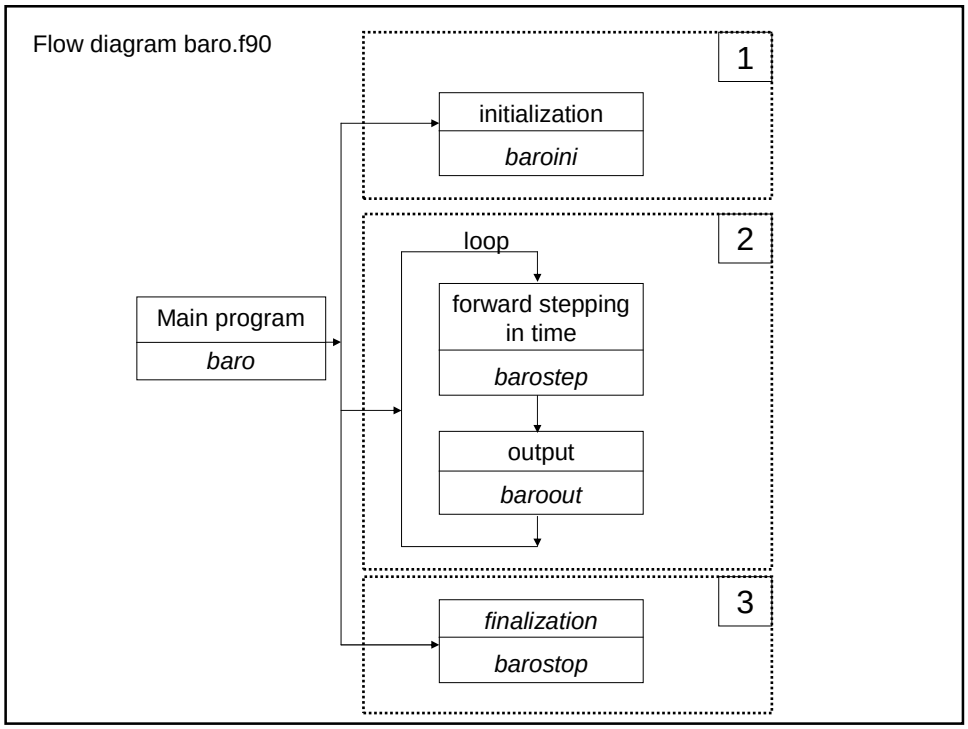
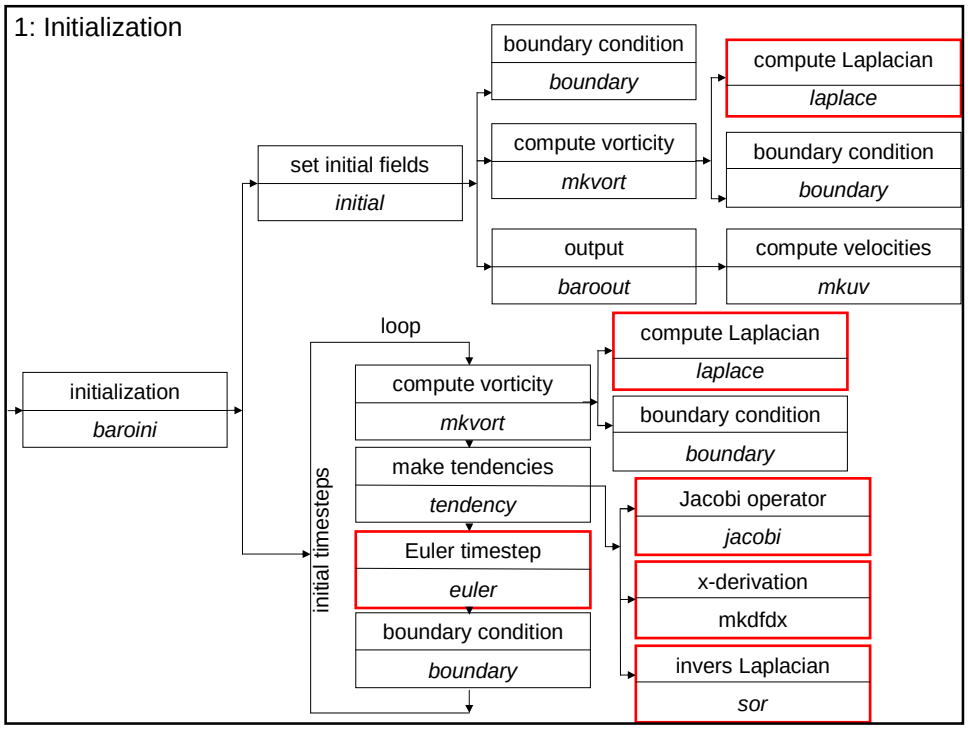
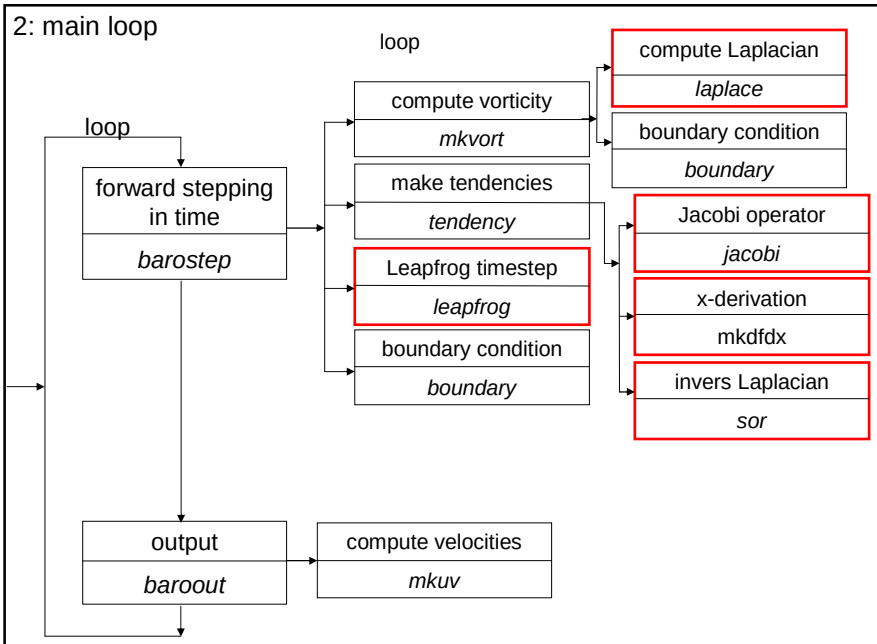
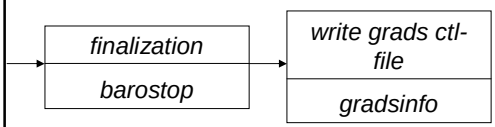

Flow diagram baro.f90
1
initialization
baroini
2
loop
Main program
baro
forward stepping in time
barostep
output
baroout
3
finalization
barostop


1: Initialization
boundary condition
boundary
compute Laplacian
laplace
set initial fields
initial
compute vorticity
mkvort
boundary condition
boundary
output
baroout
compute velocities
mkuv
loop
compute Laplacian
laplace
initialization
baroini
compute vorticity
mkvort
boundary condition
boundary
make tendencies
tendency
Jacobi operator
jacobi
initial timesteps
Euler timestep
euler
x-derivation
mkdfdx
boundary condition
boundary
invers Laplacian
sor

2: main loop



3: finalization



dummy subroutines which need to be completed

example: subroutine ***sor***

```
      subroutine sor(pdf,pf,pdx,pdy,kx,ky)
      implicit none
!
!     subroutine sor computes the inverse Laplacian from a given field
!     by using the Successive OverRelaxation method (SOR)
!
      integer :: kx               ! x dimension
      integer :: ky               ! y dimension
      real :: pdx                 ! x grid point distance
      real :: pdy                 ! y grid point distance
      real :: pdf(0:kx+1,0:ky+1)  ! input: field
      real :: pf(0:kx+1,0:ky+1)   ! output: inverse Laplacian of input
!
      return
      end
```

module baromod (part 1)

```
      module baromod
      implicit none
!
!     module baromod contains all global parameters/variables
!     (i.e. which might be needed in  more that one subroutine)
!     to be included be *use baromod*
!
!     a) constants:
!
      integer,parameter :: NX = 64         ! x dimension
      integer,parameter :: NY = 32         ! y dimension
      real,parameter :: radea = 6.371E6    ! radius of the earth [m]
      real,parameter :: omega = 0.00007292 ! angular velocity [1/s]
!
      integer :: nrun     = 1        !  no timesteps to be computed
      integer :: nout     = 1        !  output interval
!
      real :: rlat    = 50.          ! central latitude of the channel
      real :: xchannel= 360.         ! channel (x-) length [dec]
      real :: ychannel= 40.          ! channel (y-) width  [dec]
      real :: delt    = 3600.        ! time step [s]
```

module baromod (part 2)

```
!
!     b) variables
!
      integer :: nstep    = 0        ! time step counter
      integer :: nwout    = 0        ! output counter
!
      real :: s(0:NX+1,0:NY+1)   = 0. ! streamfct. [m**2/2]
      real :: sm(0:NX+1,0:NY+1)  = 0. ! streamfct. (old timestep)[m**2/s]
      real :: dsdt(0:NX+1,0:NY+1)= 0. ! streamfct tendnecy (ds/dt)
[m**2/s**2]
      real :: vo(0:NX+1,0:NY+1)  = 0. ! vorticity [1/s]
!
      real :: pi                  ! pi
      real :: beta                ! beta parameter
      real :: dx                  ! gridpoint distance in x [m]
      real :: dy                  ! gridpopint distance in y [m]
      real :: rossby              ! Rossby number
      real :: f0                  ! corriolis par. at cent.latitude[1./s]
      real :: delt2               ! 2*delt [s]
!
      end module baromod
```