Programming the numerical solution of the “shallow water equations model”

The shallow water equations simulate the propagation of disturbances in incompressible fluids resulting from gravitational and/or rotational accelerations.

They are a simplification of the Navier-Stokes equations under the assumptions:

- the fluid has uniform density and is in hydrostatic balance;
- the fluid flows over a weakly sloping bottom;
- the surface elevation slope is small;
- the horizontal scale of the flow is large compared to the fluid depth;
- the flow velocity is independent of depth and bottom friction is negligible.

The shallow water approximation provides, for instance, a reasonable model for the propagation of tsunamis in the ocean, since tsunamis are very long waves (with hundreds of km wavelength) propagating over depths of a few kilometers.

Solving the equations means to calculate the prognostic output fields (water elevation and velocity) at specified time and space coordinates.

The shallow water equations

The set of linearized partial differential equations, enclosing the principles of conservation of mass and momentum, is composed of a x-momentum equation (1), a y-momentum equation (2) and a continuity equation (3), as follows:

$$\frac{\partial u}{\partial t} - fv + g\frac{\partial h}{\partial x} = 0 \qquad (1)$$

$$\frac{\partial v}{\partial t} + fu + g\frac{\partial h}{\partial y} = 0 \qquad (2)$$

$$\frac{\partial h}{\partial t} + d\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \qquad (3)$$

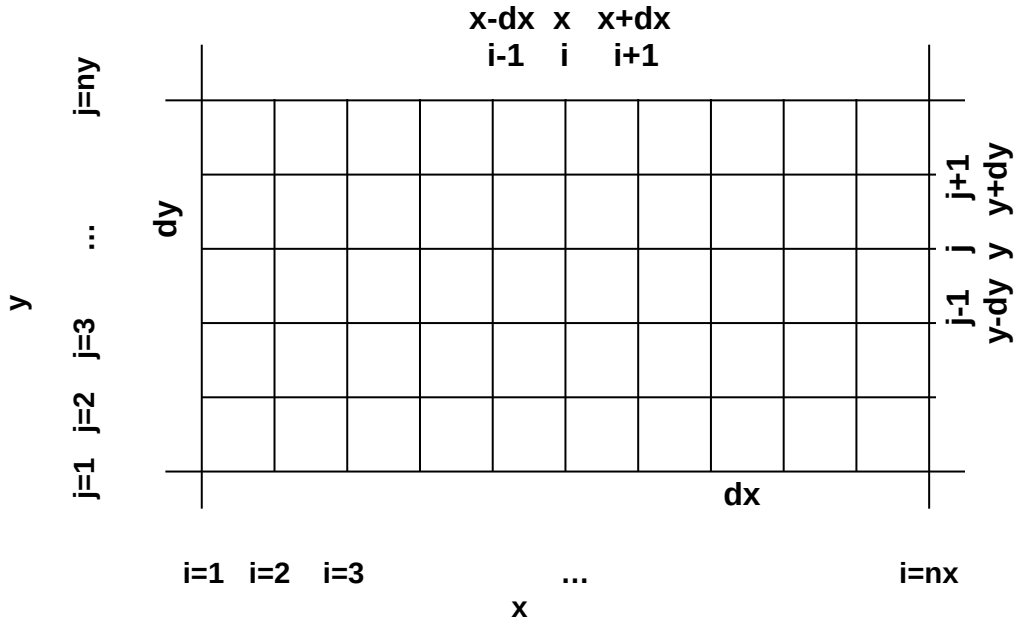
where u = u(x,y,t), v = v(x,y,t) and h = h(x,y,t) are the dependent variables we want to solve for, corresponding to the two horizontal components of the (depth averaged) water velocity and to the perturbed water elevation. The independent variables x and y are the horizontal coordinates and t is time. Variable d is the total water depth at rest (i.e., when h=0), f is the planetary vorticity (also known as the Coriolis parameter) and g is gravity. The former variable accounts for the effects of earth rotation and the latter to the gravitational effects.

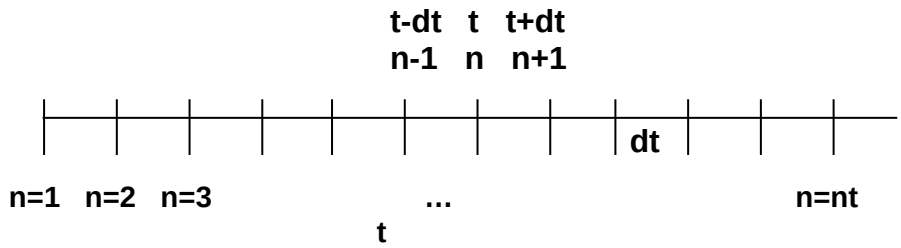
Discretization of the equations

To solve the equations, the explicit finite-differences method is employed. In order to apply finite-differences, a discretization in space and time is needed:

$$u^n_{i,j} = u (i\,dx, j\,dy, n\,dt) \qquad (4)$$
$$v^n_{i,j} = v (i\,dx, j\,dy, n\,dt)$$
$$h^n_{i,j} = h (i\,dx, j\,dy, n\,dt)$$

with the spatial discretization intervals dx and dy, and the temporal discretization interval dt. The spatial and temporal positions are then given by x = i dx (i = 1, 2, ... nx), y = j dy (j = 1, 2, ... ny) and t = n dt (n = 1, 2, ... nt).





Approximation of derivatives using finite differences

Having the space and time coordinates discretized, the temporal first derivatives are approximated by forward finite-differences:

$$\frac{\partial u}{\partial t} \approx \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{dt} \qquad (5)$$

$$\frac{\partial v}{\partial t} \approx \frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{dt}$$

$$\frac{\partial h}{\partial t} \approx \frac{h_{i,j}^{n+1} - h_{i,j}^{n}}{dt}$$

and the spatial first derivatives are approximated by centered finite-differences:

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1,j}^{n} - u_{i-1,j}^{n}}{2dx} \qquad (6)$$

$$\frac{\partial v}{\partial y} \approx \frac{v_{i,j+1}^{n} - v_{i,j-1}^{n}}{2dy}$$

$$\frac{\partial h}{\partial x} \approx \frac{h_{i+1,j}^{n} - h_{i-1,j}^{n}}{2dx}$$

$$\frac{\partial h}{\partial y} \approx \frac{h_{i,j+1}^{n} - h_{i,j-1}^{n}}{2dy}$$

Finite-differences version of the shallow water equations

Inserting expressions 5-6 into Equations 1-3 and solving for $u_{i,j}^{n+1}$, $v_{i,j}^{n+1}$ and $h_{i,j}^{n+1}$, the discretized shallow water equations become:

$$u_{i,j}^{n+1} = u_{i,j}^{n} + f_{i,j}\, dt\, v_{i,j}^{n} - g\, \frac{dt}{2dx} \left(h_{i+1,j}^{n} - h_{i-1,j}^{n}\right) \qquad (7)$$

$$v_{i,j}^{n+1} = v_{i,j}^{n} - f_{i,j}\, dt\, \mathbf{u_{i,j}^{n+1}} - g\, \frac{dt}{2dy} \left(h_{i,j+1}^{n} - h_{i,j-1}^{n}\right) \qquad (8)$$

$$h_{i,j}^{n+1} = h_{i,j}^{n} - d_{i,j}\, \frac{dt}{2dx} \left(\mathbf{u_{i+1,j}^{n+1}} - \mathbf{u_{i-1,j}^{n+1}}\right) - d_{i,j}\, \frac{dt}{2dy} \left(\mathbf{v_{i,j+1}^{n+1}} - \mathbf{v_{i,j-1}^{n+1}}\right) \qquad (9)$$

There are two different time levels (n+1 and n). Knowing $u^{n}_{i,j}$, $v^{n}_{i,j}$ and $h^{n}_{i,j}$ at any location (i,j), one can immediately calculate $u^{n+1}_{i,j}$ (step 1). Using that updated $u^{n+1}_{i,j}$ (see modification in bold in Eq. 8) and the previously known $v^{n}_{i,j}$ and $h^{n}_{i,j}$ one can compute $v^{n+1}_{i,j}$ (step 2). Finally, using the updated $u^{n+1}_{i,j}$ and $v^{n+1}_{i,j}$ (see modification in bold in Eq. 9) and the previously known $h^{n}_{i,j}$ one can compute $h^{n+1}_{i,j}$ (step 3).

In this way, the three equations are coupled and the solution can recursively be computed, time step by time step, until some predefined time $t_{max} = nt\, dt$ is arrived. At the

beginning of the recursion, the initial conditions $u^1_{i,j,}$ $v^1_{i,j}$ and $h^1_{i,j}$ need to be known. In order to continue the iteration procedure, once the prognostic fields are known and stored in files, the time steps are swapped (i.e., $u^{n+1}_{i,j}$ will become $u^n_{i,j}$, $v^{n+1}_{i,j}$ will become $v^n_{i,j}$ and $h^{n+1}_{i,j}$ will become $h^n_{i,j}$).

Fortran computer program

It is important to choose the names for the variables exactly as suggested below, otherwise it is very difficult to help debugging the Fortran program in case it is necessary. In addition, the values of the variables dx, dy, dt, must be followed otherwise a numerically unstable algorithm can be produced.

List of variables and their meaning

nx = 500	- (integer) total number of grid points in x-direction
ny = 500	- (integer) total number of grid points in y-direction
nt = 3600	- (integer) total number of time steps
dx = 1000.0	- (real) spatial discretization in meters in x-direction
dy = 1000.0	- (real) spatial discretization in meters in y-direction
dt = 1.0	- (real) time step size in seconds
dump=30.	- (real) output periodicity in seconds
g=9.81	- (real) gravity in meters per squared second
f = 0.8e-4	- (real 2D array) Coriolis parameter in one over squared seconds
d=1000.	- (real 2D array) total water depth at rest in meters
u	- (real 2D array) velocity x-direction in meters per second
ub	- (real 2D array) velocity x-direction of previous time step
v	- (real 2D array) velocity y-direction in meters per second
vb	- (real 2D array) velocity y-direction of previous time step
h	- (real 2D array) perturbed elevation in meters
hb	- (real 2D array) perturbed elevation of previous time step
i, j	- (integer) indices for x- and y-direction, respectively
n	- (integer) time step index

Structure of the main program (swm.f90)

The program will have the following structure:

program start
    declaration of constants and variables
    initialization of all variables
    open output files for u,v,h

loop over time steps
solve equation 7
solve equation 8
solve equation 9
calculate doubly-periodic boundary conditions
swap time steps
calculate and write to screen at specified times the maximum value of h
write unformatted at specified times the entire u,v,h to three individual files
write the entire u,v,h and d to one NetCDF file
end of loop over time steps
close output files
program end

The main program should call six subroutines, which will conduct the specified tasks, namely initialization, solution of eq. 7, solution of eq. 8, solution of eq. 9, boundary conditions and output in NetCDF (the latter subroutine will be given).

Initialization subroutine (init.f90)

The initial conditions for u,v and h are here specified as follows:

```
u(i,j)=(-1/(d(i,j)*(kx**2+ky**2)))
*(kx*sqrt(f(i,j)**2+g*d(i,j)*(kx**2+ky**2))
*cos(kx*i*dx+ky*j*dy)+f(i,j)*ky*sin(kx*i*dx+ky*j*dy))
*exp(-((0.2*kx*(i-350)*dx)**2+(0.2*ky*(j-350)*dy)**2))

v(i,j)=(1/(d(i,j)*(kx**2+ky**2)))
*(-ky*sqrt(f(i,j)**2+g*d(i,j)*(kx**2+ky**2))
*cos(kx*i*dx+ky*j*dy)+f(i,j)*kx*sin(kx*i*dx+ky*j*dy))
*exp(-((0.2*kx*(i-350)*dx)**2+(0.2*ky*(j-350)*dy)**2))

h(i,j)=cos(kx*i*dx+ky*j*dy)
*exp(-((0.2*kx*(i-350)*dx)**2+(0.2*ky*(j-350)*dy)**2))
```

with

```
kx=10*(2*pi)/(nx*dx)
ky=10*(2*pi)/(ny*dy)
```

These equations represent a geostrophically-balanced wave train modulated by a localized (at i=350, j=350) Gaussian function.

Equations subroutines (loop_u.f90, loop_v.f90 and loop_h.f90)

The equations 7-9 are coded in their discretized form. Be careful with the input/output arguments of these subroutines in order to insure the coupling between them.

Boundary conditions subroutine (bound.f90)

Cyclic (also known as periodic) boundary conditions should be implemented in the x- and y-directions. They consist in repeating the values of i=2 at i=nx and the values of i=1 at i=nx-1 (and the values of j=2 at j=ny and the values of j=1 at j=ny-1) for all three prognostic variables.

NetCDF output subroutine (outcdf.f90)

Subroutine will be supplied.

Compiling and linking the program

Compile your source files individually and then link them as follows:

```
gfortran -c swm.f90
gfortran -c init.f90
gfortran -c loop_u.f90
gfortran -c loop_v.f90
gfortran -c loop_h.f90
gfortran -c bound.f90
gfortran -c -I/sw/jessie-x64/netcdf_fortran-4.5.1-static-gccsys/include outcdf.f90
gfortran -o swm.x swm.o init.o loop_u.o loop_v.o loop_h.o bound.o outcdf.o -L/sw/jessie-x64/netcdf_fortran-4.5.1-static-gccsys/lib -lnetcdff -L/sw/jessie-x64/szip-2.1.1-static-gccsys/lib -L/sw/jessie-x64/hdf5-1.10.5-static-gccsys/lib -L/sw/jessie-x64/netcdf-4.7.1-static-gccsys/lib -lnetcdf -lnetcdf -lm -lhdf5_hl -lhdf5 -lsz -lz -lm -ldl
```

Running the program

To run the program type in the command line: ./swm.x

Experiments

Two integrations of the model should be conducted:

- Case 1: wave train over a flat bottom domain with H=1000m.
- Case 2: wave train over a domain with two regions of constant H. In the northeast H=1000m and in the southwest H=500m.