIABLES TO NEW
DO 14 J=1,je
U(J)=UN(J)
H1(J)=H(J)+B*(ZN(J)) ! NEW TOTAL DEPTH
Z(J)=ZN(J)
14 CONTINUE
140 CONTINUE
!if(mo.eq.NS.or.mo.eq.15000.or.mo.eq.30000.or.mo.eq.47000. &
!r.mo.eq.64000)then
!if(mo.eq.20000+NS.or. &
!mo.eq.40000+20000.or.mo.eq.65000+20000)then
! IF(MOD(mo,500).eq.0)THEN
IF(MOD(MO,MON1).EQ.0)THEN
print *, mo
do j=1,je-1,2
write (10,*)zN(j)
WRITE(12,*)un(j)
END do
end if
IF(MO.EQ.MON)GO TO 23
GO TO 50
23 CONTINUE
END program channel
```

3. Rectangular and non-rectangular water bodies

Here the vertically integrated equations will be rendered into numerical form for the rectangular water bodies. This problem is well described in Ch.III, Sec.4 of NM. In the brief explanation given below we shall use equations in both rectangular and spherical coordinate systems. Rectangular system is used because it is simple in the applications, but when the horizontal scale of a computational domain is exceeding 1000km the spherical system must be used to describe properly all consequences related to the spherical shape of Earth. For numerical formulation in rectangular system we shall use eqs. (3.95)-(3.97) from NM with modifications made in the frictional terms. These terms instead at time $m-1$, as in NM, will be taken at time step m.

$$\frac{u_{j,k}^{m+1}-u_{j,k}^{m}}{T}+UPOS(u_{j,k}^{m}-u_{j-1,k}^{m})/h+UNEG(u_{j+1,k}^{m}-u_{j,k}^{m})/h$$
$$+VAUP(u_{j,k}^{m}-u_{j,k-1}^{m})/h+VAUN(u_{j,k+1}^{m}-u_{j,k}^{m})/h$$
$$-f\bar{v}^{u,m}=-\frac{g}{h}(\zeta_{j,k}^{m}-\zeta_{j-1,k}^{m})+\frac{1}{\rho_o D_{u,j,k}^{m}}(\tau_{x,j,k}^{s,m}$$
$$-R_{x,j,k}^{m}u_{j,k}^{m})+\frac{N_h}{h^2}(u_{j+1,k}^{m}+u_{j-1,k}^{m}+u_{j,k+1}^{m}+u_{j,k-1}^{m}-4u_{j,k}^{m}) \tag{II.35}$$

$$\frac{v_{j,k}^{m+1}-v_{j,k}^{m}}{T}+UAVP(v_{j,k}^{m}-v_{j-1,k}^{m})/h+UAVN(v_{j+1,k}^{m}-v_{j,k}^{m})/h$$
$$+VPOS(v_{j,k}^{m}-v_{j,k-1}^{m})/h+VNEG(v_{j,k+1}^{m}-v^{m}j,k)/h$$
$$+f\bar{u}^{v,m+1}=-\frac{g}{h}(\zeta_{j,k+1}^{m}-\zeta_{j,k}^{m})+\frac{1}{\rho_o D_{v,j,k}^{m}}(\tau_{y,j,k}^{s,m}$$
$$-R_{y,j,k}^{m}v_{j,k}^{m})+\frac{N_h}{h^2}(v_{j+1,k}^{m}+v_{j-1,k}^{m}+v_{j,k+1}^{m}+v_{j,k-1}^{m}-4v_{j,k}^{m}) \tag{II.36}$$

$$\frac{\zeta_{j,k}^{m+1}-\zeta_{j,k}^{m}}{T}=-\frac{1}{h}(u_{j+1,k}^{m+1}D_{u,j+1,k}^{m}-u_{j,k}^{m+1}D_{u,j,k}^{m})-\frac{1}{h}(v_{j,k}^{m+1}D_{v,j,k}^{m}-v_{j,k-1}^{m+1}D_{v,j,k-1}^{m}) \tag{II.37}$$

The nonlinear terms are given according to the upwind–downwind approximation introduced in Ch.II of NM. The following notation is used:

$$UPOS=(u_{j,k}^{m}+|u_{j,k}^{m}|)/2 \qquad UNEG=(u_{j,k}^{m}-|u_{j,k}^{m}|)/2$$

$$VAUP=(\bar{v}^{u,m}+|\bar{v}^{u,m}|)/2 \qquad VAUN=(\bar{v}^{u,m}-|\bar{v}^{u,m}|)/2$$

$$VPOS = (v^m_{j,k} + |v^m_{j,k}|)/2 \qquad VNEG = (v^m_{j,k} - |v^m_{j,k}|)/2$$

$$UAVP = (\bar{u}^{v,m+1} + |\bar{u}^{v,m+1}|)/2 \qquad UAVN = (\bar{u}^{v,m+1} - |\bar{u}^{v,m+1}|)/2 \tag{II.38}$$

In the ensuing FORTRAN program the following notation is used. New velocities, calculated at the $m+1$ time step are called un, vn and new sea level is called zn. Old variables are uo, vo and zo respectively. The average of v velocity in the u point $\bar{v}^{u,m}$ is calculated as

$$vau = (vo(j,k) + vo(j-1,k) + vo(j,k-1) + vo(j-1,k-1)) * .25 \tag{II.39}$$

The average of u velocity in the v point $\bar{u}^{v,m+1}$ is given as

$$uav = (un(j,k) + un(j+1,k) + un(j,k+1) + un(j+1,k+1)) * .25 \tag{II.40}$$

The total depth $D = H + \zeta$ is changed to H, because in all computations $H >> \zeta$. In the space staggered-grid under j, k index we actually consider three different points. Therefore, we have to decide where the depth points should be located. A good choice is usually to locate it in ζ points, thus the depth in the u and v points will calculated as an average.

$$HU = 0.5 * (H(J,K) + H(J-1,K)) \quad \text{and} \quad HV = 0.5 * (H(J,K) + H(J,K+1)) \tag{II.41}$$

.

In the above numerical equations the bottom friction term is expressed by coefficient R, therefore, explicit expressions are given below for the friction terms. Along x direction

$$-R^m_{x,j,k} u^m_{j,k} = -rf1 * uo(j,k) * sqrt(uo(j,k) * uo(j,k) + vau * vau)/hu(j,k) \tag{II.42}$$

and along y direction,

$$-R^m_{y,j,k} v^m_{j,k} = -rf1 * vo(j,k) * sqrt(vo(j,k) * vo(j,k) + uav * uav)/hv(j,k) \tag{II.43}$$

There are several small changes in the notation between the above equations and FORTRAN programs. Bottom friction coefficient r is called in program as $rf1, rf2$, or $rf3$ depending on the magnitude of this coefficient.

Along with the above explicit formulas the fully implicit expressions will be used. The latter should improve stability of the numerical system. Considering, for the simplicity sake, acceleration and bottom terms only, one can write an implicit formula (along x direction),

$$\frac{u^{m+1} - u^m}{T} = -R^m u^{m+1} \tag{II.44}$$

To calculate velocity at the $m+1$ time step from (II.44) one obtains,

$$u^{m+1} = \frac{u^m}{(1+TR^m)} \tag{II.45}$$

Horizontal friction in all equations is denoted as N_h but in the FORTRAN program is called ah. To improve stability, along with the explicit algorithm we shall use semi-implicit formulas given by (3.94a) from NM. Considering only horizontal friction and acceleration

$$\frac{u_j^{m+1} - u_j^m}{T} = \frac{N_h}{h^2}[(u_{j+1}^m - u_j^m) - (u_j^{m+1} - u_{j-1}^{m+1})] \tag{II.46}$$

The new velocity will be computed as

$$u_j^{m+1} = u_j^m \frac{h^2}{h^2 + TN_h} + coef(u_{j+1}^m + u_{j-1}^{m+1} - u_j^m) \tag{II.47}$$

Here

$$coef = \frac{TN_h}{(h^2 + TN_h)}$$

We have also introduced coefficients called $cfric$, $cnl2$, and $ccor$, equal to 0 or 1, with the aim of facilitating numerical experiments by switching off and on various terms in the equations. Coriolis parameter f is equal $2w \sin\phi$.

In the spherical system of coordinate we shall use the vertically integrated equation of motion and continuity (1.92)–(1.95) from Ch.I of NM.

$$\frac{Du}{Dt} - fv - \frac{uv\sin\phi}{R\cos\phi} = -\frac{g}{R\cos\phi}\frac{\partial\zeta}{\partial\lambda} + \frac{\tau_\lambda^s - \tau_\lambda^b}{\rho D} + Au \tag{II.48}$$

$$\frac{Dv}{Dt} + fu + \frac{uu\sin\phi}{R\cos\phi} = -\frac{g}{R}\frac{\partial\zeta}{\partial\phi} + \frac{\tau_\phi^s - \tau_\phi^b}{\rho D} + Av \tag{II.49}$$

$$\frac{\partial\zeta}{\partial t} + \frac{1}{R\cos\phi}\frac{\partial(Du)}{\partial\lambda} + \frac{1}{R\cos\phi}\frac{\partial}{\partial\phi}(Dv\cos\phi) = 0 \tag{II.50}$$

Here

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \frac{u}{R\cos\phi}\frac{\partial}{\partial\lambda} + \frac{v}{R}\frac{\partial}{\partial\phi} \tag{II.51}$$

The advective nonlinear terms in (II.51) are about two orders of magnitude larger than the remaining nonlinear terms in (II.48) and (II.49), therefore in ensuing computations only advective nonlinear terms are taken into account.

The operator in the horizontal friction term in (II.48) and (II.49) is

$$A = N_h\left(\frac{1}{R^2\cos^2\phi}\frac{\partial^2}{\partial\lambda^2} + \frac{1}{R^2\cos\phi}\frac{\partial}{\partial\phi}(\cos\phi\frac{\partial}{\partial\phi})\right) \tag{II.52}$$

The bottom friction components are taken as

$$\tau^b_\lambda = \rho r u\sqrt{u^2+v^2}; \qquad \tau^b_\phi = \rho r v\sqrt{u^2+v^2}. \tag{II.53}$$

Numerical form of the equations written in the spherical coordinates differs only slightly from equations given above for the rectangular system.

To control the computation one can check temporal and spatial behavior of the sea level and velocities. Therefore the sea level is recorded in a few grid-points and time series are plotted. We assume that the time change of the sea level will shed some light on the temporal behavior of the computed process. For example time series for the tides should behave in the oscillatory manner. Time series for the motion imparted by the constant wind should arrive to the steady state after transient processes will decay.

The total energy of the system, i.e. kinetic and potential energy calculated over the entire domain is also a very good parameter for checking the behavior of the system and especially its temporal properties. Total energy in the FORTRAN program is called *eni* and is calculated in the sea level point. Velocities required for the calculations are taken as average from the neighboring points. Formula (3.5a) from NM is used for the calculations.

Problem No. 7, storm surge in rectangular domain

To learn about intricate problems of the 2-D modeling we shall consider a rectangular water body 1500km by 1500km, with gently sloping bottom from 200m to 20m along x direction. Along y direction depth is constant. Along x direction wind is blowing, changing linearly in time from 0 to 2 CGS in two hours, and remains constant thereafter. The time step of numerical integration is 100s, and space step is 10 km. Below, the FORTRAN program surge.f is given.

```
c=========================================
C PROGRAM SURGE.F
C ORIGINATED BY Z.K. AT IMS
c program to compute storm surge in the rectangular domains
parameter (jx=150,ky=150)
dimension uo(jx,ky),un(jx,ky),vo(jx,ky),vn(jx,ky),
* zo(jx,ky),zn(jx,ky)
c uo, un, vo, vn old and new velocity component along x (longitude)
c and along y (latitude)
c zo and zn old and new sea level
dimension h(jx,ky),hu(jx,ky),hv(jx,ky),eni(500)
```

```
c depth h and depth dependent functions hu,hv (averages along x and y direc-
tions)
c eni is total energy of the system
c
dimension cfric(jx,ky),ccor(jx,ky),cnl2(jx,ky)
c cfric,ccor,cnl2 coefficients equal to 1 or 0 to switch off and on
c friction, coriolis and nonlinear terms
dimension taux(jx,ky),tauy(jx,ky)
c wind stress along x and y
open (unit=10,file='zn.dat',status='unknown') ! sea level
open (unit=20,file='momaxm2.dat',status='unknown') ! time series
open (unit=21,file='un.dat',status='unknown') ! u-velocity
open (unit=22,file='vn.dat',status='unknown') ! v-velocity
open (unit=25,file='enim2.dat',status='unknown') ! energy
pi = 4.*atan(1.)
rad = pi/180.
c grid size along x (longitude) and y (latitude)
dx=1.0e6 ! 10km grid step expressed in cm
dy=1.0e6 ! dx and dy may be different
blat = 50 ! lat 50 degree, it stays constant in domain
blatr=50*pi/180.
c set depth distribution (from 200m to 20m in x direction,
c or constant equal to 200m)
do j=1,jx
do k=1,ky
h(j,k)=(201.20805-1.20805*float(j))*1.0e2
c h(j,k)=20000. ! constant depth
enddo
enddo
print *, h(30,30)
c
c Intialize constants
c
w=7.29e-05 !angular velocity of earth
rf1=3.3e-03 !coefficient of friction (in cgs) - open ocean
rf2=6.6e-03 !coefficient of friction (in cgs) - shelf
rf3=1.e-02 !coefficient of friction (in cgs) - shallow water
rf4=3.e-02 !coefficient of friction (in cgs) - on land
c
c controlling coefficients : cfric = bottom friction term
c ccor = coriolis term
c cnl2 = non-linear term
```

```
do j=1,jx
do k=1,ky
cfric(j,k) = 1.
ccor(j,k) = 1.
cnl2(j,k) = 1.
end do
end do
c t : time step in second
c
t = 100.
c maximum time steps
momax=10000
g=981.
c ah=0. ! horizontal eddy viscosity
ah=5.0e9 ! unstable in explicit scheme
c ah = 5.0e7
c ah = 2.0e7
c coefficients for the equations
c combination of space steps and time steps
pl = t/(dx)
pf = t/(dy)
pr1 = t*rf1
pr2 = t*rf2
pr3 = t*rf3
pr4 = t*rf4
pw = t*2*w
ptc=pw*sin(blatr)
ph1 = ah*t/(dx*dx)
ph2 = ah*t/(dy*dy)
gpl=g*pl
gpf=g*pf
c this variables are related to depth distributions and
c are used mainly in equation of continuity
do j=2,jx
do k=1,ky
hu(j,k)=0.5*(h(j,k)+h(j-1,k)) ! depth in the U points
end do
end do
do j=1,jx
do k=1,ky-1
hv(j,k) = (h(j,k)+h(j,k+1))*.5 ! depth in the V points
end do
end do
c ************************************************************
c * main computation loop *
c * u : horizontal particle velocity in east/west direction *
c * v : horizontal particle velocity in north/south direction *
c * z : sea level variation in cm *
c ************************************************************
c===========================================================
do 3000 mo= 1,momax
c specify wind driven forces, first seventy two time steps linear grow in time
c of taux from 0 to 2.01 tauy=0. Aand afterwards taux=2.01
motau=72
r=mo/motau
if(r.gt.1)r=1.
do j=1,jx
do k=1,ky
taux(j,k)= 2.01*r
tauy(j,k)=0.
end do
end do
c
****************************************************************
c U-Computation
c domain
do j=2,jx-1
do k=2,ky-1
vau = (vo(j,k)+vo(j-1,k)+vo(j,k-1)+vo(j-1,k-1))*.25
upos = .5*(uo(j,k)+abs(uo(j,k)))
uneg = .5*(uo(j,k)-abs(uo(j,k)))
vaup = .5*(vau+abs(vau))
vaun = .5*(vau-abs(vau))
uojk=uo(j,k)
un(j,k) = uojk+T*taux(j,k)/hu(j,k)
& - gpl*(zo(j,k)-zo(j-1,k))
& + ptc*vau*ccor(j,k)
c & +ph1*(uo(j+1,k)+uo(j-1,k)-2.*uojk) ! explicit hor. friction
c & +ph2*(uo(j,k+1)+uo(j,k-1)-2.*uojk) ! explicit hor. friction
c & -pr1*uojk*sqrt(uojk*uojk+vau*vau)/hu(j,k)*cfric(j,k) !expl.bottom
c if explicit method is used uncomment 3 above lines and delete all
c lines below down to endo
& +ph1*(uo(j+1,k)+un(j-1,k)-uojk) ! semi-expl
&+ph2*(uo(j,k+1)+un(j,k-1)-uojk) ! semi-expl
```

```
    &-pl*(upos*(uo(j,k)-uo(j-1,k))+uneg*(uo(j+1,k)-uo(j,k)))*cnl2(j,k)
    &-pf*(vaup*(uo(j,k)-uo(j,k-1))+vaun*(uo(j,k+1)-uo(j,k)))*cnl2(j,k)
    hfr2=ph1+ph2
    uni=1.+hfr2+pr1*sqrt(uojk*uojk+vau*vau)/hu(j,k)*cfric(j,k)
    un(j,k)=un(j,k)/uni
    enddo
    enddo
    c
*****************************************************************
    c
    c V-Computation
    do j=2,jx-1
    do k=2,ky-2
    uav = (un(j,k)+un(j+1,k)+un(j,k+1)+un(j+1,k+1))*.25
    uavp = .5*(uav+abs(uav))
    uavn = .5*(uav-abs(uav))
    vpos = .5*(vo(j,k)+abs(vo(j,k)))
    vneg = .5*(vo(j,k)-abs(vo(j,k)))
    vojk=vo(j,k)
    vn(j,k) = vojk+T*tauy(j,k)/hv(j,k)
    & - gpf*(zo(j,k+1)-zo(j,k))
    & - ptc*uav*ccor(j,k)
    c & +ph1*(vo(j+1,k)+vo(j-1,k)-2.*vojk)
    c & +ph2*(vo(j,k+1)+vo(j,k-1)-2.*vojk)
    c & -pr1*vojk*sqrt(vojk*vojk+uav*uav)/hv(j,k)*cfric(j,k)
    & +ph1*(vo(j+1,k)+vn(j-1,k)-vojk) ! semi-explicit
    &+ph2*(vo(j,k+1)+vn(j,k-1)-vojk) ! semi-explicit
    &-pf*(vpos*(vo(j,k)-vo(j,k-1))+vneg*(vo(j,k+1)-vo(j,k)))*cnl2(j,k)
    &-pl*(uavp*(vo(j,k)-vo(j-1,k))+uavn*(vo(j+1,k)-vo(j,k)))*cnl2(j,k)
    hfr1=ph1+ph2
    vni=1.+hfr1+pr1*sqrt(vojk*vojk+uav*uav)/hv(j,k)*cfric(j,k)
    vn(j,k)=vn(j,k)/vni
    enddo
    enddo
    c
*****************************************************************
    c
    c Z - computation
    do j=1,jx-1
    do k=2,ky-1
    zn(j,k)=zo(j,k)-pl*(un(j+1,k)*hu(j+1,k)-un(j,k)*hu(j,k))
    * - hv(j,k)*vn(j,k)*pf+pf*vn(j,k-1)*hv(j,k-1)
```

```
enddo
enddo
c ************************************************************
c
c update old variables with new
do j=1,jx
do k=1,ky
uo(j,k)=un(j,k)
vo(j,k)=vn(j,k)
zo(j,k)=zn(j,k)
enddo
enddo
c **************************************************************
c
c
if(mod(mo,20).eq.0) then
c Computing kinetic and potential energy every 20 steps
c ENI enrgy every 20 steps , averaged over whole space
NC=MO/20
ES=0.
do j=1,jx-1
do k=2,ky
ua=0.5*(un(j,k)+un(j+1,k))
va=0.5*(vn(j,k)+vn(j,k-1))
en=h(j,k)*(ua*ua+va*va)/(2.*g)+zn(j,k)*zn(j,k)/2.
es=en+es
end do
end do
eni(nc)=es
write(25,*)es
endif
if(mod(mo,20).eq.0) then
print *, 'MO=',mo, 'entrance=',zn(2,10)
print *, 'east basin=',zn(50,10)
print *,'west basin=',zn(125,10)
write(20,*),mo,zn(2,10),zn(50,10),zn(145,10)
endif
if(mod(mo,8640).eq.0) then
c sea level and velocity in whole domain at day 10
write(10,*)((zn(j,k),j=1,jx),k=1,ky)
write(21,*)((un(j,k),j=1,jx),k=1,ky)
write(22,*)((vn(j,k),j=1,jx),k=1,ky)
```

```
endif
3000 continue
stop
end
c=====================================
```

This simple program open numerous possibilities to study process of storm surge generation and propagations under various conditions. Before embarking this train one should carefully study numerical properties of the algorithm used. Some coefficients in the numerical system are either variable or not well established. To such category belongs horizontal eddy viscosity coefficient. A few experiments with an explicit numerical formulation for the horizontal eddy viscosity will demonstrate that this numerical scheme is both unstable for the large and small eddy viscosity coefficients. Does window for the numerical stability cover the whole range of necessary variation from the physical point of view? Surely not, and therefore, we have introduced a semi-explicit scheme with stronger stability properties. A few results are given in Figs. II.10 and II.11. In Fig.II.10 the sea level distribution in the whole domain is depicted at day 10 of the process. As one should expect along a shallow eastern shoreline the sea level is high and at the western shoreline the sea level is negative. It is always important to ask whether this result is correct. In the case of the sea level any forcing involved in the closed domain should retain an average sea level equal to zero. Therefore, to test this assumption the results of computations for the sea level should be summed up over all grid points.

In Fig.II.11 we depicted total energy of the system. Due to steady wind one can expect that the energy of the system will arrive to constant value as well. It does, but quite slowly, depicting some decaying oscillations. We have done computations with the two horizontal eddy viscosity coefficients and clearly the large coefficient damps oscillations much faster. Temporal behavior of the energy is motivating further important questions. First, we may ask: what are these oscillations? After elucidating the nature of the oscillations, we may ask more important question: can we use this approach to study these oscillations? So, at this juncture we are a bit away from our main topic i.e., study of the storm surges, but serendipity in numerical computations happens through the skillful invoking of the side questions.

Fig.II.10: SEA LEVEL (cm), DAY 10

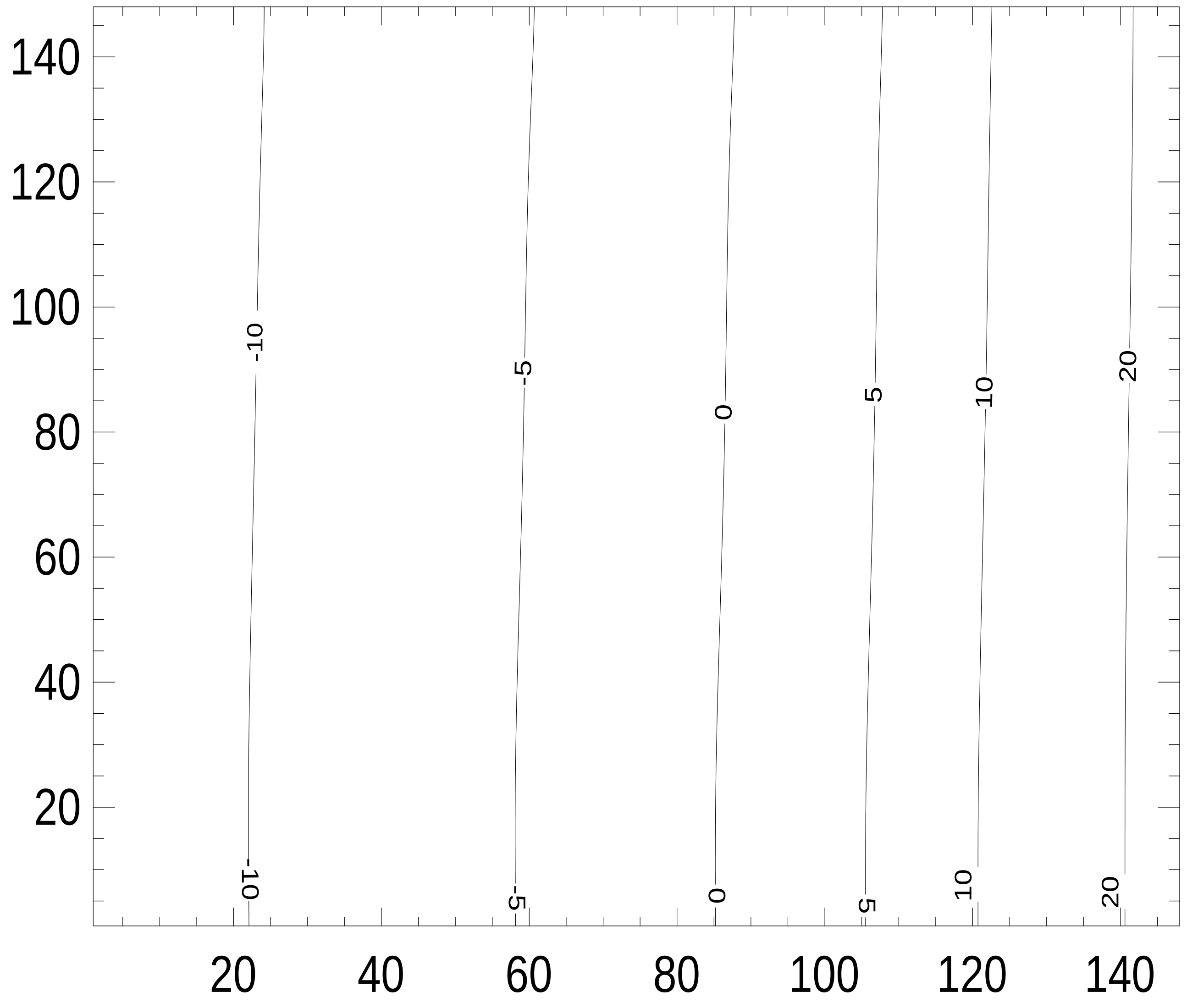
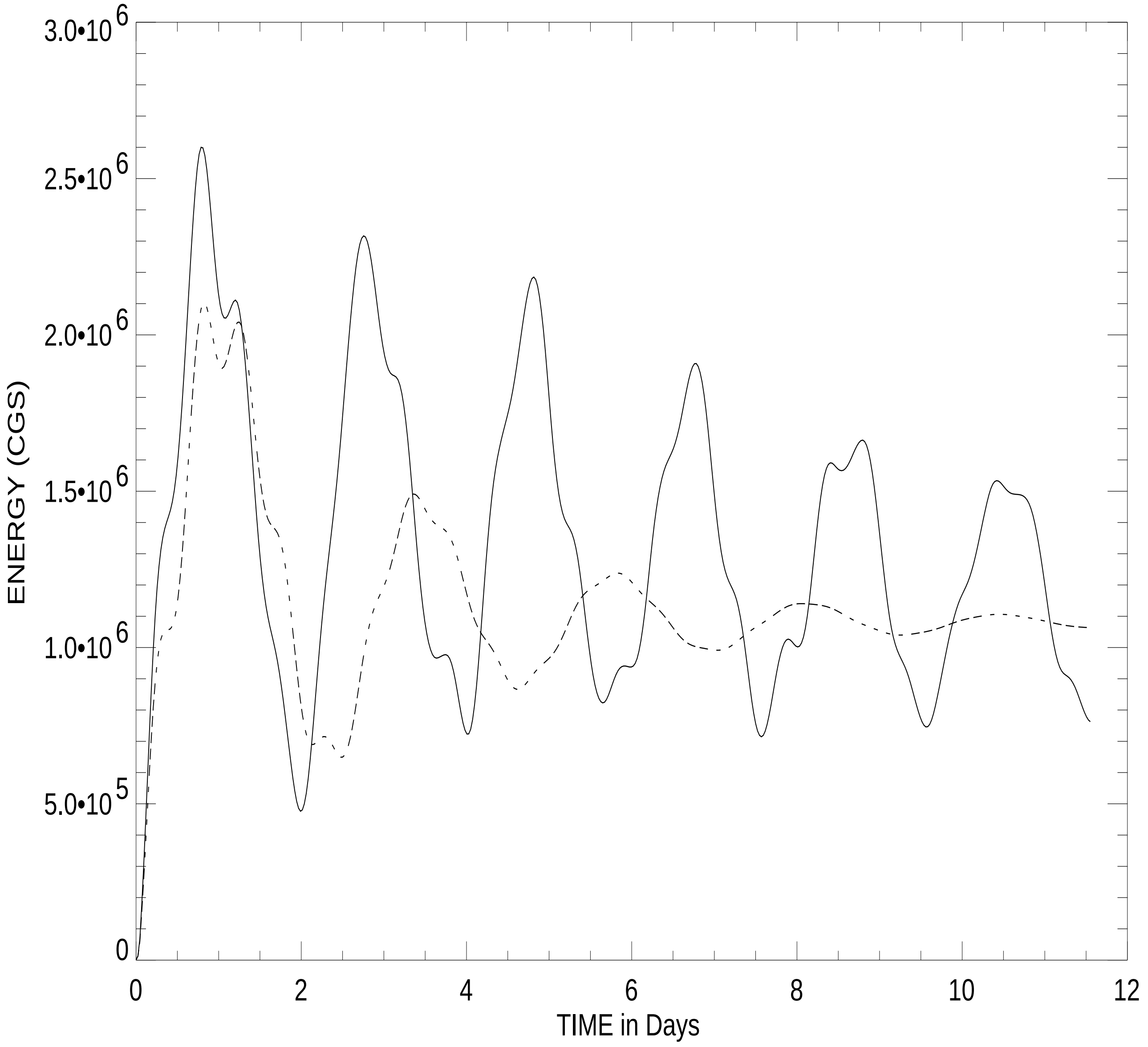


Fig.II.11: TOTAL ENERGY



Problem No. 8, storm surge in the non-rectangular domain

To calculate storm surges in the non-rectangular domains we need to learn how to account for the complicated shape of the domain. In irregular water bodies we first learn how to enumerate the grid points which are located in water. An explanation of such procedure is given on p. 134 of NM. We shall explore first this procedure to establish the set of indices for the horizontal u velocity computation. Thus checking each row from k=1 to k=ky, one can locate in every row the starting point (js) and the ending point (je) of the water domain along the x axis. The depth over the land and along the coastal lines is set to zero. Because of islands it may happen that in the some row (same k) a few different starting and ending points will be prescribed. This is done in the program by counting the number of segments. An individual segment in the program is called, a line. Proceeding in similar way from the first column (j=1) to the last column (j=jx), one can establish beginning and ending indices for the v velocity integration along the y direction. This is all done in the FORTRAN program named index.f (see Chap.3 of this Workbook). The program reads depth distribution from the file parab.dat (again, see Chap.3) and outputs indices for the u, v and sea level integration. It also calculates the number of lines along the x and y direction, therefore every set of indices can be related to the proper line number.

The program to compute storm surges is called surge1.f and is listed below. The main difference between surge.f and surge1.f occurs in the space integration. Velocity along x direction is computed in the loop 4. First this loop counts line number. This number proceeds from 1 to nlh, which is the total number of rows (horizontal lines). Beginning of every line is given by js(n)+1, and the end is set by je(n). We keep also the index of an row (k index). Horizontal lines also serve to calculate the new sea level (loop 13). Here index j starts from js(n), i.e., the number of the sea level points is equal to the number of u velocity points plus one. Calculation of the v velocity is proceeding along the vertical lines (columns). Total number of columns is equal to nlv.

```
C PROGRAM SURGE1.F
C ORIGINATED BY Z.K. AT IMS
c program to compute storm surge in the 2-d domains of any shape
parameter (jx=21,ky=21,nlh=17,nlv=17)
dimension uo(jx,ky),un(jx,ky),vo(jx,ky),vn(jx,ky),
* zo(jx,ky),zn(jx,ky)
c uo, un, vo, vn old and new velocity component along x (longitude)
c and along y (latitude)
c zo and zn old and new sea level
dimension h(jx,ky),hu(jx,ky),hv(jx,ky),eni(500)
c depth h and depth dependent functions hu,hv (averages along x and y direc-
tions)
```

```
c eni is total energy of the system
c
dimension cfric(jx,ky),ccor(jx,ky),cnl2(jx,ky)
c cfric,ccor,cnl2 coefficients equal to 1 or 0 to switch off and on
c friction, coriolis and nonlinear terms
dimension taux(jx,ky),tauy(jx,ky)
c wind stress along x and y
integer js(nlh),je(nlh),ks(nlh),ke(nlh)
c indices along x direction
integer jsv(nlv),jev(nlv),ksv(nlv),kev(nlv)
c indices along y direction
open (unit=4,file='parab.dat',status='unknown')
c depth file
open (unit=3,file='parab.ind',status='unknown')
c file to keep indices
c read depth
read(4,*)((h(j,k),j=1,JX),k=1,ky)
c Index : horizontal lines and vertical lines
do i=1,nlh
read (3,100) ks(i),ke(i),js(i),je(i)
100 format(4i4)
enddo
do i=1,nlv
read (3,100) jsv(i),jev(i),ksv(i),kev(i)
enddo
c
close (3)
open (unit=10,file='zn.dat',status='unknown') ! sea level
open (unit=20,file='momaxm2.dat',status='unknown') ! time series
open (unit=21,file='un.dat',status='unknown') ! u-velocity
open (unit=22,file='vn.dat',status='unknown') ! v-velocity
open (unit=25,file='enim2.dat',status='unknown') ! energy
pi = 4.*atan(1.)
rad = pi/180.
c grid size along x (longitude) and y (latitude)
dx=1.0e6 ! 10km grid step expressed in cm
dy=1.0e6 ! dx and dy may be different
blat = 50 ! lat 50 degree, it stays constant in domain
blatr=50*pi/180.
print *, h(8,8)
c
c Intialize constants
```

```
c
w=7.29e-05 !angular velocity of earth
rf1=3.3e-03 !coefficient of friction (in cgs) - open ocean
rf2=6.6e-03 !coefficient of friction (in cgs) - shelf
rf3=1.e-02 !coefficient of friction (in cgs) - shallow water
rf4=3.e-02 !coefficient of friction (in cgs) - on land
c
c controlling coefficients : cfric = bottom friction term
c ccor = coriolis term
c cnl2 = non-linear term
do j=1,jx
do k=1,ky
cfric(j,k) = 1.
ccor(j,k) = 1.
cnl2(j,k) = 1.
end do
end do
c t : time step in second
c
t = 100.
c maximum time steps
momax=10000
g=981.
c ah=0. ! horizontal eddy viscosity
c ah=5.0e9 ! unstable in explicit scheme
c ah = 5.0e7
ah = 2.0e7
c coefficients for the equations
c combination of space steps and time steps
pl = t/(dx)
pf = t/(dy)
pr1 = t*rf1
pr2 = t*rf2
pr3 = t*rf3
pr4 = t*rf4
pw = t*2*w
ptc=pw*sin(blatr)
ph1 = ah*t/(dx*dx)
ph2 = ah*t/(dy*dy)
gpl=g*pl
gpf=g*pf
c this variables are related to depth distributions and
```

```
c are used mainly in equation of continuity
do j=2,jx
do k=1,ky
hu(j,k)=0.5*(h(j,k)+h(j-1,k)) ! depth in the U points
end do
end do
do j=1,jx
do k=1,ky-1
hv(j,k) = (h(j,k)+h(j,k+1))*.5 ! depth in the V points
end do
end do
c ********************************************************
c * main computation loop *
c * u : horizontal particle velocity in east/west direction *
c * v : horizontal particle velocity in north/south direction *
c * z : sea level variation in cm *
c ********************************************************
c==============================================
do 3000 mo= 1,momax
c specify wind driven forces, first seventy two time steps linear grow in time
c of taux from 0 to 2.01 tauy=0. Aand afterwards taux=2.01
motau=72
r=mo/motau
if(r.gt.1)r=1.
do j=1,jx
do k=1,ky
taux(j,k)= 2.01*r
tauy(j,k)=0.
end do
end do
c
************************************************************
c U-Computation
DO 4 n=1,nlh
do 4 j=js(n)+1,je(n)
do 4 k=ks(n),ke(n)
c domain
vau = (vo(j,k)+vo(j-1,k)+vo(j,k-1)+vo(j-1,k-1))*.25
upos = .5*(uo(j,k)+abs(uo(j,k)))
uneg = .5*(uo(j,k)-abs(uo(j,k)))
vaup = .5*(vau+abs(vau))
vaun = .5*(vau-abs(vau))
```

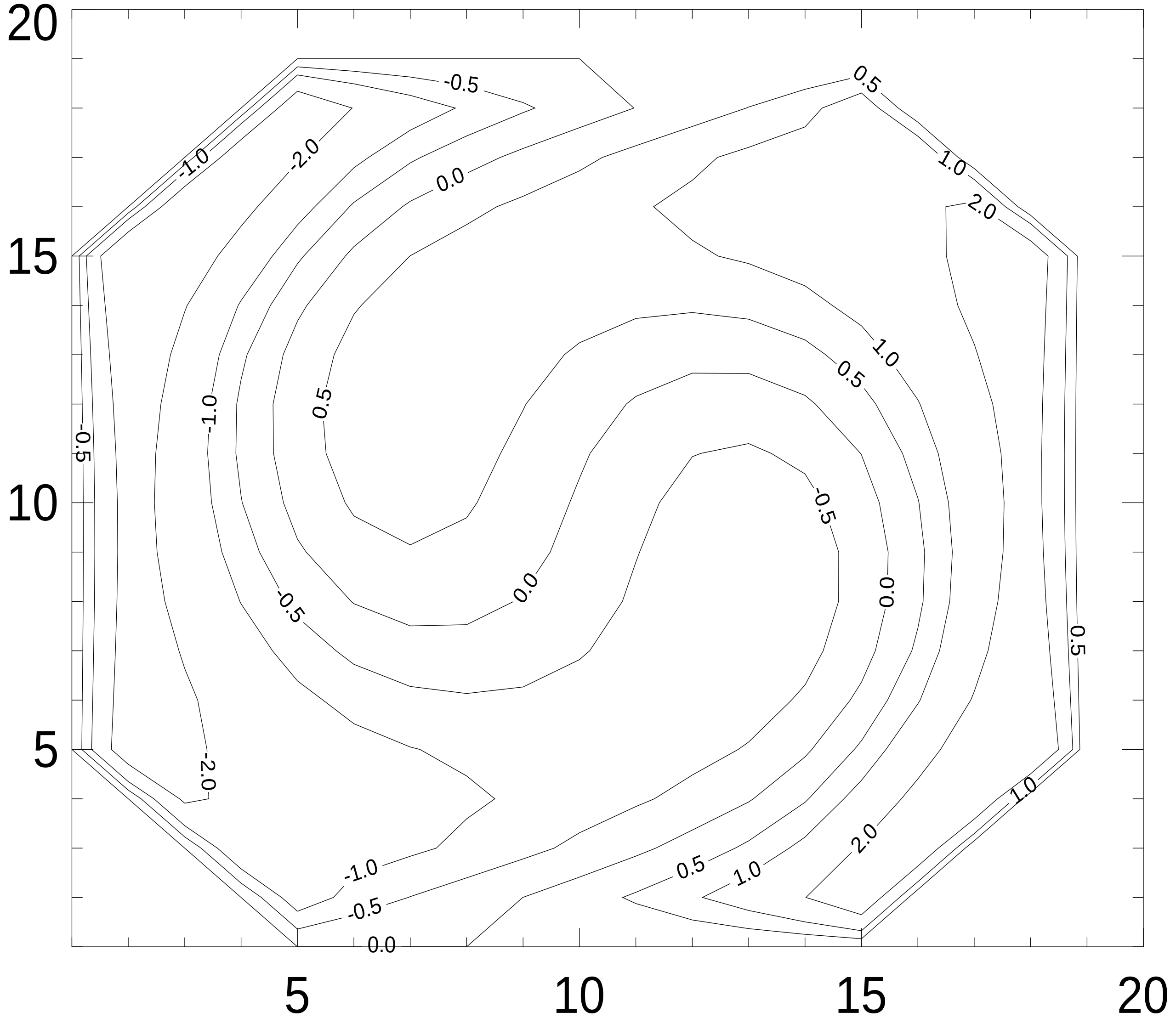
```
     uojk=uo(j,k)
     un(j,k) = uojk+T*taux(j,k)/hu(j,k)
     & - gpl*(zo(j,k)-zo(j-1,k))
     & + ptc*vau*ccor(j,k)
     c & +ph1*(uo(j+1,k)+uo(j-1,k)-2.*uojk) ! explicit hor. friction
     c & +ph2*(uo(j,k+1)+uo(j,k-1)-2.*uojk) ! explicit hor. friction
     c & -pr1*uojk*sqrt(uojk*uojk+vau*vau)/hu(j,k)*cfric(j,k) !expl.bottom
     c if explicit method is used uncomment 3 above lines and delete all
     c lines below down to endo
     & +ph1*(uo(j+1,k)+un(j-1,k)-uojk) ! semi-expl
     &+ph2*(uo(j,k+1)+un(j,k-1)-uojk) ! semi-expl
     &-pl*(upos*(uo(j,k)-uo(j-1,k))+uneg*(uo(j+1,k)-uo(j,k)))*cnl2(j,k)
     &-pf*(vaup*(uo(j,k)-uo(j,k-1))+vaun*(uo(j,k+1)-uo(j,k)))*cnl2(j,k)
     hfr2=ph1+ph2
     uni=1.+hfr2+pr1*sqrt(uojk*uojk+vau*vau)/hu(j,k)*cfric(j,k)
     un(j,k)=un(j,k)/uni
     4 continue
     c
*****************************************************************
     c
     c V-Computation
     do 2001 m=1,nlv
     c
     do 2001 j=jsv(m),jev(m)
     do 2001 k=ksv(m),kev(m)-1
     uav = (un(j,k)+un(j+1,k)+un(j,k+1)+un(j+1,k+1))*.25
     uavp = .5*(uav+abs(uav))
     uavn = .5*(uav-abs(uav))
     vpos = .5*(vo(j,k)+abs(vo(j,k)))
     vneg = .5*(vo(j,k)-abs(vo(j,k)))
     vojk=vo(j,k)
     vn(j,k) = vojk+T*tauy(j,k)/hv(j,k)
     & - gpf*(zo(j,k+1)-zo(j,k))
     & - ptc*uav*ccor(j,k)
     c & +ph1*(vo(j+1,k)+vo(j-1,k)-2.*vojk)
     c & +ph2*(vo(j,k+1)+vo(j,k-1)-2.*vojk)
     c & -pr1*vojk*sqrt(vojk*vojk+uav*uav)/hv(j,k)*cfric(j,k)
     & +ph1*(vo(j+1,k)+vn(j-1,k)-vojk) ! semi-explicit
     &+ph2*(vo(j,k+1)+vn(j,k-1)-vojk) ! semi-explicit
     &-pf*(vpos*(vo(j,k)-vo(j,k-1))+vneg*(vo(j,k+1)-vo(j,k)))*cnl2(j,k)
     &-pl*(uavp*(vo(j,k)-vo(j-1,k))+uavn*(vo(j+1,k)-vo(j,k)))*cnl2(j,k)
     hfr1=ph1+ph2
```

```
vni=1.+hfr1+pr1*sqrt(vojk*vojk+uav*uav)/hv(j,k)*cfric(j,k)
vn(j,k)=vn(j,k)/vni
2001 continue
c
*****************************************************************
c
c Z - computation
do 13 n=1,nlh
do 13 j=js(n),je(n)
do 13 k=ks(n),ke(n)
zn(j,k)=zo(j,k)-pl*(un(j+1,k)*hu(j+1,k)-un(j,k)*hu(j,k))
* - hv(j,k)*vn(j,k)*pf+pf*vn(j,k-1)*hv(j,k-1)
13 continue
c ************************************************************
c
c update old variables with new
do j=1,jx
do k=1,ky
uo(j,k)=un(j,k)
vo(j,k)=vn(j,k)
zo(j,k)=zn(j,k)
enddo
enddo
c **************************************************************
c
c
if(mod(mo,20).eq.0) then
c Computing kinetic and potential energy every 20 steps
c ENI enrgy every 20 steps , averaged over whole space
NC=MO/20
ES=0.
do j=1,jx-1
do k=2,ky
ua=0.5*(un(j,k)+un(j+1,k))
va=0.5*(vn(j,k)+vn(j,k-1))
en=h(j,k)*(ua*ua+va*va)/(2.*g)+zn(j,k)*zn(j,k)/2.
es=en+es
end do
end do
eni(nc)=es
write(25,*)es
endif
```

```
if(mod(mo,20).eq.0) then
print *, 'MO=',mo, 'entrance=',zn(2,8)
print *, 'east basin=',zn(4,8)
print *,'west basin=',zn(15,8)
write(20,*),mo,zn(2,8),zn(4,8),zn(15,8)
endif
if(mod(mo,8640).eq.0) then
c sea level and velocity in whole domain at day 10
write(10,*)((zn(j,k),j=1,jx),k=1,ky)
write(21,*)((un(j,k),j=1,jx),k=1,ky)
write(22,*)((vn(j,k),j=1,jx),k=1,ky)
endif
3000 continue
stop
end
```

The sea level distribution in the parabolic basin after 10 days of process is given in Fig.II.12. An intricate pattern of the isolines can be changed through application of the various horizontal eddy viscosities.

Fig.II.12: SEA LEVEL (cm), DAY 10



Problem No. 9, storm surge in the rectangular domain – migration to Fortran 90

Fortran 90 is becoming more popular, hence, below we describe two numerical experiments in this language. First, storm surge computations given by the program surge.f in the rectangular domain are repeated here. One major topic to be remembered that the tricks which worked well in Fortran 77 may not work so well in the Fortran 90. The Fortran 77 through "do loops" allowed sequential computations while the Fortran 90 is directed towards parallel computations without sequential "do loops". Well, actually we still can keep do loops in the Fortran 90 as well. Here the main problem occured with the semi-implicit form for the horizontal friction. This form allowed us to use a somewhat larger eddy viscosity as compared to the explicit formulas. Unfortunately, this new program is using only explicit numerical formulas. Entire program has all necessary comments to understand the computational process and comparison with program surge.f can be very helpful. The program organization is such that the main parameters are located in the module mparameter.f90. This module will be use by all programs written in Fortran90.

```
! Last change: ZK 1 Aug 100 12:27 pm
MODULE MPARAMETER
REAL, PARAMETER::G=981.
REAL,PARAMETER:: PI=3.141592653589793238462643383279502884197
REAL, PARAMETER::RAD= 0.0174532930056254
!EARTH'S ROTATION
REAL, PARAMETER::ROT=7.2919E-5
!EARTH'S RADIUS
REAL, PARAMETER::REA=6378.e5
! BOTTOM FRICTION COEFFICIENT RF1
REAL, PARAMETER::RF1=3.3E-3
!TIDAL PERIODS
REAL,PARAMETER::TM2=44714.1353
REAL,PARAMETER::TS2=43200.03035
REAL,PARAMETER::TN2=45569.9578
REAL,PARAMETER::TK1=86164.2796
REAL,PARAMETER::TO1=92949.2845
END MODULE MPARAMETER
========================================
! Last change: ZK 1 Aug 100 12:22 pm
PROGRAM SURGE
!PROGRAM SURGE.F90
! MOST IMPORTANT TO REMEMBER WHEN MIGRATING TO FOR-
TRAN99
```

```
!,HFR2,ES,ENIS THAT NUMERICAL APPROACH RELATED TO SE-
QUENTIAL
!COMPUTATION
! IS NOT WORKING ANY MORE. COMPUTATIONS ARE DONE IN PAR-
ALELL
!AND NOT SEQUENTIALLY. THEREFORE THE SEMI-IMPLICIT
METHOD FOR
! THE HORIZONTAL FRICTION DOES NOT WORK ANY MORE
!——————-
USE MPARAMETER
!——————————
IMPLICIT NONE
! MOMAX DENOTES NUMBER OF TIME STEPS
INTEGER, PARAMETER::JX=150,KY=150,MOMAX=10000
INTEGER ::MO, MOTAU,J,K
REAL,DIMENSION(1:JX,1:KY)::TAUX,TAUY,UO,UN,VO,VN,VAU,UNI
REAL,DIMENSION(1:JX,1:KY)::UPOS,H,HU,VNEG,ZN,ZO,VPOS,UAVN
REAL,DIMENSION(1:JX,1:KY)::UNEG,VAUP,VAUN,HV,UAV,UAVP
REAL::R,T,DX,DY,PL,PF,GPL,PW,PTC,BLAT,AH,PH1,PH2,HFR1
REAL::UA,VA,PR1,GPF,HFR2,ES,EN
OPEN(UNIT=10,FILE='ZN.DAT',STATUS='UNKNOWN')
! SEA LEVEL IN SPACE
OPEN(UNIT=25,FILE='ENIM2.DAT',STATUS='UNKNOWN') ! ENERGY
!PRESCRIBE DEPTH: EITHER VARIABLE ALONG X (J) DIRECTION
OR
! CONSTANT AND EQUAL TO 200M
DO K =1,KY
DO J=1,JX
H(J,K)=(201.20805-1.20805*FLOAT(J))*1.0E2 ! IN CM
! H(J,K)=20000.
END DO
END DO
! SPACE STEPS
DX=1.0E6 !10KM IN CM
DY=1.0E6
AH=2.0E7 ! HORIZONTAL EDDY VISCOSITY
! STATE LATITUDE IN DEGREE
BLAT=50.
BLAT=BLAT*RAD ! THIS ONE IS EXPRESSED IN RADIANS
!T IS TIME STEP
T=100.
! COMBINATION OF SPACE AND TIME PARAMETERS
```

```
    PL=T/DX
    PF=T/DY
    PR1=T*RF1
    PW=T*2.*ROT
    GPL=G*PL
    GPF=G*PF
    PTC=PW*SIN(BLAT)
    PH1=AH*T/(DX*DX)
    PH2=AH*T/(DX*DX)
    PRINT *, 'ROT=',ROT,'PR1=',PR1
    ! SET DEPTH IN THE U POINTS
    HU(2:JX,1:KY)=0.5*(H(2:JX,1:KY)+H(1:JX-1,1:KY))
    ! SET DEPTH IN THE V POINTS
    HV(1:JX,1:KY-1)=0.5*(H(1:JX,1:KY-1)+H(1:JX,2:KY))
    ! MOMAX TOTAL NUMBER OF TIME STEPS
    PRINT*,'HV=',HV(10,11)
    uO=0.
    un=0.
    vO=0.
    vn=0.
    TIMELOOP: DO MO=1,MOMAX
    !  SPECIFY WIND DRIVEN FORCES. APPLY EXTERNAL FORCES
SLOWLY
    ! SO THE SYSTEM CAN ADJUST TO THEM. MOTAU DENOTES INI-
TIAL
    ! NUMBER OF STEPS FOR EXTERNAL FORCE TO GROW LINEARLY.
    MOTAU=72
    R=FLOAT(MO)/FLOAT(MOTAU)
    IF(R.GT.1.)R=1.
    ! TAUX IS WIND STRESS ALONG X DIRECTION
    !TAUY IS WIND STRESS ALONG Y DIRECTION
    TAUX=2.01*R
    TAUY=0.
    ! U COMPUTATION COMPUTATION
    DO K =2,KY
    DO J=2,JX-1
    VAU(J,K)=(VO(J-1,K)+ VO(J,K-1)+VO(J-1,K-1) &
    +VO(J,K))*0.25
    UPOS(J,K)=0.5*(UO(J,K)+ABS(UO(J,K)))
    UNEG(J,K) =0.5*(UO(J,K)-ABS(UO(J,K)))
    VAUP(J,K)=0.5*(VAU(J,K)+ABS(VAU(J,K)))
    VAUN(J,K)=0.5*(VAU(J,K)-ABS(VAU(J,K)))
```

```
END DO
END DO
UN(2:JX-1,2:KY-1)= UO(2:JX-1,2:KY-1)+T*TAUX(2:JX-1,2:KY-1)&
/HU(2:JX-1,2:KY-1)&
-GPL*(ZO(2:JX-1,2:KY-1)-ZO(1:JX-2,2:KY-1))&
+PTC*VAU(2:JX-1,2:KY-1) &
+PH1*(UO(3:JX,2:KY-1)+UO(1:JX-2,2:KY-1)-2.*UO(2:JX-1,2:KY-1))&
+PH2*(UO(2:JX-1,3:KY)+UO(2:JX-1,1:KY-2)-2.*UO(2:JX-1,2:KY-1))&
-PL*UPOS(2:JX-1,2:KY-1)*(UO(2:JX-1,2:KY-1)-UO(1:JX-2,2:KY-1))&
-PL*UNEG(2:JX-1,2:KY-1)*(UO(3:JX,2:KY-1)-UO(3:JX,2:KY-1))&
-PF*VAUP(2:JX-1,2:KY-1)*(UO(2:JX-1,2:KY-1)-UO(2:JX-1,1:KY-2))&
-PF*VAUN(2:JX-1,2:KY-1)*(UO(2:JX-1,3:KY)-UO(2:JX-1,2:KY-1))&
-PR1*UO(2:JX-1,2:KY-1)*SQRT(UO(2:JX-1,2:KY-1)*UO(2:JX-1,2:KY-1)&
+VAU(2:JX-1,2:KY-1)*VAU(2:JX-1,2:KY-1))/HU(2:JX-1,2:KY-1)
!==========================================
! COMPUTATION OF THE V COMPONENT
UAV(2:JX-1,2:KY-2)=0.25*(UN(2:JX-1,2:KY-2)+UN(3:JX,2:KY-2)&
+ UN(2:JX-1,3:KY-1)+UN(3:JX,3:KY-1))
UAVP(2:JX-1,2:KY-2)=0.5*(UAV(2:JX-1,2:KY-2)&
+ABS(UAV(2:JX-1,2:KY-2)))
UAVN(2:JX-1,2:KY-2)=0.5*(UAV(2:JX-1,2:KY-2)&
-ABS(UAV(2:JX-1,2:KY-2)))
VPOS(2:JX-1,2:KY-2)=0.5*(VO(2:JX-1,2:KY-2)&
+ABS(VO(2:JX-1,2:KY-2)))
VNEG(2:JX-1,2:KY-2)=0.5*(VO(2:JX-1,2:KY-2)&
-ABS(VO(2:JX-1,2:KY-2)))
VN(2:JX-1,2:KY-2)=VO(2:JX-1,2:KY-2)+T*TAUY(2:JX-1,2:KY-2)/&
HV(2:JX-1,2:KY-2)&
-GPF*(ZO(2:JX-1,3:KY-1)-ZO(2:JX-1,2:KY-2))&
-PTC*UAV(2:JX-1,2:KY-2)&
+PH1*(VO(3:JX,2:KY-2)+VO(1:JX-2,2:KY-2)-2.*VO(2:JX-1,2:KY-2))&
+PH2*(VO(2:JX-1,3:KY-1)+VO(2:JX-1,1:KY-3)-2.*VO(2:JX-1,2:KY-2))&
-PF*VPOS(2:JX-1,2:KY-2)*(VO(2:JX-1,2:KY-2)-VO(2:JX-1,1:KY-3))&
-PF*VNEG(2:JX-1,2:KY-2)*(VO(2:JX-1,3:KY-1)-VO(2:JX-1,2:KY-2))&
-PL*UAVP(2:JX-1,2:KY-2)*(VO(2:JX-1,2:KY-2)-VO(1:JX-2,2:KY-2))&
-PL*UAVN(2:JX-1,2:KY-2)*(VO(3:JX,2:KY-2)-VO(2:JX-1,2:KY-2))&
-PR1*VO(2:JX-1,2:KY-2)*SQRT(VO(2:JX-1,2:KY-2)*VO(2:JX-1,2:KY-2)&
+UAV(2:JX-1,2:KY-2)*UAV(2:JX-1,2:KY-2))/HV(2:JX-1,2:KY-2)
!SEA LEVEL
ZN(1:JX-1,2:KY-1)=ZO(1:JX-1,2:KY-1)&
-PL*UN(2:JX,2:KY-1)*HU(2:JX,2:KY-1)&
+PL*UN(1:JX-1,2:KY-1)*HU(1:JX-1,2:KY-1)&
```

```
-PF*HV(1:JX-1,2:KY-1)*VN(1:JX-1,2:KY-1)&
+PF*HV(1:JX-1,1:KY-2)*VN(1:JX-1,1:KY-2)
!UPDATE OLD VARIABLE WITH NEW ONES
UO=UN
VO=VN
ZO=ZN
IF(MOD(MO,20).EQ.0)THEN
!COMPUTE KINETIC AND POTENTIAL ENERGY
ES=0.
DO J=1,JX-1
DO K=2,KY
UA=0.5*(UN(J,K)+UN(J+1,K))
VA=0.5*(VN(J,K)+VN(J,K-1))
EN=H(J,K)*(UA*UA+VA*VA)/(2.*G)+ZN(J,K)*ZN(J,K)/2.
ES=EN+ES
END DO
END DO
WRITE(25,*)ES
PRINT *, 'MO=',MO,'WEST=',ZN(3,75)
PRINT*,'CENTER=',ZN(75,75),'EAST=',ZN(145,75)
END IF
IF(MOD(MO,8640).EQ.0)THEN
DO K=1,KY
WRITE(10,*)(ZN(J,K),J=1,JX)
END DO
END IF
END DO TIMELOOP
END PROGRAM SURGE
```

Problem No. 10, storm surge in the non-rectangular domain – migration to Fortran 90

In this program we describe the problem solved in program surge1.f, but both language and approach are very different. Here the computations are done in the entire rectangular domain, along x (from 1 to JX) and along y (from 1 to KY). Only after computation is done (for each time step) the results in the grid points located over the land are multiplied by zero. Such approach somewhat simplifies programming. Although, we still need to differentiate between wet and dry points. This is achieved through the set of masking indices. After the depth is read from the file parab.dat, and indices from the file parab.ind, we use the indices to establish three sets of masking parameters: FSM, DUM, and DVM. FSM is equal to zero over land, and to 1 over water. DUM is accounting for the wet and dry *u* velocity points and DVM is equal to one in the wet

v points and zero in the dry *v* points. The marching in time is done in such a fashion that the new velocity along the x direction is multiplied by DUM to account for the land, and similarly the new values of VN are multiplied by DVM and the new values of the sea level by FSM. The program surge1.f90 is given below.
==

```
    ! Last change: ZK 3 Aug 100 1:12 pm
    PROGRAM SURGE1
    !PROGRAM SURGE1.F90
    !A DIFFERENT APPROACH TO integration
    !through the maskindices
    !——————————-
    USE MPARAMETER
    IMPLICIT NONE
    ! MOMAX DENOTES NUMBER OF TIME STEPS
    INTEGER,
PARAMETER::JX=21,KY=21,NLH=17,NLV=17,MOMAX=10000
    INTEGER ::MO, MOTAU,J,K,I,M,N
    REAL,DIMENSION(1:JX,1:KY)::TAUX=0.,TAUY=0.,UO=0.,UN=0.,VO=0.
    REAL,DIMENSION(1:JX,1:KY)::VAU=0.,UNI=0.,UPOS=0.,H=0.,HU=0.
    REAL,DIMENSION(1:JX,1:KY)::UNEG=0.,VAUP=0.,VAUN=0.,HV=0.
    REAL,DIMENSION(1:JX,1:KY)::VN=0.,VNEG=0.,UAV=0.,VPO=0.
    REAL,DIMENSION(1:JX,1:KY)::UAVP=0.,UAVN=0.,VPOS=0.
    REAL,DIMENSION(1:JX,1:KY)::FSM=0.,DUM=0.,DVM=0.,ZN=0.,ZO=0.
    REAL::R,T,DX,DY,PL,PF,GPL,PW,PTC,BLAT,AH,PH1,PH2,HFR1,HFR2
    REAL::UA,VA,PR1,GPF,ES,EN
    INTEGER:: js(nlh),je(nlh),ks(nlh),ke(nlh)
    ! indices along x direction
    INTEGER:: jsv(nlv),jev(nlv),ksv(nlv),kev(nlv)
    ! indices along y direction
    OPEN(UNIT=25,FILE='ENIM2.DAT',STATUS='UNKNOWN') ! ENERGY
    OPEN(UNIT=10,FILE='ZN.DAT',STATUS='UNKNOWN') ! SEA LEVEL
IN SPACE
    OPEN(UNIT=3,FILE='parab.ind',STATUS='UNKNOWN')
    OPEN(UNIT=1,FILE='Parab.dat',STATUS='UNKNOWN')
    ! Index : horizontal lines and vertical lines
    do i=1,nlh
    read (3,100) ks(i),ke(i),js(i),je(i)
    100 format(4i4)
    enddo
    do i=1,nlv
    read (3,100) jsv(i),jev(i),ksv(i),kev(i)
    enddo
```

```
close (3)
! THIS IS FOR MASK INDECS TO SHOW WHERE U OR V ARE ZERO
AND
! WHERE DEPTH DENOTE SHORE LINE (depth is zero)
! THE ACTUAL COMPUTATIONS IS PERFORMED OVER THE ENTIRE
DOMAIN
!AND AFTERWARDS THE VELOCITIES AND SEA LEVEL IS MULTI-
PLIED
! BY THE MASK INDEX. FSM FOR THE SEA LEVEL, DUM FOR THE U
! VELOCITY AND DVM FORMAT (THE V VELOCITY)
FSM=0.
DUM=0.
DVM=0.
DO 4 n=1,nlh
do 4 j=js(n)+1,je(n)
do 4 k=ks(n),ke(n)
DUM(J,K)=1.
4 CONTINUE
DO 2001 M=1,NLV
do 2001 j=jsv(m),jev(m)
do 2001 k=ksv(m),kev(m)-1
DVM(J,K)=1.
2001 CONTINUE
do 13 n=1,nlh
do 13 j=js(n),je(n)
do 13 k=ks(n),ke(n)
FSM(J,K)=1.0
13 CONTINUE
READ(1,*)((H(J,K),J=1,JX),K=1,KY)
WHERE (H.LT.100.)
H=100.
END WHERE
! NOW WE WILL INTEGRATE IN THE ENTIRE DOMAIN, AND JUST
IN CASE OF
!DIVISION OVER ZERO DEPTH WE PUT 100CM OVER LAND
! BECAUSE COMPUTATION IS REGULATED BY MASKS THERE IS NO
NEED
! FOR DIFFERENT STARTING AND ENDING INDECES FOR THE VE-
LOCITY
! AND SEA LEVEL AS LONG AS THESE INDECES ARE STARTING
! AND ENDING OVER LAND
! SPACE STEPS
```

```
DX=1.0E6 !10KM IN CM
DY=1.0E6
AH=2.0E7 ! HORIZONTAL EDDY VISCOSITY
! STATE LATITUDE IN DEGREE
BLAT=50.
BLAT=BLAT*RAD ! THIS ONE IS EXPRESSED IN RADIANS
!T IS TIME STEP
T=100.
! COMBINATION OF SPACE AND TIME PARAMETERS
PL=T/DX
PF=T/DY
PR1=T*RF1
PW=T*2.*ROT
GPL=G*PL
GPF=G*PF
PTC=PW*SIN(BLAT)
PH1=AH*T/(DX*DX)
PH2=AH*T/(DX*DX)
PRINT *, 'ROT=',ROT,'PR1=',PR1
! SET DEPTH IN THE U POINTS
HU(2:JX,1:KY)=0.5*(H(2:JX,1:KY)+H(1:JX-1,1:KY))
! SET DEPTH IN THE V POINTS
HV(1:JX,1:KY-1)=0.5*(H(1:JX,1:KY-1)+H(1:JX,2:KY))
!START A GENERAL TIME LOOP
! MOMAX TOTAL NUMBER OF TIME STEPS
PRINT*,'HV=',HV(10,11)
TIMELOOP: DO MO=1,MOMAX
! SPECIFY WIND DRIVEN FORCES. aPPLY EXTERNAL FORCES
SLOWLY
!SO THE SYSTEM CAN ADJUST TO THEM. MOTAU DENOTES INITIAL
! NUMBER OF STEPS FOR EXTERNAL FORCE TO GROW LINEARLY.
MOTAU=72
R=FLOAT(MO)/FLOAT(MOTAU)
IF(R.GT.1.)R=1.
! TAUX IS WIND STRESS ALONG X DIRECTION
!TAUY IS WIND STRESS ALONG Y DIRECTION
TAUX=2.01*R
TAUY=0.
! U COMPUTATION
DO K=2,KY-1
DO J=2,JX-1
VAU(J,K)=(VO(J-1,K)+ VO(J,K-1)+VO(J-1,K-1) &
+VO(J,K))*0.25
UPOS(J,K)=0.5*(UO(J,K)+ABS(UO(J,K)))
UNEG(J,K) =0.5*(UO(J,K)-ABS(UO(J,K)))
VAUP(J,K)=0.5*(VAU(J,K)+ABS(VAU(J,K)))
VAUN(J,K)=0.5*(VAU(J,K)-ABS(VAU(J,K)))
END DO
END DO
UN(2:JX-1,2:KY-1)= UO(2:JX-1,2:KY-1)+T*TAUX(2:JX-1,2:KY-1)&
/HU(2:JX-1,2:KY-1)&
-GPL*(ZO(2:JX-1,2:KY-1)-ZO(1:JX-2,2:KY-1))&
+PTC*VAU(2:JX-1,2:KY-1) &
+PH1*(UO(3:JX,2:KY-1)+UO(1:JX-2,2:KY-1)-2.*UO(2:JX-1,2:KY-1))&
+PH2*(UO(2:JX-1,3:KY)+UO(2:JX-1,1:KY-2)-2.*UO(2:JX-1,2:KY-1))&
-PL*UPOS(2:JX-1,2:KY-1)*(UO(2:JX-1,2:KY-1)-UO(1:JX-2,2:KY-1))&
-PL*UNEG(2:JX-1,2:KY-1)*(UO(3:JX,2:KY-1)-UO(3:JX,2:KY-1))&
-PF*VAUP(2:JX-1,2:KY-1)*(UO(2:JX-1,2:KY-1)-UO(2:JX-1,1:KY-2))&
-PF*VAUN(2:JX-1,2:KY-1)*(UO(2:JX-1,3:KY)-UO(2:JX-1,2:KY-1))
! IMPLICIT BOTTOM FRICTION
UNI(2:JX-1,2:KY-1)=1.+PR1*SQRT(UO(2:JX-1,2:KY-1)*UO(2:JX-1,2:KY-
1)&
+VAU(2:JX-1,2:KY-1)*VAU(2:JX-1,2:KY-1))/HU(2:JX-1,2:KY-1)
UN(2:JX-1,2:KY-1)=UN(2:JX-1,2:KY-1)/UNI(2:JX-1,2:KY-1)
UN(2:JX,1:KY-1)=UN(2:JX,1:KY-1)*DUM(2:JX,1:KY-1)
!==================
! COMPUTATION OF THE V COMPONENT
UAV(2:JX-1,2:KY-2)=0.25*(UN(2:JX-1,2:KY-2)+UN(3:JX,2:KY-2)&
+ UN(2:JX-1,3:KY-1)+UN(3:JX,3:KY-1))
UAVP(2:JX-1,2:KY-2)=0.5*(UAV(2:JX-1,2:KY-2)&
+ABS(UAV(2:JX-1,2:KY-2)))
UAVN(2:JX-1,2:KY-2)=0.5*(UAV(2:JX-1,2:KY-2)&
-ABS(UAV(2:JX-1,2:KY-2)))
VPOS(2:JX-1,2:KY-2)=0.5*(VO(2:JX-1,2:KY-2)&
+ABS(VO(2:JX-1,2:KY-2)))
VNEG(2:JX-1,2:KY-2)=0.5*(VO(2:JX-1,2:KY-2)&
-ABS(VO(2:JX-1,2:KY-2)))
VN(2:JX-1,2:KY-1)=VO(2:JX-1,2:KY-1)+T*TAUY(2:JX-1,2:KY-1)/&
HV(2:JX-1,2:KY-1)&
-GPF*(ZO(2:JX-1,3:KY)-ZO(2:JX-1,2:KY-1))&
-PTC*UAV(2:JX-1,2:KY-1)&
+PH1*(VO(3:JX,2:KY-1)+VO(1:JX-2,2:KY-1)-2.*VO(2:JX-1,2:KY-1))&
+PH2*(VO(2:JX-1,3:KY)+VO(2:JX-1,1:KY-2)-2.*VO(2:JX-1,2:KY-1))&
-PF*VPOS(2:JX-1,2:KY-1)*(VO(2:JX-1,2:KY-1)-VO(2:JX-1,1:KY-2))&
```

```
-PF*VNEG(2:JX-1,2:KY-1)*(VO(2:JX-1,3:KY)-VO(2:JX-1,2:KY-1))&
-PL*UAVP(2:JX-1,2:KY-1)*(VO(2:JX-1,2:KY-1)-VO(1:JX-2,2:KY-1))&
-PL*UAVN(2:JX-1,2:KY-1)*(VO(3:JX,2:KY-1)-VO(2:JX-1,2:KY-1))&
-PR1*VO(2:JX-1,2:KY-1)*SQRT(VO(2:JX-1,2:KY-1)*VO(2:JX-1,2:KY-1)&
+UAV(2:JX-1,2:KY-1)*UAV(2:JX-1,2:KY-1))/HV(2:JX-1,2:KY-1)
VN(2:JX,1:KY-1)=VN(2:JX,1:KY-1)*DVM(2:JX,1:KY-1)
! AHHH, FINALY SEA LEVEL COMPUTATIONS
ZN(2:JX-1,2:KY-1)=ZO(2:JX-1,2:KY-1)&
-PL*UN(3:JX,2:KY-1)*HU(3:JX,2:KY-1)&
+PL*UN(2:JX-1,2:KY-1)*HU(2:JX-1,2:KY-1)&
-PF*HV(2:JX-1,2:KY-1)*VN(2:JX-1,2:KY-1)&
+PF*HV(2:JX-1,1:KY-2)*VN(2:JX-1,1:KY-2)
ZN=ZN*FSM
!UPDATE OLD VARIABLE WITH NEW ONES
UO=UN
VO=VN
ZO=ZN
IF(MOD(MO,20).EQ.0)THEN
!COMPUTE KINETIC AND POTENTIAL ENERGY
ES=0.
DO J=1,JX-1
DO K=2,KY
UA=0.5*(UN(J,K)+UN(J+1,K))
VA=0.5*(VN(J,K)+VN(J,K-1))
EN=H(J,K)*(UA*UA+VA*VA)/(2.*G)+ZN(J,K)*ZN(J,K)/2.
ES=EN+ES
END DO
END DO
WRITE(25,*)ES
PRINT *, 'MO=',MO,'WEST=',ZN(3,10)
PRINT*,'CENTER=',ZN(10,10),'EAST=',ZN(18,10)
END IF
IF(MOD(MO,8640).EQ.0)THEN
DO K=1,KY
WRITE(10,*)(ZN(J,K),J=1,JX)
END DO
END IF
END DO TIMELOOP
DO K=1,21
PRINT '(1X,21F3.0)',(ZN(J,K),J=1,21)
END DO
END PROGRAM SURGE1
```

CHAPTER III

THREE-DIMENSIONAL MODELS

Problem No. 1, vertical direction

Consider equations of motion along the vertical direction

$$-fv = -g\frac{\partial \zeta}{\partial x} + N\frac{\partial^2 u}{\partial z^2} \tag{III.1}$$

$$fu = -g\frac{\partial \zeta}{\partial y} + N\frac{\partial^2 v}{\partial z^2} \tag{III.2}$$

with wind stress at the free surface τ_x^s=1, τ_y^s=o (units are CGS), and at the bottom $u = 0, v = 0$. In this problem we assume that the sea level slopes are given. This steady state problem is focused on the vertical change of the current caused by the Coriolis force and the vertical friction. Depth is 40m. Space step along the vertical direction is variable and is denoted as $HZ(L)$. N=50 cm^2/s. Beginning of the system of coordinates is located at the free surface and z axis is directed upward.

First we introduce complex variables $w = u + iv$, $Z = g\frac{\partial \zeta}{\partial x} + ig\frac{\partial \zeta}{\partial y}$ and $\tau = \tau_x^s + i\tau_y^s$, and derive one equation for the complex variable w. For this purpose eq. (2) is multiplied by i and afterwards eqs. (1) and (2) are added on either side.

$$N\frac{\partial^2 w}{\partial z^2} - ifw = Z \tag{III.3}$$

We shall search solution for the variable w by the line inversion for the two cases: 1) the sea level slope is equal zero ($Z = 0$) and wind is defined as above; and 2) the wind stress is zero and the sea level slope along x direction is 10^{-7}, and along y direction the sea level slope is zero. For solution we shall use the line inversion method from NM as described in Appendix 1. To clarify the placement of the vertical grid points we use figures 4.1 and 4.7 from NM. Using a variable vertical grid resolution eq.(III.3) becomes

$$-N * HZ(L) * W(L-1) + (I * F * CCD + N * CD) * W(L)$$

$$-N * HZ(L-1) * W(L+1) = -Z * CCD \tag{III.4}$$

Here, $CD = HZ(L) + HZ(L-1)$; $CCD = CD * 0.5 * HZ(L) * HZ(L-1)$ and I is complex unit. Benith program ekman.f is given to solve eq.(III.4).

```
C Program ekman.f
c originated by z.k. at ims
c PROGRAM TO CALCULATE WIND DRIVEN CURRENT ALONG
C VERTICAL DIRECTION
parameter(LE=31)
REAL U(LE),V(LE)
COMPLEX W(2*LE),UN,TW,S(2*LE),E(2*LE),Z,DENOM
REAL HZ(30)/1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,
*1,1,1,1,1,1,1,1,1,1,1,1,1/
C CHANGE TO CM
hs=0
DO L=1,LE-1
HZ(L)=HZ(L)*100
hs=hs+hz(l)
END DO
C FIRST GRID POINT IS DIVIDED HALF TO AIR AND HALF TO OCEAN
C H=47.5M
C HZ-SPACE STEP IN CM,
C Q- EDDY VISCOSIYT-CONSTANT WITH THE DEPTH
C F Coriolis parameter
Q=50.
F=1.E-4
C BEFORE CALC. SET WIX-WIND ALONG X,
C WIY-WIND ALONG Y ,IN CM/S
WIX=1000.
WIY=0.
WABS=SQRT(WIX*WIX+WIY*WIY)
CDG=(.9375+WABS*.8375E-3)*1.0E-6
TWX=CDG*WIX*WABS
TWY=CDG*WIY*WABS
HCR=2.E2*CDG*WABS/F
C COMPLEX UNIT
UN=(0,1)
C F-CORIOLIS PARAMETER,UN-COMPLEX UNIT,
C Z-SEA LEVEL CHANGE
C Z=A+IB,A=-G*DZ/DX,B=-GDZ/DY
C IF YOU WISH SLOPE CURRENT ONLY SET SET TW=0
c Z=(1.E-3,0)
```

```
Z=(0,0)
TW=TWX+UN*TWY
c TW=(0.,0.)
C TW WIND STRESS TX+ITX
S(1)=Q/(Q+UN*F*HZ(1)*HZ(1))
E(1)=(Z*HZ(1)+TW)*HZ(1)/(Q+UN*F*HZ(1)*HZ(1))
DO L=2,LE-1
CD=HZ(L)+HZ(L-1)
CCD=CD*.5*HZ(L)*HZ(L-1)
DENOM=Q*CD+UN*F*CCD-Q*HZ(L)*S(L-1)
S(L)=Q*HZ(L-1)/DENOM
E(L)=(Z*CCD+Q*HZ(L)*E(L-1))/DENOM
END DO
C AT THE LOWER BOUNDARY DW/DZ=0
W(LE)=(0.,0.)
DO L=LE-1,1,-1
W(L)=S(L)*W(L+1)+E(L)
END DO
DO L=1,LE
U(L)=REAL(W(L))
V(L)=AIMAG(W(L))
END DO
PRINT *,Q,hs
OPEN(UNIT=12,FILE='EKMU.dat',STATUS='unknown')
OPEN(UNIT=13,FILE='EKMV.dat',STATUS='unknown')
WRITE(12,*)(U(L),L=1,le)
WRITE(13,*)(V(L),L=1,le)
STOP
END
```

The results of calculation for the wind-driven currents are given in Figs. III.1 and III.2. Both u and v components of velocity decrease towards the bottom. This behavior is depicted in Fig.III.1. The resulting current given in Fig.III.2 is both decreasing with depth and is rotating clockwise. The tips of the vectors describe so-called Ekman spiral.

Fig.III.1: Vertical distribution of velocity

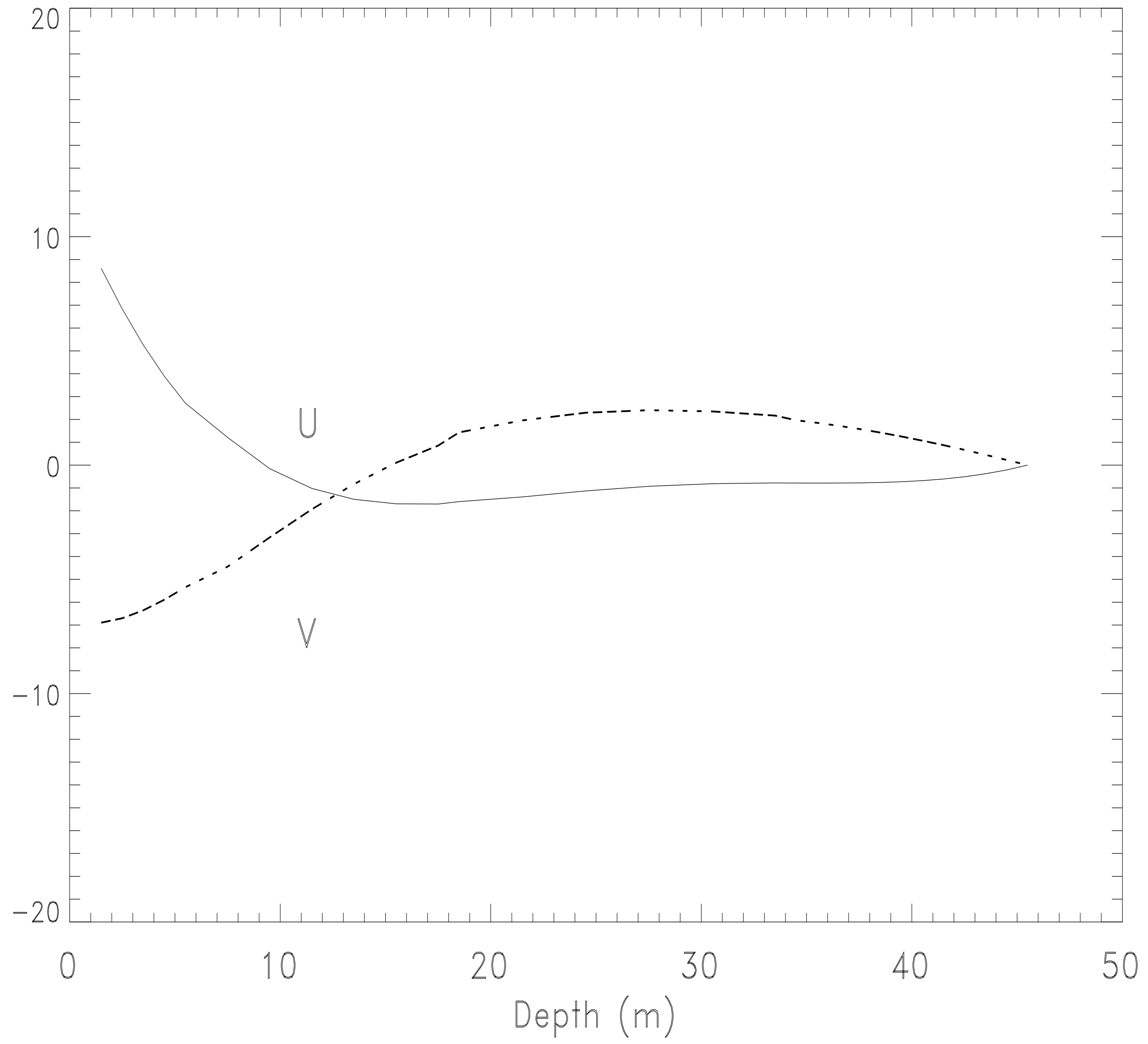
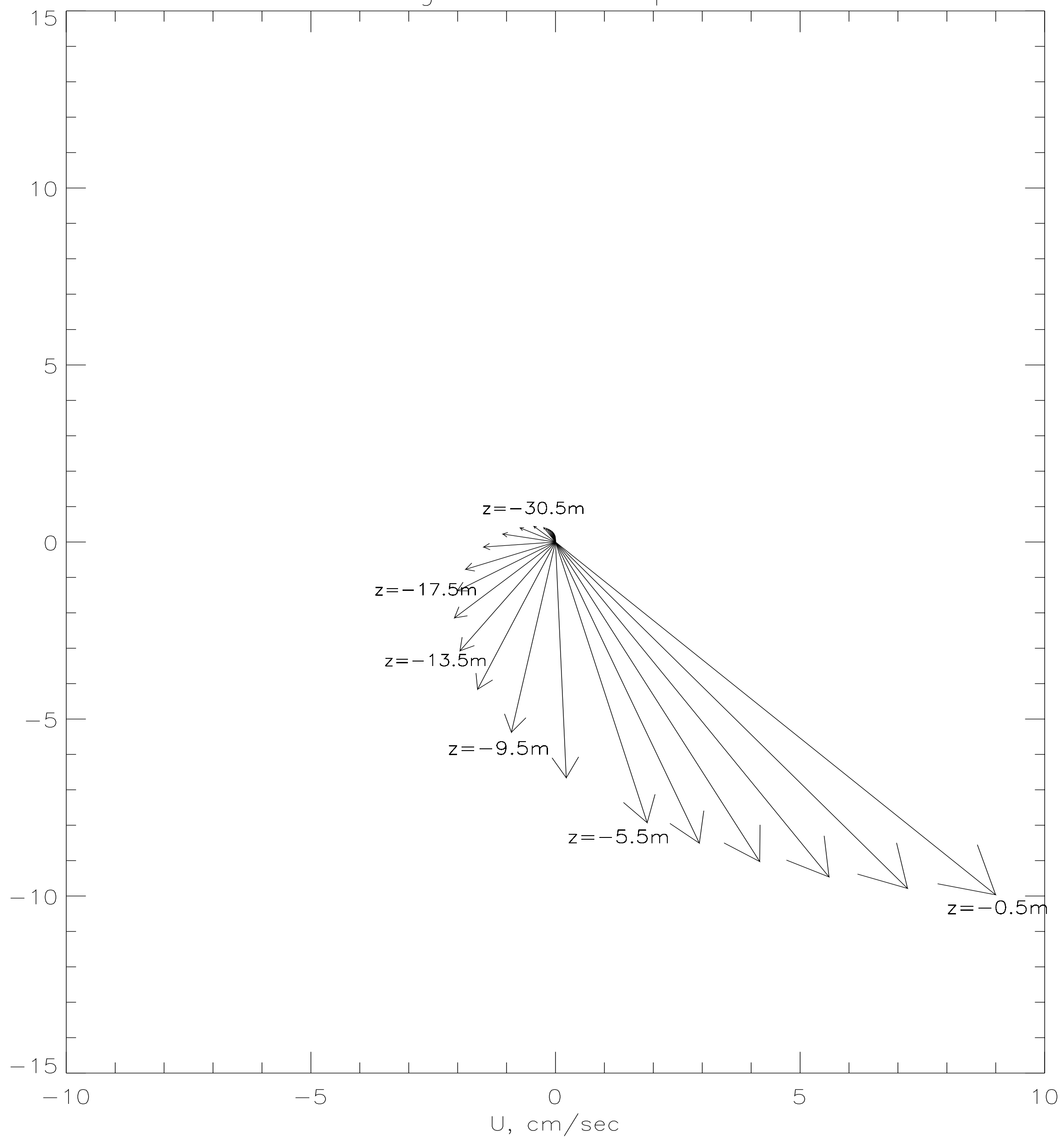


Fig.III.2: Ekman spiral



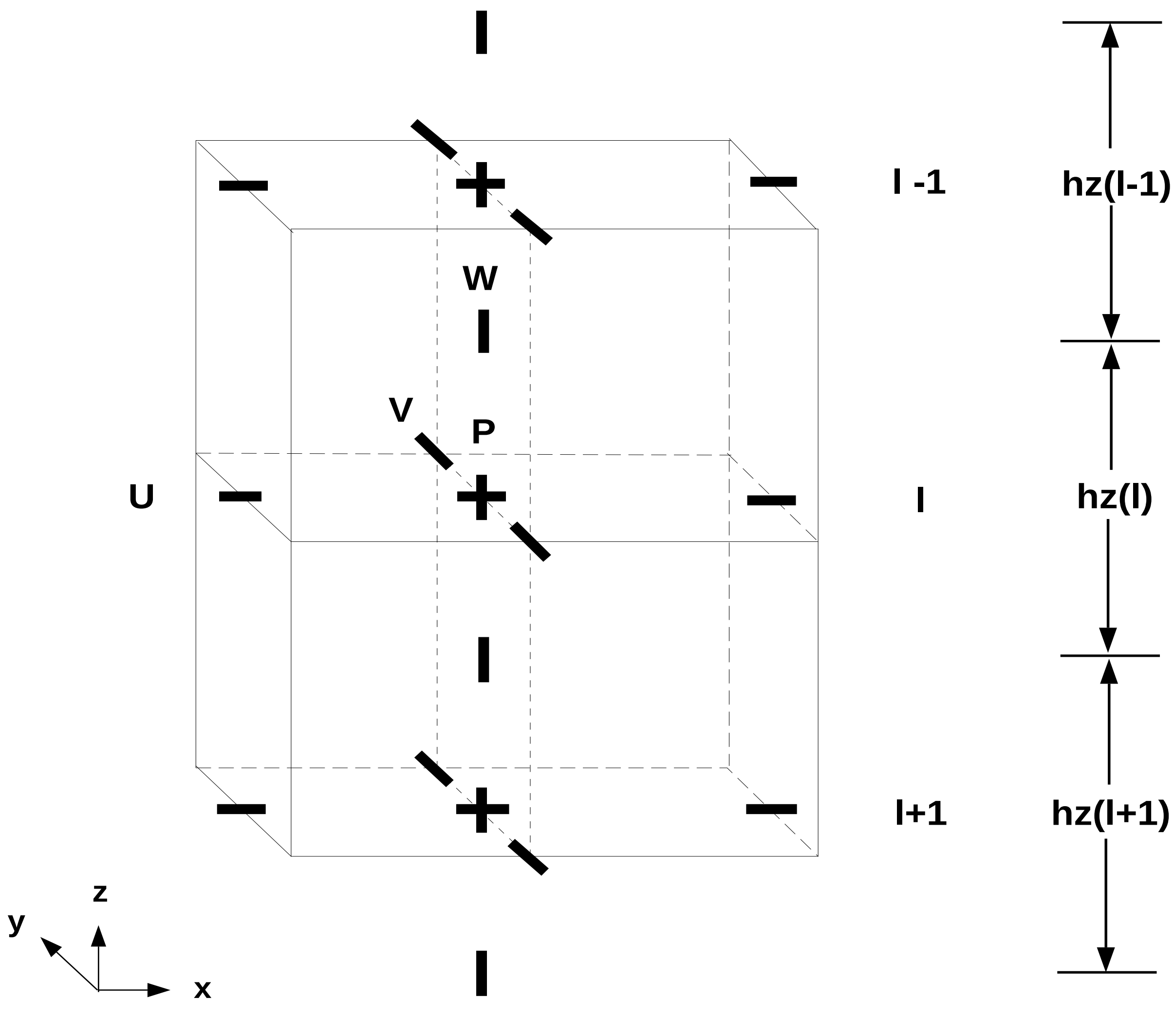


Fig.III.3: Staggered grid distribution in space.

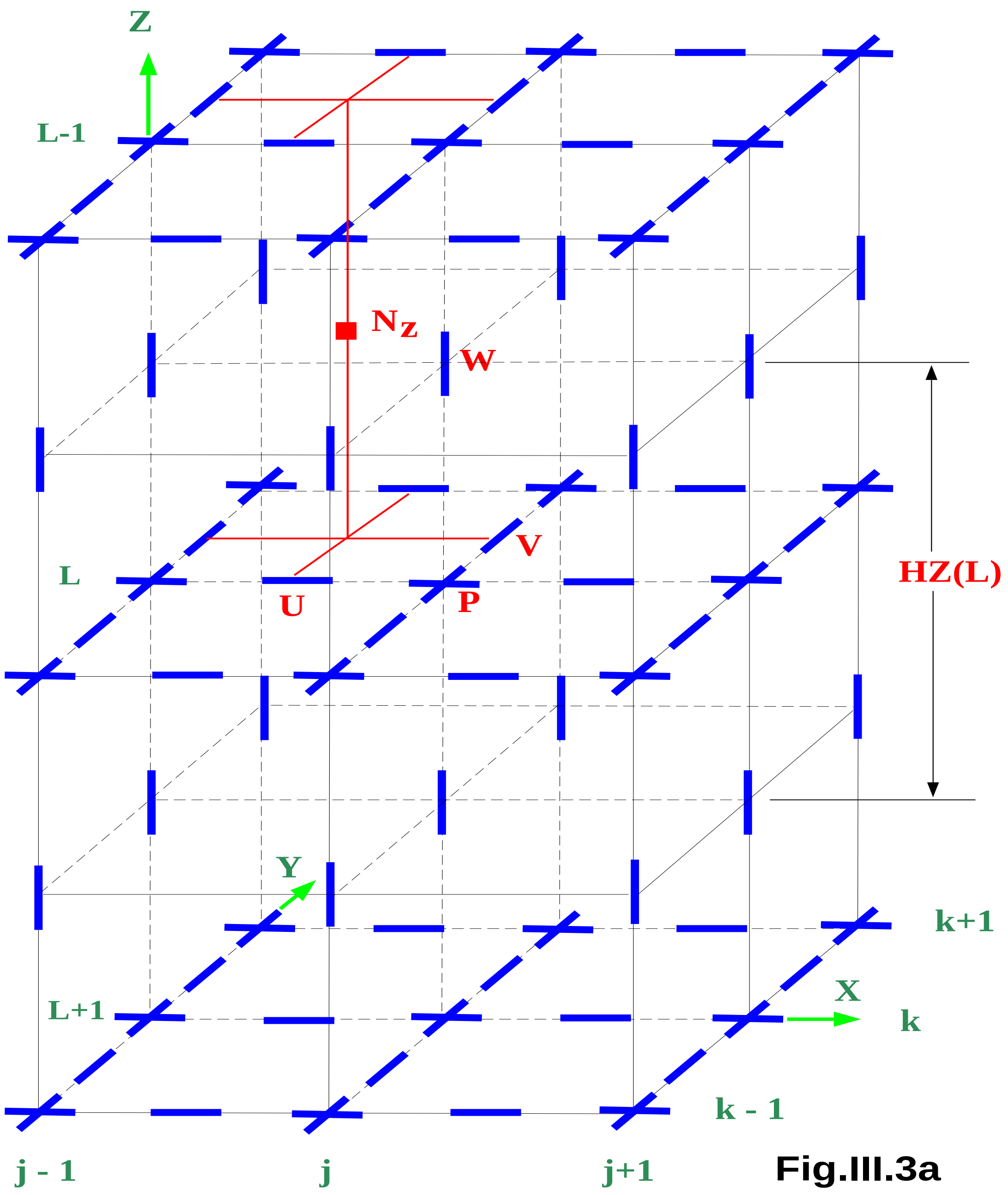


Fig.III.3a

Problem No. 2, 3D- barotropic model

To learn basics of 3D-model construction we consider barotropic motion only, i.e., density driven currents will be neglected. Two-time-level numerical schemes will be used. This approach is delineated in Ch.IV, Sec. 3.2 of NM. To show the arrangement for the time stepping we shall write first the equations of motion along x and y directions

$$\frac{u^{m+1}-u^m}{T}+(u\frac{\partial u}{\partial x})^m+(v\frac{\partial u}{\partial y})^m+(w\frac{\partial u}{\partial z})^m-fv^m$$

$$=-\frac{1}{\rho_o}\frac{\partial p_a}{\partial x}^m-g\frac{\partial \zeta}{\partial x}^m+\frac{\partial}{\partial z}N_z\frac{\partial u}{\partial z}^m+N_h\Delta u^m \tag{III.5}$$

and

$$\frac{v^{m+1}-v^m}{T}+(u\frac{\partial v}{\partial x})^m+(v\frac{\partial v}{\partial y})^m+(w\frac{\partial v}{\partial z})^m+fu^{m+1}$$

$$=-\frac{1}{\rho_o}\frac{\partial p_a}{\partial y}^m-g\frac{\partial \zeta}{\partial y}^m+\frac{\partial}{\partial z}N_z\frac{\partial v}{\partial z}^m+N_h\Delta v^m \tag{III.6}$$

Construction of the finite-difference equations will be carried out with the help of Fig.III.3 and Fig.III.3a (Fig.4.7 from NM), in which three-dimensional grid distribution is shown.

To derive the vertical velocity at any layer l, the equation of continuity is vertically integrated from $l+1$ to l,

$$\int_{l+1}^{l}(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y})dz+w^m_{j,k,l}-w^m_{j,k,l+1}=0$$

Approximating the above integral as

$$h_{z,l}(\frac{u^m_{j+1,k,l}-u^m_{j,k,l}}{hx}+\frac{v^m_{j,k,l}-v^m_{j,k-1,l}}{hy})+w^m_{j,k,l}-w^m_{j,k,l+1}=0$$

the vertical velocity at the l layer becomes

$$w^m_{j,k,l}=w^m_{j,k,l+1}-h_{z,l}(\frac{u^m_{j+1,k,l}-u^m_{j,k,l}}{hx}+\frac{v^m_{j,k,l}-v^m_{j,k-1,l}}{hy}) \tag{III.7}$$

Integration of equation of continuity over the depth yields an additional equation for the unknown sea level,

$$\frac{\partial \zeta}{\partial t}+\frac{\partial}{\partial x}\int_{-H}^{\zeta}udz+\frac{\partial}{\partial y}\int_{-H}^{\zeta}vdz=0 \tag{III.8}$$

which can be rendered for the time stepping as,

$$\frac{\zeta^{m+1}_{j,k} - \zeta^{m}_{j,k}}{T} + \frac{\partial}{\partial x}\int_{-H}^{\zeta} u^{m+1} dz + \frac{\partial}{\partial y}\int_{-H}^{\zeta} v^{m+1} dz = 0 \qquad \text{(III.9)}$$

In three-dimensional barotropic modeling a staggered grid depicted in the Figs.III.3 and Fig.III.3a (Fig 4.7 from NM), will be used. Some details of this grid can be gleaned from Figs. 4.8 and 4.1 which describe location of the grid-points in the x-z and y-z planes. These figures are very useful for understanding of the placing of variables in the staggered grid and for the construction of space derivatives. The indices of numerical integration along x, y and z directions are j, k and l respectively (both lower case and upper case letters are used in the program). The grid distance along x and y directions are constant, and are called in the program HX and HY. The grid-distance along z direction is taken as variable and is denoted HZ(L). (Notice l is the same variable as L). The vertical coordinate is a new direction and it requires very special approach. Beginning of the system of coordinates is located at the free surface with z axis pointing upward. Such geometry leads to negative depth distribution but positive HZ(L). Because we measure depth from the free surface we shall also enumerate index L starting from the free surface down. This approach will introduce only slight inconvenience because central derivative in the L grid-point will be derived as difference u(L-1)-u(L+1), and not the way we used to take difference as u(L+1)-u(L-1). To elucidate this approach one has to go back to Ch.I of this workbook and refresh the advection problems solved along the vertical direction.

Necessary stability criteria will limit time step. Along with CFL condition used previously, the condition due to the vertical friction may introduce new limitations.

$$T < \frac{h_z^2}{2N_z} \qquad \text{(III.10)}$$

If one is going to apply the fine resolution in the vertical direction using, e.g., 1m grid step, and assuming that the vertical friction coefficient N_z=500cm^2/s, the time step will be limited to $T < 10$s.

Benith a code in FORTRAN for computation of the wind driven current in a rectangular water body is specified. All necessary explanations are described in the comments lines

```
c Program simple3d.f
c originated by z.k. at ims
c This is 3d motion in rectangular water body. Depth is constant.
PARAMETER (JX=20,KY=10,LE=30)
REAL UN(JX,KY,LE),UO(JX,KY,LE)
REAL VN(JX,KY,LE),VO(JX,KY,LE)
REAL ZN(JX,KY),ZO(JX,KY),H(JX,KY),ENU(JX,KY,LE)
REAL HZ(LE),TAUSX(JX,KY),TAUBX(JX,KY),VA(JX,KY),UA(JX,KY)
REAL TSN(KY),TAUSY(JX,KY),TAUBY(JX,KY)
REAL W(JX,KY,LE+1),up(jx,ky),vp(jx,ky)
open (unit=10,file='z.dat',status='unknown')
open (unit=11,file='momaxz.dat',status='unknown')
open (unit=12,file='eni.dat',status='unknown')
open (unit=13,file='u.dat',status='unknown')
open (unit=14,file='v.dat',status='unknown')
open (unit=15,file='w.dat',status='unknown')
open (unit=16,file='ua.dat',status='unknown')
open (unit=17,file='va.dat',status='unknown')
C SET DEPTH
DO J=1,JX
DO K=1,KY
ua(j,k)=0.
va(j,k)=0.
H(J,K)=4700.
END DO
END DO
DO J=1,JX
DO K=1,KY
zo(j,k)=0.
do l=1,le
un(j,k,l)=0.
vn(j,k,l)=0.
uo(j,k,l)=0.
vo(j,k,l)=0.
end do
end do
end do
C SET variable SPACE STEP ALONG VERTICAL DIRECTION IN CM
DO L=1,5
HZ(L)=100.
END DO
DO L=6,12
HZ(L)=200.
END DO
DO L=13,17
HZ(L)=300.
END DO
DO L=18,30
```

```
HZ(L)=100.
END DO
C VARIABLE LATITUDE and table for sin tsn(k)
pi = 4.*atan(1.)
rad = pi/180.
ygrid = 1./60. ! ( 1 min : y-axis )
blat = 55.+ygrid
tsN(1) = sin(blat*rad)
templat=blat
c
do k=2,ky+1
templat = templat+ygrid
tsN(k) = sin(templat*rad)
enddo
C ALL COEFFICIENTS
c========================================
C SET WIND FOR INITIAL EXPERIMENT
DO J=1,JX
DO K=1,KY
TAUSX(J,K)=1.
TAUSY(J,K)=0.
END DO
END DO
c Time step
T = 10.
c HORIZONTAL EDDY VISCOSITY
AH=1.0e7
G=981.
c SPACE STEPS ALONG X AND Y
HY=13.89E5
HX=13.89E5
FC=1.458E-4
RF1=2.6E-3
C COMBINATIONAL PARAMETERS
TRF=T*RF1
PL=T/HX
GPL=G*PL
PLL=AH*T/(HX*HX)
PF=T/HY
GPF=G*PF
PFF=AH*T/(HY*HY)
TFC=T*FC
```

```
C START TIME LOOP INDEX N RUNS UNTIL N=MON
c******************************
MON=86400
C8888888888888888888888888888888
50 N=N+1
C SET WIND FOR INITIAL EXPERIMENT
DO J=1,JX
DO K=1,KY
TAUSX(J,K)=1.
TAUSY(J,K)=0.
END DO
END DO
c if (n.lt.1000)then
c DO J=1,JX
c DO K=1,KY
c TAUSX(J,K)=float(n)/1000.
c TAUSY(J,K)=0.
c END DO
c END DO
c end if
C EDDY VISCOSITY IS CONSTANT
DO 33J=1,JX
DO 33K=1,KY
DO 33 L=1,Le
ENU(J,K,L)=50.
33 CONTINUE
C COMPUTATION OF U VELOCITY
DO 4 J=3,JX-2
DO 4 K=2,KY-1
C SURFACE CONDITION
UAV=0.25*(VO(J-1,K,1)+VO(J-1,K-1,1)+VO(J,K-1,1)+VO(J,K,1))
DSL=ZO(J,K)-ZO(J-1,K)
DSLX=GPL*DSL
HZP=0.5*(HZ(1)+HZ(2))
DSU=(UO(J,K,1)-UO(J,K,2))/HZP
VFRS=(TAUSX(J,K)-ENU(J,K,2)*DSU)/HZ(1)
HFR=PLL*(UO(J+1,K,1)+UO(J-1,K,1)-2.*UO(J,K,1))+
2PFF*(UO(J,K+1,1)+UO(J,K-1,1)-2.*UO(J,K,1))
C NONLINEAR TERMS
Q1=0.5*(UAV+ABS(UAV))
Q2=0.5*(UAV-ABS(UAV))
ONHY1=PF*Q1*(UO(J,K,1)-UO(J,K-1,1))
```

```
ONHY2=PF*Q2*(UO(J,K+1,1)-UO(J,K,1))
Q3=0.5*(UO(J,K,1)+ABS(UO(J,K,1)))
Q4=0.5*(UO(J,K,1)-ABS(UO(J,K,1)))
ONH1=PL*Q3*(UO(J,K,1)-UO(J-1,K,1))
ONH2=PL*Q4*(UO(J+1,K,1)-UO(J,K,1))
SUMNONX=ONH1+ONH2+ONHY1+ONHY2
UN(J,K,1)=UO(J,K,1)+TFC*UAV*TSN(K)-DSLX+VFRS*T+HFR-
2SUMNONX
LMM=LE-1
DO 5 L=2,LMM-1
UAV=0.25*(VO(J-1,K,L)+VO(J-1,K-1,L)+VO(J,K-1,L)+VO(J,K,L))
HZP=0.5*(HZ(L)+HZ(L+1))
HZN=0.5*(HZ(L)+HZ(L-1))
DSUN=(UO(J,K,L-1)-UO(J,K,L))/HZN
DSUP=(UO(J,K,L)-UO(J,K,L+1))/HZP
VFR=(ENU(J,K,L)*DSUN-ENU(J,K,L+1)*DSUP)/HZ(L)
HFR=PLL*(UO(J+1,K,L)+UO(J-1,K,L)-2.*UO(J,K,L))+
2PFF*(UO(J,K+1,L)+UO(J,K-1,L)-2.*UO(J,K,L))
C NONLINEAR TERMS
Q1=0.5*(UAV+ABS(UAV))
Q2=0.5*(UAV-ABS(UAV))
ONHY1=PF*Q1*(UO(J,K,L)-UO(J,K-1,L))
ONHY2=PF*Q2*(UO(J,K+1,L)-UO(J,K,L))
Q3=0.5*(UO(J,K,L)+ABS(UO(J,K,L)))
Q4=0.5*(UO(J,K,L)-ABS(UO(J,K,L)))
ONH1=PL*Q3*(UO(J,K,L)-UO(J-1,K,L))
ONH2=PL*Q4*(UO(J+1,K,L)-UO(J,K,L))
SUMNONX=ONH1+ONH2+ONHY1+ONHY2
UN(J,K,L)=UO(J,K,L)+TFC*UAV*TSN(K)-DSLX+VFR*T+HFR-
2SUMNONX
5 CONTINUE
C BOTTOM CONDITION GIVEN BOTTOM STRESS
UAV=0.25*(VO(J-1,K,LMM)+VO(J-1,K-1,LMM)
2+VO(J,K-1,LMM)+VO(J,K,LMM))
HZP=HZ(LMM)
HZN=0.5*(HZ(LMM)+HZ(LMM-1))
DSUP=2.*UO(J,K,LMM)/HZP
DSUN=(UO(J,K,LMM-1)-UO(J,K,LMM))/HZN
c VFR=(ENU(J,K,LMM)*DSVN-ENU(J,K,LMM+1)*DSUP)/HZ(LMM)
PIERW=SQRT(UAV*UAV+UO(J,K,LMM)*UO(J,K,LMM))
TAUBX(J,K)=TRF*PIERW*UO(J,K,LMM)
VFR=(ENU(J,K,LMM)*DSUN-TAUBX(J,K))/HZ(LMM)
```

```
HFR=PLL*(UO(J+1,K,LMM)+UO(J-1,K,LMM)-2.*UO(J,K,LMM))+
2PFF*(UO(J,K+1,LMM)+UO(J,K-1,LMM)-2.*UO(J,K,LMM))
C NONLINEAR TERMS
Q1=0.5*(UAV+ABS(UAV))
Q2=0.5*(UAV-ABS(UAV))
ONHY1=PF*Q1*(UO(J,K,LMM)-UO(J,K-1,LMM))
ONHY2=PF*Q2*(UO(J,K+1,LMM)-UO(J,K,LMM))
Q3=0.5*(UO(J,K,LMM)+ABS(UO(J,K,LMM)))
Q4=0.5*(UO(J,K,LMM)-ABS(UO(J,K,LMM)))
ONH1=PL*Q3*(UO(J,K,LMM)-UO(J-1,K,LMM))
ONH2=PL*Q4*(UO(J+1,K,LMM)-UO(J,K,LMM))
SUMNONX=ONH1+ONH2+ONHY1+ONHY2
UN(J,K,LMM)=UO(J,K,LMM)+TFC*UAV*TSN(K)-DSLX+VFR*T+HFR-
2SUMNONX
UN(J,K,LMM)=0.
4 CONTINUE
C COMPUTATION OF V VELOCITY
DO 6 J=2,JX-1
DO 6 K=3,KY-2
C SURFACE CONDITION
VAUN=0.25*(UN(J,K,1)+UN(J,K+1,1)+UN(J+1,K,1)+UN(J+1,K+1,1))
DSL=ZO(J,K+1)-ZO(J,K)
DSLY=GPF*DSL
HZP=0.5*(HZ(1)+HZ(2))
DSV=(VO(J,K,1)-VO(J,K,2))/HZP
VFRS=(TAUSY(J,K)-ENU(J,K,2)*DSV)/HZ(1)
HFR=PLL*(VO(J+1,K,1)+VO(J-1,K,1)-2.*VO(J,K,1))+
2PFF*(VO(J,K+1,1)+VO(J,K-1,1)-2.*VO(J,K,1))
C COMPUTING NONLINEAR TERMS
VAU=0.25*(UO(J,K,1)+UO(J,K+1,1)+UO(J+1,K,1)+UO(J+1,K+1,1))
Q1=0.5*(VAU+ABS(VAU))
Q2=0.5*(VAU-ABS(VAU))
ONHX1=PL*Q1*(VO(J,K,1)-VO(J-1,K,1))
ONHX2=PL*Q2*(VO(J+1,K,1)-VO(J,K,1))
Q3=0.5*(VO(J,K,1)+ABS(VO(J,K,1)))
Q4=0.5*(VO(J,K,1)-ABS(VO(J,K,1)))
ONHY1=PF*Q3*(VO(J,K,1)-VO(J,K-1,1))
ONHY2=PF*Q4*(VO(J,K+1,1)-VO(J,K,1))
SUMNONY=ONHX1+ONHX2+ONHY1+ONHY2
VN(J,K,1)=VO(J,K,1)-TFC*VAUN*TSN(K)-DSLY+VFRS*T+HFR-
2SUMNONY
LMM=LE-1
```

```
DO 7 L=2,LMM-1
VAUN=0.25*(UN(J,K,L)+UN(J,K+1,L)+UN(J+1,K,L)+UN(J+1,K+1,L))
HZP=0.5*(HZ(L)+HZ(L+1))
HZN=0.5*(HZ(L)+HZ(L-1))
DSVN=(VO(J,K,L-1)-VO(J,K,L))/HZN
DSVP=(VO(J,K,L)-VO(J,K,L+1))/HZP
VFR=(ENU(J,K,L)*DSVN-ENU(J,K,L+1)*DSVP)/HZ(L)
HFR=PLL*(VO(J+1,K,L)+VO(J-1,K,L)-2.*VO(J,K,L))+
2PFF*(VO(J,K+1,L)+VO(J,K-1,L)-2.*VO(J,K,L))
C COMPUTING NONLINEAR TERMS
VAU=0.25*(UO(J,K,L)+UO(J,K+1,L)+UO(J+1,K,L)+UO(J+1,K+1,L))
Q1=0.5*(VAU+ABS(VAU))
Q2=0.5*(VAU-ABS(VAU))
ONHX1=PL*Q1*(VO(J,K,L)-VO(J-1,K,L))
ONHX2=PL*Q2*(VO(J+1,K,L)-VO(J,K,L))
Q3=0.5*(VO(J,K,L)+ABS(VO(J,K,L)))
Q4=0.5*(VO(J,K,L)-ABS(VO(J,K,L)))
ONHY1=PF*Q3*(VO(J,K,L)-VO(J,K-1,L))
ONHY2=PF*Q4*(VO(J,K+1,L)-VO(J,K,L))
SUMNONY=ONHX1+ONHX2+ONHY1+ONHY2
VN(J,K,L)=VO(J,K,L)-TFC*VAUN*TSN(K)-DSLY+VFR*T+HFR-
2SUMNONY
7 CONTINUE
C BOTTOM CONDITION GIVEN BOTTOM STRESS OR VELOCITY
C GOES TO ZERO
VAUN=0.25*(UN(J,K,LMM)+UN(J,K+1,LMM)
2+UN(J+1,K,LMM)+UN(J+1,K+1,LMM))
HZP=HZ(LMM)
HZN=0.5*(HZ(LMM)+HZ(LMM-1))
DSVN=(VO(J,K,LMM-1)-VO(J,K,LMM))/HZN
DSVP=2.*VO(J,K,LMM)/HZP
PIERW=SQRT(VAUN*VAUN+VO(J,K,LMM)*VO(J,K,LMM))
TAUBY(J,K)=TRF*PIERW*VO(J,K,LMM)
VFR=(ENU(J,K,LMM)*DSVN-TAUBY(J,K))/HZ(LMM)
C VFR=(ENU(J,K,LMM)*DSVN-ENU(J,K,LMM+1)*DSVP)/HZ(LMM)
HFR=PLL*(VO(J+1,K,LMM)+VO(J-1,K,LMM)-2.*VO(J,K,LMM))+
2PFF*(VO(J,K+1,LMM)+VO(J,K-1,LMM)-2.*VO(J,K,LMM))
C COMPUTING NONLINEAR TERMS
VAU=0.25*(UO(J,K,LMM)+UO(J,K+1,LMM)+UO(J+1,K,LMM)
2+UO(J+1,K+1,LMM))
Q1=0.5*(VAU+ABS(VAU))
Q2=0.5*(VAU-ABS(VAU))
```

```
ONHX1=PL*Q1*(VO(J,K,LMM)-VO(J-1,K,LMM))
ONHX2=PL*Q2*(VO(J+1,K,LMM)-VO(J,K,LMM))
Q3=0.5*(VO(J,K,LMM)+ABS(VO(J,K,LMM)))
Q4=0.5*(VO(J,K,LMM)-ABS(VO(J,K,LMM)))
ONHY1=PF*Q3*(VO(J,K,LMM)-VO(J,K-1,LMM))
ONHY2=PF*Q4*(VO(J,K+1,LMM)-VO(J,K,LMM))
SUMNONY=ONHX1+ONHX2+ONHY1+ONHY2
VN(J,K,LMM)=VO(J,K,LMM)-TFC*VAUN*TSN(K)-DSLY+VFR*T+HFR-
2SUMNONY
VN(J,K,LMM)=0.
6 CONTINUE
C SEA LEVEL
C FIRST AVERAGE VELOCITY ALONG VERTICAL DIRECTION
c print *, un(2,5,1),un(3,5,1)
DO 9 J=3,JX-2
DO 9 K=2,KY-1
MM=LE
C++++++++++++++++++++++++++++++++++++++++++
HZ(1)=100.+(ZO(J,K)+ZO(J-1,K))*0.5
UA(J,K)=0.
HZS=0.
DO 10 L=MM,1,-1
UA(J,K)=UA(J,K)+UN(J,K,L)*HZ(L)
HZS=HZS+HZ(L)
10 CONTINUE
c if (ua(j,k).gt.11) print *, j,k
c IN CASE YOU NEED AVERAGE VELOCITY
c UAR(J,K)=UA(J,K)/HZS
9 CONTINUE
c*************************************
c print *, UA(2,5),UA(3,5)
DO 11 J=2,JX-1
DO 11 K=3,KY-2
C+++++++++++++++++++++++++++++++++++++++++
HZ(1)=100.+(ZO(J,K)+ZO(J,K+1))*0.5
MM=LE
VA(J,K)=0.
HZS=0.
DO 12 L=MM,1,-1
VA(J,K)=VA(J,K)+VN(J,K,L)*HZ(L)
HZS=HZS+HZ(L)
12 CONTINUE
```

```
c VAR(J,K)=VA(J,K)/HZS
11 CONTINUE
C NOW USE CONTINUITY EQUATION
c LZ=0
DO 13 J=2,JX-1
DO 13 K=2,KY-1
c93 CONTINUE
ZN(J,K)=ZO(J,K)-PL*(UA(J+1,K)-UA(J,K))-PF*(VA(J,K)-VA(J,K-1))
c LZ=LZ+1
c IF(LZ.EQ.2)THEN
c LZ=0
c ZS(J,K)=ZO(J,K)
c GO TO 13
c END IF
c ZO(J,K)=0.25*ZN(J,K)+0.5*ZO(J,K)+0.25*ZS(J,K)
c GO TO 93
13 CONTINUE
c vertical velocity from continuity equation
HZ(1)=100+ZO(J,K)
DO 25 J=2,JX-1
DO 25 K=2,KY-1
MLEZ=LE
W(J,K,MLEZ+1)=0.
DO 26 L=1,MLEZ
HZX=HZ(L)/HX
HZY=HZ(L)/HY
DIFU=UN(J+1,K,L)-UN(J,K,L)
DIFV=VN(J,K,L)-VN(J,K-1,L)
W(J,K,L)=W(J,K,L+1)-HZX*DIFU-HZY*DIFV
26 CONTINUE
25 CONTINUE
C CHANGE VARIABLES
DO 15 J=1,JX
DO 15 K=1,KY
DO 15L=1,LE
UO(J,K,L)=UN(J,K,L)
VO(J,K,L)=VN(J,K,L)
15 CONTINUE
do J=1,JX
DO K=1,KY
zo(j,k)=zn(j,k)
end do
end do
c print *, ua(10,10),ua(19,10)
c print *, un(10,5,1),un(10,5,2),un(10,5,3)
c print *, vn(10,5,1),vn(10,5,2),vn(10,5,3)
c print *, zn(3,5),zn(10,5),zn(18,5)
c print *, 'zn(3,5)','ua(4,5)','ua(3,5)','zo(3,5)'
c print *,'va(3,5)','va(3,4)'
if(mod(N,200).eq.0) then
C THIS FOR THE VERTICALLY AVERAGE FLOW
c Computing kinetic and potential energy every 200 steps
c ENI enrgy every 180 steps , averaged over whole space
NC=N/200
ES=0.
do j=1,JX-1
do k=2,KY
uaE=0.5*(ua(j,k)+ua(j+1,k))
vaE=0.5*(va(j,k)+va(j,k-1))
en=(uaE*uaE+vaE*vaE)/(2.*g*h(j,k))+zn(j,k)*zn(j,k)/2.
es=en+es
end do
end do
c eni(nc)=es
write(12,*)es
write(11,*),n,zn(2,5),zn(10,5),zn(18,5)
print *,n,'zn(2,5)=',zn(2,5),'zn(10,5)=',zn(10,5)
print *,'zn(18,5)=',zn(18,5)
end if
IF(N.LT.MON)GO TO 50
c WRITE SEA LEVEL AND VELOCITIES
WRITE(10,*)((ZN(J,K),J=1,JX),K=1,KY)
Do j=1,jx
do k=1,ky
up(j,k)=0.5*(un(J,K,1)+un(j+1,k,1))
vp(j,k)=0.5*(vn(j,k-1,1)+vn(J,K,1))
end do
end do
WRITE(16,*)((up(j,k),J=1,JX),K=1,KY)
WRITE(17,*)((vp(j,k),J=1,JX),K=1,KY)
DO L=1,LE
uw=(Un(9,5,L)+Un(10,5,L))*0.5
WRITE(13,*)(uw)
END DO
```

```
DO L=1,LE
vw=(Vn(9,5,L)+Vn(9,4,L))*0.5
WRITE(14,*)vw
END DO
c DO L=1,LE
c WRITE(15,*)((w(J,K,L),J=1,JX),K=1,KY)
c END DO
stop
end
```

Some results of computations are given in figs. III.4, III.5 and III.6. Fig.III.4 depicts temporal changes of the sea level at the three locations along x directions. Because wind is directed along x axis the sea level will increase along the same axis. Periodical oscillations of the sea level are defined by own oscillations of the water body, or seiche. After about 4 day of the process a steady state occurs. In Fig.III.5 a vertical distribution of the horizontal current is depicted. The point for this distribution is located at the center of the rectangular water body. Comparing this distribution with distribution from Fig.III.1 one can see both similar and dissimilar behavior. This is caused by the presence of the slope current in the latter computations. In fig.III.6 calculated current at the surface is given after about four days of process.

Fig.III.4: Sea Level

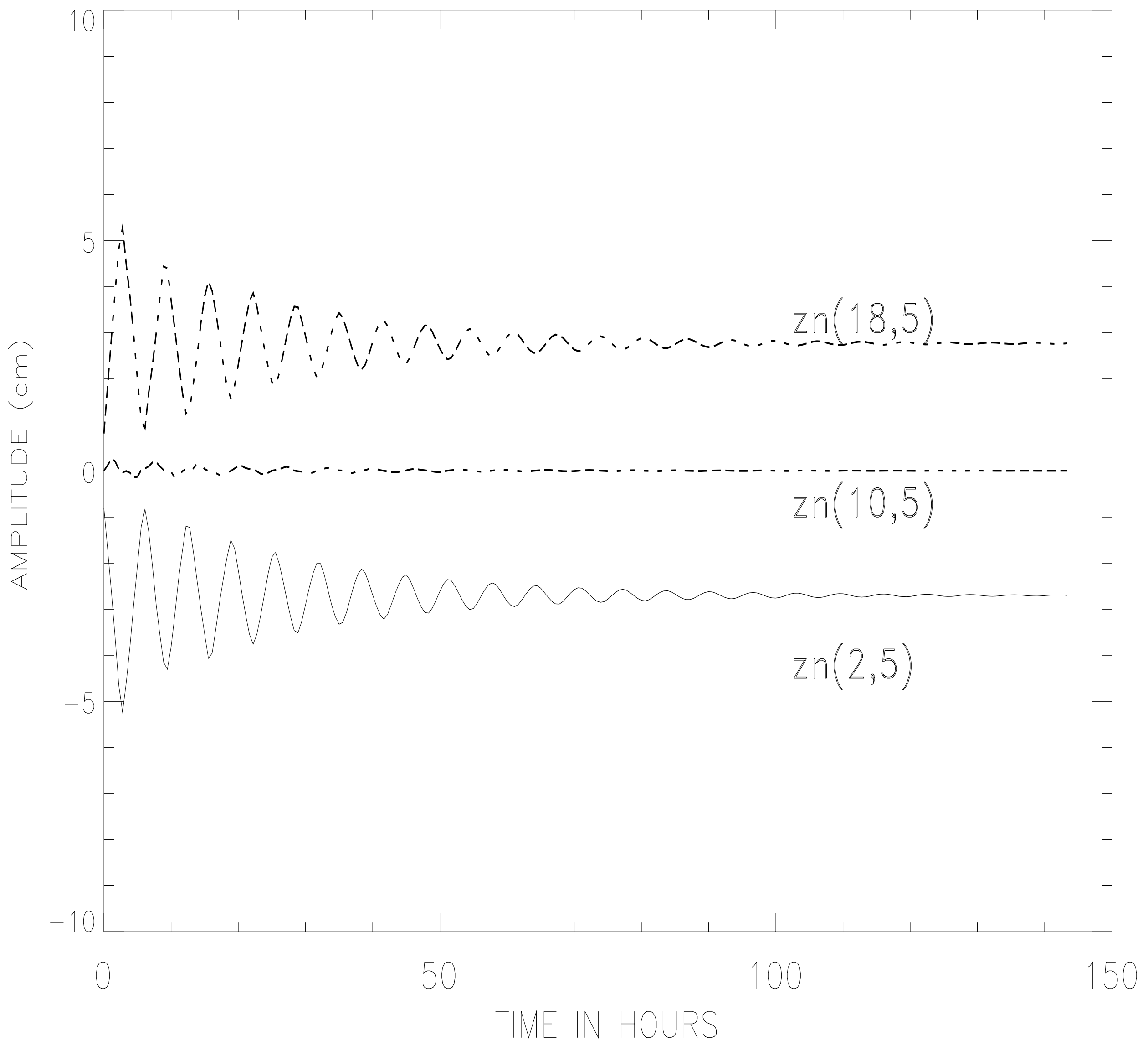


Fig.III.5: Vertical distribution of velocity

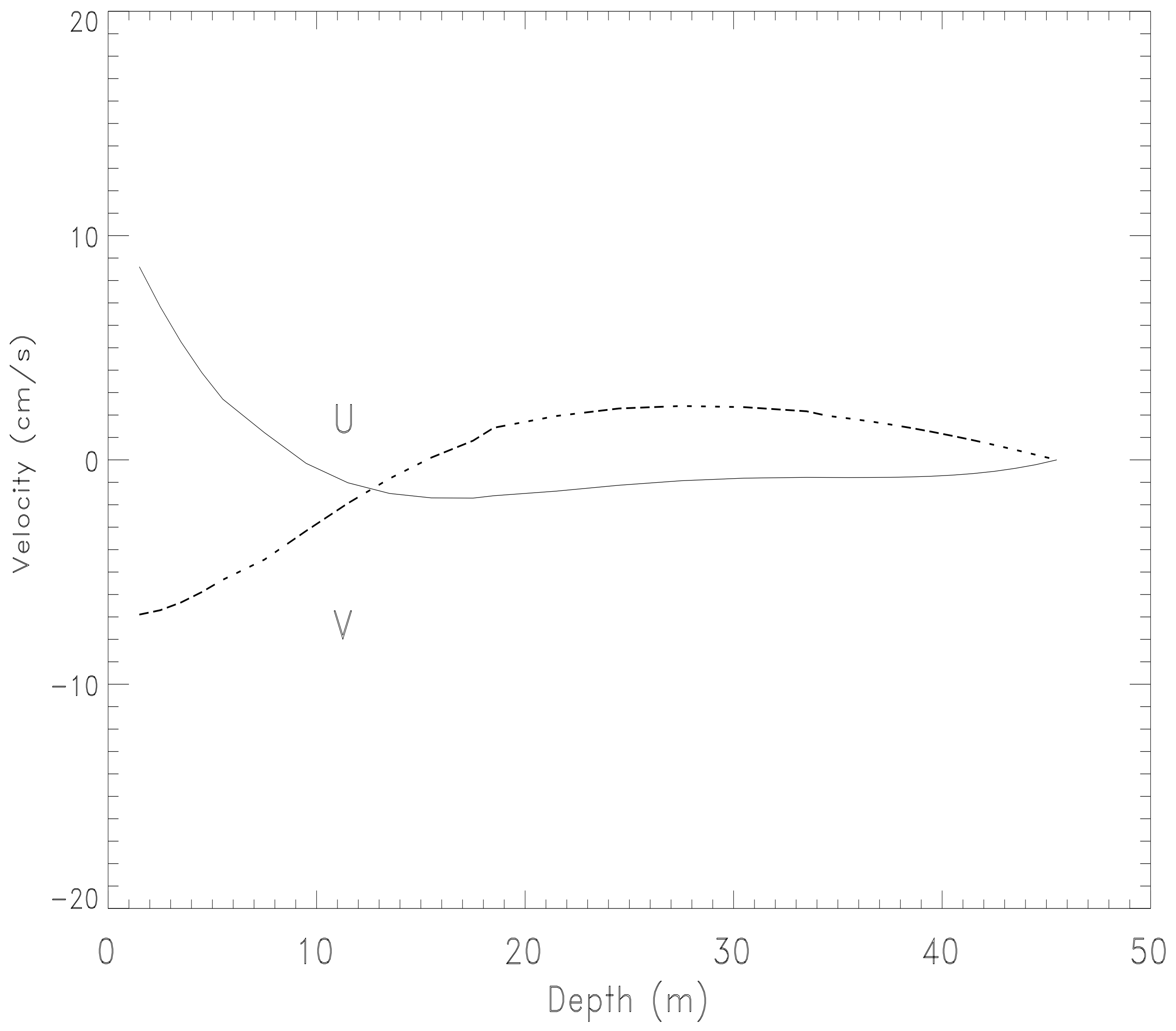
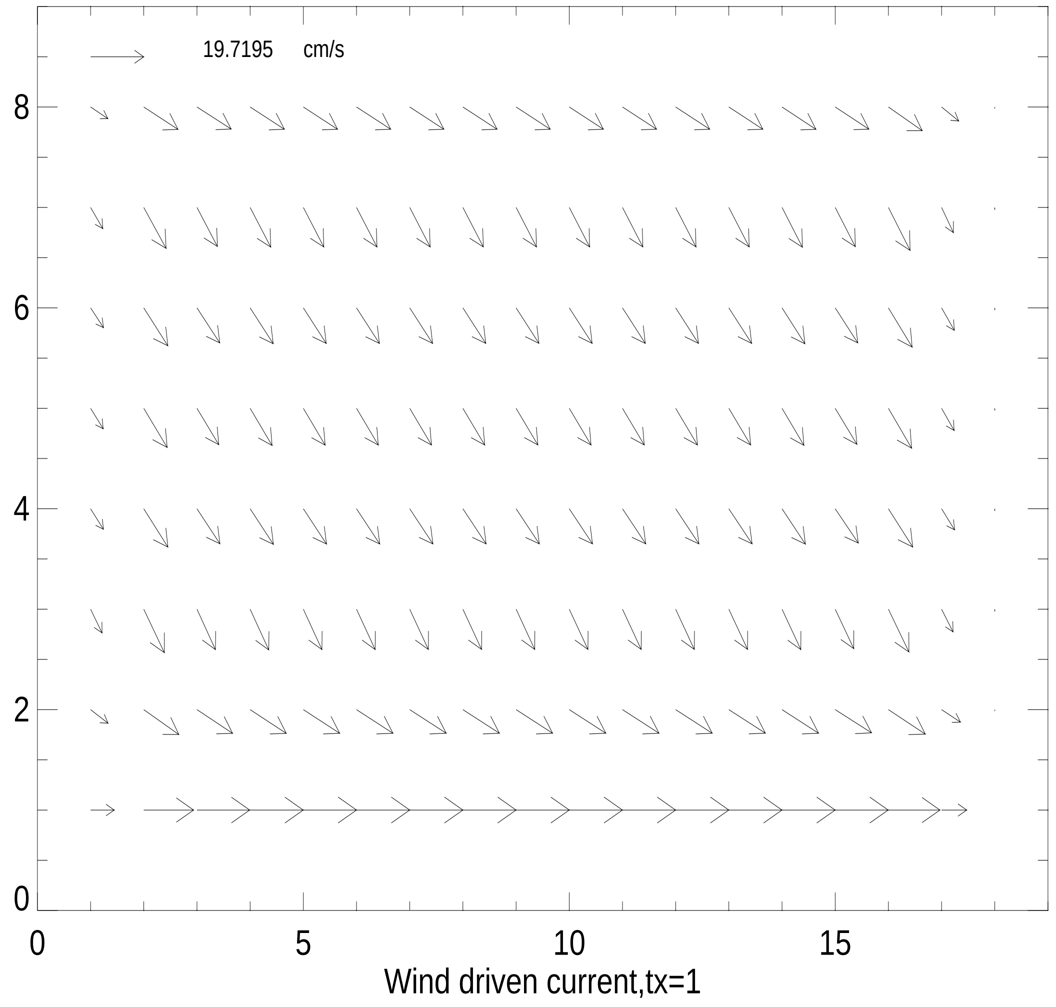


Fig.III.6 : Rectangular water body



Problem No. 3, 3D- barotropic model, sigma coordinate

Introduction

The modeling of the oceanic flow in the so-called sigma coordinate requires a change from the regular, everyday prospective into the one with the vertical coordinate stretched thus vertical distance is expressed in relation to the total depth. This approach is similar to application of percentage. The total depth instead of 100% is equal to 1. This approach, presumably simple, requires a lot of changing in the equations and in the programming. Let assume the total depth is 100 m and that the measurements of temperature were done at the irregular depth intervals. These data can not be taken as an initial data for the sigma model, they must be transferred into sigma coordinate by approximating values from the regular coordinate system, and vice versa, the model results obtained in the sigma coordinate are very difficult to analyze and to understand, therefore, one must plot data in regular z coordinate system. The sigma coordinate, or so-called stretching coordinate defines the new vertical coordinate as

$$\sigma = \frac{z-\zeta}{H+\zeta} \tag{III.11}$$

The total depth $D = H + \zeta$. The new coordinate transforms entire water column into a unit depth. The depth in sigma coordinate is changing from 0, at the free surface to -1 at the bottom. (Fig. 4.10 from (NM) explains applications of the sigma coordinate and its relation to z coordinate system). Relations between old and new coordinates and derivatives in the new system of coordinate are given in NM Ch. 4, section 4.3. A set of equations of motion and continuity in the new independent variables (x, y, σ, t) is given below:

$$\frac{\partial}{\partial t}(uD) + \frac{\partial}{\partial x}(u^2 D) + \frac{\partial}{\partial y}(uvD) + \frac{\partial}{\partial \sigma}(Wu) - fDv$$

$$= -gD\frac{\partial \zeta}{\partial x} + \frac{1}{D}\frac{\partial}{\partial \sigma}(N_z\frac{\partial u}{\partial \sigma}) + +N_h[\frac{\partial}{\partial x}D\frac{\partial u}{\partial x} + \frac{\partial}{\partial y}D\frac{\partial u}{\partial y}] \tag{III.12}$$

$$\frac{\partial}{\partial t}(vD) + \frac{\partial}{\partial x}(uvD) + \frac{\partial}{\partial y}(v^2 D) + \frac{\partial}{\partial \sigma}(Wv) + fDu$$

$$= -gD\frac{\partial \zeta}{\partial y} + \frac{1}{D}\frac{\partial}{\partial \sigma}(N_z\frac{\partial v}{\partial \sigma}) + N_h[\frac{\partial}{\partial x}D\frac{\partial v}{\partial x} + \frac{\partial}{\partial y}D\frac{\partial v}{\partial y}] \tag{III.13}$$

$$\frac{\partial \zeta}{\partial t} + \frac{\partial}{\partial x}(uD) + \frac{\partial}{\partial y}(vD) + \frac{\partial}{\partial \sigma}(W) = 0 \tag{III.14}$$

In the above equations W is the vertical velocity in the new system of coordinate. It is defined by eq. (4.178) from NM. The relation between the old vertical velocity w and the new one is defined as

$$W = w - (\sigma + 1)\frac{\partial \zeta}{\partial t} - uQ_x - vQ_y \tag{III.15}$$

Here

$$W = D\frac{D\sigma}{Dt}, \quad Q_x = \sigma\frac{\partial D}{\partial x} + \frac{\partial \zeta}{\partial x}, \quad Q_y = \sigma\frac{\partial D}{\partial y} + \frac{\partial \zeta}{\partial y} \tag{III.16}$$

Boundary conditions for the vertical velocity at the free surface and at the bottom (eqs. 1.64 and 1.65 a from NM) change in the new system of coordinate to:

$$W(x, y, 0, t) = 0 \qquad W(x, y, -1, t) = 0 \tag{III.17}$$

Boundary conditions at the surface and at the bottom are defined by horizontal stresses

$$\tau_x^s = \frac{\rho N_z}{D}\frac{\partial u}{\partial \sigma} \qquad \tau_y^s = \frac{\rho N_z}{D}\frac{\partial v}{\partial \sigma} \quad at \quad \sigma = 0 \tag{III.17a}$$

$$\tau_x^b = \frac{\rho N_z}{D}\frac{\partial u}{\partial \sigma} \qquad \tau_y^b = \frac{\rho N_z}{D}\frac{\partial v}{\partial \sigma} \quad at \quad \sigma = -1 \tag{III.17b}$$

Integrating eq.(III.14) with respect to σ and applying the boundary condition expressed by eq.(III.17) we arrive at the vertically integrated continuity equation (eq.4.182b from NM),

$$\frac{\partial}{\partial x}(\bar{u}D) + \frac{\partial}{\partial y}(\bar{v}D) + \frac{\partial \zeta}{\partial t} = 0 \tag{III.18}$$

The components of the depth-averaged velocity in the above equation are expressed as

$$\bar{u} = \int_{-1}^{0} u d\sigma \qquad \bar{v} = \int_{-1}^{0} v d\sigma \tag{III.19}$$

How to write code in the nonrectangular domain

Let us consider water body that has circular shape in the horizontal plain. Its bathymetry is defined by parabolic function. Rectangular system of coordinate changes along x direction from x=0 to x=200 km and in similar fashion along the y direction. Space step of numerical integration is 10km. General equation for the depth reads

$$z = -a(x^2 + y^2) + c \tag{III.20}$$

Ensuing FORTRAN program prescribes depth according to the above equation. Depth is given in the positive numbers (cm).

```
C Program depthdat.f
C PROGRAM TO PRESCRIBE DEPTH IN CIRCULAR BASIN
```

```
PARAMETER (JX=21,KY=21)
REAL H(JX,KY)
open (unit=1,file='parab.dat',status='unknown')
DO J=1,JX
DO K=1,KY
HXD=((Float(j)-1.)*0.1-1.)*((Float(j)-1.)*0.1-1.)
HYD=((Float(K)-1.)*0.1-1.)*((Float(K)-1.)*0.1-1.)
c the maximum depth is 13000 cm
H(j,K)=-4500.*3.*(HXD+HYD)+13000.
c at any location where depth is less or equal 500cm the depth is
c set to zero. So that inside
c computational domain only depth greater than 500cm will occur.
if(h(j,k).le.500.)h(j,k)=0.
END DO
END DO
write(1,*)((h(j,k),j=1,jx),k=1,ky)
stop
end
```

To describe 3-D time dependent motion we shall solve eq. III.12, III.13, III.14 and III.18. Numerical solution will proceed separately for the u, v, W and for the sea level ζ. These equations are to be solved in the staggered grid shown in Fig.III.3 (Fig.4.7 from NM). A good approach is to organize numerical solution along the horizontal and vertical lines. We shall first explain organization of computation in the horizontal direction, i.e. at the u and v grid points, because code along the vertical direction is quite similar to the solution in z coordinate. The important consequence of the sigma coordinate application (which can be easily deduced from the Fig. 4.10 of NM) is that number of grid points along the vertical direction (i. e., number of layers) is always the same at each location. This is not the case for the z coordinate. Due to assumption of a constant layer thickness in z coordinate the number of grid points along the vertical direction will be greater at the greater depth and smaller at the smaller depth. Because each layer in sigma coordinate occurs throughout the entire horizontal domain, one can conclude that sigma coordinate is analogous to the vertically averaged set of equations. Therefore, grid distribution and enumeration along the horizontal direction for the sigma coordinate model is similar to the vertically averaged model. For the irregular water bodies it is important to differentiate between greed points located over water and over dry land. The number of grid points over water along each column and row is variable due to variable coastline.

In irregular water bodies we first learn how to enumerate the grid points which are located in water. The explanation of such procedure is given on p. 134 of NM. We shall explore first this procedure to establish the set of indices for the horizontal

u velocity computation. Thus checking each row from k=1 to k=ky, one can locate at each row starting point (js) and ending point (je) of the water domain. The depth of land and coastal line is set to zero. Because of islands it may happen that in some rows to each k a few different starting and ending points will be prescribed. This is done in the program by counting the number of segments. An individual segment in the program is called, a line. Proceeding in similar way from the first column (j=1) to the last column (j=jx), one can establish beginning and ending indices for the v velocity integration along the y direction. This is all done in the FORTRAN program named index.f. This program reads depth distribution from the file parab.dat and outputs indices for the u and v integration. It also calculates the number of lines along the x and y direction, therefore every set of indices can be related to the proper line number.

```
c PROGRAM ORIGINATED BY DOKUY LEE FROM IOS, VICTORIA
parameter (jx=21,ky=21)
dimension dep(jx,ky)
open (unit=1,file='parab.dat',status='unknown')
open (4,file='parab.ind')
read(1,*)((dep(j,k),j=1,JX),k=1,ky)
c set around your domain zero., because program senses zero
c and non zero depth
c to establish starting and ending index for the horizontal and
c vertical lines.
do j=1,jx
dep(j,1) = 0.
dep(j,2)=0.
c dep(j,3)=0.
enddo
c
do j=1,jx
dep(j,ky) = 0.
dep(j,ky-1) = 0
c dep(j,ky-2)=0.
enddo
c
do k=1,ky
dep(1,k) = 0.
dep(2,k)=0.
c dep(3,k)=0.
enddo
c
do k=1,ky
```

```
dep(jx,k) = 0
dep(jx-1,k)=0.
c dep(jx-2,k)=0.
enddo
c
write(4,101)
101 format(' Begining of horizontal')
c
sum=0 ! sum give the total number of horizontal and vertical lines.
ih = 0 ! ih will tell how many horizontal lines have depth
do k=1,ky ! k stands for the row number
ifirst = 0 !this index helps to recognize line's beginning and end
do j=1,jx
if(dep(j,k).gt.0.0.and.ifirst.eq.0) then
js = j
ifirst = 1
endif
if(dep(j,k).le.0.0.and.ifirst.eq.1) then
je = j-1
ifirst = 0
sum=sum+1
write(4,100) k,k,js,je
100 format(4i4)
ih = ih+1
endif
enddo
enddo
c write(*,*)sum
write(4,102)
102 format(' Begining of vertical')
c
iv = 0 ! this will tell how many vertical lines
do j=1,jx
ifirst = 0
do k=1,ky
if(dep(j,k).gt.0.0.and.ifirst.eq.0) then
ks = k
ifirst = 1
endif
if(dep(j,k).le.0.0.and.ifirst.eq.1) then
ke = k-1
ifirst = 0
```

```
sum=sum+1
c write(6,100) j,j,ks,ke
write(4,100) j,j,ks,ke
iv = iv+1
endif
enddo
enddo
write(*,*)ih,iv,sum
stop
end
```

The output from the above program written to the file parab.ind is
given below. There are 17 lines along horizontal and 17 lines along vertical
directions. Along the horizontal direction first two indices denote row
number,
for example 3 3, stands for the third row. The js=6 and je=16 denotes
beginning and end of integration for the u velocity in the third row.
Similar approach
is taken for the v velocity (vertical lines). Here first two indices denote
column number, for example 3 3 stands for the third column (j=3).
Calculation of v in this column is done from ks=6 to ke=16.

```
3 3 6 16 Begining of horizontal
4 4 5 17
5 5 4 18
6 6 3 19
7 7 3 19
8 8 3 19
9 9 3 19
10 10 3 19
11 11 3 19
12 12 3 19
13 13 3 19
14 14 3 19
15 15 3 19
16 16 3 19
17 17 4 18
18 18 5 17
19 19 6 16
3 3 6 16 Begining of vertical
4 4 5 17
5 5 4 18
6 6 3 19
```

7 7 3 19
8 8 3 19
9 9 3 19
10 10 3 19
11 11 3 19
12 12 3 19
13 13 3 19
14 14 3 19
15 15 3 19
16 16 3 19
17 17 4 18
18 18 5 17
19 19 6 16

As we know how to enumerate variable domain in the horizontal plane let's construct numerical form of the system of equation of motion and continuity in the sigma coordinate. We start by considering the time differencing. A two-time-level scheme will be used for the equations of motion along x and y direction (eqs.III.12, III.13)

$$\frac{(uD)^{m*}-(uD)^m}{T}+\frac{\partial}{\partial x}(u^2D)^m+\frac{\partial}{\partial y}(uvD)^m+\frac{\partial}{\partial \sigma}(Wu)^m-f(Dv)^m$$

$$=-gD^m(\frac{\partial \zeta}{\partial x})^m+N_h[\frac{\partial}{\partial x}D^m\frac{\partial u^m}{\partial x}+\frac{\partial}{\partial y}D^m\frac{\partial u^m}{\partial y}] \tag{III.21a}$$

$$\frac{(uD)^{m+1}-(uD)^{m*}}{T}=\frac{1}{D^m}\frac{\partial}{\partial \sigma}(N_z\frac{\partial u}{\partial \sigma})^{m+1} \tag{III.21b}$$

$$\frac{(vD)^{m*}-(vD)^m}{T}+\frac{\partial}{\partial x}(uvD)^m+\frac{\partial}{\partial y}(v^2D)^m+\frac{\partial}{\partial \sigma}(Wv)^m+f(Du)^{m+1}$$

$$=-gD^m(\frac{\partial \zeta}{\partial y})^m+N_h[\frac{\partial}{\partial x}D^m\frac{\partial v^m}{\partial x}+\frac{\partial}{\partial y}D^m\frac{\partial v^m}{\partial y}] \tag{III.22a}$$

$$\frac{(vD)^{m+1}-(vD)^{m*}}{T}=\frac{1}{D^m}\frac{\partial}{\partial \sigma}(N_z\frac{\partial v}{\partial \sigma})^{m+1} \tag{III.22b}$$

The time differencing in the above equations is split into two substeps. The first substep, from m to $m*$, uses an explicit scheme to compute intermediate value of velocity. (Notice that in eq.III.22a Coriolis force is implicit). At the second substep (marching from $m*$ to $m+1$) an implicit scheme is used to calculate vertical friction terms. This is done in anticipation of the stability problems which can be caused by the vertical friction (N_z). In the sigma coordinate the layer thickness at the shallow water regions is significantly reduced, therefore stability condition (p. 62 from NM) requires small time step of numerical integration, possibly even smaller than the CFL condition. The physics of the vertical exchange of momentum can exacerbate problem as well, because the vertical eddy viscosity can change over the wide range of values, again setting restrictions on the time step used in computations.

Space differencing wil be done with the help of staggered grid. It is useful to notice that the total depth is defined in the ζ points, while above equations require the depth in the u and v points as well. Therefore, in the FORTRAN program two important magnitudes HU and HV are introduced. They define the average depth in the u and v points. Denoting the grid point indices as j, k (see Fig.4.7 from NM), the average depth reads,

$$HU(j,k) = 0.5 * (D(j,k) + D(j-1,k)) \qquad HV(j,k) = 0.5 * (D(j,k) + D(j,k+1)) \tag{III.23}$$

Part of the total depth is variable in time, therefore the new values are computed at every time step. They are called HUN and HVN.

Let's consider time derivative for the u velocity and its interaction with the total depth. It is useful to notice that the values at the $m*$ step are intermediate values and these do not require the knowledge of the total depth. Thus,

$$\frac{(uD)^{m+1} - (uD)^m}{T} = \frac{HUN_{j,k} * u^{m+1}_{j,k,l} - HU_{j,k} * u^m_{j,k,l}}{T} \tag{III.24}$$

Horizontal derivative of advective terms can be constructed in many different ways. Our aim is to achieve second order of approximation in space. Consider term $\frac{\partial}{\partial x}(u^2D)^m$. (Inner product is composed of two terms uD and u). Approach taken here is to find an average value of each term at the ζ points which surround u point and afterwards take difference for the derivative. These ζ points are indexed by j, k and $j-1, k$ respectively. At the right ζ grid point

$$(uD)_r = 0.5 * (HU_{j+1,k} * u_{j+1,k,l} + HU_{j,k} * u_{j,k,l}) \quad u_r = 0.5 * (u_{j,k,l} + u_{j+1,k,l}) \tag{III.25a}$$

At the left ζ grid-point the above terms change to:

$$(uD)_l = 0.5 * (HU_{j-1,k} * u_{j-1,k,l} + HU_{j,k} * u_{j,k,l}) \quad u_l = 0.5 * (u_{j,k,l} + u_{j-1,k,l}) \tag{III.25b}$$

From these two expressions, the finite difference of the first advective term follows

$$\frac{\partial}{\partial x}(u^2D)^m \simeq \frac{(uD)_r * u_r - (uD)_l * u_l}{hx} \tag{III.25}$$

To conceive above expression we used quite a few averaging operations. It is always a good idea to pose and to ask what is the meaning of such operations. We have enough knowledge to conclude that averaging in proximity to the given grid point is equivalent to filtering out some of the short waves. This procedure increases stability of numerical form. To learn what actually happened some numerical experiments are advised, let's say using staggered grid and grid in which velocities and sea level are located at the same grid-point.

Let's continue with the numerical construction of the advective terms. Derivative $\frac{\partial}{\partial y}(uvD)^m$ is taken along the y direction, therefore all variables will be located up and down of the point $u_{j,k}$. At the upper location the two terms are considered

$$(vD)_{up} = 0.5 * (HV_{j,k} * v_{j,k,l} + HV_{j-1,k} * v_{j-1,k,l}) \quad u_{up} = 0.5 * (u_{j,k,l} + u_{j,k+1,l}) \tag{III.26a}$$

and in the similar way at the lower location

$$(vD)_{dw} = 0.5 * (HV_{j,k-1} * v_{j,k-1,l} + HV_{j-1,k-1} * v_{j-1,k-1,l})$$

$$u_{dw} = 0.5 * (u_{j,k,l} + u_{j,k-1,l}) \tag{III.26b}$$

From these expression the derivative is expressed as,

$$\frac{\partial}{\partial y}(uvD)^m \simeq \frac{(vD)_{up} * u_{up} - (vD)_{dw} * u_{dw}}{hy} \tag{III.26}$$

The third nonlinear term, a convective transport $\frac{\partial}{\partial \sigma}(Wu)^m$, requires differencing along the vertical direction. One has to construct derivative based on the values above and below u grid point. Again, since W is not located above u grid, but above pressure grid, the averaging is the order of the day. At the location above $u_{j,k,l}$ grid point

$$W_a = 0.5 * (W_{j,k,l} + W_{j-1,k,l})$$

$$u_a = (u_{j,k,l-1} * hz_l + u_{j,k,l} * hz_{l-1})/(hz_l + hz_{l-1}) \tag{III.27a}$$

$$W_b = 0.5 * (W_{j,k,l+1} + W_{j-1,k,l+1})$$

$$u_b = (u_{j,k,l} * hz_{l+1} + u_{j,k,l+1} * hz_l)/(hz_l + hz_{l+1}) \tag{III.27b}$$

The vertical grid step hz_l is chosen to be variable, therefore above averaging procedure for the u velocity uses weighted averages.

From III.27 a,b the derivative for the convective term follows:

$$\frac{\partial}{\partial \sigma}(Wu)^m \simeq \frac{u_a * W_a - u_b * W_b}{hz_l} \tag{III.27}$$

The Coriolis term along the x direction requires averaging of v component to the u grid point, concomitantly the total depth is also averaged. The following formulas are used

$$v_r = 0.5(v_{j,k,l} + v_{j,k-1,l}) \qquad v_l = 0.5 * (v_{j-1,k,l} + v_{j-1,k-1,l}) \tag{III.28a}$$

$$f(Dv)^m = f * 0.5 * (D_{j,k} * v_r + D_{j-1,k} * v_l) \tag{III.28}$$

Note that v_r and v_l are defined at the pressure (or depth) grid points.

To render numerical form for the sea level and horizontal friction terms we use previous numerical formulas with the small change due to the total depth term.

$$gD^m(\frac{\partial \zeta}{\partial x})^m = g * HU_{j,k} * \frac{\zeta_{j,k} - \zeta_{j-1,k}}{hx} \tag{III.29}$$

$$N_h \frac{\partial}{\partial x} D^m \frac{\partial u^m}{\partial x} \simeq \frac{N_h}{(hx)^2}[u_{j+1,k,l} * D_{j,k} + u_{j-1,k,l} * D_{j-1,k} - u_{j,k,l} * (D_{j,k} + D_{j-1,k})] \tag{III.30a}$$

$$N_h \frac{\partial}{\partial y} D^m \frac{\partial u^m}{\partial y} \simeq \frac{N_h}{(hy)^2}[0.5 * (HU_{j,k} + HU_{j,k-1}) * u_{j,k-1,l} +$$

$$0.5 * (HU_{j,k} + HU_{j,k+1}) * u_{j,k+1,l} - u_{j,k,l} * (0.5 * (HU_{j,k-1} + HU_{j,k+1}) + HU_{j,k})] \tag{III.30b}$$

The vertical friction given by III.21b is implicit in time, therefore it requires not only space differencing but the method for the solving an implicit algorithm as well. To clarify location of the vertical grid points one can use figures 4.1 and 4.7 from NM and also figure III.3. This derivative is not along the vertical directions but along direction perpendicular to the surfaces of constant σ.

$$\frac{\partial}{\partial \sigma}(N_z \frac{\partial u}{\partial \sigma})^{m+1} \simeq \frac{1}{h_{z,l}}[N_{z,l} \frac{u^{m+1}_{j,k,l-1} - u^{m+1}_{j,k,l}}{0.5 * (h_{z,l} + h_{z,l-1})} - N_{z,l+1} \frac{u^{m+1}_{j,k,l} - u^{m+1}_{j,k,l+1}}{0.5 * (h_{z,l} + h_{z,l+1})}] \tag{III.31a}$$

The above formula combined with the time derivative leads to:

$$\frac{HUN_{j,k} * u^{m+1}_{j,k,l} - HU_{j,k} * u^{m*}}{T} =$$

$$\frac{1}{HU_{j,k}}\frac{1}{h_{z,l}}[N_{z,l}\frac{u^{m+1}_{j,k,l-1}-u^{m+1}_{j,k,l}}{0.5*(h_{z,l}+h_{z,l-1})}-N_{z,l+1}\frac{u^{m+1}_{j,k,l}-u^{m+1}_{j,k,l+1}}{0.5*(h_{z,l}+h_{z,l+1})}] \tag{III.31}$$

Here u^{m*} was calculated at the previous time step, when computation from the level m to the level $m*$ by the explicit method was done. Equation III.31 allows us to organize a three-point algorithm suitable for the line inversion method. Solution of this problem is described in Appendix 1 from NM, and also in the Problem No. 1 of this chapter.

Along the y (north-south) direction all terms in the equation (III.22a) ought to be cast into numerical form in the manner we just did for the x direction. The numerical form of the time derivative for the v velocity is,

$$\frac{(vD)^{m+1}-(vD)^m}{T}\simeq\frac{HVN_{j,k}*v^{m+1}_{j,k}-HV_{j,k}*v^m_{j,k}}{T} \tag{III.32}$$

Let's now establish numerical form of the advective terms at the v point. The term $\frac{\partial}{\partial x}(vuD)^m$ is composed of two terms uD and v. Approach taken here is to find an average value of each term at the points which are located half of the space step to the right and half of the space step to the left from the v point;

$$(uD)_r = 0.5*(HU_{j+1,k+1}*u_{j+1,k+1,l}+HU_{j+1,k}*u_{j+1,k,l})$$

$$v_r = 0.5*(v_{j,k,l}+v_{j+1,k,l}) \tag{III.33a}$$

$$(uD)_l = 0.5*(HU_{j,k+1}*u_{j,k+1,l}+HU_{j,k}*u_{j,k,l})$$

$$v_l = 0.5*(v_{j,k,l}+v_{j-1,k,l}) \tag{III.33b}$$

Formula for the advective derivative follows,

$$\frac{\partial}{\partial x}(vuD)^m \simeq \frac{(uD)_r*v_r-(uD)_l*v_l}{hx} \tag{III.33}$$

The second advective term $\frac{\partial}{\partial y}(vvD)^m$ will be cast into numerical form by considering its values at the ζ points which surround the v point. These ζ points are indexed as $j,k+1$ and j,k. Thus, at the location of the upper ζ grid point

$$(vD)_{up} = 0.5*(HV_{j,k+1}*v_{j,k+1,l}+HV_{j,k}*v_{j,k,l})$$

$$v_{up} = 0.5*(v_{j,k+1,l}+v_{j,k,l}) \tag{III.34a}$$

$$(vD)_{dw} = 0.5 * (HV_{j,k} * v_{j,k,l} + HV_{j,k-1} * v_{j,k-1,l})$$

$$v_{dw} = 0.5 * (v_{j,k,l} + v_{j,k-1,l}) \tag{III.34b}$$

From the above formulas

$$\frac{\partial}{\partial y}(vvD)^m \simeq \frac{(vD)_{up} * v_{up} - (vD)_{dw} * v_{dw}}{hy} \tag{III.34}$$

The construction of the numerical form for the convective term will require the values, above and below W velocity grid point

$$W_a = 0.5 * (W_{j,k,l} + W_{j,k+1,l})$$

$$v_a = (v_{j,k,l-1} * hz_l + v_{j,k,l} * hz_{l-1})/(hz_l + hz_{l-1}) \tag{III.35a}$$

$$W_b = 0.5 * (W_{j,k,l+1} + W_{j,k+1,l+1})$$

$$v_b = (v_{j,k,l} * hz_{l+1} + v_{j,k,l+1} * hz_l)/(hz_l + hz_{l+1}) \tag{III.35b}$$

From above expressions, the convective term for the v grid-point follows:

$$\frac{\partial}{\partial \sigma}(Wv)^m \simeq \frac{v_a * W_a - v_b * W_b}{hz_l} \tag{III.35}$$

The Coriolis term (fDu) along the y direction requires averaging of u component to the v grid point with a concomitant depth averaging. The following formulas are used

$$u_u = 0.5(un_{j+1,k+1,l} + un_{j,k+1,l}) \qquad u_{dw} = 0.5 * (un_{j+1,k,l} + un_{j,k,l}) \tag{III.36a}$$

$$f(Du)^m = f * 0.5 * (DN_{j,k+1} * u_u + DN_{j,k} * u_{dw}) \tag{III.36}$$

Note that u_u and u_{dw} are defined at the pressure (or depth) grid points.

Differencing the sea level along the y direction results in following pressure term,

$$gD^m(\frac{\partial \zeta}{\partial y}) \simeq g * HV_{j,k} * \frac{\zeta_{j,k+1} - \zeta_{j,k}}{hy} \tag{III.37}$$

The horizontal friction term requires separate consideration for the x and y directions

$$N_h \frac{\partial}{\partial x} D^m \frac{\partial v^m}{\partial x} \simeq \frac{0.5 * N_h}{(hx)^2}[(HV_{j,k} + HV_{j+1,k}) * v_{j+1,k,l} +$$

$$(HV_{j-1,k} + HV_{j,k}) * v_{j-1,k,l} - v_{j,k,l} * (HV_{j+1,k} + HV_{j-1,k}) + 2. * HV_{j,k})] \quad \text{(III.38a)}$$

$$N_h \frac{\partial}{\partial y} D^m \frac{\partial v^m}{\partial y} \simeq \frac{N_h}{(hy)^2}[v_{j,k+1,l} * D_{j,k+1} + v_{j,k-1,l} * D_{j,k} - v_{j,k,l} * (D_{j,k} + D_{j,k+1})] \quad \text{(III.38b)}$$

Solution of the the vertical friction term given by III.22b is implicit in time, therefore it requires not only space differencing but the method for the solving an implicit algorithm in time is needed as well. Here we shall use the method used previously along the x direction. To clarify location of the v points along the vertical direction figures 4.1 and 4.7 from NM and also figures III.3 and III.3a can be very useful. It is important to notice that variable eddy viscosity coefficient is located between the v points.

$$\frac{\partial}{\partial \sigma}(N_z \frac{\partial v}{\partial \sigma})^{m+1} \simeq \frac{1}{h_{z,l}}[N_{z,l} \frac{v^{m+1}_{j,k,l-1} - v^{m+1}_{j,k,l}}{0.5 * (h_{z,l} + h_{z,l-1})} - N_{z,l+1} \frac{v^{m+1}_{j,k,l} - v^{m+1}_{j,k,l+1}}{0.5 * (h_{z,l} + h_{z,l+1})}] \quad \text{(III.39a)}$$

Using previous approach, the above formula can be combined with the time derivative

$$\frac{HVN_{j,k} * v^{m+1}_{j,k,l} - HV_{j,k} * v^{m*}}{T} =$$

$$\frac{1}{HV_{j,k}} \frac{1}{h_{z,l}}[N_{z,l} \frac{v^{m+1}_{j,k,l-1} - v^{m+1}_{j,k,l}}{0.5 * (h_{z,l} + h_{z,l-1})} - N_{z,l+1} \frac{v^{m+1}_{j,k,l} - v^{m+1}_{j,k,l+1}}{0.5 * (h_{z,l} + h_{z,l+1})}] \quad \text{(III.39)}$$

In the above expression v^{m*} is known from the previous computational step when the time steping from level m to $m*$ was performed by an explicit method. The above equation allows us to organize a three-point algorithm suitable for the line inversion method. Thus the time stepping is achieved by solving a three-point algorithm as a boundary problem.

Two dependent variables are still unknown in the equations III.21 and III.22, i.e., vertical velocity (W) and sea level (ζ). To define these variables equation of continuity and vertically averaged equation of continuity will be used. Let us begin by considering the vertically averaged equation III.18

$$\frac{\partial}{\partial x}(\bar{u}D) + \frac{\partial}{\partial y}(\bar{v}D) + \frac{\partial \zeta}{\partial t} = 0 \tag{III.40}$$

The components of the depth-averaged velocity in the above equation are expressed as

$$\bar{u} = \int_{-1}^{0} u d\sigma \qquad \bar{v} = \int_{-1}^{0} v d\sigma$$

First step is to derive the vertically averaged velocity in every horizontal grid point by changing above integrals to the sums. The vertically averaged velocity components are named *ua* and *va*. The space derivatives in III.40 are cast into numerical form through approach we have applied above,

$$\frac{\zeta_{j,k}^{m+1} - \zeta_{j,k}^{m}}{T} = -\frac{ua_{j+1,k}^{m} * HUN_{j+1,k} - ua_{j,k}^{m} * HUN_{j,k}}{hx}$$
$$-\frac{va_{j,k}^{m} * HVN_{j,k} - va_{j,k-1}^{m} * HVN_{j,k-1}}{hy} \tag{III.41}$$

Vertical velocity in the σ coordinate is calculated from the equation of continuity III.14. This equation will be solved at the ζ point, thus

$$\frac{\zeta_{j,k}^{m+1} - \zeta_{j,k}^{m}}{T} + \frac{u_{j+1,k,l}^{m+1} * HUN_{j+1,k} - u_{j,k,l}^{m+1} * HUN_{j,k}}{hx} +$$
$$\frac{v_{j,k,l}^{m+1} * HVN_{j,k} - v_{j,k-1,l}^{m+1} * HVN_{j,k-1}}{hy} + \frac{W_{j,k,l}^{m+1} - W_{j,k,l+1}^{m+1}}{hz_l} = 0 \tag{III.42}$$

Where W is the vertical velocity in the new system of coordinate. It is defined as a normal velocity to σ surfaces. It can be quite challenging to plot graphics in the σ coordinate, and we will not follow the "call of this challenge". Graphics will be done in the old z system. To plot velocity vectors we need to know how to change the vertical velocity from σ to z coordinate. This transformation is given by III.15. The old vertical velocity is denoted as w.

$$w = W + (\sigma + 1)\frac{\partial \zeta}{\partial t} + uQ_x + vQ_y \tag{III.43(III.15)}$$

It can be easily programmed but the variables in the above equations ought to be averaged to the vertical velocity grid points.

Ensuing program seasie.f is given for computation of the wind-driven currents in the circular water body with parabolic bathymetry. The program is self explanatory.

Maximum depth is 130m and this depth is divided into 20 levels. Minimim depth is 5m and it is also divided into 20 levels.

```
cc program seasi.f Wind-driven motion, implicit vert. friction
c sigma coordinate c density is not included
c This is 3d motion in circular water body with parabolic depth
PARAMETER (JX=21,KY=21,LE=20,NLH=17,NLV=17)
REAL UN(JX,KY,LE),UO(JX,KY,LE),UN1(JX,KY,LE)
REAL ZN(JX,KY),ZO(JX,KY),H(JX,ky),ENU(JX,KY,LE)
REAL HZ(LE),TAUSX(JX,KY),TAUBX(JX,KY),UA(JX,KY)
REAL W(JX,KY,LE+1),hu(jx,ky),upX(jx,le),wpX(jx,le)
REAL upY(KY,le),wpY(KY,le),upxx(JX,KY,LE),UPYY(JX,KY,LE)
real uai(jx,KY),HUN(JX,ky),HN(JX,ky)
REAL VAI(JX,KY),VO(JX,KY,LE),VN(JX,KY,LE),ENV(JX,KY,LE)
REAL DHU(JX,ky),DV(JX,ky,LE),TAUSY(JX,KY)
REAL TAUBY(JX,KY),HO(JX,KY),VA(JX,KY),hv(jx,ky),hvn(jx,ky)
REAL VN1(JX,KY,LE),WZ(JX,KY,LE+1)
real a(le+1),b(LE+1),c(LE+1),EN(JX,KY,LE)
real d(LE+1),e(LE+1),s(LE+1)
integer js(nlh),je(nlh),ks(nlh),ke(nlh)
integer jsv(nlv),jev(nlv),ksv(nlv),kev(nlv)
open (unit=4,file='parab.dat',status='unknown')
open (unit=3,file='parab.ind',status='unknown')
open (unit=9,file='wpy.dat',status='unknown')
open (unit=10,file='z.dat',status='unknown')
open (unit=11,file='momaxz1.dat',status='unknown')
open (unit=12,file='eni.dat',status='unknown')
open (unit=15,file='wpx.dat',status='unknown')
open (unit=16,file='upx.dat',status='unknown')
open (unit=17,file='upy.dat',status='unknown')
open (unit=20,file='depthx.dat',status='unknown')
open (unit=21,file='depthy.dat',status='unknown')
open (unit=7,file='UPXX.dat',status='unknown')
open (unit=8,file='UPYY.dat',status='unknown')
read(4,*)((h(j,k),j=1,JX),k=1,ky)
do j=1,jx
do K=1,KY
IF(H(J,K).LT.500.)H(J,K)=500.
hn(j,k)=h(j,k)
ho(j,k)=h(j,k)
END DO
END DO
```

```
c Index : horizontal line, vertical
c write (6,101) nlh
c 101 format(4i6)
do i=1,nlh
read (3,100) ks(i),ke(i),js(i),je(i)
100 format(4i4)
enddo
c
c write (6,101) nlv
do i=1,nlv
read (3,100) jsv(i),jev(i),ksv(i),kev(i)
enddo
c
close (3)
write(20,*)(h(j,11),j=1,jx)
write(21,*)(h(11,k),k=1,ky)
DO K=1,KY
HU(1,K)=HO(1,K)
HUN(1,K)=HN(1,K)
end do
do j=2,jx
do k=1,ky
HU(j,k)=0.5*(Ho(j-1,k)+ho(j,k))
HUN(j,k)=0.5*(Hn(j-1,k)+hn(j,k))
end do
end do
do j=1,JX
HV(J,KY)=Ho(J,KY)
HVN(J,KY)=Hn(J,KY)
END DO
DO J=1,JX
DO K=1,KY-1
HV(J,K)=0.5*(HO(J,K)+HO(J,K+1))
HVN(J,K)=0.5*(HN(J,K)+HN(J,K+1))
end do
end do
DO J=1,JX
DO K=1,KY
ua(j,K)=0.
va(j,k)=0.
zo(j,K)=0.
zn(j,k)=0.
```

```
end do
END DO
do j=1,jx
DO K=1,KY
do l=1,le
un(j,K,l)=0.
un1(j,k,l)=0.
uo(j,K,l)=0.
vn(J,K,L)=0.
VO(J,K,L)=0.
VN1(J,K,L)=0.
end do
end do
END DO
C SET variable SPACE STEP ALONG VERTICAL (sigma) DIRECTION
DO L=1,le
HZ(L)=0.05
END DO
C ALL COEFFICIENTS
c****************************************************
c Time step
T = 100.
c HORIZONTAL EDDY VISCOSITY
AH=1.0e8
C HORIZONTAL EDDY DIFFUSIVITY along X AXIX
DO J=1,JX
DO K=1,KY
DHU(J,K)=0.
END DO
END DO
do n=1,nlh
do j=js(n)+1,je(n)
do k=ks(n),ke(n)
DHU(J,K)=1.0e7
END DO
END DO
end do
FC=1.458E-4
G=981.
c SPACE STEPS ALONG X AND Y
HX=10.E5
HY=10.E5
```

```
RF1=2.6E-3
RF2=2.6E-3
C COMBINATIONAL PARAMETERS
TRF=T*RF1
PL=T/HX
GPL=G*PL
PLL=AH*T/(HX*HX)
PF=T/HY
GPF=G*PF
PFF=AH*T/(HY*HY)
TFC=T*FC
TWHX=2.*hx ! for vertical velocity
TWHY=2.*hY ! for vertical velocity
C START TIME LOOP INDEX NN RUNS UNTIL NN=MON
c***********************************
c MON=86400
mon=10000
nn=0
C888888888888888888888888888888888888
50 Nn=Nn+1
C SOME COEFFICIENTS ARE VARIABLE
C SET WIND FOR INITIAL EXPERIMENT
DO J=1,JX
DO K=1,KY
TAUSX(J,K)=1.
TAUSY(J,K)=0.
END DO
END DO
c if (n.lt.1000)then
c DO J=1,JX
c DO K=1,KY
c TAUSX(J,K)=float(n)/1000.
c TAUSY(J,K)=0.
c END DO
c END DO
c end if
C EDDY VISCOSITY IS CONSTANT
DO 33J=1,JX
do 33 k=1,ky
DO 33 L=1,Le
EN(J,k,L)=200.
33 CONTINUE
```

```
DO J=1,JX
do k=1,ky
DO L=1,LE
ENU(J,k,L)=0.5*(EN(J,K,L)+EN(J,K-1,L))
ENV(J,K,L)=0.5*(EN(J,K,L)+EN(J+1,K,L))
DV(J,K,L)=100.
END DO
END DO
END DO
c IN SPACE COEFFICIENT OF EDDY VISCOSITY SHOULD BE
C LOCATED IN Z (OR PRESSURE) POINTS
C COMPUTATION OF U VELOCITY, closed boundary, velocity is zero at
C J=1 AND J=JX.
c - + + - + - +
c 1 jx jx+1
c left right
C VELOCITY IN THE AIR
C go to 225
DO 499 n=1,nlh
do 499j=js(n)+1,je(n)
do 499k=ks(n),ke(n)
UAI(J,K)=UO(J,K,1)+HZ(1)*hu(j,K)*TAUSX(J,K)/ENU(J,K,1)
499 continue
DO 4 n=1,nlh
do 4 j=js(n)+1,je(n)
do 4 k=ks(n),ke(n)
C THIS GOES INTO HORIZONTAL FRICTION
DHU1=0.5*(HU(J,K)+HU(J,K+1))
DHU2=0.5*(HU(J,K)+HU(J,K-1))
C SURFACE CONDITION
C CORIOLIS
VR=0.5*(VO(J,K,1)+VO(J,K-1,1))
VL=0.5*(VO(J-1,K,1)+VO(J-1,K-1,1))
UAV=0.5*(HO(J,K)*VR+HO(J-1,K)*VL)
DSL=ZO(J,K)-ZO(J-1,K)
DSLX=DSL*GPL*HU(J,K)
HZP=0.5*(HZ(1)+HZ(2))*hu(j,K)
DSU=(UO(J,K,1)-UO(J,K,2))/HZP
c VFRS=(TAUSX(J,K)-ENU(J,K,2)*DSU)/HZ(1)
C HORIZONTAL FRICTION
HFRUX=PLL*(UO(J+1,K,1)*H(J,K)+UO(J-1,K,1)*H(J-1,K)
2-(H(J,K)+H(J-1,K))*UO(J,K,1))
```

```
HFRUY=PFF*(UO(J,K+1,1)*DHU1+UO(J,K-1,1)*DHU2
3-(DHU1+DHU2)*UO(J,K,1))
HFR=HFRUX+HFRUY
C EVEN BETTER IMPROVED NONLINEAR TERMX
c uw
USN=0.5*(UAI(J,K)+UO(J,K,1))
USP=(UO(J,K,1)*HZ(2)+UO(J,K,2)*HZ(1))/(HZ(1)+HZ(2))
waxn=0.5*(W(J-1,K,1)+W(J,K,1))
WAXP=0.5*(W(J-1,K,2)+W(J,K,2))
UW=T*(WAXN*USN-WAXP*USP)/HZ(1)
C EVEN BETTER IMPROVED NONLINEAR TERMX
c uv uu
UAF=0.5*(UO(J,K,1)+UO(J+1,K,1))
UAB=0.5*(UO(J,K,1)+UO(J-1,K,1))
ONXF=0.5*(UO(J,K,1)*HU(J,K)+UO(J+1,K,1)*HU(J+1,K))
ONXB=0.5*(UO(J,K,1)*HU(J,K)+UO(J-1,K,1)*HU(J-1,K))
SUMNONX=PL*(ONXF*UAF-ONXB*UAB)
C UV
UAU=0.5*(UO(J,K,1)+UO(J,K+1,1))
UAD=0.5*(UO(J,K-1,1)+UO(J,K,1))
ONYU=0.5*(VO(J,K,1)*HV(J,K)+VO(J-1,K,1)*HV(J-1,K))
ONYD=0.5*(VO(J,K-1,1)*HV(J,K-1)+VO(J-1,K-1,1)*HV(J-1,K-1))
SUMNONY=PF*(UAU*ONYU-UAD*ONYD)
SUMNON=SUMNONX+SUMNONY+UW
UN1(J,K,1)=UO(J,K,1)*HU(J,K)-DSLX+TFC*UAV+HFR-SUMNON
UN1(J,K,1)=UN1(J,K,1)/HUN(J,K)
LMM=LE
DO 5 L=2,LMM-1
HZP=0.5*(HZ(L)+HZ(L+1))*HU(J,K)
HZN=0.5*(HZ(L)+HZ(L-1))*HU(J,K)
DSUN=(UO(J,K,L-1)-UO(J,K,L))/HZN
DSUP=(UO(J,K,L)-UO(J,K,L+1))/HZP
c VFR=(ENU(J,K,L)*DSUN-ENU(J,K,L+1)*DSUP)/HZ(L)
C HORIZONTAL FRICTION
HFRUX=PLL*(UO(J+1,K,L)*H(J,K)+UO(J-1,K,L)*H(J-1,K)
2-UO(J,K,L)*(H(J,K)+H(J-1,K)))
HFRUY=PFF*(UO(J,K+1,L)*DHU1+UO(J,K-1,L)*DHU2
3-(DHU1+DHU2)*UO(J,K,L))
HFR=HFRUX+HFRUY
C CORIOLIS
VR=0.5*(VO(J,K,L)+VO(J,K-1,L))
VL=0.5*(VO(J-1,K,L)+VO(J-1,K-1,L))
```

```
UAV=0.5*(HO(J,K)*VR+HO(J-1,K)*VL)
C EVEN BETTER IMPROVED NONLINEAR TERMX
UAF=0.5*(UO(J,K,L)+UO(J+1,K,L))
UAB=0.5*(UO(J,K,L)+UO(J-1,K,L))
ONXF=0.5*(UO(J,K,L)*HU(J,K)+UO(J+1,K,L)*HU(J+1,K))
ONXB=0.5*(UO(J,K,L)*HU(J,K)+UO(J-1,K,L)*HU(J-1,K))
SUMNONX=PL*(ONXF*UAF-ONXB*UAB)
C UV
UAU=0.5*(UO(J,K,L)+UO(J,K+1,L))
UAD=0.5*(UO(J,K-1,L)+UO(J,K,L))
ONYU=0.5*(VO(J,K,L)*HV(J,K)+VO(J-1,K,L)*HV(J-1,K))
ONYD=0.5*(VO(J,K-1,L)*HV(J,K-1)+VO(J-1,K-1,L)*HV(J-1,K-1))
SUMNONY=PF*(UAU*ONYU-UAD*ONYD)
SUMNON=SUMNONX+SUMNONY
c vert deriv.
waxn=0.5*(W(J-1,K,L)+W(J,K,L))
WAXP=0.5*(W(J-1,K,L+1)+W(J,K,L+1))
USN=(UO(J,K,L-1)*HZ(L)+UO(J,K,L)*HZ(L-1))/(HZ(L)+HZ(L-1))
USP=(UO(J,K,L)*HZ(L+1)+UO(J,K,L+1)*HZ(L))/(HZ(L)+HZ(L+1))
UW=T*(WAXN*USN-WAXP*USP)/HZ(L)
SUMNON=SUMNON+UW
UN1(J,K,L)=UO(J,K,L)*HU(J,K)-DSLX+TFC*UAV+HFR-SUMNON
UN1(J,K,L)=UN1(J,K,L)/HUN(J,K)
5 CONTINUE
C BOTTOM CONDITION
C CORIOLIS
VR=0.5*(Vo(J,K,LMM)+Vo(J,K-1,LMM))
VL=0.5*(Vo(J-1,K,LMM)+Vo(J-1,K-1,LMM))
UAV=0.5*(HO(J,K)*VR+HO(J-1,K)*VL)
c UAV1=0.5*(VR+VL)
HZP=HZ(LMM)*HU(J,K)
HZN=0.5*(HZ(LMM)+HZ(LMM-1))*HU(J,K)
DSUP=2.*UO(J,K,LMM)/HZP
DSUN=(UO(J,K,LMM-1)-UO(J,K,LMM))/HZN
C HORIZONTAL FRICTION
HFRUX=PLL*(UO(J+1,K,LMM)*H(J,K)+UO(J-1,K,LMM)*H(J-1,K)
2-UO(J,K,LMM)*(H(J,K)+H(J-1,K)))
HFRUY=PFF*(UO(J,K+1,LMM)*DHU1+UO(J,K-1,LMM)*DHU2
3-(DHU1+DHU2)*UO(J,K,LMM))
HFR=HFRUX+HFRUY
C EVEN BETTER IMPROVED NONLINEAR TERMX
UAF=0.5*(UO(J,K,LMM)+UO(J+1,K,LMM))
```

```
UAB=0.5*(UO(J,K,LMM)+UO(J-1,K,LMM))
ONXF=0.5*(UO(J,K,LMM)*HU(J,K)+UO(J+1,K,LMM)*HU(J+1,K))
ONXB=0.5*(UO(J,K,LMM)*HU(J,K)+UO(J-1,K,LMM)*HU(J-1,K))
SUMNONX=PL*(ONXF*UAF-ONXB*UAB)
C UV
UAU=0.5*(Uo(J,K,LMM)+Uo(J,K+1,LMM))
UAD=0.5*(Uo(J,K-1,LMM)+Uo(J,K,LMM))
ONYU=0.5*(Vo(J,K,LMM)*HV(J,K)+Vo(J-1,K,LMM)*HV(J-1,K))
ONYD=0.5*(Vo(J,K-1,LMM)*HV(J,K-1)+
2Vo(J-1,K-1,LMM)*HV(J-1,K-1))
SUMNONY=PF*(UAU*ONYU-UAD*ONYD)
SUMNON=SUMNONX+SUMNONY
c vert deriv.
HZMM=HZ(LMM)+HZ(LMM-1)
waxn=0.5*(W(J-1,K,LMM)+W(J,K,LMM))
WAXP=0.
USN=(UO(J,K,L-1)*HZ(LMM)+UO(J,K,L)*HZ(LMM-1))/HZMM
USP=0.
UW=T*(WAXN*USN-WAXP*USP)/HZ(LMM)
SUMNON=SUMNON+UW
UN1(J,K,LMM)=UO(J,K,LMM)*HU(J,K)-DSLX+
TFC*UAV+HFR-SUMNON
UN1(J,K,LMM)=UN1(J,K,LMM)/HUN(J,K)
C UN(J,LMM)=0.
4 CONTINUe
c VERTICAL FRICTION SPLIT FROM THE REST
DO 46 n=1,nlh
do 46 j=js(n)+1,je(n)
do 46 k=ks(n),ke(n)
TAUSX(J,k)=1.
VR=0.5*(Vo(J,K,LE)+Vo(J,K-1,LE))
VL=0.5*(Vo(J-1,K,LE)+Vo(J-1,K-1,LE))
UAV1=0.5*(VR+VL)
HZPP=HZ(LE)*HUn(J,K)
C LOG OR NOT TO LOG
CD=2.5*ALOG(1.25*HZPP)
CD=1./(CD*CD)
RF1=AMAX1(CD,RF2)
PIERW=SQRT(UO(J,K,Le)*UO(J,K,Le)+uav1*uav1)
TAUBX(J,k)=RF1*PIERW*UO(J,K,Le)
HZP=0.5*(HZ(1)+HZ(2))*hu(j,k)
C(1)=T*ENU(J,K,2)/(HZ(1)*huN(j,k)*HZP)
```

```
D(1)=UN1(J,K,1)
DO L=2,LE-1
HZP=0.5*(HZ(L)+HZ(L+1))*hu(j,k)
HZN=0.5*(HZ(L)+HZ(L-1))*hu(j,k)
A(L)=T*ENU(j,k,L)/(HZ(L)*huN(j,k)*HZN)
C(L)=T*ENU(J,K,L+1)/(HZ(L)*huN(j,k)*HZP)
B(L)=1+A(L)+C(L)
D(L)=UN1(J,K,L)
END DO
HZN=0.5*(HZ(LE)+HZ(LE-1))*hu(j,k)
HZP=0.5*(HZ(LE)+HZ(LE+1))*hu(j,k)
D(LE)=UN1(J,K,LE)
A(Le)=T*ENU(j,k,Le)/(HZ(Le)*huN(j,k)*Hzn)
C(Le)=T*ENU(J,K,Le)/(HZ(Le)*huN(j,k)*HZP)
B(Le)=1+A(Le)+C(Le)
S(1)=C(1)/(1.+C(1))
E(1)=UN1(J,K,1)+T*TAUSX(J,K)/(HZ(1)*huN(j,k))
E(1)=E(1)/(1.+C(1))
DO L=2,LE-1
C VERTICAL FRICTION CONTINUE
DENOM=B(L)-S(L-1)*A(L)
S(L)=C(L)/DENOM
E(L)=(D(L)+A(L)*E(L-1))/DENOM
END DO
A1=A(LE)*E(LE-1)+D(LE)-T*TAUBX(J,K)/(HZ(LE)*huN(j,k))
A2=1.+A(le)*(1.-S(LE-1))
UN(J,K,LE)=A1/A2
DO L=LE-1,1,-1
UN(J,K,L)=UN(J,K,L+1)*S(L)+E(L)
END DO
46 CONTINUE
C COMPUTATION OF V VELOCITY
C VELOCITY IN THE AIR
do 2001 m=1,nlv
c
do 2001 j=jsv(m),jev(m)
do 2001 k=ksv(m),kev(m)-1
C THIS GOES INTO HORIZONTAL FRICTION
DHV1=0.5*(HV(J,K)+HV(J+1,K))
DHV2=0.5*(HV(J,K)+HV(J-1,K))
VAI(J,K)=VO(J,K,1)+HZ(1)*hV(j,K)*TAUSY(J,K)/ENV(J,K,1)
2001 continue
```

```
do 6 m=1,nlv
c if(ksv(m).eq.kev(m)) go to 6
do 6 j=jsv(m),jev(m)
do 6 k=ksv(m),kev(m)-1
C SURFACE CONDITION
DSL=ZO(J,K+1)-ZO(J,K)
DSLY=GPF*DSL*HV(J,K)
HZP=0.5*(HZ(1)+HZ(2))*HV(J,K)
C CORIOLIS TERM
UG=0.5*(UN(J,K+1,1)+UN(J+1,K+1,1))
UD=0.5*(UN(J,K,1)+UN(J+1,K,1))
VAUN=0.5*(HN(J,K+1)*UG+HN(J,K)*UD)
DSV=(VO(J,K,1)-VO(J,K,2))/HZP
C VFRS=(TAUSY(J,K)-ENV(J,K,2)*DSV)/HZ(1)
C HORIZONTAL FRICTION
HFRVY=PFF*(VO(J,K+1,1)*H(J,K+1)+VO(J,K-1,1)*H(J,K)
2-(H(J,K)+H(J,K+1))*VO(J,K,1))
HFRVX=PLL*(VO(J+1,K,1)*DHV1+VO(J-1,K,1)*DHV2
3-(DHV1+DHV2)*VO(J,K,1))
HFR=HFRVX+HFRVY
C HFR=PLL*HV(J,K)*(VO(J+1,K,1)+VO(J-1,K,1)-2.*VO(J,K,1))+
C 2PFF*HV(J,K)*(VO(J,K+1,1)+VO(J,K-1,1)-2.*VO(J,K,1))
C EVEN BETTER NONLINEAR
C UV
VALX=0.5*(VO(J,K,1)+VO(J-1,K,1))
VARX=0.5*(VO(J,K,1)+VO(J+1,K,1))
ONXL=0.5*(HU(J,K+1)*UO(J,K+1,1)+HU(J,K)*UO(J,K,1))
ONXR=0.5*(HU(J+1,K+1)*UO(J+1,K+1,1)+HU(J+1,K)*UO(J+1,K,1))
SUMNONX=PF*(ONXR*VARX-ONXL*VALX)
C VV
VAGY=0.5*(VO(J,K+1,1)+VO(J,K,1))
VADY=0.5*(VO(J,K-1,1)+VO(J,K,1))
ONYU=0.5*(HV(J,K+1)*VO(J,K+1,1)+HV(J,K)*VO(J,K,1))
ONYD=0.5*(HV(J,K-1)*VO(J,K-1,1)+HV(J,K)*VO(J,K,1))
SUMNONY=PF*(ONYU*VAGY-ONYD*VADY)
SUMNON=SUMNONX+SUMNONY
CWV
VSU=0.5*(VO(J,K,1)+VAI(J,K))
VSD=(VO(J,K,1)*HZ(2)+VO(J,K,2)*HZ(1))/(HZ(1)+HZ(2))
WAG=0.5*(W(J,K,1)+W(J,K+1,1))
WAD=0.5*(W(J,K,2)+W(J,K+1,2))
WV=T*(VSU*WAG-VSD*WAD)/HZ(1)
```

```
SUMNON=SUMNON+WV
VN1(J,K,1)=VO(J,K,1)*HV(J,K)-TFC*VAUN-DSLY+HFR-
2SUMNON
VN1(J,K,1)=VN1(J,K,1)/HVN(J,K)
DO L=2,LE-1
C CORIOLIS TERM
UG=0.5*(UN(J,K+1,L)+UN(J+1,K+1,L))
UD=0.5*(UN(J,K,L)+UN(J+1,K,L))
VAUN=0.5*(HN(J,K+1)*UG+HN(J,K)*UD)
HZP=0.5*(HZ(L)+HZ(L+1))*HV(J,K)
HZN=0.5*(HZ(L)+HZ(L-1))*HV(J,K)
DSVN=(VO(J,K,L-1)-VO(J,K,L))/HZN
DSVP=(VO(J,K,L)-VO(J,K,L+1))/HZP
C VFR=(ENV(J,K,L)*DSVN-ENV(J,K,L+1)*DSVP)/HZ(L)
C HORIZONTAL FRICTION
C HFR=PLL*HV(J,K)*(VO(J+1,K,L)+VO(J-1,K,L)-2.*VO(J,K,L))+
C 2PFF*HV(J,K)*(VO(J,K+1,L)+VO(J,K-1,L)-2.*VO(J,K,L))
C HORIZONTAL FRICTION
HFRVY=PFF*(VO(J,K+1,L)*H(J,K+1)+VO(J,K-1,L)*H(J,K)
2-(H(J,K)+H(J,K+1))*VO(J,K,L))
HFRVX=PLL*(VO(J+1,K,L)*DHV1+VO(J-1,K,L)*DHV2
3-(DHV1+DHV2)*VO(J,K,L))
HFR=HFRVX+HFRVY
C EVEN BETTER NONLINEAR
C UV
VALX=0.5*(VO(J,K,L)+VO(J-1,K,L))
VARX=0.5*(VO(J,K,L)+VO(J+1,K,L))
ONXL=0.5*(HU(J,K+1)*UO(J,K+1,L)+HU(J,K)*UO(J,K,L))
ONXR=0.5*(HU(J+1,K+1)*UO(J+1,K+1,L)+HU(J+1,K)*UO(J+1,K,L))
SUMNONX=PF*(ONXR*VARX-ONXL*VALX)
C VV
VAGY=0.5*(VO(J,K+1,L)+VO(J,K,L))
VADY=0.5*(VO(J,K-1,L)+VO(J,K,L))
ONYU=0.5*(HV(J,K+1)*VO(J,K+1,L)+HV(J,K)*VO(J,K,L))
ONYD=0.5*(HV(J,K-1)*VO(J,K-1,L)+HV(J,K)*VO(J,K,L))
SUMNONY=PF*(ONYU*VAGY-ONYD*VADY)
SUMNON=SUMNONX+SUMNONY
CWV
VSU=(VO(J,K,L-1)*HZ(L)+VO(J,K,L)*HZ(L-1))/(HZ(L)+HZ(L-1))
VSD=(VO(J,K,L)*HZ(L+1)+VO(J,K,L+1)*HZ(L))/(HZ(L)+HZ(L+1))
WAG=0.5*(W(J,K,L)+W(J,K+1,L))
WAD=0.5*(W(J,K,L+1)+W(J,K+1,L+1))
```

```
WV=T*(VSU*WAG-VSD*WAD)/HZ(L)
SUMNON=SUMNON+WV
VN1(J,K,L)=VO(J,K,L)*HV(J,K)-TFC*VAUN-DSLY+HFR-
2SUMNON
VN1(J,K,L)=VN1(J,K,L)/HVN(J,K)
end do
C BOTTOM
C CORIOLIS TERM
UG=0.5*(UN(J,K+1,LE)+UN(J+1,K+1,LE))
UD=0.5*(UN(J,K,LE)+UN(J+1,K,LE))
VAUN=0.5*(HN(J,K+1)*UG+HN(J,K)*UD)
C VAU=0.25*(UO(J,K+1,LE)+UO(J+1,K+1,LE)+UO(J,K,LE)
C 2+UO(J+1,K,LE))
HZP=HZ(LE)*HV(J,K)
HZN=0.5*(HZ(LE)+HZ(LE-1))*HV(J,K)
DSVN=(VO(J,K,LE-1)-VO(J,K,LE))/HZN
C HORIZONTAL FRICTION
HFRVY=PFF*(VO(J,K+1,LE)*H(J,K+1)+VO(J,K-1,LE)*H(J,K)
2-(H(J,K)+H(J,K+1))*VO(J,K,LE))
HFRVX=PLL*(VO(J+1,K,LE)*DHV1+VO(J-1,K,LE)*DHV2
3-(DHV1+DHV2)*VO(J,K,LE))
HFR=HFRVX+HFRVY
C HFR=PLL*HV(J,K)*(VO(J+1,K,LE)+VO(J-1,K,LE)-2.*VO(J,K,LE))+
C 2PFF*HV(J,K)*(VO(J,K+1,LE)+VO(J,K-1,LE)-2.*VO(J,K,LE))
C EVEN BETTER NONLINEAR
C UV
VALX=0.5*(VO(J,K,LE)+VO(J-1,K,LE))
VARX=0.5*(VO(J,K,LE)+VO(J+1,K,LE))
ONXL=0.5*(HU(J,K+1)*UO(J,K+1,LE)+HU(J,K)*UO(J,K,LE))
ONXR=0.5*(HU(J+1,K+1)*UO(J+1,K+1,LE)+HU(J+1,K)*UO(J+1,K,LE))
SUMNONX=PF*(ONXR*VARX-ONXL*VALX)
C VV
VAGY=0.5*(VO(J,K+1,LE)+VO(J,K,LE))
VADY=0.5*(VO(J,K-1,LE)+VO(J,K,LE))
ONYU=0.5*(HV(J,K+1)*VO(J,K+1,LE)+HV(J,K)*VO(J,K,LE))
ONYD=0.5*(HV(J,K-1)*VO(J,K-1,LE)+HV(J,K)*VO(J,K,LE))
SUMNONY=PF*(ONYU*VAGY-ONYD*VADY)
SUMNON=SUMNONX+SUMNONY
CWV
VSU=(VO(J,K,LE-1)*HZ(LE)+VO(J,K,LE)*HZ(LE-1))/
2(HZ(LE)+HZ(LE-1))
VSD=0.
```

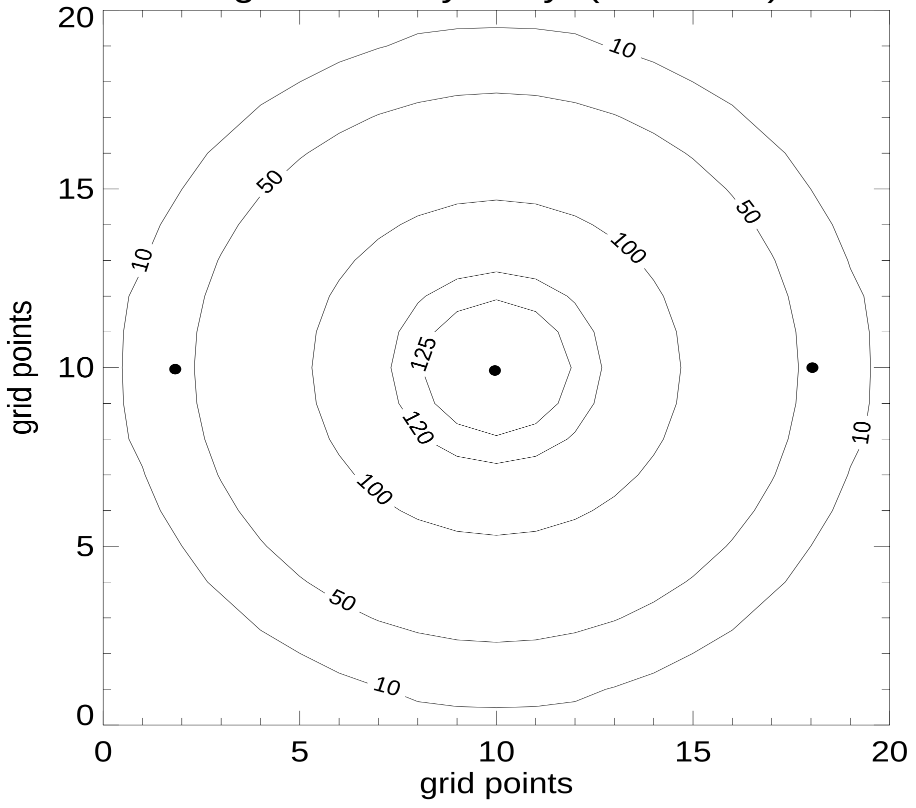
```
WAG=0.5*(W(J,K,LE)+W(J,K+1,LE))
WAD=0.
WV=T*(VSU*WAG-VSD*WAD)/HZ(LE)
SUMNON=SUMNON+WV
VN1(J,K,LE)=VO(J,K,LE)*HV(J,K)-TFC*VAUN-DSLY+HFR-
2SUMNON
VN1(J,K,LE)=VN1(J,K,LE)/HVN(J,K)
6 continue
c VERTICAL FRICTION SPLIT FROM THE REST
do 7 m=1,nlv
c
do 7 j=jsv(m),jev(m)
do 7 k=ksv(m),kev(m)-1
HZPP=HZ(LE)*HVn(J,K)
VAU=0.25*(UO(J,K+1,LE)+UO(J+1,K+1,LE)+UO(J,K,LE)
2+UO(J+1,K,LE))
C LOG OR NOT TO LOG
CD=2.5*ALOG(1.25*HZPP)
CD=1./(CD*CD)
RF1=AMAX1(CD,RF2)
PIERW=SQRT(VO(J,K,Le)*VO(J,K,Le)+VAU*VAU)
TAUBY(J,k)=RF1*PIERW*VO(J,K,Le)
HZP=0.5*(HZ(1)+HZ(2))*hV(j,k)
C(1)=T*ENV(J,K,2)/(HZ(1)*hVN(j,k)*HZP)
D(1)=VN1(J,K,1)
DO L=2,LE-1
HZP=0.5*(HZ(L)+HZ(L+1))*hV(j,k)
HZN=0.5*(HZ(L)+HZ(L-1))*hV(j,k)
A(L)=T*ENV(j,k,L)/(HZ(L)*HVN(j,k)*HZN)
C(L)=T*ENV(J,K,L+1)/(HZ(L)*HVN(j,k)*HZP)
B(L)=1+A(L)+C(L)
D(L)=VN1(J,K,L)
END DO
HZN=0.5*(HZ(LE)+HZ(LE-1))*HV(j,k)
HZP=0.5*(HZ(LE)+HZ(LE+1))*HV(j,k)
D(LE)=VN1(J,K,LE)
A(Le)=T*ENV(j,k,Le)/(HZ(Le)*HVN(j,k)*Hzn)
C(Le)=T*ENV(J,K,Le)/(HZ(Le)*HVN(j,k)*HZP)
B(Le)=1+A(Le)+C(Le)
S(1)=C(1)/(1.+C(1))
E(1)=VN1(J,K,1)+T*TAUSX(J,K)/(HZ(1)*HVN(j,k))
E(1)=E(1)/(1.+C(1))
```

```
DO L=2,LE-1
C VERTICAL FRICTION CONTINUE
DENOM=B(L)-S(L-1)*A(L)
S(L)=C(L)/DENOM
E(L)=(D(L)+A(L)*E(L-1))/DENOM
END DO
A1=A(LE)*E(LE-1)+D(LE)-T*TAUBX(J,K)/(HZ(LE)*HVN(j,k))
A2=1.+A(le)*(1.-S(LE-1))
VN(J,K,LE)=A1/A2
DO L=LE-1,1,-1
VN(J,K,L)=VN(J,K,L+1)*S(L)+E(L)
END DO
7 CONTINUE
C SEA LEVEL
C FIRST AVERAGE VELOCITY ALONG VERTICAL DIRECTION
c print *, un(2,5,1),un(3,5,1)
DO 9 n=1,nlh
do 9 j=js(n)+1,je(n)
do 9 k=ks(n),ke(n)
MM=LE
C++++++++++++++++++++++++++++++++++++
UA(J,k)=0.
DO 10 L=MM,1,-1
UA(J,K)=UA(J,K)+UN(J,K,L)*HZ(L)
10 CONTINUE
9 CONTINUE
c**********************************
do 2000 m=1,nlv
if(ksv(m).eq.kev(m)) go to 2000
c
do 2000 j=jsv(m),jev(m)
do 2000 k=ksv(m),kev(m)-1
MM=LE
VA(J,K)=0.
DO 12 L=MM,1,-1
VA(J,K)=VA(J,K)+VN(J,K,L)*HZ(L)
12 CONTINUE
2000 CONTINUE
C NOW USE CONTINUITY EQUATION. FOR THE CLOSED DOMAIN COMPUTATION STARTS
c LZ=0
do 13 n=1,nlh
do 13 j=js(n),je(n)
do 13 k=ks(n),ke(n)
c93 CONTINUE
ZN(J,K)=ZO(J,K)-PL*(UA(J+1,K)*HUN(J+1,K)-UA(J,K)*HUN(J,K))-
2PF*(VA(J,K)*HVN(J,K)-VA(J,K-1)*HVN(J,K-1))
c if(j.eq.jx)then
c
c print *,ua(j+1),hu(j+1)
c end if
c LZ=LZ+1
c IF(LZ.EQ.2)THEN
c LZ=0
c ZS(J)=ZO(J)
c GO TO 13
c END IF
c ZO(J)=0.25*ZN(J)+0.5*ZO(J)+0.25*ZS(J)
c GO TO 93
13 CONTINUE
c vertical velocity from continuity equation
do 25 n=1,nlh
do 25 j=js(n),je(n)
do 25 k=ks(n),ke(n)
DIFZ=ZN(J,K)-ZO(J,K)
MLEZ=LE
W(J,K,MLEZ+1)=0.
DO 26 L=MLEZ,1,-1
HZT=HZ(L)/T
HZX=HZ(L)/HX
HZY=HZ(L)/HY
DIFU=UN(J+1,K,L)*HUN(J+1,K)-UN(J,K,L)*HUN(J,K)
DIFV=(VN(J,K,L)*HVN(J,K)-VN(J,K-1,L)*HVN(J,K-1))
W(J,K,L)=W(J,K,L+1)-HZX*DIFU-HZY*DIFV-HZT*DIFZ
26 CONTINUE
25 CONTINUE
C CHANGE VERTICAL VELOCITY FROM SIGMA TO Z COORDINATES
IF(NN.EQ.MON) THEN
do 27 n=1,nlh
do 27 j=js(n),je(n)
do 27 k=ks(n),ke(n)
DIFZT=(ZN(J,K)-ZO(J,K))/T
DIFZX=(ZN(J+1,K)-ZN(J-1,K))/TWHX
DIFDX=(H(J+1,K)+ZN(J+1,K)-H(J-1,K)+ZN(J-1,K))/TWHX
```

```
DIFZY=(ZN(J,K+1)-ZN(J,K-1))/TWHY
DIFDY=(H(J,K+1)+ZN(J,K+1)-H(J,K-1)+ZN(J,K-1))/TWHY
Uso=(UN(J+1,K,1)+UN(J,K,1))*0.5
VSO=0.5*(vN(j,k,1)+vN(j,k-1,1))
HZSs=0.
WZ(J,K,1)=W(J,K,1)+DIFZT +Uso*DIFZX+VSO*DIFZY
DO L=2,LE
HZSs=HZSs+HZ(L-1)
Uso=(UN(J+1,K,L)+UN(J,K,L))*HZ(L-1)
1+(UN(J+1,K,L-1)+UN(J,K,L-1))*HZ(L)
Uso=Uso/(HZ(L)+HZ(L-1))
VSO=(VN(J,K,L)+VN(J,K-1,L))*HZ(L-1)
2+(VN(J,K,L-1)+VN(J,K-1,L-1))*HZ(L)
Vso=Vso/(HZ(L)+HZ(L-1))
WZ(J,K,L)=W(J,K,L)+(-HZSs+1)*DIFZT +Uso*(-HZss*DIFDX+DIFZX)
3+Vso*(-HZss*DIFDY+DIFZY)
end do
WZ(J,K,LE+1)=W(J,K,LE+1)
27 CONTINUE
end if
C OLD AND NEW DEPH
DO J=1,JX
DO K=1,KY
HO(J,K)=H(J,K)+ZO(J,K)
HN(J,K)=H(J,K)+ZN(J,K)
END DO
END DO
DO K=1,KY
HU(1,K)=HO(1,K)
HUN(1,K)=HN(1,K)
end do
do j=2,jx
do k=1,ky
HU(j,k)=0.5*(Ho(j-1,k)+ho(j,k))
HUN(j,k)=0.5*(Hn(j-1,k)+hn(j,k))
end do
end do
do j=1,JX
HV(J,KY)=Ho(J,KY)
HVN(J,KY)=Hn(J,KY)
END DO
DO J=1,JX
DO K=1,KY-1
HV(J,K)=0.5*(HO(J,K)+HO(J,K+1))
HVN(J,K)=0.5*(HN(J,K)+HN(J,K+1))
end do
end do
cC CHANGE VARIABLES
DO 15 J=1,JX
DO 15 K=1,KY
ZO(j,k)=ZN(J,K)
DO 15L=1,LE
UO(J,K,L)=UN(J,K,L)
VO(J,K,L)=VN(J,K,L)
15 CONTINUE
if(mod(Nn,10).eq.0) then
C THIS FOR THE VERTICALLY AVERAGE FLOW
c Computing kinetic and potential energy every 10 steps
c and also computing average sea level
ZS=0.
ES=0.
do 130 n=1,nlh
do 130 j=js(n),je(n)
do 130 k=ks(n),ke(n)
vae=0.5*(va(j,k)+va(j,k-1))
uaE=0.5*(ua(j,k)+ua(j+1,k))
enR=(uaE*uaE+vae*vae)/(2.*g*h(j,k))+zn(j,k)*zn(j,k)/2.
es=enR+es
ZS=ZS+ZN(J,K)
130 continue
write(12,*)es
print *,nn,'zn(3)=',zn(3,11),'zn(11)=',zn(11,11)
print *,'zn(19)=',zn(19,11),'ZS=',ZS
write(11,*),n,zn(3,11),zn(11,11),zn(19,11)
c write (18,*)n,zn(3,10),zn(10,10),zn(18,10)
c
end if
IF(NN.LT.MON)GO TO 50
c WRITE SEA LEVEL AND VELOCITIES
WRITE(10,*)((zn(J,k),J=1,JX),k=1,ky)
Do j=1,jx-1
do l=1,le
upX(j,l)=0.5*(UN(J,11,L)+UN(J+1,11,L))
END DO
END DO
WRITE(16,*)((upX(j,L),J=1,JX),L=1,LE)
DO K=2,KY
do l=1,le
upY(K,l)=0.5*(VN(11,K,L)+VN(11,K-1,L))
END DO
END DO
WRITE(17,*)((upy(k,L),k=1,ky),L=1,LE)
Do j=1,jx-1
DO K=1,KY
do l=1,le
upXX(j,K,l)=0.5*(UN(J,K,L)+UN(J+1,K,L))
END DO
END DO
END DO
WRITE(7,*)(((upXX(J,k,L),J=1,JX),k=1,ky),L=1,LE)
Do j=1,jx
DO K=2,KY
do l=1,le
upYY(j,K,l)=0.5*(VN(J,K,L)+VN(J,K-1,L))
END DO
END DO
END DO
WRITE(8,*)(((upYY(J,k,L),J=1,JX),k=1,ky),L=1,LE)
DO J=1,JX
DO L=1,LE-1
WPX(J,L)=0.5*(WZ(J,11,L)+WZ(J,11,L+1))
END DO
wPX(j,le)=0.5*(wz(j,11,le))
END DO
write(15,*)((wPX(J,L),J=1,JX),L=1,LE)
DO K=1,KY
DO L=1,LE-1
WPY(K,L)=0.5*(WZ(11,K,L)+WZ(11,K,L+1))
END DO
WPY(K,LE)=0.5*WZ(11,K,LE)
END DO
write(9,*)((wPY(J,L),J=1,JX),L=1,LE)
c222 continue
stop
end
```

The above program has been used to compute wind-driven circulation in the circular water body of an diameter close to 200 km. The bathymetry is given in Fig.III.7. The wind is directed along the x axis toward the positive values. Wind stress is constant and equal to 1 CGS. Temporal changes of the sea level at the three points located along diameter of the water body, the are shown in Fig.III.8. As in rectangular water body, one may suspect that the periodical oscillations of the sea level are related to the own oscillations. These oscillations are more complicated than the ones given in Fig.III.4 in the rectangular water body. The signal is time modulated thus there is no unique dominance of a single period. Surface current is depicted in Fig.III.9. Dotted lines in this figure denote the magnitude of the current. One can effortlessly check here a few assumptions related to the vertical eddy viscosity and answer such questions as: How does current direction and magnitude depend on magnitude of the eddy viscosity? How big change into current distribution will introduce a variable eddy viscosity? The distribution of current in the vertical cross-section along the diameter that is parallel to the x axis is shown in Fig.III.10. The magnitude of the current is given by continuous lines. In this figure arrows show vertical slope as well. This occurs because in the plotting routine the vertical component of velocity is multiplied by 1000. One should also remember that the plotting is done in the rectangular system of coordinate.

Fig.III.7: Bathymetry (in meters)



Bullets show points where sea level was recorded for the Fig. III.8

FigIII.8: SEA LEVEL

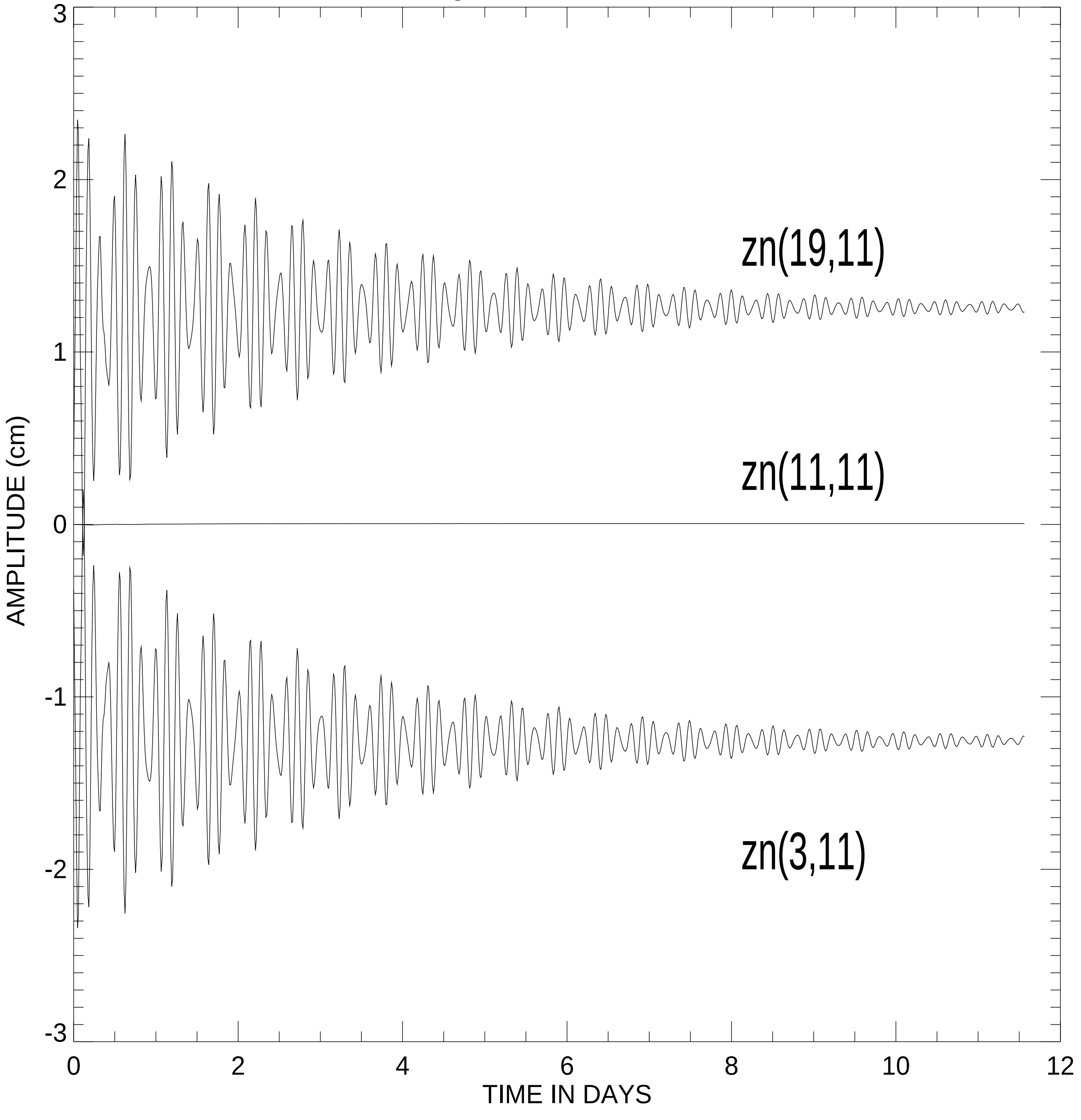


Fig.III.9: Surface Current

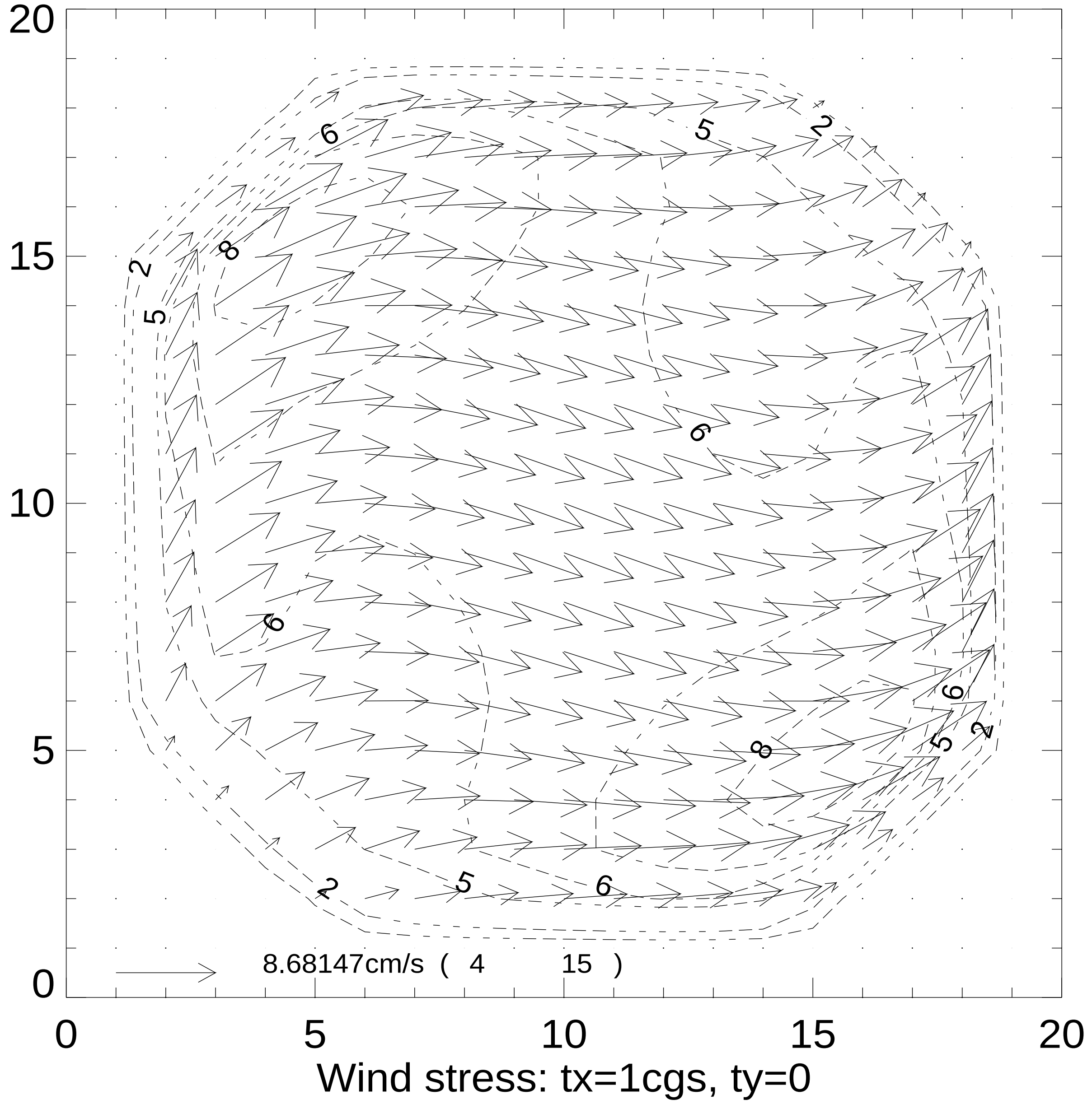
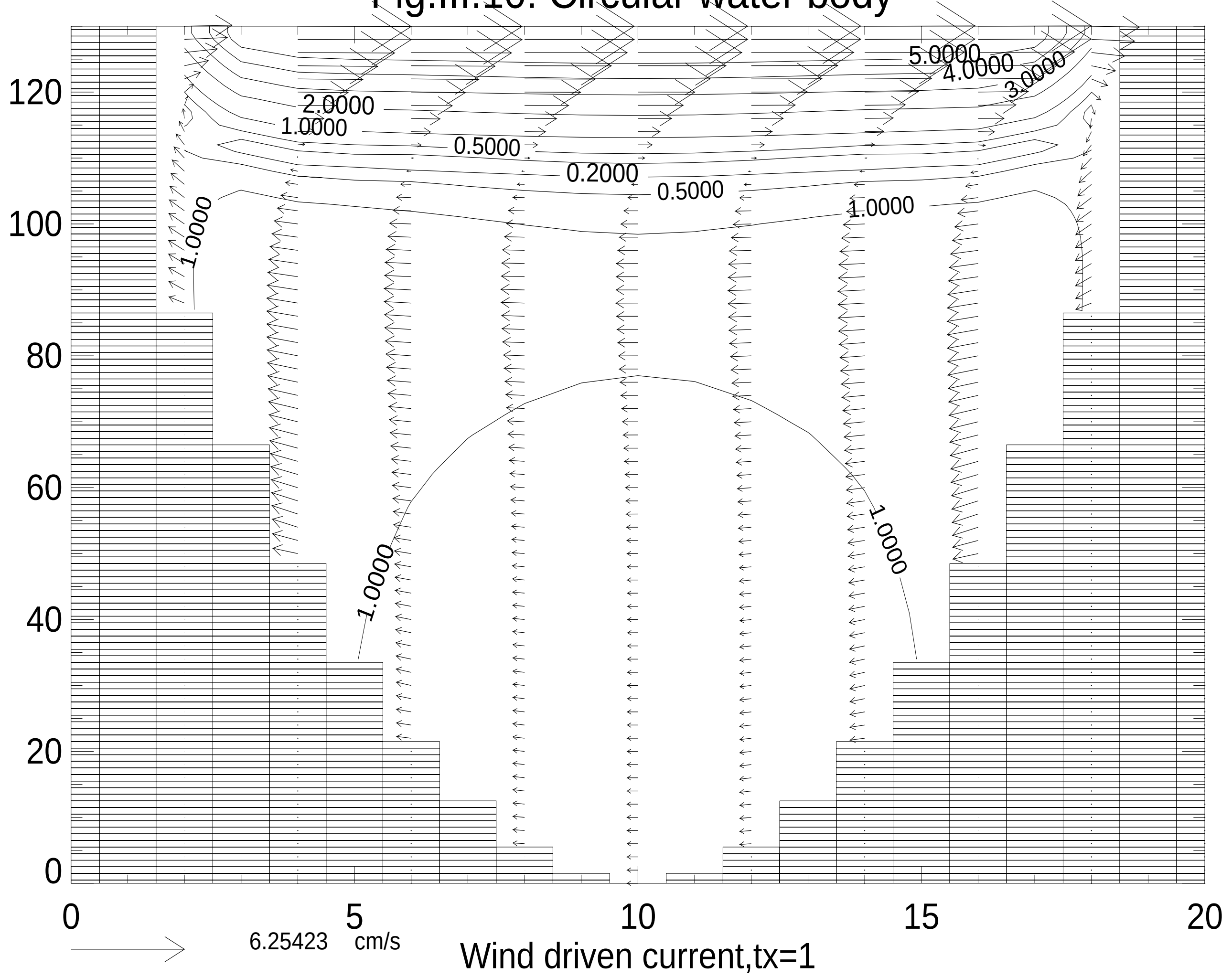


Fig.III.10: Circular water body



Problem No. 4, 3D- baroclinic model, sigma coordinate

In this problem the density stratification will be included. The density will cause additional gradients of pressure in the equations of motion. These gradients along the x and y directions can be expressed in σ coordinate as,

$$-\frac{gD^2}{\rho}\int_\sigma^0 (\frac{\partial\rho}{\partial x} - \frac{Q_x}{D}\frac{\partial\rho}{\partial\sigma})d\sigma = -\frac{gD^2}{\rho}\frac{\partial}{\partial x}\int_\sigma^0 \rho d\sigma + \frac{gD}{\rho}\frac{\partial D}{\partial x}\int_\sigma^0 \sigma\frac{\partial\rho}{\partial\sigma}d\sigma$$

$$-\frac{gD^2}{\rho}\int_\sigma^0 (\frac{\partial\rho}{\partial y} - \frac{Q_y}{D}\frac{\partial\rho}{\partial\sigma})d\sigma = -\frac{gD^2}{\rho}\frac{\partial}{\partial y}\int_\sigma^0 \rho d\sigma + \frac{gD}{\rho}\frac{\partial D}{\partial y}\int_\sigma^0 \sigma\frac{\partial\rho}{\partial\sigma}d\sigma$$

The full equations of motion now reads

$$\frac{\partial}{\partial t}(uD) + \frac{\partial}{\partial x}(u^2D) + \frac{\partial}{\partial y}(uvD) + \frac{\partial}{\partial\sigma}(Wu) - fDv$$
$$= -gD\frac{\partial\zeta}{\partial x} + \frac{1}{D}\frac{\partial}{\partial\sigma}(N_z\frac{\partial u}{\partial\sigma}) + +N_h[\frac{\partial}{\partial x}D\frac{\partial u}{\partial x} + \frac{\partial}{\partial y}D\frac{\partial u}{\partial y}]$$
$$-\frac{gD^2}{\rho}\frac{\partial}{\partial x}\int_\sigma^0 \rho d\sigma + \frac{gD}{\rho}\frac{\partial D}{\partial x}\int_\sigma^0 \sigma\frac{\partial\rho}{\partial\sigma}d\sigma \qquad \text{(III.44)}$$

$$\frac{\partial}{\partial t}(vD) + \frac{\partial}{\partial x}(uvD) + \frac{\partial}{\partial y}(v^2D) + \frac{\partial}{\partial\sigma}(Wv) + fDu$$
$$= -gD\frac{\partial\zeta}{\partial y} + \frac{1}{D}\frac{\partial}{\partial\sigma}(N_z\frac{\partial v}{\partial\sigma}) + N_h[\frac{\partial}{\partial x}D\frac{\partial v}{\partial x} + \frac{\partial}{\partial y}D\frac{\partial v}{\partial y}]$$
$$-\frac{gD^2}{\rho}\frac{\partial}{\partial y}\int_\sigma^0 \rho d\sigma + \frac{gD}{\rho}\frac{\partial D}{\partial y}\int_\sigma^0 \sigma\frac{\partial\rho}{\partial\sigma}d\sigma \qquad \text{(III.45)}$$

In the above equations the density distribution is assumed to be given either from observations or from calculations. Because density is function of salinity and temperature one needs to introduce additional equations which express conservation of mass and heat. Therefore, to the above set of dependent variables the salinity and temperature ought to be added. Sea water density is calculated accordingly to the formulas given by Friedrich, H. J and S. Levitus in J. Phys. Ocean., 2, 514-517, 1972. Salt and heat conservation equations are expressed as,

$$\frac{\partial}{\partial t}(DS) + \frac{\partial}{\partial x}(uDS) + \frac{\partial}{\partial y}(vDS) + \frac{\partial}{\partial\sigma}(WS) =$$

$$\frac{1}{D}\frac{\partial}{\partial\sigma}(D_z\frac{\partial S}{\partial\sigma}) + D_h\frac{\partial}{\partial x}D\frac{\partial S}{\partial x} + D_h\frac{\partial}{\partial y}D\frac{\partial S}{\partial y} \tag{III.46}$$

$$\frac{\partial}{\partial t}(D\theta) + \frac{\partial}{\partial x}(uD\theta) + \frac{\partial}{\partial y}(vD\theta) + \frac{\partial}{\partial\sigma}(W\theta) =$$

$$\frac{1}{D}\frac{\partial}{\partial\sigma}(D_z\frac{\partial \theta}{\partial\sigma}) + D_h\frac{\partial}{\partial x}D\frac{\partial \theta}{\partial x} + D_h\frac{\partial}{\partial y}D\frac{\partial \theta}{\partial y} \tag{III.47}$$

It is evident from these equations that they are closely related to equations of motion. Advective and convective terms in conservation equation are governed by velocity components that are calculated through equations of motion, on the other hand salinity and temperature define the density distribution and the baroclinic pressure which, in turn, drives currents. Calculations of salinity and currents, must therefore, proceed in the same time loop. We start by constructing numerical form for pressure terms in equations of motion. Consider term

$$\int_\sigma^0 \rho d\sigma$$

which occurs along both x and y directions, and if σ will be changed to z, the expression is similar to the term defined on page 224 of NM. From the free surface to the first pressure point, the distance is $0.5 * hz_1$ and the pressure at the first grid point benith the free surface is $prh_{j,k,1} = 0.5 * hz_1 * \rho_{j,k,1}$. The above integral can, therefore, be defined as a sum along the vertical direction by calculating pressure at any depth as

$$prh_{j,k,l} = prh_{j,k,l-1} + 0.5 * (hz_{l-1} * \rho_{j,k,l-1} + hz_l * \rho_{j,k,l}) \tag{III.48}$$

Moreover, the pressure is defined in the pressure (salinity, temperature) points which are located between velocity grid points, therefore it is easy to take derivatives along x and y directions for the u and v components in the equations of motion.

The second pressure term is more complicated, it is resulted from coordinate's transformation,

$$\int_\sigma^0 \sigma\frac{\partial\rho}{\partial\sigma}d\sigma$$

It can be changed to

$$\int_\sigma^0 \sigma d\rho$$

as long as one remembers to take differences along the vertical (σ) direction. Labeling this pressure as prs, its value in the first point at the distance $0.5 * hz_1$ from the free surface can be assumed as zero. (If it is not zero, it can be estimated by extrapolation from the points located below the free surface.) The pressure at any depth can be calculated, as the following sum,

$$prs_{j,k,l} = prs_{j,k,l-1} + \sigma * d\rho \tag{III.49}$$

Here $\sigma = \sigma_{l-1} + 0.5 * (hz_{l-1} + hz_l)$ and $d\rho = \rho_{j,k,l-1} - \rho_{j,k,l}$. Again, it is easy to take derivatives along x and y directions to obtain gradients of pressure at the u and v points for the equations of motion. One very important point of the above calculation is density. The variability of density is very small compared to the average value of density. In our calculations we need to know only derivatives, which are functions of the variable part of the density. Moreover, the above integrals can be a source of the error because the average values are enhanced through the vertical integration. Thus instead of full density the value $\rho - 1$ is considered. Our calculations are done in CGS units, if mks units are implemented the new density will be $\rho - 1000$. Such approach alleviates the error generating problem.

Let's consider now the way of constructing a numerical solution for the temperature or salinity. Here we consider salinity only. The approach for the temperature is analogous. Several terms in eq. III.46 are easily duplicated through our previous construction of numerical solution for the equation of motion. First thing to remember: equation is solved in the salinity (pressure) points. The time derivative of the salinity is

$$\frac{(SD)^{m*} - (SD)^m}{T} = \frac{HN_{j,k} * S^{m+1}_{j,k,l} - HO_{j,k} * S^m_{j,k,l}}{T} \tag{III.50}$$

Here $m*$ denotes intermediate substep. In this solution the vertical diffusion is going to be split from the remaining terms as in equations of motion the vertical friction was split from the other terms.

Advective terms along x direction ($\frac{\partial}{\partial x}(uDS)$) can be considered as inflowing and outflowing flux of salinity through the walls of an cubicle (see Fig.I.9 from this workbook). The flux through the right wals is

$$FLXR = 0.5 * (S_{j,k,l} + S_{j+1,k,l}) * u^m_{j+1,k,l} * HU_{j+1,k}$$

and the flux through the left wall is

$$FLXL = 0.5 * (S_{j,k,l} + S_{j-1,k,l}) * u^m_{j,k,l} * HU_{j,k}$$

Therefore,

$$\frac{\partial}{\partial x}(uDS) \simeq \frac{FLXR - FLXL}{hx} \tag{III.51}$$

In the similar way the advective term along the y direction $\frac{\partial}{\partial y}(vDS)$ is written

$$FLYR = 0.5 * (S_{j,k,l} + S_{j,k+1,l}) * V^m_{j,k,l} * HV_{j,k}$$

and the flux through the left wall is

$$FLYL = 0.5 * (S_{j,k,l} + S_{j,k-1,l}) * V^m_{j,k-1,l} * HV_{j,k-1}$$

Therefore,

$$\frac{\partial}{\partial y}(vDS) \simeq \frac{FLYR - FLYL}{hy} \tag{III.52}$$

The convective motion caused by the vertical velocity W is quite similar to the above considerations. Here the flux through the upper and lower surfaces is considered

$$FLZU = W_{j,k,l} * (S_{j,k,l} + S_{j,k,l-1}) * 0.5$$
$$FLZD = W_{j,k,l+1} * (S_{j,k,l+1} + S_{j,k,l}) * 0.5$$

Thus the numerical form for the convective term reads

$$\frac{\partial}{\partial \sigma}(WS) \simeq \frac{FLZU - FLZD}{hz_l} \tag{III.53}$$

The second order approximation works well at the interior of the fluid but not in proximity to the surface and the bottom. At the boundaries a directional flux is required that transports salt from the surface and bottom regions toward the interior. Upon consulting figures 4.1 and 4.7 from NM it can be seen that both at the bottom and at the surface convective flux is zero. Only velocity which is directed from the inside regions of the fluid can bring change in the boundary value of salinity. Thus the directional velocity at the second grid point from the boundaries needs to be formulated and the fluxes will follow. At the free surface the convective flux is

$$FLZU = 0 \qquad FLZD = wpd * S^m_{j,k,2} + wnd * S^m_{j,k,1} \tag{III.54a}$$

Here:

$$wpd = 0.5 * (w^m_{j,k,2} + |w^m_{j,k,2}|) \qquad wnd = 0.5 * (w^m_{j,k,2} - |w^m_{j,k,2}|)$$

Convective flux at the bottom is defined in similar manner,

$$FLZU = wpu * S^m_{j,k,LE} + wnu * S^m_{j,k,LE-1} \qquad FLZD = 0 \tag{III.54b}$$

Here:

$$wpu = 0.5 * (w^m_{j,k,LE} + |w^m_{j,k,LE}|) \qquad wnu = 0.5 * (w^m_{j,k,LE} - |w^m_{j,k,LE}|)$$

On the first substep of the time marching algorithm, the horizontal diffusion term is computed as well. The following expression needs to be cast into numerical form.

$$HDIF = D_h \frac{\partial}{\partial x} D^m \frac{\partial S^m}{\partial x} + D_h \frac{\partial}{\partial y} D^m \frac{\partial S^m}{\partial y} \tag{III.55}$$

The horizontal diffusion is based on the horizontal diffusion fluxes. To fulfill noflux boundary condition at the shore line instead of diffusion coefficient D_h we introduce two coefficients D_{hu} and D_{hv}. Such coefficients will be equal to the D_h inside computational domain and 0 outside computational domain. This approach is going to work in the following way for the fluxes along x direction. The flux to the right from the point j, k, l is

$$HDFX1 = HU_{j+1,k} D_{hu,j+1,k} * (S_{j+1,k,l} - S_{j,k,l})/h_x \tag{III.56a}$$

and the flux to the left from point j, k, l is

$$HDFX2 = HU_{j,k} D_{hu,j,k} * (S_{j,k,l} - S_{j-1,k,l})/h_x \tag{III.56b}$$

The above fluxes, coefficients D_{hu}, and depth HU are set in the u velocity points. At the shore line locations the normal velocity to the shore is zero and at the same points the horizontal flux will be set to zero through the choice of values of D_{hu}.

Similar approach can be implemented along the y direction

$$HDFY1 = HV_{j,k} D_{hv,j,k} * (S_{j,k+1,l} - S_{j,k,l})/h_y \tag{III.57a}$$

$$HDFY2 = HV_{j,k-1} D_{hv,j,k-1} * (S_{j,k,l} - S_{j,k-1,l})/h_y \tag{III.57b}$$

Combining III.56 and III.57 we arrive at the following numerical form for the horizontal diffusion

$$D_h \frac{\partial}{\partial x} D^m \frac{\partial S^m}{\partial x} + D_h \frac{\partial}{\partial y} D^m \frac{\partial S^m}{\partial y} \simeq$$

$$\frac{HDFX1 - HDFX2}{h_x} + \frac{HDFY1 - HDFY2}{h_y} \tag{III.58}$$

Solution of the diffusive term along the vertical direction will proceed in analogous way as the vertical friction in the equations of motion.

$$\frac{\partial}{\partial \sigma}(D_z \frac{\partial S}{\partial \sigma})^{m+1} \simeq \frac{1}{h_{z,l}}[D_{z,l} \frac{S^{m+1}_{j,k,l-1} - S^{m+1}_{j,k,l}}{0.5 * (h_{z,l} + h_{z,l-1})} - D_{z,l+1} \frac{S^{m+1}_{j,k,l} - S^{m+1}_{j,k,l+1}}{0.5 * (h_{z,l} + h_{z,l+1})}] \tag{III.59}$$

Combining above expression with the time derivative, we arrive at the formula,

$$\frac{HN_{j,k} * S^{m+1}_{j,k,l} - HO_{j,k} * S^{m*}_{j,k,l}}{T} =$$

$$\frac{1}{h_{z,l}}[D_{z,l}\frac{S^{m+1}_{j,k,l-1} - S^{m+1}_{j,k,l}}{0.5 * (h_{z,l} + h_{z,l-1})} - D_{z,l+1}\frac{S^{m+1}_{j,k,l} - S^{m+1}_{j,k,l+1}}{0.5 * (h_{z,l} + h_{z,l+1})}] \qquad \text{(III.60)}$$

This equation can be organized into a three-point algorithm and solved as boundary value problem by the line inversion method (Appendix 1, from NM). The boundary conditions for this equation is usually taken as noflux both at the bottom and the surface. The heat exchange boundary condition can be easily modified to bring a heat flux from the atmosphere.

In the baroclinic current computation important role play the vertical convective term. When hydrostatic instability occurs in the fluid, this term transfers unstable parcel of fluid to a new location. Implementation of the convective mixing in the numerical models is usually achieved either through mechanical mixing or by large values of the vertical diffusion coefficient.

This procedure is turn on during computation until instability vanishes. We have implemented mechanical mixing. The explanation of this code is given on page 236 of NM.

Not being overly philosophical I consider a numerical rendition of convective processes as one of the biggest (and subtle) source of errors. One has to make effort for implementation of the highest order of approximation, preferably flux-corrected method. The method also should conserve salinity (or heat) to a high degree of approximation. The second order approximation which we applied in the beneath FORTRAN program is not very good, it is enough to glance at the Fig. 4.5, page 237 from NM to understand that dispersive wiggles in the second order method obfuscate physics of the process.

Beneath is given program d_seasie.f which computes density driven currents from density and salinity prescribed in the program. The computation proceeds in the circular shaped fluid body with the parabolic depth. The process slowly arrives to equilibrium. The situation is described after 12 day of process. Small vertical eddy viscosity and diffusivity (5cm^2/s) mix fluid so slowly that advective and convective terms were able to adjust pressure field due to density distribution to the depth distribution. A well developed circular vortex flow results from this adjustment.

```
cc program d_seasi.f Wind-driven motion, implicit vert. friction
c sigma coordinate c density is included
c This is 3d motion in circular water body with parabolic depth
PARAMETER (JX=21,KY=21,LE=20,NLH=17,NLV=17)
REAL UN(JX,KY,LE),UO(JX,KY,LE),UN1(JX,KY,LE)
REAL ZN(JX,KY),ZO(JX,KY),H(JX,ky),ENU(JX,KY,LE)
REAL HZ(LE),TAUSX(JX,KY),TAUBX(JX,KY),UA(JX,KY)
REAL W(JX,KY,LE+1),hu(jx,ky),upX(jx,le),wpX(jx,le)
REAL upY(KY,le),wpY(KY,le),upxx(JX,KY,LE),UPYY(JX,KY,LE)
real uai(jx,KY),HUN(JX,ky),HN(JX,ky)
REAL VAI(JX,KY),VO(JX,KY,LE),VN(JX,KY,LE),ENV(JX,KY,LE)
REAL DV(JX,ky,LE),TAUSY(JX,KY),SN(JX,KY,LE)
REAL TAUBY(JX,KY),HO(JX,KY),VA(JX,KY),hv(jx,ky),hvn(jx,ky)
REAL VN1(JX,KY,LE),WZ(JX,KY,LE+1),STV(JX,KY,LE)
REAL DHU(JX,KY),DHV(JX,KY),SO(JX,KY,LE),ST(JX,KY,LE)
REAL RHO(JX,KY,LE),PRH(JX,KY,LE),PRS(JX,KY,LE)
REAL TO(JX,KY,LE),TT(JX,KY,LE),TN(JX,KY,LE),TTV(JX,KY,LE)
real a(le+1),b(LE+1),c(LE+1),EN(JX,KY,LE)
real d(LE+1),e(LE+1),s(LE+1)
REAL ALP(7)/-7.2169E-2,4.9762E-2,8.056E-1,-7.5911E-3,
1-3.0063E-3,3.5187E-5,3.7297E-5/
integer js(nlh),je(nlh),ks(nlh),ke(nlh)
integer jsv(nlv),jev(nlv),ksv(nlv),kev(nlv)
open (unit=4,file='parab.dat',status='unknown')
open (unit=3,file='parab.ind',status='unknown')
open (unit=9,file='wpy.dat',status='unknown')
open (unit=10,file='z.dat',status='unknown')
open (unit=11,file='momaxz1.dat',status='unknown')
open (unit=12,file='eni.dat',status='unknown')
open (unit=15,file='wpx.dat',status='unknown')
open (unit=16,file='upx.dat',status='unknown')
open (unit=17,file='upy.dat',status='unknown')
open (unit=20,file='depthx.dat',status='unknown')
open (unit=21,file='depthy.dat',status='unknown')
open (unit=7,file='UPXX.dat',status='unknown')
open (unit=8,file='UPYY.dat',status='unknown')
open (unit=18,file='s.dat',status='unknown')
open (unit=19,file='s1.dat',status='unknown')
open (unit=22,file='t.dat',status='unknown')
open (unit=23,file='t1.dat',status='unknown')
open (unit=24,file='rho.dat',status='unknown')
open (unit=25,file='rho1.dat',status='unknown')
read(4,*)((h(j,k),j=1,JX),k=1,ky)
do j=1,jx
do K=1,KY
IF(H(J,K).LT.500.)H(J,K)=500.
hn(j,k)=h(j,k)
```

```
ho(j,k)=h(j,k)
END DO
END DO
c Index : horizontal line, vertical
c write (6,101) nlh
c 101 format(4i6)
do i=1,nlh
read (3,100) ks(i),ke(i),js(i),je(i)
100 format(4i4)
enddo
c
c write (6,101) nlv
do i=1,nlv
read (3,100) jsv(i),jev(i),ksv(i),kev(i)
enddo
c
close (3)
write(20,*)(h(j,11),j=1,jx)
write(21,*)(h(11,k),k=1,ky)
C PRINT*,(ALP(I),I=1,7)
DO K=1,KY
HU(1,K)=HO(1,K)
HUN(1,K)=HN(1,K)
end do
do j=2,jx
do k=1,ky
HU(j,k)=0.5*(Ho(j-1,k)+ho(j,k))
HUN(j,k)=0.5*(Hn(j-1,k)+hn(j,k))
end do
end do
do j=1,JX
HV(J,KY)=Ho(J,KY)
HVN(J,KY)=Hn(J,KY)
END DO
DO J=1,JX
DO K=1,KY-1
HV(J,K)=0.5*(HO(J,K)+HO(J,K+1))
HVN(J,K)=0.5*(HN(J,K)+HN(J,K+1))
end do
end do
DO J=1,JX
DO K=1,KY
```

```
ua(j,K)=0.
va(j,k)=0.
zo(j,K)=0.
zn(j,k)=0.
DHU(J,K)=0.
DHV(J,K)=0.
end do
END DO
do j=1,jx
DO K=1,KY
do l=1,le
un(j,K,l)=0.
un1(j,k,l)=0.
uo(j,K,l)=0.
vn(J,K,L)=0.
VO(J,K,L)=0.
VN1(J,K,L)=0.
w(j,k,l)=0.
wZ(j,k,l)=0.
ENU(J,K,L)=0.
ENV(J,K,L)=0.
SO(J,K,L)=0.
ST(J,K,L)=0.
STV(J,K,L)=0.
SN(J,K,L)=0.
PRS(J,K,L)=0.
PRH(J,K,L)=0.
RHO(J,K,L)=0.
TO(J,K,L)=0.
TT(J,K,L)=0.
TTV(J,K,L)=0.
TN(J,K,L)=0.
end do
end do
END DO
c INITIAL SALINITY
c THE STUFF BELOW IS FOR INITIAL SALINITY
C AND TEMPERATURE IT CAN BE USED FOR PLOTTING THROUGH
C PROGRAM PREPARE_SAL.F
do j=1,jx
DO K=1,KY
do l=1,le
```

```
SO(J,K,L)=FLOAT(L)*0.345+19.655+2.5*float(j)/float(jx)
2+2.5*float(K)/float(KY)
TO(J,K,L)=15.+1.*FLOAT(LE)/(5.*FLOAT(L))+2.*float(j)/float(jx)
*+2.*float(K)/float(KY)
end do
end do
END DO
DO L=1,LE
write(19,*)(so(j,11,L),j=1,jx)
write(23,*)(TO(j,11,L),j=1,jx)
END DO
C GO TO 222
C SET variable SPACE STEP ALONG VERTICAL DIRECTION
DO L=1,le
HZ(L)=0.05
END DO
C ALL COEFFICIENTS
c=====================================
c Time step
T = 100.
FC=1.458E-4
G=981.
c SPACE STEPS ALONG X AND Y
HX=10.E5
HY=10.E5
RF1=2.6E-3
RF2=2.6E-3
c HORIZONTAL EDDY VISCOSITY
AH=1.0e7
c HORIZONTAL EDDY DIFFUSIVITY ALONG X DIRECTION
do n=1,nlh
do j=js(n)+1,je(n)
do k=ks(n),ke(n)
DHU(J,K)=(1.0e7)*T/(HX*HX)
END DO
END DO
end do
do m=1,nlv
c
do j=jsv(m),jev(m)
do k=ksv(m),kev(m)-1
DHV(J,K)=(1.0e7)*T/(HY*HY)
```

```
END DO
END DO
end do
C COMBINATIONAL PARAMETERS
TRF=T*RF1
PL=T/HX
GPL=G*PL
PLL=AH*T/(HX*HX)
PF=T/HY
GPF=G*PF
PFF=AH*T/(HY*HY)
TFC=T*FC
TWHX=2.*hx ! for vertical velocity
TWHY=2.*hY ! for vertical velocity
C START TIME LOOP INDEX N RUNS UNTIL N=MON
c**********************************
c MON=86400
mon=10000
c mon=10
nn=0
C8888888888888888888888888888888888
50 Nn=Nn+1
C SOME COEFFICIENTS ARE VARIABLE
C SET WIND FOR INITIAL EXPERIMENT
DO J=1,JX
DO K=1,KY
TAUSX(J,K)=0.
TAUSY(J,K)=0.
END DO
END DO
C DO NOT INTRODUCE STRONG WIND UBRUPTLY
c if (n.lt.1000)then
c DO J=1,JX
c DO K=1,KY
c TAUSX(J,K)=float(nN)/1000.
c TAUSY(J,K)=0.
c END DO
c END DO
c end if
C EDDY VISCOSITY IS CONSTANT
DO 33J=1,JX
do 33 k=1,ky
```

```
DO 33 L=1,Le
EN(J,k,L)=5.
33 CONTINUE
DO J=1,JX
do k=1,ky
DO L=1,LE
ENU(J,k,L)=0.5*(EN(J,K,L)+EN(J,K-1,L))
ENV(J,K,L)=0.5*(EN(J,K,L)+EN(J+1,K,L))
DV(J,K,L)=5.
END DO
END DO
END DO
c IN SPACE COEFFICIENT OF EDDY VISCOSITY SHOULD BE
C LOCATED IN Z (OR PRESSURE) POINTS
C COMPUTATION OF U VELOCITY, closed boundary, velocity is zero at
C J=1 AND J=JX.
c - + + - + - +
c 1 jx jx+1
c left right
C VELOCITY IN THE AIR
C go to 225
DO 499 n=1,nlh
do 499j=js(n)+1,je(n)
do 499k=ks(n),ke(n)
UAI(J,K)=UO(J,K,1)+HZ(1)*hu(j,K)*TAUSX(J,K)/ENU(J,K,1)
499 continue
DO 4 n=1,nlh
do 4 j=js(n)+1,je(n)
do 4 k=ks(n),ke(n)
C THIS GOES INTO HORIZONTAL FRICTION
DHU1=0.5*(HU(J,K)+HU(J,K+1))
DHU2=0.5*(HU(J,K)+HU(J,K-1))
C PRESSURE DUE TO DENSITY
DPRH=HU(J,K)*HU(J,K)*GPL*(PRH(J,K,1)-PRH(J-1,K,1))
PRSS=0.5*(PRS(J,K,1)+PRS(J-1,K,1))
DPRS=GPL*HU(J,K)*(H(J,K)-H(J-1,K))*PRSS
C SURFACE CONDITION
C CORIOLIS
VR=0.5*(VO(J,K,1)+VO(J,K-1,1))
VL=0.5*(VO(J-1,K,1)+VO(J-1,K-1,1))
UAV=0.5*(HO(J,K)*VR+HO(J-1,K)*VL)
DSL=ZO(J,K)-ZO(J-1,K)
```

```
DSLX=DSL*GPL*HU(J,K)
HZP=0.5*(HZ(1)+HZ(2))*hu(j,K)
DSU=(UO(J,K,1)-UO(J,K,2))/HZP
c VFRS=(TAUSX(J,K)-ENU(J,K,2)*DSU)/HZ(1)
C HORIZONTAL FRICTION
HFRUX=PLL*(UO(J+1,K,1)*H(J,K)+UO(J-1,K,1)*H(J-1,K)
2-(H(J,K)+H(J-1,K))*UO(J,K,1))
HFRUY=PFF*(UO(J,K+1,1)*DHU1+UO(J,K-1,1)*DHU2
3-(DHU1+DHU2)*UO(J,K,1))
HFR=HFRUX+HFRUY
C EVEN BETTER IMPROVED NONLINEAR TERMX
c uw
USN=0.5*(UAI(J,K)+UO(J,K,1))
USP=(UO(J,K,1)*HZ(2)+UO(J,K,2)*HZ(1))/(HZ(1)+HZ(2))
waxn=0.5*(W(J-1,K,1)+W(J,K,1))
WAXP=0.5*(W(J-1,K,2)+W(J,K,2))
UW=T*(WAXN*USN-WAXP*USP)/HZ(1)
C EVEN BETTER IMPROVED NONLINEAR TERMX
c uv uu
UAF=0.5*(UO(J,K,1)+UO(J+1,K,1))
UAB=0.5*(UO(J,K,1)+UO(J-1,K,1))
ONXF=0.5*(UO(J,K,1)*HU(J,K)+UO(J+1,K,1)*HU(J+1,K))
ONXB=0.5*(UO(J,K,1)*HU(J,K)+UO(J-1,K,1)*HU(J-1,K))
SUMNONX=PL*(ONXF*UAF-ONXB*UAB)
C UV
UAU=0.5*(UO(J,K,1)+UO(J,K+1,1))
UAD=0.5*(UO(J,K-1,1)+UO(J,K,1))
ONYU=0.5*(VO(J,K,1)*HV(J,K)+VO(J-1,K,1)*HV(J-1,K))
ONYD=0.5*(VO(J,K-1,1)*HV(J,K-1)+VO(J-1,K-1,1)*HV(J-1,K-1))
SUMNONY=PF*(UAU*ONYU-UAD*ONYD)
SUMNON=SUMNONX+SUMNONY+UW
UN1(J,K,1)=UO(J,K,1)*HU(J,K)-DSLX+TFC*UAV+HFR-SUMNON
2-DPRH+DPRS
UN1(J,K,1)=UN1(J,K,1)/HUN(J,K)
LMM=LE
DO 5 L=2,LMM-1
C PRESSURE DUE TO DENSITY
DPRH=HU(J,K)*HU(J,K)*GPL*(PRH(J,K,L)-PRH(J-1,K,L))
PRSS=0.5*(PRS(J,K,L)+PRS(J-1,K,L))
DPRS=GPL*HU(J,K)*(H(J,K)-H(J-1,K))*PRSS
HZP=0.5*(HZ(L)+HZ(L+1))*HU(J,K)
HZN=0.5*(HZ(L)+HZ(L-1))*HU(J,K)
```

```
DSUN=(UO(J,K,L-1)-UO(J,K,L))/HZN
DSUP=(UO(J,K,L)-UO(J,K,L+1))/HZP
c VFR=(ENU(J,K,L)*DSUN-ENU(J,K,L+1)*DSUP)/HZ(L)
C HORIZONTAL FRICTION
HFRUX=PLL*(UO(J+1,K,L)*H(J,K)+UO(J-1,K,L)*H(J-1,K)
2-UO(J,K,L)*(H(J,K)+H(J-1,K)))
HFRUY=PFF*(UO(J,K+1,L)*DHU1+UO(J,K-1,L)*DHU2
3-(DHU1+DHU2)*UO(J,K,L))
HFR=HFRUX+HFRUY
C CORIOLIS
VR=0.5*(VO(J,K,L)+VO(J,K-1,L))
VL=0.5*(VO(J-1,K,L)+VO(J-1,K-1,L))
UAV=0.5*(HO(J,K)*VR+HO(J-1,K)*VL)
C EVEN BETTER IMPROVED NONLINEAR TERMX
UAF=0.5*(UO(J,K,L)+UO(J+1,K,L))
UAB=0.5*(UO(J,K,L)+UO(J-1,K,L))
ONXF=0.5*(UO(J,K,L)*HU(J,K)+UO(J+1,K,L)*HU(J+1,K))
ONXB=0.5*(UO(J,K,L)*HU(J,K)+UO(J-1,K,L)*HU(J-1,K))
SUMNONX=PL*(ONXF*UAF-ONXB*UAB)
C UV
UAU=0.5*(UO(J,K,L)+UO(J,K+1,L))
UAD=0.5*(UO(J,K-1,L)+UO(J,K,L))
ONYU=0.5*(VO(J,K,L)*HV(J,K)+VO(J-1,K,L)*HV(J-1,K))
ONYD=0.5*(VO(J,K-1,L)*HV(J,K-1)+VO(J-1,K-1,L)*HV(J-1,K-1))
SUMNONY=PF*(UAU*ONYU-UAD*ONYD)
SUMNON=SUMNONX+SUMNONY
c vert deriv.
waxn=0.5*(W(J-1,K,L)+W(J,K,L))
WAXP=0.5*(W(J-1,K,L+1)+W(J,K,L+1))
USN=(UO(J,K,L-1)*HZ(L)+UO(J,K,L)*HZ(L-1))/(HZ(L)+HZ(L-1))
USP=(UO(J,K,L)*HZ(L+1)+UO(J,K,L+1)*HZ(L))/(HZ(L)+HZ(L+1))
UW=T*(WAXN*USN-WAXP*USP)/HZ(L)
SUMNON=SUMNON+UW
UN1(J,K,L)=UO(J,K,L)*HU(J,K)-DSLX+TFC*UAV+HFR-SUMNON
2-DPRH+DPRS
UN1(J,K,L)=UN1(J,K,L)/HUN(J,K)
5 CONTINUE
C BOTTOM CONDITION
C PRESSURE DUE TO DENSITY
DPRH=HU(J,K)*HU(J,K)*GPL*(PRH(J,K,LMM)-PRH(J-1,K,LMM))
PRSS=0.5*(PRS(J,K,LMM)+PRS(J-1,K,LMM))
DPRS=GPL*HU(J,K)*(H(J,K)-H(J-1,K))*PRSS
```

```
C CORIOLIS
VR=0.5*(Vo(J,K,LMM)+Vo(J,K-1,LMM))
VL=0.5*(Vo(J-1,K,LMM)+Vo(J-1,K-1,LMM))
UAV=0.5*(HO(J,K)*VR+HO(J-1,K)*VL)
c UAV1=0.5*(VR+VL)
HZP=HZ(LMM)*HU(J,K)
HZN=0.5*(HZ(LMM)+HZ(LMM-1))*HU(J,K)
DSUP=2.*UO(J,K,LMM)/HZP
DSUN=(UO(J,K,LMM-1)-UO(J,K,LMM))/HZN
C HORIZONTAL FRICTION
HFRUX=PLL*(UO(J+1,K,LMM)*H(J,K)+UO(J-1,K,LMM)*H(J-1,K)
2-UO(J,K,LMM)*(H(J,K)+H(J-1,K)))
HFRUY=PFF*(UO(J,K+1,LMM)*DHU1+UO(J,K-1,LMM)*DHU2
3-(DHU1+DHU2)*UO(J,K,LMM))
HFR=HFRUX+HFRUY
C EVEN BETTER IMPROVED NONLINEAR TERMX
UAF=0.5*(UO(J,K,LMM)+UO(J+1,K,LMM))
UAB=0.5*(UO(J,K,LMM)+UO(J-1,K,LMM))
ONXF=0.5*(UO(J,K,LMM)*HU(J,K)+UO(J+1,K,LMM)*HU(J+1,K))
ONXB=0.5*(UO(J,K,LMM)*HU(J,K)+UO(J-1,K,LMM)*HU(J-1,K))
SUMNONX=PL*(ONXF*UAF-ONXB*UAB)
C UV
UAU=0.5*(Uo(J,K,LMM)+Uo(J,K+1,LMM))
UAD=0.5*(Uo(J,K-1,LMM)+Uo(J,K,LMM))
ONYU=0.5*(Vo(J,K,LMM)*HV(J,K)+Vo(J-1,K,LMM)*HV(J-1,K))
ONYD=0.5*(Vo(J,K-1,LMM)*HV(J,K-1)+
2Vo(J-1,K-1,LMM)*HV(J-1,K-1))
SUMNONY=PF*(UAU*ONYU-UAD*ONYD)
SUMNON=SUMNONX+SUMNONY
c vert deriv.
HZMM=HZ(LMM)+HZ(LMM-1)
waxn=0.5*(W(J-1,K,LMM)+W(J,K,LMM))
WAXP=0.
USN=(UO(J,K,L-1)*HZ(LMM)+UO(J,K,L)*HZ(LMM-1))/HZMM
USP=0.
UW=T*(WAXN*USN-WAXP*USP)/HZ(LMM)
SUMNON=SUMNON+UW
UN1(J,K,LMM)=UO(J,K,LMM)*HU(J,K)-DSLX+TFC*UAV+HFR-
SUMNON
2-DPRH+DPRS
UN1(J,K,LMM)=UN1(J,K,LMM)/HUN(J,K)
C UN(J,LMM)=0.
```

```
4 CONTINUe
c VERTICAL FRICTION SPLIT FROM THE REST
DO 46 n=1,nlh
do 46 j=js(n)+1,je(n)
do 46 k=ks(n),ke(n)
TAUSX(J,k)=0.
VR=0.5*(Vo(J,K,LE)+Vo(J,K-1,LE))
VL=0.5*(Vo(J-1,K,LE)+Vo(J-1,K-1,LE))
UAV1=0.5*(VR+VL)
HZPP=HZ(LE)*HUn(J,K)
C LOG OR NOT TO LOG
CD=2.5*ALOG(1.25*HZPP)
CD=1./(CD*CD)
RF1=AMAX1(CD,RF2)
PIERW=SQRT(UO(J,K,Le)*UO(J,K,Le)+uav1*uav1)
TAUBX(J,k)=RF1*PIERW*UO(J,K,Le)
HZP=0.5*(HZ(1)+HZ(2))*hu(j,k)
C(1)=T*ENU(J,K,2)/(HZ(1)*huN(j,k)*HZP)
D(1)=UN1(J,K,1)
DO L=2,LE-1
HZP=0.5*(HZ(L)+HZ(L+1))*hu(j,k)
HZN=0.5*(HZ(L)+HZ(L-1))*hu(j,k)
A(L)=T*ENU(j,k,L)/(HZ(L)*huN(j,k)*HZN)
C(L)=T*ENU(J,K,L+1)/(HZ(L)*huN(j,k)*HZP)
B(L)=1+A(L)+C(L)
D(L)=UN1(J,K,L)
END DO
HZN=0.5*(HZ(LE)+HZ(LE-1))*hu(j,k)
HZP=0.5*(HZ(LE)+HZ(LE+1))*hu(j,k)
D(LE)=UN1(J,K,LE)
A(Le)=T*ENU(j,k,Le)/(HZ(Le)*huN(j,k)*Hzn)
C(Le)=T*ENU(J,K,Le)/(HZ(Le)*huN(j,k)*HZP)
B(Le)=1+A(Le)+C(Le)
S(1)=C(1)/(1.+C(1))
E(1)=UN1(J,K,1)+T*TAUSX(J,K)/(HZ(1)*huN(j,k))
E(1)=E(1)/(1.+C(1))
DO L=2,LE-1
C VERTICAL FRICTION CONTINUE
DENOM=B(L)-S(L-1)*A(L)
S(L)=C(L)/DENOM
E(L)=(D(L)+A(L)*E(L-1))/DENOM
END DO
```

```
A1=A(LE)*E(LE-1)+D(LE)-T*TAUBX(J,K)/(HZ(LE)*huN(j,k))
A2=1.+A(le)*(1.-S(LE-1))
UN(J,K,LE)=A1/A2
DO L=LE-1,1,-1
UN(J,K,L)=UN(J,K,L+1)*S(L)+E(L)
END DO
46 CONTINUE
C COMPUTATION OF V VELOCITY
C VELOCITY IN THE AIR
do 2001 m=1,nlv
c
do 2001 j=jsv(m),jev(m)
do 2001 k=ksv(m),kev(m)-1
VAI(J,K)=VO(J,K,1)+HZ(1)*hV(j,K)*TAUSY(J,K)/ENV(J,K,1)
2001 continue
do 6 m=1,nlv
c if(ksv(m).eq.kev(m)) go to 6
do 6 j=jsv(m),jev(m)
do 6 k=ksv(m),kev(m)-1
C THIS GOES INTO HORIZONTAL FRICTION
DHV1=0.5*(HV(J,K)+HV(J+1,K))
DHV2=0.5*(HV(J,K)+HV(J-1,K))
C PRESSURE DUE TO DENSITY
DPRH=HU(J,K)*HU(J,K)*GPF*(PRH(J,K+1,1)-PRH(J,K,1))
PRSS=0.5*(PRS(J,K,1)+PRS(J,K+1,1))
DPRS=GPF*HU(J,K)*(H(J,K+1)-H(J,K))*PRSS
C SURFACE CONDITION
DSL=ZO(J,K+1)-ZO(J,K)
DSLY=GPF*DSL*HV(J,K)
HZP=0.5*(HZ(1)+HZ(2))*HV(J,K)
C CORIOLIS TERM
UG=0.5*(UN(J,K+1,1)+UN(J+1,K+1,1))
UD=0.5*(UN(J,K,1)+UN(J+1,K,1))
VAUN=0.5*(HN(J,K+1)*UG+HN(J,K)*UD)
DSV=(VO(J,K,1)-VO(J,K,2))/HZP
C VFRS=(TAUSY(J,K)-ENV(J,K,2)*DSV)/HZ(1)
C HORIZONTAL FRICTION
HFRVY=PFF*(VO(J,K+1,1)*H(J,K+1)+VO(J,K-1,1)*H(J,K)
2-(H(J,K)+H(J,K+1))*VO(J,K,1))
HFRVX=PLL*(VO(J+1,K,1)*DHV1+VO(J-1,K,1)*DHV2
3-(DHV1+DHV2)*VO(J,K,1))
HFR=HFRVX+HFRVY
```

```
C HFR=PLL*HV(J,K)*(VO(J+1,K,1)+VO(J-1,K,1)-2.*VO(J,K,1))+
C 2PFF*HV(J,K)*(VO(J,K+1,1)+VO(J,K-1,1)-2.*VO(J,K,1))
C EVEN BETTER NONLINEAR
C UV
VALX=0.5*(VO(J,K,1)+VO(J-1,K,1))
VARX=0.5*(VO(J,K,1)+VO(J+1,K,1))
ONXL=0.5*(HU(J,K+1)*UO(J,K+1,1)+HU(J,K)*UO(J,K,1))
ONXR=0.5*(HU(J+1,K+1)*UO(J+1,K+1,1)+HU(J+1,K)*UO(J+1,K,1))
SUMNONX=PL*(ONXR*VARX-ONXL*VALX)
C VV
VAGY=0.5*(VO(J,K+1,1)+VO(J,K,1))
VADY=0.5*(VO(J,K-1,1)+VO(J,K,1))
ONYU=0.5*(HV(J,K+1)*VO(J,K+1,1)+HV(J,K)*VO(J,K,1))
ONYD=0.5*(HV(J,K-1)*VO(J,K-1,1)+HV(J,K)*VO(J,K,1))
SUMNONY=PF*(ONYU*VAGY-ONYD*VADY)
SUMNON=SUMNONX+SUMNONY
CWV
VSU=0.5*(VO(J,K,1)+VAI(J,K))
VSD=(VO(J,K,1)*HZ(2)+VO(J,K,2)*HZ(1))/(HZ(1)+HZ(2))
WAG=0.5*(W(J,K,1)+W(J,K+1,1))
WAD=0.5*(W(J,K,2)+W(J,K+1,2))
WV=T*(VSU*WAG-VSD*WAD)/HZ(1)
SUMNON=SUMNON+WV
VN1(J,K,1)=VO(J,K,1)*HV(J,K)-TFC*VAUN-DSLY+HFR-
2SUMNON-DPRH+DPRS
VN1(J,K,1)=VN1(J,K,1)/HVN(J,K)
DO L=2,LE-1
C PRESSURE DUE TO DENSITY
DPRH=HU(J,K)*HU(J,K)*GPF*(PRH(J,K+1,L)-PRH(J,K,L))
PRSS=0.5*(PRS(J,K,L)+PRS(J,K+1,L))
DPRS=GPF*HU(J,K)*(H(J,K+1)-H(J,K))*PRSS
C CORIOLIS TERM
UG=0.5*(UN(J,K+1,L)+UN(J+1,K+1,L))
UD=0.5*(UN(J,K,L)+UN(J+1,K,L))
VAUN=0.5*(HN(J,K+1)*UG+HN(J,K)*UD)
HZP=0.5*(HZ(L)+HZ(L+1))*HV(J,K)
HZN=0.5*(HZ(L)+HZ(L-1))*HV(J,K)
DSVN=(VO(J,K,L-1)-VO(J,K,L))/HZN
DSVP=(VO(J,K,L)-VO(J,K,L+1))/HZP
C VFR=(ENV(J,K,L)*DSVN-ENV(J,K,L+1)*DSVP)/HZ(L)
C HORIZONTAL FRICTION
C HFR=PLL*HV(J,K)*(VO(J+1,K,L)+VO(J-1,K,L)-2.*VO(J,K,L))+
C 2PFF*HV(J,K)*(VO(J,K+1,L)+VO(J,K-1,L)-2.*VO(J,K,L))
C HORIZONTAL FRICTION
HFRVY=PFF*(VO(J,K+1,L)*H(J,K+1)+VO(J,K-1,L)*H(J,K)
2-(H(J,K)+H(J,K+1))*VO(J,K,L))
HFRVX=PLL*(VO(J+1,K,L)*DHV1+VO(J-1,K,L)*DHV2
3-(DHV1+DHV2)*VO(J,K,L))
HFR=HFRVX+HFRVY
C EVEN BETTER NONLINEAR
C UV
VALX=0.5*(VO(J,K,L)+VO(J-1,K,L))
VARX=0.5*(VO(J,K,L)+VO(J+1,K,L))
ONXL=0.5*(HU(J,K+1)*UO(J,K+1,L)+HU(J,K)*UO(J,K,L))
ONXR=0.5*(HU(J+1,K+1)*UO(J+1,K+1,L)+HU(J+1,K)*UO(J+1,K,L))
SUMNONX=PL*(ONXR*VARX-ONXL*VALX)
C VV
VAGY=0.5*(VO(J,K+1,L)+VO(J,K,L))
VADY=0.5*(VO(J,K-1,L)+VO(J,K,L))
ONYU=0.5*(HV(J,K+1)*VO(J,K+1,L)+HV(J,K)*VO(J,K,L))
ONYD=0.5*(HV(J,K-1)*VO(J,K-1,L)+HV(J,K)*VO(J,K,L))
SUMNONY=PF*(ONYU*VAGY-ONYD*VADY)
SUMNON=SUMNONX+SUMNONY
CWV
VSU=(VO(J,K,L-1)*HZ(L)+VO(J,K,L)*HZ(L-1))/(HZ(L)+HZ(L-1))
VSD=(VO(J,K,L)*HZ(L+1)+VO(J,K,L+1)*HZ(L))/(HZ(L)+HZ(L+1))
WAG=0.5*(W(J,K,L)+W(J,K+1,L))
WAD=0.5*(W(J,K,L+1)+W(J,K+1,L+1))
WV=T*(VSU*WAG-VSD*WAD)/HZ(L)
SUMNON=SUMNON+WV
VN1(J,K,L)=VO(J,K,L)*HV(J,K)-TFC*VAUN-DSLY+HFR-
2SUMNON-DPRH+DPRS
VN1(J,K,L)=VN1(J,K,L)/HVN(J,K)
end do
C BOTTOM
C PRESSURE DUE TO DENSITY
DPRH=HU(J,K)*HU(J,K)*GPF*(PRH(J,K+1,LE)-PRH(J,K,LE))
PRSS=0.5*(PRS(J,K,LE)+PRS(J,K+1,LE))
DPRS=GPF*HU(J,K)*(H(J,K+1)-H(J,K))*PRSS
C CORIOLIS TERM
UG=0.5*(UN(J,K+1,LE)+UN(J+1,K+1,LE))
UD=0.5*(UN(J,K,LE)+UN(J+1,K,LE))
VAUN=0.5*(HN(J,K+1)*UG+HN(J,K)*UD)
C VAU=0.25*(UO(J,K+1,LE)+UO(J+1,K+1,LE)+UO(J,K,LE)
C 2+UO(J+1,K,LE))
HZP=HZ(LE)*HV(J,K)
HZN=0.5*(HZ(LE)+HZ(LE-1))*HV(J,K)
DSVN=(VO(J,K,LE-1)-VO(J,K,LE))/HZN
C HORIZONTAL FRICTION
HFRVY=PFF*(VO(J,K+1,LE)*H(J,K+1)+VO(J,K-1,LE)*H(J,K)
2-(H(J,K)+H(J,K+1))*VO(J,K,LE))
HFRVX=PLL*(VO(J+1,K,LE)*DHV1+VO(J-1,K,LE)*DHV2
3-(DHV1+DHV2)*VO(J,K,LE))
HFR=HFRVX+HFRVY
C HFR=PLL*HV(J,K)*(VO(J+1,K,LE)+VO(J-1,K,LE)-2.*VO(J,K,LE))+
C 2PFF*HV(J,K)*(VO(J,K+1,LE)+VO(J,K-1,LE)-2.*VO(J,K,LE))
C EVEN BETTER NONLINEAR
C UV
VALX=0.5*(VO(J,K,LE)+VO(J-1,K,LE))
VARX=0.5*(VO(J,K,LE)+VO(J+1,K,LE))
ONXL=0.5*(HU(J,K+1)*UO(J,K+1,LE)+HU(J,K)*UO(J,K,LE))
ONXR=0.5*(HU(J+1,K+1)*UO(J+1,K+1,LE)+HU(J+1,K)*UO(J+1,K,LE))
SUMNONX=PL*(ONXR*VARX-ONXL*VALX)
C VV
VAGY=0.5*(VO(J,K+1,LE)+VO(J,K,LE))
VADY=0.5*(VO(J,K-1,LE)+VO(J,K,LE))
ONYU=0.5*(HV(J,K+1)*VO(J,K+1,LE)+HV(J,K)*VO(J,K,LE))
ONYD=0.5*(HV(J,K-1)*VO(J,K-1,LE)+HV(J,K)*VO(J,K,LE))
SUMNONY=PF*(ONYU*VAGY-ONYD*VADY)
SUMNON=SUMNONX+SUMNONY
CWV
VSU=(VO(J,K,LE-1)*HZ(LE)+VO(J,K,LE)*HZ(LE-1))/
2(HZ(LE)+HZ(LE-1))
VSD=0.
WAG=0.5*(W(J,K,LE)+W(J,K+1,LE))
WAD=0.
WV=T*(VSU*WAG-VSD*WAD)/HZ(LE)
SUMNON=SUMNON+WV
VN1(J,K,LE)=VO(J,K,LE)*HV(J,K)-TFC*VAUN-DSLY+HFR-
2SUMNON-DPRH+DPRS
VN1(J,K,LE)=VN1(J,K,LE)/HVN(J,K)
6 continue
c VERTICAL FRICTION SPLIT FROM THE REST
do 7 m=1,nlv
c
do 7 j=jsv(m),jev(m)
do 7 k=ksv(m),kev(m)-1
HZPP=HZ(LE)*HVn(J,K)
VAU=0.25*(UO(J,K+1,LE)+UO(J+1,K+1,LE)+UO(J,K,LE)
2+UO(J+1,K,LE))
C LOG OR NOT TO LOG
CD=2.5*ALOG(1.25*HZPP)
CD=1./(CD*CD)
RF1=AMAX1(CD,RF2)
PIERW=SQRT(VO(J,K,Le)*VO(J,K,Le)+VAU*VAU)
TAUBY(J,k)=RF1*PIERW*VO(J,K,Le)
HZP=0.5*(HZ(1)+HZ(2))*hV(j,k)
C(1)=T*ENV(J,K,2)/(HZ(1)*hVN(j,k)*HZP)
D(1)=VN1(J,K,1)
DO L=2,LE-1
HZP=0.5*(HZ(L)+HZ(L+1))*hV(j,k)
HZN=0.5*(HZ(L)+HZ(L-1))*hV(j,k)
A(L)=T*ENV(j,k,L)/(HZ(L)*HVN(j,k)*HZN)
C(L)=T*ENV(J,K,L+1)/(HZ(L)*HVN(j,k)*HZP)
B(L)=1+A(L)+C(L)
D(L)=VN1(J,K,L)
END DO
HZN=0.5*(HZ(LE)+HZ(LE-1))*HV(j,k)
HZP=0.5*(HZ(LE)+HZ(LE+1))*HV(j,k)
D(LE)=VN1(J,K,LE)
A(Le)=T*ENV(j,k,Le)/(HZ(Le)*HVN(j,k)*Hzn)
C(Le)=T*ENV(J,K,Le)/(HZ(Le)*HVN(j,k)*HZP)
B(Le)=1+A(Le)+C(Le)
S(1)=C(1)/(1.+C(1))
E(1)=VN1(J,K,1)+T*TAUSX(J,K)/(HZ(1)*HVN(j,k))
E(1)=E(1)/(1.+C(1))
DO L=2,LE-1
C VERTICAL FRICTION CONTINUE
DENOM=B(L)-S(L-1)*A(L)
S(L)=C(L)/DENOM
E(L)=(D(L)+A(L)*E(L-1))/DENOM
END DO
A1=A(LE)*E(LE-1)+D(LE)-T*TAUBX(J,K)/(HZ(LE)*HVN(j,k))
A2=1.+A(le)*(1.-S(LE-1))
VN(J,K,LE)=A1/A2
DO L=LE-1,1,-1
VN(J,K,L)=VN(J,K,L+1)*S(L)+E(L)
END DO
```

```
7 CONTINUE
C SEA LEVEL
C FIRST AVERAGE VELOCITY ALONG VERTICAL DIRECTION
c print *, un(2,5,1),un(3,5,1)
DO 9 n=1,nlh
do 9 j=js(n)+1,je(n)
do 9 k=ks(n),ke(n)
MM=LE
C++++++++++++++++++++++++++++++++++++++++
UA(J,k)=0.
DO 10 L=MM,1,-1
UA(J,K)=UA(J,K)+UN(J,K,L)*HZ(L)
10 CONTINUE
9 CONTINUE
c**************************************
do 2000 m=1,nlv
if(ksv(m).eq.kev(m)) go to 2000
c
do 2000 j=jsv(m),jev(m)
do 2000 k=ksv(m),kev(m)-1
MM=LE
VA(J,K)=0.
DO 12 L=MM,1,-1
VA(J,K)=VA(J,K)+VN(J,K,L)*HZ(L)
12 CONTINUE
2000 CONTINUE
C NOW USE CONTINUITY EQUATION. FOR THE CLOSED DOMAIN
COMPUTATION STARTS
c LZ=0
do 13 n=1,nlh
do 13 j=js(n),je(n)
do 13 k=ks(n),ke(n)
c93 CONTINUE
ZN(J,K)=ZO(J,K)-PL*(UA(J+1,K)*HUN(J+1,K)-UA(J,K)*HUN(J,K))-
2PF*(VA(J,K)*HVN(J,K)-VA(J,K-1)*HVN(J,K-1))
c if(j.eq.jx)then
c
c print *,ua(j+1),hu(j+1)
c end if
c LZ=LZ+1
c IF(LZ.EQ.2)THEN
c LZ=0
```

```
c ZS(J)=ZO(J)
c GO TO 13
c END IF
c ZO(J)=0.25*ZN(J)+0.5*ZO(J)+0.25*ZS(J)
c GO TO 93
13 CONTINUE
c vertical velocity from continuity equation
do 25 n=1,nlh
do 25 j=js(n),je(n)
do 25 k=ks(n),ke(n)
DIFZ=ZN(J,K)-ZO(J,K)
MLEZ=LE
W(J,K,MLEZ+1)=0.
DO 26 L=MLEZ,1,-1
HZT=HZ(L)/T
HZX=HZ(L)/HX
HZY=HZ(L)/HY
DIFU=UN(J+1,K,L)*HUN(J+1,K)-UN(J,K,L)*HUN(J,K)
DIFV=(VN(J,K,L)*HVN(J,K)-VN(J,K-1,L)*HVN(J,K-1))
W(J,K,L)=W(J,K,L+1)-HZX*DIFU-HZY*DIFV-HZT*DIFZ
26 CONTINUE
25 CONTINUE
C CHANGE VERTICAL VELOCITY FROM SIGMA TO Z COORDINATES
IF(NN.EQ.MON) THEN
do 27 n=1,nlh
do 27 j=js(n),je(n)
do 27 k=ks(n),ke(n)
DIFZT=(ZN(J,K)-ZO(J,K))/T
DIFZX=(ZN(J+1,K)-ZN(J-1,K))/TWHX
DIFDX=(H(J+1,K)+ZN(J+1,K)-H(J-1,K)+ZN(J-1,K))/TWHX
DIFZY=(ZN(J,K+1)-ZN(J,K-1))/TWHY
DIFDY=(H(J,K+1)+ZN(J,K+1)-H(J,K-1)+ZN(J,K-1))/TWHY
Uso=(UN(J+1,K,1)+UN(J,K,1))*0.5
VSO=0.5*(vN(j,k,1)+vN(j,k-1,1))
HZSs=0.
WZ(J,K,1)=W(J,K,1)+DIFZT +Uso*DIFZX+VSO*DIFZY
DO L=2,LE
HZSs=HZSs+HZ(L-1)
Uso=(UN(J+1,K,L)+UN(J,K,L))*HZ(L-1)
1+(UN(J+1,K,L-1)+UN(J,K,L-1))*HZ(L)
Uso=Uso/(HZ(L)+HZ(L-1))
VSO=(VN(J,K,L)+VN(J,K-1,L))*HZ(L-1)
```

```
2+(VN(J,K,L-1)+VN(J,K-1,L-1))*HZ(L)
Vso=Vso/(HZ(L)+HZ(L-1))
WZ(J,K,L)=W(J,K,L)+(-HZSs+1)*DIFZT +Uso*(-HZss*DIFDX+DIFZX)
3+Vso*(-HZss*DIFDY+DIFZY)
end do
WZ(J,K,LE+1)=W(J,K,LE+1)
27 CONTINUE
end if
C OLD AND NEW DEPH
DO J=1,JX
DO K=1,KY
HO(J,K)=H(J,K)+ZO(J,K)
HN(J,K)=H(J,K)+ZN(J,K)
END DO
END DO
DO K=1,KY
HU(1,K)=HO(1,K)
HUN(1,K)=HN(1,K)
end do
do j=2,jx
do k=1,ky
HU(j,k)=0.5*(Ho(j-1,k)+ho(j,k))
HUN(j,k)=0.5*(Hn(j-1,k)+hn(j,k))
end do
end do
do j=1,JX
HV(J,KY)=Ho(J,KY)
HVN(J,KY)=Hn(J,KY)
END DO
DO J=1,JX
DO K=1,KY-1
HV(J,K)=0.5*(HO(J,K)+HO(J,K+1))
HVN(J,K)=0.5*(HN(J,K)+HN(J,K+1))
end do
end do
c go to 223
CDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
C DENSITY COMPUTATION BASED ON SALINITY AND
C TEMPERATURE CHANGES
do 1300 n=1,nlh
do 1300 j=js(n),je(n)
do 1300 k=ks(n),ke(n)
```

```
DO L=1,LE
C ADVECTIVE TERMS
FLXR=0.5*(SO(J,K,L)+SO(J+1,K,L))*UO(J+1,K,L)*HU(J+1,K)
FLXL=0.5*(SO(J-1,K,L)+SO(J,K,L))*UO(J,K,L)*HU(J,K)
ADU=PL*(FLXR-FLXL)
FLYU=0.5*(SO(J,K,L)+SO(J,K+1,L))*VO(J,K,L)*HV(J,K)
FLYD=0.5*(SO(J,K,L)+SO(J,K-1,L))*VO(J,K-1,L)*HV(J,K-1)
ADU=PL*(FLXR-FLXL)+PF*(FLYU-FLYD)
C HORIZONTAL DIFFUSION
C DHU ARE HORIZONTAL EDDY DIFFUSIVITY DEFINED
C AT U POINTS (*T/HX*HX)
C DHV ARE HORIZONTAL EDDY DIFFUSIVITY DEFINED
C AT V POINTS(*T/HY*HY)
HDFX1=HU(j+1,K)*DHU(J+1,K)*(SO(J+1,K,L)-SO(J,K,L))
HDFX2=HU(J,K)*DHU(J,K)*(SO(J,K,L)-SO(J-1,K,L))
HDFX=HDFX1-HDFX2
HDFY1=HV(J,K)*DHV(J,K)*(SO(J,K+1,L)-SO(J,K,L))
HDFY2=HV(J,K-1)*DHV(J,K-1)*(SO(J,K,L)-SO(J,K-1,L))
HDFY=HDFY1-HDFY2
HDF=HDFX+HDFY
ST(J,K,L)=SO(J,K,L)*(HO(J,K)/HN(J,K))-ADU/HN(J,K)+HDF/HN(J,K)
END DO
1300 CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C CONVECTIVE TERMS
C CONVECTIVE TERMS
C —
C + S(J,L-1)
C —– ——– w(j,l),DV(J,K,L)
C HZ(L) + S(J,L)
C ——– ——– w(j,l+1),DV(J,K,L+1)
C HZ(L+1) + S(J,L+1)
C —
C —
do 1301 n=1,nlh
do 1301 j=js(n),je(n)
do 1301 k=ks(n),ke(n)
C SURFACE CONDITION
C CONVECTION
C WPU=0.5*(W(J,K,1)+ABS(W(J,K,1)))
WPD=0.5*(W(J,K,2)+ABS(W(J,K,2)))
WND=0.5*(W(J,K,2)-ABS(W(J,K,2)))
```

```
FLZU=0.
FLZD=WPD*SO(J,K,2)+WND*SO(J,K,1)
CON=T*(FLZU-FLZD)/HZ(1)
STV(J,K,1)=ST(J,K,1)-CON/HN(J,K)
C print *,W(3,11,2)
C INTERNAL DOMAIN
DO L=2,LE-1
C VERTICAL convectioN
FLZU=W(J,K,L)*(SO(J,K,L)+SO(J,K,L-1))*0.5
FLZD=W(J,K,L+1)*(SO(J,K,L+1)+SO(J,K,L))*0.5
CON=T*(FLZU-FLZD)/HZ(L)
C VERTICAL DIFFUSION
C HZP=0.5*(HZ(L)+HZ(L+1))*H(J)
C HZN=0.5*(HZ(L)+HZ(L-1))*H(J)
C DSN=(SO(J,K,L-1)-SO(J,K,L))/HZN
C DSP=(SO(J,K,L)-SO(J,K,L+1))/HZP
C VDF=T*(DV(J,K,L)*DSN-DV(J,L+1)*DSP)/HZ(L)
STV(J,K,L)=ST(J,K,L)-CON/HN(J,K)
END DO
C BOTTOM
C VERTICAL CONVECTION
WPU=0.5*(W(J,K,LE)+ABS(W(J,K,LE)))
WNU=0.5*(W(J,K,LE)-ABS(W(J,K,LE)))
WPD=0.
WND=0.
FLZU=WPU*SO(J,K,LE)+WNU*SO(J,K,LE-1)
FLZD=0.
CON=T*(FLZU-FLZD)/HZ(LE)
STV(J,K,LE)=ST(J,K,LE)-CON/HN(J,K)
1301 CONTINUE
C VERTICAL DIFFUSION
do 1302 n=1,nlh
do 1302 j=js(n),je(n)
do 1302 k=ks(n),ke(n)
HZP=0.5*(HZ(1)+HZ(2))*HUN(J,K)
C(1)=T*DV(J,K,2)/(HZ(1)*HUN(J,K)*HZP)
D(1)=STV(J,K,1)
DO L=2,LE
HZP=0.5*(HZ(L)+HZ(L+1))*HUN(J,K)
HZN=0.5*(HZ(L)+HZ(L-1))*HUN(J,K)
A(L)=T*DV(J,K,L)/(HZ(L)*HUN(J,K)*HZN)
C(L)=T*DV(J,K,L+1)/(HZ(L)*HUN(J,K)*HZP)
```

```
B(L)=1.+A(L)+C(L)
D(L)=STV(J,K,L)
END DO
D(LE)=STV(J,K,LE)
HZN=0.5*(HZ(LE)+HZ(LE-1))*HUN(J,K)
A(LE)=T*DV(J,K,LE)/(HZ(LE)*HUN(J,K)*HZN)
S(1)=C(1)/(1.+C(1))
C IN CASE FLUX OF HEAT (OR SAL) OCCURS AT THE
C SURFACE FLUX=DV(1)*(S(0)-S(1)/(HZ(1)HUN(J,K))
C E(1)= STV(J,K,1)+ T*FLUX(J,K)/(HZ(1)*huN(j))
C E(1)= E(1)/(1.+C(1))
E(1)=STV(J,K,1)/(1.+C(1))
DO L=2,LE-1
C VERTICAL DIFFUSION
DENOM=B(L)-S(L-1)*A(L)
S(L)=C(L)/DENOM
E(L)=(D(L)+A(L)*E(L-1))/DENOM
END DO
A1=A(LE)*E(LE-1)+D(LE)
A2=1.+A(LE)*(1.-S(LE-1))
SN(J,K,LE)=A1/A2
DOL=LE-1,1,-1
SN(J,K,L)=SN(J,K,L+1)*S(L)+E(L)
END DO
1302 CONTINUE
C GO TO 1360
C TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
C TEMPERATURE
do 1350 n=1,nlh
do 1350 j=js(n),je(n)
do 1350 k=ks(n),ke(n)
DO L=1,LE
C ADVECTIVE TERMS
FLXR=0.5*(TO(J,K,L)+TO(J+1,K,L))*UO(J+1,K,L)*HU(J+1,K)
FLXL=0.5*(TO(J-1,K,L)+TO(J,K,L))*UO(J,K,L)*HU(J,K)
ADU=PL*(FLXR-FLXL)
FLYU=0.5*(TO(J,K,L)+TO(J,K+1,L))*VO(J,K,L)*HV(J,K)
FLYD=0.5*(TO(J,K,L)+TO(J,K-1,L))*VO(J,K-1,L)*HV(J,K-1)
ADU=PL*(FLXR-FLXL)+PF*(FLYU-FLYD)
C HORIZONTAL DIFFUSION
C DHU ARE HORIZONTAL EDDY DIFFUSIVITY DEFINED AT
C U POINTS (*T/HX*HX)
```

```
C DHV ARE HORIZONTAL EDDY DIFFUSIVITY DEFINED AT
C V POINTS(*T/HY*HY)
HDFX1=HU(j+1,K)*DHU(J+1,K)*(TO(J+1,K,L)-TO(J,K,L))
HDFX2=HU(J,K)*DHU(J,K)*(TO(J,K,L)-TO(J-1,K,L))
HDFX=HDFX1-HDFX2
HDFY1=HV(J,K)*DHV(J,K)*(TO(J,K+1,L)-TO(J,K,L))
HDFY2=HV(J,K-1)*DHV(J,K-1)*(TO(J,K,L)-TO(J,K-1,L))
HDFY=HDFY1-HDFY2
HDF=HDFX+HDFY
TT(J,K,L)=TO(J,K,L)*(HO(J,K)/HN(J,K))-ADU/HN(J,K)+HDF/HN(J,K)
END DO
1350 CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C CONVECTIVE TERMS
C CONVECTIVE TERMS
C —
C + T(J,L-1)
C —— ——— w(j,l),DV(J,K,L)
C HZ(L) + T(J,L)
C ——— ——— w(j,l+1),DV(J,K,L+1)
C HZ(L+1) + T(J,L+1)
C —
C —
do 1351 n=1,nlh
do 1351 j=js(n),je(n)
do 1351 k=ks(n),ke(n)
C SURFACE CONDITION
C CONVECTION
C WPU=0.5*(W(J,K,1)+ABS(W(J,K,1)))
WPD=0.5*(W(J,K,2)+ABS(W(J,K,2)))
WND=0.5*(W(J,K,2)-ABS(W(J,K,2)))
FLZU=0.
FLZD=WPD*TO(J,K,2)+WND*TO(J,K,1)
CON=T*(FLZU-FLZD)/HZ(1)
TTV(J,K,1)=TT(J,K,1)-CON/HN(J,K)
C print *,W(3,11,2)
C INTERNAL DOMAIN
DO L=2,LE-1
C VERTICAL convectioN
FLZU=W(J,K,L)*(TO(J,K,L)+TO(J,K,L-1))*0.5
FLZD=W(J,K,L+1)*(TO(J,K,L+1)+TO(J,K,L))*0.5
CON=T*(FLZU-FLZD)/HZ(L)
```

```
C VERTICAL DIFFUSION
C HZP=0.5*(HZ(L)+HZ(L+1))*H(J)
C HZN=0.5*(HZ(L)+HZ(L-1))*H(J)
C DSN=(TO(J,K,L-1)-TO(J,K,L))/HZN
C DSP=(TO(J,K,L)-TO(J,K,L+1))/HZP
C VDF=T*(DV(J,K,L)*DSN-DV(J,L+1)*DSP)/HZ(L)
TTV(J,K,L)=TT(J,K,L)-CON/HN(J,K)
END DO
C BOTTOM
C VERTICAL CONVECTION
WPU=0.5*(W(J,K,LE)+ABS(W(J,K,LE)))
WNU=0.5*(W(J,K,LE)-ABS(W(J,K,LE)))
WPD=0.
WND=0.
FLZU=WPU*TO(J,K,LE)+WNU*TO(J,K,LE-1)
FLZD=0.
CON=T*(FLZU-FLZD)/HZ(LE)
TTV(J,K,LE)=TT(J,K,LE)-CON/HN(J,K)
1351 CONTINUE
C VERTICAL DIFFUSION
do 1352 n=1,nlh
do 1352 j=js(n),je(n)
do 1352 k=ks(n),ke(n)
HZP=0.5*(HZ(1)+HZ(2))*HUN(J,K)
C(1)=T*DV(J,K,2)/(HZ(1)*HUN(J,K)*HZP)
D(1)=TTV(J,K,1)
DO L=2,LE
HZP=0.5*(HZ(L)+HZ(L+1))*HUN(J,K)
HZN=0.5*(HZ(L)+HZ(L-1))*HUN(J,K)
A(L)=T*DV(J,K,L)/(HZ(L)*HUN(J,K)*HZN)
C(L)=T*DV(J,K,L+1)/(HZ(L)*HUN(J,K)*HZP)
B(L)=1.+A(L)+C(L)
D(L)=TTV(J,K,L)
END DO
D(LE)=TTV(J,K,LE)
HZN=0.5*(HZ(LE)+HZ(LE-1))*HUN(J,K)
A(LE)=T*DV(J,K,LE)/(HZ(LE)*HUN(J,K)*HZN)
S(1)=C(1)/(1.+C(1))
C IN CASE FLUX OF HEAT (OR SAL) OCCURS AT THE
C SURFACE FLUX=DV(1)*(S(0)-S(1)/(HZ(1)HUN(J,K))
C E(1)= TTV(J,K,1)+ T*FLUX(J,K)/(HZ(1)*huN(j))
C E(1)= E(1)/(1.+C(1))
```

```
E(1)=TTV(J,K,1)/(1.+C(1))
DO L=2,LE-1
C VERTICAL DIFFUSION
DENOM=B(L)-S(L-1)*A(L)
S(L)=C(L)/DENOM
E(L)=(D(L)+A(L)*E(L-1))/DENOM
END DO
A1=A(LE)*E(LE-1)+D(LE)
A2=1.+A(LE)*(1.-S(LE-1))
TN(J,K,LE)=A1/A2
DOL=LE-1,1,-1
TN(J,K,L)=TN(J,K,L+1)*S(L)+E(L)
END DO
1352 CONTINUE
C1360 CONTINUE
C DENSITY SIGMAT
c density variations
C according to Friedrich and Levitus, 1972, JPO
do 1303 n=1,nlh
do 1303 j=js(n),je(n)
do 1303 k=ks(n),ke(n)
DO L=1,LE
T2=TN(J,K,L)*TN(J,K,L)
T3=TN(J,K,L)*TN(J,K,L)*TN(J,K,L)
RHO(J,K,L)=ALP(1)+ALP(2)*TN(J,K,L)+ALP(3)*SN(J,K,L)
1+ALP(4)*T2+ALP(5)*SN(J,K,L)*TN(J,K,L)
2+ALP(6)*T3+ALP(7)*SN(J,K,L)*T2
RHO(J,K,L)=RHO(J,K,L)*0.001
C RHO1(J,K,L)=SN(J,K,L)*0.99891*7.5E-4
END DO
C PRINT *,RHO(10,10,2),RHO1(10,10,2)
1303 CONTINUE
IF(NN.EQ.1)THEN
WRITE(25,*)(((RHO(J,k,L)*1000.,J=1,JX),k=1,ky),L=1,LE)
END IF
C=========================================
C VERTICAL DENSITY AND ALSO SAL AND TEMP MIXING
C FOR AN UNSTABLE PROFILE
C=========================================
do 235 n=1,nlh
do 235 j=js(n),je(n)
do 235 k=ks(n),ke(n)
```

```
MSS=0
P7=0.
P7S=0.
P7T=0.
P8=0.
DO 235 L=1,LE-1
SDIF=RHO(J,K,L)-RHO(J,K,L+1)
IF(SDIF.GT.0.0)THEN
MSS=MSS+1
P7T=P7T+0.5*(TN(J,K,L)+TN(J,K,L+1))*HZ(L)*HN(J,K)
P7S=P7S+0.5*(SN(J,K,L)+SN(J,K,L+1))*HZ(L)*HN(J,K)
P7=P7+0.5*(RHO(J,K,L)+RHO(J,K,L+1))*HZ(L)*HN(J,K)
P8=P8+HZ(L)*HN(J,K)
TN(J,K,L+1)=P7T/P8
SN(J,K,L+1)=P7S/P8
RHO(J,K,L+1)=P7/P8
DO 236 N1=1,MSS
L1=L+1-N1
RHO(J,K,L1)=RHO(J,K,L1+1)
SN(J,K,L1)=SN(J,K,L1+1)
TN(J,K,L1)=TN(J,K,L1+1)
236 CONTINUE
IF(L.EQ.LE-1)THEN
MSS=0
P7=0.
P7S=0.
P7T=0.
P8=0.
GO TO 235
END IF
GO TO 235
END IF
MSS=0
P7=0.
P7S=0.
P7T=0.
P8=0.
235 CONTINUE
C PRESSURE DUE TO DENSITY PRH
do 1304 n=1,nlh
do 1304 j=js(n),je(n)
do 1304 k=ks(n),ke(n)
```

```
    PRH(J,K,1)=RHO(J,K,1)*HZ(1)*0.5
    DO L=2,LE
    PRH(J,K,L)=PRH(J,K,L-1)+(RHO(J,K,L-1)*HZ(L-
1)+RHO(J,K,L)*HZ(L))*0.5
    C PRH(J,K,L)=0.
    END DO
    1304 CONTINUE
    C PRESSURE DUE TO SIGMA COORDINATE
    do 1305 n=1,nlh
    do 1305 j=js(n),je(n)
    do 1305 k=ks(n),ke(n)
    PRS(J,K,1)=0.
    HZSI=HZ(1)*0.5
    DO L=2,LE
    C RHU=0.5*(RHO(J,K,L-1)+RHO(J-1,K,L-1))
    C RHM=0.5*(RHO(J,K,L)+RHO(J-1,K,L))
    RHU=RHO(J,K,L-1)
    RHM=RHO(J,K,L)
    HZSI=HZSI+0.5*(HZ(L-1)+HZ(L))
    DRH=RHU-RHm
    PRS(J,K,L)=PRS(J,K,L-1)+HZSI*DRH
    C PRS(J,L)=0.
    END DO
    1305 CONTINUE
    223 continue
    C CHANGE VARIABLES
    DO 15 J=1,JX
    DO 15 K=1,KY
    ZO(j,k)=ZN(J,K)
    DO 15L=1,LE
    UO(J,K,L)=UN(J,K,L)
    VO(J,K,L)=VN(J,K,L)
    SO(J,K,L)=SN(J,K,L)
    TO(J,K,L)=TN(J,K,L)
    15 CONTINUE
    if(mod(Nn,100).eq.0) then
    C THIS FOR THE VERTICALLY AVERAGE FLOW
    c Computing kinetic and potential energy every 10 steps
    c and also computing average sea level
    ZS=0.
    ES=0.
    do 130 n=1,nlh
    do 130 j=js(n),je(n)
    do 130 k=ks(n),ke(n)
    vae=0.5*(va(j,k)+va(j,k-1))
    uaE=0.5*(ua(j,k)+ua(j+1,k))
    enR=(uaE*uaE+vae*vae)/(2.*g*h(j,k))+zn(j,k)*zn(j,k)/2.
    es=enR+es
    ZS=ZS+ZN(J,K)
    130 continue
    write(12,*)es
    print *,nn,'zn(3)=',zn(3,11),'zn(11)=',zn(11,11)
    print *,'zn(19)=',zn(19,11),'ZS=',ZS
    print *,'sn(11,11,1)=',sn(3,11,1),sn(11,11,1),sn(19,11,1)
    print *,'WW=',W(3,11,2)
    write(11,*),n,zn(3,11),zn(11,11),zn(19,11)
    end if
    IF(NN.LT.MON)GO TO 50
    c WRITE SEA LEVEL AND VELOCITIES
    WRITE(10,*)((zn(J,k),J=1,JX),k=1,ky)
    Do j=1,jx-1
    do l=1,le
    upX(j,l)=0.5*(UN(J,11,L)+UN(J+1,11,L))
    END DO
    END DO
    WRITE(16,*)((upX(j,L),J=1,JX),L=1,LE)
    DO K=2,KY
    do l=1,le
    upY(K,l)=0.5*(VN(11,K,L)+VN(11,K-1,L))
    END DO
    END DO
    WRITE(17,*)((upy(k,L),k=1,ky),L=1,LE)
    Do j=1,jx-1
    DO K=1,KY
    do l=1,le
    upXX(j,K,l)=0.5*(UN(J,K,L)+UN(J+1,K,L))
    END DO
    END DO
    END DO
    WRITE(7,*)(((upXX(J,k,L),J=1,JX),k=1,ky),L=1,LE)
    Do j=1,jx
    DO K=2,KY
    do l=1,le
    upYY(j,K,l)=0.5*(VN(J,K,L)+VN(J,K-1,L))
```

```
END DO
END DO
END DO
WRITE(8,*)(((upYY(J,k,L),J=1,JX),k=1,ky),L=1,LE)
DO J=1,JX
DO L=1,LE-1
WPX(J,L)=0.5*(WZ(J,11,L)+WZ(J,11,L+1))
END DO
wPX(j,le)=0.5*(wz(j,11,le))
END DO
write(15,*)((wPX(J,L),J=1,JX),L=1,LE)
DO K=1,KY
DO L=1,LE-1
WPY(K,L)=0.5*(WZ(11,K,L)+WZ(11,K,L+1))
END DO
WPY(K,LE)=0.5*WZ(11,K,LE)
END DO
write(9,*)((wPY(J,L),J=1,JX),L=1,LE)
do L=1,LE
write(18,*)(sn(J,11,L),J=1,JX)
END DO
do L=1,LE
write(22,*)(TN(J,11,L),J=1,JX)
END DO
WRITE(24,*)(((RHO(J,k,L)*1000.,J=1,JX),k=1,ky),L=1,LE)
222 continue
stop
end
```

Numerical experiments with the density stratified flow are done in the circular shaped water body with bathymetry used in the previous experiment (Fig.III.7). Initial density distribution in the entire body of fluid is prescribed in the program. Fig.III.11a depicts initial density distribution at the free surface and Fig.III.11b describes density after approximately 11.5 days of process. Due to interaction of the density with the bathymetry the density both at the surface (Fig.III.11b) and throughout the entire column of fluid shows circular isocontours. Although the process of adjustment is not completed even after 11.5 days. In Fig.III.12 the sea level (continuous lines) and bathymetry (dotted lines) show some asymmetry in the central region of the water body. The spatial distribution of the density and sea level resulted in the counterclockwise circular flow centered at the sea level minimum (Fig.III.13) with the maximum flow close to 12 cm/s.

Fig.III.11a: Surface density (sigma_t)

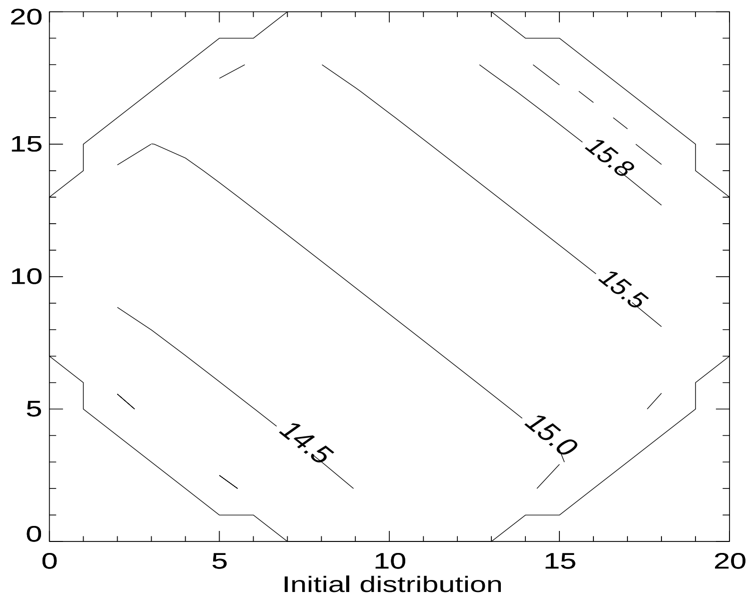


Fig.III.11b: Surface density (sigma_t)

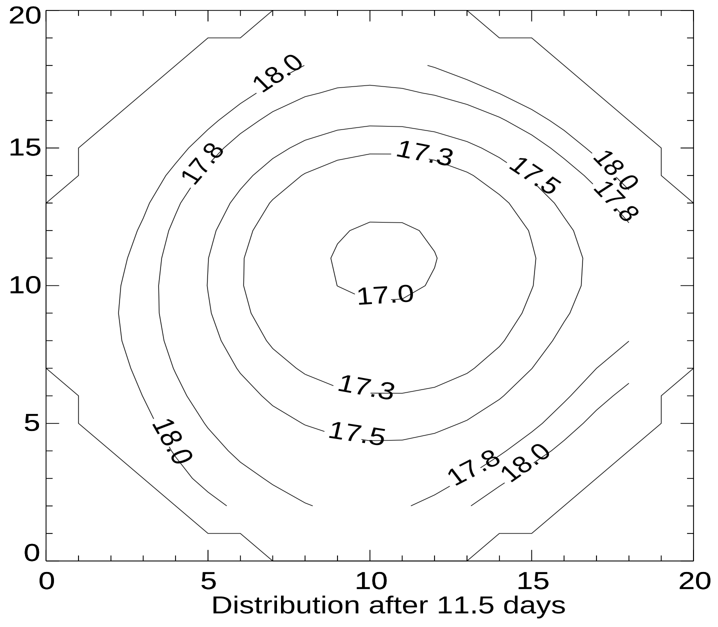


Fig.III.12: Sea level distribution - continuous lines

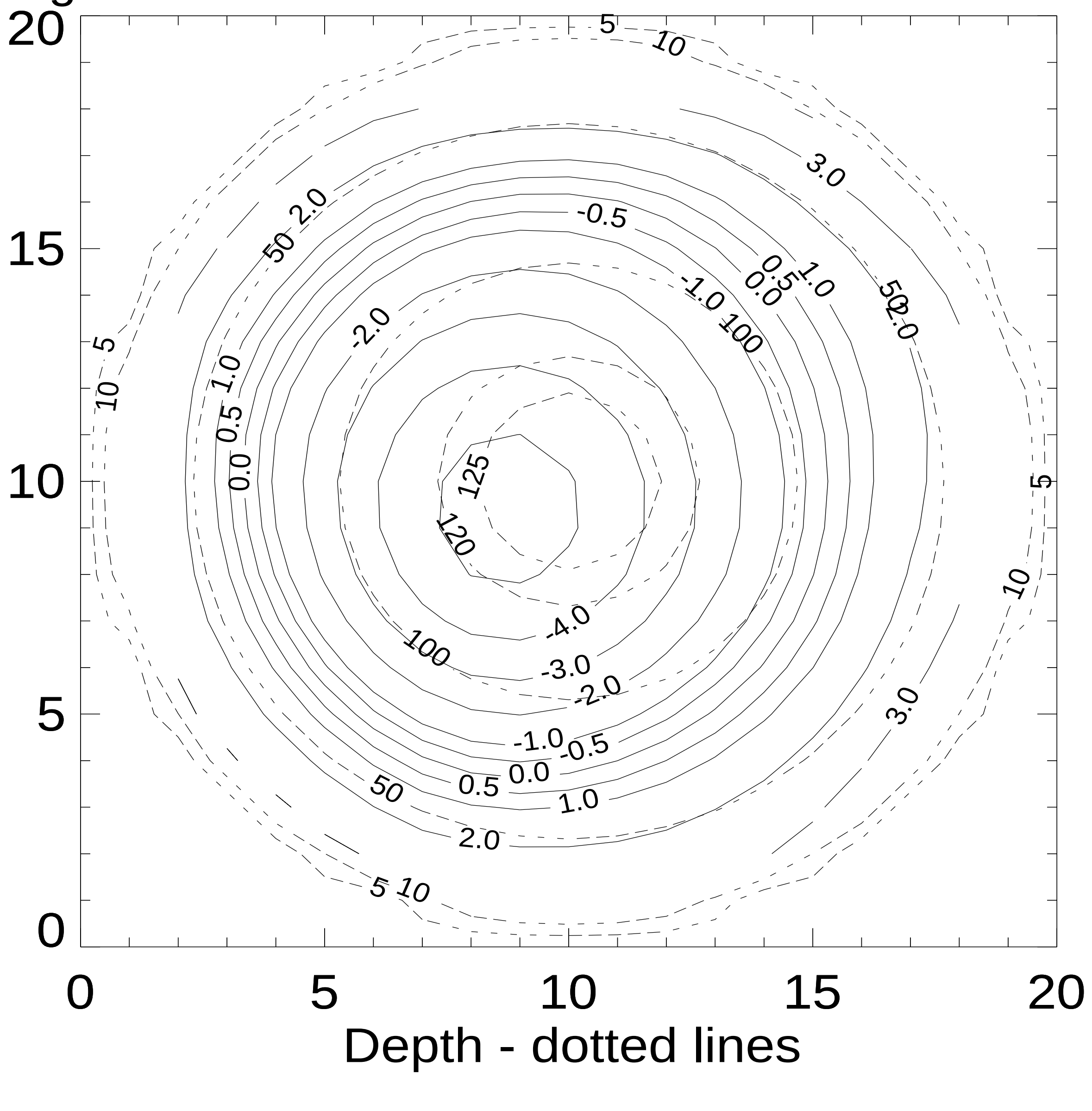
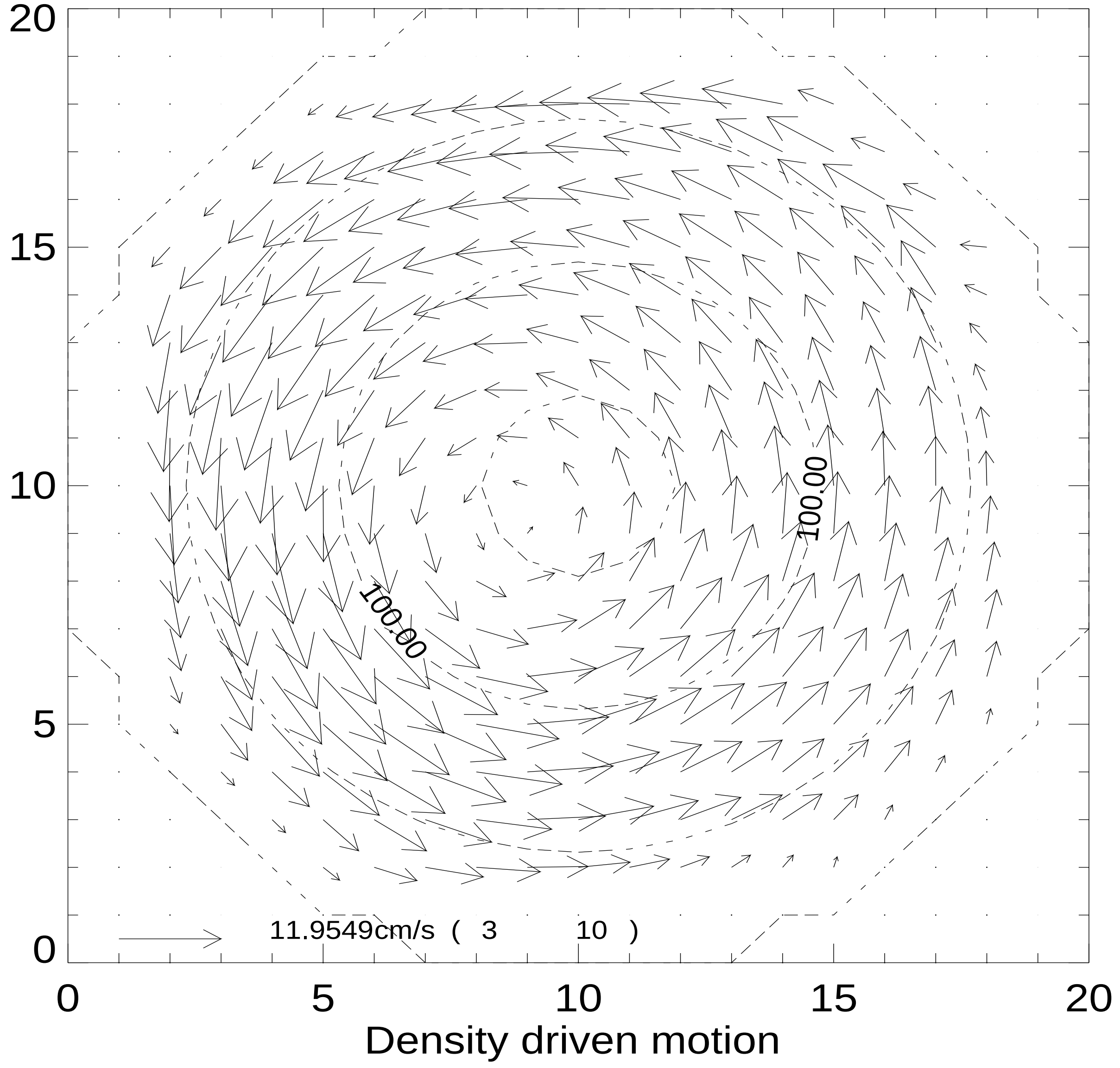


Fig.III.13: Surface Current - Day 11.5



APPENDIX

BASIC MATHEMATICAL FORMULAS

RUDIMENTS OF TRIGONOMETRY

1. **Trigonometric Functions**

$$\sin^2 x + \cos^2 x = 1$$

$$\sin x = \frac{\tan x}{\sqrt{1+\tan^2 x}} \qquad \cos x = \frac{1}{\sqrt{1+\tan^2 x}}$$

$$\sin 2x = 2\sin x \cos x \qquad \cos 2x = \cos^2 x - \sin^2 x$$

$$\sin 3x = 3\sin x - 4\sin^3 x \qquad \cos 3x = 4\cos^3 x - 3\cos x$$

$$1 - \cos x = 2\sin^2 \frac{x}{2} \qquad 1 + \cos x = 2\cos^2 \frac{x}{2}$$

$$\sin(x \pm y) = \sin x \cos y \pm \cos x \sin y$$

$$\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y$$

2. **Hyperbolic Functions**

$$\sinh x = \frac{(e^x - e^{-x})}{2} \qquad \cosh x = \frac{(e^x + e^{-x})}{2}$$

$$\cosh^2 x - \sinh^2 x = 1$$

$$\sinh 2x = 2\sinh x \cosh x \qquad \cosh 2x = \cosh^2 x + \sinh^2 x$$

$$\cosh x - 1 = 2\sinh^2 \frac{x}{2} \qquad \cos x + 1 = 2\cosh^2 \frac{x}{2}$$

$$\sin z = -i \sinh iz \qquad \cos z = \cosh iz$$

$$\sinh z = -i \sin iz \qquad \cosh z = \cos iz$$

RUDIMENTS OF COMPLEX NUMBERS THEORY

1. The **complex number** z is defined by setting

$$z = a + bi,$$

where $\mathbf{a}$ and $\mathbf{b}$ are real numbers and $\mathbf{i}$ is the imaginary unit: $\mathbf{i^2 = -1}$.

2. The **complex conjugate** $\bar{z}$ of a complex number $z = a + bi$ is defined by setting

$$\bar{z} = a - bi;$$

$$z\bar{z} = (a + bi)(a - bi) = a^2 + b^2$$

3. **To add or subtract** complex numbers, we add or subtract the real and imaginary
parts separately:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

Multiplication is defined as follows:

$$(a + bi)(c + di) = (ac - bd) + (ad + bc)i$$

To divide one complex number by another we multiply a numerator and a denominator by the complex conjugate of the denominator:

$$\frac{a + bi}{c + di} = \frac{(a + bi)(c - di)}{(c + di)(c - di)} = \frac{ac + bd}{c^2 + d^2} + i\frac{bc - ad}{c^2 + d^2}$$

4. **Trigonometric form** of a complex number:

$$z = r(\cos\phi + i\sin\phi),$$

where $r = \sqrt{a^2 + b^2}$ is called an absolute value of a complex number and
$\phi = \arctan\frac{b}{a}$ is called an argument of a complex number.

5. **Exponential form** of a complex number:

$$z = re^{i\phi}$$

$$z = re^{i\phi} = r(\cos\phi + i\sin\phi)$$

$$\bar{z} = re^{-i\phi} = r(\cos\phi - i\sin\phi)$$

$$\sin\phi = \frac{(e^{i\phi} - e^{-i\phi})}{2i} \qquad \cos\phi = \frac{(e^{i\phi} + e^{-i\phi})}{2}$$

RUDIMENTS OF ALGEBRA

1. **Quadratic Equation.**

$$ax^2 + bx + c = 0$$

$$D = b^2 - 4ac$$

$$x_1 = \frac{-b + \sqrt{D}}{2a} \qquad x_2 = \frac{-b - \sqrt{D}}{2a}$$

If $D > 0$, the roots are real; $x_1 \neq x_2$.
If $D = 0$, the roots are real; $x_1 = x_2$.
If $D < 0$, the roots are complex conjugates.

$$ax^2 + bx + c = a(x - x_1)(x - x_2)$$

2. The **linear homogeneous algebraic** system of equations

$$\begin{cases} a_1x + b_1y = 0 \\ a_2x + b_2y = 0 \end{cases}$$

has a solution only if its determinant is equal to zero:

$$\det = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1b_2 - a_2b_1 = 0.$$

The same approach is valid for the 3-rd order linear algebraic system of equations:

$$\begin{cases} a_1x + b_1y + c_1z = 0 \\ a_2x + b_2y + c_2z = 0 \\ a_3x + b_3y + c_3z = 0 \end{cases}$$

$$1)\quad \det = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = a_1b_2c_3 + a_3b_1c_2 + a_2b_3c_1 - a_3b_2c_1 - a_1b_3c_2 - a_2b_1c_3 = 0.$$

$$2)\quad \det = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = a_1\begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix} - b_1\begin{vmatrix} a_2 & c_2 \\ a_3 & c_3 \end{vmatrix} + c_1\begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix} = 0.$$

RUDIMENTS OF POWER SERIES AND INTEGRATION

1. **Newton's binomial series**

For any constant s

$$(a+b)^s = a^s + sa^{s-1}b + \frac{s(s-1)}{2!}a^{s-2}b^2 + \frac{s(s-1)(s-2)}{3!}a^{s-3}b^3 + \cdots$$

Especially useful when a or b is small, e.g., if $x \to 0$, then,

$$(1-x)^{-1} \simeq 1 + x$$

2. **Maclaurin series**

$$e^x = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + ...$$

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + ...$$

$$\cos x = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 + ...$$

3. Integral with variable limits

$$\frac{\partial}{\partial x}\int_a^b f(x,z)dz = f(b)\frac{\partial b}{\partial x} - f(a)\frac{\partial a}{\partial x} + \int_a^b \frac{\partial f(x,z)}{\partial x}dz$$