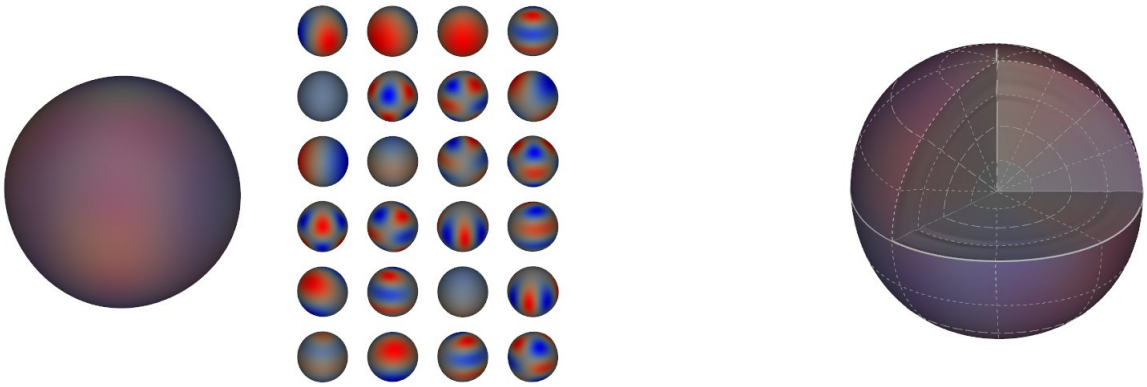
GLPULSE3D

Stéphane Charpinet

Institut de Recherche en Astrophysique et Planétologie (IRAP)
CNRS / Université Paul Sabatier (Toulouse III)
stephane.charpinet[at]irap.omp.eu

Version 1.0.0 - April 18, 2012



1 Introduction

GLPULSE3D is a program designed to simulate/visualize a pulsating star in 3 dimension (3D). Its main purpose is to provide a versatile, easy-to-use tool aimed at educational and public demonstration (the very first versions were developed back in 2009, in the context of the International Year of Astronomy). However, the code may also be a useful tool for assisting your reasearch activities in stellar pulsations or asteroseismology, as it can visualize radial eigenfunctions computed from pulsations codes applied to realistic stellar models. Examples of pulsations occuring in various types of stars are provided in the package. GLPULSE3D is written in C and uses OpenGL to exploit the accelerated 3D rendering capabilities offered by modern GPUs.

This program is freely released under the General Public License v3 (see the COPYING file) and usable by those interested in the community. Feel free to install and use it at your convenience. You can also modify the source code to suit better your needs if you wish (or, better, you could suggest some improvements that I may implement in future versions), but in any case, keeping the appropriate acknowledgements and credits for this program would be greatly appreciated. Hopefully you won't have major problems installing the code on your system. I can provide some very limited help if absolutely needed, but be aware that I don't have much time to spend on this.

Credit: Stéphane Charpinet (IRAP, Université de Toulouse / CNRS, France)

Thanks to: Masao Takata (University of Tokyo, Japan), Valérie Van Grootel (Université de Liège, Belgium), Gilles Fontaine & Pierre Brassard (Université de Montréal, Canada), Sylvie & Gérard Vauclair (IRAP), Alain Hui Bon Hoa (IRAP).

2 How to install it

The code should work with any Linux distribution and on Mac systems having a number of libraries installed. First, you need to have a working 3D-accelerated X-windows environment on your desktop/laptop using either an ATI or NVIDIA video card. Just make sure to set up your system properly for 3D acceleration before attempting to run GLPULSE3D. It might work without 3D acceleration, but it would probably be extremely slow (as the CPU would have to do both the real time calculation and the 3D rendering).

The libraries (with their associated header files) and packages that need to be installed on the system are :

- pkg-config (usually a standard package in modern Linux distributions; not necessary on a Mac)
- GL (usually installed with the X-server libraries)
- Glut (not always installed, you can install a package called freeglut)
- ftgl (generally not installed by default)
- freetype (usually a standard package in modern Linux distributions)
- fontconfig (usually a standard package in modern Linux distributions)
- gsl (the Gnu Scientific Library usually not installed by default)

Download the archive file containing the source code and "untar" it in your preferred directory:
$ tar xzvf glpulse3d-1.0.0.tgz

Then go into the newly created directory "glpulse3d-1.0.0" and check the content of the file called "Makefile" (choose first the Makefile suitable for your system: linux or mac). You shouldn't have to modify it except for the choice of the C compiler and the optimization flags. You can either use gcc (the standard C compiler on Linux/Mac systems) or icc (the Intel C compiler). With icc, the code seems to run 20 to 30% faster, but gcc will work fine. You may have to adapt the COPTS (optimization) flags to your own system if it doesn't compile/execute properly.

Build the program with the command:
$ make clean; make

and you should be ready to run it.

3 Running the code / key controls

To run the code, just call the executable followed by the name of a configuration file (several are provided; see Section 4), e.g.:

$ ./glpulse3d glpulse3d-example1.conf

A window pops up showing the 3D animation, which you can resize at your convenience or even switch to full screen mode by pressing the "F" key ("Maj + f"; the key controls are case sensitive). The program starts with the view showing the superposition of all pulsation modes defined in the configuration file (see again Section 4) and the animation is in real time. Depending on the values specified for the periods (frequencies) of the modes, you may want to accelerate time by pressing "+" to increase the time acceleration factor. The "-" key will allow you to decrease this factor and slow down the animation. You may also want to press "r" to activate the radial view and press "a" 3 times to show the pulsation axis, latitudes and longitudes. These are the most basic keys controlling the scene, but there are many more options, as listed below. Pressing the left or right buttons of the mouse while moving it will allow you to rotate the sphere to any viewing angle. The mouse wheel can also be used to zoom in and out.

Tip: If your graphic card is powerful enough, it is possible to improve the rendering quality by setting the anti-aliasing option up to whatever your card can support and handle. With Nvidia cards under Linux, this can be done with the program "nvidia-settings" (provided that you are using the video drivers from Nvidia).

List of key controls (case sensitive):

"**Q**" Exit the program.

"**F**" Enter/Leave full screen mode.

"**SPACE**" Enter/Leave the "decomposed view". In this view, along with the main representation, all the modes (up to 24) are also shown individually.

"**<**" In decomposed view, decrease the number of modes shown per row (minimum 1).

"**>**" In decomposed view, increase the number of modes shown per row (maximum 4).

"w" In decomposed view, if there are more modes than can be displayed at once on screen, scroll down.

"x" In decomposed view, if there are more modes than can be displayed at once on screen, scroll up.

"o" In decomposed view, increase the number of modes shown per column (maximum 6).

"l" In decomposed view, decrease the number of modes shown per column (minimum 1).

"Y" In decomposed view, lock/unlock the number of modes shown per column. By default, this number is locked, which means that it is automatically adjusted depending on the number of modes shown per row. Pressing **"o"** or **"l"** deactivate this automatic adjustment. You can come back to the default setting by pressing **"Y"**.

"**TAB**" Change the selection of the main representation (all modes combined, by default) and navigate among the individual modes.

"**BACKSPACE**" Change the selection of the main representation (all modes combined, by default) and navigate among the individual modes.

"+" Accelerate time.

"–" Slow down time.

"=" Set time flow to real time.

"**r**" Activate/Deactivate the radial view (i.e., open the star to show what is going on inside).

"**R**" Activate/Deactivate the radial view also for the "decomposed" views (the radial and decomposed views must be activated for this to have an effect).

"**a**" Show the pulsation axis and longitudinal, latitudinal, and radial grids. Press this key as many time as needed to cycle among the possibilities.

"**A**" Show also the grids for the "decomposed views" (the decomposed view must be activated).

"**i**" Change the textual information mode (cycle through the various modes by pressing this key).

"**L**" Change the language for the textual information (only French and English are available for now).

"**PAGEDOWN**" By default, the surface of the sphere shown is the surface of the star ($r = 1R_{\rm star}$). With this key, you can remove layers and expose deeper parts of the star.

'PAGEUP" Do the opposite of "PAGEDOWN".

"**LEFT**" When the radial view is activated, pressing this key opens up further in longitude the part of the sphere that is removed to show the interior.

"**RIGHT**" Do the opposite of "LEFT".

"**DOWN**" When the radial view is activated, pressing this key opens up further in latitude the part of the sphere that is removed to show the interior.

"**UP**" Do the opposite of "DOWN".

"**p**" Cycle through various rendering modes: plain (default), grid only, and vertices only.

"**c**" Cycle through various color schemes.

"S" Increase the ratio between diffuse and ambiant light (increases the contrast).

"D" Decrease the ratio between diffuse and ambiant light (decreases the contrast).

"**b**" Change the background color (black or white; white is better for screen shots sent to a printer).

"**2**" Zoom out (can also be done with the mouse wheel).

"**8**" Zoom in (can also be done with the mouse wheel). Note that there is no limit on the "zoom-in factor". Zooming in to much can lead to part of the sphere not being rendered on screen (they are clipped). Zooming in can be useful to see the details in the core when showing the radial view for instance.

"**5**" Set the zoom factor back to 1 (no zoom).

"**n**" Change the renormalization factor used to compute the amplitude of the combined deformations. Just adding-up contributions from individual modes of amplitudes scaled to be visible on screen can lead to global deformations that are two large (through constructive interferences). By default, the amplitude of the multi-mode representation is therefore scaled down to keep it reasonable for rendering purposes. If you don't want this renormalization, press "n" (and "n" again to reactivate it).

"**s**" Disable/Enable the rendering of the sphere. Only the radial view is shown, when enabled.

"**V**" Deactivate/Activate the radial deformation (ξ_r).

"**H**" Deactivate/activate the horizontal deformation (ξ_θ and ξ_ϕ).

"**Z**"' Change the quantity represented by the color variations. By default, this is the temperature perturbation. Other options are the radial displacement (ξ_r), the horizontal displacement (ξ_h), the dimensionless eigenfunctions y_1, y_2, y_3, and y_4. Note that these quantities are generally available when the real eigenfunctions (i.e., calculated with a adiabatic or non-adiabatic pulsation code) are provided to GLPULSE3D (see Section 4). When a generic radial eigenfunction is used, the temperature perturbation is just set to be $-\xi_r$ and the other quantities are simply set to zero.

"**C**" Deactivate/Activate the acknowledgment text. Feel free to do it, but of course giving proper credit and acknowledgments, if you use this program for your own illustrations/animations aimed at the public or the medias, will be greatly appreciated.

4 Configuration file

Several control parameters and the information on the oscillation modes that we want to represent are provided in a configuration file. Several examples of configuration files (ending with the suffix ".conf") are provided with the program. The global control parameters are the following (see also comments given in the configuration files):

language Define the language used at start by the program (can be changed afterward with the "L" key).

fontfile Indicate the filename of the TrueType font that will be used when rendering text on screen. You can choose any font that is installed on your system.

title Set the title of the animation. Note: use "underscore" ("_") to specify where there should be a blank in the text string.

frequencies Specify whether you provide the frequencies or the periods of the modes in this configuration file ("yes" means frequencies are used)

amplitudes Specify if the amplitude information provided for each mode is used. "yes" means that modes can be rendered with relative amplitude differences between them. "no" means that all modes will be set to the same amplitude.

deform_fac This parameter provides the global amplitude given to the oscillation modes. It's value needs to be adjusted in order to have the best rendering of the oscillations.

ampl_rescale_exp When "real" eigenfunctions are used, their amplitudes as a function of "r" can be rescaled nonlinearly following a power law of exponent specified by this parameter. This is useful to rescale amplitudes that can span several orders of magnitude inside a star in order to make them observable on screen.

expand_radius Specify if you want to remap nonlinearly the radial mesh using a power law. If "yes" is specified, then the following number is the exponent of this power law. This option is useful when activity is concentrated in a very narrow region of the star. It is the case for red giants, where a lot happens inside the helium core. But this core is tiny relative to the whole star on a linear radial scale. Remapping the radial grid with the appropriate exponent permits to expand the core and make it visible.

decomp_nxdiv Specify the number of modes shown per row in the decomposed view (minimum 1, maximum 4). If this value is not defined (commented out in the configuration file) the program will evaluate itself the optimal value. Note that the chosen configuration can be changed afterward while the program is running, with the keys "<" and ">".

decomp_nydiv Specify the number of modes shown per column in the decomposed view (minimum 1, maximum 6). If this value is not defined (commented out in the configuration file) the program will evaluate itself the optimal value. Note that the chosen configuration can be changed afterward while the program is running, with the keys "o" and "l".

mesh_nr Specify the number of mesh points on the radial grid. This value will depend on how precise you want the radial representation of the eigenmodes to be. For situations with modes having many radial nodes (like in red giants), it is useful to increase this number (up to 200 or 500). The price to pay is an increase of the computation load which your computer may handle or not. Note that it is also possible to change the non-radial grid resolution (in latitude and longitudes). But this has to be done in the source code. You can change the values of NLAT and NLONG given in the file "src/data.h" and recompile the code with the command **$make clean; make**. However, if you change these values, make sure that NLONG and NLAT are always multiples of 8.

The list of individual modes is then specified. A new mode is defined by the keyword "**mode**" followed by a number of parameters whose values must be given on the same line:

Id A short string giving a name to the mode. Underscore will be replaced by a "blank" on screen. If you don't want to provide a text there, just put a single underscore.

commentary A short string giving a commentary. Underscore will be replaced by a "blank" on screen. If you don't want to provide a text there, just put a single underscore.

l The degree of the mode

m The azimuthal order of the mode ($-\ell \leq m \leq \ell$)

k The radial order of the mode. Note that by convention, positive radial orders are for p-modes and negative radial orders are for g-modes.

Period/Freq Period or Frequency of the mode (in seconds or μHz). Provide one value depending of your choice to use frequencies or periods.

Ampl. Amplitude of the mode (only relative values between modes matter). These values are not used if the **"amplitudes"** parameter is not set to "yes"

Phase Phase of the mode (in second)

dxh/dxr This parameter can introduce (if the value differs from 1.0) an artificial amplitude factor between the horizontal displacement (ξ_h) and the radial displacement (ξ_r). This can be useful to enhance or decrease the relative importance of the radial vs non-radial deformations.

eigenfunctions This parameter indicates the name of a file where the "real" eigenfunctions of the mode are stored. An alternate option is to use ad-hoc (non physical) eigenfunctions computed from a simple analytical formula. If, instead of giving a filename, you specify the keyword "**cosinewave**" there, the radial and horizontal displacements as functions of the radius r will be set to $\xi_r = \xi_h \propto (r/R)\cos(\pi kr/R)$. The "temperature" perturbation eigenfunction used for color modulations is then set to$-\xi_r$.

Finally, note that "mode" lines can be commented out. Each line in the configuration file that starts with "#" is treated as a comment.

5 Providing real eigenfunctions to GLPULSE3D

A most interesting capability of GLPULSE3D is to simulate/represent the radial behavior of the oscillations based on "real" eigenfunctions. By real, I mean mode eigenfunctions previously calculated by a pulsation code applied to a realistic stellar model. This option permits to visualize what the pulsation may look like for all kind of stars, from the main sequence to the white dwarfs, including intermediate stages like the red giants and the extreme horizontal branch stars. The eigenfunctions must be provided in a separate file, called for each mode in the configuration file (see Section 4). A single file may contain the eigenfunctions for all modes (the program searches for the corresponding modes based on the given k and ℓ values) or several files (even one file per mode) can be specified. This file is one of the outputs of the pulsation code PULSE (Brassard et al. 1992, ApJS, 80, 725). It has the following format (see also examples):

For each mode, a six line header is expected (see examples). Only the value of ℓ and k (degree and radial order) are used and should be the same as the values given for the mode in the configuration file. Below this header, the columns then provide the following quantities (but not all are used by the code).

n (not used) mesh point number. The first line corresponds to the center of the star.

xi (not used) pulsation code integration variable.

y1 (optional) dimensionless y_1 eigenfunction (can be represented as a color modulation).

y2 (optional) dimensionless y_2 eigenfunction (can be represented as a color modulation).

y3 (optional) dimensionless y_3 eigenfunction (can be represented as a color modulation)

y4 (optional) dimensionless y_4 eigenfunction (can be represented as a color modulation)

ri (mandatory) radius (given as a fraction of the star total radius)

xir (mandatory) radial displacement $\xi_r \equiv r y_1$

xih (mandatory) horizontal displacement $\xi_h = \frac{Gm}{r^2\sigma^2} y_2$

dT/T (mandatory) temperature perturbation (the quantity selected by default for the color modulations). For adiabatic pulsations, note that $\frac{\delta T}{T} \simeq \nabla_{\rm ad} \frac{\delta P}{P}$ and $\frac{\delta P}{P} = V(y_2 - y_1 - y_3)$ with $V = \frac{\rho g r}{P}$.

wfi (not used) weight function.

In making your own files following this format, quantities that are not used by GLPULSE3D can be set to zero. Optional quantities can also be set to zero, but then if you select them to be color coded and represented on screen, you will see nothing interesting.

6 Making videos

GLPULSE3D is doing the calculations and the rendering on screen in real time. While this is very flexible, it is sometimes not appropriate when, for instance, one wants to show it on the internet or during a presentation. A solution to this limitation is to make a video capture of the animation.

For this, you will need a fairly powerful computer that is able to handle both the video capture and the real-time calculations of the code. The best method for a smooth recording is to create a ramdisk. You will need root access on your computer to do this. If you don't have this access, you can skip steps 1 to 3 and record directly on your hard-drive.

1. as the superuser: add in your "/etc/fstab" file the line "**none /mnt/ramdisk tmpfs defaults,size=2G,users 0 0**". You also need to create a directory called ramdisk in '/mnt/' if it does not exist. The size limit (here 2Gb) can be changed to fit your system.

2. As a normal user: **$ mount /mnt/ramdisk**

3. go to the ramdisk: **$ cd /mnt/ramdisk**

4. Start GLPULSE3D and move the window in the upper left corner of your screen

5. In the ramdisk directory, type: **$ ffmpeg -f x11grab -r 25 -s 850x550 -i :0.0+nomouse -vcodec libx264 -preset ultrafast -crf 0 video.mkv**
   In this example, the animation is being captured at 25fps. You can then control GLPULSE3D as usual during the recording. To stop recording, just do Ctrl+C where the above command has been issued.

6. When the video has been recorded, you can do another pass to crop the sides of the video (and remove unwanted parts of the screen that were captured): **$ ffmpeg -i video.mkv -vf crop=in_w-70:in_h-60:10:30 -vcodec libx264 -preset slow -crf 0 video2.mkv**

Other video formats can be created using ffmpeg or similar softwares.

7 On the calculation of displacement eigenvectors

GLPULSE3D computes in real time spherical harmonics and their derivatives for rendering temperature waves (colors) and physical deformations of the star. For the latter, the code computes at each grid point and at each time step the displacement vectors, $\vec{\xi}_{n\ell m} = \{\vec{\xi}_r, \vec{\xi}_\theta, \vec{\xi}_\phi\}_{n\ell m}$ of each mode and sums them up (with a renormalization factor to keep them at reasonable amplitudes for visualization) to produce the multi-mode representation. In spherical coordinates, the components of the displacement eigenvectors are given by:

$$\begin{aligned} \vec{\xi_r} &= \xi_r(r) Y_\ell^m(\theta,\phi) e^{i\sigma t} \vec{e_r} \\ \vec{\xi_\theta} &= \xi_h(r) \frac{\partial Y_\ell^m}{\partial \theta} e^{i\sigma t} \vec{e_\theta} \\ \vec{\xi_\phi} &= \xi_h(r) \frac{1}{\sin\theta} \frac{\partial Y_\ell^m}{\partial \phi} e^{i\sigma t} \vec{e_\phi} \end{aligned} \tag{1}$$

The spherical harmonics, $Y_\ell^m(\theta,\phi)$, are expressed in terms of the normalized associated Legendre polynomials, for $\ell = 0,1,$ ... and $m = 0, ..., \ell$:

$$Y_\ell^m(\theta,\phi) = \bar{P}_\ell^m(\cos\theta) e^{im\phi} = N_\ell^m P_\ell^m(\cos\theta) e^{im\phi} \qquad \text{with} \qquad N_\ell^m = \sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}} \tag{2}$$

For negative values of m, the relations $\bar{P}_\ell^{-|m|}(x) = (-1)^{|m|} \bar{P}_\ell^{|m|}(x)$ or $Y_\ell^{-|m|}(\theta,\phi) = (-1)^{|m|} Y_\ell^{|m|}(\theta,\phi)$ are used. Derivatives of the spherical harmonics have to be computed. For the simplest case, we have

$$\begin{aligned} \frac{\partial}{\partial\phi} Y_\ell^m(\theta,\phi) &= im \bar{P}_\ell^m(\cos\theta) e^{im\phi} \\ &= im Y_\ell^m(\theta,\phi) \end{aligned} \tag{3}$$

The differentiation relative to θ can be obtained with recurrence relations between associated Legendre polynomials, we have:

$$\begin{aligned} \frac{\partial}{\partial\theta} Y_\ell^m(\theta,\phi) &= \frac{d\cos\theta}{d\theta} \frac{d}{d\cos\theta} \bar{P}_\ell^m(\cos\theta) e^{im\phi} \\ &= -\sin\theta \frac{d}{dx} \bar{P}_\ell^m(x) e^{im\phi} \qquad \text{where} \qquad x = \cos\theta \end{aligned} \tag{4}$$

Starting from the recurrence relation (for $m \geq 0$)

$$(1-x^2)\frac{d}{dx} P_\ell^m(x) = (\ell+1) x P_\ell^m(x) - (\ell-m+1) P_{\ell+1}^m(x) \,, \tag{5}$$

we can write (for $m \geq 0$)

$$\begin{aligned} \frac{d}{dx} \bar{P}_\ell^m(x) &= N_\ell^m \frac{d}{dx} P_\ell^m(x) \\ &= \frac{1}{1-x^2} \left\{ (\ell+1) x \bar{P}_\ell^m(x) - (\ell-m+1) \frac{N_\ell^m}{N_{\ell+1}^m} \bar{P}_{\ell+1}^m(x) \right\} \,. \end{aligned} \tag{6}$$

For negative m-indices, this derivative simply obeys the relation:

$$\frac{d}{dx} \bar{P}_\ell^{-|m|}(x) = (-1)^{|m|} \frac{d}{dx} \bar{P}_\ell^{|m|}(x) \,. \tag{7}$$

Considering that $1 - x^2 = 1 - \cos^2\theta = \sin^2\theta$, the above equations lead to

$$\frac{\partial}{\partial\theta}Y_\ell^m(\theta,\phi) = \frac{1}{\sin\theta}\left\{(\ell - m + 1)\frac{N_\ell^m}{N_{\ell+1}^m}Y_{\ell+1}^m(\theta,\phi) - (\ell+1)\cos\theta Y_\ell^m(\theta,\phi)\right\} . \tag{8}$$

Taking the real part of these expressions, the components of the displacement eigenvector for a given mode are then

$$\begin{aligned}
\vec{\xi}_r &= \xi_r(r)\bar{P}_\ell^m(\cos\theta)\cos(m\phi + \sigma t)\ \vec{e_r} \\
\vec{\xi}_\theta &= \frac{\xi_h(r)}{\sin\theta}\left\{(\ell - m + 1)\frac{N_\ell^m}{N_{\ell+1}^m}\bar{P}_{\ell+1}^m(\cos\theta) - (\ell+1)\cos\theta\bar{P}_\ell^m(\cos\theta)\right\}\cos(m\phi + \sigma t)\ \vec{e}_\theta \\
\vec{\xi}_\phi &= -m\frac{\xi_h(r)}{\sin\theta}\bar{P}_\ell^m(\cos\theta)\sin(m\phi + \sigma t)\ \vec{e}_\phi
\end{aligned} \tag{9}$$

The deformations are then calculated by applying these vectors to the unperturbed sphere at each grid point and at each time step. The perturbed grid points are then projected from the spherical coordinate system on to the Cartesian reference frame in which the OpenGL scene is displayed.