Building Models with AMUSE

introductory tutorial

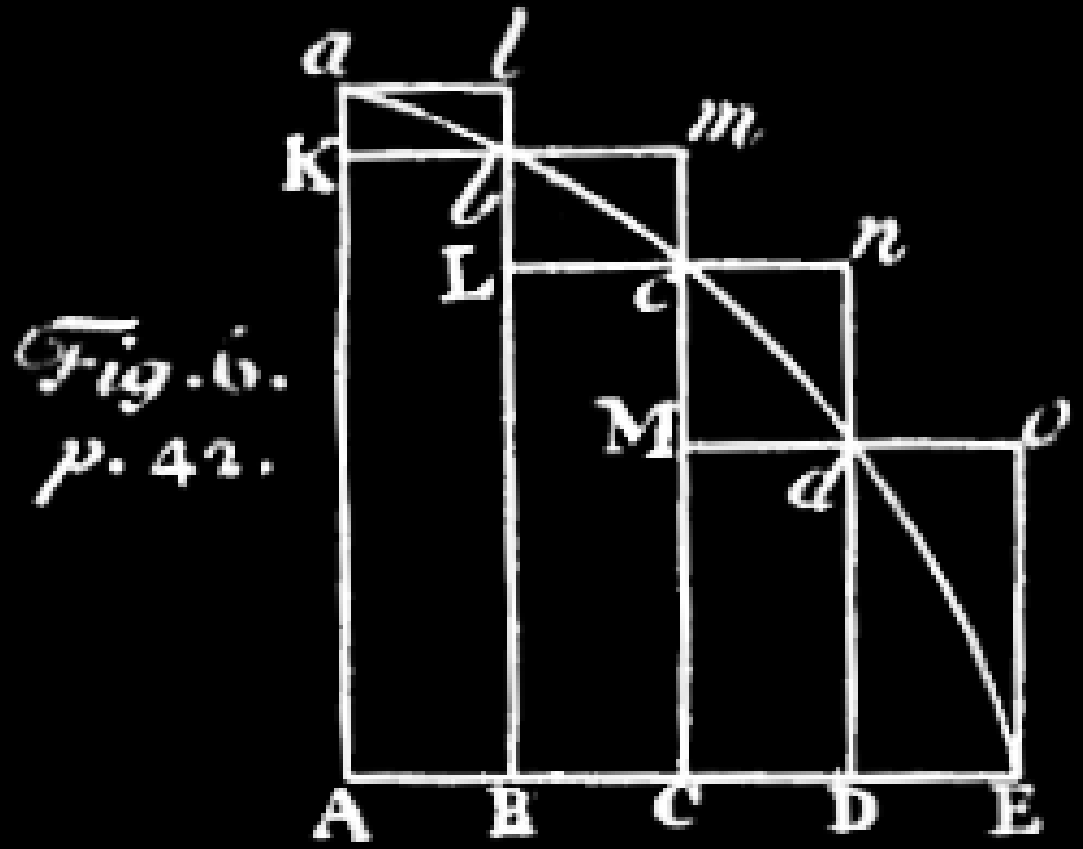


Fig. 6.
p. 42.

GD & SE

BHtree
hermite0
phiGRAPE
twobody
smallN
octgrav
mercury
huayno
mmc

sse
bse
evtwin
mesa

http://www.amusecode.org/doc/reference/available-codes.html

GD

imports

```
import numpy
from amuse.community.mercury.interface import MercuryWayWard
from amuse.ext.solarsystem import Solarsystem
from amuse.support.units import units
from amuse.support.data.values import VectorQuantity as Vq
from amuse.plot import *
```

```
def integrate_and_store():
```

initial conditions

```
  sun, planets = Solarsystem.new_solarsystem()
  timerange = Vq.arrange(0|units.day, 120|units.yr, 1|units.day))
```

setup code

```
  instance = MercuryWayWard()
  instance.initialize_code()
  instance.central_particle.add_particles(sun)
  instance.orbiters.add_particles(planets)
  instance.commit_particles()
  channels = instance.orbiters.new_channel_to(planets)
```

evolve

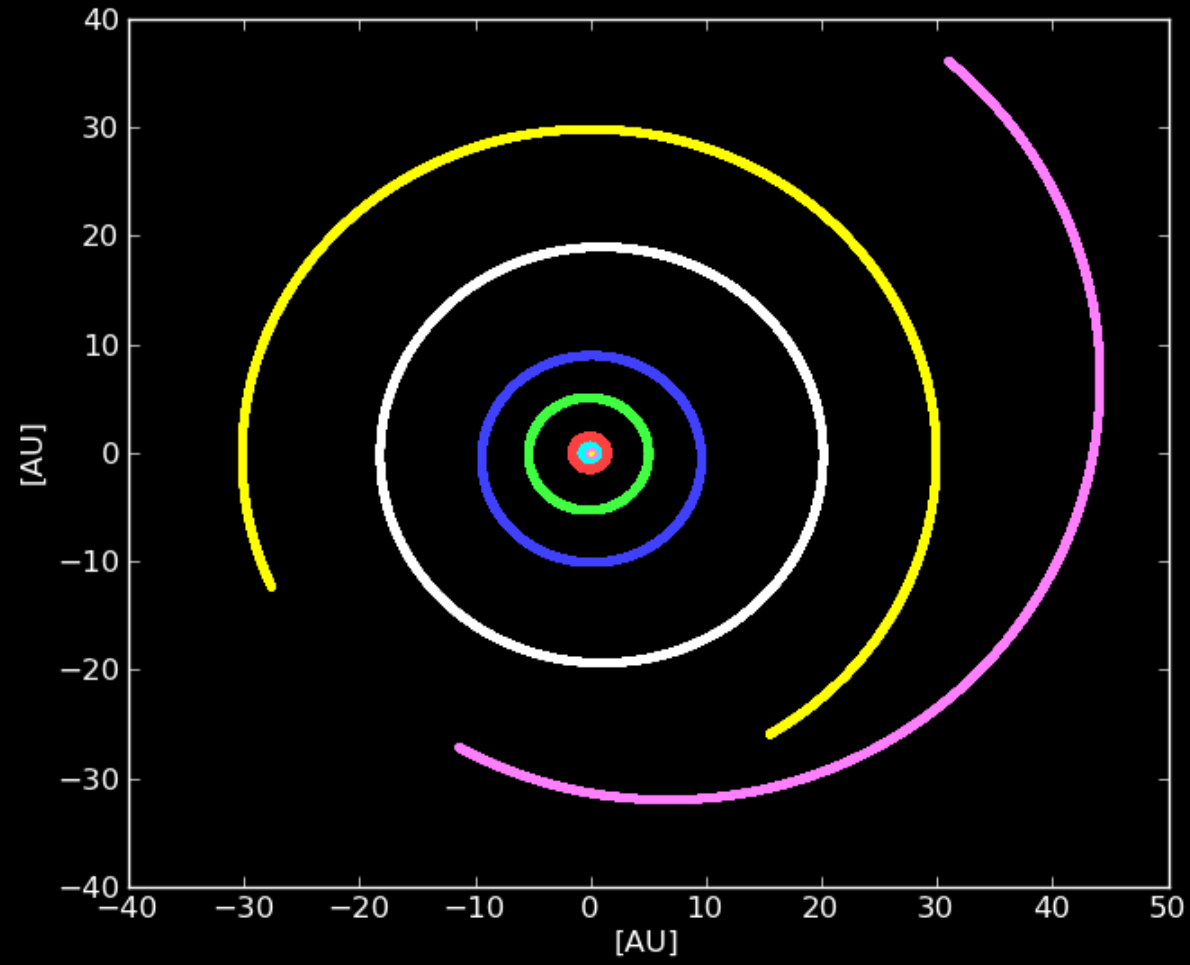
```
  for time in timerange:
        err = instance.evolve_model(time)
        channels.copy()
        planets.savepoint(time)

    instance.stop()
  return planets
```

```
def plotdata(planets):
```

process

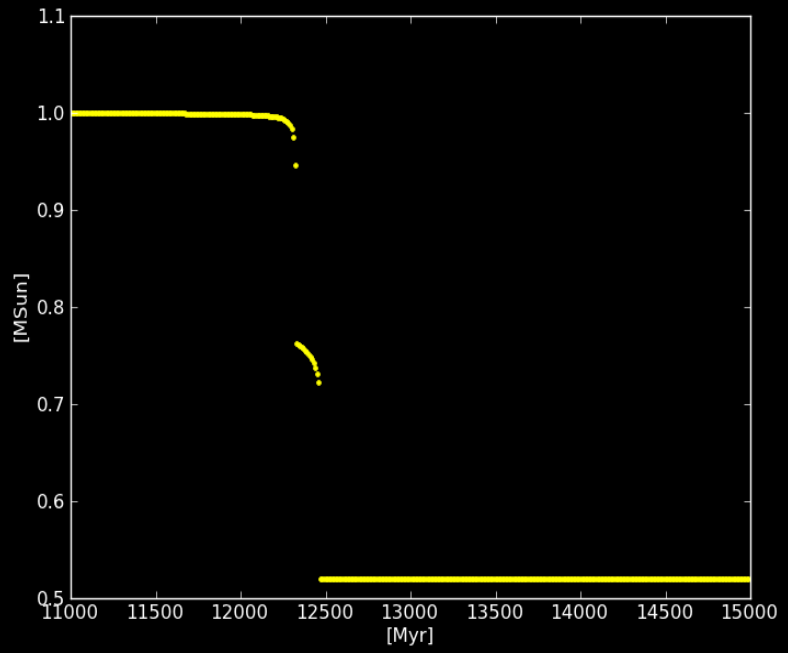
```
   for planet in planets:
       t, x = planet.get_timeline_of_attribute_as_vector("x")
       t, y = planet.get_timeline_of_attribute_as_vector("y")
       plot(x, y,'.')

   native_plot.show()
```

```
if __name__ == "__main__":
    planets = integrate_and_store()
    plotdata(planets)
```


40
30
20
10
0
−10
−20
−30
−40
[AU]
−40
−30
−20
−10
0
10
20
30
40
50
[AU]

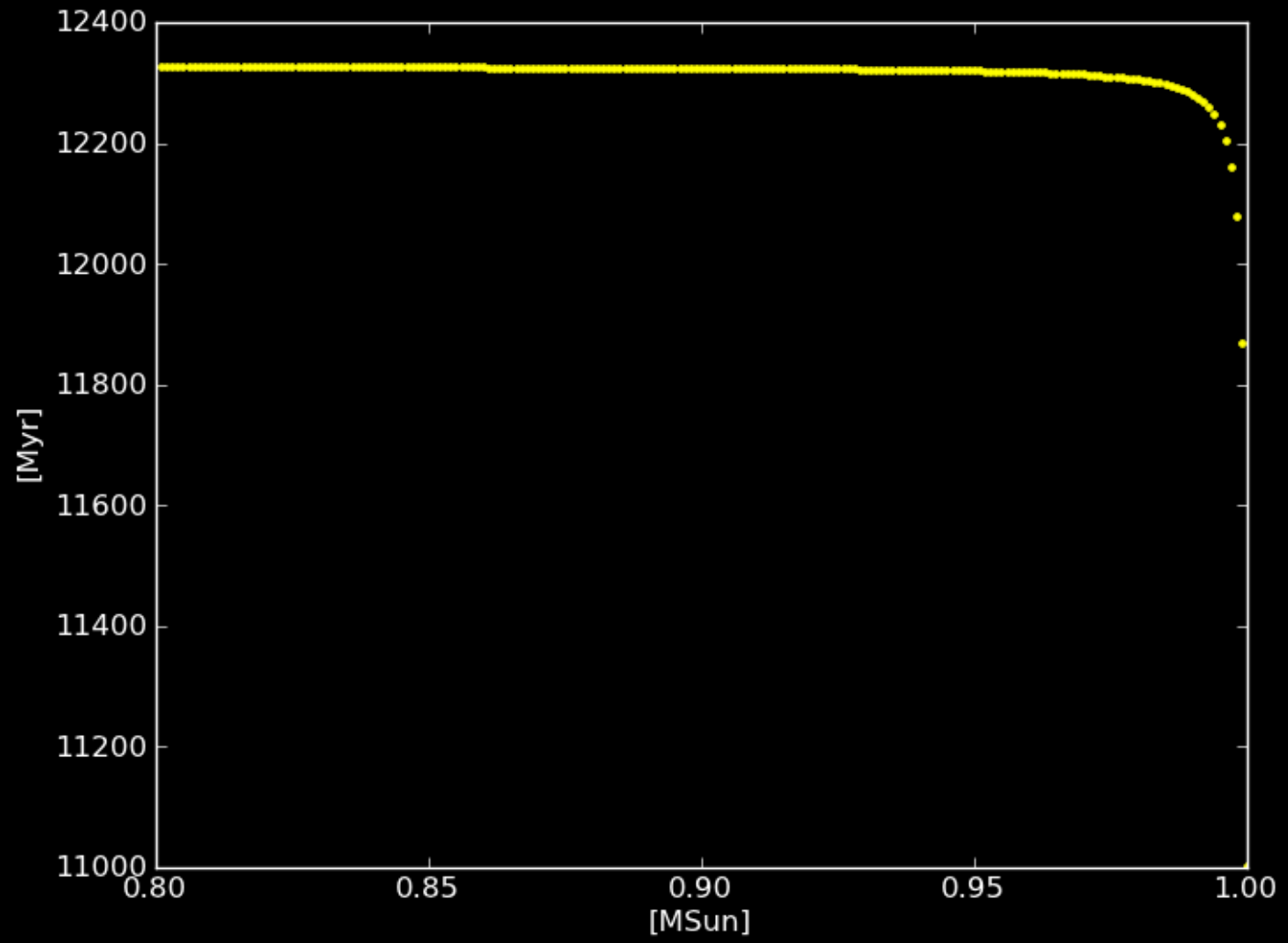
SE

```
import numpy
from amuse.community.sse.interface import SSE
from amuse.support.data import core
from amuse.support.units import units
from amuse.ext.solarsystem import Solarsystem
from amuse.plot import *

def plottillagb():
    sse = SSE()
    sun, planets = Solarsystem.new_solarsystem()
    sse.particles.add_particles(sun)
    sse.commit_particles()
    channel = sse.particles.new_channel_to(sun)
    channel.copy()

    timerange = units.Myr(numpy.arange(11000, 15000,10
    masses = [] | units.MSun

    for time in timerange:
        sse.evolve_model(time)
        channel.copy()
        masses.append(sse.particles[0].mass)

    sse.stop()
    plot(timerange, masses,'.')
    native_plot.show()

if __name__ in ("__main__", "__plot__"):
    plottillagb()
```

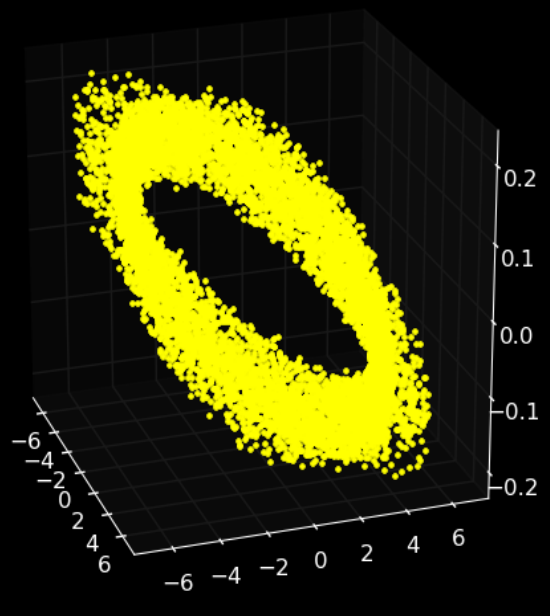


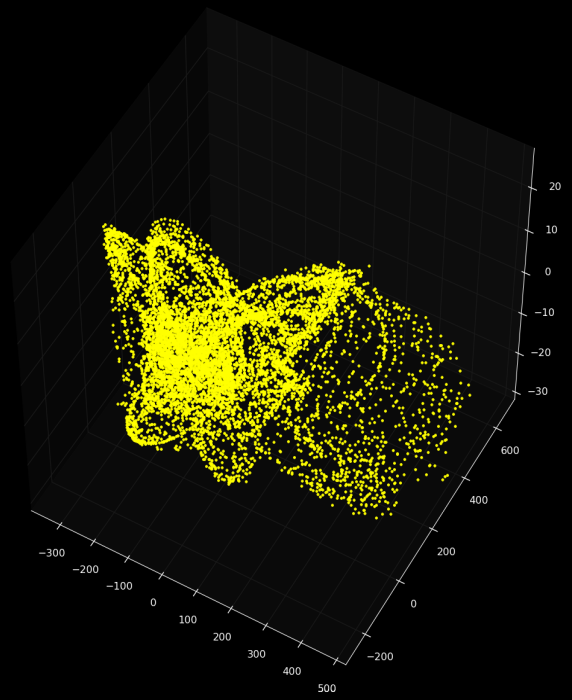
```
def plottillagb():
    sse = SSEWithMassEvolve()
    sse.commit_parameters()
    sun, planets = Solarsystem.new_solarsystem()
    sse.particles.add_particles(sun)
    sse.commit_particles()
    channel = sse.particles.new_channel_to(sun)
    channel.copy()

    massrange = units.MSun(numpy.arange(1, 0.8 ,-0.001))
    masses = []|units.MSun
    timerange = [] | units.Myr

    for mass in massrange:
        #sse.evolve_model(time)
        sse.evolve_mass(mass)
        timerange.append(sse.evolve_mass(mass))
        channel.copy()
        masses.append(sse.particles[0].mass)


    sse.stop()
    plot(massrange, timerange,'.')
    native_plot.show()

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

```
class SSEWithMassEvolve(SSE):
    def __init__(self, **options):
        SSE.__init__(self, convert_nbody = None,
**options)

    def evolve_mass(self, mass):
        timestep = 1.0 | units.Myr
        current_mass = self.particles[0].mass
        current_time = self.particles[0].age

        while current_mass >= mass:
            current_time += timestep
            self.evolve_model(current_time)
            current_mass = self.particles[0].mas
        return current_time
```


12400
12200
12000
11800
11600
11400
11200
11000
[Myr]
0.80
0.85
0.90
0.95
1.00
[MSun]

```
sun, planets = Solarsystem.new_solarsystem()
gd, se = setup_codes(sun, planets)

channelp = gd.orbiters.new_channel_to(planets)
channels = se.particles.new_channel_to(sun)

prev_mass = sun[0].mass

massrange = units.MSun(numpy.arange(0.9, 0.89999 ,-1e-9))
masses = []|units.MSun
timerange = [] | units.Myr

for i, mass in enumerate(massrange):
    time, dummymass = se.evolve_mass(mass)
    err = gd.evolve_model(gdtime)
    channelp.copy()
    planets.savepoint(time)
    channels.copy()
    gd.central_particle.mass = sun[0].mass

gd.stop()
se.stop()
```


0.2
0.1
0.0
−0.1
−0.2
−6
−4
−2
0
2
4
6
−6
−4
−2
0
2
4
6


20
10
0
−10
−20
−30
600
400
200
0
−200
−300
−200
−100
0
100
200
300
400
500