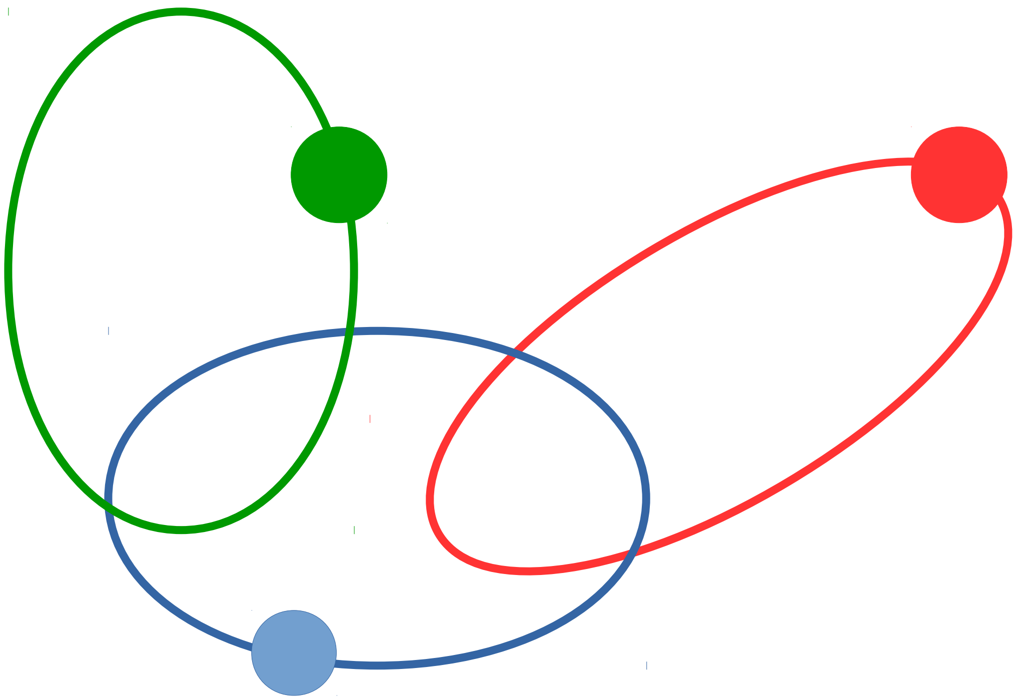
Workshop on Advanced Techniques for Scientific Programming and Management of Open Source Software Packages

Gravitation Project

Bellomo, Franco @fnbellomo
Aguena da Silva, Michel
Fogliatto, Ezequiel
Romero Abad, David

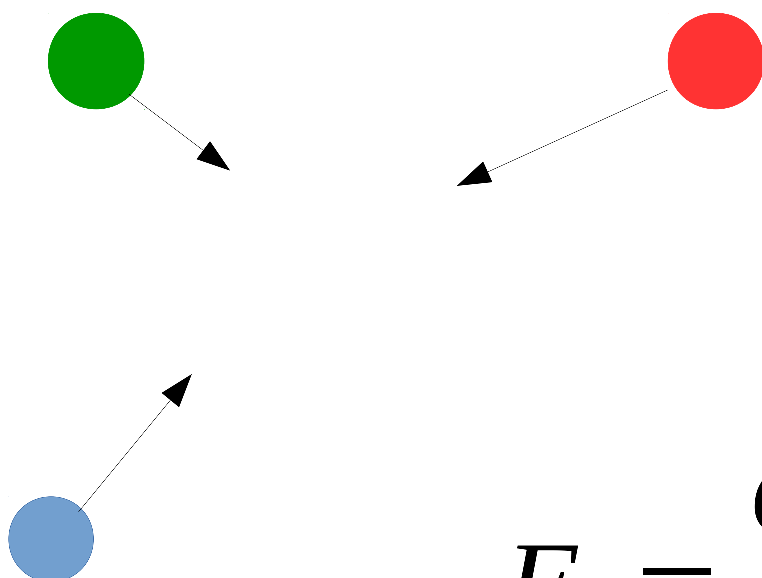
PROBLEM DESCRIPTION



MAIN TASK

Compute the movement of bodies under gravity forces **using collaborative techniques**

PROBLEM DESCRIPTION



MAIN TASK

Compute the movement of bodies under gravity forces **using collaborative techniques**

$$F_{ij} = \frac{G\, m_i m_j}{|\vec{x}_i - \vec{x}_j|^2} \frac{\vec{x}_j - \vec{x}_i}{\|\vec{x}_i - \vec{x}_j\|}$$

EQUATION DISCRETIZATION

$$\vec{F}_{ij} = m_i g_{ij} (\vec{x}_j - \vec{x}_i)$$

$$\ddot{\vec{X}}_i = \sum_j g_{ij} (\vec{x}_j - \vec{x}_i)$$

$$\dot{\vec{X}}_i = \vec{v}_i$$

$$\ddot{\vec{V}}_i = \sum_j g_{ij} (\vec{x}_j - \vec{x}_i)$$

$$\begin{bmatrix} X' \\ V' \end{bmatrix} = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{0} & \text{Identity} \\ \hline g & 0 & \\ 0 & g & 0 \end{array}\right] \begin{bmatrix} X \\ V \end{bmatrix}$$

$$\dot{\vec{\alpha}} = M \alpha$$

EQUATION DISCRETIZATION

Explicit Euler

$$\vec{\alpha}_{n+1} = \vec{\alpha}_n + \delta t \, M \, \vec{\alpha}_n$$

Crank-Nicholson

$$\vec{\alpha}_{n+1} = \vec{\alpha}_n + \frac{\delta t}{2} \left[M \, \vec{\alpha}_{n+1} + M \, \vec{\alpha}_n \right]$$

Runge – Kutta 4th order

$$\vec{\alpha}_{n+1} = \vec{\alpha}_n + \frac{1}{6} \left(K_1 + 2 K_2 + 2 K_3 + K_4 \right)$$

$$\begin{bmatrix} X' \\ V' \end{bmatrix} = \begin{bmatrix} 0 & \text{Identity} \\ \begin{matrix} g & 0 \\ 0 & g \end{matrix} & 0 \end{bmatrix} \begin{bmatrix} X \\ V \end{bmatrix}$$

$$\dot{\vec{\alpha}} = M \quad \alpha$$

Gravitation project

PYTHON + GIT + GITHUB

```
class Gravitation(object):
    """ This is the main gravitation wrapper"""
```

→ List of bodies
Time advancement

```
class Body(object):
    """ Base class for space objects"""
```

→ Mass, position, velocity

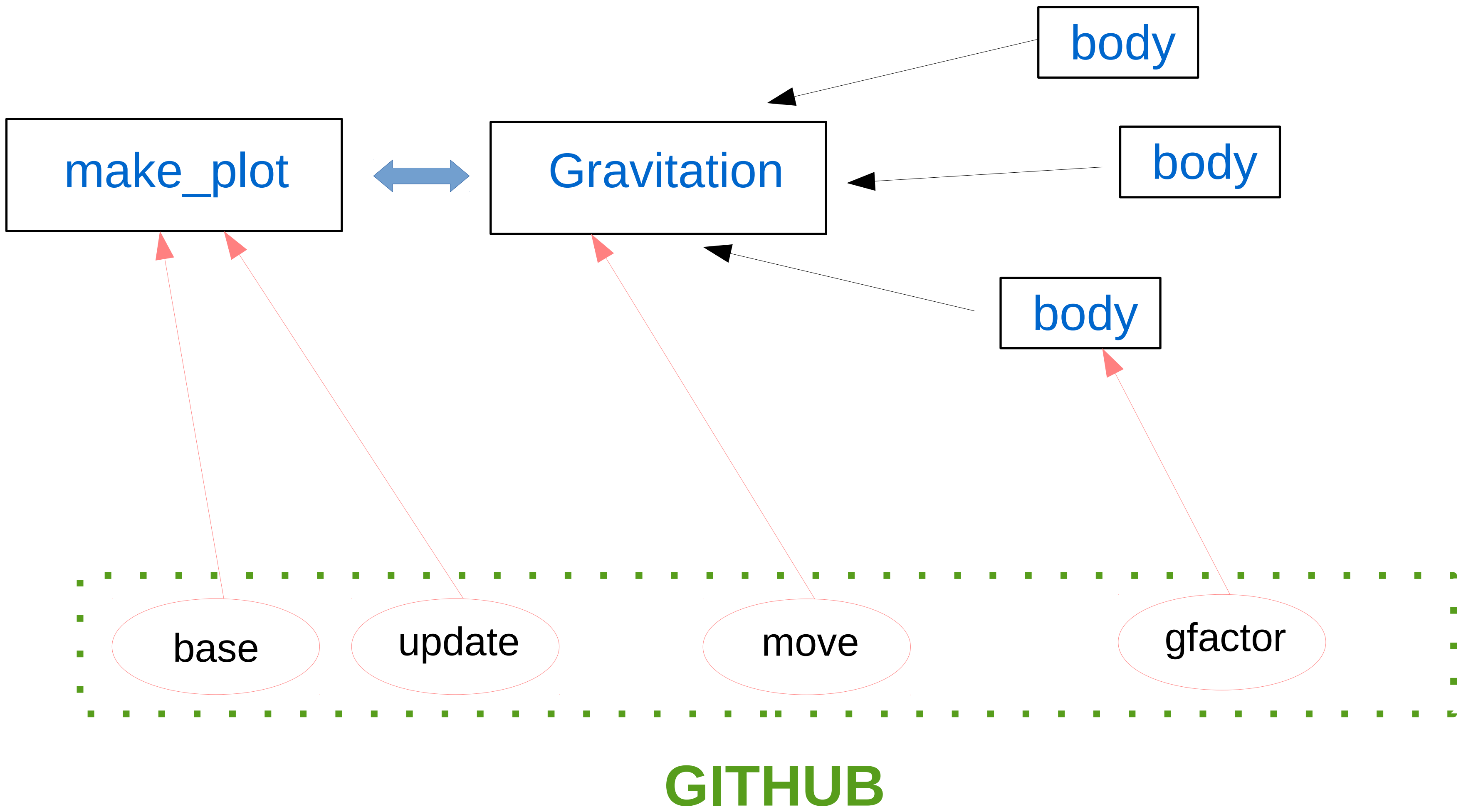
```
class make_plot(object):
    """Class designed for runtime plotting"""
```

→ Runtime plotting
Image and video
Multi-processing

```
def main():
    """Main function"""
```

→ usage: Main.py [-h] [--method METHOD] [--tstep TSTEP] [--file FILENAME] [--plot] [--profile] [--nsteps NSTEPS] [--config][--confile CONFIG_FILE]

PYTHON + GIT + GITHUB



Gravitation project

2015 ICTP-SAIFR School - Gravitaion Project — Edit

77 commits | 1 branch | 0 releases | 3 contributors

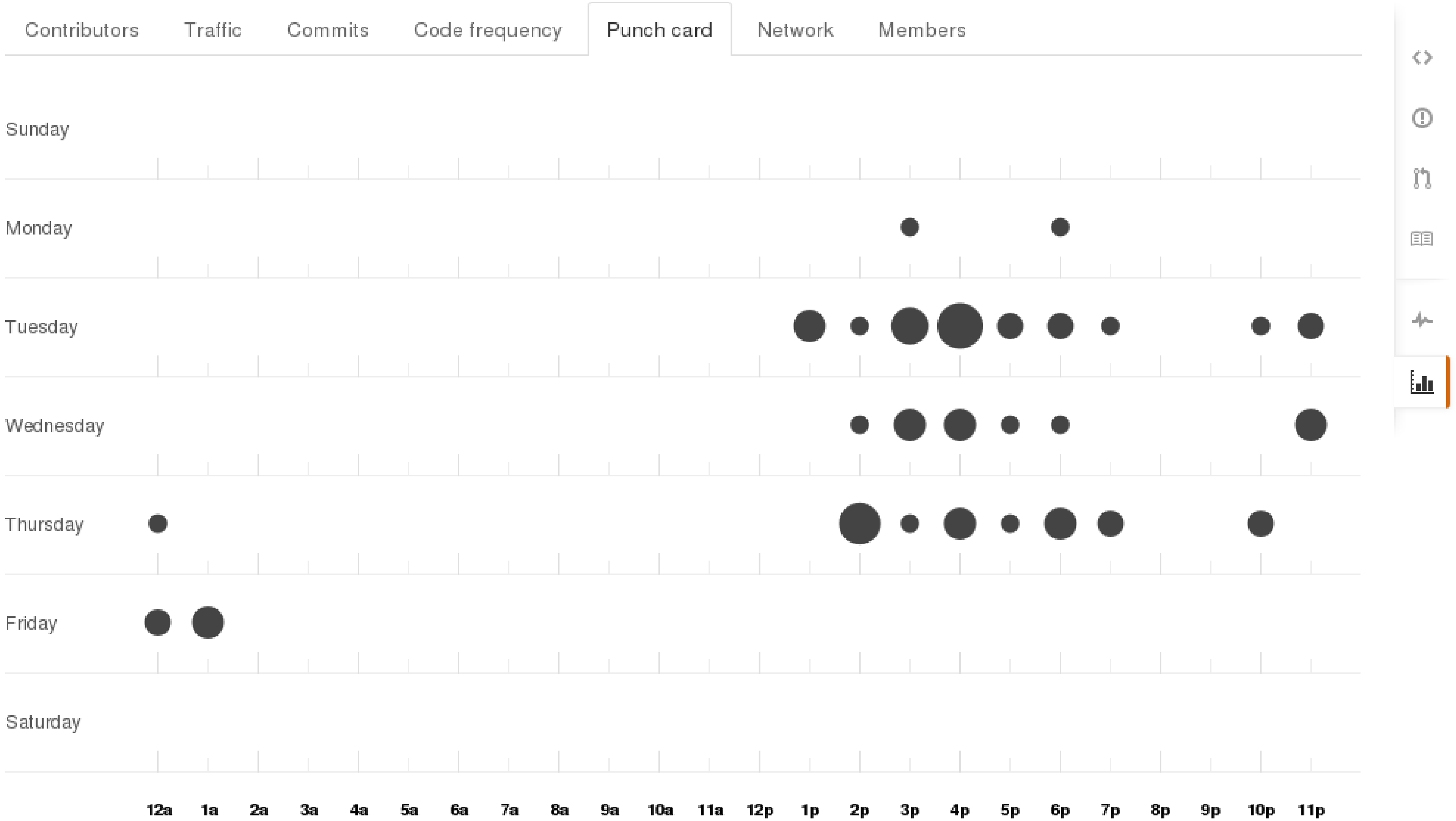
branch: master | GProject / +

Merge branch 'master' of https://github.com/fnbellomo/GProject

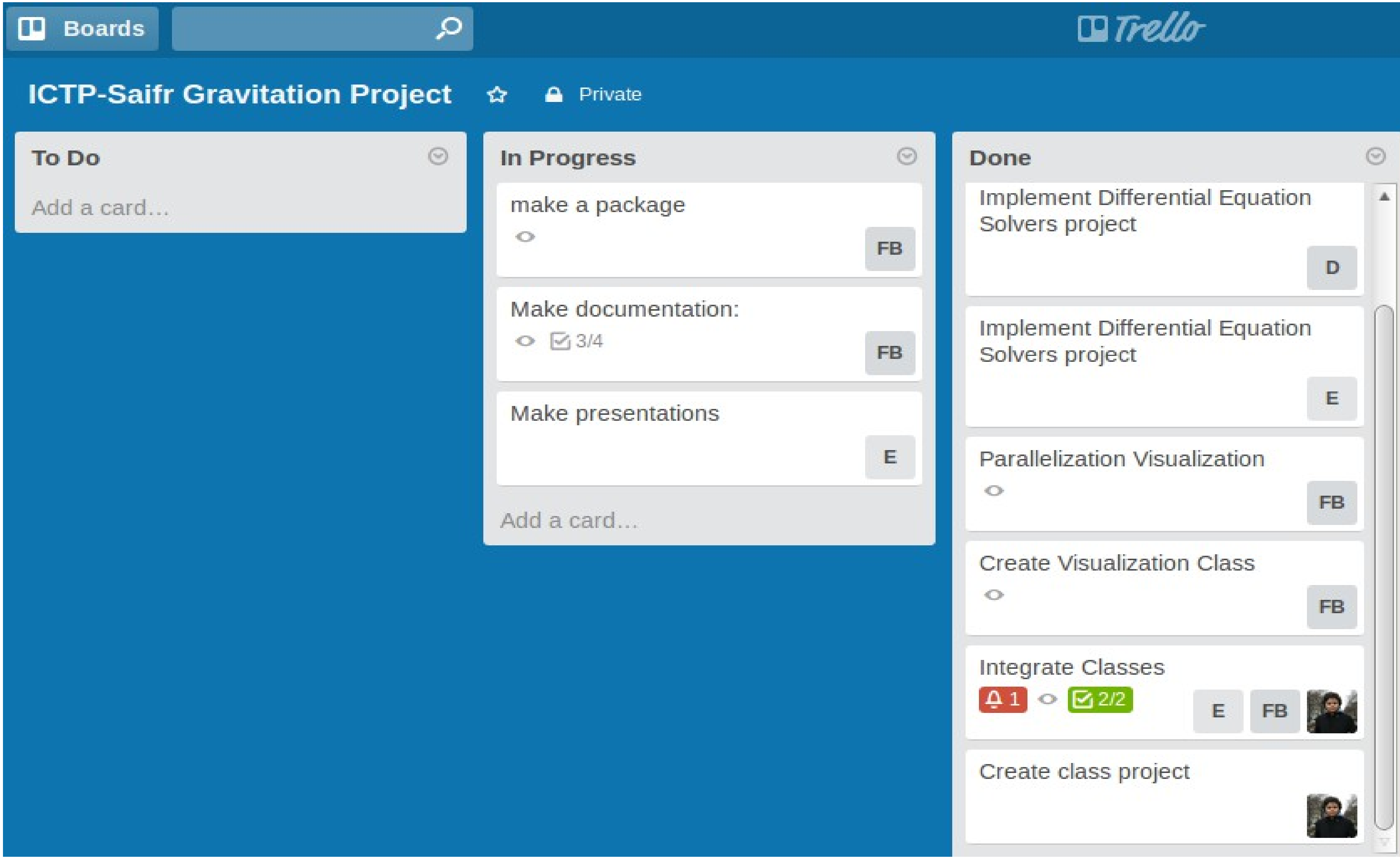
efogliatto authored 6 hours ago — latest commit 9e14d4b2e4

Gravitation	Write the doc of each method	7 hours ago
Slides	update slides. add pdf	6 hours ago
LICENSE	Initial commit	4 days ago
Main.py	add --mp to use multiprocessing	7 hours ago
README.md	Install process	6 hours ago
Results.gnumeric	Add slides	6 hours ago
config.py	data change	7 hours ago
ez_setup.py	Install process	6 hours ago
generate_people.py	add file that generates workshop members	12 hours ago
setup.py	Install process	6 hours ago

Gravitation project



Gravitation project

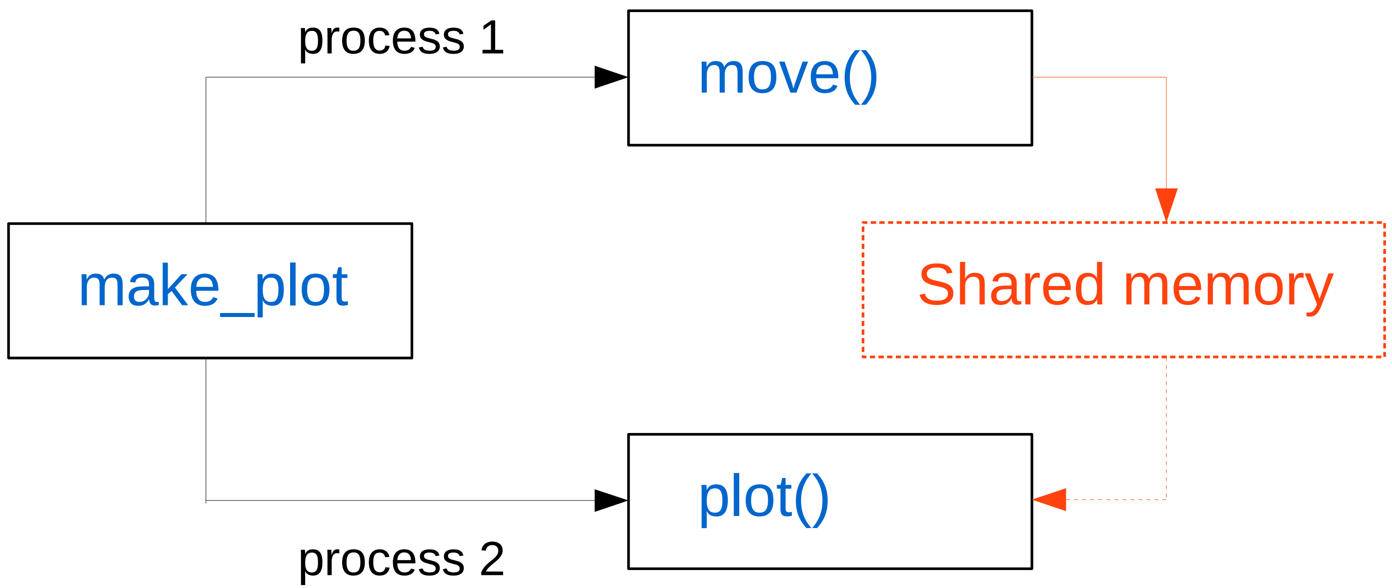


ICTP SAIFR

Workshop on Advanced Techniques for Scientific Programming and Management of Open Source Software Packages

make_plot

- First approach: sequential plotting
- Second approach: multi-processing



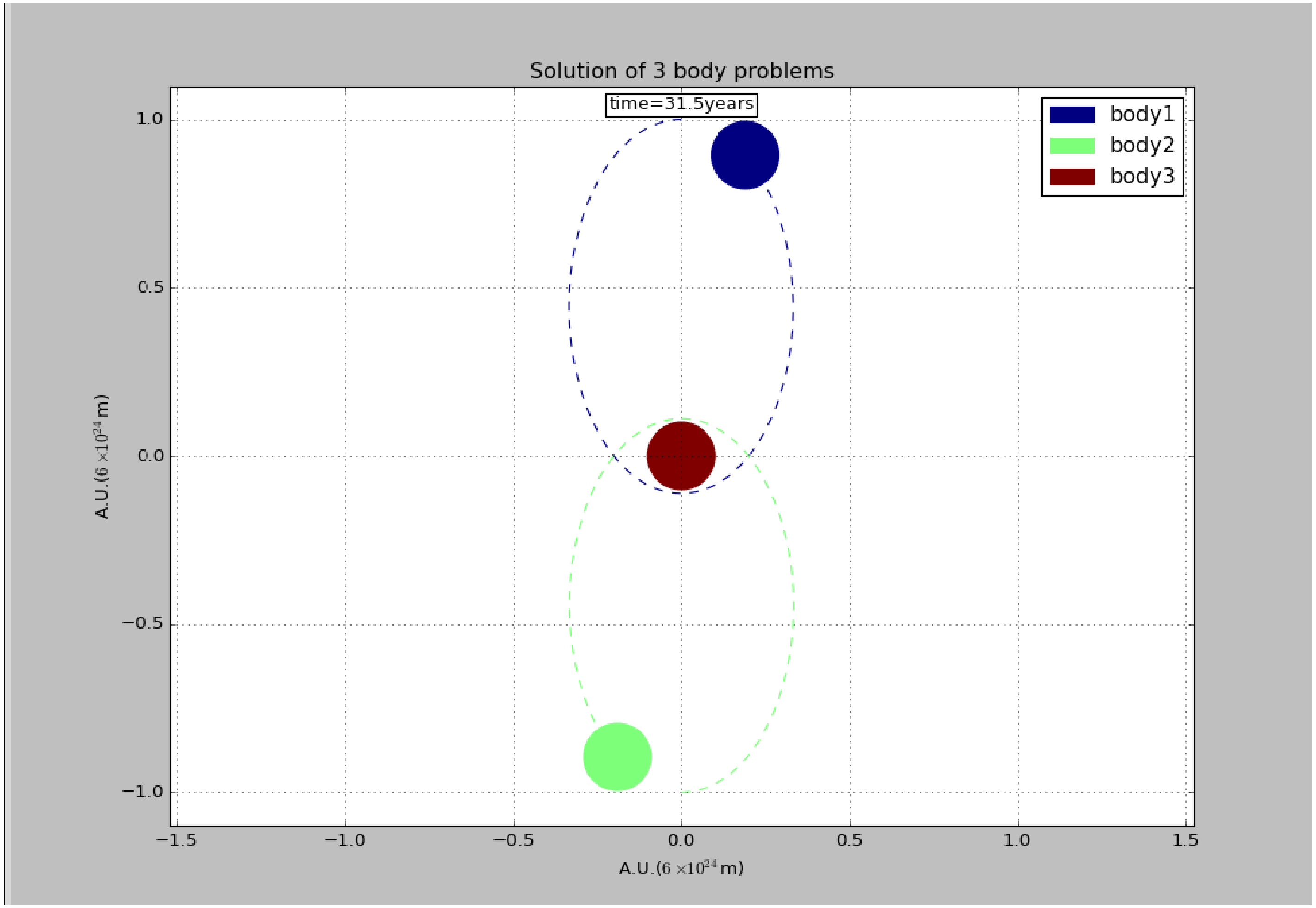
LATEST VERSION

A python program that integrates the equation of movement for an arbitrary number of bodies

Main features

- **Collaborative project**
- Command-line options. Reads data and options from file or during runtime
- Several numerical methods: Explicit Euler, Crank-Nicholson, Runge-Kutta4, **adaptive Runge-Kutta**
- Runtime plotting with multiprocessing

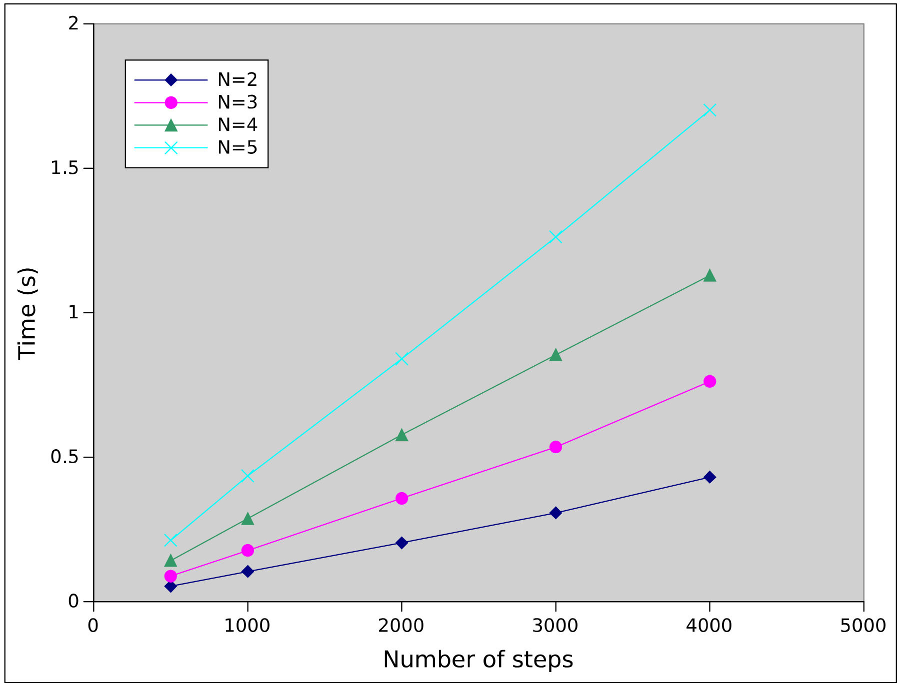
SOME RESULTS

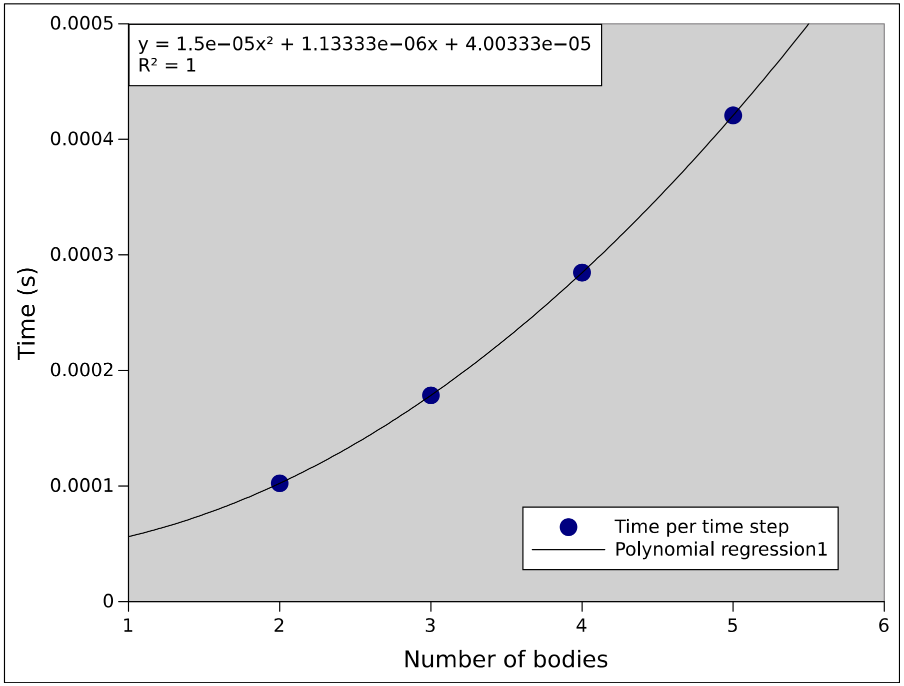


PROFILING

```
import cProfile, pstats

Pr = cProfile.Profile()
pr.enable()
pr.disable()
```





PROFILING

```
      99039 function calls in 0.535 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
 45000    0.223    0.000    0.223    0.000 Body.py:20(gfactor)
     3    0.000    0.000    0.000    0.000 Body.py:9(__init__)
  3000    0.261    0.000    0.531    0.000 Gravitation.py:107(move)
  3000    0.016    0.000    0.018    0.000 Gravitation.py:175(update)

     6    0.000    0.000    0.000    0.000 Gravitation.py:8(float_list)
  6003    0.001    0.000    0.001    0.000 {len}

     3    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
 12000    0.019    0.000    0.019    0.000 {numpy.core._dotblas.dot}
     3    0.000    0.000    0.000    0.000 {numpy.core.multiarray.zeros}
     1    0.000    0.000    0.000    0.000 {open}
     4    0.000    0.000    0.000    0.000 {print}
 30002    0.011    0.000    0.011    0.000 {range}
```

INSTALL

Dependences:

- Numpy
- Matplotlib

Installation:

$ git clone https://github.com/fnbellomo/GProject.git

$ cd GProyect

$ sudo python install setup.py

DOCUMENTATION (pydoc)

```
Help on module Body:

NAME
    Body

CLASSES
    builtins.object
        Body

    class Body(builtins.object)
     |  Base class for space bodies.
     |
     |  This class is responsible for creating objects that would be attracted in the same Gravitational object.
     |  The class contains specific information such as position, velocity and mass.
     |
     |  Methods defined here:
     |
     |  __init__(self, obj_id, obj_mass, obj_position, obj_velocity)
     |      Start a Body objects.
     |
     |      Parameters
     |      ----------
     |      obj_id : str
     |              Body name.
     |      obj_mass : str
     |              Body mass.
     |      obj_position : array_like
     |              Position in x and y. [x, y]
     |      obj_velocity : array_like
     |              Velocity in x and y. [V_x, V_y]
     |
```

ICTP SAIFR

UNIT TESTS

```
import unittest
import doctest

class TestProject(unittest.TestCase):
    “””Testing Body class”””
```

WHAT WE LEARNED?

- **Working in collaboration is not easy!**
- Implementation a good way to program
- Use of Control Version Software

TO DO

- Rewrite class design
- Split the program into a larger number of independent modules
- Optimize calculations
- Optimize communication between processes

Thanks!

Any questions?