Nemesis_and_the_Sun

September 29, 2023

1 AMUSE tutorial on modules and channels

AMUSE is composed of domain-specific modules (i.e. the physics modules), which are often written in some native compiled language, and data-handling. The latter tends to be confusing for novel users. Data representation in AMUSE is often replicated. So there can be a parameter *mass* to indicate a property of a star in a stellar-evolution code, in a gravitational dynamics code and in your user script. These three parameters *mass* can mean the same, or they can have a different meaning. This makes running amuse somewhat confusing at times. Here we show how to formally separate these data streams.

But first we will enter the realm of modules.

[2]:
```
#Load in the amuse units module, the particle module and
# some generator for producing some conditions.
from amuse.units import units, constants
from amuse.lab import Particles
from amuse.ext import orbital_elements as oe
```

We now want to generate the Solar system, as it was observed on April 5th 2063 using the JPL emphemeris. Let's start with the inner most planet, Mercury, in the Sun's barycenter, which gives the following output:

$$SOE 2474649.500000000 = A.D. 2063-Apr-05 00:00:00.0000 TDB X =-2.689701945882702E-01 Y = 1.947622508089924E-01 Z = 4.148911797144442E-02 VX=-2.230751806031045E-02 VY=-2.157655548570704E-02 VZ= 2.791759037949523E-04 LT= 1.932846451835678E-03 RG= 3.346619889187973E-01 RR= 5.406475142917982E-03

We can use these to start an AMUSE particle set, as follows

[3]:
```
Sun = Particles(1)
Sun.mass = 1 | units.MSun
Sun.position = (0,0,0) | units.km
Sun.velocity = (0,0,0) | units.km/units.s
planets = Particles(8)
planets[0].mass = 3.302e+23 | units.kg # also according to JPL)
planets[0].position = (-2.689701945882702E-01,
                       1.947622508089924E-01,
                       4.148911797144442E-02) | units.au
```

```
planets[0].velocity = (-2.230751806031045E-02,
                       -2.157655548570704E-02,
                       2.791759037949523E-04) | units.au/units.yr
```

And continue doing this for the other 7 planets. Then add the Sun and we have the Solar system's particle set. It would be easier if there is a handy routine with the same effect, in particular because we will frequently be using the Solar system as some sort of template for a rather typical planetary system, or for specifically sutdying this planetary system. We therefore have a handy routine that allows us to initialize the Solar system.

[4]:
```
from amuse.ext.solarsystem import new_solar_system
sun_and_planets = new_solar_system()
#sun_and_planets = sun_and_planets[0].as_set()
print(sun_and_planets)
```

```
|S8
                 key         mass         name       radius            vx
vy           vz            x            y            z
                   -         MSun         none         RSun       km / s
km / s       km / s            AU           AU           AU
==================  ===========  ===========  ===========  ===========
==========  ==========  ==========  ==========  ==========
  540660266701983520    1.000e+00          SUN    1.000e+00   -1.874e-03
-1.560e-02    1.104e-04   -8.354e-03    1.923e-03    2.145e-04
1584830979462845461 5    1.660e-07      MERCURY    5.654e-03    1.032e+01
-4.214e+01   -4.388e+00   -3.923e-01   -1.749e-01    2.101e-02
108699492566780017 00    2.448e-06        VENUS    1.403e-02   -1.704e+01
3.049e+01    1.401e+00    6.251e-01    3.518e-01   -3.157e-02
128395858856924526 64    3.040e-06     EARTHMOO    1.482e-02    2.844e+01
6.966e+00    1.301e-04    2.337e-01   -9.855e-01    2.100e-04
124212329561765481 52    3.227e-07         MARS    7.853e-03   -2.299e+01
5.985e+00    6.908e-01    2.435e-01    1.528e+00    2.599e-02
129249825900410423 63    9.548e-04      JUPITER    1.619e-01    2.872e+00
1.331e+01   -1.194e-01    4.833e+00   -1.158e+00   -1.034e-01
 76387038790474961 55    2.859e-04       SATURN    1.342e-01   -4.794e+00
8.639e+00    4.001e-02    8.335e+00    4.127e+00   -4.033e-01
176125817963864425 41    4.366e-05       URANUS    5.834e-02    5.131e+00
4.103e+00   -5.124e-02    1.289e+01   -1.511e+01   -2.231e-01
   14258115921289155    5.151e-05      NEPTUNE    5.572e-02    4.640e+00
2.804e+00   -1.647e-01    1.537e+01   -2.592e+01    1.795e-01
 69251980397798196 72    7.396e-09         PLUTO    3.413e-03    5.144e+00
-2.956e+00   -1.172e+00   -1.152e+01   -2.708e+01    6.229e+00
==================  ===========  ===========  ===========  ===========
==========  ==========  ==========  ==========  ==========
```

We adopted the module `ext.new_solar_system` to generate a ready-made solar system. You can also make a solar system that includes it's moons. For this you will have to use the routine 'new_lunar_system', which is a module from the 'amuse.ic.solar_system_moons' package.

Now you can plot the positions of the sun and planets in the orbital plane.

In 1984
Whitemire and Jackson and Davies et al 1984(preprint) argued that the Solar system was orbited by a hypothetical star called nemesis, named after the goddess of Rhamnous who punished those who did not honor the Olympians.

This companion to the Sun would have an orbital separation of about 95,000 au, and due to its interaction with the Oort cloud would initiate periodic comet shouwers in the inner Solar system; punishing life on Earth for being atheistic?

Let's adopt the Solar system as we have just initialized and add Nemesis as an 0.2MSun main-sequence star in a 95,000 au orbit with an eccentricity of 0.7.

Let's construct tha binary system of the Sun (without planets) and Nemesis.

[5]:
```
msun = sun_and_planets.mass.sum()
mnemesis = 0.2 | units.MSun
a = 95000 | units.au
e = 0.7
from amuse.ext.orbital_elements import new_binary_from_orbital_elements
sun_and_nemesis = new_binary_from_orbital_elements(msun, mnemesis,
                                          a, e, G=constants.G)
setattr(sun_and_nemesis, "name", ["Companion", "Nemesis"])
print(sun_and_nemesis)
```

```
                 key          mass          name           vx           vy
vz            x            y            z
                   -  1.98892e+30 * kg         none  3646245880.3572216 * m *
s**-1  3646245880.3572216 * m * s**-1  3646245880.3572216 * m * s**-1
149597870691.0 * m  149597870691.0 * m  149597870691.0 * m
====================  ===========  ===========  ===========  ===========
===========  ===========  ===========  ===========
 6522804020832650623    1.001e+00    Companion    0.000e+00   -1.151e-08
-0.000e+00   -4.745e+03   -0.000e+00   -0.000e+00
12429777838030029187    2.000e-01      Nemesis    0.000e+00    5.764e-08
0.000e+00    2.376e+04    0.000e+00    0.000e+00
====================  ===========  ===========  ===========  ===========
===========  ===========  ===========  ===========
```

Now, the sun in this binary-particle set and the sun in the Solar system particle-set do not have the same position.

[6]:
```
sun = sun_and_planets[sun_and_planets.name=="SUN"]
print(sun.position.in_(units.au))
print(sun_and_nemesis[sun_and_nemesis.mass>0.9|units.MSun].position.in_(units.
  ↪au))
```

```
[[-0.00835408821766, 0.0019229394648, 0.000214547107396]] au
[[-4744.69448798, -0.0, -0.0]] au
```

And, equally they will have different velocity.

However, we would like to adot these binary parameters for the Sun-Nemesis binary system. For this, we first move the entire Solar system to the Sun barycenter, move it to Nemesis' companion and add Nemesis.

[7]:
```
sun_and_planets.position -= sun.position
sun_and_planets.velocity -= sun.velocity
companion = sun_and_nemesis[sun_and_nemesis.name=="Companion"]
nemesis = sun_and_nemesis[sun_and_nemesis.name=="Nemesis"]
sun_and_planets.position += companion.position
sun_and_planets.velocity += companion.velocity
sun_and_planets.add_particle(sun_and_nemesis[1])
print(sun_and_planets)
```

```
                 key          mass          name        radius            vx
vy            vz             x             y             z
                  -          MSun          none          RSun        km / s
km / s        km / s           AU            AU            AU
====================  ===========  ===========  ===========  ===========
===========  ===========  ===========  ===========  ===========
  540660266701983520    1.000e+00           SUN    1.000e+00    0.000e+00
-4.198e-02    0.000e+00   -4.745e+03    0.000e+00    0.000e+00
1584830979462845 4615    1.660e-07       MERCURY    5.654e-03    1.032e+01
-4.217e+01   -4.389e+00   -4.745e+03   -1.769e-01    2.080e-02
1086994925667800 1700    2.448e-06         VENUS    1.403e-02   -1.704e+01
3.046e+01    1.400e+00   -4.744e+03    3.499e-01   -3.179e-02
1283958588569245 2664    3.040e-06      EARTHMOO    1.482e-02    2.845e+01
6.939e+00    1.967e-05   -4.744e+03   -9.875e-01   -4.543e-06
1242123295617654 8152    3.227e-07          MARS    7.853e-03   -2.298e+01
5.959e+00    6.907e-01   -4.744e+03    1.526e+00    2.578e-02
1292498259004104 2363    9.548e-04        JUPITER    1.619e-01    2.874e+00
1.329e+01   -1.196e-01   -4.740e+03   -1.160e+00   -1.036e-01
 763870387904749 6155    2.859e-04         SATURN    1.342e-01   -4.792e+00
8.613e+00    3.990e-02   -4.736e+03    4.125e+00   -4.035e-01
1761258179638644 2541    4.366e-05         URANUS    5.834e-02    5.133e+00
4.076e+00   -5.135e-02   -4.732e+03   -1.511e+01   -2.233e-01
   14258115921289155     5.151e-05        NEPTUNE    5.572e-02    4.641e+00
2.777e+00   -1.648e-01   -4.729e+03   -2.592e+01    1.793e-01
 6925198039779819672     7.396e-09          PLUTO    3.413e-03    5.146e+00
-2.982e+00   -1.172e+00   -4.756e+03   -2.708e+01    6.229e+00
1242977783803002 9187    2.000e-01         Nemesis    0.000e+00    0.000e+00
2.102e-01    0.000e+00    2.376e+04    0.000e+00    0.000e+00
====================  ===========  ===========  ===========  ===========
===========  ===========  ===========  ===========  ===========
```

You have created a hypothetical planetary system with the Sun, 8 planets and 1 dwarf planet (mimiking the Solar system), and a secondary star (Nemesis) in a wide and elliptic orbit.

1.1 Assignments and questions:

1.1.1 Assignment 1:

To check the orbit of Nemesis around the Sun you can calculate the orbital elements from the Cartesian coordinates of the Sun and Nemesis. Perform this operation and check the orbital elements of the nemesis-Sun binary.

1.1.2 Question 1:

When performing this assignment, we found that the orbit between the Sun and Nemesis is slightly off from what we put in. The semi-major axis is about 95,606 au with an eccentricity of about 0.702.

Explain why the orbit is not what you anticipated it to be, and what can you do to mitigate this?

1.1.3 Assignment 2:

Add 100 zero-mass Oort cloud object in circular orbits around the Sun with semi-major axes equally spaced between 10000au and 50000 au. Integrate those orbits for 10 full orbits of the Sun-Nemesis binary. Make a scatter plot of eccentricity as function of the semi-major axis for the initial and final Oort-cloud objects. Explain the figure.

Be aware that 10 Nemesis periods is ~160 Myr, and running the system for that time will take a lot of time (about an hour). Start with shorter times and build up towards longer times when you know your code runs. You can also try different codes, and leaving out some planets. Is Mercury really that important? But leaving it out may let the code take greater timesteps.

1.1.4 Question 3:

Do you think that Nemesis could be the cause of periodic cometary visitors?

1.1.5 Assignment 1

[8]:
```
elements = oe.get_orbital_elements_from_arrays(
    (sun_and_planets[sun_and_planets.name=="SUN"].
  ↪position)-(sun_and_planets[sun_and_planets.name=="Nemesis"].position),
    (sun_and_planets[sun_and_planets.name=="SUN"].
  ↪velocity)-(sun_and_planets[sun_and_planets.name=="Nemesis"].velocity),
    sun_and_planets[sun_and_planets.name=="SUN"].
  ↪mass+sun_and_planets[sun_and_planets.name=="Nemesis"].mass)

print("The eccentricity is:", elements[1][0])
print("The semi major axis is:", elements[0][0].in_(units.AU))
```

```
The eccentricity is: 0.701900938082
The semi major axis is: 95605.8023685 AU
```

1.1.6 Question 1

The orbit is slightly off, because there are other planets in play. They slightly distort the orbit of the nemesis! We can fix this by taking the total mass of the system!

1.1.7 Assignment 2

[10]:
```
import numpy as np
from amuse.units import nbody_system
from amuse.community.ph4.interface import ph4

begin = 10000
end = 50000
range_au = np.linspace(begin, end, 100)

for i in range(100):
    _, object = oe.generate_binaries(
        1 | units.MSun,
        0 | units.MEarth,
        range_au[i] | units.AU,
        0)
    object.position += sun.position
    object.velocity += sun.velocity
    object.name = "Object{}".format(i)
    sun_and_planets.add_particle(object)
    sun_and_planets.move_to_center()
```

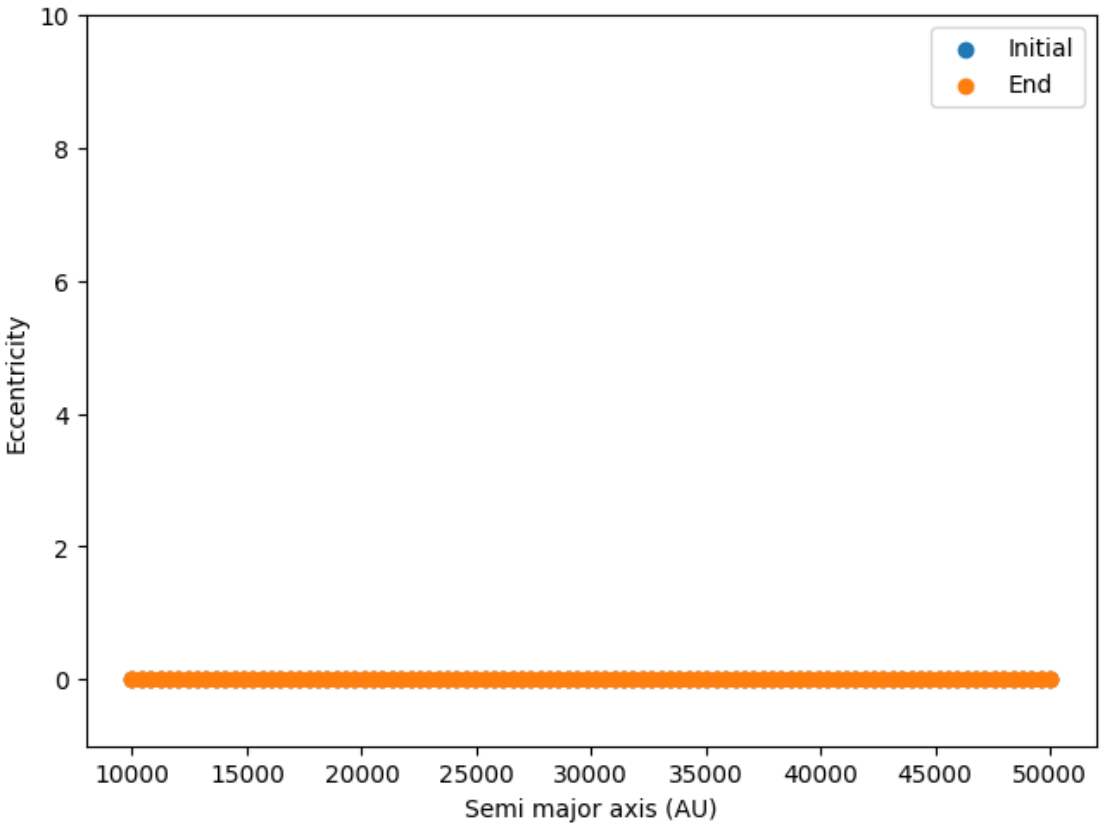
[11]:
```
def sim(sim_time):
    converter=nbody_system.nbody_to_si(sun_and_planets.mass.sum(), 100000 |␣
 ↪units.AU)
    bodies = sun_and_planets
    #bodies.scale_to_standard(converter)
    # WHY DON´T WE USE IT THIS TIME?????????
    gravity = ph4(converter)
    gravity.particles.add_particles(bodies)
    channel = gravity.particles.new_channel_to(bodies)
    times = np.arange(0, sim_time.number, 0.1) | units.Myr
    for time in times:
        gravity.evolve_model(time)
        channel.copy() # Copy from gravity.particles to bodies
        if not time.value_in(units.Myr)%0.1:
            print("cluster at Time=", time.in_(units.Myr))
    gravity.stop()
    end_eccentricity = []
    end_semi_major_axis = []
    for j in range(100):
        elements = oe.get_orbital_elements_from_arrays(
```

```
                (sun_and_planets[sun_and_planets.name=="SUN"].
↪position)-(sun_and_planets[sun_and_planets.name=="Object{}".format(j)].
↪position),
                (sun_and_planets[sun_and_planets.name=="SUN"].
↪velocity)-(sun_and_planets[sun_and_planets.name=="Object{}".format(j)].
↪velocity),
                    sun_and_planets[sun_and_planets.name=="SUN"].mass)
        ###WHY IS TAKING THE FULL MASS OF THE SYSTEM INCORRECT THIS TIME OR IS␣
↪MY CONCLUSION IN Q1 WRONG?
        end_eccentricity.append(elements[1])
        end_semi_major_axis.append(elements[0].in_(units.AU).number)
    return end_eccentricity, end_semi_major_axis
```

[ ]:
```
sim_time = 0.2 | units.Myr
end_eccentricity, end_semi_major_axis = sim(sim_time)

#My code takes way too long to run, also tried Hermite but it didn't speed up
```

```
cluster at Time= 0.0 Myr
```

[13]:
```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(1, 1, layout='constrained', sharey=True)#,␣
 ↪sharey='row', sharex="col")
axs.scatter(range_au, np.zeros(100), label="Initial")
axs.scatter(end_semi_major_axis, end_eccentricity, label="End")
axs.set_xlabel("Semi major axis (AU)")
axs.set_ylabel("Eccentricity")
axs.legend()
axs.set_ylim(-1,10)

plt.show()
```


Initial
End
Eccentricity
Semi major axis (AU)
10
8
6
4
2
0
10000
15000
20000
25000
30000
35000
40000
45000
50000