

The background features a series of concentric circles in white and light gray against a dark red-to-blue gradient. Several arrows point clockwise around these circles, creating a sense of motion.

# GREAT

PABLO CERDÁ DURÁN

# GREAT: GENERAL RELATIVISTIC EIGENMODE ANALYSIS TOOL

- Free to use (GNU General Public License)
- <https://www.uv.es/cerdupa/codes/GREAT>
- Based on Torres-Forné et al 2018 ([arXiv:1708.01920](https://arxiv.org/abs/1708.01920)) and Torres-Forné et al 2019 ([arXiv:1806.11366](https://arxiv.org/abs/1806.11366)).

# METRIC

General 3+1 metric

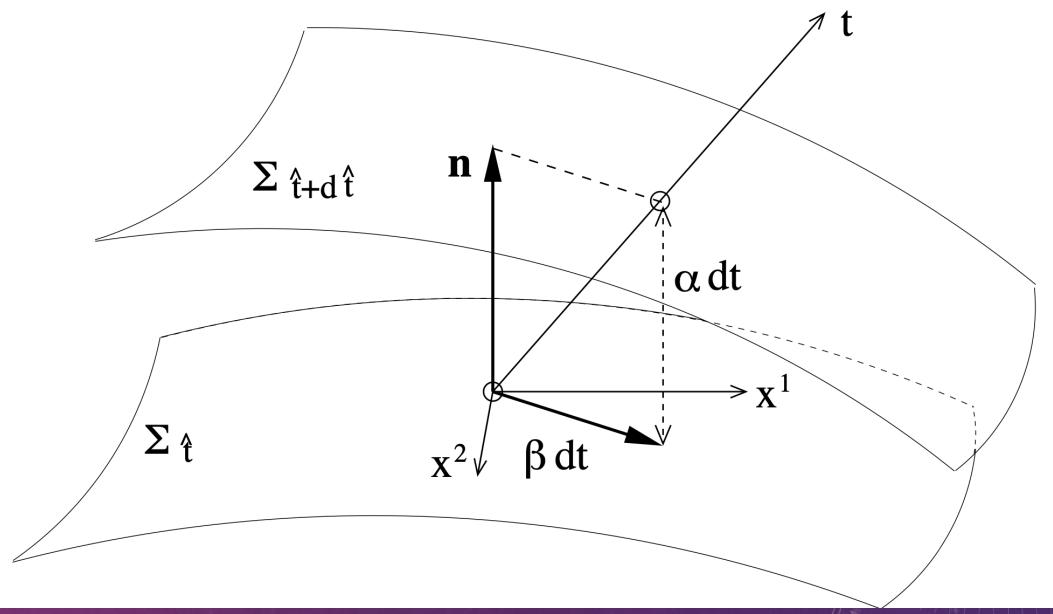
$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu = (\beta^i \beta_i - \alpha^2) dt^2 + 2\beta_i dt dx^i + \gamma_{ij} dx^i dx^j,$$

$\alpha$  : lapse  
 $\beta^i$  : shift vector  
 $\gamma^{ij}$  : 3-metric

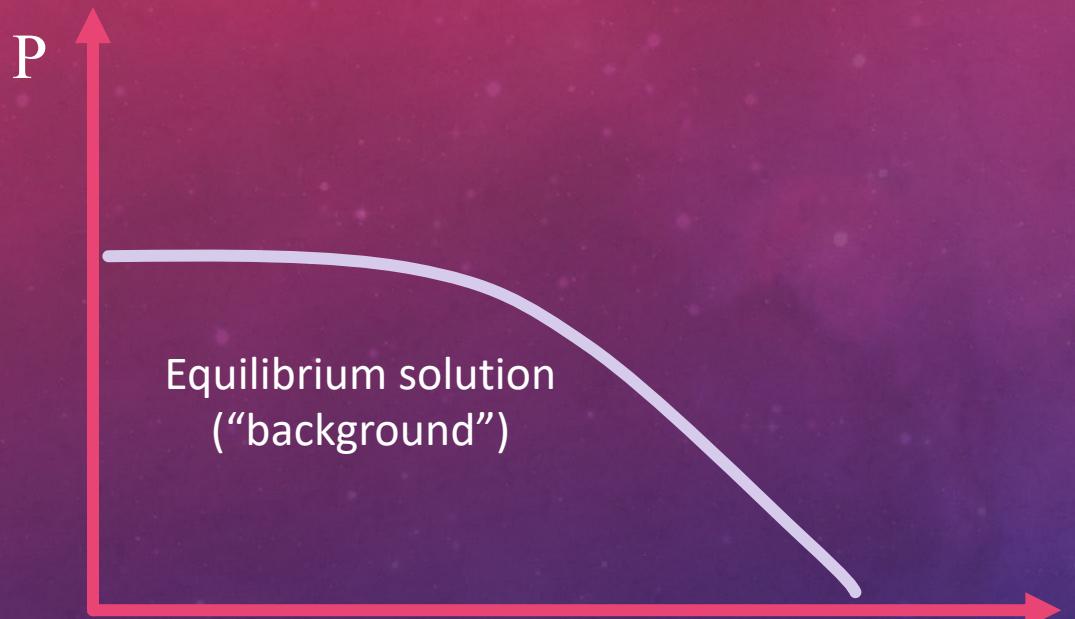
General spherical metric in isotropic coordinates

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu = -\alpha^2 dt^2 + \psi^4 f_{ij} dx^i dx^j$$

$\psi$  : conformal factor



# LINEAR PERTURBATIONS



TOV equations  
(isotropic  
coordinates)

Hydrostatic equilibrium  
equations

$$\nabla^2 \psi = -2\pi\psi^5 E$$

$$\nabla^2 Q = 2\pi Q\psi^4(E + 2S)$$

$$\frac{1}{\rho h} \partial_i P = -\partial_i \ln \alpha \equiv G_i$$

Spherical  
symmetry

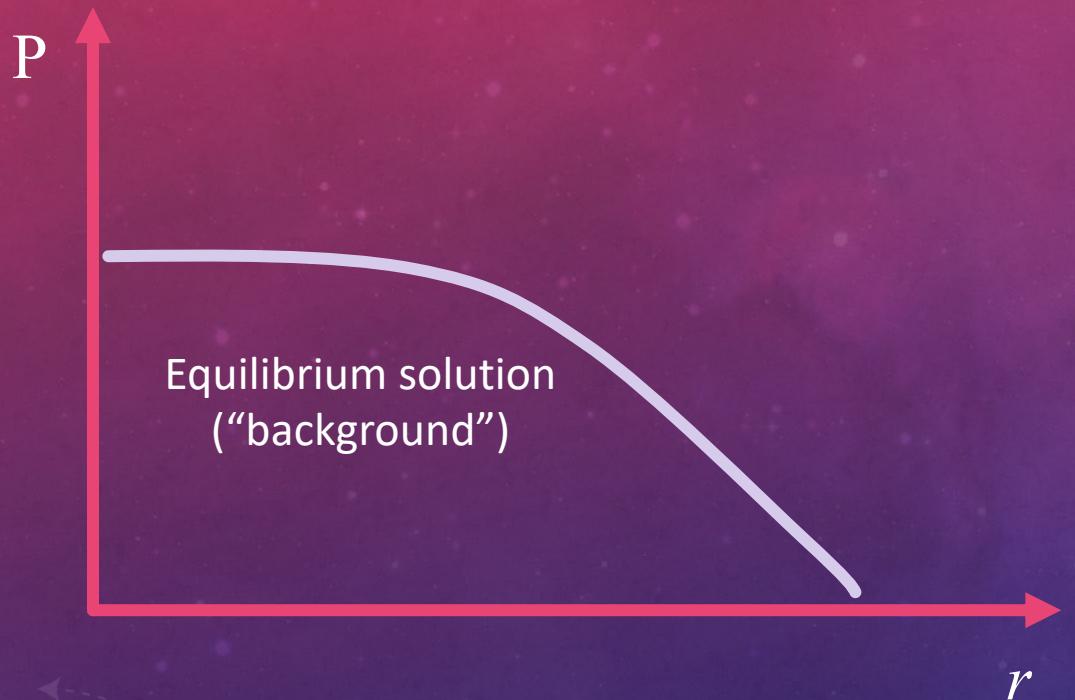
Newtonian  
limit

$$Q := \alpha^\circ \beta$$

$$\nabla^2 U = 4\pi\rho$$

$$\frac{1}{\rho} \partial_r P = \partial_r U$$

# LINEAR PERTURBATIONS

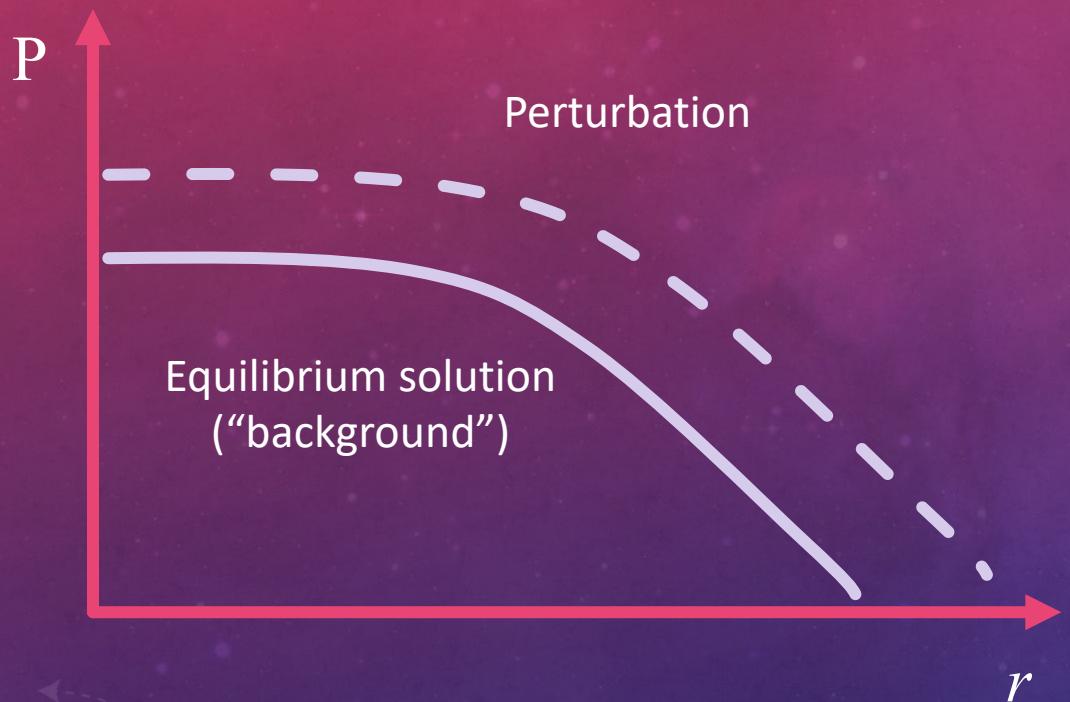


Hydrostatic equilibrium  
equations



$\rho(r), P(r) \dots$   
 $\alpha(r), \psi(r) \dots$

# LINEAR PERTURBATIONS



Hydrostatic equilibrium  
equations

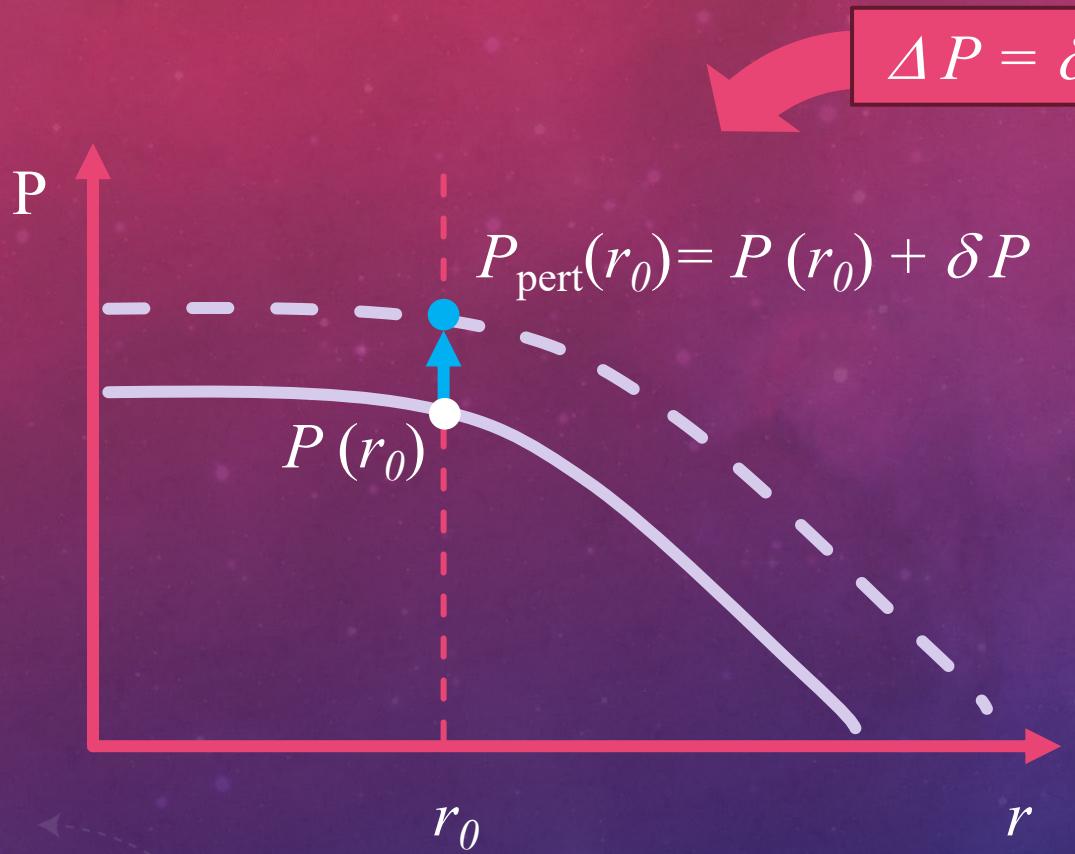


$\rho(r), P(r) \dots$   
 $\alpha(r), \psi(r) \dots$



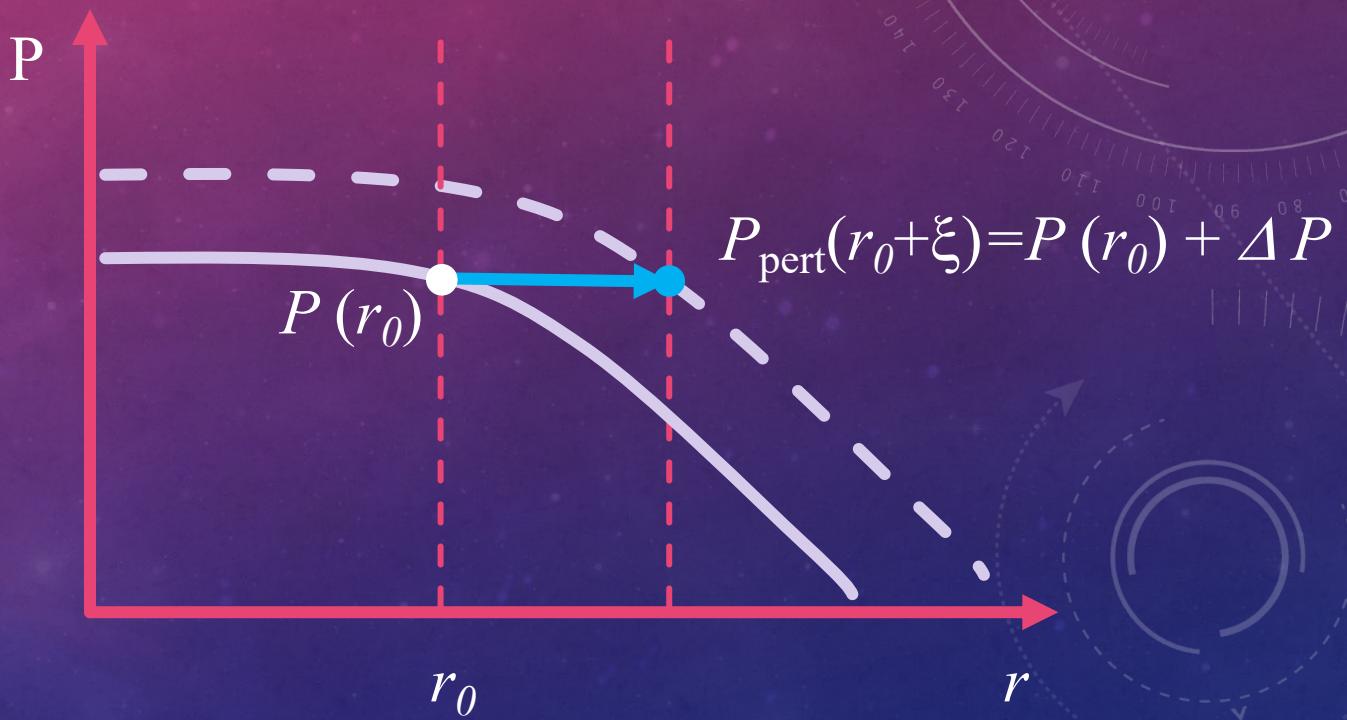
$\rho_{\text{pert}}(r), P_{\text{perp}}(r) \dots$   
 $\alpha_{\text{perp}}(r), \psi_{\text{perp}}(r) \dots$

# LINEAR PERTURBATIONS



Eulerian perturbation:  $\delta P$

$$\Delta P = \delta P + \xi^i \partial_i P$$

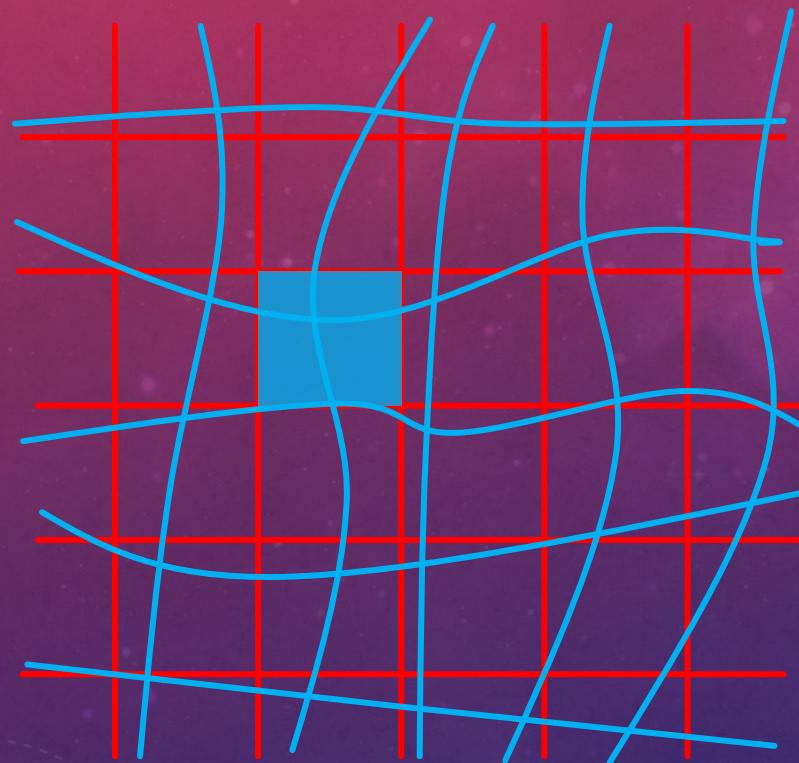


Lagrangian perturbation:  $\Delta P$   
Lagrangian displacement:  $\xi$

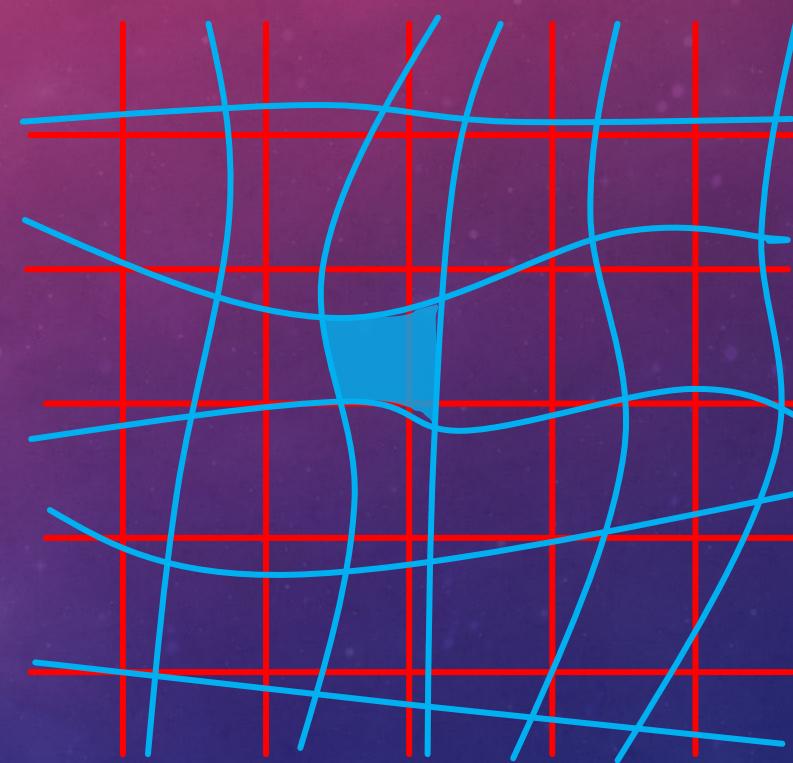
$$\partial_t \xi^i = \delta v^{*i}$$

$$v^{*i} = \alpha v^i - \beta^i \quad : \text{"advective" velocity}$$

## LINEAR PERTURBATIONS



Eulerian frame



Lagrangian frame

# LINEAR PERTURBATIONS

$$\delta P = \delta \hat{P} Y_{lm} e^{-i\sigma t}$$

$$\xi^r = \eta_r Y_{lm} e^{-i\sigma t}$$

$$\xi^\theta = \eta_\perp \frac{1}{r^2} \partial_\theta Y_{lm} e^{-i\sigma t}$$

$$\xi^\varphi = \eta_\perp \frac{1}{r^2 \sin^2 \theta} \partial_\varphi Y_{lm} e^{-i\sigma t}$$

Spherically symmetric background

- (Vector-)spherical harmonic decomposition
- Harmonic time dependence



Radial functions

$$\eta_r(r), \eta_\perp(r), \delta \hat{P}(r), \delta \hat{\rho}(r), \\ \delta \alpha(r), \delta Q(r)$$

Cowling: Torres-Forné et al 2018  
Non-Cowling: Torres-Forné et al 2019

# LINEAR PERTURBATIONS

$$\partial_r \eta_r + \left[ \frac{2}{r} + \frac{1}{\Gamma_1} \frac{\partial_r P}{P} + 6 \frac{\partial_r \psi}{\psi} \right] \eta_r + \frac{\psi^4}{\alpha^2 c_s^2} (\sigma^2 - \mathcal{L}^2) \eta_\perp = 0$$

$$\partial_r \eta_\perp - \left( 1 - \frac{\mathcal{N}^2}{\sigma^2} \right) \eta_r + \left[ \partial_r \ln q - G \left( 1 + \frac{1}{c_s^2} \right) \right] \eta_\perp = 0$$

4 boundary (regularity) conditions at  $r=0$

Give a value for  $\sigma^2$

$\eta_r(r), \eta_\perp(r)$

Outer boundary condition  
 $\eta_r(r_{\max})=0?$

Search for  $\sigma^2$  (eigenvalues)

**Cowling** ( $\delta \alpha = 0, \delta Q = 0$ )

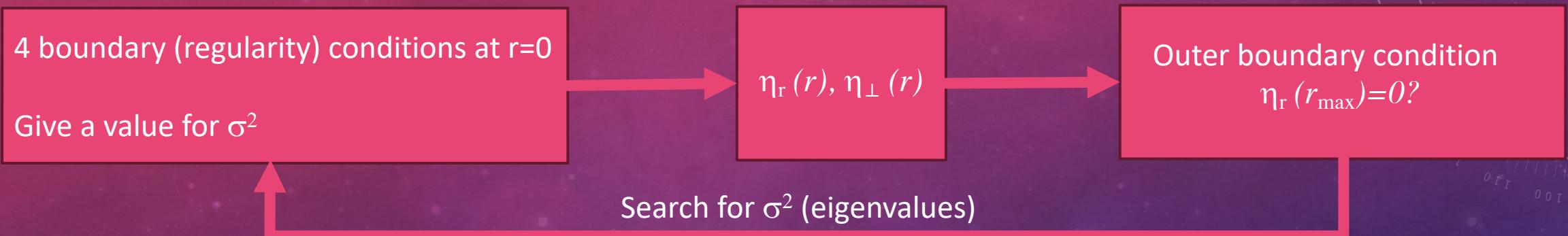
2 second order ODEs

2 unknown variables:  $\eta_r(r), \eta_\perp(r)$

1 unknown constant:  $\sigma^2 \in \text{Reals}$

Sturm-Liouville problem

# LINEAR PERTURBATIONS



Sign of  $\sigma^2$ :

$\sigma^2 > 0 \rightarrow \sigma$  real

$\rightarrow$  oscillation (stable) mode

$\sigma^2 < 0 \rightarrow \sigma$  purely imaginary  $\rightarrow$  unstable mode

We want all possible values of  $\sigma$  (within a range), not only one.

**Root finder**

- 1.- Coarse search in  $\sigma^2$   
(bound solutions in intervals)
- 2.- Refined search (bisection)

# GREAT LIBRARY

## Access library

```
use module_param  
use module_background  
use module_eigen  
use module_mode_analysis
```

Modules from GREAT that will be used

## Parameters

```
type(parameters_t) :: param
```

Define a parameters type (similar to a struct in C)

```
call Read_parameters(param, "parameters", ierr)
```

Read parameters file

```
param%  
param%newtonian  
...  
...
```

Access the different parameters (same names as in parameters file)

# NOW OPEN DOCKER

## 1. Close other docker sessions (e.g. compose\_env)

- You can see if it is open with:

```
docker ps -a
```

- To close it:

```
docker attach compose_env
```

```
exit
```

## 2. Follow the instructions sent by email to run the container

## 3. Update to the latest version of great

- Run inside the great directory:

```
git pull
```

- If you already tried the test, you may have modified files

```
git checkout <file>
```

## 4. Open an additional terminal/window with your local files

# GREAT LIBRARY

## Background data

```
type(BGData_t) :: data
```

```
call Init_background_data (data, m, ierr)
```

```
real*8 :: time
```

```
integer :: nt
```

```
integer :: iR
```

```
character (len=255) :: output_directory
```

```
real*8, dimension (:), allocatable :: r
```

```
real*8, dimension (:), allocatable :: rho
```

```
real*8, dimension (:), allocatable :: eps
```

```
real*8, dimension (:), allocatable :: p
```

```
real*8, dimension (:), allocatable :: c_sound_squared
```

```
real*8, dimension (:), allocatable :: phi
```

```
real*8, dimension (:), allocatable :: alpha
```

```
call Impose_boundary_conditions_bg (data, ierr)
```

```
call Free_background_data (data, ierr)
```

See e.g. great/src/test.f90

Define a background data type (similar to a struct in C)

Initialize background arrays of size m

Time of the profile

Number of the file

Shock location (index of the last point INSIDE the shock)

Directory for the output

Radius

Rest mass density

Specific internal energy

Pressure

Speed of sound square

Conformal factor

Lapse

Fill ghost cells needed to compute background derivatives

Deallocate (free) memory. Call at the end.

# GREAT LIBRARY

## Compute eigenvalues

```
call Perform_mode_analysis (data, param, ierr)
```

Compute the eigenvalues

```
call Print_error_message (ierr)
```

If error ( $ierr \neq 0$ ), print error message

## Output

```
call Output_background_data(data, param, ierr)
```

Output data to file

See e.g. great/src/test.f90

# GREAT UNITS

**G=c=1 (geometrized) & length in cm**

Magnitude	Dimensions	cgs	→	GREAT units	Conversion factor
Length	L	cm	→	cm	1
Time	T	s	→	cm	$c = 2.998 \times 10^{10} \text{ cm s}^{-1}$
Frequency	$T^{-1}$	Hz	→	$\text{cm}^{-1}$	$1/c = 3.336 \times 10^{-11} \text{ s cm}^{-1}$
Density	$M L^{-3}$	$\text{g cm}^{-3}$	→	$\text{cm}^{-2}$	$G/c^2 = 7.426 \times 10^{-28} \text{ cm g}^{-1}$
Pressure/energy density	$M L^{-1} T^2$	$\text{g cm}^{-1} \text{s}^{-2}$	→	$\text{cm}^{-2}$	$G/c^4 = 8.263 \times 10^{-50} \text{ s}^2 \text{g}^{-1} \text{cm}^{-1}$
Specific internal energy*	$L^2 T^{-2}$	$\text{cm}^2 \text{s}^{-2}$	→	1	$1/c^2 = 1.11 \times 10^{-21} \text{ s}^2 \text{cm}^{-2}$
Velocity/sound speed	$L T^{-1}$	$\text{cm s}^{-1}$	→	1	$1/c = 3.336 \times 10^{-11} \text{ s cm}^{-1}$

Rest mass density	: $\rho$
(Total) energy density	: $e = \rho(1+\epsilon)$
Internal energy density	: $\rho\epsilon$
Specific internal energy density	: $\epsilon$

From GREAT units to  $G=c=M_\odot=1$  units

cm → 1

Conversion factor  $1/(GM_\odot/c^2) = 1/(1.477 \times 10^5 \text{ cm})$

\*Caution! Sometimes units of  $c^2$

# PARAMETER FILE

```
# --- General parameters -----
input_directory="./Data/" # Data input directory

output_directory=./output/ # Data output directory

l=2                      # Value of l (degree of the spherical harmonic decomposition)

newtonian=.true.          # Enable newtonian calculation of ggrav (.true.).
                           # Otherwise (.false.) general relativistic calculation.

cowling=.true.            # Compute with cowling approximation (.true.) or in dynamical
                           # space-time (.false.)

cal_dphi=.false.          # Compute perturbations of the conformal factor (delta psi)
```

See e.g. great/parameters\_test

$l=0$  : radial modes  
 $l=2$  : dominant GW channel

Newtonian=.true.  
Need to provide  $\alpha = 1 + U$

cowling=.true.  
→ no metric perturbations

cowling=.true. & cal\_dphi=.false.  
→ only  $\delta\alpha$

cowling=.true. & cal\_dphi=.true.  
→  $\delta\alpha$  and  $\delta\psi$

# PARAMETER FILE

mode=1

```
# Mode approximation  
# 1: All modes  
# 2: Compute p-mode approximation  
# 3: Compute g-mode approximation
```

bc\_type=1

```
# Outer boundary conditions  
# 1: Shock boundary condition ( $\eta_r=0$ ). This is consistent  
# with the conditions at the shock, if the background  
# accretion shock is neglected (Torres-Forne et al 2018,2019).  
# 2: Vacuum boundary conditions ( $\Delta P=0$ ). This is the  
# standard boundary conditions used for neutron stars  
# surrounded by vacuum. If used for proto-neutron stars  
# it will compute wrongly most of the p-modes.
```

See e.g. great/parameters\_test

p-mode approximation:  $N^2 = 0$

g-mode approximation:  $1 / c_s^2 = 0$

# PARAMETER FILE

See e.g. great/parameters\_test

```
ggrav_mode=2      # Select the ggrav (G) computation formula  
#   1: ggrav = -dalpha_dr  
#   2: ggrav = dp_dr/(rho h)  
#   3: Mean value of 1 and 2
```

```
use_precomputed_n2=.false. # .true.: Do not compute N^2 and use pre-computed  
#           values of N^2. Do not set to .true.  
#           unless you know what you are doing.
```

```
verb=.false.       # Verbose mode (for debugging)
```

$$\frac{1}{\rho h} \partial_i P = -\partial_i \ln \alpha \equiv \mathcal{G}_i$$

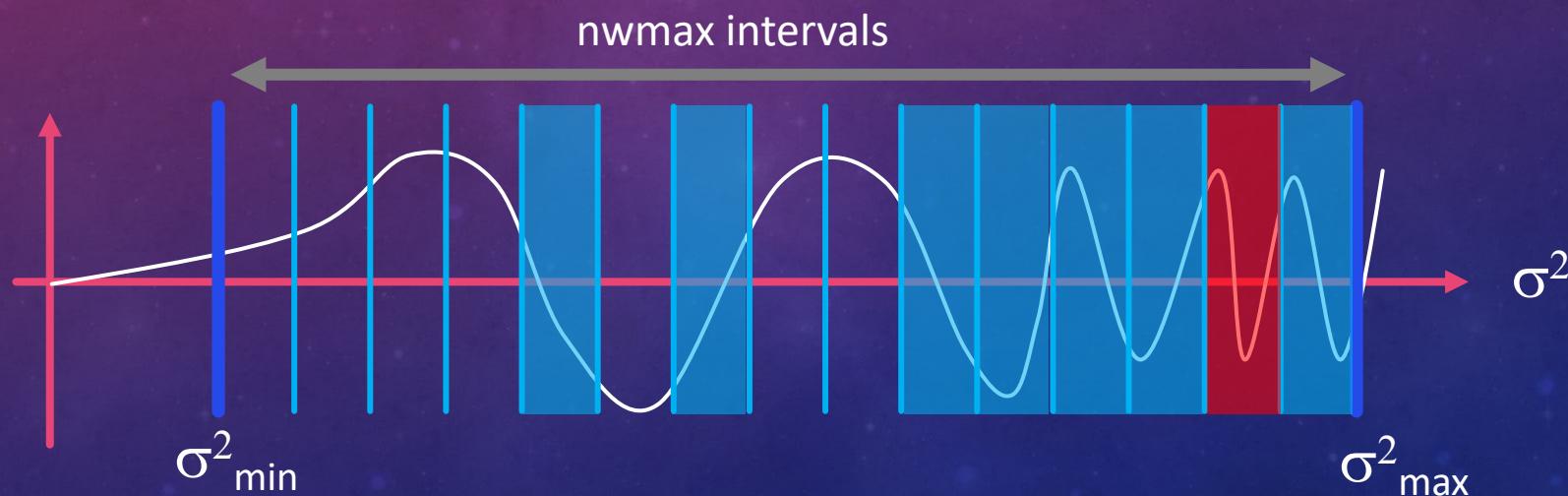
# PARAMETER FILE

See e.g. great/parameters\_test

# --- Parameters for the coarse eigenmode search ---

```
fmin=1.0d1          # Minimum value of frequency for the eigenvalue search
fmax=1000.0d0        # Maximum value of frequency for the eigenvalue search
funits=2             # Units for the frequency
                    # 0: frequency code units (G=C=1 + cm), i.e. cm^-1
                    # 1: frequency Hz
                    # 2: angular frequency in code units (cm^-1)
nwmax = 1000         # Number of frequencies for the coarse eigenfrequency search
logf_mode = .true.    # .true.: Coarse search equally spaced in logarithm of f
                    # .false.: equally spaced in f
```

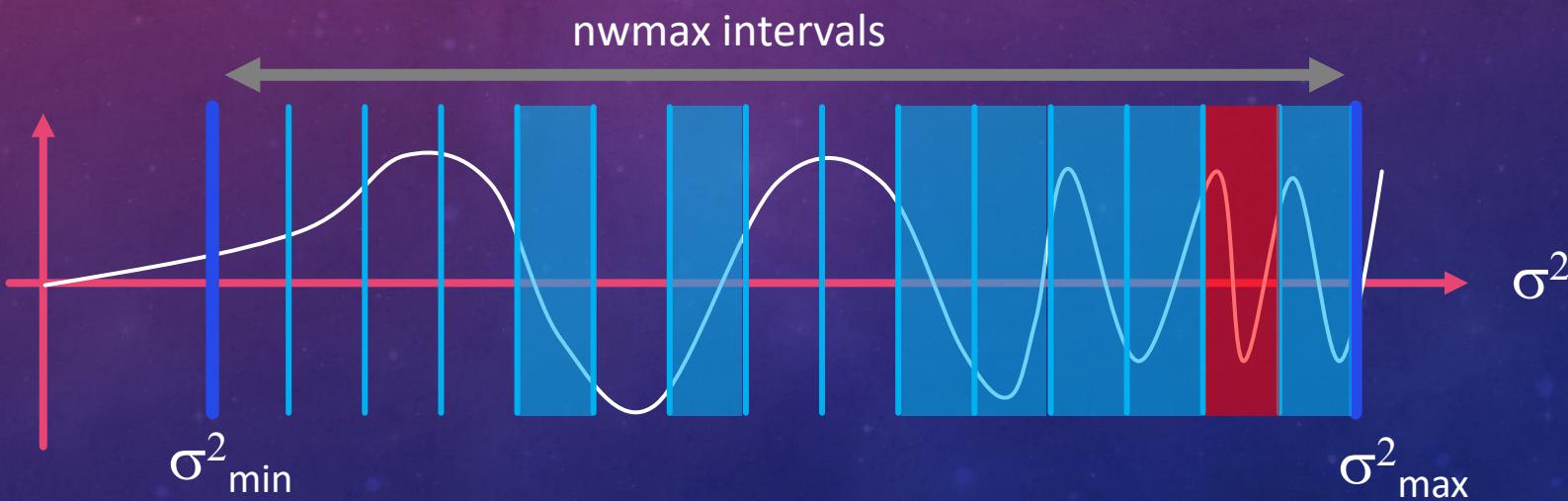
Sets the units of the code  
0:  $\sigma^2 = (2 \pi <\text{input}>)^2$   
1:  $\sigma^2 = (2 \pi <\text{input}> / c)^2$   
2:  $\sigma^2 = <\text{input}>^2$



# PARAMETER FILE

```
w2_sign = 1      # Sign of omega^2  
#   +1 : search for positive values of omega^2 (real eigenvalues)  
#   -1 : search for negative values of omega^2 (purely imaginary eigenvalues)
```

See e.g. great/parameters\_test



# PARAMETER FILE

```
# --- Parameters for the fine eigenmode search -----
tol = 1.0d-3          # Tolerance for eigenvalue calculation

# --- Parameters for the eigenmode integration -----
maxiter = 50          # Maximum number of iterations in the shooting method to match
                      # the outer boundary conditions

integration_mode=1     # Select how to integrate equations (only one option available)
                      #   1: Explicit integration for all equations (6x6)

ip_coeffs=4            # Regularize first ip_coeff points

# --- Parameters for output -----
output_directory=".//output/"

# Parameters not used in the module
nt_ini=0
nt_last=100000
nt_step=1
input_mode=3
interpolate_grid=false.
int_factor=4
input_directory=
outer_boundary_mode=1
rho_thr=0.
```

See e.g. great/parameters\_test

Don't touch this

Auxiliary parameters

We are not using them

GREAT!



**Now what?**

# The GREAT hackathon

# GROUPS

Form groups of three

At least one fortran expert and one python expert

Register groups here: <https://go.uv.es/spgD3ym>

In all cases: Use Newtonian=.true., cowling=false.

# Challenges

- ▶ A Compute the radial ( $l=0$ ) oscillations of a series os stars
  - ▶ Compute the eigefrequencies of the models in /Data/stars
  - ▶ All models in GREAT units and readable by src/test\_file.f90
  - ▶ BONUS: what is each star?
- ▶ B Write a python routine to compute the number of nodes of  $\eta_r$
- ▶ C Compute the oscillation frequencies of a TOV star with polytropic EOS
  - ▶ 1.4 Msun ( $K=100, \Gamma=2$ )
  - ▶ Use Aussois2023/tov\_example.ipynb, Compose ...
  - ▶  $l=2$  modes
  - ▶ BONUS: Compute how the frequency changes for TOV models with increasing density
  - ▶ BONUS+: color the points according to the number of nodes

In all cases: Use Newtonian=.true., cowling=false.

# Challenges

- ▶ D Compute the oscillation frequencies of a TOV star using Compose cold EOS
  - ▶ Take your favourite EOS
  - ▶ 1.4 Msun
  - ▶ l=2 modes
  - ▶ BONUS: Compute how the frequency changes for TOV models with increasing density
  - ▶ BONUS+: color the points according to the number of nodes
  
- ▶ E Same as D but with a finite temperature EOS from Compose
  - ▶ Take your favourite EOS
  - ▶ Set constant entropy or temperature
  - ▶ 1.4 Mcun
  - ▶ l=2 modes
  - ▶ BONUS: Compute how the frequency changes for TOV models with increasing density
  - ▶ BONUS+: color the points according to the number of nodes

In all cases: Use Newtonian=.true., cowling=false.

# Challenges

- ▶ F Create an animation of the Lagrangian displacement showing the deformation of the star
  - ▶ BONUS: Nicest animation
- ▶ G Modify the exercise 2 to have a negative Brunt-Väisälä frequency and compute modes
- ▶ EXTRA BONUS (except for F)
  - ▶ Each completed challenge
  - ▶ First group to finish
  - ▶ Second group to finish

500

+1500

1000

3 chocolates\*

x 2

x 1.5

\*Until we run out of chocolates

**WINNER**

**CLOSING TIME: 16:55**

**WINNER: Largest amount of points at closing time**