HELM2D User Manual

Yogi A. Erlangga*

May 19, 2006

1 Introduction

The software HELM2D solves sparse, large linear systems arising from finite difference discretizations of the 2D Helmholtz equation

$$-\frac{\partial^2\phi}{\partial x^2} - \frac{\partial^2\phi}{\partial y^2} - k^2(x,y)\phi = g, \tag{1}$$

in a rectangular domain $\Omega = [0, L_x] \times [0, L_y]$. ϕ usually represents the pressure (acoustic) wave, and $k = k(x, y)$ is the reduced frequency, which can be written as

$$k(x,y) = \frac{\omega}{c(x,y)} = \frac{2\pi f}{c(x,y)}, \tag{2}$$

with f and ω respectively the frequency and the angular frequency, and c the local speed of sound. f is defined as the frequency which is generated by the source g. In the case that $L_x = L_y = 1$, k is then called the wavenumber. The source function g is usually given in the form of Dirac δ function, where

$$\delta(x,y) = \begin{cases} 1, & (x,y) = (x_s, y_s), \\ 0, & \text{elsewhere}, \end{cases} \tag{3}$$

where (x_s, y_s) the position of the source in a coordinate system.

To solve the Helmholtz, non-reflecting conditions are applied at the boundaries $\Gamma = \partial\Omega$. The boundary conditions can be formulated in many ways. Some of them are used in HELM2D, included those Engquist & Majda [], absorption boundary layer/sponge layer widely used in geophysics and perfectly match layer (PML).

*TU Delft, Delft Institute of Applied Mathematics (DIAM), Mekelweg 4, 2628 CD Delft, The Netherlands,
Currently at TU Berlin, Institut für Mathematik, Strasse des 17. Juni 136, D-10623 Berlin, Germany
Email: erlangga@math.tu-berlin.de

Finite difference discretization of (1) on a rectangular, equidistant grid leads to a linear system

$$Ax = b, \qquad A \in \mathbb{C}^{n\times n}, \qquad x, b \in \mathbb{C}^n. \tag{4}$$

A is complex-valued due to the inclusion of the boundary conditions. To solve (4), an iterative method based on Krylov subspace methods is used.

To accelerate the convergence a preconditioner M is used, which is obtained from discretizations of the following operator

$$-\frac{\partial^2\phi}{\partial x^2} - \frac{\partial^2\phi}{\partial y^2} - (\alpha_1 - \alpha_2\hat{i})k^2(x,y)\phi = g, \tag{5}$$

with $\alpha_1, \alpha_2 \geq 0$ a parameter tuned in order to get fast convergence, and $\hat{i} = \sqrt{-1}$ the complex identity. By preconditioning we solve the equivalent linear system

$$AM^{-1}z = b, \qquad z = Mx, \qquad M \in \mathbb{C}^{n\times n} \tag{6}$$

for x. In this way, multigrid is used to approximately compute M^{-1}.

To summarize, in solving (1) HELM2D uses an inner-outer iteration technique with a Krylov subspace method and multigrid acting as the outer and inner iteration, respectively.

In the present manual, some technicalities (for example, on multigrid) will be skip. These will be added later.

2 How to use

HELM2D was written in FORTRAN F77 language and is compatible to standard FORTRAN F77 compilers, like G77. It has been compiled on Intel FORTRAN 95. We, however, found some incompatibilities which require some adaptations. Thus, we recommend to use F77 compiler, e.g. G77 of GNU (see `www.gnu.org`).

HELM2D is packed in `h2d.tgz`. Put this file in a working directory and unpack the precompiled codes by using the command

```
tar xvfz h2d.tgz
```

To create an executable file, compile the `Makefile` file using the command

```
make
```

This will result in an executable `helm2d`. By using the command

```
helm2d
```

the software is executed.

3 Input file

The way HELM2D is executed and the way the problem is defined are stored in an input file `input.dat`, which is located on the same working directory. It is a simple ASCII file with some ordering which is read by the main program `helm2d.f`.

The input file can be divided into several parts which are related to problem definition (size of the domain, source location, boundary conditions and so on), outer iteration, inner iteration and output data.

3.1 Domain data

Domain data requires four data and this can be arbitrary depending on the problem that is wanted to be solved.

- `xl` $= L_x$ and `yl` $= L_y$ define the size of the domain in x and y direction, respectively, which forms a rectangular domain. `xl` and `yl` must be positive.
- `re(wave)` is the frequency (in Hz) and is any positive number. In case `xl` $=$ `yl` $= 1$, `re(wave)` represents the wavenumber.
- `im(wave)` is the so-called impedance (or damping) in the case where some uniform damping is needed. This damping term is expressed in terms of ratio with respect of the frequency or damping layer. Usually, $0 \leq$ `re(wave)` ≤ 0.1 is used.

3.2 Source position

Source position is given in `xs` and `ys`, which define a point (x_s, y_s). In the program, this point is transfered into the nearest grid point to (x_s, y_s).

3.3 Grid parameters

Once the computational domain is determined, a rectangular, equidistant grid is constructed as shown in Figure 1. For a domain shown in Figure 1, we define `ngx` and `ngy` as the number of grid points without counting the grid points on the boundaries. Thus, in our example, `ngx` $= 3$ and `ngy` $= 2$. Thus,

$$h_x = \frac{L_x}{ngx + 1}, \qquad h_y = \frac{L_y}{ngy + 1}. \tag{7}$$

For accuracy reason, we require that $h_x = h_y$. In case of absorption layer or PML added, the width of this layer (`wpml`) is defined as the fraction of the L_x and L_y. We usually set $0.2 \leq$ `wpml` ≤ 0.3.

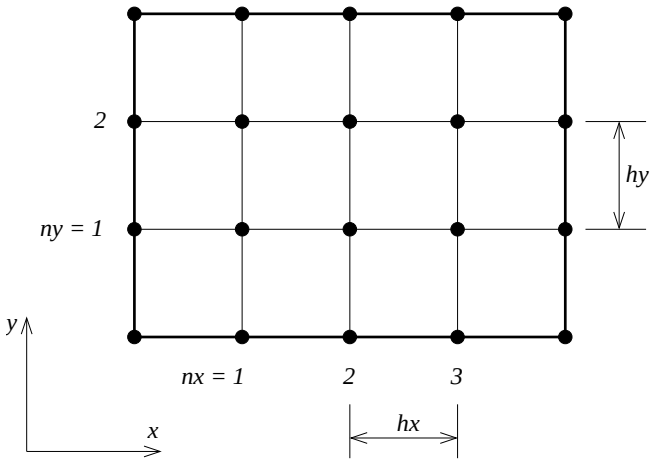

Figure 1: Grid definition

3.4 Problems

The parameter `icase` is used to build the data for the spatial variation of speed of sound in the domain. The function to build this data can be found in subroutine `wdata.f`. At this moment, we have 9 (nine) options for `icase`, i.e.:

- `irhs` = 0. Constant wavenumber in case of $L_x = L_y = 1$.
- `irhs` = 1. Two-layer problem in case of $L_x = L_y = 1$
- `irhs` = 2. Three-layer problem in case of $L_x = L_y = 1$.
- `irhs` = 3. Wedge problem in case of $L_x = L_y = 1$.
- `irhs` = 4. Constant wave number for $L_x, L_y \neq 1$. In this case, $c = 1500$ m/s.
- `irhs` = 5. Three-layer problem for $L_x, L_y \neq 1$.
- `irhs` = 6. Wedge problem in case of $L_x, L_y \neq 1$.

For c is given as an external data, modification has to be made. This will be incorporated later.

In case of absorption layer, spatial variation of damping in that layer is determined by a quadratic function

$$\alpha = C * |x_{al} - x_{bound}|^2, \tag{8}$$

where $C :=$ `coeffdl` is a positive constant. It is usually set equal to 0.3.

3.5 Boundary conditions

Refering to Figure 1, boundary conditions must be set on the west (`W`), east (`E`), north (`N`) and south (`S`). For temporarily, the only option is 2, which means radiation condistions.

4 Outer iteration

4.1 Krylov subspace

There are several outer iterations can be used to solve linear system (6). Among them are Bi-CGSTAB, GMRES, COCG and multigrid (as a solver). In this version, only Bi-CGSTAB is available. The rest is removed and will be incorporated again later. In this case,

- `itsol = 1`. Bi-CGSTAB.

In case of preconditioning, the second parameter `iprec` is important and hs two options

- `iprec = 0`. No preconditioner.
- `iprec = 2`. Preconditioned by shifted Laplace operator (5).

The parameters `ipsol, insol, inprec` are not relevant at this moment, and are left as it is.

4.2 Preconditioning `iprec = 2`

Refering to (5) we require two parameters to define the preconditioner for (1). Thus α_1 and α_2 must be given, and in the `input2d.dat` file, they are defined as `coeff1` and `coeff2`. It is found that the "best" option for these parameter is `coeff1` $= 1.0$ and `coeff2` $= 0.5$.

To build the preconditioning matrix M, the same boundary conditions must be applied to (5). Thus, `W, E, N, S` must be set equal to 2.

4.3 Convergence criteria

To terminate Krylov iteration process, two parameters are required: the maximum number of iterations `itmax` $\in \mathbb{N}$ and the relative residual at convergence. The latter is defined such that the relative residual is below some tolerance, namely

$$\frac{\|r^k\|_2}{\|r_0\|_2} < tolerance := 10^{\texttt{itol}}. \tag{9}$$

`itol` is a negative integer, and is usually from -5 to -7.

The other two parameters, `Itstart` and `Ittrunc` have no relevance whatsoever as `ipsol`,`insol`,`inprec` are not in use.

5 Inner iteration; Multigrid

Inner iteration is used to approximately compute M^{-1}. This is done by performing a few number of multigrid iterations. We skip some technical details about multigrid in this section.

Leave `npx` and `npy` as they are. In multigrid one uses a hierarchy of grids to efficiently reduce the residuals. This hierarchy of grids is usually identified by level. Each level corresponds to a subset of gridpoints after coarsening the finer grid level. Thus, in multigrid several levels of grid are used. This is indicated by `m` in the input file. In our method, the choice of m is rather arbitrary and does not follow the $2^{\texttt{m}}$ rule in standard multigrid. This allows us to use a sequence of grid points rather than e.g. 1, 2, 4, 16, 32, 64, 128, 256, $\cdots$.

For multigrid, some important components are

- `niter`. Positive integer, indicates the number of multigrid iteration. Our observation hints that `niter = 1` is sufficient.
- `npre`, positive integer, indicates the number of pre-smoothing. `npre = 1` is sufficient.
- `npost`, positive integer, indicates the number of post-smoothing. `npost = 1` is sufficient.
- `cycle` which indicates type of multigrid cycle. `cycle = 0` means V-cycle. `cycle = 1` is used for F-cycle. The best option is `cycle = 1`.
- `ids` which is used to choose the interpolation method for prolongation P. `ids = 0` is bilinear interpolation. `ids = 1` is prolongator of Dendy. `ids = 2` is that of de Zeeuw. The latter `ids = 2` is found to be the most robust one.

  In our case, coarse grid matrix is not built by using Galerkin method. Instead, a Petrov-Galerkin method is used. Thus

  $$M_H = RA_hP, \qquad \text{where } R \neq P^*, \tag{10}$$

  if M_H the coarse grid matrix and M_h the fine grid one. In this case, R is explicitly the full-weighting.
- `iswp` is used to choose the smoother. Point-Jacobi with relaxation `iswp = 2` is the best choice.
- *relax* is used in the smoother as the relaxation parameter ω. In general, ω can be a complex constant. For `iswp = 2` it is found that `relax(1)` $= \Re(\omega) = 0.5$ and `relax(2)` $= \Im(\omega) = 0.0$ are the best option.

6 Output parameter

There are two parameters:

- `iprt`. To save the solution, `iprt = 1`. If `iprt = 0` no solution is saved in a file. The solution is written in `solution.out`.
- `iprtres`. To save the residual data. If `iprt = 0` no residual is saved in a file. `iprt = 1`, residuals are saved. The data is written in `residual.out`.

Along side with this a `matlab-script` is included, which can be used to visualize the data.