Author: Yann Capdeville
email: yann.capdeville@univ-nantes.fr
Version 0.1, Nantes, May 18, 2015.

1 Prelude and warning

The is an attempt to provide a (poor) documentation for different programs to compute normal modes and synthetic seismograms with normal mode summation in spherically symmetric Earth models. The present version (0.2) of these programs may not have been used extensively. For example the normal mode summation code (`nms`) is not the one I use usually to test and validate Spectral Element codes in the Earth, but have been specially developed for the occasion of the spice meeting from a time reversal code I wrote this year. This is to say that a lot of errors and bugs may still be present in the codes. Be warned if you used these codes for your own work (but if you do, a little thank you in the acknowledgment of your corresponding paper would be appreciated) .

1.1 Documentations? About what?

- `minosy` (alias `yannos`): this program computes normal modes (eigenfunctions and eigenfrequencies) for a given spherically symmetric Earth model in a given frequency band. An eigenfunction for given eigenfrequency $\omega_{q,n,\ell}$ is written as

$$\mathbf{u}(\mathbf{r}) = \left[{}_nU_\ell(r)\mathbf{e}_r + {}_nV_\ell(r)\boldsymbol{\nabla}_1 - {}_nW_\ell(r)(\mathbf{e}_r \times \boldsymbol{\nabla}_1) \right] Y_\ell^m(\theta,\phi)$$

  where q stands for spheroidal or toroidal, n is the radial order, ℓ the angular order and m the azimuthal order. Note that, because of the spherical symmetry, eigenfunctions and eigenfrequencies do not depend on m. $\boldsymbol{\nabla}_1 f(\mathbf{r}) = \partial_\theta f(\mathbf{r})\,\mathbf{e}_\theta + (\sin\theta)^{-1}\partial_\phi f(\mathbf{r})\,\mathbf{e}_\phi$. Y_ℓ^m is the spherical harmonic. The mode normalization is such that

$$\int_0^{r_\Omega} \rho(r)\left({}_nU_\ell^2(r) + \ell(\ell+1){}_nV_\ell^2(r)\right) r^2\,\mathrm{d}r = \int_0^{r_\Omega} \rho(r)\,\ell(\ell+1){}_nW_\ell^2(r)\,r^2\mathrm{d}r = 1\,,$$

  where r_Ω is the Earth radius. `minosy` computes $\omega_{q,n,\ell}$, ${}_nU_\ell(r)$, ${}_nV_\ell(r)$ and ${}_nW_\ell(r)$ for a given Earth model (note that we have ${}_nW_\ell(r) = 0$ for spheroidal modes and ${}_nU_\ell(r) = {}_nV_\ell(r) = 0$ for toroidal modes).

- `stock_boule`: Same as `minosy`, but for homogeneous sphere only. The eigenfunctions are computed analytically with Bessel functions, which can be sometimes be interesting for accuracy reasons.

- `nms`: this program computes synthetic seismograms for a given list of receivers and sources using the normal mode catalog produced by `minosy`.

- `get_fctp`: little program to extract a given normal mode from the output of `minosy`. It can be useful to visualize the eigenfunctions.

- `generate_prem`: little program to generate different kind of Earth models to be used by `minosy`.

- `generate_hmodel`: program that compute the homogenized, or residual homogenized version of given model. The output is also an Earth models to be used by `minosy`, but with layer correction information and local source correctors.

1.2 References

For normal mode computations, it can be useful to read Takeuchi & Saito (1972), Saito (1988), Woodhouse (1988) and for normal mode summation, e.g. Woodhouse & Girnius (1982). A good review of the subject can be found in Dahlen & Tromp (1998).

Dahlen, F. A. & Tromp, J., 1998. *Theoretical Global Seismology*. Princeton University Press. NJ.

Saito, M., 1988. DISPER80: A subroutine package for the calculation of seismic normal–mode solutions. In D. J. Doornbos (Ed.), *Seismological alogorithms*, pp. 294–319. Academic Press, New York.

Takeuchi, H. & Saito, M., 1972. Seismic surface waves. *Methods in computational Physics.*, **11**, 217–295.

Woodhouse, J. H., 1988. The calculation of eigenfrequencies and eigenfunctions of the free oscillations of the earth and the sun. In D. J. Doornbos (Ed.), *Seismological algorithms*, pp. 321–370. Academic Press, New York.

Woodhouse, J. H. & Girnius, T. P., 1982. Surface waves and free oscillations in a regionalized earth model. *Geophys. J. R. Astron. Soc.*, **78**, 641–660.

2 Compiling the package

You will need a Fortran 90 compiler to compile the package. Run first the `configure` script which will generate the `flags.mk` to be used by the different makefile. It will ask you what comiler do you have:

- `intel` for intel compiler (ifort)
- `gfortran` for GNU fortran 2003 compiler
- `pgf` not supported anymore ... but you can try (it uses to work)
- `xlf90` not supported anymore ... but you can try (it uses to work)
- `dec` not supported anymore ... but you can try (it uses to work)
- `sun` not supported anymore ... but you can try (it uses to work)

If you need another compiler, you can easily add an new entry in the `configure` script.

Run `make` or `gmake` at the top of the package tree and you should get all the executable files in `bin/` directory. If you don't ... well you will have to work a little bit more than expected or send me an email (try the first option first!).

3 Programs use

An example of each input file ("`.dat`") is given in the `examples` directory.

3.1 generate_prem

`generate_prem` produces some different Earth model ascii files to be used as an input by `minosy`. When using this program, an number of layers of 500 may be a good idea for period down to 50 s. It may be interesting to increase this number of layers for higher frequencies to get a good accuracy for attenuation, group velocity and Rayleigh coefficients. The output of format `generate_prem` (and input of `minosy`) is the following:

- line 1: name of the model
- line 2: `ifani`, reference frequency for attenuation dispersion, `ifdeck`. It is a good idea to keep `ifani` to one. Only `ifdeck=1` will be consider in the document.
- line 3: total number of model lines, index of the ICB, index of the OCB, number of line for the ocean (can be set to 0).
- line 4 and more: radius, density, V_{pv}, V_{sv}, Q_{kappa}, Q_{shear}, V_{ph}, V_{sh}, η. The format (in fortran) must be: `f8.0, 3f9.2, 2f9.1, 2f9.2, f9.5`
- last line: the model may contain also 5 more real for the layer correction. These 5 real are computed homogeneization program. They are required only if the shallow layer correction flag is set true.

3.2 MINOS

3.2.1 History

`minosy` (I used to call it `yannos`; I know, this is a stupid name that is why I changed that for the spice meeting) is my version of the MINOS program. MINOS is a program with long history which I only know partially. As I understand, at least 3 persons contribute to its development more than twenty years ago: F. Gilbert, J. Woodhouse and G. Masters, but it is unclear to me who exactly did what. It seems that the first version of the program was `eos` and then MINOS. Another normal modes program with a lot of common features with MINOS from J. Woodhouse is OBANI. In 1997 I started working on the coupling of spectral elements with modal solution. To compute the modal solution, I started from MINOS. Because most of MINOS was written with an old Fortran standard (a lot of "goto" and indexed "if") I had to translate many parts of the program to more standard Fortran because I wasn't able to work efficiently with the older standard. Finally, after finishing working with the coupling of spectral elements with modal solution, I also started to use my modified MINOS code to compute classical normal mode solution and it appears that this version is good enough. Compared to the original version, I didn't introduce anything new, it is even less efficient than the original version (not for the MPI case). The original version had some little bugs that make it difficult to use to get the full wavefield at relatively high frequency. If you look at `minosy`, you'll see that it is messy. That is true, but it is a mess that I'm able to understand which wasn't the case for MINOS. Finally, you should always be careful when using this program as I may have introduced bugs that weren't in the older version (it has been the case many times in the past). I'm sure that J. Woodhouse version would do a better job that this one.

3.2.2 Running the code

There is only one input file, `yannos.dat` that must be in the running directory. Here is an example (where line number have been added and should be removed for an actual run):

```
 1 #input earth model file (e.g. created with generate_prem) :
 2 prem
 3 #output information
 4 per_premR
 5 #prefix for output eigunfunction files
 6 fct_premR
 7 #type code (3=spheroidal, 2=toroidal, 0=both)
 8 3
 9 #precision (2) and switch off gravity perturbation frequency (eps1,eps2,wgrav)
10 1.E-10 1.E-10 10.
11 #lmin, lmax, fmin, fmax, nmax:
12 0 1500 0.1 15. 3000
13 ############if you need to output only some radius layers of eigenf
14 #number of layer for output:
15 1
16 #radius layers
17 3480000. 6371000.
18 #### computations flags.
19 # force fmin (usually F):
20 F
21 # Cancel Gravity (usually F)
22 F
23 # never use start level (usually F):
24 F
25 #use T ref (usually T):
26 T
27 #Check modes (usually F):
28 F
29 # use_remedy  awkward modes (usually T):
30 T
31 # rescue  (try to do something if missing a mode ) (usually T):
32 T
33 #restart (to restart in case of crash during computation
34 F
35 #force_systemic_search
36 F
37 # keep_bad_modes
38 F
39 #  modout_format (ipg, ucb, olm) (ipg is the standard for me)
40 ipg
41 # seuil_ray
42 0.0001
43 # l_startlevel
44 0
45 #layer correction (from homogeneization)
46 .false.
```

- line 2: input model (e.g. output from `generate_prem`).
- line 4: normal mode catalog information output in ASCII. It will give the eigenfrequency, group velocity, attenuation for each mode. The last two column are the Rayleigh Quotient which give an indication of the accuracy (the smallest is the best) and a logical that is .true. if

the mode is an inner core mode (not always accurate). (Rayleigh Quotient: if $\mathbf{H}$ is the elastic operator of the wave equation and if $\mathbf{u}_k$ the eigenmode associated with the eigenfrequency ω_k, therefore $(\mathbf{u}_k, \mathbf{H}\mathbf{u}_k) = \omega_k^2$ ($(.,.)$ is the inner product). In this file, the Rayleigh Quotient is $1 - (\mathbf{u}_k, \mathbf{H}\mathbf{u}_k)/\omega_k^2$ and should be equal to a small value if $\mathbf{u}_k$ is indeed a mode associated with the frequency ω_k).

- line 6: prefix of the eigenfunction output. If the "ipg" format is used, on binary file (`.direct`) and one ascii file (`.info`) will be created from this prefix.

- line 8: if 2 is entered, toroidal modes will be computed, if 3 spheroidal modes will be computed. If 0, both toroidal and spheroidal modes will be computed. In the output files, a "S" will be added for the spheroidal files and "T" for the toroidal ones.

- line 10: the two first numbers are respectively integration precision and precision on the eigenfrequency search. The last number is the frequency (in mHz) after which the gravity potential perturbation is not computed anymore. If you set this last value to 0, you will be in the Cowling approximation.

- line 12: the normal modes will be computed between angular orders `lmin` to `lmax` and the frequency `fmin` to `fmax` (in mHz). `nmax` in the maximum number of overtones that will be computed for each angular order. If you need the whole wavefield solution for a given frequency band, set `lmin` to 0 and `lmax` and `nmax` to large value (e.g. 5000). If you need only the fundamental branch, set `nmax` to 1.

- line 14 to 17: allow to choose one or more depth range where eigenfunctions are stored. Usually one layer for 700km depth up to surface is enough.

- **Unless you know what you are doing, bellow line 18, all flags should be set to default**

- `force_fmin` (line 20): In normal use, the start frequency search increases with the angular degree l. But for some situations, it can make sense to deactivate this feature by saying "true". Usual answer is "false"

- `cancel_gravity` (line 22): "true" means that ALL gravity terms will be set to 0.0 Usual value is ".false." Warning: if that option is set, the starting frequency search shouldn't be to low because it makes failed the alternate Bessel subroutine I'm using is that case. The consequence of this, is that the 0S1 and 1S1 may not be found. (for prem, fmin<0.05mHz is not good)

- `never_use_startlevel` (line 24): startlevel is subroutine that determine the integration starting point in depth. Usual value is ".false." If you have problems with some strange model (e.g. full homogeneous models) it may be a good idea to set this to ".true."

- `use_tref` (line 26) : Use the reference period of the model (for example: prem 1 s) Usual value is ".true."

- `check_modes` (line 28) : try to cancel some bad modes (some Stonley and inner core modes) Don't use it unless you know what you are doing. Usual answer is ".false."

- `use_remedy` (line 30) : remedy was build to recompute some modes like inner core modes. The original minos has it, but some problems come from this feature. Usual, I don't use it, but up to you. No Usual value.

- `rescue` (line 32) : Try to do something if the bracketing of modes fails. Us a systematic search for some frequency. It's not very efficient, and it can't heart anyway. Usual value is ".true."
- `modout_format` (line 34):
  - ucb: UC Berkeley
  - olm: old minos
  - ipg: my format

  Only "ipg" is implemented in the MPI version, but easy to do yourself. The `nms` program uses the ipg format.
- `l_startlevel` (line 44): If you use "startlevel", specify after which "l" it should do so. 0 is the standard value.
- `seuil_ray` (line 38) : cancels the mode if its Rayleigh quotient is larger than this value. 10^{-4} can be a good choice, but it really depends of what you are doing. It's really up to you. Information about the discarded modes will be found in the `.lost` file.
- `restart`: allows to restart after a crash and avoids to restart from start each time. For debug only.
- `force_systemic_search`: an alternate frequency search: don't use it.
- `layer correction`: do we need to use shallow layer correction (from homogenization)? If "`.true.`", then the earth model must contains an extra line form layer correction parameter information (see model description).

3.3 yannos_MPI version

For large frequency band, the computation can take time. In that case and if you have access to a parallel computer, the MPI version of `minosy`, `yannos_MPI` can be useful. To compile this version, you will need to set the MF90 variable in the flag.mk to the proper mpi f90 script (e.g. mpif90 for mpich) and execute `make yannos_MPI` in the `yannos/src` directory.

3.4 stock_boule

`stock_boule.dat` does the same job as `minosy` but for homogeneous sphere only using Bessel functions to compute the solutions (see Takeuchi & Saito 1972). There is a single input file: `stock_boule.dat`:

```
 1                Lmax= 0500
 2 number of layers   = 0600
 3 maximum of overtone= 0401
 4      rho (kg m^-3) = 03000.0
 5           vp (m/s) = 08000.0
 6           vs (m/s) = 06000.0
 7  bowl radius(m)    = 6.371E+06
 8  frequency minimum = 1.000E-08
 9  frequency maximum = 0.300E-02
10 prefix file Rayleig= fctS
11 prefix file Love   = fctT
```

where the format must be respected (the first 15 characters of each lines won't be used as an input).

- line 1 is the maximum angular number.
- line 2 is the number of radial sample that will be used to output the eigenfunctions.
- maximum number of overtones to be computed
- line 4 to 6: density, vp and vs
- line 8 and 9 frequency band in Hz
- line 10 and 11: prefix for spheroidal and toroidal eigenfunction files.

3.5 To read a single mode to ASCII format: `get_fctp`

`get_fctp` allows you to extract a particular eigenmode form the binary output of `minosy`. The output file form `get_fctp` is a two columns (depth , amplitude) ASCII file.
Usage:
`get_fctp eigenfile_prefix function_code n l`
with

- `eigenfile_prefix`: eigenfunction file prefix (file name without the "`.direct`" and "`.info`")
- `function_code`:
  - 1: ${}_nU_\ell(r)$
  - 11: $\frac{\partial\, {}_nU_\ell(r)}{\partial r}$
  - 2: $\gamma_\ell\ {}_nV_\ell(r)$
  - 22: $\gamma_\ell \frac{\partial\, {}_nV_\ell(r)}{\partial r}$
  - 3: $\gamma_\ell\ {}_nW_\ell(r)$
  - 33: $\gamma_\ell \frac{\partial\, {}_nW_\ell(r)}{\partial r}$

  with $\gamma_\ell = \sqrt{\ell(\ell+1)}$
- `n`: n
- `l`: ℓ

Remark: if $n < 0$ then all the available n in the file for the given ℓ will be output.

3.6 Normal mode summation program: NMS

There are 3 input files for `nms`: `nms.dat receivers.dat sources.dat`
`nms.dat`:

```
1 #time step
2 5
3 #number of time steps
4 2400
```

```
 5 #source t0:
 6 1000.0
 7 #source amplitude:
 8 1.e00
 9 #source frequency band (f1,f2,f3,f4):
10 0.10E-2
11 1.00E-2
12 1.20E-2
13 1.50E-2
14 #index min, index max of overtones to be used (-1,-1 for all)
15 -1 -1
16 #geocentric correction? (T=yes, F=no)
17 T
18 #component rotation? (T=yes, F=no)
19 T
20 #S eigenfunction file prefix
21 fct_premR
22 #T eigenfunction file prefix
23 fct_premT
24 # source force (T) or moment tensor (F)?
26 F
27 # output channel (1: displacement, 2: velocity 3: acceleration
28 2
29 #station response (yes=T, no=F).?. Yes implies to have response files ready
30 F
31 #tag homogenization: -1: none; 0: 0 order , 1: first order
32 -1
```

with

- line 2: time step in second
- line 4: number of time step
- line 6: central time of the source wavelet in second. In general, this number should be 0, otherwise your source wavelet will be truncated. To check this number is large enough, visualize the `source.gnu` file that will be generated by `nms`. If you need an origin time at 0, you will need to shift the traces afterward.
- line 8: amplitude of the source filter.
- line 10 to 13: definition of the source wavelet in the frequency domain. The spectrum is flat between `f2` and `f3` and a cosine tapper between `f1` and `f2` and between `f3` and `f4` is applied.
- line 15: overtone selection. e.g. 0 0 will select only the fundamental mode. (-1 -1 will take all the overtone)
- line 17: do we apply geocentric correction to source and receiver coordinate (see Dahlen and Tromp, p603)?
- line 19: do we apply component rotation?
- line 21 and 23: eigenmode catalog prefix output of minosy for spheroidal and toroidal modes respectively.

- line 26: most users will use a moment tensor source: put F. If you need a vector source force, put T (somehow, I'm not completely sure that the sign of the result will be correct). In the last case the Mrr, Mtt and Mpp of the sources.dat file will be used for the Fr,Ft,Fp component of the vector force.
- line 28: output channel 1: displacement, 2: velocity 3: acceleration. You must be **careful** with that. 1 is indeed displacement (2 velocity et ..) if the source is a step function in time. If the source need to be a Dirac (**which is usually not the case for earthquake)**, then 2 is displacement and 3 is velocity. To get acceleration, you need to take the time derivative of the channel 3. (see remark bellow)
- line 30: station response (yes=T, no=F).?. Yes implies to have response files ready. The response file must have the name `STATresponse` where `STAT` is the station name on four letters. An example of such a file is given later.
- line 32: tag homogenization: -1: none; 0: 0 order , 1: first order. If the model used to compute the normal modes is an output from `generate_hmodel`, we can use source and receiver correction. -1 indicates that no correction will be applied, 0 order 0 correction and 1, first order (the first order must be used with caution. Close to the free surface, it may be counterproductive if ε_0 is not small enough). The file `info_layer_corr.dat`, output of `generate_hmodel` must be present in the running directory.

`sources.dat` contains all the information about the source(s):

```
1 #nombre d'evenements
2 1
3 # stacking them (.true.) or not (.false.):
4 .false.
5 #n name      Mrr,     Mtt,       Mpp,     Mrt,      Mrp,       Mtp,
dep,   lat,   phi, half dur, tdelay
6 01 M012601A  1.04e+19 -0.43e+19 -0.61e+19  2.98e+19 -2.40e+19
    0.43e+19  10.00  03.30 095.94 000.00 000.00
```

Note line 5 and 6 have been cut to fit in the page.

- line 2: number of events to be computed
- line 4: in the case of several sources, should the be stacked in a single trace per receiver and component?
- line 6: number, name, moment tensor, depth, location of the event and half duration of the source. The last number is an extra time delay that can be applied to origin time given in `nms.dat`

`receivers.dat` contains all the information about the receivers:

```
1 #number of receivers
2 11
3 #name (4 characters), latitude, longitude in degrees
4 109C   32.889 -117.105
```

```
 5 _AAK   42.639   74.494
 6 ACSO   40.232  -82.982
 7 AHID   42.765 -111.100
 8 _AML   42.131   73.694
 9 ANMO   34.946 -106.457
10 ANTO   39.869   32.794
11 _APE   37.069   25.531
12 _AQU   42.354   13.405
13 _ARU   56.430   58.562
14 _ARV   35.127 -118.830
```

- : line2 : number of receivers
- line 4 to number of receivers +3: name (4 characters), latitude, longitude in degrees of each receiver.

`STATresponse`, for example `109Cresponse`:

```
 1 ZEROS 7
 2 -463.1000  -430.5000
 3 -463.1000  430.5000
 4 -176.6000  0.0000
 5 -15.1500  0.0000
 6 0.0 0.0
 7 0.0 0.0
 8 0.0 0.0
 9 POLES 11
10 -13300.0000  0.0000
11 -10530.0000  10050.0000
12 -10530.0000  -10050.0000
13 -520.3000  0.0000
14 -374.8000  0.0000
15 -97.3400  -400.7000
16 -97.3400  400.7000
17 -15.6400  0.0000
18 -0.0370  0.0370
19 -0.0370  -0.0370
20 -255.1000  0.0000
21 CONSTANT 8.902125e+26
```

To format is self consistent.

The outputs are 3 ASCII files for each couple source–receiver. The two first characters of each file will be `UZ`, `UR` and `UT` (for vertical, radial and transverse components) if the rotation is used and `UZ`, `UN` and `UE` (for vertical, North and Est components) if not.

A remark on the output **Channels** as this is often a source of confusion. With simplified notations

$$u = G * s$$

where u is the displacement at a receiver, G the green function from a given source location and s the source time function. For earthquake, we often have $s(t) = H(t) * f(t)$ where $f(t)$ is the time

derivative of the source time function and $H(t)$ the step function. The source definition (line 7 to 13) is intended to design $f(t)$. With such a definition, we have output channel 1: displacement, 2: velocity 3: acceleration.
Now, is some cases (other type of sources, comparison with other codes etc ..), we need to have $s(t) = f(t)$ with $f(t)$ still designed with lines 7 to 13. In that case, we need to take channel 2 as displacement and channel 3 as velocity. To get acceleration, you need to take the time derivative of the channel 3.

3.7 Layered homogenization and residual homogenization: `generate_hmodel`

`generate_hmodel` is a little program that allows to compute homogenized and residual homogenized spherically symmetric earth models as described by Capdeville & al (2007, 2008, 2013).
It uses a single input file: `hmodel.dat` Here is an example:

```
 1 #residual homogenization
 2 T 0
 3 #ref model
 4 prem
 5 # model name to be homogenized
 6 premt
 7 #the output will be for minosy (T) or SEM (F)
 8 T
 9 #starting radius for homogenization (km):
10 4000.d0
11 #main filter base (km)
12 100.d0
13 #base cst
14 8
15 #main filter in km
16 80
17 160
18 #second filter in km
19 300
20 600
21 #normal mode filtering?
22 F
23 #for model fiiltering:
24 ../../test2/smoothiso/fct_premfS
25 ../../test2/smoothiso/fct_premfT
26 ../../test2/smoothiso/premf
```

with

- line 2:
  - first flag is:
    - T residual homogenization;
    - F regular homogenization
  - second flag is an integer
    - 0: classical backup upscaling (as in Capdeville & all papers)

    - 1: only smooth elastic parameters (for test)
    - 2: only smooth velocities (for test)
    - 4: only smooth slowness(for test)

- line 4: for normal homogenization: model name to be homogenized. For residual homogenization, this is the reference model
- line 6: for normal homogenization: not used; For residual homogenization, model name to be homogenized with respect to line 4.
- line 8: explicit
- line 10: starting radius: must be above the CMB;
- line 12: support of the filtering wavelet (cst*base). Check the files (output) `filterwavelet1` and `filterwavelet2` to see visually is the support is wide enough.
- line 19 and 20: 1/k2 and 1/k1 of the cosine tapper filter used to compute $\mathcal{F}^{\varepsilon_0}$ (see Capdeville & all 2013)
- line 16 and 17: 1/k2 and 1/k1 of the cosine tapper filter used to compute $< . >$ filter (see Capdeville & all 2013). $< . >$ should be a constant average, but it is equivalent and more easy to use a cosine tapper filter, wider than $\mathcal{F}^{\varepsilon_0}$.
- line 21: do we use normal mode filtering instead of fourier filtering (see appendix B of Capdeville & al 2013).
  If yes
    - line 24: radial mode catalog computed in a very smooth model
    - line 25: spheroidal in the same model computed for at least $l = 1$. We shouldn't need that. It is a stupid big that I'm haven't corrected yet.
    - line 26: the smooth model itself.

The output model is always `premf`. It contains free surface correctors. It also output `info_layer_corr.dat` that is used by `nms` to compute source an receivers first order correction.