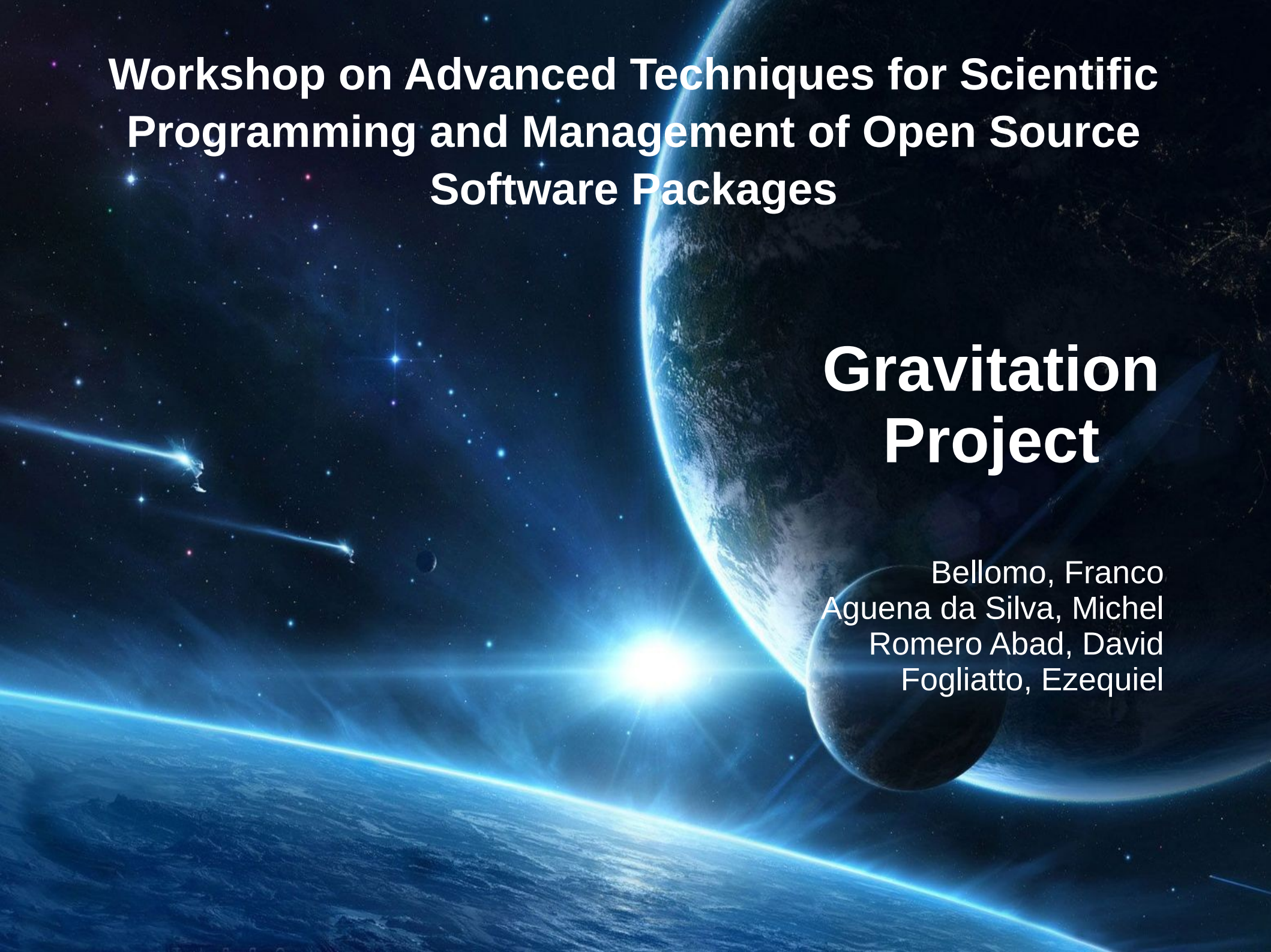
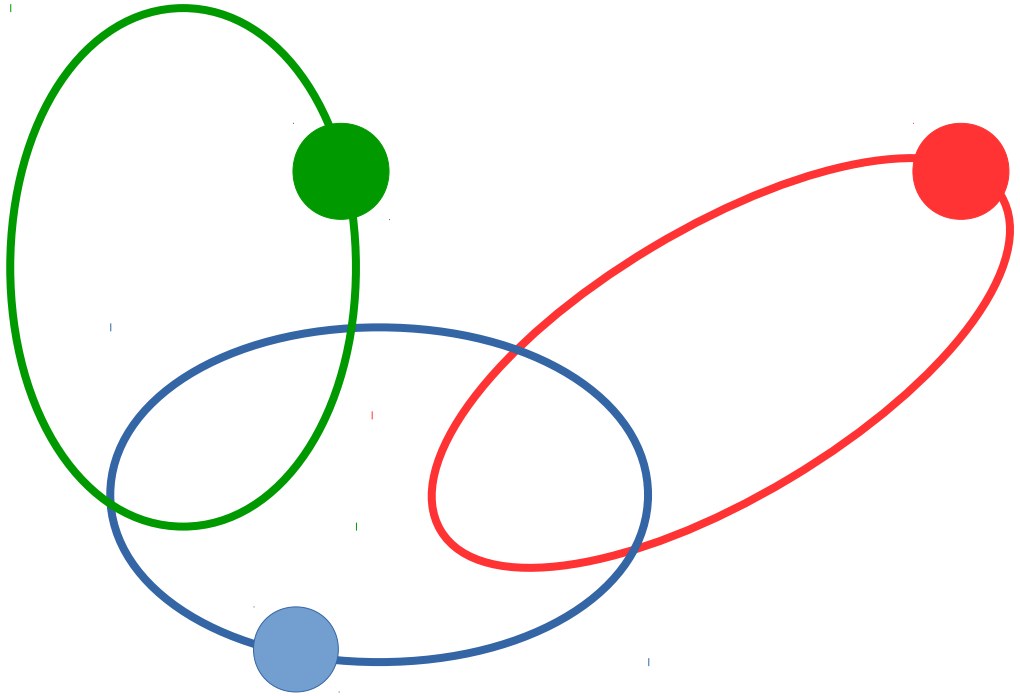


Workshop on Advanced Techniques for Scientific Programming and Management of Open Source Software Packages

Gravitation Project

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

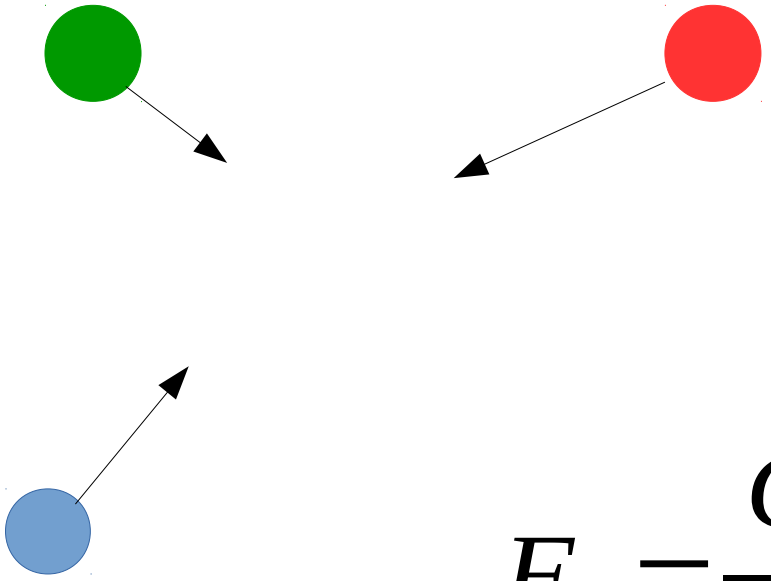
PROBLEM DESCRIPTION



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

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

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

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

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

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

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

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

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

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

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

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

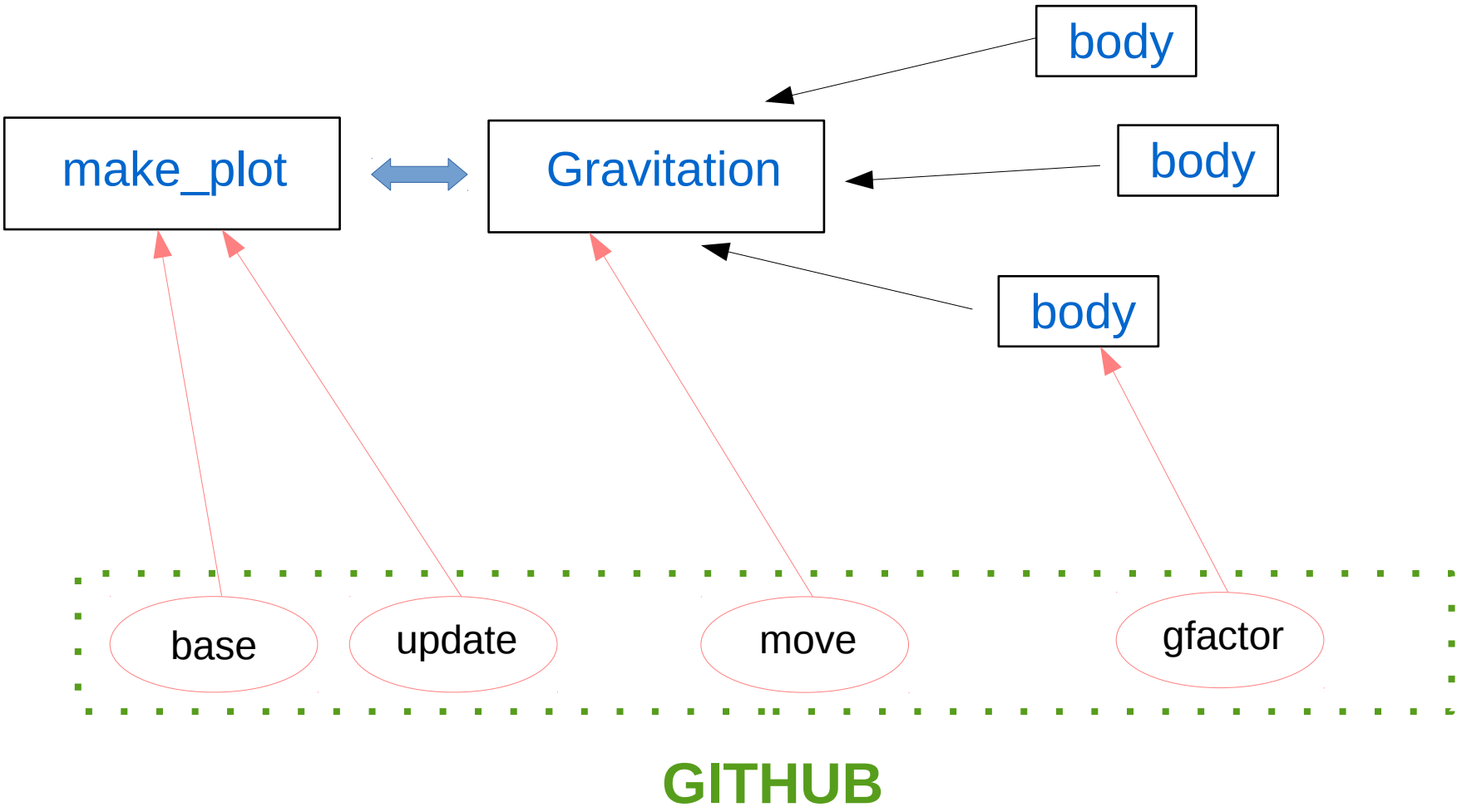
```
def main():
    """Main function"""
```

→

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

Gravitation project

PYTHON + GIT + GITHUB



Gravitation project

2015 ICTP-SAIFR School - Gravitaion Project

65 commits | **1** branch | **0** releases | **3** contributors

branch: **master** GProject / +

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

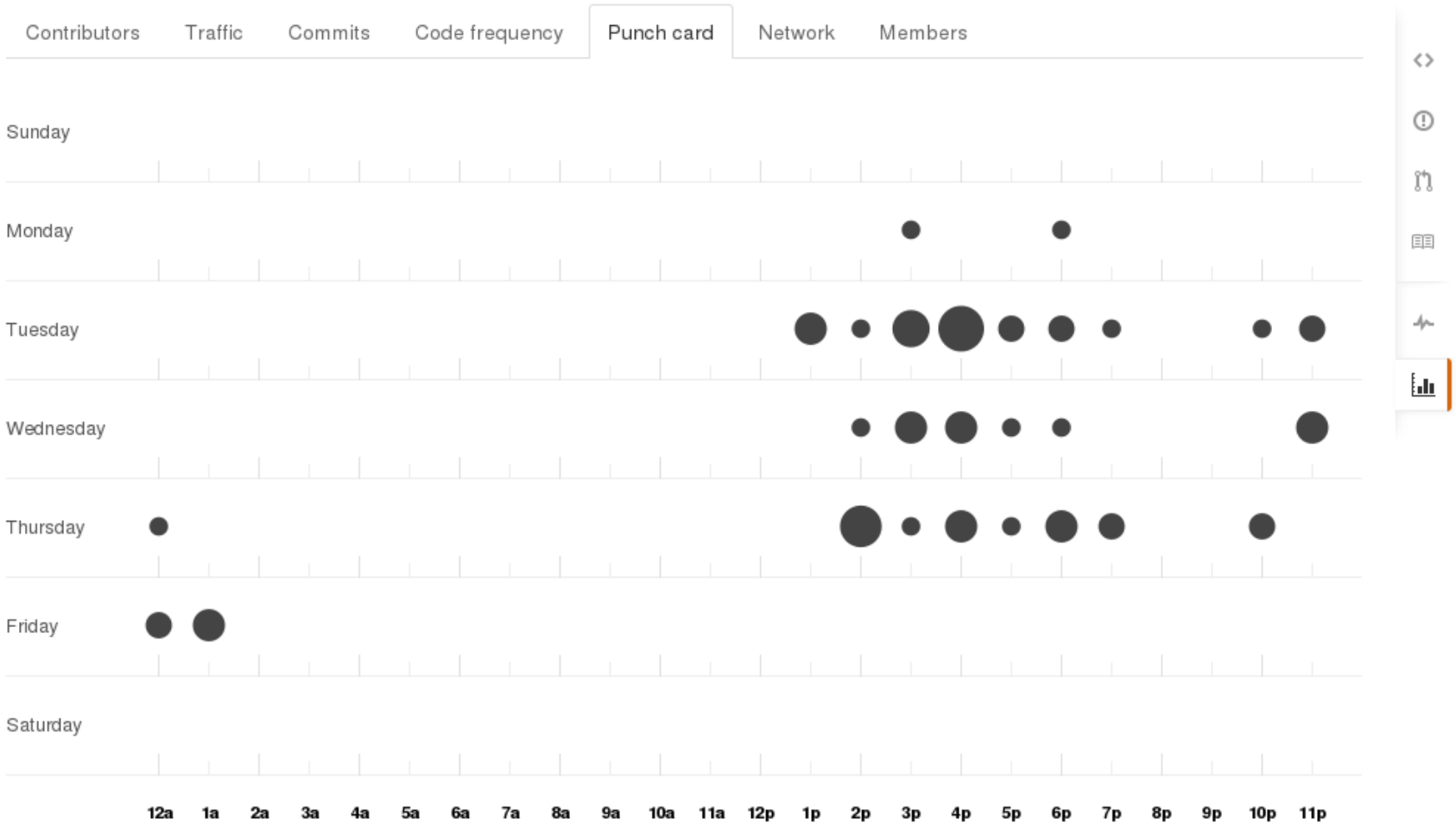
fnbellomo authored an hour ago — latest commit `019976cd35`

Gravitation	Merge branch 'master' of https://github.com/fnbellomo/GProject	an hour ago
LICENSE	Initial commit	3 days ago
Main.py	--mp option to make plot with multiprocessing	an hour ago
README.md	Initial commit	3 days ago
config.py	saving the plot made optional	7 hours ago
generate_people.py	add file that generates workshop members	4 hours ago
test_bodies.dat	Correction to update() in class gravitation. Remove RK.py	2 days ago

ICTP SAIFR

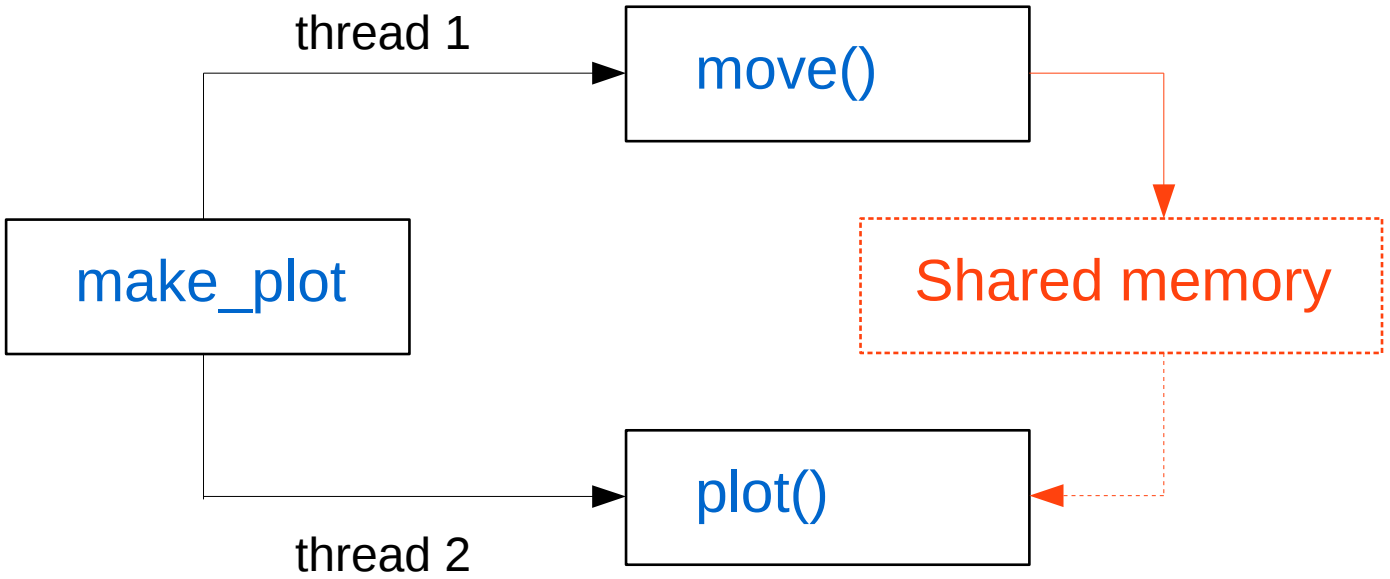
Workshop on Advanced Techniques for Scientific Programming and Management of Open Source Software Packages

Gravitation project



make_plot

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



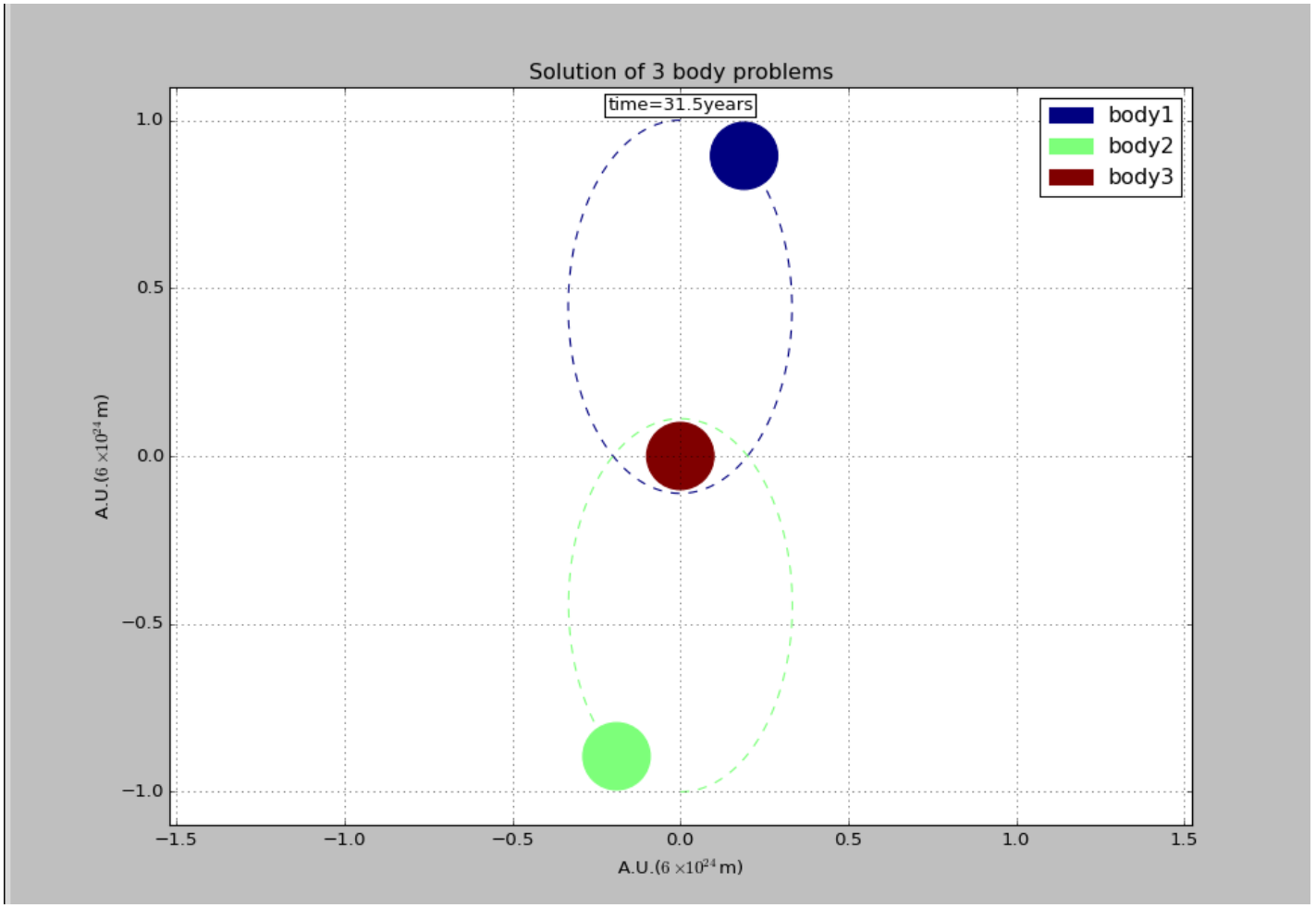
LATEST VERSION

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

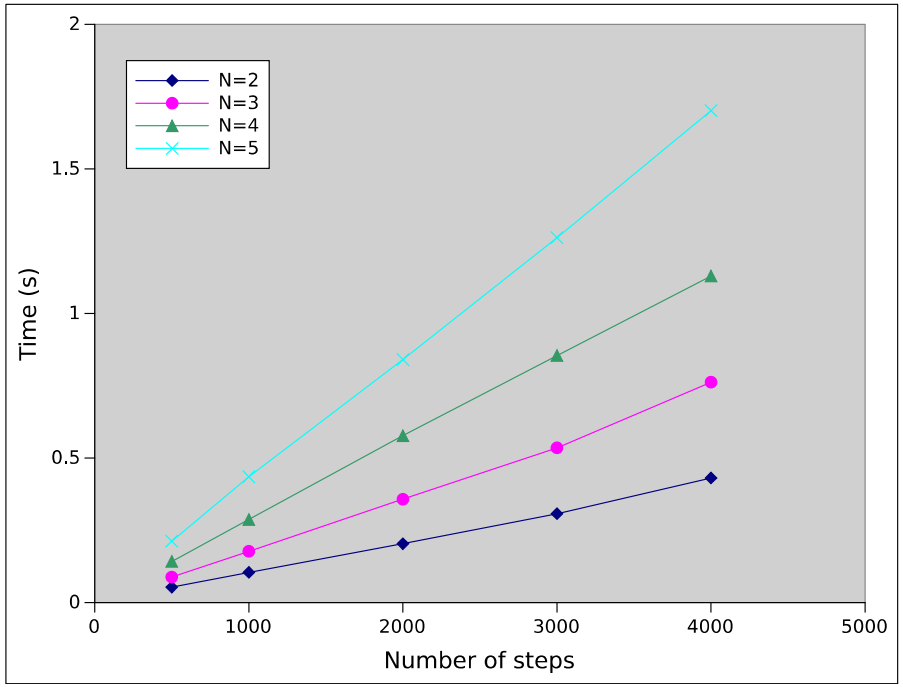
SOME RESULTS

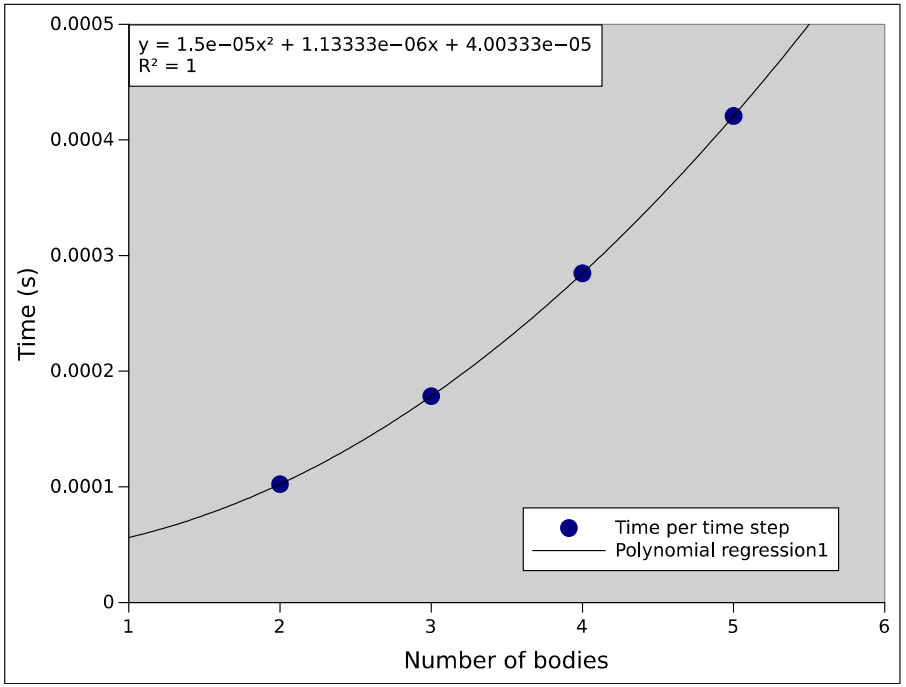


PROFILING

```
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}
```

DOCUMENTATION

UNIT TESTS

```
import unittest
import doctest

class TestProject(unittest.TestCase):
    """Testing Body class"""
```

Gravitation project

WHAT WE LEARNED?

- **Working in collaboration is not easy!**
- Implementation of good programming dferfgfv
- Use of Control Version Software

TO DO

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

Thanks!

Any questions?

Gravitation project