Computing Methods for Physics – 28 January 2022

Your exam must be submitted via google classroom by 13:30 as a single `zip` file containing all relevant code files, plots, datafiles, etc.

Motion of a Satellite in Earth's Atmosphere

In a reference frame with origin in the geocenter, the Newtonian equations of motion for a satellite of mass m orbiting Earth are given by

$$m\frac{d^2\vec{r}}{dt^2} = -G\frac{mM_\oplus}{r^2}\hat{r} + \vec{D}\,, \tag{1}$$

where $\vec{r}$ is the position of the satellite, i.e., $\vec{r} = (x, y, z)$ in Cartesian coordinates.

The first term on the right hand side is the gravitational force between the two masses involved [$M_\oplus = 5.972 \cdot 10^{24}\,\mathrm{kg}$ and $G = 6.67 \cdot 10^{-11}\,\mathrm{N\,m^2\,kg^{-2}}$].

The second term is the drag force that an object undergoes in a fluid. It may be expressed via the drag formula

$$\vec{D} = -\frac{1}{2}\rho v^2 A C_d \hat{v}\,, \tag{2}$$

where ρ is the density of the fluid, $\vec{v}$ is the speed of the object relative to the fluid, A is the cross sectional area of the object, and C_d is the drag coefficient, a dimensionless number. In our scenario, Earth's atmosphere is the fluid and ρ is a function of the altitude h (height measured from the ground) which may be modelled [in $\mathrm{Kg/m^3}$] as follows:

$$\rho = 6 \cdot 10^{-10} \exp\left[-\frac{(h-175)\mu}{T}\right]\,, \tag{3}$$

where

$$\mu = 27 - 0.012(h - 200) \tag{4}$$

$$T = 900 + 2.5(F10.7 - 70) + 1.5A_p \tag{5}$$

are the molecular mass of air as a function of altitude and its temperature as a function of the solar radio flux at 10.7 cm $F10.7$ and the geomagnetic index A_p. This model is used for $180\,\mathrm{km} < h \lesssim 1000\,\mathrm{km}$, and in the equations above h is in km. Further, $F10.7 \in [65, 300]$ SFUs [Solar Flux Units; $1\,\mathrm{SFU} = 10^{-22}\,\mathrm{Watts/m^2\,Hz}$] depending on the solar activity, while $A_p \in [0, 400]$, for T expressed in Kelvin.

You will have to use C++ to integrate the equations of motion to simulate a few cases and Python to plot and verify your results.

Part 1

Write a code in C++ to implement the model and integrate the equations of motion. Recalling that $\vec{a} = d\vec{v}/dt$ and $\vec{v} = d\vec{r}/dt$, the three second order differential equations in (1) become a system of six first order differential equations

$$m\frac{d\vec{v}}{dt} = -G\frac{mM_\oplus}{(x^2+y^2+z^2)^{3/2}}\vec{r} + \vec{D} \tag{6a}$$

$$\frac{d\vec{r}}{dt} = \vec{v} \tag{6b}$$

that may be solved once the initial conditions for $\{x, y, z, v_x, v_y, v_z\}$ are specified.

1. Write a class `Planet` with mass and radius as minimal attributes (use $R_\oplus = 6371\,\mathrm{km}$ when you create the instance for Earth).

2. Write a class `Atmosphere` to be used accordingly with your Earth instance of the class `Planet`.

3. Write a class `Satellite` with proper arguments.

4. Provide two classes `Euler` and `RungeKutta2` to implement the method `simulation()` of a base class `FlySatellite`. `Euler` and `RungeKutta2` must integrate numerically Eqs. (6). Namely, given the Cauchy problem $du/dt = f(t, y)$, $u(t_0) = u_0$, the `Euler` integration method approximates the solution $u = u(t)$ with the discrete values

$$t_{i+1} = t_i + \Delta t \tag{7a}$$

$$u_{i+1} = u_i + f(t_i, u(t_i))\Delta t\,, \tag{7b}$$

while for `RungeKutta2` the approximate solution is given by

$$t_{i+1} = t_i + \Delta t \tag{8a}$$

$$K_1 = f(t_i, u_i) \tag{8b}$$

$$K_2 = f(t_i + \Delta t/2, u_i + K_1\Delta t/2) \tag{8c}$$

$$u_{i+1} = u_i + K_2\Delta t\,, \tag{8d}$$

where in both cases Δt is the step size for the iterative integration method and $i = 0, \ldots, N$. Δt and N are therefore parameters of the integration method itself. In our scenario $u = u(t) = \{\vec{r}(t), \vec{v}(t)\}$ and t_0 can be set to 0 without any loss of generality.

5. Provide an application `app.cpp` of these classes that can be used to produce simulations with the initial condition $\{\vec{r}(0) = (r_0, 0, 0), \vec{v}(0) = (0, \sqrt{GM_\oplus/r_0}, 0)\}$, with $r > R_\oplus$, providing the ability to select either of the integration methods. The parameters of the simulation — including the integration method and the number of integration steps — must be read from a text input file called `params.ini`. `app.cpp` must produce a text output file `sim.dat` with columns reporting all the $x_i, y_i, z_i, v_{x,i}, v_{y,i}, v_{z,i}$ values.

Part 2

Use Python for the following tasks.

1. Show that the results of `app.cpp` are correct if you simulate the free fall of a point mass ($A = 0$), that is, that they match $y(t) = gt^2/2$ and are independent of m. Use $r_0 = 250\,\mathrm{m}$ and $\Delta t = 0.01\,\mathrm{s}$.

2. Show the evolution of a satellite with mass 1200 Kg and $A = 25\,\mathrm{m}^2$ that starts at an altitude of 600 km. Use $\Delta t = 1\,\mathrm{s}$, $F10.7 = 80\,\mathrm{SFUs}$, $A_p = 50$, and $C_d = 2$.

3. Show how the evolution for the previous scenario changes if you vary $\Delta t = 1\,\mathrm{s}$.

4. Provide the ability to check that the value of z remains 0 (or approximately 0) throughout your simulations. Comment your results.

5. Plot the mechanical energy ($mv^2/2 - GmM_{\oplus}/r$) as a function of time for a few of your simulations. Comment your results.

In all cases, display results obtained with both algorithms.

Important Remarks

- C++ evaluation will be based on: correct syntax, proper return types, proper arguments of functions, data members and class interfaces, setters/getters, unnecessary void functions, correct implementation of the strategy pattern for the integration, correct mathematical expression and physical units, comments throughout the code, separation of class implementations and interfaces.

- Python evaluation will be based on: correct syntax, avoiding C-style loops, using Python features in general, comments throughout the notebook/scripts, labels, legends and plot styling and clarity in general. The Python coding may be carried out in a notebook or in scripts, as you wish.

- The various `params.ini` input files you use and `sim.dat` output files you produce must be submitted (and accordingly renamed). This guarantees the reproducibility of your output files with your C++ material (starting from your input files), and of your plots with your Python material (starting from output files).

- The implementation of a single integration strategy, its use, and its plots in Python are preferable with respect to a strategy pattern attempt for both integration methods with no Python material (regardless of whether the strategy pattern works or not).