# postvecp2python

It is a python program that post-processes the outputs of postvecp. It requires python **3.x** to run. It allows to display **meridional cuts** of a mode's kinetic energy, viscous dissipation, power in the differential rotation, thermal energy and thermal dissipation. Paths of **characteristics** as well as turning surfaces and critical latitudes can be overlaid. The mode's **spectral content** can also be displayed. Other features (eigenfrequencies for QZ runs, total dissipation for forced runs) are also available.

The program has been written in March 2022, so it may still have bugs — please do let me know if you find some! Feel free to adapt the program to your needs and to **contribute** to the code's development.

The program requires a parameter file named **paramsp2p.dat**. The file contains a list of parameters, their value(s), followed by comments starting by a # symbol. The following slides are meant to illustrate what can be done with the program. Enjoy!

# the paramsp2p.dat parameter file

```
########################################
### parameter file for postvecp2python ###
########################################

# local directory/ies of lsb simulations, can be:
# out01 if only directory 'out01' is to be considered
# out01,out02,out03 if you want to display results for the three mentioned directories
# all if you want to display results for all local directories starting as 'out*'
directory       outgim                  # see list of examples above
movie           No                      # if more than one directory is specified, do we animate the z-cuts?

### roadmap:
plot_spectrum         No          # display modes spectral content
plot_qz_eigenfq       No          # display eigenfrequencies for a QZ run
plot_total_dissipation   No       # display shell-integrated dissipation vs. frequency for various directories
plot_zcut             Yes         # display meridional (z-) cut of modes
field                 ek          # can be so far: ek (kinetic energy), dissv (viscous dissipation), shear (power in differential rotation),
                                  # et (thermal energy) or disst (thermal dissipation)
onemode               Yes         # display above quantities for only first computed mode?
normalizetomax        Yes         # is displayed field normalized to its maximum value?
fieldmin                          # if specified, minimum field value in colorbar
fieldmax                          # if specified, maximum field value in colorbar

### characteristics:
plot_caract           No          # overplot caracteristics on top of meridional cut?
caract_s_z            0.4,0.4     # if so, enter s and z coordinates of initial point separated by a comma
last_only             No          # only display last points along characteristics (~attractor)
eq_file_michel        Yes         # did equation file use Michel's normalization units?

### output:
saveaspdf             Yes         # save image as pdf file?
saveaspng             No          # save image as png file?

### display options:
mycolormap     nipy_spectral      # nipy_spectral, inferno, magma…
multbyaxisdist                No  # multiply field by distance to rotation axis (to enhance contrast)
plot_critical_latitudes       No  # display critical latitudes at inner and outer radial edges (solid or shellular rotation only)
plot_turning_surfaces         No  # display turning surfaces if any
plot_critical_layers          No  # display critical layers, if any
verbose                       Yes # display control stuff
```

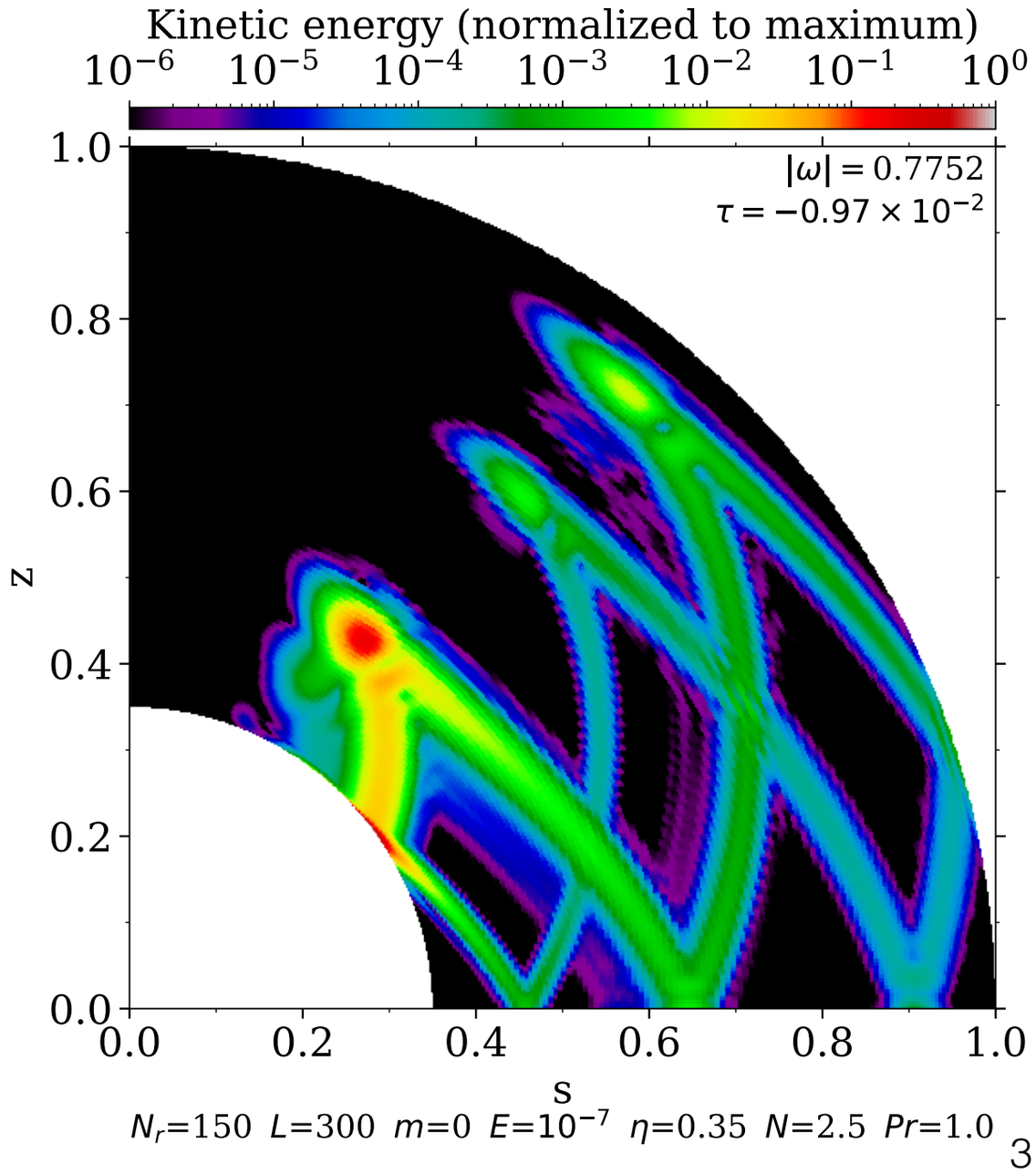
# First use

Let's assume you have run nvspeigen and postvecp in the directory 'outgim' created by the perl script onerun_gim.pl that is located in ~/lsb/src/perl_launch_scripts. This script is for the calculation of a gravito-inertial mode. You now need to have the paramsp2p.dat file located in the directory just above 'outgim'. To **execute** postvecp2python, simply type:

> python3 ~/XXX/postvecp2python.py

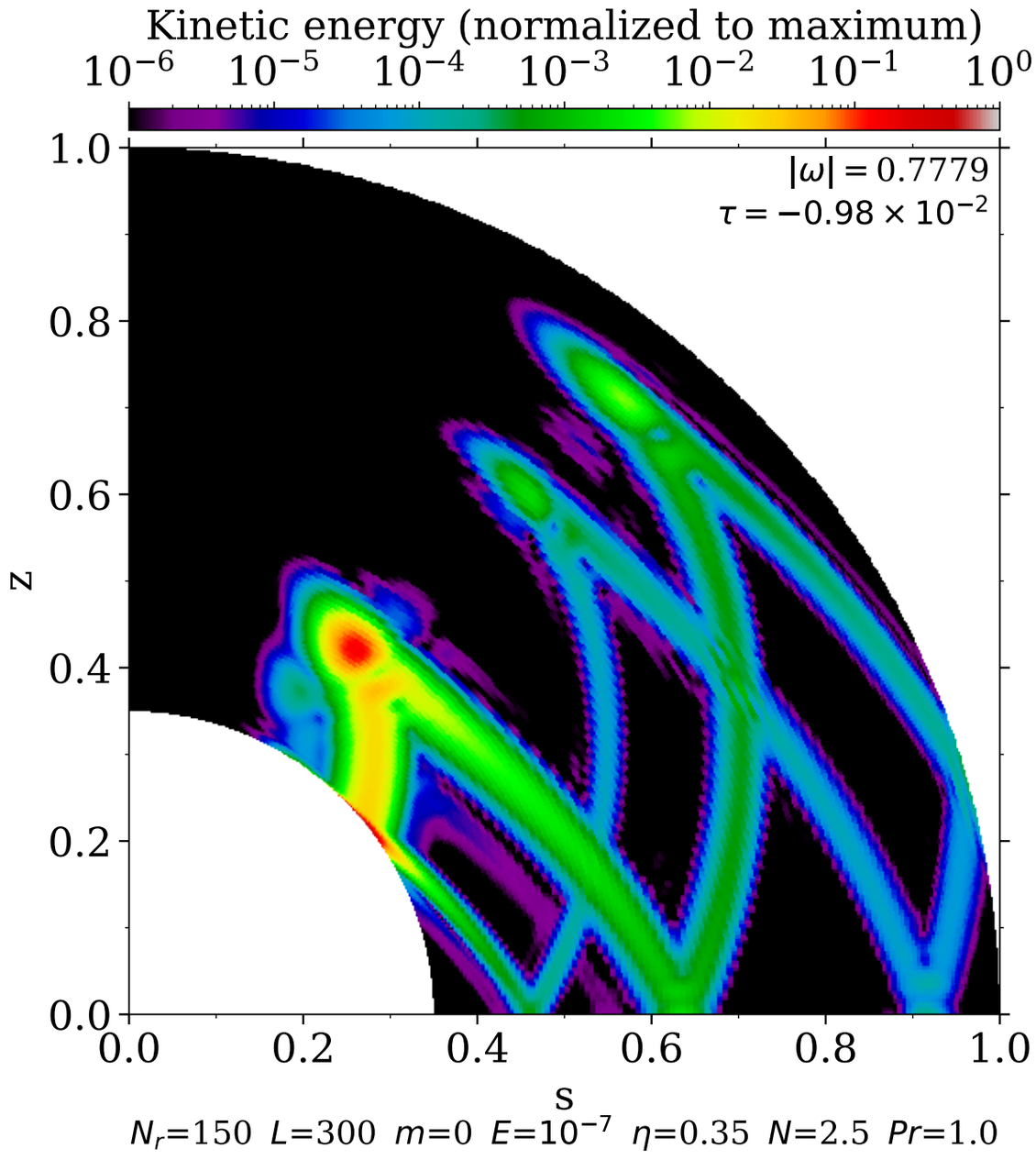
where XXX is the path of the directory where postvecp2python.py and the other python files of postvecp2python are located.

This will create a file called 'ek_mode0_outgim.pdf', which is displayed on the right-hand side. It shows a meridional cut of the mode's kinetic energy (more precisely, for the least-damped mode since 2 modes are computed in this example run).



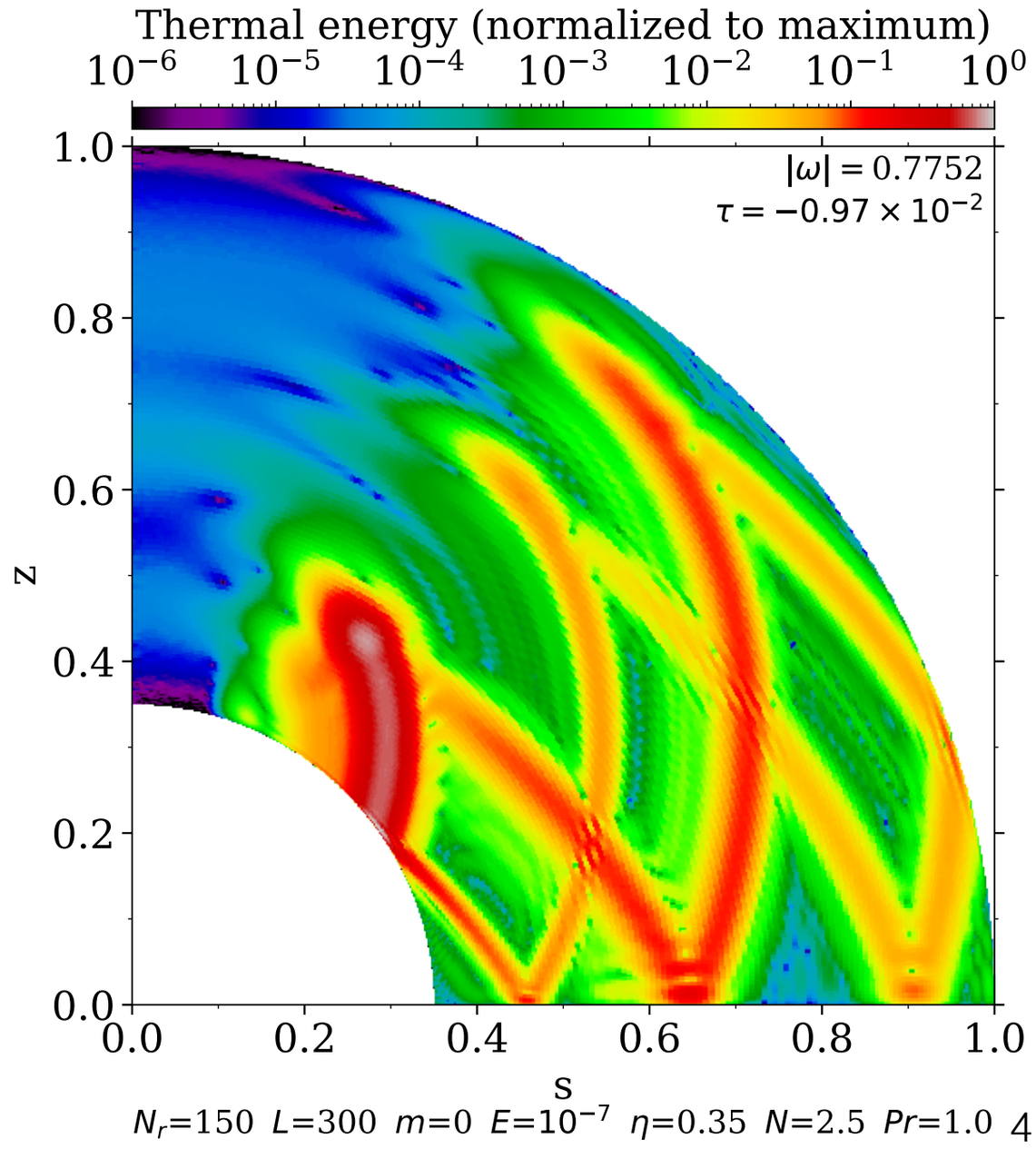
# Meridional cuts: exploring parameters

If you want to display the kinetic energy for the other computed modes, set **onemode** to No. In our example, 'ek_mode1_outgim.pdf' will also be written, which is shown below:




$N_r$=150 $L$=300 $m$=0 $E$=$10^{-7}$ $\eta$=0.35 $N$=2.5 $Pr$=1.0

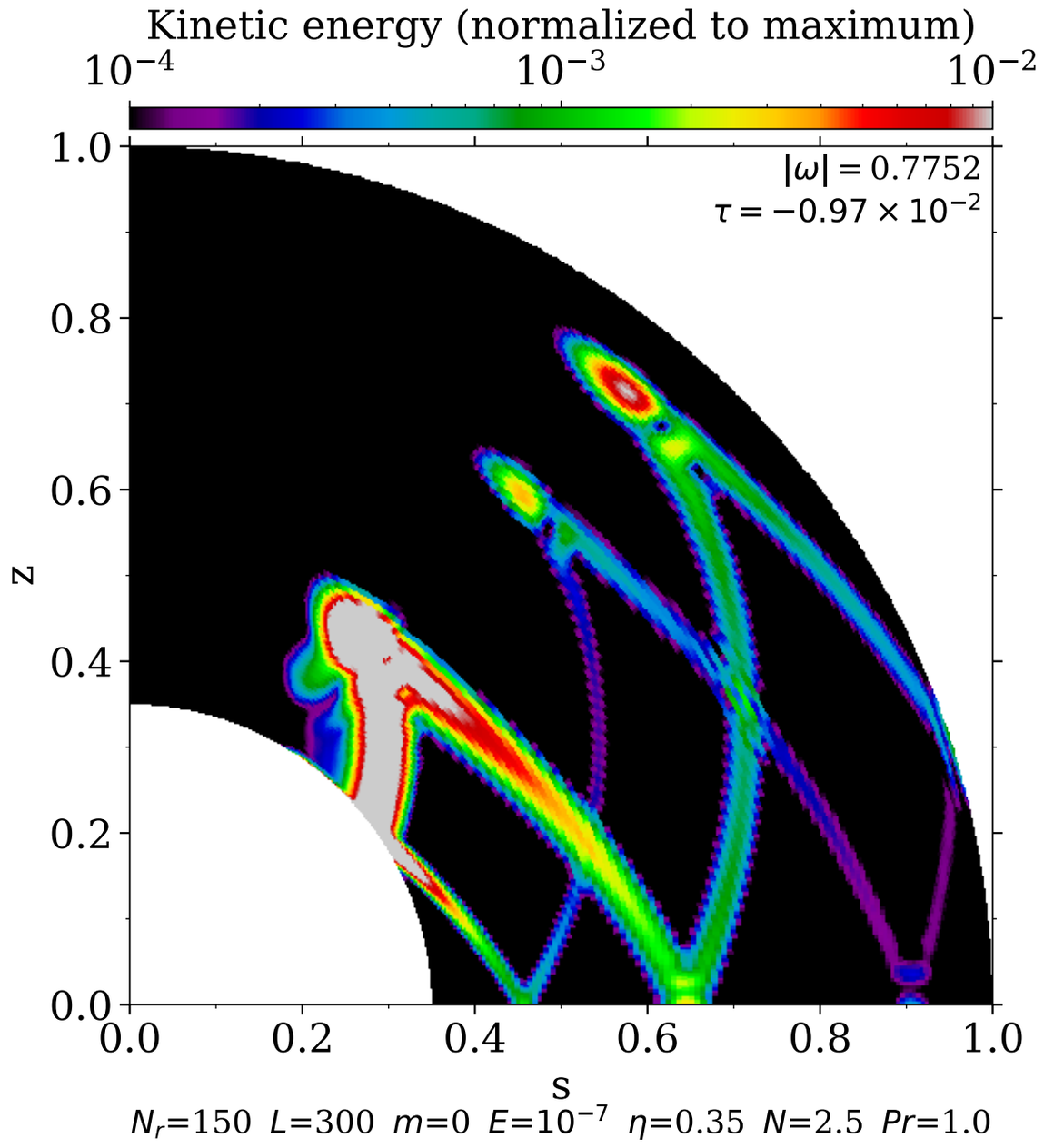
You may change the field to be displayed as a meridional cut. For instance, if you'd like to display the mode's thermal energy, set **field** to et:



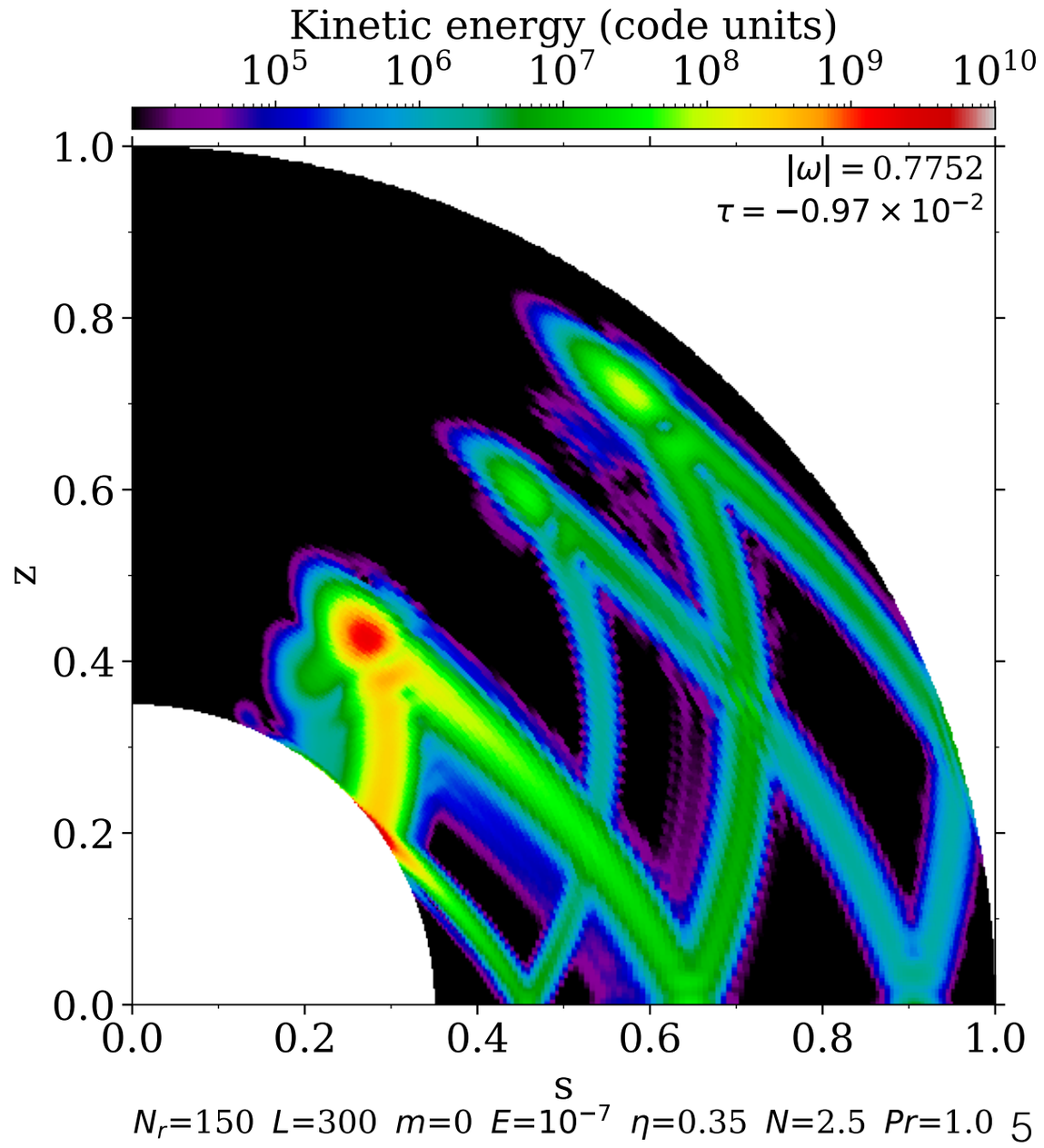

$N_r$=150 $L$=300 $m$=0 $E$=$10^{-7}$ $\eta$=0.35 $N$=2.5 $Pr$=1.0

# Meridional cuts: exploring parameters

You can also change the minimum and maximum values on the colorbar via **fieldmin** and **fieldmax**. For instance:

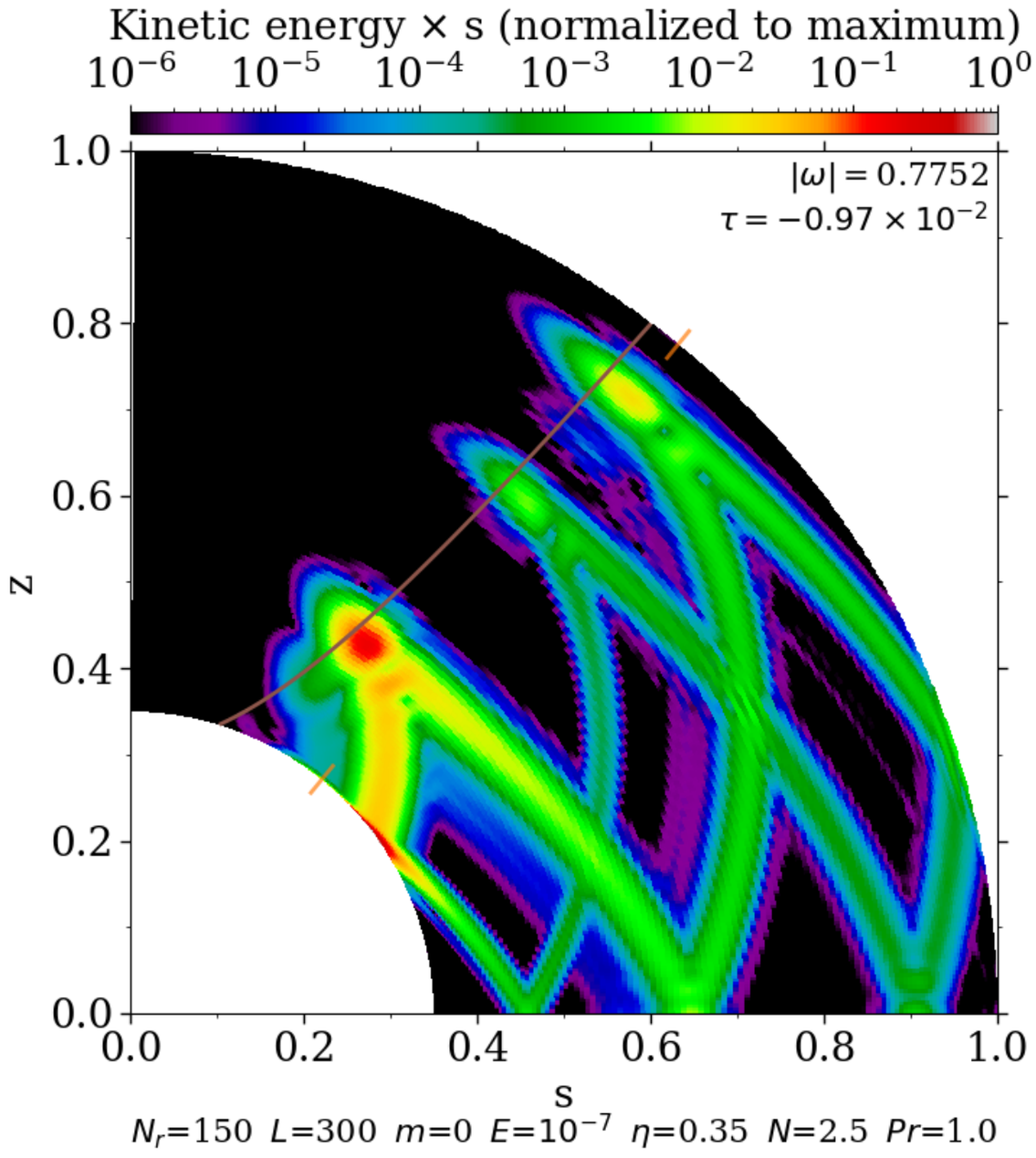



Note that you can also display the field without renormalizing it to its maximum value, by setting **normalizetomax** to No:

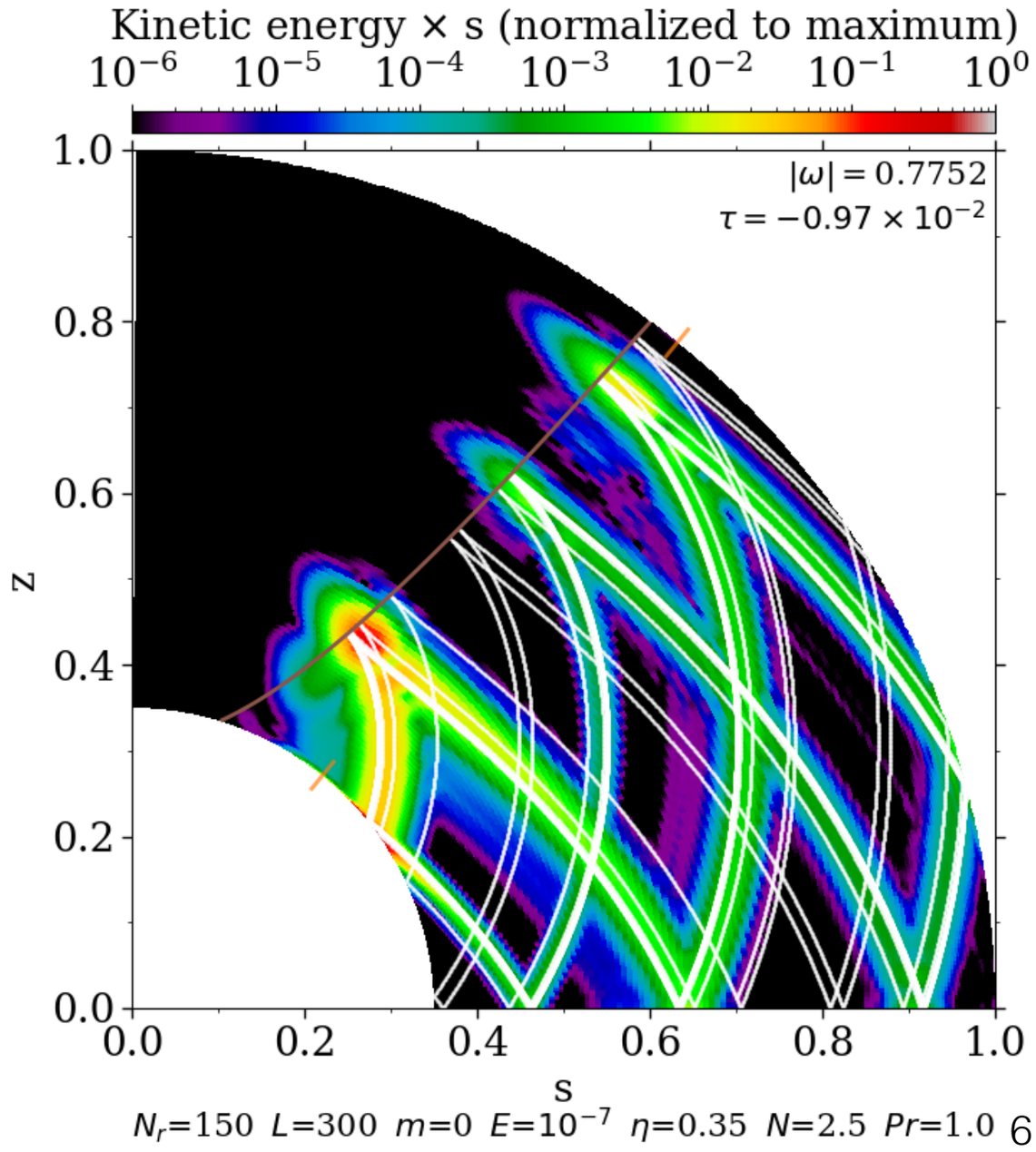


# Meridional cuts: exploring parameters

If you'd like to overplot the location of the critical latitudes (as orange ticks) and of the turning surfaces (brown curve), activate the corresponding options:

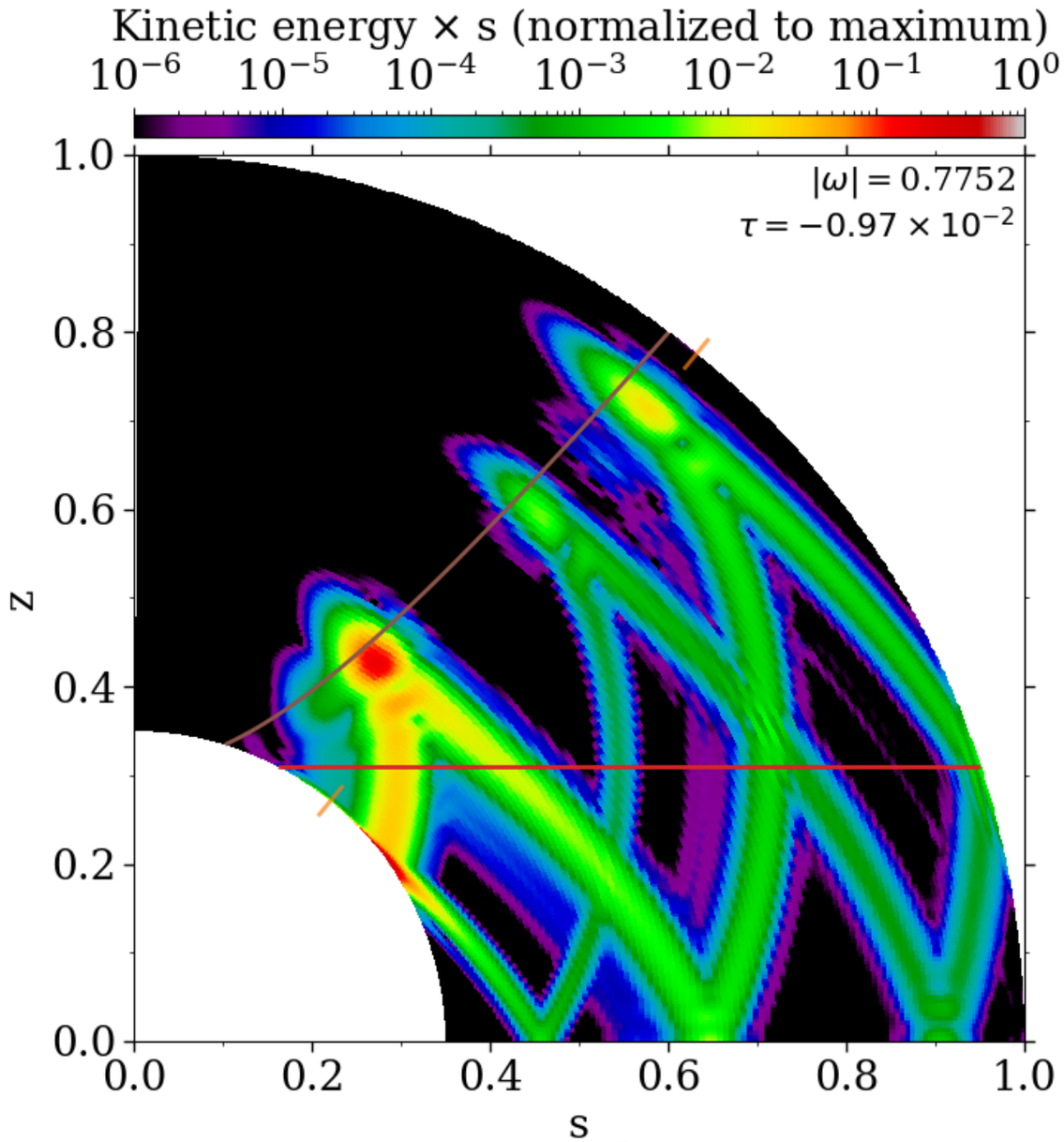


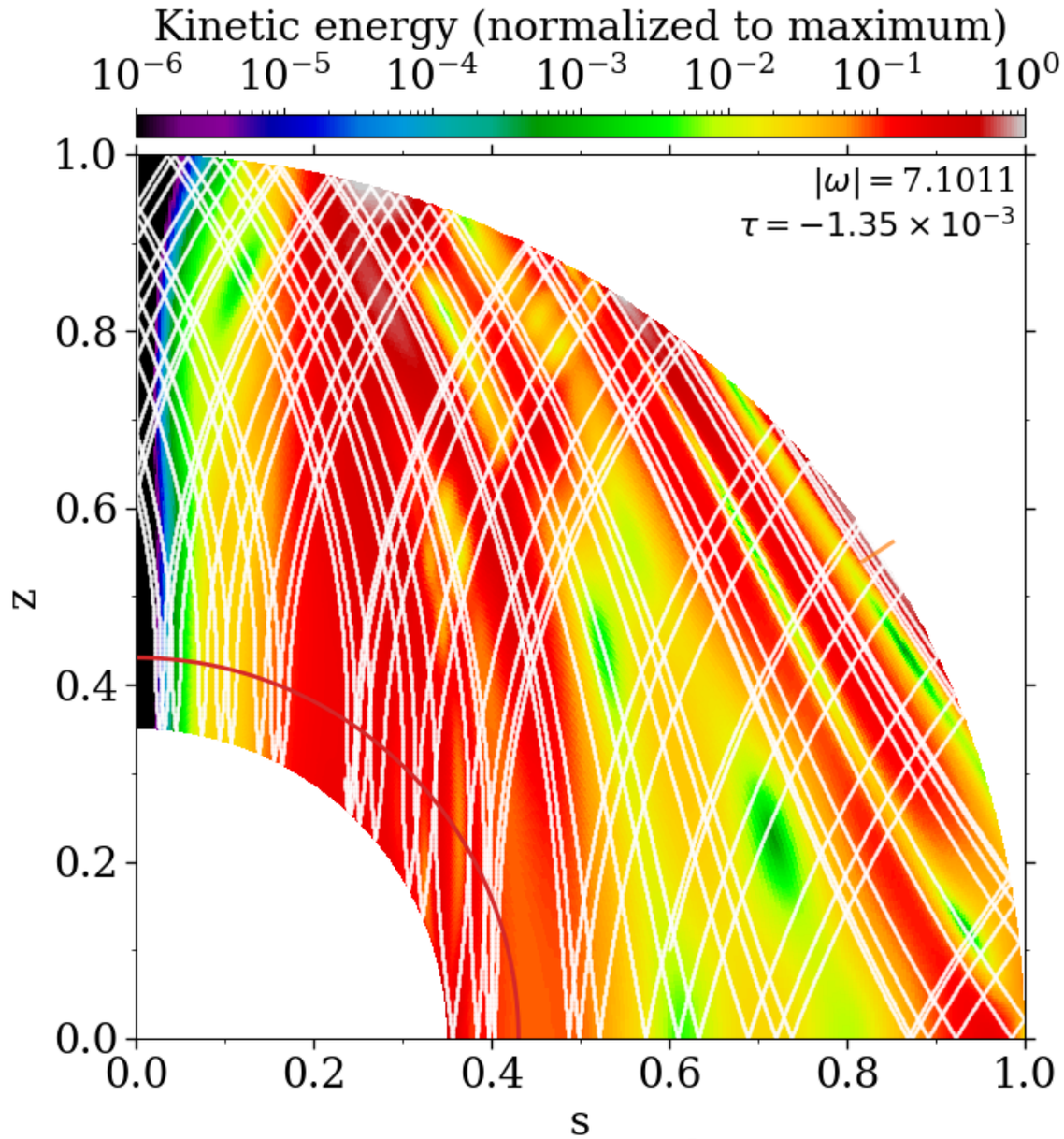

Last, but not least, you can overplot paths of characteristics as white curves by setting **plot_caract** to Yes and by specifying their initial position in the shell via **caract_s_z**:



# Meridional cuts: exploring parameters

If you'd like to overplot the location of critical layers, set **plot_critical_layers** to Yes. They will be shown by red solid curves. They are defined so far as the location where the slope of the paths of characteristics dz/ds cancels out, which is the case when the Doppler-shifted frequency ώ cancels out for pure inertial modes, and when ώ=Nz for gravito-inertial modes.



# Meridional cuts: exploring parameters

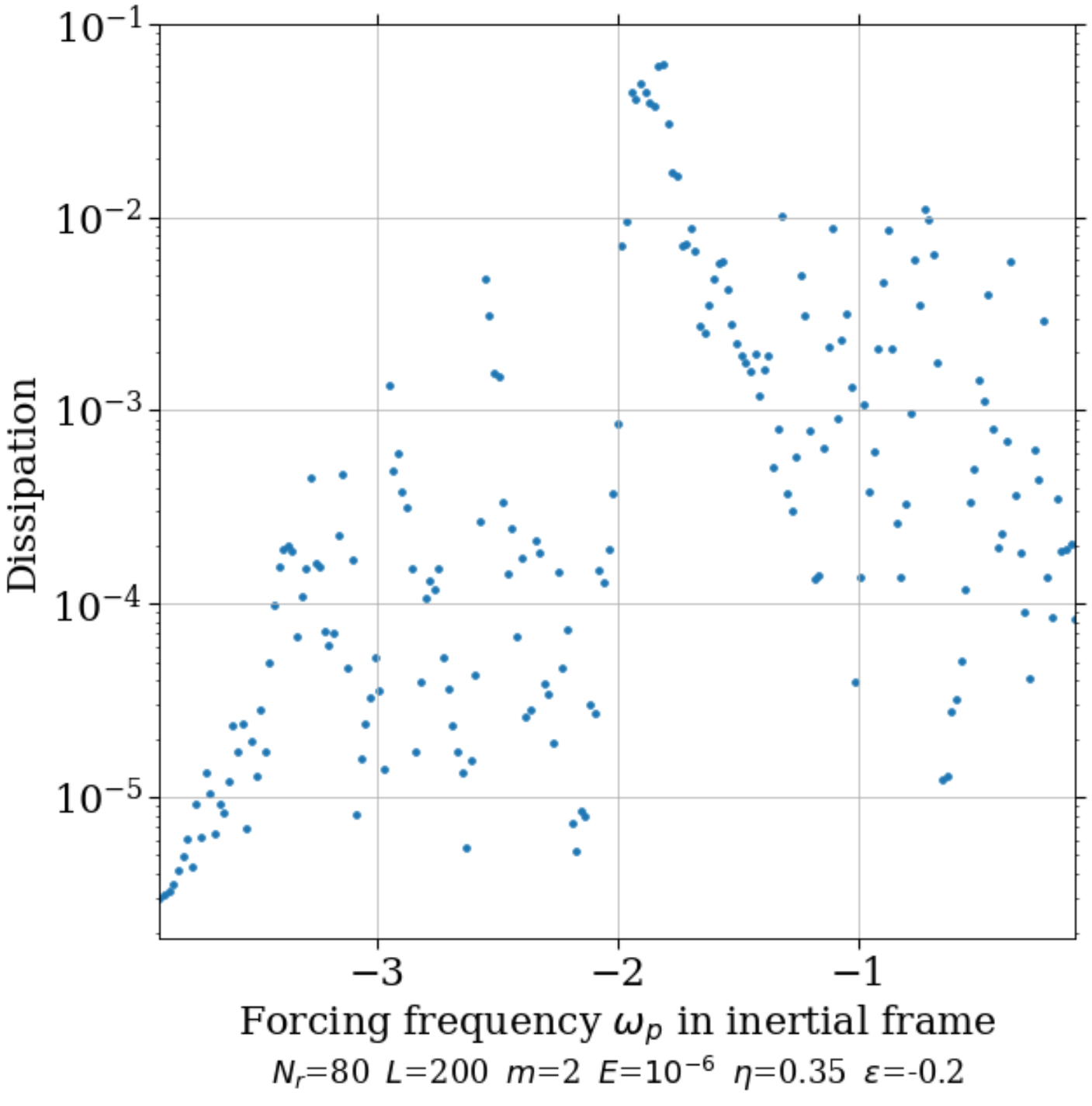
Note that you can make an **animation** of meridional cuts for all the sub-directories which are located in your current working directory. To do this, set **directory** to all and **movie** to Yes. You need to have the **ffmpeg** library for python.

As an example, run the perl script looprunwm_cyl_forced.pl. This will create 201 directories with all necessary files, and which only differ by the forcing frequency ‘gamma’. By running the ‘script’ executable (by typing ./script), you will automatically enter each sub-directory out000 to out200 successively, execute nvspeigen and postvecp.

Running postvecp2python with **directory** set to all and **movie** to Yes will make the mp4 animation that you will find at this link!

# Total dissipation vs. forcing frequency

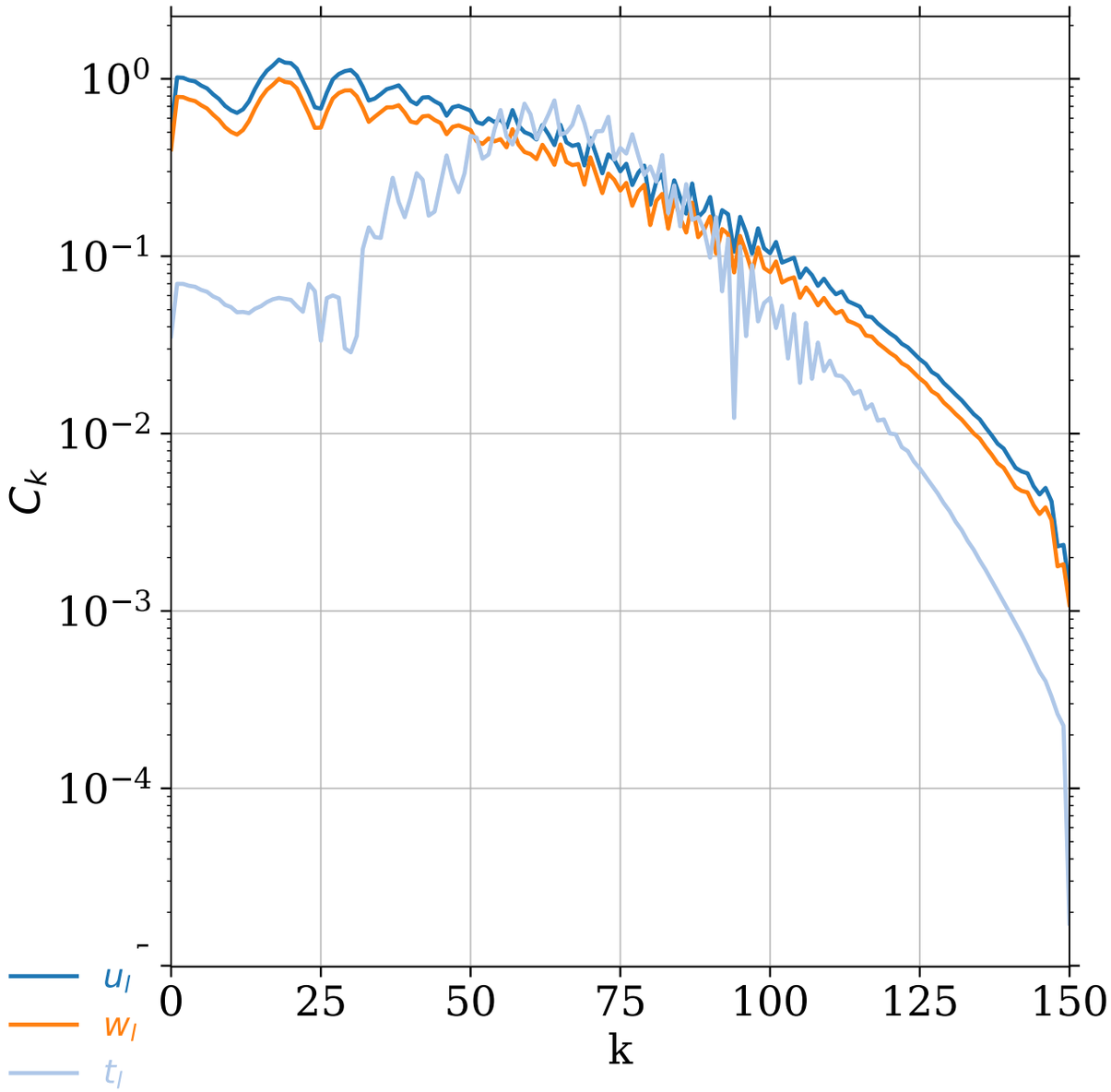
For this particular set of runs, you can display the viscous dissipation as a function of the forcing frequency by setting **plot_total_dissipation** to Yes:

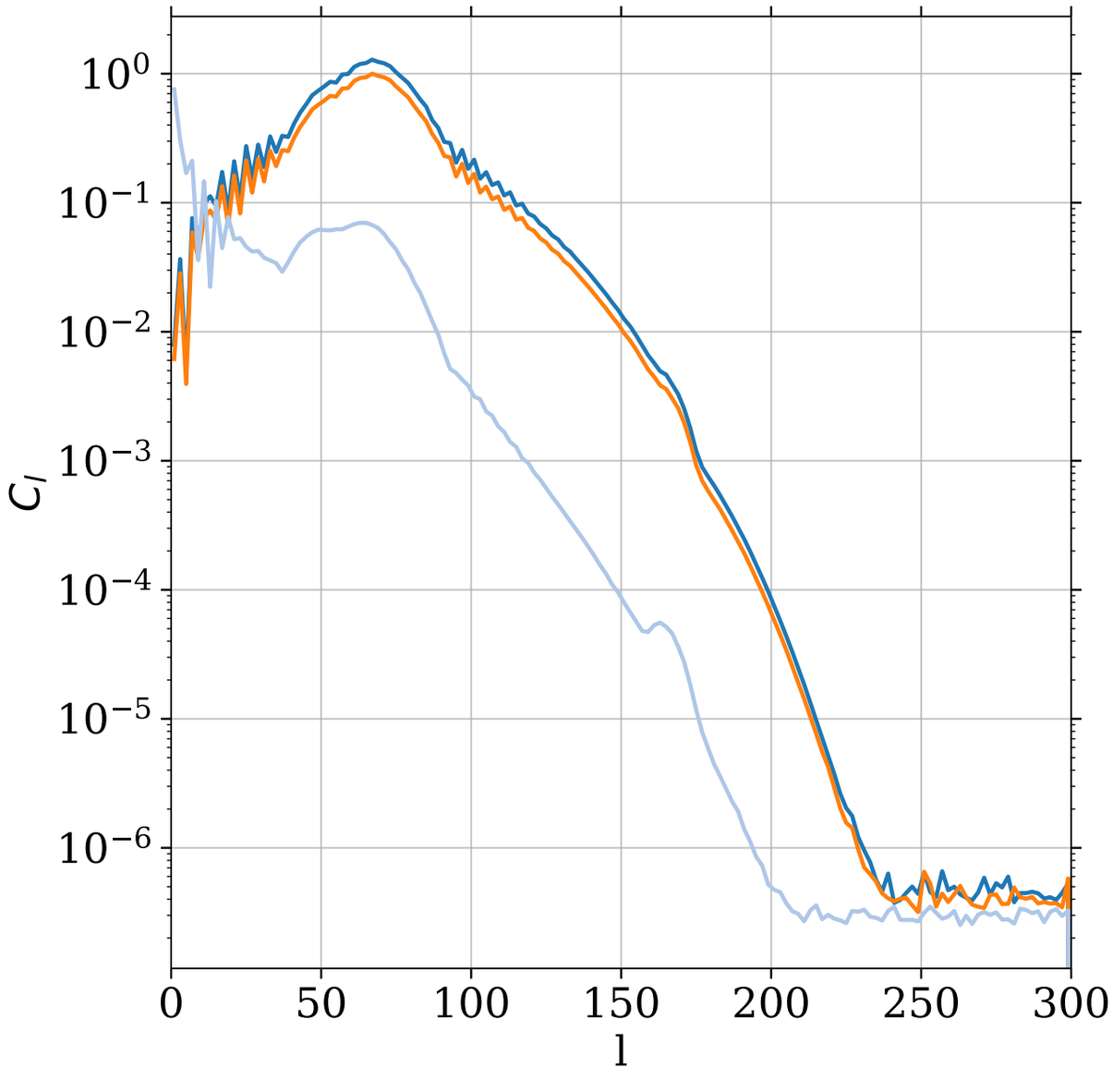



NB: if you'd prefer to display the power of the **pressure**-related force or the power of the **differential-rotation**-related term, simply edit plot_total_dissipation.py circa lines 35-40 and select the corresponding value for the Y-field

# Modes' spectral content

To display the spectral content of the modes, just set **plot_spectrum** to Yes:

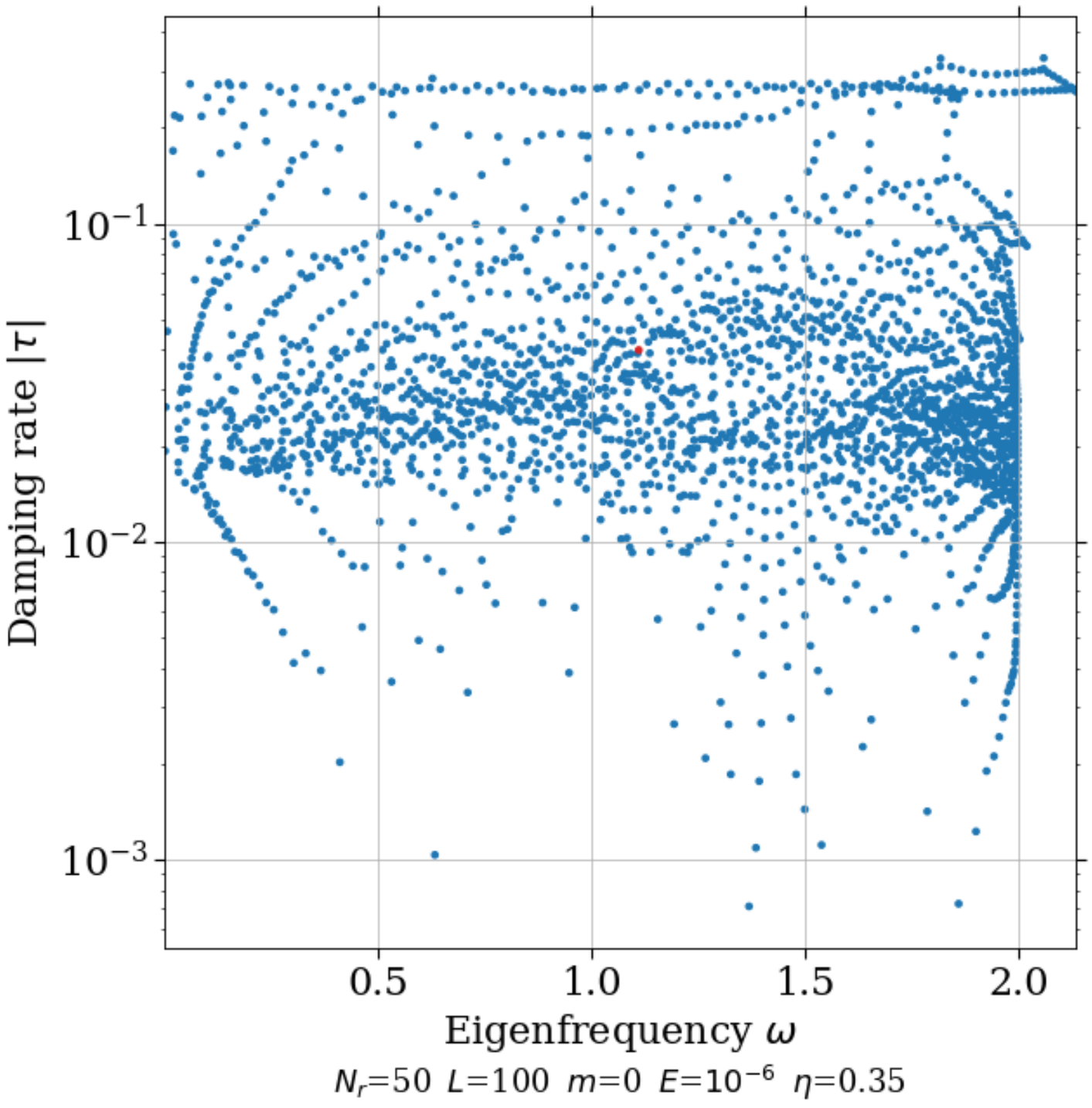



$N_r$=150 $L$=300 $m$=0 $E$=$10^{-7}$ $\eta$=0.35 $N$=2.5 $Pr$=1.0

# Eigenfrequencies for QZ runs

If you carry out a QZ calculation to get an overview of the eigenmodes, you can display the modes eigenfrequencies by setting **plot_qz_eigenfq** to Yes:




NB: modes with a negative damping rate are shown in blue, those with a positive damping rate (=growth rate) are shown in red (in the case displayed here, the red mode is fictitious!)

# Gallery of other calculations