Megyr Documentation

Christopher Wells

Nov 06, 2019

Contents:

CHAPTER 1

Getting started

This tutorial will walk you through running Megyr on a grid of parameters for MESA and GYRE. It assumes that you have some familiarity with how to configure and run MESA and GYRE.

1.1 Pre-requisites

Before starting this tutorial, you should make sure to install `pipenv` and `pyenv`. These tool are not required to use Megyr, but they make using it a lot easier since they can make sure that you are using the correct Python version.

You will also want to make sure that you have `MESA` and `GYRE` installed and working.

1.2 Setting up a project

To get started, we will first need to create a new project directory to work in. To run MESA we will need some of the files that the MESA project template has, so we can create a copy of that by running the following commands in the terminal.

```
cp -r $MESA_DIR/star/work megyr-tutorial
cd megyr-tutorial
```

Now that we have a directory we can work in, we'll want to setup a `pipenv` environment with a Python 3 installation and install `megyr`.

```
pipenv --python 3.6
pipenv install megyr
```

Next we will want to create the Python script for using Megyr. To do this, we can create a file called `megyr_config.py` with the following code.

```
import megyr

def main():
    megyr.run({
        "input": {
            "mesa_configs": ["inlist.mustache"]
        },
        "output": {
            "mesa_profile_summary_file": "mesa_profile_attributes.csv"
        },
        "stages": {
            "mesa_params": {
                "initial_mass": [1, 1.1, 1.5]
            }
        }
    })

if __name__ == "__main__":
    main()
```

For now we will just be running MESA against 3 models, with initial masses of `1`, `1.1`, and `1,5` M_sun. We will be using one inlist config file for MESA, and output summaries of the profile files that MESA generates.

1.3 MESA

1.3.1 Setup

Now that we have our script to run MESA, we will need to get the executable to use to run MESA and the configuration file to run MESA on.

Since we created our project from the MESA project template, we can run the following command to compile the MESA `star` executable.

```
./mk
```

Next we will want to setup the MESA inlist configuration file to use. By default the MESA project template gives us three inlist files, but for now we will remove them to replace them with a simpler one.

```
rm inlist inlist_pgstar inlist_project
```

Now we will create the inlist file we will use. Megyr allows you to provide uses mustache templates for configuration files which it will fill in the parameter values of at runtime. So we can create a file called `inlist.mustache` with the following contents.

```
&star_job

  ! begin with a pre-main sequence model
    create_pre_main_sequence_model = .true.

/ !end of star_job namelist


&controls

```

(continues on next page)

(continued from previous page)

```
  ! starting specifications
    initial_mass = {{initial_mass}} ! in Msun units

  ! stop at 2 Gyr
    max_age = 2000000000

/ ! end of controls namelist
```

You can see that for the `initial_mass` we are setting it to `{{initial_mass}}`. Since this is a mustache template, Megyr will fill in any instance of `{{some_variable}}` with the value of the name that is inside of the two curly braces.

1.3.2 Running

We can then execute our Megyr script by running the following command in the terminal.

```
pipenv run python megyr_config.py
```

You should then see output appear in the terminal from the MESA runs. Wait for the MESA runs for all three models to complete. You can then take a look at the files that MESA and Megyr outputted by running the following commands.

```
ls out
ls out/mesa_initial_mass_1__
less out/mesa_initial_mass_1__/mesa_profile_attributes.csv
```

The `mesa_profile_attributes.csv` files that we had Megyr create for each model contain summary information from all of the profile files that MESA outputted.

Now, if you try to rerun the Megyr script you will see one of the nice features of Megyr.

```
pipenv run python megyr_config.py
```

You will notice that the MESA runs are not repeated, since Megyr notices that they have already been run. Megyr will keep the results from MESA and GYRE runs, and for any run that completed, Megyr will not rerun it and will instead work on the next task that has not yet been completed.

1.4 GYRE

1.4.1 Setup

In order to get Megyr to run GYRE on the different MESA models we will need to make some changes to our `inlist.mustache` and `megyr_config.py` files as well as add a GYRE config file template.

First we will need to modify our `inlist.mustache` file to have MESA output pulse files for GYRE to process. To do this we will need to add the following lines to the `controls` section of the config file template.

```
! Output pulse files for GYRE
pulse_data_format = 'GYRE'
write_pulse_data_with_profile = .true.
add_center_point_to_pulse_data = .TRUE.
add_double_points_to_pulse_data = .TRUE.
```

Next we will need to create the GYRE config file template. So we'll want to create a file called `gyre.in.mustache` with the following contents.

```
&constants
/

&model
    model_type = 'EVOL'  ! Obtain stellar structure from an evolutionary model
    file = '../LOGS/profile{{profile}}.data.GYRE'    ! File name of the evolutionary␣
↪model
    file_format = 'MESA' ! File format of the evolutionary model
/

&mode
    l = {{l}}                ! Harmonic degree
/

&osc
    outer_bound = 'JCD'
    variables_set = 'JCD'
    inertia_norm = 'BOTH'
/

&num
    diff_scheme = 'COLLOC_GL4' ! 4th-order collocation scheme for difference equations
/

&scan
    grid_type = 'LINEAR'     ! Scan for modes using a uniform-in-frequency grid
    freq_min = 10            ! Minimum frequency to scan from
    freq_min_units = 'UHZ'   ! Units for freq_min
    freq_max = 500           ! Maximum frequency to scan to
    freq_max_units = 'UHZ'   ! Units for freq_max

    n_freq = 1000          ! Number of frequency points in scan
/

&grid
    alpha_osc = 10  ! Ensure at least 10 points per wavelength in propagation regions
    alpha_exp = 2   ! Ensure at least 2 points per scale length in evanescent regions
    n_inner = 5     ! Ensure at least 5 points between center and inner turning point
/


&ad_output
    summary_file = '{{ad_output_summary_file}}'    ! File name for summary file
    summary_file_format = 'TXT'                    ! Format of summary file
    summary_item_list = 'M_star,R_star,l,m,n_p,n_g,n_pg,omega,freq,freq_units,E_norm'␣
↪! Items to appear in summary file
    freq_units = 'UHZ'                             ! Units for freq column
/

&nad_output
/
```

This GYRE config file template will let us specify a grid of `profile` and `l` values to run for each MESA model. It is also set to output summary files of the adiabatic oscillation calculations to `{{ad_output_summary_file}}`, which will allow Megyr to be set to aggregate this data into a single csv file.

Next we will need to modify our `megyr_config.py` script to enable Megyr to run GYRE using the config file template we created.

First we will want to add the following setting to the `output` section of the config dict. This will set Megyr to aggregate the adiabatic oscillation summary files that GYRE outputs. This will make it a lot easier to work with the oscillation data.

```
"gyre_oscillations_ad_summary_file": "oscillations_ad.csv"
```

Next we will need to set Megyr to use the GYRE config file template that we created. We can do that by adding the following line to the `input` section of the config dict.

```
"gyre_config": "gyre.in.mustache"
```

Then we will want to add the following function in order to tell Megyr what parameter values to use for the GYRE parameter grid. Note that we can reference the MESA parameter combination and profile file data in order to set the GYRE paramters based on these values.

```
def calc_gyre_params(mesa_params, mesa_data):
    return {
        "l": [0, 1, 2],

        # Look at all the profiles that are at least 0.0001 Gyr in age
        "profile": mesa_data[mesa_data["star_age"] > 100000]["profile"]
    }
```

Finally we will want to set Megyr to use this function to determine the GYRE parameter grid to use for each MESA model. We can do this by adding the following line to the `stages` section of the config dict.

```
"gyre_params": calc_gyre_params
```

After all of these changes, the Megyr script should look like the following.

```
import megyr

def main():
    megyr.run({
        "input": {
            "mesa_configs": ["inlist.mustache"],
            "gyre_config": "gyre.in.mustache"
        },
        "output": {
            "mesa_profile_summary_file": "mesa_profile_attributes.csv",
            "gyre_oscillations_ad_summary_file": "oscillations_ad.csv"
        },
        "stages": {
            "mesa_params": {
                "initial_mass": [1, 1.1, 1.5]
            },
            "gyre_params": calc_gyre_params
        }
    })

def calc_gyre_params(mesa_params, mesa_data):
    return {
        "l": [0, 1, 2],

        # Look at all the profiles that are at least 0.0001 Gyr in age
```

(continues on next page)

(continued from previous page)

```
        "profile": mesa_data[mesa_data["star_age"] > 100000]["profile"]
    }

if __name__ == "__main__":
    main()
```

1.4.2 Running

Now that we have made all of the changes to config files that we needed to make in order to have Megyr run GYRE, we can now move on to running Megyr.

However, since we made a change to our MESA config file, we don't want Megyr to reuse the data from the previous MESA runs since we need them to be rerun in order to generate the pulse files that GYRE takes in.

We can have Megyr start over from scratch by simply removing the output directory `out`.

```
rm -Rd out
```

Now we can simply run the Megyr script in the same way that we did before.

```
pipenv run python megyr_config.py
```

Once Megyr finishes running, we can take a look at the summary files that Megyr created for the adiabatic oscillations found in the GYRE runs by running the following commands.

```
ls out/mesa_initial_mass_1__/
less out/mesa_initial_mass_1__/oscillations_ad.csv
```

1.5 Summary

In this tutorial we created a simple Megyr project where we ran MESA and GYRE over a few models with differing parameters. The general pattern we used can be adjusted to run MESA and GYRE with a different grid of parameters.

CHAPTER 2

Configuration

Contents

2.1 Input

2.1.1 mesa_configs

`list[str]`

The MESA configuration file mustache templates for Megyr to use when running MESA.

- Examples
  - `["inlist.mustache", "inlist_project.mustache"]`

2.1.2 gyre_config

`str` - [Optional]

The GYRE configuration file mustache template to use. If this value is provided, then Megyr will run GYRE against the models outputted by MESA.

- Default
  - `None`
- Examples
  - `"gyre.in.mustache"`

Note: If you do not specify a value for `gyre_config` then Megyr will not run GYRE.

2.2 Output

2.2.1 output_dir

`str` - [Optional]

The directory that Megyr will place all of the temporary and output files and directories into. The directory should be specified as a relative path from the directory that the Megyr script is run in.

- Default
  - `out`

2.2.2 mesa_profile_summary_file

`str` or `None` - [Optional]

Tells Megyr where to output a summary of the MESA profile files for each model as a csv file. To set Megyr not to output this kind of summary file, you can set this config value to `None`.

These summary values are also used by MESA to speed up re-runs, as they allow it to lookup all of the MESA profile information from one file instead of having to aggregate together all of the outputted profile files again.

- Default
  - `mesa_profile_attributes.csv`

- Examples
  - `mesa_profiles.csv`
  - `None`

2.2.3 gyre_oscillations_ad_summary_file

`str` - [Optional]

Tells Megyr to output a summary of the adiabatic oscillation summary files as a csv file. The adiabatic oscillation summary files must be set in the GYRE config template to be outputted to `{{ad_output_summary_file}}`.

For example, the GYRE config file template should have settings like the following in the `ad_output` section.

```
summary_file = '{{ad_output_summary_file}}'
summary_file_format = 'TXT'
```

- Default
  - `None`
- Examples
  - `oscillations_ad.csv`

2.3 Settings

2.3.1 mesa_star_location

`str` - [Optional]

The path to the MESA `star` executable that Megyr will use to run MESA.

- Default
  - `star`

2.3.2 gyre_location

`str` - [Optional]

The path to the GYRE execuatable that Megyr will use to run GYRE.

- Default
  - `$GYRE_DIR/bin/gyre`

2.3.3 mesa_mp_threads

`int` - [Optional]

The number of Open MP threads to have MESA use.

- Default
  - Will use the number of threads set in `$OMP_NUM_THREADS`.

- Examples
  - `4`

2.3.4 gyre_mp_threads

`int` - [Optional]

The number of Open MP threads to have GYRE use.

- Default
  - Will use the number of threads set in `mesa_mp_threads`, or if that is not set then will use the number set in `$OMP_NUM_THREADS`.
- Examples
  - `4`

2.4 Stages

2.4.1 mesa_params

`dict` or `list[dict]`

If a dictionary, the parameter value possibilities to use to construct the grid of MESA models to run.

If a list, the parameter value combinations of the models to run.

- Examples

```
# Use 6 models with varying y values and inital masses
{
    "y": [1.0, 1.2, 2.5],
    "initial_mass": [1, 5]
}

# Use 2 models with different y and initial mass values
[
    { "y": 0.27, "initial_mass": 1 },
    { "y": 0.30, "initial_mass": 1.5 }
]
```

2.4.2 mesa_derived

`function[dict, dict]` - [Optional]

The function to apply to each MESA parameter combination to extract additional values plug into the MESA config templates specified in `mesa_configs`.

- Examples

```
# Add a max age to use that is based on the initial_mass
def calc_mesa_derived(mesa_params):
    derived = dict(mesa_params)
```

(continues on next page)

(continued from previous page)

```
    initial_mass = mesa_params["initial_mass"]

    mass_lookup = {
        "1": 1000000000,
        "1.5": 500000000
    }

    derived["max_age"] = mass_lookup[str(initial_mass)]

    return derived
```

2.4.3 gyre_params

`function[dict, pd.DataFrame, dict]` - [Optional]

The function to apply to the MESA parameter combination and MESA profile data to determine the parameter value possibilities to use to construct the grid of GYRE runs to perform.

- Examples

```
# Calculate l=0, l=1, and l=2 oscillations for profiles with a star age␣
↪greater than 1 Gyr
def calc_gyre_params(mesa_params, mesa_data):
    return {
        "profile": mesa_data[mesa_data["star_age"] > 1000000000]["profile"]
        "l": [0, 1, 2]
    }
```

2.4.4 gyre_derived

`function[dict, pd.DataFrame, dict, dict]` - [Optional]

The function to apply to each group of MESA parameter combination, MESA profile data, and GYRE parameter combination to extract additional values plug into the GYRE config template specified in `gyre_config`.

- Examples

```
# Use a different frequency range for each l value
def calc_gyre_derived(mesa_params, mesa_data, gyre_params):
    derived = dict(gyre_params)

    derived["freq_min"] = gyre_params["l"] * 200
    derived["freq_max"] = gyre_params["l"] * 200 + 500

    return derived
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search