Visual Relativity using **Einstein**Py

Mentors:

- Shreyas Bapat

Ratin Kumar
Email: ratin.kumar.2k@gmail.com
Phone: +91-9896225424

Index:

Introduction

Personal Information

Full Name	Ratin Kumar
Institute	1st Year B.Tech Student Computer Engineering National Institute of Technology Kurukshetra
Email	ratin.kumar.2k@gmail.com ratin_11822004@nitkkr.ac.in
Phone	+91-9896225424
Blog	https://medium.com/@ratin.kumar.2k
Github	https://github.com/DumbMachine
Riot Chat	@dumbmachine:matrix.org
Skype	live:ratin.kumar.2k
IRC Nick	DumbMachine (Freenode)
Timezone	Indian Standard Time (GMT +0530)
Address	317, Abhimanyu Bhavan Hostel-1, NIT campus, National Institute of Technology Kurukshetra Kurukshetra, Haryana, India 136119

About Me

I am Ratin Kumar, a 1st-year undergraduate Computer Engineering student at National Institute of Technology Kurukshetra. I am experienced in using Python and Javascript for creating applications as well as for hacky scripts to get things done. I really enjoy coding all the time, and as a result, I have numerous small projects on Github. It's my passion to help the world of science by means of code. I have always believed that “coders are the magicians of the 21st century”. Since childhood, I have loved the field of Science, especially space and quantum world.

I have created multiple web applications making use of D3.js and Plotly which made me realize its potential and simplicity. KagglePlotter is my WIP library for making initial preprocessing and analysis of Kaggle Datasets easy. It employs the use of multiple libraries like Numpy. Scipy, Sklearn to preprocess data, find features and their similarities/importance and then Visualize the data with the help of the Plotly for interactive and beautiful Plots. Apart from the above, I have many more small projects which have helped me hone my skills in python and programming practices in general.

Apart from coding, I have always been interested in the world of space, quantum mechanics and the fields with a mix of Computer Science and Biology. A few years back, I took part in my school's TEDx talk about the Revolution Machine Learning will bring when combined with Biology.

Open Source

I am part of many organizations on Github and have made multiple minor and a few major contributions. I love the spirit of Open Source, promoting universal access to code, and thus have become a part of this amazing community. Being a part of the Open Source group at my College, I have organized workshops on GIT and using Machine Learning for Image Captioning and continually help colleagues.

Skills

- Good knowledge of using Python and some knowledge of the internals of Python.
- Great knowledge of Visualization and related libraries like Matplotlib, Plotly.
- Good background in Mathematics; Calculus and Linear Algebra.
- High School level understanding of General Relativity.

Preparation I have done:

- Benchmarked different options to find the best.
- Created DEMO's to understand the problem better and make prototype implementations for the tasks.
- Read the documentation of **EinsteinPy** to understand the flow.

Why SOCIS with EinsteinPy?

I have been always passionate about the projects which link Sciences with Programming, which is surely the main inspiration for me to work with **EinsteinPy**. The idea of getting the chance to help Scientist and Researcher, who are influencing the future, really excites me and also aligns with the ambition to help humanity with the help of Science.

How much time will I be able to contribute to this project?

I will be working full time for the entire duration of the project.

Other commitments during summer

I have my end semester exams from May 3 to Mar 24 during which I'll be a little busy, other than that I do not have any commitments for summer. As the coding period starts on the 27th, I would be ready with my fullest then.

Preferred medium for communication

I am perfectly fine with IRC, Email, Skype or any other medium of communication. My preferred language for communication is English.

Synopsis:

EinsteinPy is an open source pure Python package dedicated to problems arising in General Relativity and relativistic physics, such as geodesics calculation for vacuum solutions for Einstein's field equations, calculation of various quantities in these geometries like Schwarzschild Radius and the event horizon. The library aims to solve Einstein's field equations for arbitrarily complicated matter distribution as one of the main goals. It provides for Static and Animated Visualizations of Perihelion, Event Horizon, Ergosphere and many more.
The project aims to introduce an interactive and dynamic plotting interface based on **EinsteinPy** & debug existing plotting functions. This also includes the development of a web interface to show results.

Comparison b/w the Libraries:

- Bokeh:
    - Documentation is not as good as that of Plotly.
    - Code tends to be harder to write.

- PyGal:
    - Interactive but returns, SVG.
    - If the image is saved as PNG, then It loses the Interactivity.
    - Displaying is an issue.

- Matplotlib:
    - Very good and in-depth Documentation.
    - Go-to Solution for Static plots.
    - Lacks interactive, of the scale of Plotly.
    - Animations are not as smooth as that of Plotly.

- Plotly:
    - Beautiful and Interactive React Plots.
    - Easier to understand and code.
    - Code tends to be smaller.

- Dash, web-framework for Plotly, supports user-defined React components.

Benchmark Matplotlib

In [18]:
```
%%timeit
plt.figure(figsize=(figwidth/dpi, figheight/dpi))
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.savefig('benchmarking/matplotlib_fig1.' + format)
plt.close()
```

```
216 ms ± 2.41 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Benchmark plotly.py 3.2.0

In [19]:
```
# Warmstart server for benchmarking (done automatically on first use)
pio.orca.ensure_orca_server()
```

In [20]:
```
%%timeit
fig = go.Figure(data=[
    dict(type='scatter', x=x, y=y, marker={'color': colors,
                                           'opacity': 0.5,
                                           'size': sz*30,
                                           'colorscale': 'Viridis',
                                           'line': {'width': 0},
                                          }, mode='markers')],
                layout={'width': figwidth, 'height': figheight,
                        'margin': {'l': 40, 'r': 40, 't': 40, 'b': 40}})
pio.write_image(fig, 'benchmarking/plotly_fig1.' + format)
```

```
211 ms ± 16.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Benchmark Bokeh

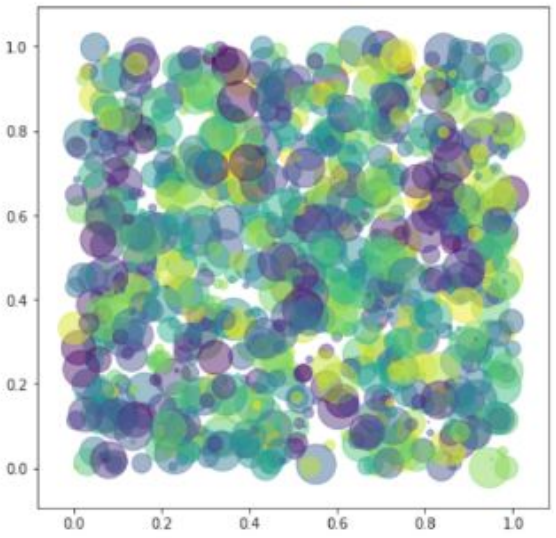
In [6]:
```
%%timeit
p = bkplt.figure(toolbar_location=None, plot_width=figwidth, plot_height=figheight)

p.scatter(x, y,
          size=sz*30,
          fill_color=bkcolors,
          fill_alpha=0.5,
          line_color=None)

bkio.export_png(p, filename="benchmarking/bokeh_fig1.png")
```

```
1.74 s ± 4.62 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Show matplotlib

```
In [7]: Image(filename='benchmarking/matplotlib_fig1.png')
```

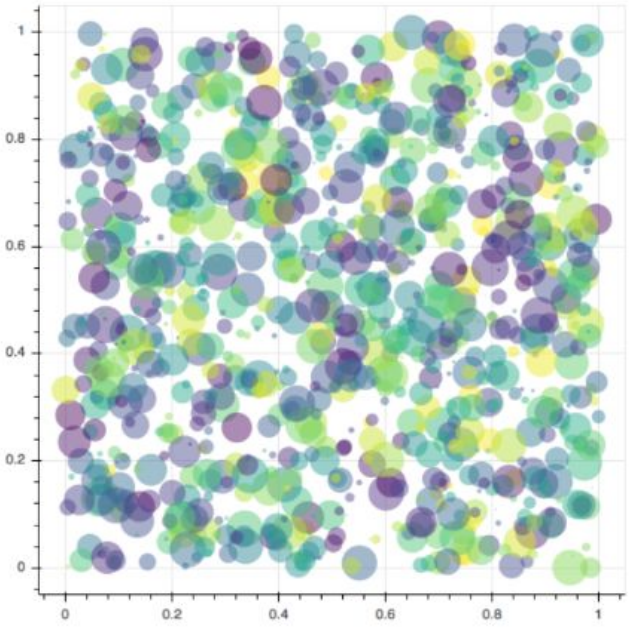
Out[7]:



Show bokeh

```
In [9]: Image(filename='benchmarking/bokeh_fig1.png')
```

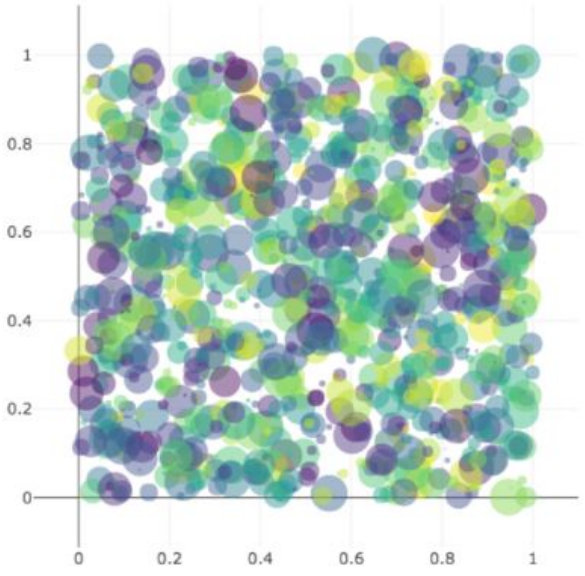
Out[9]:



Show plotly.py

```
In [8]: Image(filename='benchmarking/plotly_fig1.png')
```

Out[8]:



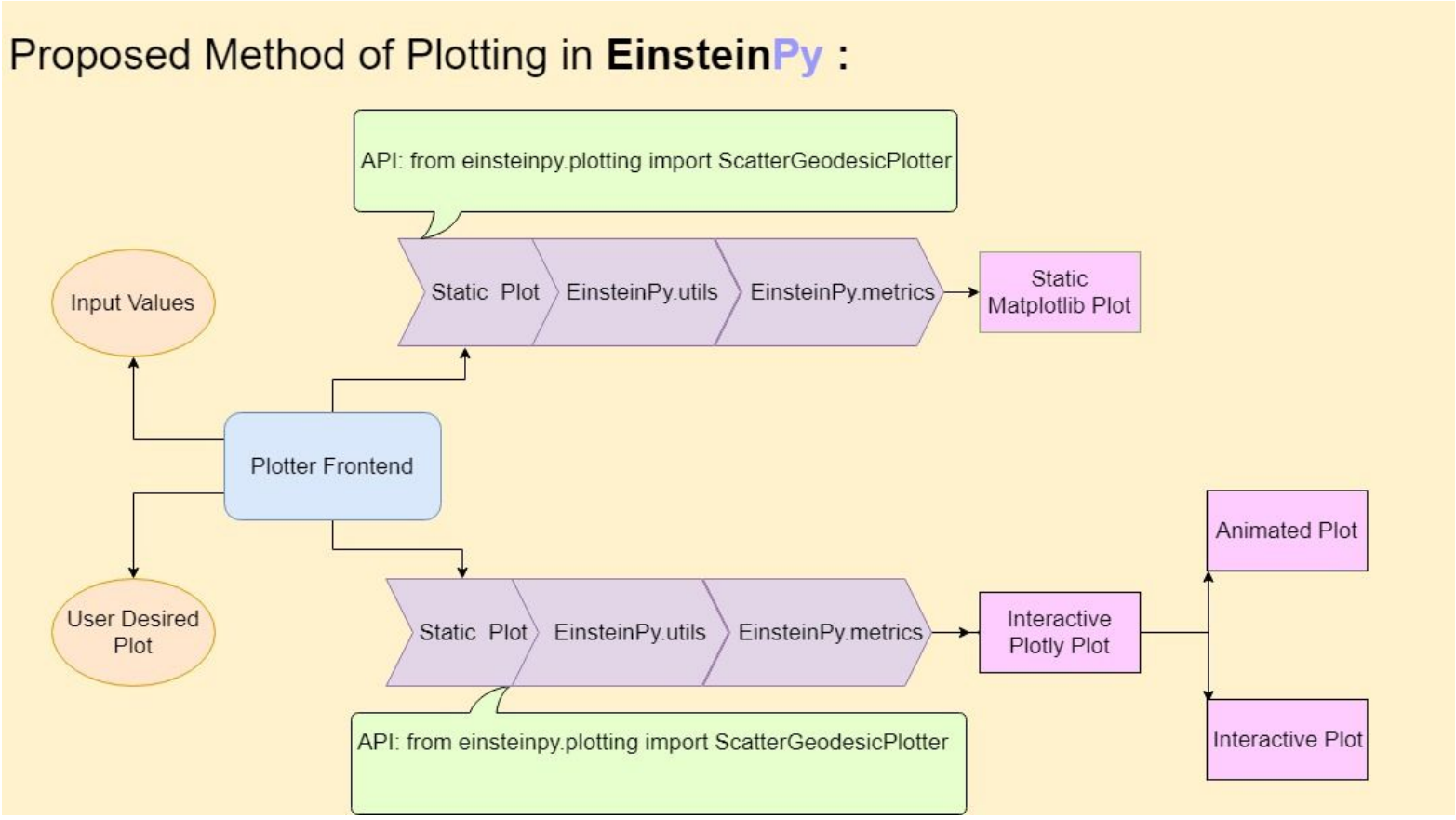
Speed Comparison (Static vs Interactive):

Time taken by Matplotlib to plot a Static Graph with nearly 7000 Scatter Points:

```
In [17]:
st = time.time()
# %matplotlib notebook
obj = ScatterGeodesicPlotter(M)
obj.plot(sph_obj, 0.002, 5e-8)
obj.show()
fl = time.time()-st
print(fl)
OUTPUT:
============================================================
2.818711519241333
```

Time taken by Plotly to plot an Interactive Graph with nearly 7000 Scatter Points:

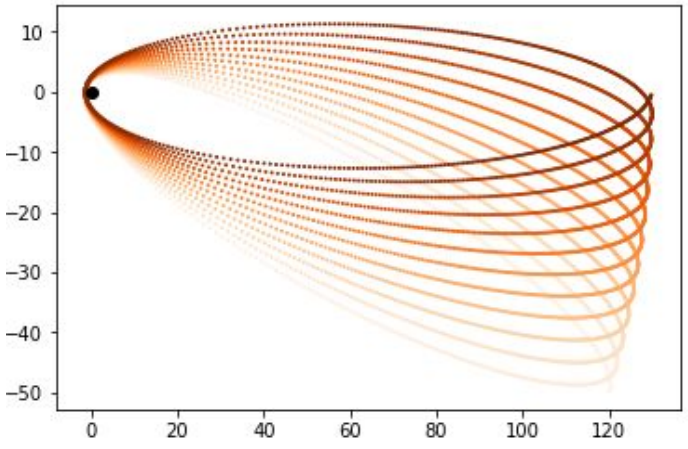
```
In [19]:
st = time.time()
obj = InteractiveScatterGeodesicPlotter(M)
obj.plot(sph_obj, 0.002, 5e-8)
obj.show()
fl = time.time() - st
print(fl)
OUTPUT:
============================================================
3.104757785797119
```



Users choice for the Plot will be exercised in the following manner:

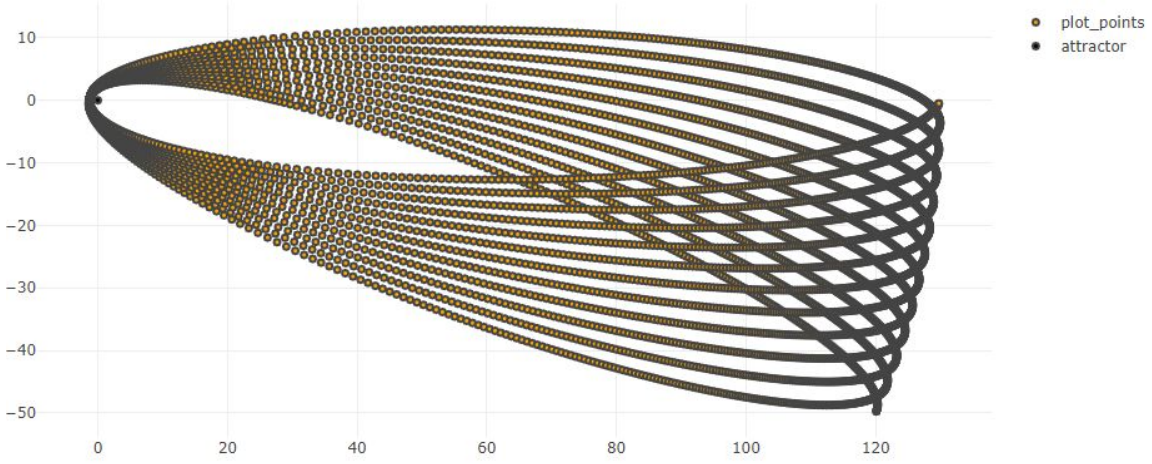
- **For using the Matplotlib Plots:**

```
M = 6e24 * u.kg
sph_obj = SphericalDifferential(130*u.m, np.pi/2*u.rad, -np.pi/8*u.rad,
                                0*u.m/u.s, 0*u.rad/u.s, 1900*u.rad/u.s)

obj = ScatterGeodesicPlotter(M)
obj.plot(sph_obj, 0.002, 5e-8, interval=5)
obj.show()
```



- **For using the Plotly Plots:**

```
M = 6e24 * u.kg
sph_obj = SphericalDifferential(130*u.m, np.pi/2*u.rad, -np.pi/8*u.rad,
                                0*u.m/u.s, 0*u.rad/u.s, 1900*u.rad/u.s)
obj = InteractiveScatterGeodesicPlotter(M)
obj.plot(sph_obj, 0.002, 5e-8)
obj.show()
```



Benefits to Community

After this project, the users of **EinsteinPy** enjoy the benefits of Interactive and Animated Visualizations through Plotly. They will still have the option to render Static charts with the help of Matplotlib.
NOT SURE Users will also have access to the Dash Web App for easy Visualization of Data over the Web.

Why I suit the Task

- Very good experience with Python.
- Experience with Plotly, Matplotlib, and related packages.
- Good experience with Open Source related work-flow.
- Strong knowledge of Visualization.

Project Goals

Objectives

- Design and Prototype the interface/api for Plotting.
- Make the necessary changes to the Metric/Utils sub-modules to support the Plotting.
- Code the interface for Interactive plotting with Plotly.
- Add tests for the new functions added.
- Develop example notebooks that demonstrate the basic features of the library.

Tasks:

Task 0: Design and Prototype the interface/api for Plotting.

- Design the api endpoints for the Interactive Plots.
- Design the api endpoints for the Animated Plots.
- Prototype the functions for Plotting.

Task 1: Make the necessary changes to the Metric/Utils sub-modules to support the Plotting.

- **Discussion required for this section**

Task 2: Code the interface for Interactive plotting with Plotly.

- Write the code for Interactive Plotly plots.
- Write the code for Animated Plotly plots.
- ** If DASH:** Develop the Dash Web App for **EinsteinPy**.

Task 3: Develop example notebooks that demonstrate the basic features of the library.

- Write and Develop upon the Binder examples to showcase the new plotting functions.
- Showcase the use of Dash Web App for Visualizing Data.

Task 4: Add tests for the new functions added.

- Add Unit tests for the new code added.
- Add Integrations tests for the new code added.

- Write/Finish Documentation for the functions added and code changed.

Timeline

Duration	Task
May 4, 2019	**Deadline for submitting Project Proposal**
May 16 - 31	**Community Bonding Period** ● Get Involved with **EinsteinPy** community. ● Know more about mentors such as their timezone, preferred medium of communication, etc. ● Learn about other projects at **EinsteinPy**. ● Get acquainted with various tools used at AWAKE. ● Discuss the basic functionalities required. ● Create a list of all the functions required from the interactive plotter.
June 16 - June 27	**Begin Task 0:** ● Design and Prototype the Plotting Class/Functions. ● Inculcate all the required features in the Plotting Class.
June 30 - July 21	**Begin Task 1:** ● Start development of the plotting tool. ● Test the made tool on real data. ● Write tests for the aforementioned to ensure the integrity. ● Discuss and tweaks to get the required results. ● Continue to next task, after discussion with mentors.
July 21 - 29	**Begin Task 2:** ● **Will be added soon.**

July 29 - 31	**Time for unexpected delay** ● Finish documentation for task 1 and 2. ● Fix bugs and provide patches for them.
August 1 - 28	**Begin Task 3:** ● Generate examples for using the new Plotting interface. ● Develop the Dask Web App.
August 28 - 31	**Time for unexpected delay** ● Finish documentation for task 3. ● Fix bugs and provide patches for them.
September 1-18	**Begin Task 4:** ● Write all the tests for the new code added. ● Continue to finishing/polishing everything, after discussion with mentors
September 18 - 20	**Time for unexpected delay** ● Finish all documentation and tests.
September 20-22	**Final Submission** ● Submit git repository of final code with complete documentation, unit/integration tests.
September 23 – 26	**Evaluations:** ● Mentor organization submits their evaluations of the students work.

September 23 – 26	**Students Submit Code and Report** ● Students submit their code and a project report.

Deliverables

- Interactive and Animated Plots powered by Plotly.
- Dask Web App for easy web-based visualization.
- Bug-free, well written and fully documented code.
- Unit and Integration tests for the new code.
- Fortnightly blogs on developmental advances and milestones.

Future Goals

If selected, after finishing the project, I will continue helping and fixing bugs. Develop new features in the library if required.
I would love to work on this package (in my spare college time) if the project is not done this time in Summer of Code in Space. In case the project is completed this time, I'll try to provide patches and bug fixes for it since I would love to contribute to the project in some of the other way.

References

1. EinstienPy
2. Plotly
3. Github Repo