Orbits in Schwarzshild spacetime: numerical solutions

UNIGE assistants: Anton CHUDAYKIN, Ajith SAMPATH, Ahmad NOURI
(Anton.Chudaykin@unige.ch, Ajith.Sampath@unige.ch ahmadreza.nourizonoz@unige.ch,)

EPFL assistants: Antoine VUIGNIER, Mattia VARRONE
(antoine.vuignier@epfl.ch, mattia.varrone@epfl.ch)

In class you studied in detail the geodesics in a Schwarzshild spacetime. In particular you derived the equation of the orbit for a massive particle (see Eq. 3.71 in the lecture notes).

$$\left(\frac{dr}{d\phi}\right)^2 + \frac{1}{L^2}r^4 - \frac{2GM}{L^2}r^3 + r^2 - 2GMr = \frac{2\epsilon}{L^2}r^4. \tag{1}$$

You have seen that, after some algebraic manipulation, Eq. (1) can be written as a *2nd order ordinary differential equation*. Using the auxiliary variable $x = \frac{L^2}{GM}\frac{1}{r}$ and differentiating with respect to ϕ, you find [1]

$$\frac{d^2x}{d\phi^2} = 1 - x + \frac{3G^2M^2}{L^2}x^2. \tag{2}$$

In class you computed a solution of this equation using a perturbative method and you computed the perihelion advance of the elliptical orbit. In this exercise session we will learn how to find a numerical solution of this equation, that does not make any assumption on the amplitude of the relativistic correction.

1 Preliminaries: getting started with `python` and `Notebooks`

Most of the numerical codes needed for this tutorial will be provided to you by the assistant. However, to run the code, you will need to set up a coding environment.

The numerical computations carried out in this tutorial can be performed using any software or programming language that you like. `Mathematica`, based on the Wolfram language, and `Jupyter`, based on the `python` language, are common choices for the kind of computations we need: they offer the 'readable' format of a notebook and some packages / modules to perform symbolic and tensorial operations.

To run this tutorial, we choosed the `python` language as it is open source and relatively easy to set up. You can set up the environment needed for the tutorials in two ways.

1.1 Setting up a `Jupyter Notebook`

This is a list of steps you should follow (ask for help if something doesn't work!):

- Download and install `Anaconda3` (currently, the latest version is 3.8), which can be downloaded for free from `https://www.anaconda.com/distribution`. Anaconda includes most of modules and packages needed for scientific computing.

- You will need to install two packages not included in the Anaconda distribution: `simpy` (which allows for symbolic computations in `python`) and `gravipy` (which is a tensor calculus package for General Relativity). You can use the package manager `pip` for installing them, i.e. you should type from your terminal

```
pip install gravipy
pip install sympy
```

- If nothing went wrong, you can open your notebook typing in your terminal `jupyter notebook`.

[1] You can check this as an exercise.

1.2 Running a notebook from browser with `google Colab`

`google Colab` is an interactive environment which allows you to run codes in `pyhton` from the browser. It only requires a google account. You can check how `google Colab` works here: `https://colab.research.google.com/notebooks/intro.ipynb`, sign-in with your account and start a new notebook. To collect the needed package, in the first cell of the notebook, you should type

```
!pip install gravipy
!pip install sympy
```

If following these steps you haven't received any error message, you should have everything set up correctly to follow the tutorials. Please, contact us if you encounter problems in setting this up.

2 Finite-difference methods for solving differential equations

Before going through the numerical implementation, we need some generalities on how to solve numerically an equation in the form (2). This is a 2nd order differential equation, in the form

$$\frac{d^2x}{dt^2} = f(x). \tag{3}$$

Q1. Show that a 2nd order differential equations in the form (3) can be written as a system of two 1st order differential equations.

Therefore, in order to solve the equation for the orbit of a massive particle in Schwarzshild, we need to know how to solve numerically a 1st order differential equation.

Q2. Consider a 1st order differential equation in the form

$$\frac{dx}{dt} = g(x), \tag{4}$$

with initial condition $x(t_0) = x_0$. To solve this equation numerically, you need to discretize the different terms in Eq. (4). The independent variable t can be discretized in the range where you want to integrate the equation (i.e. split the range of variation for t in n intervals):

$$t \to \{t_0, t_1, ...t_{n-1}, t_n\} \tag{5}$$

the differential operator df can be discretized as

$$df \to \Delta f = f(t_{i+1}) - f(t_i) = f_{i+1} - f_i, \tag{6}$$

and the function $g(x)$ can be discretized evaluating it at x_i:

$$g(x) \to g(x_i) = g(x(t_i)). \tag{7}$$

Given this simple discretization scheme, find an algorithm that provides you the solution of the equation $x_{i+1} \equiv x(t_{i+1})$, given $\{x_i, t_{i+1}, t_i\}$ and explain why this scheme gives you the solution of the (4) in the full range $\{t_0, t_1, ...t_{n-1}, t_n\}$.

This method is known as *Euler's method* and it's a first-order method to find the solution of ordinary differential equation with a given initial condition.

Q3. Adapt this scheme to the system of two differential equations derived in Q1.

3 On units and initial conditions

From the previous section, we have a scheme to solve the equation for the orbit of a massive particle in Schwarzshild spacetime. Now we need to set up our initial conditions to model a system that can be interesting in physics. Similarly to what you did in class to compute the perihelion advance, we will consider a model of the solar system, with the Sun as source of our gravitational field and Mercury as *test* massive particle. We need now to estimates the numerical factors and the initial conditions in the correct units.

Q1. Write Eq. (2) in term of the Schwarzshild radius $r_{\rm S} = 2GM$ and estimate $r_{\rm S}$ of the Sun in units of Km.

Q2. What does L represents? What are its units?

Q3. We will start our simulation at the perihelion of Mercury. Assuming that the initial position of Mercury is $r_0 = 46 \times 10^6$ Km and it's initial velocity is $v_0 = 59$ Km/s, estimate L^2. What is the value of the amplitude of the GR correction in Eq. (2)?

Q4. How would you set the initial condition for $\frac{dx}{d\phi}(\phi_0)$?

4 Geodesic equations for given space-time metric using the `python` module `gravipy`

In the course, you have learnt to calculate the Christoffel symbols, Riemann curvature tensor, Ricci tensor, and Ricci scalar, which ultimately help you to calculate the geodesic equations for a space-time metric. In this exercise illustrated in a separate `python` notebook, you can learn getting help from a special `python` module called `gravipy` (installation instructions in 1.1) to calculate in an analytical form these quantities for any given metric. The module also allows you to evaluate the geodesic equations for the metric in symbolic form.

Using the brief tutorial of the module `gravipy` given in the `python` notebook, use the metric of an arbitrary static, spherically symmetric space-time given by

$$\mathrm{d}s^2 = -f(r)\mathrm{d}t^2 + h(r)\mathrm{d}r^2 + r^2(\mathrm{d}\theta^2 + \sin^2\theta\mathrm{d}\phi^2) = -e^{2\alpha(r)}\mathrm{d}t^2 + e^{2\beta(r)}\mathrm{d}r^2 + r^2\mathrm{d}\Omega^2, \tag{8}$$

Q1. Calculate all the non-vanishing Christoffel symbols in Eq. (3.33) in the lecture notes (page 81)

Q2. Calculate all the non-vanishing components of the Riemann tensor in Eq. (3.34) of the lecture notes (page 81)

Q3. Calculate the Ricci tensor and verify from the lecture notes (Eq. (3.35) to (3.38) on page 81)

Q4. Simplify the metric to get the Schwarzschild metric and calculate the Kretschmann scalar given by $K = R_{\mu\nu\lambda\sigma}R^{\mu\nu\lambda\sigma}$ to show that r=0 is a singularity for the Schwarzschild metric

Q5. Evaluate the geodesic equations for the Schwarzschild metric and verify from the lecture notes (Eq. (3.49) to (3.52) on page 83)