A Turbulent Boundary Layer Model for the Linearized Shallow Water Equations

NUBBLE USER'S MANUAL (Release 1.1)

Christopher E. Naimie

Thayer School of Engineering

Dartmouth College

Hanover, NH 03755, USA

July 31, 1996

Report NML-96-1

Contents

List of Figures

Chapter 1

Introduction

NUBBLE is a 1-D (z) time-stepping point model which uses linear finite elements to determine the vertical structure of the horizontal components of velocity (u, v) and density (ρ) under specified surface forcing. The governing equations are the linearized shallow water equations (see Appendix A). Two forms of turbulence closure are implemented during every numerical simulation (see Appendix A for a detailed discussion). First, the vertical structure of the horizontal components of velocity (u, v) is computed using the quadratic closure based on the vertically averaged velocity from *Davies and Furnes* (1980) (hereinafter DF80 closure or quadratic eddy viscosity closure). Second, the vertical structure of the horizontal components of velocity (u, v) and density (ρ) are computed using the level 2.5 closure from *Mellor and Yamada* (1974,1982), *Galperin et al.* (1988), and *Blumberg et al.* (1992) (hereinafter level 2.5 closure or advanced turbulence closure). Calculation of the velocity solution using these two turbulence closure schemes provides information regarding the sensitivity of results to the vertical mixing parameterization.

NUBBLE was initially developed to promote understanding of the features and limitations of the level 2.5 closure, especially in the treatment of the boundary conditions for the turbulent master length scale (which is a known weakness of the level 2.5 turbulence closure [*Mellor and Yamada* (1982)]) along with discretization and convergence issues. *Naimie* (1995), *Lynch et al.* (1995) detail conclusions drawn from these experiments with NUBBLE.

The programming language for NUBBLE is FORTRAN 77.

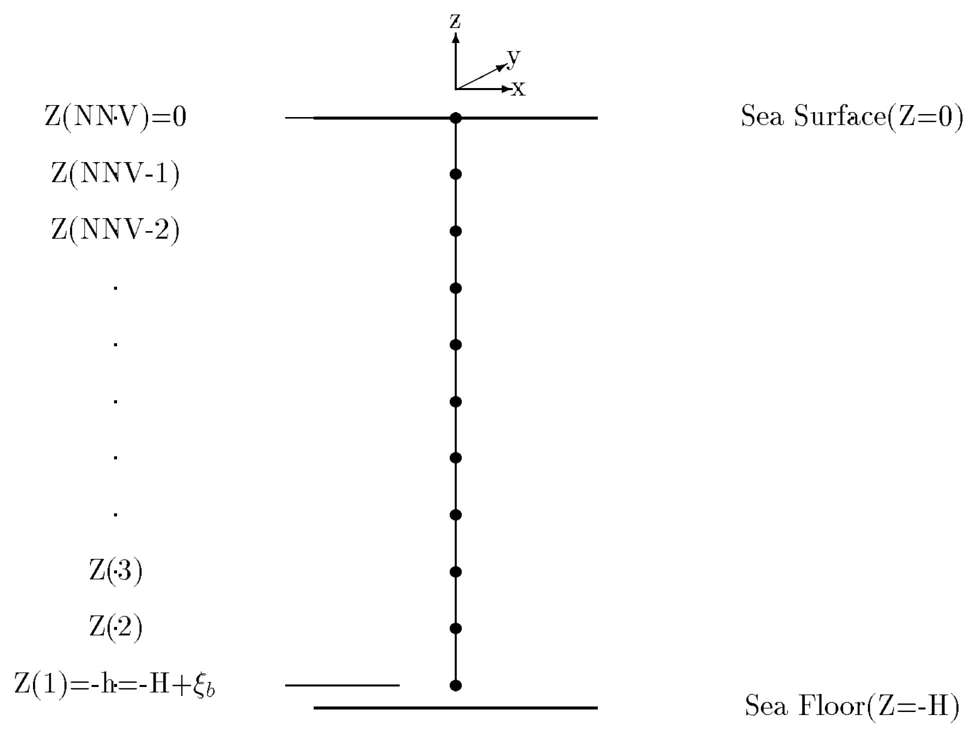

Figure 1.1: Example vertical discretization for NUBBLE. Solid circles indicate node locations. Node numbers increase from 1 at the bottom of the modeled water column (where the quadratic bottom slip condition is applied) to NNV at the sea surface.

Chapter 2

Model Overview for NUBBLE; Release 1.1

2.1 Program Structure

Main Program and Fixed Subroutines - NUBBLE is the core program, which performs all finite element assembly and solution operations, according to Appendix A (see Figure 2.1).

User Subroutines - Three user-built subroutines must be linked to NUBBLE, each of which performs a particular task. NUBBLE.USER.f is a shell from which these subroutines may be constructed (see Figure 2.1):

- SUBROUTINE INITIALIZEN1 - intialize the model (i.e. specify the physical parameters, the vertical discretization, and the initial conditions for the dependent variables),
- SUBROUTINE SURFACEN1 - set the surface forcing during each time-step,
- SUBROUTINE OUTPUTN1 - specify the manner in which the results are to be written.

Include File - The include file NUBBLE.DIM assigns values to parameters required for dimensioning purposes and SI (MKS) values of the reference density and gravitational acceleration (see Figure 2.1).

2.2 User Interface

Running NUBBLE - All interaction with NUBBLE during execution is dependent on the user's treatment of the User Subroutines.

Creating a Set of User Subroutines - Specifications for each of the user subroutines (INITIALIZEN1, SURFACEN1, and OUTPUTN1) is available in its respective shell. These shells (see Figures 2.2, 2.3, and 2.4) contain comments regarding the overall structure, argument list, dimensioning, and declarations which are sufficient to serve as starting points for their construction.

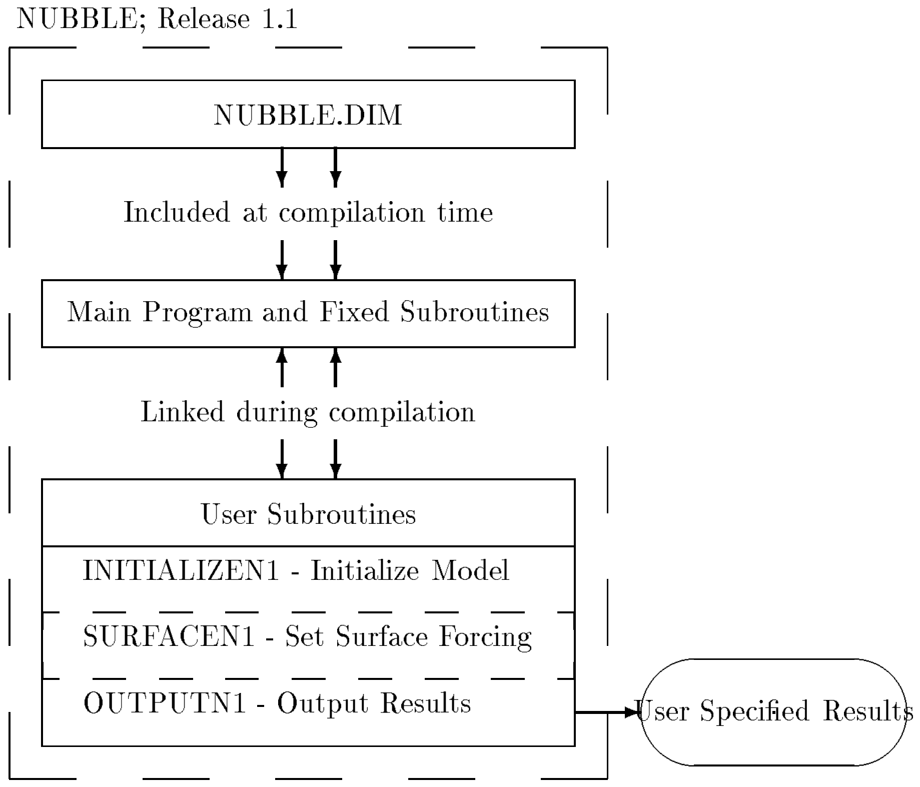
NUBBLE; Release 1.1
NUBBLE.DIM
Included at compilation time
Main Program and Fixed Subroutines
Linked during compilation
User Subroutines
INITIALIZEN1 - Initialize Model
SURFACEN1 - Set Surface Forcing
OUTPUTN1 - Output Results
User Specified Results

Figure 2.1: NUBBLE program structure

```
C**********************************************************************
C**********************************************************************
      SUBROUTINE INTITALIZEN1(DT,ITMAX,BAT,Zlog,f,CD,ENZDFMIN,
     &NNV,Z,UT,VT,U,V,RHO,Q2,Q2L)
C Include file for parameter statements
      INCLUDE 'NUBBLE.DIM'
C Dimension global arrays
      REAL Z(NNVDIM),UT(NNVDIM),VT(NNVDIM)
      REAL U(NNVDIM),V(NNVDIM),RHO(NNVDIM),Q2(NNVDIM),Q2L(NNVDIM)
C----------------------------------------------------------------------
C This user subroutine is provided for initialization purposes.  It is
C  called at the beginning of the numerical simulation to:
C   set time stepping information,
C   set physical parameters,
C   set the vertical grid,
C   initialize dependent variables.
C
C Time stepping information:
C   DT   - size of time step [s]
C   ITMAX- maximum number of time steps for the simulation
C
C Physical parameters:
C   BAT  - depth below the surface to the position where the bottom slip
C           condition is applied [m]
C   Zlog - height above the sea-floor at which the bottom slip condition
C           is applied [m]
C   CD   - quadratic bottom slip coefficient (0.005 is suggested)
C   f    - coriolis parameter [1/s]
C   ENZDFMIN - minimum vertical eddy viscosity for DF80 turbulence
C               closure [m^2/s]
C
C Vertical discretization information:
C   NNV  - number of vertical nodes (note:  J indexes from 1 to NNV)
C   Z(J) - vertical descritization array (Z(1)=-BAT,.,Z(NNV)=0.0) [m]
C
C Dependent variables initialized for the DF80 turbulence closure:
C   UT(J),VT(J) - horizontal velocities [m/s]
C
C Dependent variables initialized for the level 2.5 turbulence closure:
C   U(J),V(J)   - horizontal velocities [m/s]
C   RHO(J)      - (perturbation density-reference density)/reference
C                  density [unitless]
C   Q2(J)       - twice the turbulent kinetic energy [m^2/s^2]
C   Q2L(J)      - twice the turbulent kinetic energy times the master
C                  length scale [m^3/s^2]
C
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   NUBBLE Release 1.1 - July 31, 1996
C----------------------------------------------------------------------
C Two standard subroutines are available for the vertical grid
C  assignment.
C
C  1. Subroutine UNIZ assigns uniform vertical spacing.
C     Assignment of NNV and BAT are required before this subroutine is
C     called.  The form of the call statement is:
C
C      CALL UNIZ(NNVDIM,NNV,BAT,Z)
C
C  2. Subroutine SINEZ assigns sinusiodal vertical spacing, with the
C     discretization at the extremes of the water column set to
C     DZBL [m].  It requires assignment of NNV, DZBL, and BAT before it
C     is called.  The form of the call statement is:
C
C      CALL SINEZ(NNVDIM,NNV,DZBL,BAT,Z)
C===SUBROUTINE INITIALIZEN1: Beginning-of-user-specified-instructions===
      RETURN
      END
C**********************************************************************
C**********************************************************************
```

Figure 2.2: User Subroutine Shell for SUBROUTINE INITIALIZEN1.

```
C***********************************************************************
C***********************************************************************
      SUBROUTINE SURFACEN1(ITER,TIME,RHOSURF,RHOFLUX,GX,GY,WX,WY)
C Include file for parameter statements
      INCLUDE 'NUBBLE.DIM'
C-----------------------------------------------------------------------
C This subroutine is provided for specification of nonzero surface
C  forcing.  It is called every time step:
C
C Provided by calling routine:
C   ITER - current iteration number (ranges from 1 to ITMAX)
C   TIME - time at the middle of the current time step (i.e. the
C            time when this subroutine is called) [s]
C          note that TIME=0.0 at the beginning of the simulation
C   RHOSURF - the surface value of RHO at the beginning of the current
C              time step [unitless]
C
C Surface nonzero conditions set within this subroutine during each time
C  step (note:  just prior to the call to this subroutine, all of the
C  following variables are set to zero in the main program):
C   RHOFLUX - the surface flux of rho (kh*drho/dz) [kg/(s m^2)]
C   (GX,GY) - (x,y) components of the surface pressure gradient [m/s^2]
C              (note (GX,GY) are positive on the RHS of the
C              momentum equation => (GX,GY)=(-g*dzeta/dx,-g*dzeta/dy))
C   (WX,WY) - (x,y) components of the surface wind stress divided by
C              the reference density [m^2/s^2]
C
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   NUBBLE Release 1.1 - July 31, 1996
C=====SUBROUTINE SURFACEN1: Beginning-of-user-specified-instructions====
      RETURN
      END
C***********************************************************************
C***********************************************************************
```

Figure 2.3: User Subroutine Shell for SUBROUTINE SURFACEN1.

```
C*********************************************************************
C*********************************************************************
      SUBROUTINE OUTPUTN1(DT,ITMAX,BAT,Zlog,f,CD,ENZDFMIN,
     &ITER,TIME,RHOFLUX,GX,GY,WX,WY,
     &NNV,Z,UT,VT,ENZ,U,V,RHO,Q2,Q2L,ENZM,ENZH,ENZQ)
C Include file for parameter statements
      INCLUDE 'NUBBLE.DIM'
C Dimension global arrays
      REAL Z(NNVDIM), UT(NNVDIM),VT(NNVDIM),ENZ(NNVDIM)
      REAL U(NNVDIM),V(NNVDIM),RHO(NNVDIM),Q2(NNVDIM),Q2L(NNVDIM),
     &ENZM(NNVDIM),ENZH(NNVDIM),ENZQ(NNVDIM)
C---------------------------------------------------------------------
C This subroutine is used to write any combination of parameters and
C  3-D quantities; all of which are provided by the calling routine:
C
C Time stepping parameters:
C   DT  - size of time step (s)
C   ITMAX - maximum number of time steps for the current simulation
C
C Physical parameters:
C   BAT  - depth below the surface to the position where the bottom slip
C            condition is applied [m]
C   Zlog - height above the sea-floor at which the bottom slip condition
C            is applied [m]
C   CD   - quadratic bottom slip coefficient (0.005 is suggested)
C   f    - coriolis parameter [1/s]
C   ENZDFMIN - minimum vertical eddy viscosity for DF80 turbulence
C                closure [m^2/s]
C
C Time stepping parameters which indicate the current time:
C   ITER - current time step number (ranges from 1 to ITMAX)
C   TIME - time at the end of the current time step (i.e. the time
C            when this subroutine is called) [s]
C          note: TIME=0.0 at the beginning of the simulation
C
C Surface forcing for the current time step:
C   RHOFLUX - the surface flux of rho (kh*drho/dz) [kg/(s m^2)]
C   (GX,GY) - (x,y) components of the surface pressure gradient [m/s^2]
C               (note (GX,GY) are positive on the RHS of the
C               momentum equation => (GX,GY)=(-g*dzeta/dx,-g*dzeta/dy))
C   (WX,WY) - (x,y) components of the surface wind stress divided by
C               the reference density [m^2/s^2]
C
C Vertical discretization information:
C   NNV  - number of vertical nodes (note:  J indexes from 1 to NNV)
C   Z(J) - vertical descritization array (Z(1)=-BAT,.,Z(NNV)=0.0) [m]
C
C Current values of dependent variables for the DF80 turbulence closure:
C   UT(J),VT(J) - horizontal velocities [m/s]
C   ENZ(J)      - vertical eddy viscosity [m^2/s]
C
C Current values of dependent variables for the level 2.5 turbulence
C  closure:
C   U(J),V(J)   - horizontal velocities [m/s]
C   RHO(J)      - (perturbation density-reference density)/reference
C                   density [unitless]
C   Q2(J)       - twice the turbulent kinetic energy [m^2/s^2]
C   Q2L(J)      - twice the turbulent kinetic energy times the master
C                   length scale [m^3/s^2]
C   ENZM(J)     - vertical eddy viscosity [m^2/s]
C   ENZH(J)     - vertical heat diffusivity [m^2/s]
C   ENZQ(J)     - vertical diffusivity for turbulent quantities [m^2/s]
C
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   NUBBLE Release 1.1 - July 31, 1996
C=====SUBROUTINE OUTPUTN1: Beginning-of-user-specified-instructions=====
      RETURN
      END
C*********************************************************************
C*********************************************************************
```

Figure 2.4: User Subroutine Shell for SUBROUTINE OUTPUTN1.

Chapter 3

Examples

3.1 Surface Ekman Dynamics for a Homogeneous Fluid

Simulation Objectives:

- Initialize model with quiescent, yet quite turbulent, 100 m deep water column,
- Ramp up wind forcing to a steady breeze of 0.1414 Pa directed toward 45° counter-clockwise from true north (i.e. (τ_x, τ_y)=(0.1 Pa, 0.1 Pa)),
- Set minimum eddy viscosity for quadratic eddy viscosity closure to 0.02 m^2 s^{-1} to yield a "more realistic" ekman response than is predicted by this closure,
- Report results after a converged solution is obtained.

NUBBLE Results: See Figure 3.1

User-Specified-Instruction sections of User Subroutines:

- SUBROUTINE INITIALIZEN1:

```
C===SUBROUTINE INITIALIZEN1: Beginning-of-user-specified-instructions===
C Set physical and time-stepping parameters
C  (BAT,Zlog,CD,f,ENZDFMIN,DT,ITMAX):
      BAT=99.0
      Zlog=1.0
      CD=0.005
      f=0.0001
      ENZDFMIN=0.02
      DT=120.0
      ITMAX=20000
C Set the vertical grid
C  (Recall that Z(J) increases from Z(1)=-BAT to Z(NNV)=0.0)
      NNV=21
      DZBL=1.0
      CALL SINEZ(NNVDIM,NNV,DZBL,BAT,Z)
C Initialize dependent variables
      DO J=1,NNV
         UT(J)=0.0
         VT(J)=0.0
         U(J)=0.0
         V(J)=0.0
         RHO(J)=0.0
         Q2(J)=0.001
         Q2L(J)=0.001
      ENDDO
      RETURN
      END
```

- SUBROUTINE SURFACEN1:

```
C=====SUBROUTINE SURFACEN1: Beginning-of-user-specified-instructions====
C Ramp up nonzero surface wind stress/reference density
      RAMPUP=AMIN1(REAL(ITER)/1000.0,1.0)
      PASCALX=0.1
      PASCALY=0.1
      WX=RAMPUP*PASCALX/RHOREF
      WY=RAMPUP*PASCALY/RHOREF
      RETURN
      END
```

- SUBROUTINE OUTPUTN1:

```
C=====SUBROUTINE OUTPUTN1: Beginning-of-user-specified-instructions=====
C
C Declarations for internal file
      CHARACTER*72 INTFILE,CITER
C
C Convert ITER to a character string (CITER)
      WRITE(INTFILE,'(I10)')ITER
      READ(INTFILE,'(A)')CITER
C
C Locate beginning and end of CITER
      DO I=1,72
         ISTART=I
 IF(CITER(I:I).NE.' ')GO TO 11
      ENDDO
      WRITE(*,'(A)')'Error finding the beginning of CITER in OUTPUTN1'
 11   DO I=ISTART,72
         IEND=I-1
 IF(CITER(I:I).EQ.' ')GO TO 21
      ENDDO
      WRITE(*,'(A)')'Error finding the end of CITER in OUTPUTN1'
 21   CONTINUE
C
C Output results when simulation is 95% completed and at the end of the
C   simulation
      IF(ITER.NE.NINT(0.95*REAL(ITMAX)).AND.(ITER.NE.ITMAX))RETURN
      OPEN(1,FILE=CITER(istart:iend)//'.n1n',STATUS='UNKNOWN')
      REWIND(1)
      DO J=1,NNV
          WRITE(1,'(100e12.4)')Z(J),UT(J),VT(J),ENZ(J),
     &    U(J),V(J),Q2(J),Q2L(J),Q2L(J)/Q2(J),RHO(J)
      ENDDO
      CLOSE(1)
      OPEN(2,FILE=CITER(istart:iend)//'.n1e',STATUS='UNKNOWN')
      REWIND(2)
      DO J=1,NNV-1
         WRITE(2,'(100e12.4)')z(J),ENZM(J),ENZH(J),ENZQ(J)
         WRITE(2,'(100e12.4)')z(J+1),ENZM(J),ENZH(J),ENZQ(J)
      ENDDO
      CLOSE(2)
      RETURN
      END
```

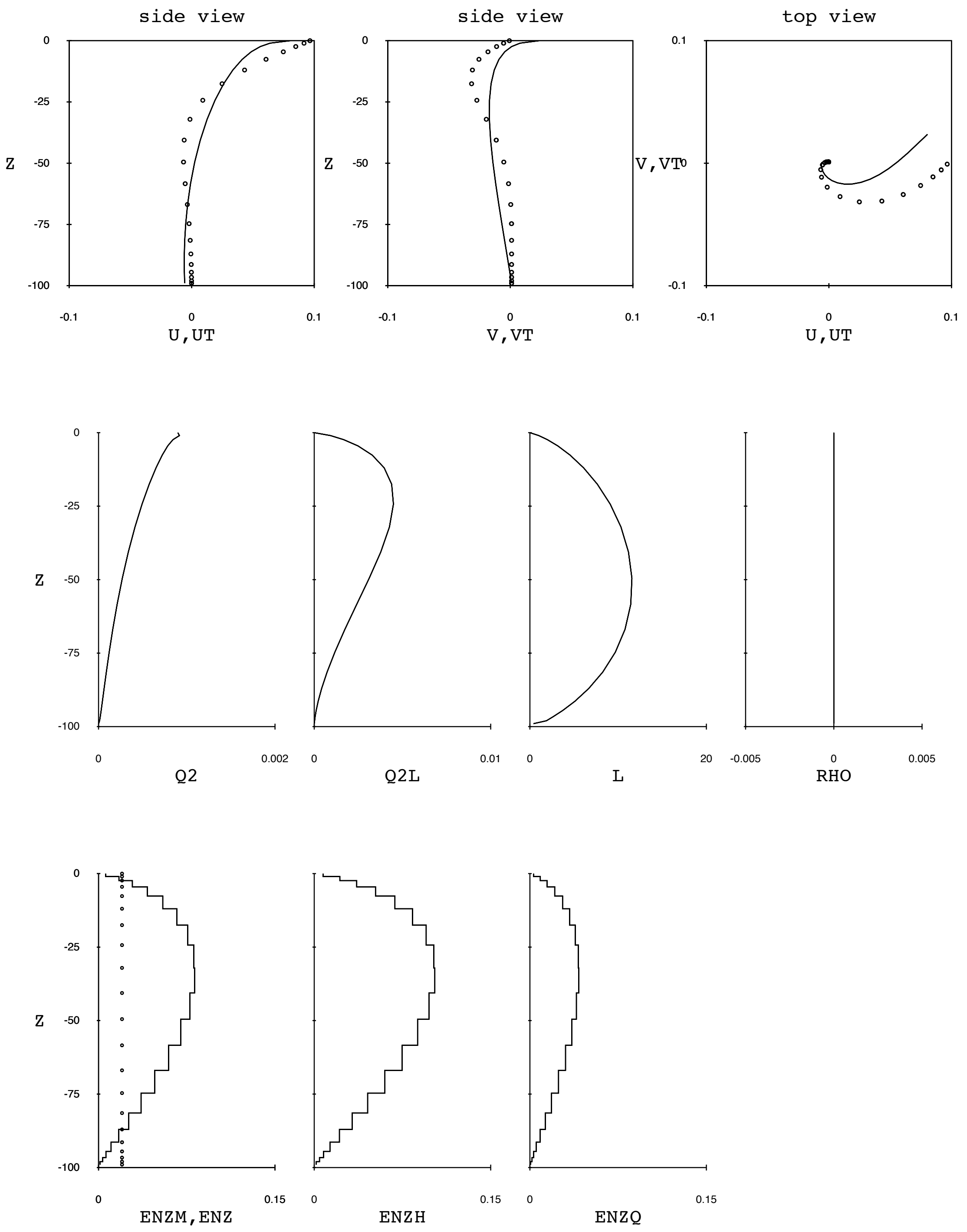

Figure 3.1: NUBBLE results for Example 1 - Surface Ekman Dynamics for a Homogeneous Fluid - The solid lines (circles) represent results computed using the 1-D time-domain model NUBBLE with advanced turbulence closure (a constant eddy viscosity of 0.02 m^2s^{-1}). Results are plotted for iterations 19000 and 20000. The fact that the plots for these two iterations are indistinguishable indicates that the solution is converged. MKS Units.

3.2 Bottom Boundary Layer Dynamics for a Homogeneous Fluid

Simulation Objectives:

- Initialize model with quiescent, yet quite turbulent, 100 m deep water column,
- Ramp up the surface pressure gradient forcing to a steady value which would result in an x-directed current of 0.30 m s^{-1} under a geostrophic balance,
- Report results after a converged solution is obtained.

NUBBLE Results: See Figure 3.2

User-Specified-Instruction sections of User Subroutines:

- `SUBROUTINE INITIALIZEN1` is identical to that for Example 3.1, with the exception that `ENZDFMIN` is set to 0.001 m^2 s^{-1}. This modification is made because the quadratic eddy viscosity response predicts a realistic response under the type of forcing employed here.
- `SUBROUTINE SURFACEN1`:

```
C=====SUBROUTINE SURFACEN1: Beginning-of-user-specified-instructions====
C
C Ramp up nonzero surface pressure gradient
      RAMPUP=AMIN1(REAL(ITER)/1000.0,1.0)
      f=0.0001
      UGEO=0.30
      VGEO=0.00
      GX=-RAMPUP*f*VGEO
      GY=+RAMPUP*f*UGEO
      RETURN
      END
C**********************************************************************
```

- `SUBROUTINE OUTPUTN1` is identical to that for Example 3.1.

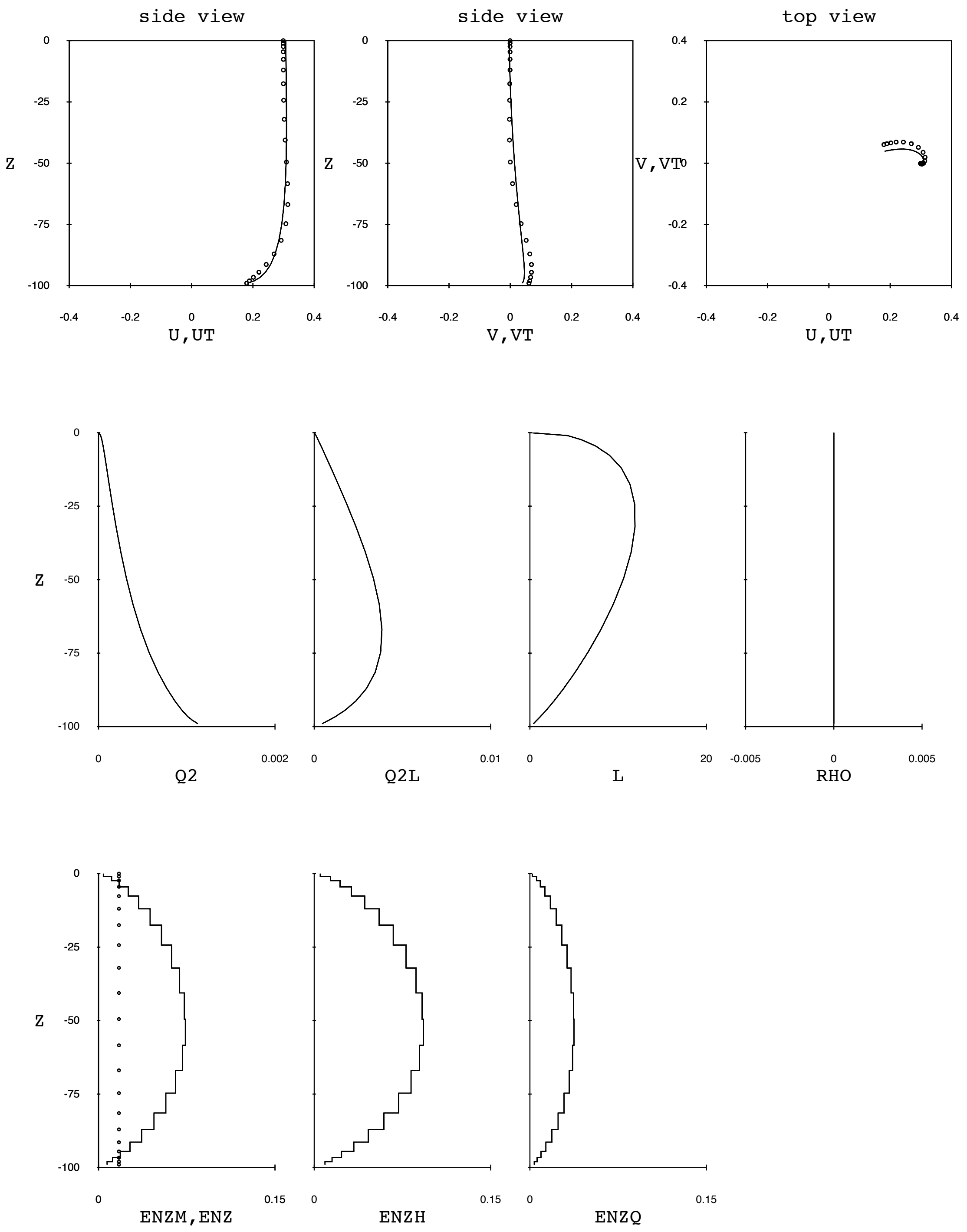

Figure 3.2: NUBBLE results for Example 2 - Bottom Boundary Layer Dynamics for a Homogeneous Fluid - The solid lines (circles) represent results computed using the 1-D time-domain model NUBBLE with advanced (quadratic eddy viscosity) turbulence closure. Results are plotted for iterations 19000 and 20000. The fact that the plots for these two iterations are indistinguishable indicates that the solution is converged. MKS Units.

3.3 Surface and Bottom Boundary Layer Dynamics for a Homogeneous Fluid

Simulation Objective: Combine the forcing for Examples 3.1 and 3.2 to examine the interaction of the surface and bottom boundary layers.

NUBBLE Results: See Figure 3.3

User-Specified-Instruction sections of User Subroutines:

- `SUBROUTINE INITIALIZEN1` is identical to that for Example 3.2.
- `SUBROUTINE SURFACEN1`: The user specified instruction set for User Subroutine `SURFACEN1` is the union of those for Examples 3.1 and 3.2:

```
C=====SUBROUTINE SURFACEN1: Beginning-of-user-specified-instructions====
C
C Ramp up nonzero surface pressure gradient and surface wind stress
C  divided by reference density
      RAMPUP=AMIN1(REAL(ITER)/1000.0,1.0)
      f=0.0001
      UGEO=0.30
      VGEO=0.00
      GX=-RAMPUP*f*VGEO
      GY=+RAMPUP*f*UGEO
      PASCALX=0.1
      PASCALY=0.1
      WX=RAMPUP*PASCALX/RHOREF
      WY=RAMPUP*PASCALY/RHOREF
      RETURN
      END
```

- `SUBROUTINE OUTPUTN1` is identical to that for Example 3.2.

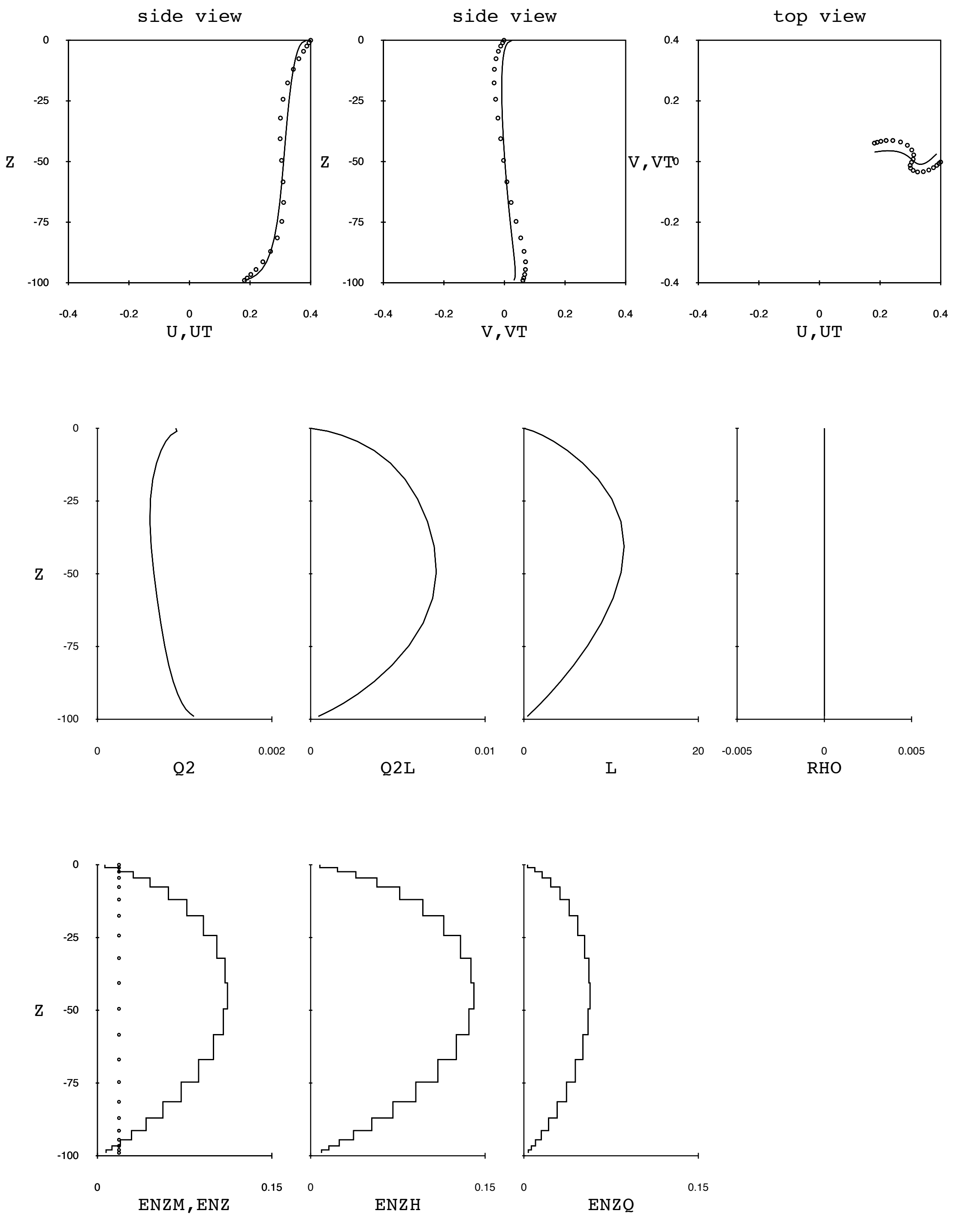

Figure 3.3: NUBBLE results for Example 3 - Surface and Bottom Boundary Layer Dynamics for a Homogeneous Fluid - The solid lines (circles) represent results computed using the 1-D time-domain model NUBBLE with advanced (quadratic eddy viscosity) turbulence closure. Results are plotted for iterations 19000 and 20000. The fact that the plots for these two iterations are indistinguishable indicates that the solution is converged. MKS Units.

3.4 Barotropic M_2 Tidal Forcing and Steady Wind Forcing on Georges Bank for a Homogeneous Fluid

Simulation Objective: In the final example, we use NUBBLE to compute the vertical profiles of the dependent variables for a location on the southern flank of Georges Bank (Site A in Figure 3.4) under winter conditions.

Geographic Location of Site A on Georges Bank: See Figure 3.4

NUBBLE Results: See Figure 3.5

User-Specified-Instruction sections of User Subroutines:

- SUBROUTINE INITIALIZEN1 is modified to set physical parameters for Site A on the southern flank of Georges Bank, dictate a 20 tidal period simulation with 96 time steps per tidal period, set the vertical grid, and initialize the model with a quiescent, yet quite turbulent, water column:

```
C===SUBROUTINE INITIALIZEN1: Beginning-of-user-specified-instructions===
C
C Tidal timing information common block
      COMMON/TIMING/TPRD,NPERPRD,NPRD
C
C Set physical and time-stepping parameters
C  (BAT,Zlog,CD,f,ENZDFMIN,DT,ITMAX):
      BAT=81.93
      Zlog=1.0
      CD=0.005
      DLAT=43.5
      PI=ATAN2(0.0,-1.0)
      f=2.0*2.0*pi/24.0/3600.0*SIN(DLAT*PI/180.0)
      ENZDFMIN=0.001
      TPRD=12.42
      NPERPRD=96
      NPRD=20
      DT=TPRD*3600.0/REAL(NPERPRD)
      ITMAX=NPERPRD*NPRD
C
C Set the vertical grid
C  (Recall that Z(J) increases from Z(1)=-BAT to Z(NNV)=0.0)
      NNV=21
      DZBL=1.0
      CALL SINEZ(NNVDIM,NNV,DZBL,BAT,Z)
C
C Initialize dependent variables
      DO J=1,NNV
         UT(J)=0.0
         VT(J)=0.0
         U(J)=0.0
         V(J)=0.0
         RHO(J)=0.0
         Q2(J)=0.001
         Q2L(J)=0.001
      ENDDO
      RETURN
      END
```

- SUBROUTINE SURFACEN1 is modified to ramp up the barotropic M_2 tidal forcing for this location to that computed during an earlier numerical simulation of the regional 3-D barotropic M_2 tidal response and the wind stress to the climatological value for

Georges Bank winter conditions (0.0955 Pa, directed 118.5° from true north) reported in *Naimie et al.* (1994):

```
C=====SUBROUTINE SURFACEN1: Beginning-of-user-specified-instructions====
C
C Tidal timing information common block
      COMMON/TIMING/TPRD,NPERPRD,NPRD
C
C Set the components of wind stress divided by reference density
C  (note: wind direction (ANGLECWT) is indicated in terms of the angle
C  (in degrees) toward which the wind is blowing, measured clockwise from
C  true north)
      PASCAL=0.0955
      ANGLECWT=118.5
      PI=ATAN2(0.0,-1.0)
      WX=PASCAL*SIN(ANGLECWT*PI/180.0)/RHOREF
      WY=PASCAL*COS(ANGLECWT*PI/180.0)/RHOREF
C
C Compute time varying surface pressure gradient from harmonic model
C  output
      CALL F4GGRADZT(TIME,GX,GY)
C
C Apply ramp up for surface pressure gradient and surface wind stress
C  divided by reference density
      RAMPUP=AMIN1(REAL(ITER)/(0.25*REAL(NPRD)*REAL(NPERPRD)),1.0)
      GX=RAMPUP*GX
      GY=RAMPUP*GY
      WX=RAMPUP*WX
      WY=RAMPUP*WY
      RETURN
      END
C**********************************************************************
C**********************************************************************
      SUBROUTINE F5GGRADZT(time,GX,GY)
C----------------------------------------------------------------------
C This subroutine computes the time-domain barotropic forcing
C   (-g*gradzeta) at node 1805 resulting from an M2 FUNDY5 run, by
C   converting frequency domain results
C   to the time domain.
C GRHS=-g*gradzeta
C g2s node 1805 is the closest node to SITE A
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   latest revision - 15 March 93
C----------------------------------------------------------------------
C
C Local assignments
      complex eye
C
C Set constants
      eye=CMPLX(0.0,1.0)
      g=9.806
      pi=ATAN2(0.0,-1.0)
      freq=2*pi/12.42/3600.0
      fac=2.0*ASIN(1.0)/180.0
C
C Compute scalar values of zeta at t=time
      dzdx=.146416E-05*EXP(eye*(freq*time-fac*.265250E+03))
      dzdy=.234538E-05*EXP(eye*(freq*time-fac*.926304E+02))
      GX=-g*dzdx
      GY=-g*dzdy
      return
      end
C**********************************************************************
C**********************************************************************
```

- `SUBROUTINE OUTPUTN1` is configured to write results at six regularly spaced intervals during the last tidal period of the simulation. Figure 3.5 displays these results:

```
C=====SUBROUTINE OUTPUTN1: Beginning-of-user-specified-instructions=====
C
C Tidal timing information common block
      COMMON/TIMING/TPRD,NPERPRD,NPRD
C
C Declarations for internal file
      CHARACTER*72 INTFILE,CITER
C
C Convert ITER to a character string (CITER)
      WRITE(INTFILE,'(I10)')ITER
      READ(INTFILE,'(A)')CITER
C
C Locate beginning and end of CITER
      DO I=1,72
         ISTART=I
 IF(CITER(I:I).NE.' ')GO TO 11
      ENDDO
      WRITE(*,'(A)')'Error finding the beginning of CITER in OUTPUTN1'
 11   DO I=ISTART,72
         IEND=I-1
 IF(CITER(I:I).EQ.' ')GO TO 21
      ENDDO
      WRITE(*,'(A)')'Error finding the end of CITER in OUTPUTN1'
 21   CONTINUE
C
C Output results when simulation is 95% completed and at the end of the
C   simulation
      IF((ITMAX-ITER).GE.NPERPRD)RETURN
      IF(MOD((ITMAX-ITER),(NPERPRD/6)).NE.0)RETURN
      OPEN(1,FILE=CITER(istart:iend)//'.n1n',STATUS='UNKNOWN')
      REWIND(1)
      DO J=1,NNV
          WRITE(1,'(100e12.4)')Z(J),UT(J),VT(J),ENZ(J),
     &    U(J),V(J),Q2(J),Q2L(J),Q2L(J)/Q2(J),RHO(J)
      ENDDO
      CLOSE(1)
      OPEN(2,FILE=CITER(istart:iend)//'.n1e',STATUS='UNKNOWN')
      REWIND(2)
      DO J=1,NNV-1
         WRITE(2,'(100e12.4)')z(J),ENZM(J),ENZH(J),ENZQ(J)
         WRITE(2,'(100e12.4)')z(J+1),ENZM(J),ENZH(J),ENZQ(J)
      ENDDO
      CLOSE(2)
      RETURN
      END
```

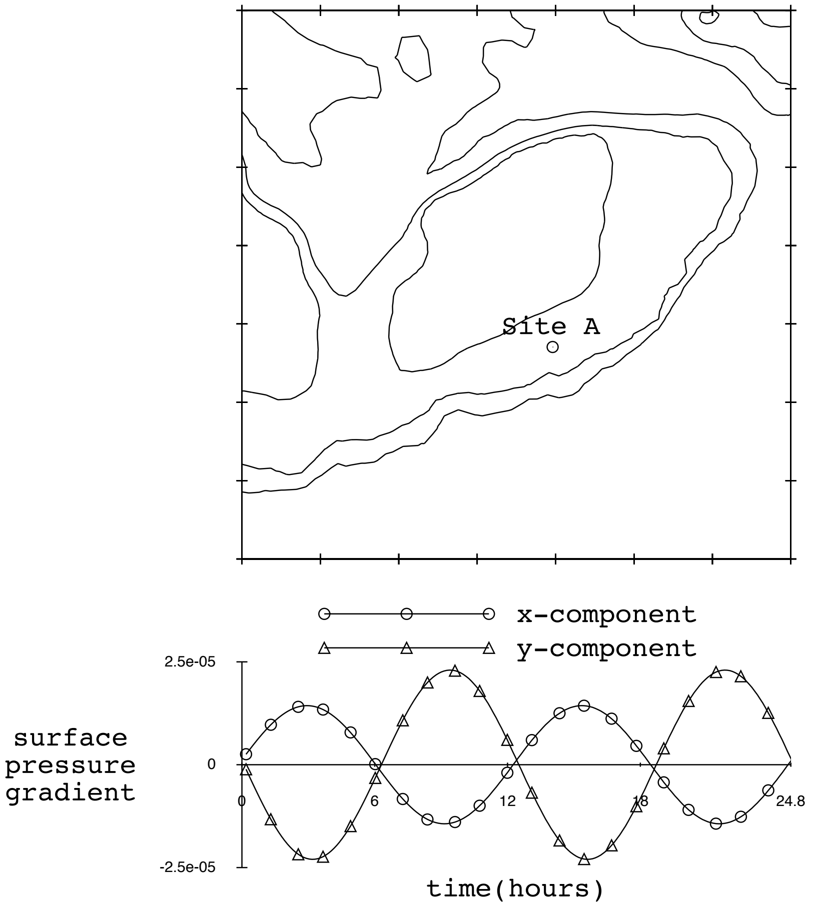

Figure 3.4: Location of Site A on the southern flank of Georges Bank and the barotropic surface pressure gradient forcing. MKS Units

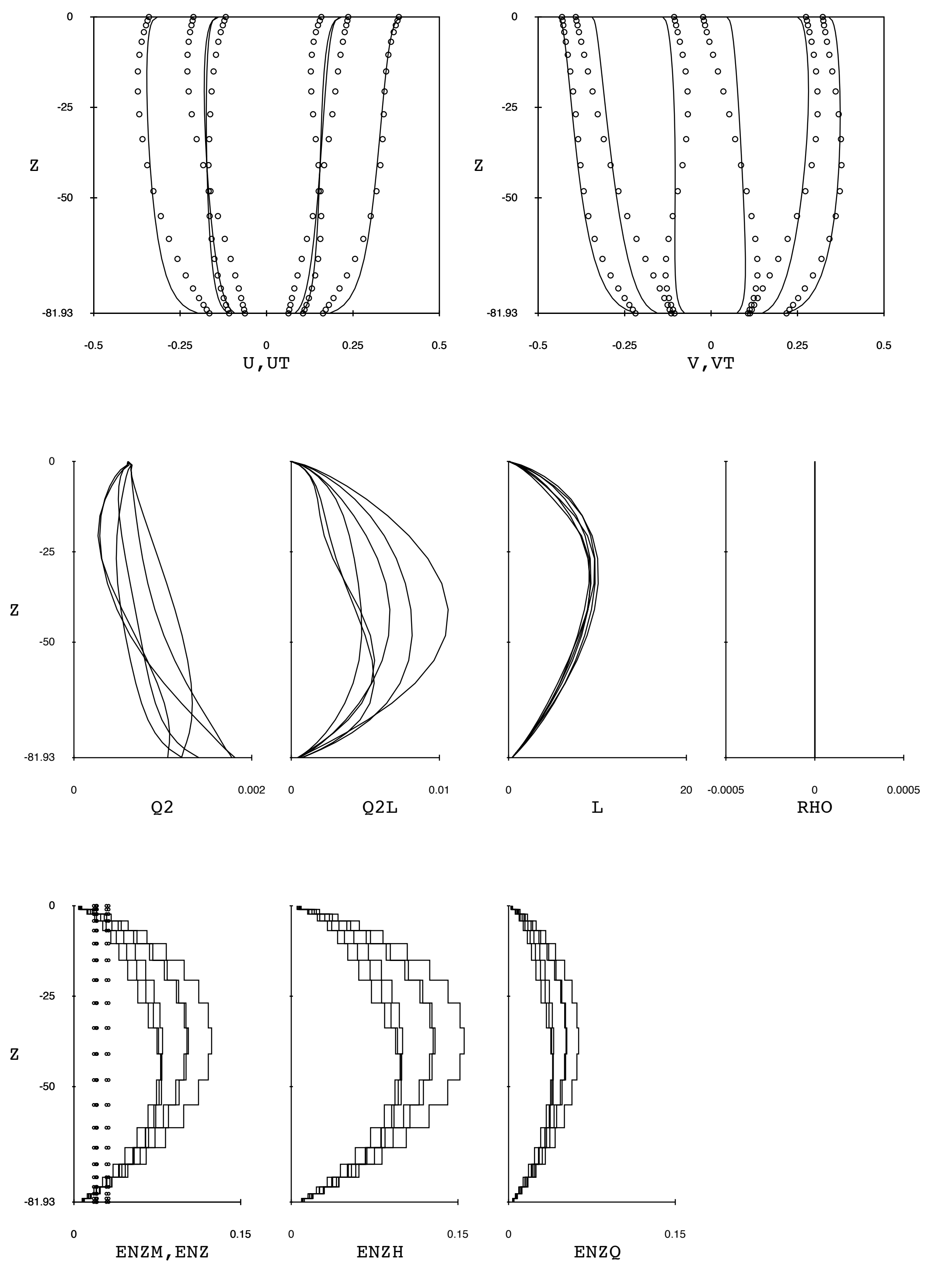

Figure 3.5: Modeled vertical profiles of parameters at Site A on the southern flank of Georges Bank under barotropic forcing at the M2 frequency and steady wind forcing. The solid lines (circles) represent results computed using the 1-D time-domain model NUBBLE with advanced (quadratic eddy viscosity) turbulence closure. MKS Units.

Bibliography

[*Blumberg and Mellor* (1987)]A. F. Blumberg and G. L. Mellor. A description of a three-dimensional coastal ocean circulation model. In Norman S. Heaps, editor, *Three-Dimensional Coastal Ocean Models*, volume 4. American Geophysical Union, Washington, D.C., 1987.

[*Blumberg et al.* (1992)]A. F. Blumberg, B. Galperin, and D. J. O'Connor. Modeling vertical structure of open-channel flows. In *Journal of Hydraulic Engineering*, volume 118, pages 1119–1134. ASCE, 1992.

[*Davies and Furnes* (1980)]A. M. Davies and G. K. Furnes. Observed and computed M2 tidal currents in the North Sea. *J. Phys. Oceanogr.*, 10:237–257, 1980.

[*Davies and Jones* (1990)]A. M. Davies and J. E. Jones. Application of a three-dimensional turbulence energy model to the determination of tidal currents on the northwest european shelf. *J. Geophys. Res.*, 95(C10):18143–18162, 1990.

[*Davies and Jones* (1992)]A. M. Davies and J. E. Jones. A three dimensional wind driven circulation model of the Celtic and Irish seas. *Contin. Shelf Res.*, 12(1):159–188, 1992.

[*Galperin et al.* (1988)]B. Galperin, L. H. Kantha, S. Hassid, and A. Rosati. A quasi-equilibrium turbulent energy model for geophysical flows. *J. Atmos. Sci.*, 45(1):55–62, 1988.

[*Lynch and Naimie* (1993)]D. R. Lynch and C. E. Naimie. The M2 tide and its residual on the outer banks of the Gulf of Maine. *J. Phys. Oceanogr.*, 23:2222–2253, 1993.

[*Lynch and Werner* (1991)]D. R. Lynch and F. E. Werner. Three-dimensional hydrodynamics on finite elements. part II: Nonlinear time-stepping model. *Int. J. Num. Methods Fluids*, 12:507–533, 1991.

[*Lynch et al.* (1995)]D. R. Lynch, J. T. C. Ip, C. E. Naimie, and F. E. Werner. Convergence studies of tidally-rectified circulation on Georges Bank. In D. R. Lynch and A. M. Davies, editors, *Quantitative Skill Assessment for Coastal Ocean Models*, pages 153–174. Coastal and Estuarine Series **48**, American Geophysical Union, Washington,DC, 1995.

[*Mellor and Yamada* (1974)]G. L. Mellor and T. Yamada. A hierarchy of turbulence closure models for planetary boundary layers. *Journal of the Atmospheric Sciences*, 31:1791–1806, 1974.

[*Mellor and Yamada* (1982)]G. L. Mellor and T. Yamada. Development of a turbulence closure model for geophysical fluid problems. *Reviews of Geophysics and Space Physics*, 20(4):851–875, 1982.

[*Naimie* (1995)] C. E. Naimie. *On the modeling of the seasonal variation in the three-dimensional circulation near Georges Bank.* PhD thesis, Dartmouth College, 1995.

[*Naimie et al.* (1994)]C. E. Naimie, J. W. Loder, and D. R. Lynch. Seasonal variation of the three-dimensional residual circulation on Georges Bank. *J. Geophys. Res.*, 99(C8):15967–15989, 1994.

Appendix A

Finite Element Method Implementation

A.1 Hydrodynamic Equations

The governing equations are the linearized shallow-water equations with the conventional Boussinesq and hydrostatic assumptions. The turbulent Reynolds stress and flux terms in the conservation equations for momentum and heat are parameterized in terms of eddy diffusivities. Horizontal variations of all computed variables are assumed negligible and vertical advection is ignored. The continuity equation is not included, requiring that the barotropic pressure gradient be specified.

A.1.1 Equations of Motion

$$\frac{\partial u}{\partial t} - fv = -g\frac{\partial \eta}{\partial x} + \frac{\partial}{\partial z}\left(\kappa_m \frac{\partial u}{\partial z}\right) \tag{A.1}$$

$$\frac{\partial v}{\partial t} + fu = -g\frac{\partial \eta}{\partial y} + \frac{\partial}{\partial z}\left(\kappa_m \frac{\partial v}{\partial z}\right) \tag{A.2}$$

A.1.2 Conservation Equation for Energy

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(\kappa_h \frac{\partial T}{\partial z}\right) \tag{A.3}$$

A.1.3 Constitutive Relationship for Perturbation Density

$$\rho \equiv -\alpha(T - T_\circ) \tag{A.4}$$

The use of (A.4) enables (A.3) to be written as a conservation equation for the perturbation density:

$$\frac{\partial \rho}{\partial t} = \frac{\partial}{\partial z}\left(\kappa_h \frac{\partial \rho}{\partial z}\right). \tag{A.5}$$

A.1.4 Boundary Conditions

At the free surface, the applied wind stress (τ_w) and heat flux ($\dot{H}$) are sources of horizontal momentum and energy, respectively:

$$\kappa_m \frac{\partial \mathbf{v}}{\partial z} = \frac{\tau_w}{\rho_\circ}, \tag{A.6}$$

$$\kappa_h \frac{\partial T}{\partial z} = \dot{H} \Rightarrow \kappa_h \frac{\partial \rho}{\partial z} = -\alpha \dot{H}. \tag{A.7}$$

The bottom boundary layer is generally considered to be made up of three regions: the subviscous layer, the constant stress layer (logarithmic layer), and the bottom Ekman layer. The subviscous layer is the region in which the turbulent and viscous stresses are of the same order of magnitude. In the constant stress layer, turbulent mixing dominates and a mixing length strategy is used to describe the eddy viscosity while the shear stress (τ_b) is assumed to be constant. The friction velocity (u_*) provides a scale for the velocity fluctuations in the constant stress layer and is related to the shear stress via

$$u_* = \sqrt{\frac{\tau_b}{\rho_\circ}}. \tag{A.8}$$

Under these assumptions, a log mean velocity profile results:

$$u(\xi) = \frac{u_*}{\kappa} ln(\xi/\xi_\circ), \tag{A.9}$$

where ξ is the height above the bottom, $\xi_\circ$ is the roughness height (typically taken as 0.005-0.01 cm), and $\kappa = 0.4$ is the von Karman constant.

The finite element model NUBBLE includes the dynamics of the bottom boundary layer without resolving the constant stress layer through the use of a conventional quadratic slip condition that relates the velocity at a given position in the constant stress layer (the "bottom" of the modeled domain) to the shear stress:

$$\frac{\tau_b}{\rho_\circ} = u_*^2 = C_d |\mathbf{v}_b| \mathbf{v}_b. \tag{A.10}$$

Hence, the combination of (A.8-A.10) serves as a definition for the drag coefficient (C_d)

$$C_d = \frac{u_*^2}{\mathbf{v}_b^2} = [\frac{1}{\kappa} ln(\xi_b/\xi_\circ)]^{-2}. \tag{A.11}$$

Thus, the "bottom" of the modeled domain is actually displaced to a small distance above the sea floor (ξ_b) making the value of the vertical coordinate at the bottom $z_b = -h = -H + \xi_b$, where H is the depth of the sea floor relative to an undisturbed free surface. At this defined "bottom", the bottom stress is a source of horizontal momentum and the heat and salinity fluxes are assumed to be negligible:

$$\kappa_m \frac{\partial \mathbf{v}}{\partial z} = \frac{\tau_b}{\rho_\circ} = C_d |\mathbf{v}_b| \mathbf{v}_b, \tag{A.12}$$

$$\kappa_h \frac{\partial T}{\partial z} = 0 \Rightarrow \kappa_h \frac{\partial \rho}{\partial z} = 0. \tag{A.13}$$

A.2 Turbulence Closure

Two distinct turbulence closure schemes are employed, to provide means of model verification and intracomparison.

A.2.1 Quadratic Eddy Viscosity Closure

The simpler scheme assumes that the vertical momentum diffusivity is independent of vertical position, but allows for temporal variations due to quadratic dependence on the vertical average of the horizontal speed. This parameterization has been successfully applied to model barotropic circulation in coastal regions by a variety of authors [e.g. *Lynch and Naimie* (1993); *Davies and Jones* (1992)]:

$$\kappa_m = N_\circ |\bar{\mathbf{v}}|^2. \tag{A.14}$$

where $N_\circ$ is set to 0.2 as recommended by Davies and coworkers.

A.2.2 Mellor and Yamada Level 2.5 Advanced Turbulence Closure

The second, more complicated, closure involves the introduction of conservation equations for turbulent kinetic energy and the turbulent master length scale. Despite the computational overhead introduced by this parameterization, it has very favorable characteristics with respect to the temporal and spatial evolution of the mixing parameters. As discussed in *Mellor and Yamada* (1982), conservation equations are written for the turbulent kinetic energy and the turbulent master length scale:

$$\frac{\partial q^2}{\partial t} = \frac{\partial}{\partial z}\left(\kappa_q \frac{\partial q^2}{\partial z}\right) + 2\kappa_m \left[\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial z}\right)^2\right] + \frac{2g}{\rho_\circ}\kappa_h \frac{\partial \rho}{\partial z} - \frac{2q}{B_1 \ell} q^2, \tag{A.15}$$

$$\frac{\partial q^2 \ell}{\partial t} = \frac{\partial}{\partial z}\left(\kappa_q \frac{\partial q^2 \ell}{\partial z}\right) + \ell E_1 \kappa_m \left[\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial z}\right)^2\right] + \frac{\ell E_1 g}{\rho_\circ}\kappa_h \frac{\partial \rho}{\partial z} - \frac{q\tilde{w}}{B_1 \ell} q^2 \ell, \tag{A.16}$$

where the four terms on the right-hand-sides of (A.15) and (A.16) account for diffusion, shear production, buoyancy production, and dissipation of the respective conserved quantities. The turbulent diffusivities are defined by expressions containing the turbulent kinetic energy, the turbulent master length scale, and stability functions

$$\kappa_m \equiv \ell q S_m, \tag{A.17}$$

$$\kappa_h \equiv \ell q S_h, \tag{A.18}$$

$$\kappa_q \equiv \ell q S_q. \tag{A.19}$$

Details regarding the different versions of the Mellor and Yamada level 2.5 advanced turbulence closure scheme employed are discussed in *Mellor and Yamada* (1974), *Mellor and Yamada* (1982), *Blumberg and Mellor* (1987), *Galperin et al.* (1988), and *Blumberg et al.* (1992). The "version" of the closure employed in NUBBLE is essentially that published in *Blumberg and Mellor* (1987) with minor corrections from *Mellor and Yamada* (1982), the quasi-equilibrium adjustment from *Galperin et al.* (1988), and the wall proximity function from *Blumberg et al.* (1992). A departure from the closures employed by these previous investigators comes in the application of the $q^2\ell$ boundary condition at the bottom of the modeled domain. As in these studies, a bottom slip condition is employed with the bottom node located a small height off the sea floor (ξ_b), which is assumed to be in the logarithmic layer. At the bottom of the modeled domain, these earlier authors have used the Dirichlet boundary conditions:

$$q^2 = B_1^{2/3} u_*^2 \tag{A.20}$$

$$q^2\ell = 0 \tag{A.21}$$

where,

$$u_*^2 = \frac{\tau_b}{\rho_\circ} = C_d |\mathbf{v}_b|^2 \tag{A.22}$$

However, in the vicinity of the sea floor, the asymptotic behavior of the ℓ is constrained such that its value is given by the von Karman constant (κ) times the height off the actual sea floor (ξ). The numerical model NUBBLE accounts for the fact that $\ell = \kappa\xi_b$ at the bottom of the modeled domain. Thus, instead of the Dirichlet boundary condition specified in (A.21), NUBBLE applies the Dirichlet boundary condition:

$$q^2\ell = B_1^{2/3} u_*^2 \kappa \xi_b \tag{A.23}$$

$$\tilde{w} \equiv 1 + E_2\left(\frac{\ell}{\kappa(z - z_b + \xi_b)}\right)^2 + E_3\left(\frac{\ell}{\kappa(\eta - z)}\right)^2, \tag{A.24}$$

where $B_1 = 1.66$, $E_2 = 1.33$, and $E_3 = 0.25$.

Hence, the Dirichlet bottom boundary conditions employed for the turbulent conservation equations are written is such a manner that they depend exclusively on the velocity at the bottom node of the modeled domain, the height of the bottom node above the sea floor, the drag coefficient, B_1 and κ.

The surface of the modeled domain is assumed to be located at the undisturbed free surface, where the boundary conditions presented by *Mellor and Yamada* (1982) are used:

$$q^2 = B_1^{2/3} u_*^2, \tag{A.25}$$

$$q^2\ell = 0, \tag{A.26}$$

where the friction velocity is related to the wind stress via

$$u_*^2 = \frac{\tau_w}{\rho_\circ}. \tag{A.27}$$

It is worthy of mention that these boundary conditions are not entirely satisfying. (A.25) seems suspect when there is no wind stress and the depth is sufficiently shallow that the turbulence generated by the bottom stress penetrates the surface. Indeed, other authors [e.g. *Davies and Jones* (1990)] have employed a zero flux condition instead of (A.25), under these circumstances. In the presence of wind, setting the master turbulent length scale to zero via (A.26) seems suspect. However, numerical experiments with versions of NUBBLE which specified no flux conditions instead of (A.25-A.26) have supported the observation by *Mellor and Yamada* (1982) that "Solutions are quite insensitive to free-stream values of $q^2\ell$." and demonstrated that the no flux surface boundary condition for q^2 yields results very similar to those obtained when (A.25) is employed.

A.3 Finite Element Formulation

A.3.1 Vertical Discretization

Figure A.1 demonstrates the two available vertical discretization options available in NUBBLE, both of which are set using the relationship:

$$z(\epsilon) = -h + \epsilon h - Asin(2\pi\epsilon), \tag{A.28}$$

where ϵ increases linearly with node number, from 0 at the bottom to 1 at the free surface. The uniform grid option deploys uniformly spaced nodes ($A = 0$ in 1.1). The sinusoidal grid option departs from the uniform spacing via the inclusion of a trigonometric function that reduces the nodal spacing near the extremes of the water column. For this option, the constant A is set to obtain the desired nodal spacing at the extremes of the water column.

A.3.2 Conservation Equations for heat, turbulent kinetic energy, and turbulent master length scale

Under the assumptions previously discussed, the conservation equations for heat, turbulent kinetic energy, and mixing length can be rearranged such that the left-hand-side contains terms containing the quantity being conserved. Consider $\Psi(z,t)$ to be a general quantity for which the conservation equation is to be applied:

$$\frac{\partial \Psi}{\partial t} = +\frac{\partial}{\partial z}\left(\kappa \frac{\partial \Psi}{\partial z}\right) + S + \alpha\Psi, \tag{A.29}$$

where:

κ is the diffusivity,
S is the sum of domain sources which are not quasi-linear in Ψ,
$\alpha\Psi$ is the sum of domain sources which are quasi-linear in Ψ.

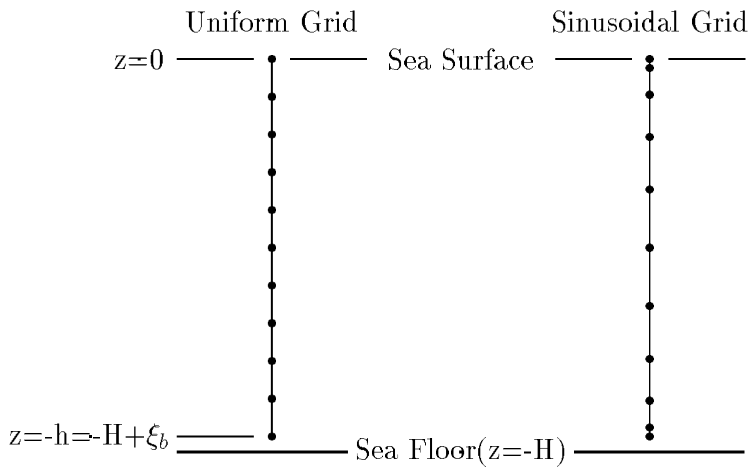

Figure A.1: Standard vertical discretization options for NUBBLE.

The weak form of (A.29) is obtained by taking the inner product with a weighting function $\phi_i(z)$ and rearranging to collect all terms containing Ψ on the left hand side:

$$\langle \frac{\partial \Psi}{\partial t}\phi_i \rangle - \langle \alpha \Psi \phi_i \rangle - \langle \frac{\partial}{\partial z}\left(\kappa \frac{\partial \Psi}{\partial z}\right)\phi_i \rangle = \langle S\phi_i \rangle, \tag{A.30}$$

where $\langle\rangle$ is the domain integral over z. Integrating the third term of (A.30) by parts yields

$$\langle \frac{\partial \Psi}{\partial t}\phi_i \rangle - \langle \alpha \Psi \phi_i \rangle + \langle \kappa \frac{\partial \Psi}{\partial z}\frac{d\phi_i}{dz} \rangle = \langle S\phi_i \rangle + \kappa \frac{\partial \Psi}{\partial z}\phi_i|_{bottom}^{top}. \tag{A.31}$$

Selecting chapeau functions as the weighting functions and using the galerkin method to also use chapeau functions as a basis for Ψ and S yields

$$\Psi(z,t) = \sum_{j=1}^{N} \Psi_j(t)\phi_j(z), \tag{A.32}$$

$$S(z,t) = \sum_{j=1}^{N} S_j(t)\phi_j(z). \tag{A.33}$$

Substituting into (A.31) yields

$$\sum_{j=1}^{N}\left[\frac{d\Psi_j}{dt}\langle \phi_j\phi_i \rangle - \alpha\Psi_j\langle \phi_j\phi_i \rangle + \Psi_j\langle \kappa \frac{d\phi_j}{dz}\frac{d\phi_i}{dz} \rangle\right] = \sum_{j=1}^{N} S_j\langle \phi_j\phi_i \rangle + F, \tag{A.34}$$

where:

$$F = \kappa \frac{\partial \Psi}{\partial z}|_{top}\phi_N - \kappa \frac{\partial \Psi}{\partial z}|_{bottom}\phi_1. \tag{A.35}$$

The approximation for the time derivative is

$$\frac{d\Psi_j}{dt} = \frac{\Psi_j^{k+1} - \Psi_j^k}{\Delta t}. \tag{A.36}$$

Substituting these quantities into (A.34)

$$\sum_{j=1}^{N}\Big[(\frac{1}{\Delta t}-\alpha)\langle\phi_j\phi_i\rangle+\langle\kappa\frac{d\phi_j}{dz}\frac{d\phi_i}{dz}\rangle\Big]\Psi_j^{k+1}=\sum_{j=1}^{N}(\frac{1}{\Delta t}\Psi_j^k+S_j^k)\langle\phi_j\phi_i\rangle+F^{k+\frac{1}{2}}. \qquad (A.37)$$

Nodal quadrature on the vertical linear elements renders the mass matrix $\langle\phi_j\phi_i\rangle$ diagonal and the stiffness matrix $\langle\kappa\frac{d\phi_j}{dz}\frac{d\phi_i}{dz}\rangle$ tridiagonal. The diffusivity κ is treated as a constant on each element.

A.3.3 Conservation Equations for Horizontal Momentum

The conservation equations for momentum are slightly more difficult to handle due to the coupling through the Coriolis term. The conservation equations for u and v can be expressed as:

$$\frac{\partial u}{\partial t}-fv-\frac{\partial}{\partial z}\left(\kappa_m\frac{\partial u}{\partial z}\right)=S_x, \qquad (A.38)$$

$$\frac{\partial v}{\partial t}+fu-\frac{\partial}{\partial z}\left(\kappa_m\frac{\partial v}{\partial z}\right)=S_y. \qquad (A.39)$$

As discussed in *Lynch and Werner* (1991), the complication of the Coriolis term may be removed by the introduction of the complex surrogate for $\mathbf{v}$, $\vartheta=u+\mathbf{j}v$, where $\mathbf{j}=\sqrt{-1}$. This leads to a form very similar to (A.34) with the important exception that the resulting matrix equation is complex. As in *Lynch and Werner* (1991), all terms in the momentum equation are centered at time level $k+\frac{1}{2}$. With the definition of the time-centered average

$$\vartheta_j^*=\frac{\vartheta_j^{k+1}+\vartheta_j^k}{2}, \qquad (A.40)$$

the approximation for the time derivative becomes

$$\frac{d\vartheta_j}{dt}=\frac{\vartheta_j^{k+1}-\vartheta_j^k}{\Delta t}=\frac{2(\vartheta_j^*-\vartheta_j^k)}{\Delta t}, \qquad (A.41)$$

and the final matrix equation is

$$\sum_{j=1}^{N}\Big[(\frac{2}{\Delta t}+\mathbf{j}f)\langle\phi_j\phi_i\rangle+\langle\kappa\frac{d\phi_j}{dz}\frac{d\phi_i}{dz}\rangle\Big]\vartheta_j^*=\sum_{j=1}^{N}(\frac{2}{\Delta t}\vartheta_j^k+S_j^*)\langle\phi_j\phi_i\rangle+\kappa_m\frac{\partial\vartheta}{\partial z}|_{top}\phi_N-\kappa_m\frac{\partial\vartheta}{\partial z}|_{bottom}\phi_1, \qquad (A.42)$$

where

$$\kappa_m\frac{\partial\vartheta}{\partial z}|_{top}=\frac{(\tau_{wx}+\mathbf{j}\tau_{wy})^*}{\rho_o}, \qquad (A.43)$$

$$\kappa_m\frac{\partial\vartheta}{\partial z}|_{bottom}=C_d|\mathbf{v}_b^k|\vartheta_b^*. \qquad (A.44)$$

Nodal quadrature on the vertical linear elements renders the mass matrix $\langle\phi_j\phi_i\rangle$ diagonal and the stiffness matrix $\langle\kappa\frac{d\phi_j}{dz}\frac{d\phi_i}{dz}\rangle$ tridiagonal. The diffusivity κ is treated as a constant on each element.

A.3.4 Solution Strategy

Figure (A.2) displays a flowchart for NUBBLE:

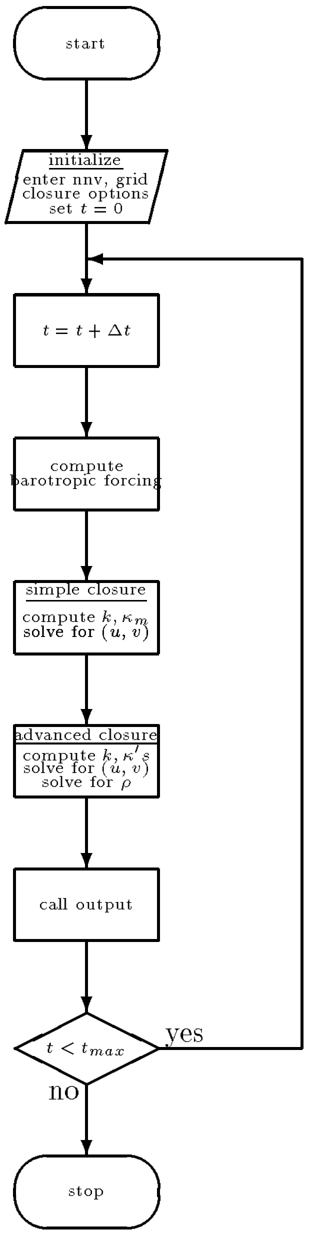

Figure A.2: Flowchart for NUBBLE.

Appendix B

Sourcecode for NUBBLE Release 1.1

Order of Sourcecode Listing:

- Include File
- User Subroutines
- Main Program
- Fixed Subroutines

```
C***********************************************************************
C***********************************************************************
C  Include File for NUBBLE Release 1.1
C-----------------------------------------------------------------------
C This variable declaration file is to be appended at the beginning of
C  some of the source programs in the NUBBLE series for implicit
C  variable types and known integer parameters declaration.
C
C Variables:
C   NNVDIM - Maximum number of vertical nodes
C   NEVDIM - Maximum number of vertical elements
C   RHOREF - Reference Density (kg/m^3)
C   G      - Gravitational acceleration (m/s^2)
C
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   NUBBLE Release 1.1 - July 31, 1996
C-----------------------------------------------------------------------
      PARAMETER(NNVDIM=101,NEVDIM=NNVDIM-1,RHOREF=1024.0,G=9.806)
C***********************************************************************
C***********************************************************************
```

```
C***********************************************************************
C***********************************************************************
C Beginning of User-Specified Subroutines for NUBBLE Release 1.1
C Beginning of User-Specified Subroutines for NUBBLE Release 1.1
C***********************************************************************
C***********************************************************************
      SUBROUTINE INTITALIZEN1(DT,ITMAX,BAT,Zlog,f,CD,ENZDFMIN,
     &NNV,Z,UT,VT,U,V,RHO,Q2,Q2L)
C Include file for parameter statements
      INCLUDE 'NUBBLE.DIM'
C Dimension global arrays
      REAL Z(NNVDIM),UT(NNVDIM),VT(NNVDIM)
      REAL U(NNVDIM),V(NNVDIM),RHO(NNVDIM),Q2(NNVDIM),Q2L(NNVDIM)
C-----------------------------------------------------------------------
C This user subroutine is provided for initialization purposes.  It is
C  called at the beginning of the numerical simulation to:
C   set time stepping information,
C   set physical parameters,
C   set the vertical grid,
C   initialize dependent variables.
C
C Time stepping information:
C   DT   - size of time step [s]
C   ITMAX- maximum number of time steps for the simulation
C
C Physical parameters:
C   BAT  - depth below the surface to the position where the bottom slip
C            condition is applied [m]
C   Zlog - height above the sea-floor at which the bottom slip condition
C            is applied [m]
C   CD   - quadratic bottom slip coefficient (0.005 is suggested)
C   f    - coriolis parameter [1/s]
C   ENZDFMIN - minimum vertical eddy viscosity for DF80 turbulence
C                closure [m^2/s]
C
C Vertical discretization information:
C   NNV  - number of vertical nodes (note:  J indexes from 1 to NNV)
C   Z(J) - vertical descritization array (Z(1)=-BAT,.,Z(NNV)=0.0) [m]
C
C Dependent variables initialized for the DF80 turbulence closure:
C   UT(J),VT(J) - horizontal velocities [m/s]
C
C Dependent variables initialized for the level 2.5 turbulence closure:
C   U(J),V(J)   - horizontal velocities [m/s]
C   RHO(J)      - (perturbation density-reference density)/reference
C                   density [unitless]
C   Q2(J)       - twice the turbulent kinetic energy [m^2/s^2]
C   Q2L(J)      - twice the turbulent kinetic energy times the master
C                   length scale [m^3/s^2]
C
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   NUBBLE Release 1.1 - July 31, 1996
C-----------------------------------------------------------------------
C Two standard subroutines are available for the vertical grid
C  assignment.
C
C  1. Subroutine UNIZ assigns uniform vertical spacing.
C     Assignment of NNV and BAT are required before this subroutine is
C     called.  The form of the call statement is:
C
C      CALL UNIZ(NNVDIM,NNV,BAT,Z)
C
C  2. Subroutine SINEZ assigns sinusiodal vertical spacing, with the
C     discretization at the extremes of the water column set to
C     DZBL [m].  It requires assignment of NNV, DZBL, and BAT before it
C     is called.  The form of the call statement is:
```

```
C
C      CALL SINEZ(NNVDIM,NNV,DZBL,BAT,Z)
C===SUBROUTINE INITIALIZEN1: Beginning-of-user-specified-instructions===
      RETURN
      END
C**********************************************************************
C**********************************************************************
      SUBROUTINE SURFACEN1(ITER,TIME,RHOSURF,RHOFLUX,GX,GY,WX,WY)
C Include file for parameter statements
      INCLUDE 'NUBBLE.DIM'
C----------------------------------------------------------------------
C This subroutine is provided for specification of nonzero surface
C  forcing.  It is called every time step:
C
C Provided by calling routine:
C   ITER - current iteration number (ranges from 1 to ITMAX)
C   TIME - time at the middle of the current time step (i.e. the
C           time when this subroutine is called) [s]
C         note that TIME=0.0 at the beginning of the simulation
C   RHOSURF - the surface value of RHO at the beginning of the current
C              time step [unitless]
C
C Surface nonzero conditions set within this subroutine during each time
C  step (note:  just prior to the call to this subroutine, all of the
C  following variables are set to zero in the main program):
C   RHOFLUX - the surface flux of rho (kh*drho/dz) [kg/(s m^2)]
C   (GX,GY) - (x,y) components of the surface pressure gradient [m/s^2]
C              (note (GX,GY) are positive on the RHS of the
C              momentum equation => (GX,GY)=(-g*dzeta/dx,-g*dzeta/dy))
C   (WX,WY) - (x,y) components of the surface wind stress divided by
C              the reference density [m^2/s^2]
C
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   NUBBLE Release 1.1 - July 31, 1996
C=====SUBROUTINE SURFACEN1: Beginning-of-user-specified-instructions====
      RETURN
      END
C**********************************************************************
C**********************************************************************
      SUBROUTINE OUTPUTN1(DT,ITMAX,BAT,Zlog,f,CD,ENZDFMIN,
     &ITER,TIME,RHOFLUX,GX,GY,WX,WY,
     &NNV,Z,UT,VT,ENZ,U,V,RHO,Q2,Q2L,ENZM,ENZH,ENZQ)
C Include file for parameter statements
      INCLUDE 'NUBBLE.DIM'
C Dimension global arrays
      REAL Z(NNVDIM), UT(NNVDIM),VT(NNVDIM),ENZ(NNVDIM)
      REAL U(NNVDIM),V(NNVDIM),RHO(NNVDIM),Q2(NNVDIM),Q2L(NNVDIM),
     &ENZM(NNVDIM),ENZH(NNVDIM),ENZQ(NNVDIM)
C----------------------------------------------------------------------
C This subroutine is used to write any combination of parameters and
C  3-D quantities; all of which are provided by the calling routine:
C
C Time stepping parameters:
C   DT  - size of time step (s)
C   ITMAX - maximum number of time steps for the current simulation
C
C Physical parameters:
C   BAT  - depth below the surface to the position where the bottom slip
C           condition is applied [m]
C   Zlog - height above the sea-floor at which the bottom slip condition
C           is applied [m]
C   CD   - quadratic bottom slip coefficient (0.005 is suggested)
C   f    - coriolis parameter [1/s]
C   ENZDFMIN - minimum vertical eddy viscosity for DF80 turbulence
C               closure [m^2/s]
C
```

```
C Time stepping parameters which indicate the current time:
C   ITER - current time step number (ranges from 1 to ITMAX)
C   TIME - time at the end of the current time step (i.e. the time
C            when this subroutine is called) [s]
C          note: TIME=0.0 at the beginning of the simulation
C
C Surface forcing for the current time step:
C   RHOFLUX - the surface flux of rho (kh*drho/dz) [kg/(s m^2)]
C   (GX,GY) - (x,y) components of the surface pressure gradient [m/s^2]
C               (note (GX,GY) are positive on the RHS of the
C               momentum equation => (GX,GY)=(-g*dzeta/dx,-g*dzeta/dy))
C   (WX,WY) - (x,y) components of the surface wind stress divided by
C               the reference density [m^2/s^2]
C
C Vertical discretization information:
C   NNV  - number of vertical nodes (note:  J indexes from 1 to NNV)
C   Z(J) - vertical descritization array (Z(1)=-BAT,.,Z(NNV)=0.0) [m]
C
C Current values of dependent variables for the DF80 turbulence closure:
C   UT(J),VT(J) - horizontal velocities [m/s]
C   ENZ(J)      - vertical eddy viscosity [m^2/s]
C
C Current values of dependent variables for the level 2.5 turbulence
C  closure:
C   U(J),V(J)   - horizontal velocities [m/s]
C   RHO(J)      - (perturbation density-reference density)/reference
C                   density [unitless]
C   Q2(J)       - twice the turbulent kinetic energy [m^2/s^2]
C   Q2L(J)      - twice the turbulent kinetic energy times the master
C                   length scale [m^3/s^2]
C   ENZM(J)     - vertical eddy viscosity [m^2/s]
C   ENZH(J)     - vertical heat diffusivity [m^2/s]
C   ENZQ(J)     - vertical diffusivity for turbulent quantities [m^2/s]
C
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   NUBBLE Release 1.1 - July 31, 1996
C=====SUBROUTINE OUTPUTN1: Beginning-of-user-specified-instructions=====
      RETURN
      END
C***********************************************************************
C***********************************************************************
C End of User-Specified Subroutines for NUBBLE Release 1.1
C End of User-Specified Subroutines for NUBBLE Release 1.1
C***********************************************************************
C***********************************************************************
```

```
C**********************************************************************
C**********************************************************************
C Beginning of Main Program for NUBBLE Release 1.1
C Beginning of Main Program for NUBBLE Release 1.1
C**********************************************************************
C**********************************************************************
      PROGRAM NUBBLE
C----------------------------------------------------------------------
C This code solves vertical transport equations for the shallow water
C   momentum equations, the heat equation, and the level
C   2.5 turbulence quantities (Q2 and Q2L).
C
C It incorporates the following revisions to the level 2.5 closure
C   scheme as presented in Mellor and Yamada (1974,1982):
C   - Galperin et. al. (1988) quasi-equilibrium adjustment
C   - Blumberg et. al. (1992) wall-proximity function adjustment
C   => MYQEBGO
C
C In its current form it computes time-domain values of U,V two ways
C   - quadratic closure from Davies and Furnes (1980)
C   - level 2.5 turbulence closure
C
C Time stepping parameters:
C   DT  - size of time step (s)
C   ITMAX - maximum number of time steps for the current simulation
C
C Physical parameters:
C   BAT  - depth below the surface to the position where the bottom slip
C            condition is applied [m]
C   Zlog - height above the sea-floor at which the bottom slip condition
C            is applied [m]
C   CD   - quadratic bottom slip coefficient (0.005 is suggested)
C   f    - coriolis parameter [1/s]
C
C Time stepping parameters which indicate the current time:
C   ITER - current time step number (ranges from 1 to ITMAX)
C   TIME - time at the end of the current time step (i.e. the time
C            when this subroutine is called) [s]
C          note: TIME=0 at the beginning of the simulation
C
C Surface forcing for the current time step:
C   RHOFLUX - the surface flux of rho (kh*drho/dz) [kg/(s m^2)]
C   (GX,GY) - (x,y) components of the surface pressure gradient [m/s^2]
C               (note (GX,GY) are positive on the RHS of the
C               momentum equation => (GX,GY)=(-g*dzeta/dx,-g*dzeta/dy))
C   (WX,WY) - (x,y) components of the surface wind stress divided by
C               the reference density [m^2/s^2]
C
C Vertical discretization information:
C   NNV  - number of vertical nodes (note:  J indexes from 1 to NNV)
C   Z(J) - vertical descritization array (Z(1)=-BAT,.,Z(NNV)=0.0) [m]
C
C Current values of dependent variables for the DF80 turbulence closure:
C   UT(J),VT(J) - horizontal velocities [m/s]
C   ENZ(J)      - vertical eddy viscosity [m^2/s]
C
C Current values of dependent variables for the level 2.5 turbulence
C   closure:
C   U(J),V(J)   - horizontal velocities [m/s]
C   RHO(J)      - (perturbation density-reference density)/reference
C                   density [unitless]
C   Q2(J)       - twice the turbulent kinetic energy [m^2/s^2]
C   Q2L(J)      - twice the turbulent kinetic energy times the master
C                   length scale [m^3/s^2]
C   ENZM(J)     - vertical eddy viscosity [m^2/s]
C   ENZH(J)     - vertical heat diffusivity [m^2/s]
C   ENZQ(J)     - vertical diffusivity for turbulent quantities [m^2/s]
C
```

```
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   TURB1D - March 16, 1993
C   NUBBLEb - March 17, 1994
C   NUBBLE Release 1.1 - July 31, 1996
C-----------------------------------------------------------------------
      INCLUDE 'NUBBLE.DIM'
C
C Dimension arrays
      REAL Z(NNVDIM),SX(NNVDIM),SY(NNVDIM)
      REAL UT(NNVDIM),VT(NNVDIM),ENZ(NNVDIM)
      REAL U(NNVDIM),V(NNVDIM),Q2(NNVDIM),Q2L(NNVDIM),RHO(NNVDIM),
     &ENZM(NNVDIM),ENZH(NNVDIM),ENZQ(NNVDIM)
C
C NBCS is a flag which toggles the boundary condition type for ALL
C turbulence quantities:
C  NBCS=1 => Dirichlet BCs
C  NBCS=2 => No Flux BCs
C For NUBBLE Release 1.1, NBCS is set to 1
      NBCS=1
C
C Initialization phase
      CALL INTITALIZEN1(DT,ITMAX,BAT,Zlog,f,CD,ENZDFMIN,
     &NNV,Z,UT,VT,U,V,RHO,Q2,Q2L)
C
C Check size of NNVDIM to ensure it is sufficient for specified NNV
      IF(NNV.GT.NNVDIM)THEN
         WRITE(*,*)'Increase NNVDIM in NUBBLE.DIM and recompile'
         STOP
      ENDIF
C
C Time-stepping loop
      ITER=0
 30   ITER=ITER+1
      TIME=DT*ITER
C
C 1.Set surface forcing:
C    zero surface forcing
C    set nonzero surface forcing
C    assemble nodal forcing for RHS of momentum equation
      RHOFLUX=0.0
      GX=0.0
      GY=0.0
      WX=0.0
      WY=0.0
      CALL SURFACEN1(ITER,TIME-0.5*DT,RHO(NNV),RHOFLUX,GX,GY,WX,WY)
      DO i=1,NNV
         SX(i)=GX
         SY(i)=GY
      ENDDO
C
C 2.Use DF80 friction to compute UT,VT at the end of this time step
C    compute ak and ENZ using DF80
C    compute UT,VT
      akDF=CD*SQRT(UT(1)**2.0+VT(1)**2.0)
      CALL ENZDF80(NNVDIM,NNV,Z,UT,VT,ENZDFMIN,ENZ)
      CALL MOMSOL(NNV,DT,f,akDF,Z,ENZ,SX,SY,WX,WY,UT,VT)
C
C 3.Use MYQEBGO friction to compute U,V at the end of this time step
C    compute akMY (linearized bottom slip coefficient)
C    compute diffusivities using MYQEBGO
C    compute U,V
C    compute RHO
      akMY=CD*SQRT(U(1)**2.0+V(1)**2.0)
      CALL ENZMYQEBGO(NNV,DT,CD,Z,RHO,U,V,WX,WY,NBCS,Zlog,
     &Q2,Q2L,ENZM,ENZH,ENZQ)
      CALL MOMSOL(NNV,DT,f,akMY,Z,ENZM,SX,SY,WX,WY,U,V)
      CALL ENERGYSOL(NNV,DT,Z,ENZH,RHOFLUX,RHO)
C
C 4.Call SUBROUTINE OUTPUTN1 with results at the end of this time step
      CALL OUTPUTN1(DT,ITMAX,BAT,Zlog,f,CD,ENZDFMIN,
     &ITER,TIME,RHOFLUX,GX,GY,WX,WY,
     &NNV,Z,UT,VT,ENZ,U,V,RHO,Q2,Q2L,ENZM,ENZH,ENZQ)
C
C 5.Continue time-stepping if ITER.LT.ITMAX
      IF(ITER.LT.ITMAX)go to 30
C
C End of Routine
      STOP
      END
C***********************************************************************
C***********************************************************************
C End of Main Program for NUBBLE Release 1.1
C End of Main Program for NUBBLE Release 1.1
C***********************************************************************
C***********************************************************************
C***********************************************************************
C***********************************************************************
C Beginning of Fixed Subroutines for NUBBLE Release 1.1
C Beginning of Fixed Subroutines for NUBBLE Release 1.1
C***********************************************************************
C***********************************************************************
      SUBROUTINE ENERGYSOL(NNV,DT,Z,ENZ,RHOFLUX,RHO)
C-----------------------------------------------------------------------
C 2-level in time, mass lumped solution to energy diffusion equation
C Current implementation solves equation for RHO with specified flux at
C   the surface and no flux at the bottom.
C It solves the transport equation for
C   RHO=pertubation pressure/reference density.  This is valid for
C   current use since the coefficient of thermal
C   expansion (alpha) is constant}
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   revision - 28 October 93
C   latest revision - 17 March 1994
C-----------------------------------------------------------------------
      INCLUDE 'NUBBLE.DIM'
C
C Dimension global arrays
      real Z(NNVDIM),ENZ(NNVDIM),RHO(NNVDIM)
C
C Dimension local arrays
      real AA(NNVDIM),BB(NNVDIM),CC(NNVDIM),RHS(NNVDIM)
C
C Set factor for level of implicity
      fa=1.0
C
C Initialize arrays
      do i=1,NNV
         AA(i)=0.0
         BB(i)=0.0
         CC(i)=0.0
         RHS(i)=0.0
      enddo
C
C Build banded matrix
C  [LHS]=1/DT*[M]+fa*ENZ*[K]
C  {RHS}=1/DT*[M]{RHO}-(1-fa)*ENZ*[K]
      do i=1,NNV-1
       he=ABS(z(i+1)-z(i))
       BB(i)=  BB(i)  +1./DT*he/2.+fa*ENZ(i)*1./he
       CC(i)=  CC(i)              -fa*ENZ(i)*1./he
       AA(i+1)=AA(i+1)            -fa*ENZ(i)*1./he
       BB(i+1)=BB(i+1)+1./DT*he/2.+fa*ENZ(i)*1./he
```

```
        RHS(i)=  RHS(i)  +1./DT*RHO(i)*he/2.
     &           -(1.-fa)*ENZ(i)*(RHO(i)-RHO(i+1))*1./he
        RHS(i+1)=RHS(i+1)+1./DT*RHO(i+1)*he/2.
     &           -(1.-fa)*ENZ(i)*(RHO(i+1)-RHO(i))*1./he
      enddo
C
C  Apply BCS to {RHS}: no flux at bottom and RHOFLUX at surface
      RHS(NNV)=RHS(NNV)+RHOFLUX
C
C  Solve
      CALL THOMAS(3,AA,BB,CC,RHS,NNV)
C
C {RHO} at the new time is returned as {RHS}
      do i=1,NNV
         RHO(i)=RHS(i)
      enddo
      return
      end
C**********************************************************************
C**********************************************************************
      SUBROUTINE MOMSOL(NNV,DT,f,ak,Z,ENZ,SX,SY,WX,WY,U,V)
C----------------------------------------------------------------------
C 2-level centered in time, mass lumped momentum diffusion equ
C Computes solution using complex formulation of Lynch and Werner(1991)
C Includes barotropic and wind forcings
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   revision - 28 October 93
C   latest revision - 17 March 1994
C----------------------------------------------------------------------
      INCLUDE 'NUBBLE.DIM'
C
C Dimension global arrays
      real z(NNVDIM),U(NNVDIM),V(NNVDIM),enz(NNVDIM),
     &SX(NNVDIM),SY(NNVDIM)
C
C Dimension local arrays
      complex AA(NNVDIM),BB(NNVDIM),CC(NNVDIM)
      complex RHS(NNVDIM),UV(NNVDIM),SXY(NNVDIM)
C
C Local assignments
      complex eye,twind
C
C Set forcing constants
      eye=CMPLX(0.0,1.0)
      twind=CMPLX(WX,WY)
C
C Compute {UV} and initialize other arrays
      do i=1,NNV
         UV(i)=CMPLX(U(i),V(i))
         SXY(i)=CMPLX(SX(i),SY(i))
         AA(i)=0.0
         BB(i)=0.0
         CC(i)=0.0
         RHS(i)=0.0
      enddo
C
C Build [LHS]:  [LHS]=(2/DT+eye*f)*[M]+enz*[K]
      do i=1,NNV-1
         he=ABS(z(i+1)-z(i))
         BB(i)=  BB(i)  +(2./DT+eye*f)*(he/2.)+enz(i)*(1./he)
         CC(i)=  CC(i)                        -enz(i)*(1./he)
         AA(i+1)=AA(i+1)                      -enz(i)*(1./he)
         BB(i+1)=BB(i+1)+(2./DT+eye*f)*(he/2.)+enz(i)*(1./he)
      enddo
C
C  Apply BCS to [LHS]
```

```
C   Bottom
C    Linear bottom slip
      BB(1)=BB(1)+ak
C   Top
C    Type 2 applied to {RHS} later
C
C Build {RHS}
C
C  {RHS}=2/DT*[M]{UV}+SOURCES
      do i=1,NNV-1
         he=ABS(z(i+1)-z(i))
         RHS(i)=  RHS(i)  +2./DT*UV(i)*he/2.  +SXY(i)*he/2.
         RHS(i+1)=RHS(i+1)+2./DT*UV(i+1)*he/2.+SXY(i+1)*he/2.
      enddo
C
C  Apply BCS to {RHS}
      RHS(NNV)=RHS(NNV)+twind
C
C Solve
      CALL CTHOMAS(3,AA,BB,CC,RHS,NNV)
C
C {UV} at the centered time is returned as {RHS}
      do i=1,NNV
         U(i)=2.0*REAL(RHS(i))-U(i)
         V(i)=2.0*AIMAG(RHS(i))-V(i)
      enddo
      return
      end
C***********************************************************************
C***********************************************************************
      SUBROUTINE ENZMYQEBGO(NNV,DT,CD,Z,RHO,U,V,WX,WY,nclo,Zlog,
     &Q2,Q2L,ENZM,ENZH,ENZQ)
C-----------------------------------------------------------------------
C This subroutine coordinates the calculation of turbulent diffusivities
C   and the solution of the turbulent diffusion equations
C Minimum values for Q2, L, ENZM, ENZH, and ENZQ are set internally to
C   0.000001 [MKS Units]
C History:
C   Written by Christopher E. Naimie
C   Dartmouth College
C   Revision - October 28, 1993
C   Revision - March 17, 1994
C   NUBBLE Release 1.1 - July 31, 1996
C-----------------------------------------------------------------------
      INCLUDE 'NUBBLE.DIM'
C
C Dimension global arrays
      real  Z(NNVDIM),U(NNVDIM),V(NNVDIM),RHO(NNVDIM)
      real Q2(NNVDIM),Q2L(NNVDIM),
     &ENZM(NNVDIM),ENZH(NNVDIM),ENZQ(NNVDIM)
C
C Dimension local arrays
      real RHSQ2(NNVDIM),RHSQ2L(NNVDIM),DKQ2(NNVDIM),DKQ2L(NNVDIM)
C
C Data assignments
      data q2min/.000001/
C
C Set constants
      vk=0.4
      NEV=NNV-1
C
C Compute diffusivities and turbulent forcings
      CALL GALPERINBGO(U,V,Q2,Q2L,RHO,Z,NNV,NEV,Zlog,
     &               RHSQ2,RHSQ2L,DKQ2,DKQ2L,ENZM,ENZH,ENZQ)
C
C Solve turbulent diffusion equations
      US=16.6**(2./3.)*SQRT(WX*WX+WY*WY)
      UB=16.6**(2./3.)*CD*(U(1)*U(1)+V(1)*V(1))
      CALL Q2SOL(NNV,DT,Z,ENZQ,RHSQ2,DKQ2,Q2,nclo,UB,US)
      do i=1,NNV
         Q2(i)=AMAX1(Q2(i),q2min)
      enddo
      US=0.0
      UB=UB*vk*Zlog
      CALL Q2SOL(NNV,DT,Z,ENZQ,RHSQ2L,DKQ2L,Q2L,nclo,UB,US)
      do j=1,NNV
         n1=j-1
         n2=j+1
         if(j.EQ.1)n1=1
         if(j.EQ.NNV)n2=NNV
         bvfreq=-G*(RHO(n2)-RHO(n1))/(Z(n2)-Z(n1))
         if(bvfreq.GT.0)Q2L(j)=
     &                 AMIN1(Q2L(j),Q2(j)*SQRT(0.28*Q2(j)/bvfreq))
         enddo
      return
      end
C************************************************************************
C************************************************************************
C
      subroutine GALPERINBGO(u,v,q2,q2l,rho,z,NNV,nev,Zlog,
     *                rhsq2,rhsq2l,dkq2,dkq2l,ekm,ekh,ekq)
c
c.....subroutine to return right hand side and turbulent eddy
c.....viscosity terms based on the Mellor-Yamada 2.5 closure scheme
c.....(Dan and Cisco, 14 April 1992)
c
c.....modified 23 april by DRL
c          to enforce locally-defined minima for ekm, ekq, ekh
c.....modified 24 april by DRL
c          cd removed from this subroutine; rely on main pgm for that.
C Modified 04 March 93 by Christopher E. Naimie
C   transformed to 1-D arrays, divided dkq2l by ell,
C   added ellmin
C
C
c
c.....References:
c
c      Mellor and Yamada (1982) Development of a turb. closure
c          model for geophys. fluid probs.  In: Rev. Geophys. and
c          Space Phys., vol. 20, 851-875.
c      Blumberg and Mellor (1987) A description of a 3D coastal ocean
c          ocean circulation model.  In: Three-Dimensional Coastal Ocean
c          Models; AGU Geophysical Monograph Series, Coastal and Estuarine
c          Sciences 4, 1-16.
c
c.....q2=kinetic energy
c.....q2l=q-squared times mixing length
c.....enm,enh,enq = momentum,heat and energy viscosity terms
c.....dkq2,dkq2l = dissipation/decay terms for q2 and q2l
c.....rhsq2,rhsq2l = right hand side terms for q2 and q2l equations
c
c.....the following arrays are dimensioned NNV in the main program:
c.....                q2,q2l,dkq2,dkq2l,rhsq2,rhsq2l
c
c.....the following arrays are dimensioned nev (NNV-1) in the main program:
c.....                enm,enh,enq
c
      INCLUDE 'NUBBLE.DIM'
      dimension u(*),v(*),z(*),rho(*)
      dimension q2(*),q2l(*),ekm(*),ekh(*),ekq(*)
      dimension rhsq2(*),rhsq2l(*),dkq2(*),dkq2l(*)
c
c.....local arrays... not returned to main program
c
      dimension ell(NNVDIM),wpf(NNVDIM),gm(nevdim),gh(nevdim),
     * sm(nevdim),sh(nevdim)
      dimension rhsq2e(nevdim),rhsq2le(nevdim)
c
c.....minimum viscosity, diffusivities (DRL, 23 april 1992)
c
      data ekmmin,ekhmin,ekqmin/ .000001, .000001, .000001/
      data q2min,ellmin/ .000001, .000001/
c
c....."universal" constants associated with closure
c
      data a1,a2,b1,b2,c1,e1,e2,e3,sq/0.92,0.74,16.6,10.1,0.08,
     *                              1.8,1.33,0.25,0.2/
c
c.....von Karman and bottom drag coefficient
c
      data vk/0.4/
c
c check if NNVDIM is ok
c
      if(NNV.gt.NNVDIM) then
      write(*,*) 'NNV is too big in SUBROUTINE GALPERINBGO',NNV,NNVDIM
      write(*,*) 'execution terminating'
      stop
      endif
c
c.....check if nev eq NNV-1
c
      if((NNV-1).ne.nev)then
         write(*,*)'nev ',nev,' not equal to NNV-1 ',NNV-1,' in SUBROUT
     &INE GALPERINBGO'
         write(*,*)'execution terminating
         stop
      endif
c
c.....compute mixing length -- ell -- at each node
c.....and wall-proximity-function: wpf
c
*      ell(1)=amax1(q2l(1)/q2(1),ellmin)
*      wpf(1)=1.+e2
      do i=1,NNV-1
         ell(i)=amax1(q2l(i)/q2(i),ellmin)
         temp2=ell(i)/vk/(abs(-z(1)+Zlog)+z(i))
         temp3=ell(i)/vk/(z(NNV)-z(i))
         wpf(i)=1.+e2*temp2*temp2+e3*temp3*temp3
      end do
      ell(NNV)=amax1(q2l(NNV)/q2(NNV),ellmin)
      wpf(NNV)=1.+e3
c
c.....element-based variables
c
      do i=1,nev
c
c.....production terms due to:
c                    shear (gm) and buoyancy (gh)
c
         ell2tq2=ell(i+1)*ell(i+1)/q2(i+1)
         ell2bq2=ell(i)*ell(i)/q2(i)
         factor=0.5*(ell2tq2+ell2bq2)
         dz=z(i+1)-z(i)
         dudz=(u(i+1)-u(i))/dz
         dvdz=(v(i+1)-v(i))/dz
         shear=dudz*dudz+dvdz*dvdz
         buoy=g*(rho(i+1)-rho(i))/dz
         gm(i)=factor*shear
C (in Blumberg and Mellor)
C         gm(i)=factor*SQRT(shear)
         gh(i)=factor*buoy
c
```

```
c.....next compute sm and sh
c
c.....NOTE: Mellor and Yamada's paper defines a12 slightly
c.....      differently than Blumberg and Mellor's.  Here
c.....      we use that in Mellor and Yamada.
c
c         a11=6.*a1*a2*gm(i)
c         a12=1.-3.*a2*b2*gh(i)-12.*a1*a2*gh(i)
C (in Blumberg and Mellor)
C         a12=1.-2.*a2*b2*gh(i)-12.*a1*a2*gh(i)
c         a21=1.+6.*a1*a1*gm(i)-9.*a1*a2*gh(i)
c         a22=-(12.*a1*a1*gh(i)+9.*a1*a2*gh(i) )
c         rhs1=a2
c         rhs2=a1*(1.-3.*c1)
c         det=a11*a22-a21*a12
c         sm(i)=(rhs1*a22-rhs2*a12)/det
c         sh(i)=(a11*rhs2-a21*rhs1)/det
C
C Here we use galperin
C
        gh(i)=AMAX1(gh(i),-0.28)
        gh(i)=AMIN1(gh(i),0.0233)
        d1=1.-9.*a1*a2*gh(i)
        d2=1.-3.*a2*gh(i)*(6.*a1+b2)
        d1=d1*d2
        sm(i)=a1*(1-3.*c1-6.*a1/b1-3.*a2*gh(i)*((b2-3.*a2)*(1.-6.*a1/b1)
     &-3.*c1*(b2+6.*a1)))/d1
        sh(i)=a2*(1-6.*a1/b1)/d2
c
c.....now for the mixing coefficients ekm,ekh,ekq...
c
         ellq=0.5*(ell(i+1)*sqrt(q2(i+1))+
     *             ell(i)*sqrt(q2(i)))
c         ekm(i)=ellq*sm(i)
c         ekh(i)=ellq*sh(i)
c         ekq(i)=ellq*sq
c
c.....DRL change 23 april 1992 for minimum values
c
         ekm(i)=amax1(ellq*sm(i),ekmmin)
         ekh(i)=amax1(ellq*sh(i),ekhmin)
         ekq(i)=amax1(ellq*sq   ,ekqmin)
c
c.....finally the right-hand side terms
c
         rhsq2e(i)=2.*ekm(i)*shear
         rhsq2e(i)=rhsq2e(i)+2.*ekh(i)*buoy
         elle1=0.5*(ell(i+1)+ell(i))*e1
         rhsq2le(i)=elle1*ekm(i)*shear
         rhsq2le(i)=rhsq2le(i)+elle1*ekh(i)*buoy
      end do
c
c.....compute the right-hand side terms on nodes (after all)
c
      do i=2,NNV-1
         dzt=z(i+1)-z(i)
         dzb=z(i)-z(i-1)
         dzj=dzt+dzb
         rhsq2(i)=(rhsq2e(i)*dzt+rhsq2e(i-1)*dzb)/dzj
         rhsq2l(i)=(rhsq2le(i)*dzt+rhsq2le(i-1)*dzb)/dzj
      end do
c
c.....at the top and bottom just keep the "active" neighboring
c.....element's contribution
c
      rhsq2(NNV)=rhsq2e(nev)
      rhsq2(1)=rhsq2e(1)
      rhsq2l(NNV)=rhsq2le(nev)
      rhsq2l(1)=rhsq2le(1)
c
c.....compute nodal-based dissipation/decay terms
c
      do i=1,NNV
         qoverb1=sqrt(q2(i))/b1
         dkq2(i)=2.*qoverb1/ell(i)
         dkq2l(i)=qoverb1*wpf(i)/ell(i)
      end do
C
      return
      end
C*********************************************************************
C*********************************************************************
      SUBROUTINE Q2SOL(NNV,DT,Z,ENZQ,RHSQ2,DKQ2,Q2,nflag,UB,US)
C---------------------------------------------------------------------
C 2-level in time, mass lumped solution to Q2 or Q2L diffusion equ
C RHSQ2 - RHS forcings
C DKQ2 -  LHS expression for dissipation
C ENZQ -  eddy diffusivity
C nflag=1 => apply type 1 BC's UB and US
C nflag=2 => apply type 2 BC's; flux=0
C History
C   Written by Christopher E. Naimie
C   Dartmouth College
C   revision - 15 March 93
C   latest revision - 17 March 1994
C---------------------------------------------------------------------
      INCLUDE 'NUBBLE.DIM'
C
C Dimension global arrays
      real Z(NNVDIM),ENZQ(NNVDIM),RHSQ2(NNVDIM),DKQ2(NNVDIM),Q2(NNVDIM)
C
C Dimension local arrays
      real AA(NNVDIM),BB(NNVDIM),CC(NNVDIM),RHS(NNVDIM)
C
C Set factor for level of implicity
      fa=1.0
C
C Initialize arrays
      do i=1,NNV
         AA(i)=0.0
         BB(i)=0.0
         CC(i)=0.0
         RHS(i)=0.0
      enddo
C
C Build banded matrix
C  [LHS]=(2/DT+DKQ2)*[M]+ENZQ*[K]
C  {RHS}=2/DT*[M]{Q2}+SOURCES
      do i=1,NNV-1
       he=ABS(z(i+1)-z(i))
       BB(i)=  BB(i)  +(1./DT+fa*DKQ2(i))*he/2.  +fa*ENZQ(i)*1./he
       CC(i)=  CC(i)                             -fa*ENZQ(i)*1./he
       AA(i+1)=AA(i+1)                           -fa*ENZQ(i)*1./he
       BB(i+1)=BB(i+1)+(1./DT+fa*DKQ2(i+1))*he/2.+fa*ENZQ(i)*1./he
       RHS(i)=  RHS(i)  +1./DT*Q2(i)*he/2.  +RHSQ2(i)*he/2.
     &           -(1.-fa)*DKQ2(i)*Q2(i)*he/2.
     &           -(1.-fa)*ENZQ(i)*(Q2(i)-Q2(i+1))*1./he
       RHS(i+1)=RHS(i+1)+1./DT*Q2(i+1)*he/2.+RHSQ2(i+1)*he/2.
     &           -(1.-fa)*DKQ2(i+1)*Q2(i+1)*he/2.
     &           -(1.-fa)*ENZQ(i)*(Q2(i+1)-Q2(i))*1./he
      enddo
C
C Apply BC's
      if(nflag.EQ.1)then
         AA(1)=0.0
         BB(1)=1.0
         CC(1)=0.0
         RHS(1)=UB
         AA(NNV)=0.0
         BB(NNV)=1.0
         CC(NNV)=0.0
         RHS(NNV)=US
      endif
C
C Solve
      CALL THOMAS(3,AA,BB,CC,RHS,NNV)
C
C {Q2} at new time is returned as {RHS}
      do i=1,NNV
         Q2(i)=RHS(i)
      enddo
      return
      end
C*********************************************************************
C*********************************************************************
      SUBROUTINE ENZDF80(NNVDIM,NNV,Z,U,V,ENZDFMIN,ENZ)
C---------------------------------------------------------------------
C This subroutine computes the DF80 momentum diffusivity and the linear
C   bottom slip coefficient used in FUNDY4
C History
C   Written by Christopher E. Naimie
C   Dartmouth College
C   latest revision - 15 March 93
C---------------------------------------------------------------------
C
C Dimension global arrays
      real Z(NNVDIM),U(NNVDIM),V(NNVDIM),ENZ(NNVDIM)
C
C Compute
      usum=0.0
      vsum=0.0
      do i=1,NNV-1
         usum=usum+0.5*(U(i+1)+U(i))*(Z(i+1)-Z(i))
         vsum=vsum+0.5*(V(i+1)+V(i))*(Z(i+1)-Z(i))
      enddo
      vbar=SQRT((usum/Z(1))**2.0+(vsum/Z(1))**2.0)
      f4n=0.2*vbar*vbar
      f4n=AMAX1(f4n,ENZDFMIN)
      do j=1,NNV
         ENZ(j)=f4n
      enddo
      return
      end
C*********************************************************************
C*********************************************************************
      SUBROUTINE SINEZ(NNVDIM,NNV,DZBL,BAT,Z)
C
C Sinusoidal mesh generating part of FUNDY5 Subroutine SINEGRID5
C
C Global Arrays
      REAL Z(NNVDIM)
C
C Set sinusoidal grid
      NEV=NNV-1
      PI=ACOS(-1.)
      A=(BAT-DZBL*NEV)/(NEV*SIN(2.*PI/NEV))
      IF(A.LT.0.)A=0.0
      Z(1)=-BAT
      DO 10 J=2,NNV-1
         EPS=(J-1.0)/(NNV-1.0)
         Z(J)=-BAT*(1.-EPS) -A*SIN(2.*PI*EPS)
  10  CONTINUE
      Z(NNV)=0.0
      RETURN
      END
C***********************************************************************
C***********************************************************************
      SUBROUTINE UNIZ(NNVDIM,NNV,BAT,Z)
C
C Uniform mesh generating part of FUNDY5 Subroutine UNIGRID5
C
C Global Arrays
      REAL Z(NNVDIM)
C
      DZ=BAT/(NNV-1.)
      Z(1)=-BAT
      DO 10 J=2,NNV-1
         Z(J)=Z(J-1)+DZ
 10   CONTINUE
      Z(NNV)=0.0
      RETURN
      END
C***********************************************************************
C***********************************************************************
C
      SUBROUTINE CTHOMAS(KKK,A,B,C,R,N)
C
C THOMAS ALGORITHM FOR TRIDIAGONAL MATRICES  -  SYSTEM FORTRAN-77
C COMPLEX VERSION
C A, B, AND C = LHS COLUMNS, DIMENSIONED (=>N) IN CALLING PROGRAM
C R = RIGHT-HAND-SIDE, DIMENSIONED (=>N) IN CALLING PROGRAM
C N = # OF EQUATIONS
C SOLUTION RETURNS IN R WHEN KKK => 2
C KKK = 1 PERFORMS LU DECOMPOSITION DESTRUCTIVELY
C KKK = 2 PERFORMS BACK SUBSTITUTION
C KKK = 3 PERFORMS OPTIONS 1 AND 2
C
      COMPLEX A(N),B(N),C(N),R(N)
C
C  STEP 1 - TRIANGULARIZE MATRIX A USING DOOLITTLE METHOD
C
      IF(KKK.EQ.2)GO TO 45
      C(1)=C(1)/B(1)
      DO 40 I=2,N
      IM=I-1
      C(I)=C(I)/(B(I)-A(I)*C(IM))
   40 CONTINUE
      IF(KKK.EQ.1)RETURN
C
C STEP 2 - MODIFY LOAD VECTOR R
C
   45 CONTINUE
      R(1)=R(1)/B(1)
      DO 41 I=2,N
      IM=I-1
      R(I)=(R(I)-A(I)*R(IM))/(B(I)-A(I)*C(IM))
   41 CONTINUE

C
C BACK SUBSTITUTE SOLUTION INTO R VECTOR
C
      DO 50 I=1,N-1
      J=N-I
      JP=J+1
      R(J)=R(J)-C(J)*R(JP)
   50 CONTINUE
      RETURN
      END
C
C***********************************************************************
C***********************************************************************
C
```

```
      SUBROUTINE THOMAS(KKK,A,B,C,R,N)
C
C THOMAS ALGORITHM FOR TRIDIAGONAL MATRICES  -  SYSTEM FORTRAN-77
C REAL VERSION
C A, B, AND C = LHS COLUMNS, DIMENSIONED (=>N) IN CALLING PROGRAM
C R = RIGHT-HAND-SIDE, DIMENSIONED (=>N) IN CALLING PROGRAM
C N = # OF EQUATIONS
C SOLUTION RETURNS IN R WHEN KKK => 2
C KKK = 1 PERFORMS LU DECOMPOSITION DESTRUCTIVELY
C KKK = 2 PERFORMS BACK SUBSTITUTION
C KKK = 3 PERFORMS OPTIONS 1 AND 2
C
      DIMENSION A(N),B(N),C(N),R(N)
C
C  STEP 1 - TRIANGULARIZE MATRIX A USING DOOLITTLE METHOD
C
      IF(KKK.EQ.2)GO TO 45
      C(1)=C(1)/B(1)
      DO 40 I=2,N
      IM=I-1
      C(I)=C(I)/(B(I)-A(I)*C(IM))
   40 continue
      IF(KKK.EQ.1)RETURN
C
C STEP 2 - MODIFY LOAD VECTOR R
C
   45 continue
      R(1)=R(1)/B(1)
      DO 41 I=2,N
      IM=I-1
      R(I)=(R(I)-A(I)*R(IM))/(B(I)-A(I)*C(IM))
   41 continue
C
C BACK SUBSTITUTE SOLUTION INTO R VECTOR
C
      DO 50 I=1,N-1
      J=N-I
      JP=J+1
      R(J)=R(J)-C(J)*R(JP)
   50 continue
      RETURN
      END
C**********************************************************************
C**********************************************************************
C End of Fixed Subroutines for NUBBLE Release 1.1
C End of Fixed Subroutines for NUBBLE Release 1.1
C************************************************************************
C************************************************************************
```