

Author: Yann Capdeville
email: yann@seismo.berkeley.edu
Version 0.0, Berkeley, June 2003.

1 Prelude and warning

This is an attempt to provide a documentation to help the use of the coupled method spectral element - modal solution codes `hsemm_classic` and `hsemm_invert`. This documentation can also be more or less used for the normal mode born codes `scat_sem_partial` and `scat_sem_synthe`.

1.1 About what?

- `hsemm_classic` is the code to compute synthetics seismograms for a given source in a given model. This is the “classic” version of the coupled method code.
- `hsemm_invert` is very similar to the previous one, but has more specific stuff related to inversion with the Léane method (e.g. having several sources at the same time, different output format ...)
- `scat_sem_partial` is normal mode Born method used to compute partial derivatives for the Léane method when the reference model is spherically symmetric.
- `scat_sem_synthe` is different version of the previous code design to perform a similar task as `hsemm_classic` but with normal mode Born method. It’s extremely inefficient (not at all design for that), but it can be useful for testing.

1.2 References

!

For a full understanding of these programs, please refer to the following papers:

- about the spectral element methods applied to seismology: please read Dimitri’s thesis (in French) [?](#), and papers [?](#); [?](#); [?](#); [?](#)
- for the application of the spectral elements method to the global earth, please read Emmanuel Chaljub thesis (in French) [?](#) and papers [?](#), but also Dimitri’s papers [?](#); [?](#)
- about the coupled spectral elements/modal solution method, please read my thesis (in French) [?](#) and papers [?](#); [?](#); [?](#).
- about the inversion refer to not yet written....
- about the born codes, please read [?](#) and ... not yet written ...

If you use the coupled method within your work, you are strongly encouraged to cite [?](#) and [?](#) in the corresponding publication.

1.3 Who work on these programs?

The code has been written by mainly two persons, Emmanuel and Myself, but also, to a smaller extent, by Carene Larmat. Emmanuel wrote the bulk of the spectral element part and I wrote the coupling part. Please note that the attenuation part of the spectral element part has been developed, written and kindly provided by Dimitri Komatitsch and Jeroem Tromp. All the mode summation codes and born codes have been written by myself. The normal mode calculation parts in spherically symmetric models has been derived from the `minos` code written by J.F. Gilbert, J. Woodhouse, G. Master and maybe some others.

1.4 Warning

These codes have mainly been written in Fortran 95 with some small parts in C. The parallelization is done using the MPI library. Please note that these codes have an history and have changed of philosophy several times which explain, to some extent and mainly for the coupling part (I mean I'm mainly the one to blame), the internal incoherences When Emmanuel and I started that research in 1996, we were both new to F90 and MPI. A lot of stuff should be cleanly rewritten one day, but unless a really big problem occurs, and that one need to do a big, big fix, I'm afraid it will never be done and we will have to live with these annoying internal incoherences. One of the drawback of this is the performances could be probably better ...

An other problem is that most of the comments and error messages are written in French. I guess that when we start the code we were not sure that somebody else but us would use these codes and, at least for me, writing in English is always such a pain. I will try to correct that with time but it will take some.

2 `hsemm_classic`

This is the main version of the coupled method code.

2.1 Compiling

In order to make it work you first have to compile it. In the main directory (probably `hsemm_classic`), you first have to run the `configure` script which creates the `flags.mk` and `cpp_define` to be used by the different makefiles. The `configure` script is currently working on SUN, IBM power3, Digital alpha, SGI and Linux cluster (with intel compiler `ifc` and `mpich` MPI library) machines. Please note it's likely you will have to edit the `configure` script to take into account your specific configuration and compiler. After that, run `make` and if everything goes fine you will find a bunch of executables in the `bin_es` directory. For debug, you can use "`configure dbg`" command which produce executables made with the "`-g`" option in the `bin_as` directory.

If the code does not compile, well, contact me or try to fix it yourself ... a lot of compiling errors can be easily fix, but sometimes it's related to one of the incoherencies of the code, which makes things more complicated.

2.2 Running the codes

Here I assume you know how to run a MPI program on your particular machine. Typically , a mpi launch command, if you don't use any queue system will be: `mpirun -np 16 -machinefile`

`machinelist/path.to.code/bin.es/code.to.run`, where 16 is the number of processor you are going to use, `machinelist` the list of machines. `man mpirun` for more information about this command.

If you use a queue system like PBS it is a bit different. Please refer to `man pbs` for example for more information.

The spectral element code is run in two stages. The first one (usually named `generate_bdx.x`, where `x.x` will be a version number) computes the database. The second stage is the solver (usually named `hsemmx.x_MPI`) that solves the wave equation in your particular case.

2.3 Configuration

Most of the configuration files have a “.dat” extension. The main one is the “`macromesh.dat`” file. There is also the “`recepteur.dat`” which has the receiver list (only if the spectral elements reach the free surface, if not the file has to exist but the receiver list will not be used). You can find an example of these file in the `doc/config-example` directory. The next section is a guide for the “`macromesh.dat`” file.

2.4 The `macromesh.dat` file

Let first give an example of a `macromesh.dat` file, where the left column (lines number) has been added to help the explanation:

```

1 #####
2 #  config file for HSEMM
3 #  gather former config files macromesh.dat, data_temp.dat
4 #  interp.dat, source.dat, stock_freq.dat et
5 #  dir_database.dat. The file recepteurs.dat is still
6 #  separated
7 #####
8 #database directory (must exist) :
9 DATA
10 #Must the SEM of the database be on local node drive?
11 .true.
12 #Must backups be also done on local node drives?
13 .true.
14 #if on of two previous is .true. yes where?
15 /state/partition1/yann/ddp_17s_src2
16 #sem##macromesh.dat#####
17 #      Parmater for the mesh builde                                #
18 #                                                                 #
19 # Enter values after the 15th column                                #
20 #  after the "=" signn                                             #
21 #                                                                 #
22 # Shells are sorted by increasing radius                            #
23 #  there are (ncour + 1) interfaces :                               #
24 #                                                                 #
25 #      ----- interface (icour + 1)                               #
26 #      shell icour                                                #

```

```

27 # ----- interface (icour) #
28 # #
29 # 'nelh1' et 'degh1' are the parametres of the first #
30 # shell. These numbers for upper shells may change #
31 # if there is some nonconfirming interfaces #
32 # #
33 # take a look at domaine.f90 for some more info #
34 # or to the documentation (if I manage to do it!) #
35 # #
36 # to run on: specify: #
37 #      nproc      npro      square_npro #
38 # #
39 #      4          2        T #
40 #      8          2        F #
41 #      16         4        T #
42 #      32         4        F #
43 #####
44 npro      =4
45 square_npro =.false.
46 nregion   =6
47 parametrique =.false.
48 ncour     =2
49 ##### radius of the (ncour+1) interfaces
50 3480000.
51 3630000.
52 3900000.
53 ##### nature of the (ncour+1) interfaces
54 8
55 2
56 8
57 ##### geometry of the (ncour) shells
58 1
59 1
60 nelh1      =8
61 ##### nelv    of the (ncour) shells
62 1
63 2
64 degh1      =7
65 ##### degv    of the (ncour) shells
66 6
67 5
68 ##### Earth model
69 modele     =sawd
70 autograv   =.false.
71 attenuation =.true.
72 ##### visualisation
73 gen_pV3     =.false.

```

```

74 visu                =.false.
75 #sou##source.dat#####
76 # source info:
77 #
78 # source coordenates r (m), theta, phi (degrees)
79 5984100.0
80 93.83
81 153.93
82 # Tensor Mrr,Mtt,Mpp,Mrt,Mrp,Mtp
83 -1.65E+06
84 1.80E+05
85 1.47E+06
86 3.80E+05
87 8.00E+05
88 -4.40E+05
89 #geocentric correction?
90 .true.
91 #tem##data_temp.dat#####
92 #info for the time (time step, sources spectrum ect ...)
93 #
94 #time step (in second):
95 0.30
96 #number of time steps:
97 10000
98 #we keep only one time step over(isamp):
99 5
100 #we write on the drive only every isamp*nwrite time steps: nwrite=
101 20
102 #we backup every nwrite*isamp*nbackup time steps; nbackup=
103 10
104 #en cas de visu, refresh tout ivisu pas de de temps:
105 10
106 #source time function: (heavis or ricker):
107 heavis
108 #info for ricker: f0:
109 0.0220E0
110 #info for heavis:
111 1.00E-3
112 1.00E-2
113 2.60E-2
114 5.70E-2
115 #center of the source (t0)=
116 250.E0
117 #fre##stock_freq.dat#####
118 #info for the modal solution
119 #
120 Lmax= 2000

```

```

121 nb overtone      max = 400
122 number of DtN      = 2
123 name of the earth models files:
124 name of the model1 = part1
125 attenuation        =.true.
126 name of the model2 = part2
127 attenuation        =.true.
128   frequency minimum = 1.000E-05
129   frequency maximum1= 0.900E-01
130   frequency maximum2= 1.650E-01
131 #int##interp.dat#####
132 # tuning constant for the interposlation par: please leave like this
133 # unless you know what you are doing
134 # frequencies specified here must be smaller than the one of
135 # "stock_freq" section
136 #
137 12.0 #facteur du nombre maxi de beta points par element (*ngll^2)
138 1.50 #facteur pour Mcol (1<=> Mcol= nb points de gll sur l'equateur)
139 8.0 #const_nb_max_gll: nombre max de gll sur un parallele
140 8   #degre de l'interpolation de 2 vers 1 pour phi
141 8   #degre de l'interpolation de 2 vers 1 pour theta
142 0.080 #frequency max permanent part
143 0.160 #frequency max transitory part
144 #yan##yannos_flag#####
145 #force_fmin: unfoxed Fmin nor not?:
146 .false.
147 #Cancel_gravity
148 .true.
149 #Use_tref (for the attenuation)
150 .true.
151 #check_modes
152 .true.
153 #seuil_ray (level of rayleigh coef after which eigenfrequecies are discarded)
154 1.e-5
155 #l_startlevel
156 32
157 #etc#####
158 #time start coupling
159 000.
160 #time stop coupling
161 000.

```

In this example there is one shell on of spectral elements between a shell and a sphere of normal modes. This is a typical case when the D" is studied. There is absolutely no flexibility on the format. If a line is not exactly where it should, the program will crash. When the line start by a #, it means that line is comment and won't be read by the program. Beware: the # is not at all interpreted by the program to detect if a particular line is a comment or not, only the line number tells to the program if

this line is going to be used or not. The # is only here as a reminder.

- **line 9:** database directory. Indicates where the program can write the database he needs to run. This directory must exist and is not created by the `generate_bdx.x`.
- **lines 10-15:** You can decide to write a part of the database and self backup files on drive local to each nodes (it can be useful for linux cluster which do not have a high speed network and drives). In this example it will be the case. The directory `/state/partition1/yann/ddp_17s_src2` will be created if it doesn't already exist and used to store large database and backup files. Note: a "backup" allow you to restart the solver at the last backup point in case it has been stopped before it reaches the end (crash or manual stop).
- **lines 16-74:** Spectral element mesh design and Earth model properties on the spectral element part. See the next for more details. Not that below this the actual lines number for your particular application may change since the length of the Spectral element mesh design section may vary upon cases.
- **lines 75-91** Source informations. No further comment needed here.
- **lines 91-117** Time and spectrum informations.

The time step in this example is 0.3s and 1000 time steps will be computed. Only 1 time steps over 5 will be kept, which means that the output will have an actual time step of 1.5s. The number on line **101** specifies the frequency of output dumping on hard drive. Here 20 means that the output will be written on drive every $20 \times 5 = 100$ time steps and will be kept in memory in between. The same principle is applied for backup frequency: here 10 means that a backup ("checkpoint") will be done every $5 \times 20 \times 10 = 1000$ time steps.

On line **107**, you specify the type of spectrum source: "heavis" for heaviside or "ricker" (second derivative of a gaussian). Line **109** specifies the central frequency of the ricker, if selected. Lines **111-114** specify the 4 frequencies required to define the cosine taper frequency box used in the "heavis" case. Finally, the t_0 time on which to center the source wavelet is specified on line **116**.

- **117-130** Modal solution and frequencies for the storage of modal solutions. `Lmax` and `nb overtone max` are constants that are used for initial allocation of the arrays. If the actual value of ℓ_{max} and the number of overtone are greater than the one specify, the program will crash. For all classical application (up to now), you are safe with 2000 and 400. The number of DtN can only be 1 (Spectral element upper shell above a sphere of modal solution) and 2 (a sandwich of spectral element between two modal solution). If this number were 1, the lines **126** and **126** would not exist. Here `part1` and `part2` are the Earth models used in the 2 modal solutions. They can be generated by the `generate_prem` program and follow a similar syntax as the one used by the `minos` program. Finally they are 3 frequencies, the first one specifies where the such of normal modes will begin, the third where the search will stop in the frequency domain and the second where it will stop in the spectral domain (this is equivalent to provide a ℓ_{max} to the search). Note that these frequencies will be used for the storage only and not for the actual coupling. These one are in the following section.
- **lines 131-143** Tuning constant for the interpolation and coupling maximal angular order. The factors on line **137** to **141** unless you get an error message for the program asking you to change

then, which should occur only in exceptional situation. The frequency on line **142** is the one that determine the ℓ_{max} of the coupling. A good value should lies between 1.25 and 2. times the maximum frequency of the source. This frequency has to be smaller than the one on line **129**. The second frequency on line **143** is used to reach a good agreement of the modal solution and the asymptotic expansion of the DtN operator used in this scheme. A good value should lies between 1.5 and 2 times the frequency of line **142** but should be kept smaller than the one on line **130**.

- **lines 144-156** Flags used by the normal mode computation part. `force_fmin` is set to `.true.` in strange model, for example homogeneous. In regular models like PREM, leave to `.false.`. `Cancel_gravity` to cancel **all** gravity term. Note that as the code is now, you can't turn on the perturbation on the potential. `Use_tref` should be set to `.true.` if the attenuation is used. `check_modes` and `seuil_ray` should stay as it is, unless you know what you are doing. Finally `l_startlevel` specify after which angular degree the routine `startlevel` which determine at which radius should start the integration should be used. In some cases, in can be useful not to use `startlevel`, e.g. in homogeneous models. In regular models like PREM set to a small number (below 32).
- **lines 157-161** Experimental, do not use.

2.5 The Spectral element mesh part of the `macromesh.dat` file

In this section we describe more precisely the part of the `macromesh.dat` configuration file that design the spectral element mesh (**lines 16-74:** of the previous example). This configuration scheme come from E. Chaljub code. In order to understand the part you need a minimum knowledge of the mesh.

The first step is to defined, in a Cartesian reference coordinate system, a spatial discretization of the spectral element shell into non overlapping elements. The spectral element method puts the additional constraint that the spherical grid must be based on hexaedra. Furthermore, the spherical mapping should be as regular as possible and such that the elements of the mesh are not too distorted and the sampling of the grid-points as uniform as possible.

To satisfy the first constraint, we must be able, for each spherical interfaces of the shell, to tile a 2-sphere with quadrangles. This is achieved using the central projection of a cube onto its circumscribed sphere. Such a cubic-gnomic projection has been introduced by [?] and further extended by [?] as the “cubed sphere”. With the help of the central projection, the 2-sphere is decomposed into six regions isomorphic to the six faces of a cube. By choosing the coordinate lines within each region to be arcs of the great circles (see figure 1) six coordinate systems, with identical metrics, are obtained free of singularity.

Figure 1: The Gnomonic projection: great circles used to mesh one region of the “cubic sphere”.

With a constant angular distance between each great circles, a regular meshing of the six regions is defined in terms of deformed squares with uniform edge width. The surface mesh is quite uniform with a maximum distortion not exceeding 30%. The construction of a 3D mesh inside the spherical shell is then straightforward by simply radially connecting the quadrangles of two concentric discretized spherical interfaces. These interfaces can be mapped onto physical interfaces associated with

radial discontinuities within the mantle. As a result, the spherical shell is discretized into regular hexaedra (see figure 2) and the associated geometrical transformation is known analytically [?]. For each element, an invertible geometrical transformation can be defined allowing to map the reference cube.

Figure 2: Left: a splitted view of the six regions produced by the gnomonic projection which map the six faces of a cube inscribed inside the sphere to the surface of the sphere. Right: a gathered view of the six regions of the sphere. Each region has its own coordinates system and the coordinate transformations can be calculated analytically.

In the reference cube, a tensorial polynomial approximation of degree $N_h \times N_h \times N_v$ is used where N_h is the degree in the horizontal direction and N_v in the vertical direction. This defines a grid of $(N_h + 1)^2 \times (N_v + 1)$ integration points in each elements (see figure 3).

Figure 3: Projected view of a six regions used to mesh a spherical surface. Here, each one of the 6 regions are horizontally cut into 4×4 elements. The horizonatal degree on each elements is 8×8 (9×9 mesh points).

Now some parallel considerations. In order to be run on a parallel machine, the mesh has to be cut into pieces of the same size according to the number of processors in order to have the same charge of computation on each of them. The strategy chosen by E. Chaljub is fully describe in ? and ?, but here is what you need to know in order to be able to design the mesh.

Lets read the mesh part of the `macromesh.dat` file more or less line by line:
First the number of processor `nproc` must be of the form `npro2` or $2 \times \text{npro}^2$, where `npro` is specified on **line 44** in our example. If `nproc = npro2` the `square_npro` must be set to `.true.` on **line 45** and If `nproc = 2 \times \text{npro}^2` the `square_npro` must be set to `.false.` .
The `parametrique` parameter on **line 47** should be set to `.false.` as the non-parametric case is not ready (for topography of the interfaces for example). Here, there is no possibility to have a central cube and therefore the number of region `nregion` should always be set to 6 on **line 48**.

The mesh is discretized in the 3 directions (two horizontal and one vertical), but the distribution on processors is only performed horizontally. The vertical discretization of the mesh which is not parallel and therefore remain constant on each processor.

Vertically, the mesh is cuts into `ncour` rings (“couronne” in french, often abbreviate “cour” here), as specified on line 48 of our example. There are `ncour + 1` interfaces to bound these rings.

- The different radius of these interfaces are specified on **lines 50-51** and these radius do not correspond necessarily to a physical interface.
- Each interface has a code which describe the nature of the interface (**lines 54-56**) (see figure 4 for an example) :
 - 1 : free surface
 - 2 : solide-solide interface
 - 3 : non conforming solide-solide interface. That’s mean that the number of elements in each of the horizontal direction of the upper ring is the double of the lower one. This allow to decrease the horizontal number of elements with increasing depth.
 - 4 : solide-liquide interface. Non available.

- 5 : non conforming solide-liquide interface. Non available.
 - 6 : liquide-liquide interface. Non available.
 - 7 : non conforming liquide-liquide interface. Non available.
 - 8 : mode souple (DtN) interface.
- Each ring has “geometry” (**lines 58–59**) which can be only set to 1 here (2 is corresponding to the sphere – cube connection geometry and 3 to central cube geometry. These two geometry are temporary unavailable in the current development of the code).
 - Each ring has vertical number of elements (`nelv` on **lines 62 and 63**)
 - Each element has a vertical degree which remains constant within a ring (`degv`), specified on **lines 66-67**.

Figure 4: Vertical cross section of a spectral element mesh in one region. Here the number of ring would be 2. The lower ring has 6 vertical element and the upper one 2. The first interface is a DtN interface and it’s code is 8. The seconde interface is an non conforming interface and it’s code is 3. The upper interace can either be a free surface (code 1) or another DtN interace (code 8)

In any cases of `square_npro`, each one of the 6 region will cut into $npro \times npro$ pieces, called domains, of same size and will be affected to a particular processor, depending on the region (in order to optimize the communication).

After that, each domains is cut into $nelh1 \times nelh1$ elements where `nelh1` is set on **line 60**. If there is no nonconforming interface, this number will remain the same in all rings. If there one or more nonconforming interfaces, `nelh1` will remain the same until it will cross a nonconforming interface starting from the smallest radius. Each time a nonconforming interface is met, `nelh1` is multiplied by two. The horizontal degree `degh1` is set once for all and for all rings on **line 64**.

After that, the model to be used in the spectral element part is set on **line 69**. The list of possible model is given `src/l_mail/modeles_sub.f90`. To add a new possibility, please follow the instructions in the header of the module.

On **line 70**, you specify if you need gravity to be taken into account or not (in the cowling approximation).

On **line 71**, you specify if you need the attenuation scheme to be turn on or not.

Forget about the visualization part for the moment.

2.6 Stopping the solver while running: the `stop_en_run` file

The `stop_en_run` file is read by the solver at each time step. You can use it to do a clean stop of the solver in order to restart it latter. The `backup_startup` file is created by the solver at the first time step and can be edited. Here how it looks like:

```
Pour stopper, metre 1 ici :0
Vous pouvez choisir de changer le pas de temps final (avec 1 ci dessus)
Pas de temps final (00000=maintenant):00000
Pour ne pas backuper au stop metre 0 :0
```

If you want to stop the program and do a backup at the current time step put a “1” in place of the “0 ” of the first line. If you want to stop without doing a backup, put a “1” on the last line. The third line can be used to stop the program at a different time step than the one specified in the `macromesh.dat` file (to use this option, you also have to put a “1” on the first line).

To know at which time step the program has done its last backup, look at the `backup_startup` file. Note that if you want to restart the program from the beginning, you need to delete this file if it exists otherwise, the solver will try to restart from the time step specified in the `backup_startup` file.

2.7 `compute_size`: a little tool to help the design of the mesh and some rules to follow

On one hand, in order to design a stable mesh, you need to be sure that the time step dt follows at any points of the mesh the condition

$$dt \leq \frac{dx_{min}}{V_{max}}$$

where dx_{min} is the smallest distance between mesh points and V_{max} the maximum wave speed at this point.

On the other hand, in order to have a well sampled wavefield you need to have, at least between 4 or 5 points per minimum wavelength. If you use a degree 8 on elements, you can have two wavelengths per element. Note that if you decrease the degree on elements, you will probably need more than 5 points per minimum wavelength to be well sampled.

At the compiling stage, the little program `compute_size` will be generated. The program computes an estimation of the memory that will use the program in your specific configuration, but also help to check that your mesh is correctly designed and deal with the two conditions previously mentioned. As you know, the possible maximum time step and the maximum frequency for the source depend on the mesh and the model used. This program asks you to enter the maximum angular degree on the coupling, but this is required only to compute the memory size, and if you use it to tune your mesh, you can enter any value. The output will give you the possible maximum time step and the maximum frequency for the whole mesh but also in each rings which can help to detect incoherencies.

2.8 `green_extract` and `spline_extract`

`green_extract` allow to extract the green functions for the backup files produced at the run time by the solver to detect any problem in the modal solution.

`spline_extract` is similar but the continues part of the spectrum of the DtN operators. Any picks in these curves would probably mean that some eigenfrequencies have been missed.

3 To be done ...

- `generate_prem` program (to generate `part1` and `part2` models for example). Correct the little bug in it.
- translate in english comments in `compute_size`.
- explain problem that can occurs in the green functions.
- `recepteur.dat` versus `receiver.dat`.

- etc.

4 References