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
Gravitation project

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

\[ F_{ij} = \frac{G m_i m_j}{|\vec{x}_i - \vec{x}_j|^2} \frac{\vec{x}_j - \vec{x}_i}{|\vec{x}_i - \vec{x}_j|} \]
\[ \vec{F}_{ij} = m_i g_{ij} (\vec{x}_j - \vec{x}_i) \]

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

\[ \vec{V}_i = \sum_j g_{ij} (\vec{x}_j - \vec{x}_i) \]

**EQUATION DISCRETIZATION**

\[
\begin{bmatrix}
X' \\
V'
\end{bmatrix} = \begin{bmatrix}
0 & \text{Identity} \\
g & 0 \\
0 & g
\end{bmatrix} \begin{bmatrix}
X \\
V
\end{bmatrix}
\]

\[ \dot{\alpha} = M \alpha \]
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
\[ \vec{\alpha}_{n+1} = \vec{\alpha}_n + \frac{1}{6} \left( K_1 + 2 K_2 + 2 K_3 + K_4 \right) \]
\[ \dot{\vec{\alpha}} = \vec{\alpha} - \frac{1}{6} \left( K_1 + 2 K_2 + 2 K_3 + K_4 \right) \]
\[ \vec{\alpha} = \begin{bmatrix} X' \\ V' \end{bmatrix} = \begin{bmatrix} 0 & Identity & | & \end{bmatrix} \begin{bmatrix} X \\ V \end{bmatrix} \]

\[\begin{bmatrix} g & 0 \\ 0 & g \end{bmatrix} = \begin{bmatrix} K_1 + 2 K_2 + 2 K_3 + K_4 \end{bmatrix} \]

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

PYTHO$$N + GIT + GITHUB$$

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

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

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

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


List of bodies
Time advancement
Mass, position, velocity
Runtime plotting
Image and video
Multi-processing
Gravitation project

PYTHON + GIT + GITHUB

- make_plot
- Gravitation
- base
- update
- move
- gfactor

GITHUB: https://github.com/fnbellomo/GProject.git

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

2015 ICTP-SAIFR School - Gravitation 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

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

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

<table>
<thead>
<tr>
<th>Day</th>
<th>Contributors</th>
<th>Traffic</th>
<th>Commits</th>
<th>Code frequency</th>
<th>Punch card</th>
<th>Network</th>
<th>Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph](image)

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

ICTP-Saifr Gravitation Project

To Do
Add a card...

In Progress

- make a package
  - [x] 3/4

- Make documentation:
  - [x] 3/4

- Make presentations
  - [x] 2/2

Add a card...

Done

- Implement Differential Equation Solvers project
  - D

- Parallelization Visualization
  - [x] 1

- Create Visualization Class
  - [x] 2/2

- Integrate Classes
  - E

- Create class project
Gravitation project

**make_plot**

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

![Diagram]

- process 1
- move()
- Shared memory
- process 2
- plot()
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
- Plot saving for post-processing
- Implements Unit test
- Class documentation with **pydoc**
Workshop on Advanced Techniques for Scientific Programming and Management of Open Source Software Packages

Gravitation project

SOME RESULTS
**PROFILING**

```python
import cProfile, pstats
Pr = cProfile.Profile()
pr.enable()
pr.disable()
```

![Graph 1: Time vs. Number of steps for different N values.](image1)

![Graph 2: Time vs. Number of bodies with polynomial regression.](image2)
# Gravitation project

## PROFILING

99039 function calls in 0.535 seconds

Ordered by: standard name

<table>
<thead>
<tr>
<th>ncalls</th>
<th>tottime</th>
<th>percall</th>
<th>cumtime</th>
<th>percall</th>
<th>filename:lineno(function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>45000</td>
<td>0.223</td>
<td>0.000</td>
<td>0.223</td>
<td>0.000</td>
<td>Body.py:20(gfactor)</td>
</tr>
<tr>
<td>3</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>Body.py:9(<strong>init</strong>)</td>
</tr>
<tr>
<td>3000</td>
<td>0.261</td>
<td>0.000</td>
<td>0.531</td>
<td>0.000</td>
<td>Gravitation.py:107(move)</td>
</tr>
<tr>
<td>3000</td>
<td>0.016</td>
<td>0.000</td>
<td>0.018</td>
<td>0.000</td>
<td>Gravitation.py:175(update)</td>
</tr>
<tr>
<td>6</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>Gravitation.py:8(float_list)</td>
</tr>
<tr>
<td>6003</td>
<td>0.001</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
<td>{len}</td>
</tr>
<tr>
<td>3</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>{method 'split' of 'str' objects}</td>
</tr>
<tr>
<td>12000</td>
<td>0.019</td>
<td>0.000</td>
<td>0.019</td>
<td>0.000</td>
<td>{numpy.core._dotblas.dot}</td>
</tr>
<tr>
<td>3</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>{numpy.core.multiarray.zeros}</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>{open}</td>
</tr>
<tr>
<td>4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>{print}</td>
</tr>
<tr>
<td>30002</td>
<td>0.011</td>
<td>0.000</td>
<td>0.011</td>
<td>0.000</td>
<td>{range}</td>
</tr>
</tbody>
</table>
Gravitation project

INSTALL

**Dependences:**

- Numpy
- Matplotlib

**Installation:**

```bash
$ git clone https://github.com/fnbellomo/GProject.git
$ cd GProyect
$ sudo python install setup.py
```
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]
    ```
Gravitation project

WHAT WE LEARNED?

- Working in collaboration is not easy!
- Implementation of better programming
- 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?