Hand-out for the third session

Hydrodynamic modelling of planet-disk interaction and planet migration

Sareh Ataiee

in case of a broken link please contact me on sarehataiee@um.ac.ir

For this session, we will employ the grid-based MHD code `FARGO3D` and plot the outputs using the small package `fargo3dplot`. You are supposed to have a working C compiler and a Python3 installation along with the standard scientific python packages such as `numpy` and `matplotlib`. You are highly recommand to install `Anaconda` because we would also need `IPython` for plotting.

1 What you need to do before coming to the session

1.1 Installing FARGO3D

If you are one of those who carefully read the manual before using an application, please visit the main page of the code, where you find some information about the code features and learn about the legacy versions. But if you have no time for details, you can directly clone the repository.

First of all, type the below lines in a terminal to make a directory named `fargo3d` in your `$HOME`, navigate to it, and clone the repository.

```
mkdir fargo3d
cd fargo3d
git clone https://bitbucket.org/fargo3d/public.git
```

Now you should see the code directories in `$HOME/fargo3d/public`. Here is the code's user guide in pdf. Please check it out and try to compile and do the first run as is explained in the manual. After the first run, it is recommanded to read a little about the structure of the code in the third chapter.

It is more convenient to make a small modification in the source code to have equal number of lines in the output files of the planets, namely `orbit%n.dat` which keeps the orbital parameters of the `n`th planet, `bigplanet%n.dat` that contains the position and velocity of the planet, and `tqwk%n.dat` which keeps the disc torque and power on the planet. To do so,

- open the file `output.c` in the `src` directory with your favourite editor,
- go to line 198, where you see `WriteTorqueAndWork(t, i);`,
- insert a new line above it and add the if condition `if (big == YES)`. You would have something like

```
if (big == YES)
    WriteTorqueAndWork(t, i);
```

- save the file and exit.

Now you can compile and run the code :-)

1.2 Using `fargo3dplot`

There are different ways to visualise the outputs of the code. You can use the python scripts that comes with `FARGO3D` and is in the `utils` directory, write your own script in any language you prefer, or download the tiny `fargo3dplot` package which will be used in the remaining of this manuscript. Preferring the later, please clone the package in your `fargo3d` directory via

```
git clone https://github.com/sataiee/fargo3dplot.git
```

Then, add the address of `fargo3dplot` to your `$PYTHONPATH` to ease importing the package in `IPython`. For example, in case of using a bash shell, you need to add the below line to your `$HOME/.bashrc` file:

```
export PYTHONPATH="$HOME/fargo3d:PYTHONPATH"
```

After that, either open a new shell or type `source $HOME/.bashrc`[1]. Now, if you type `from fargo3dplot import *`, in your favourite `IPython` environment (such as `ipython console`, `spyder`, or `Jupyter-notebook`), the package should be imported without any problem.

This package has only two main files containing a `PlotField` function. You can check it out by typing

[1] If you are using `spyder`, you need to add this path via `Tools-> PYTHONPATH manager`.

```
? PlotField
```

in your IPython environment. There are also a couple of classes named Grid, ReadField and ReadPlanet. Their description can be seen either by typing ? <name> where <name> is the name of the class or a class's method, or simply reading the file classes.py in the fargo3dplot directory.

2 What you would do during the hand-on session

In the hand-on session, we will play a bit with FARGO3D to understand how the torque on a planet and the disc surface density change in different conditions. If you have more than a free core on your computer, it would be faster to run the code in parallel (it would need openMPI though), otherwise you can go on with the sequential run. In case the simulation takes very long on a single core, you can download all outputs for this session from here.

2.1 Making a setup

For each exercise, we need to make a setup. It is easiest to copy one of the setup directories and then change it as you want. Type the below lines in your terminal to make a setup named test1.

```
cd ~/fargo3d/public/setups/  # Navigate to the setup directory
cp -r fargo/ test1           # Copy the fargo directory as test1
cd test1                     # Enter test1
# Rename the files names to the corresponding setup name
for i in fargo.*; do mv $i test1."${i#*.}"; done
```

Now, we need to modify some of the files inside the test1 directory.

- In the test1.par file, we change the name of the setup and the output directory to test1. So, we would have

  ```
  Setup                   test1
  OutputDir               @outputs/test1
  ```

- In the test1.opt file, we should have the following options activated

  ```
  MONITOR_Y_RAW  = TORQ
  FARGO_OPT += -DLEGACY
  ```

  The first line dump the monitoring file for the torque density on the planet, and the second line tells the code to output the legacy files such as orbit%n.dat,

`bigplanet%n.dat`, and `tqwk%n.dat`, where `%n=0,1,2,...` represents the index of the planet.

2.2 Lindblad torque

In this exercise, we aim to measure only the Lindblad torque on a Super-Earth. Therefore, we need vanishing co-rotation torque on the planet or, in other words, we need an isothermal disc with (almost) zero vortensity gradient. For the moment, we do not allow the planet to migrate because the Lindblad torque may differ as the planet moves in the disc. To make such a setup:

- Make a setup named `lindblad` as explained in Sec. 2.1.
- In the `.opt` file: replace the `FARGO_OPT += -DVISCOSITY` by `FARGO_OPT += -DALPHAVISCOSITY`.
- In the `.par` file:
  1. replace the parameter `NU` by `ALPHA` and set its value to `1e-4`,
  2. set the value of `SigmaSlope` to `1.5`
  3. set the values of `FlaringIndex` to `0.5`,
  4. set the `PlanetConfig` to `planets/SuperEarth.cfg`, that contains a non-migrating planet with planet-to-star mass ratio $q = 10^{-5}$,
  5. make sure `Ntot` equals to `1000`,
  6. add the below line to have logarithmic spacing,

     ```
     Spacing                 Log
     ```

     This gives a better resolution in the inner part of the disc where the planet is located.

Navigate to the `~/fargo3d/public` directory via `cd ../../; pwd`. Compile and run the code on 4 cores through

```
make SETUP=lindblad PARALLEL=1
mpirun -np 4 ./fargo3d -m setup/lindblad/lindblad.par
```

If you prefer a serial run, use the below commands instead

```
make SETUP=lindblad PARALLEL=0
./fargo3d setup/lindblad/lindblad.par
```

After you get message of `End of the simulation!`, assuming that you already have imported `fargo3dplot` in your IPython console, type below lines to see the

surface density and its perturbation, radial velocity, and azimuthal velocity. It's better to save each figure in a proper location for future reference.

```
cd ~/fargo3d/public/
main_dir = "outputs/lindblad"
# Making the figure and axes
fig, ax = plt.subplots(ncols=2, nrows=2, figsize=(12,6))
fig.subplots_adjust(top=0.966, bottom=0.099, left=0.055, \
 right=0.965, hspace=0.257, wspace=0.182)
# Plot the surface density
PlotField(directory=main_dir, field='dens', nout=30, \
 fig=fig, ax=ax[0,0], cblabel=True, cbar=True, settitle=False,\
 vmin=1e-4, vmax=2e-3)
# Plot the surface density perturbation
PlotField(directory=main_dir, field='dens', nout=30, \
 fig=fig, ax=ax[0,1], perturbation=True, settitle=False, \
 vmin=-0.2, vmax=0.2)
# Plot the azimuthal velocity
PlotField(directory=main_dir, field='vx', nout=30, \
 fig=fig, ax=ax[1,0], settitle=False)
# Plot the radial velocity
PlotField(directory=main_dir, field='vy', nout=30, \
 fig=fig, ax=ax[1,1], settitle=False)
```

Q: What are the main features? Could you guess what happens around the co-rotation region?

Let's plot the torque and torque density on the planet.

```
# Reading the grid
grid = Grid(main_dir)
# Reading the planet's data
pl0 = ReadPlanet(main_dir, nplanet=0)
# Fig. 1: total torque
fig1, ax1 = plt.subplots()
torq = pl0.GiveTorquePower()[0]
ax1.plot(pl0.time, torq)
ax1.set_xlabel(r'$time\ \rm [year]$')
ax1.set_ylabel(r'$\Gamma\ \rm [code\ unit]$')
# Fig. 2: torque density vs distance from the planet
H  = 0.05* (grid.ymid)**1.5
fig2, ax2 = plt.subplots()
for i in range(0,50,10):
    torq_dens = pl0.GiveTorqueDensity(i, grid.ny)
    t_index = np.argmax(pl0.time >= i)
```

```
    ax2.plot((grid.ymid-pl0.semi[t_index])/H, torq_dens, \
     label=r'$n_{{\rm out}}={}$'.format(i))
ax2.set_xlabel(r'$(r-r_{p})/H$')
ax2.set_ylabel(r'$d\Gamma/dr\ \rm [code\ unit]$')
ax2.set_xlim([-7,7])
ax2.legend()
```

Q: Try to find from what distance the main torque comes from? Which peak is closer to the planet and which one has a larger amplitude?

Q: Double the resolution by increasing `Nx` to `768` and `Ny` to `256`. It is better to make a separate setup for example named `lindblad_high` for storing the data. Make and repeat the simulation. How do the results differ than the low resolution run?

2.3 Migration

Make another setup named `migration` similar to the setup of previous exercise except that in the `planets/SuperEarth.cfg` file, set `Feels Disk` to `Yes`. This tells the code to apply the acceleration from the disc on the planet. Make and run again until the simulation is over and make the below plot

```
# Reading the planet's data
main_dir = "outputs/migration"
pl0 = ReadPlanet(main_dir, nplanet=0)
# Fig. 3: migration of the planet
fig3, ax3 = plt.subplots()
ax3.plot(pl0.time, pl0.semi)
ax3.set_xlabel(r'$time\ \rm [year]$')
ax3.set_ylabel(r'$a$')
```

Q: In which direction does the planet migrate? Could you approximate a migration timescale for this setup?

Q: There are steps in the curve you just plotted. What would you suggest for having a smoother curve?

2.4 An eccentric planet

In this exercise, we aim to see (a) if the torque on a non-migrating eccentric planet differs than a circular planet, and (b) how the planet's eccentricity damps by time.

To investigate the first question:

- Make a new setup similar to Sec. 2.2 named `eccentric`.
- In the `.par` file, set the paramter `Eccentricity` to 0.1.
- Make sure the planet does not feel the disc in the `.cfg` file.
- Make, run, and wait until the simulation is finished.
- Plot the torque against time.

```
main_dir = "outputs/eccentric/"
# Reading the grid
grid = Grid(main_dir)
H  = 0.05* (grid.ymid)**1.5
# Reading the planet's data
pl0 = ReadPlanet(main_dir, nplanet=0)
# Fig. 4: total torque
fig4, ax4 = plt.subplots()
torq = pl0.GiveTorquePower()[0]
ax4.plot(pl0.time, torq)
ax4.set_xlabel(r'$time\ \rm [year]$')
ax4.set_ylabel(r'$\Gamma\ \rm [code\ unit]$')
```

Q: How the torque is different compared to the one of Sec. 2.2? Could you propose a reason?

Let's check the surface density perturbation and torque density at various times in a single orbit of the planet. The torque density is already finely sampled (every `Dt`/2πth of an orbit, here 1/20th), but the fields including surface density is saved every `Dt*Ninterm` orbit (one orbit in our setup). Therefore, we continue the simulation for one more orbit except that we dump the fields also every 1/20th of orbit. In the `.par` file change

- `Ninterm` to `1`,
- `Ntot` to `71`.

Save, make, and restart via

```
make SETUP=eccentric PARALLEL=1
mpirun -np 4 ./fargo3d -m -S 50 setups/eccentric/eccentric.par
```

In `outputs/eccentric` directory, you should see 20 more outputs for the fields. Explicitly said, all field outputs with the number between $n_{out} = 50$ to 70 belong to $t = 50$ to 51 orbit with $\delta t = 1/20$ orbit between the two consecutive outputs. Now, check out the torque density and surface density perturbation at different times in an orbit.

```
main_dir = "outputs/eccentric/"
selected_times = [50,50.25,50.5,50.75,51]
selected_outputs = [50, 55, 60, 65, 70]
# Reading the grid
grid = Grid(main_dir)
H  = 0.05* (grid.ymid)**1.5
# Reading the planet's data
pl0 = ReadPlanet(main_dir, nplanet=0)
# Fig. 5: torque density vs distance from the planet at
# various time between 45th and 46th planet's orbital time
fig5, ax5 = plt.subplots()
# Fig. 6: Surface density perturbation at the same times
fig6, ax6 = plt.subplots(ncols=5, nrows=1, figsize=(18,3))
fig6.subplots_adjust(top=0.876, bottom=0.198, left=0.042,\
 right=0.968, hspace=0.257, wspace=0.28)
for i, time_now in enumerate(selected_times):
    torq_dens = pl0.GiveTorqueDensity(time_now, grid.ny)
    t_index = np.argmax(pl0.time >= time_now)
    ax5.plot((grid.ymid-pl0.semi[t_index])/H, torq_dens, \
     label=r'$t={}, n_{{out}}={}$'.format(time_now, \
     selected_outputs[i] ))
    PlotField(directory=main_dir, field='dens', \
     nout=selected_outputs[i], fig=fig6, ax=ax6[i], \
     perturbation=True, vmin=-0.2, vmax=0.2)
ax5.set_xlabel(r'$(r-r_{p})/H$')
ax5.set_ylabel(r'$d\Gamma/dr\ \rm [code\ unit]$')
ax5.set_xlim([-7,7])
ax5.legend()
```

Probing the second question needs applying the disc's force on the planet and also running for a longer time. Therefore, make a setup similar to `eccentric` and name it `eccentric_mig`, and apply the below modifications in the setup files:

- In the `.cfg` file, set `Feel disk` to `Yes`.
- In the `.par` file, change:
    1. `Ninterm` to 200 to have field outputs each 10 orbits and avoid making so many (useless) files,
    2. `Ntot` to 8000.

Make and run the simulation!

Using the below script, you can see how eccentricity damps by time.

```
# Reading the planet's data
main_dir = "outputs/eccentric_mig/"
pl0 = ReadPlanet(main_dir, nplanet=0)
# Fig. 7: migration and eccentricity damping of the planet
fig7, ax7 = plt.subplots(ncols=1, nrows=2, figsize=(6,6), \
 sharex=True)
ax7[0].plot(pl0.time, pl0.semi)
ax7[1].plot(pl0.time, pl0.GiveEcc())
ax7[1].set_xlabel(r'$time\ \rm [year]$')
ax7[0].set_ylabel(r'$a$')
ax7[1].set_ylabel(r'$e$')
```

Q: Estimate roughly the eccentricity damping and migration timescales and obtain their ratio.

2.5 A massive planet

In this exercise, we investigate how are the surface density and the torque on the planet for a massive gap-opening planet.

- Make a setup named `massive` similar to Sec. 2.2.
- In the `.par` file:
  1. change `PlanetConfig` to `planets/jupiter.cfg` in order to have a non-migrating Jupiter-mass planet,
  2. set `Ninterm` to 200 to save space in your machine,
  3. increase `Ntot` to 10000 because opening a gap around the planet needs time.

Make and run!

Using the script in Sec. 2.2, plot the 2D fields for various `nout` values and see when the gap is formed. Plotting the azimuthally averaged surface density would help to distinguish the gap depth and width. You can do it by setting the keyword `averaged=True` in the `PlotField` function, for example

```
main_dir = "outputs/massive/"
# Making the figure and axes
fig8, ax8 = plt.subplots()
# Plot the surface density for several outputs
for i in range(0,51,5):
    PlotField(directory=main_dir, field='dens',  nout=i, \
     fig=fig8, ax=ax8, averaged=True, xlog=True, ylog=True)
```

Q: Plot the torque and torque density. How these quantities are different that the ones of a low-mass planet in Sec. 2.2?

2.6 Co-rotation torque

Up to now, we examined the torque on a planet in an isothermal disc with (almost) vanishing vortensity gradient, that means the torques obtained in the previous exercises were (mostly) Lindblad torque. As the final exercise, we want to see what happens if the second important torque component, the co-rotation torque, exists.

Properly resolving the co-rotation torque is tricky and computationally expensive because (a) the half horseshoe width x_s needs to be resolved at least with four radial cells, and (b) the minimum running time is about the libration timescale $\tau_{\rm lib}$ which depends inversely on x_s. Hence, to avoid a very long running time due to the high spacial resolution and long running time, we use a more massive planet such that x_s increases enough while the planet does not open a gap. Therefore, we triple the mass of the planet in Sec. 2.2 and also increase the resolution such that we have four radial cells in every x_s. To see the oscillation and saturation of the co-rotation torque, we run only one simulation with a low value for the viscosity parameter α, but you are highly recommanded to try also larger values.

For a planet with planet-to-star mass ratio $q = 3 \times 10^{-5}$, the half horseshoe width $x_s \sim 1.1\sqrt{q/h a_p} \sim 0.027$ for a planet at $a_p = 1$. The running time needs to be about $\tau_{\rm lib} \sim 8\pi a_p / 3\Omega_p x_s$ orbits that becomes about 300 orbits. We also modify the resolution to have $dr \sim x_s/4$.

Build a new setup with the name of `corotation16` using the same setup in Sec. 2.2 and apply below changes:

- In the `.par` file, set:
  1. `ALPHA` to `1e-6`,
  2. `SigmaSlope` to `0.5`,
  3. `FlaringIndex` to `0.0`,
  4. `Ny` to `273`,
  5. `Ninterm` to `200`,
  6. `Ntot` to `8000`.
- In the `planets/SuperEarth.cfg` file, increase the `Mass` to `3e-5`.

Make, run, and go for a coffee/tea or a chat. It would take a bit longer than the previous exercises. If it takes very long, you can download all of the outputs from here.

Plot the total torque versus time and the vortensity perturbation for several

outputs using the below script

```
main_dir = "outputs/corotation16/"
# Reading the grid
grid = Grid(main_dir)
# Reading the planet's data
pl0 = ReadPlanet(main_dir, nplanet=0)
# Fig. 9: total torque
fig9, ax9 = plt.subplots()
torq = pl0.GiveTorquePower()[0]
ax9.plot(pl0.time, torq)
ax9.set_xlabel(r'$time\ \rm [year]$')
ax9.set_ylabel(r'$\Gamma\ \rm [code\ unit]$')
# Fig. 10: vortensity perturbation for several snapshots
fig10, ax10 = plt.subplots(ncols=5, nrows=1, figsize=(18,3), \
 sharex=True, sharey=True)
fig10.subplots_adjust(top=0.876, bottom=0.198, left=0.042,\
 right=0.968, hspace=0.257, wspace=0.28)
for i, nout in enumerate(range(0,41,10)):
    # Making one figure per output
    PlotField(directory=main_dir, field='vortensity', \
     nout=nout, fig=fig10, ax=ax10[i], perturbation=True, \
     vmin=-0.3, vmax=0.3, cmap='seismic', settitle=True)
```

Note that vortensity *perturbation* shows how vortensity is changed compared to our initial setup. Thus, it is obviously zero for $n_{\rm out} = 0$.

Q: Can you imply the sign of the co-rotation torque (positive or negative) for this setup?

Q: Repeat the above procedure for `ALPHA` values of `1e-2` to observe the fully unsaturated co-rotation torque.

Note: For the co-rotation torque to be properly calculated in a serious project (and not a quick exercise), we need typically 6 radial cells per horseshoe half-width. Performing resolution tests are always highly advised when the co-rotation torque has a role in the migration of the planet.

:-) Have fun!