Gezeitenreibung

TECHNISCHE UNIVERSITÄT DARMSTADT

Computational Physics 2020
Bennet Weiß and Nico Alt
July 26, 2020

Abstract

Tidal forces have a huge impact on the rotation of the earth-moon system. As part of the lecture "Computational Physics", we studied this impact by numerically solving increasingly complex models of the mentioned system. We used Python [1] with NumPy [2] to solve our differential equations, Matplotlib [3] to plot and PyOpenGL [4] and pygame [5] to create three-dimensional animations of their solutions.

When inspecting this project for the first time, we highly recommend to open the documentation in *docs/gezeiten/index.html* in a web browser or alternatively read the document *docs/gezeiten.pdf*.

2.1a: Solving differential equations of two body problem

When implementing various algorithms to solve the differential equations of the two body problem, we extensively used the object-oriented part of Python by creating multiple classes to correctly abstract our business logic. For example, the class *TwoBodyProblem* inherits from the class *DifferentialEquation*, *FourBodyProblemSimple* inherits from *TwoBodyProblem* and so on. On the other side, all our three solvers *EulerSolver*, *MagicSolver* and *RungeKuttaSolver* inherit from the class *Solver*. This allows us to dynamically switch between the two body problem and four body problem, using different solvers just by specifying it in the method call. To illustrate it:

```
earth_moon_problem = TwoBodyProblem(
    time_boundaries=(0, 365 * 24 * 60 * 60),
    data_points_amount=5000
)
earth_moon_problem.solve(RungeKuttaSolver())
```

When using matplotlib to visualize the solutions of the problems, it doesn't matter whether we just solved a two body problem or a complex four body problem. We just call *plot* on the object of type *DifferentialEquation*: *earth_moon_problem.plot()*

With Euler being the easiest algorithm, it's also the most inaccurate one. We needed a lot of time steps and thus a lot of computation time to reach a similar level of accuracy as the Runge Kutta or scipy.integrate.solve_ivp algorithm. In our self-implemented solvers the step size is constant, but the required amount of steps to reach comparable results varies between Euler and Runge Kutta. While having good results with 150 steps and scipy.integrate.solve_ivp and 5,000 steps and Runge Kutta, 5000 steps weren't enough for Euler.

When calling solve_ivp, we set the method to "Radau" (implicit Runge-Kutta method of the Radau IIA family of order 5) because with the default method "RK45" (explicit Runge-Kutta method of order 5(4)) we get inaccurate results. This leads us to the conclusion that we're working with a stiff problem that is not numerically stable.

Technically, solving the differential equations in a time period of 1 month would suffice to create the complete figure (1) and gain insight into the Earth-Moon movement. However, to illustrate the (in-)accuracy of our solvers, we decided to calculate the solution for 60 days or over 2 months, respectively.

2.1b: Plotting and animating solutions

As already stated in the abstract, we used matplotlib to generate 2D plots of the problems' solutions and pygame and PyOpenGL to create 3D animations of the Earth and Moon circulating around their center of mass. The code of the 3D animations is based on a project by elbanic [6], published under GNU LGPL version 2.1. Especially the phase plots (3) and (4) visualize the repetitive movement of the Earth-Moon system and how the Euler solver is not able to create satisfying results.

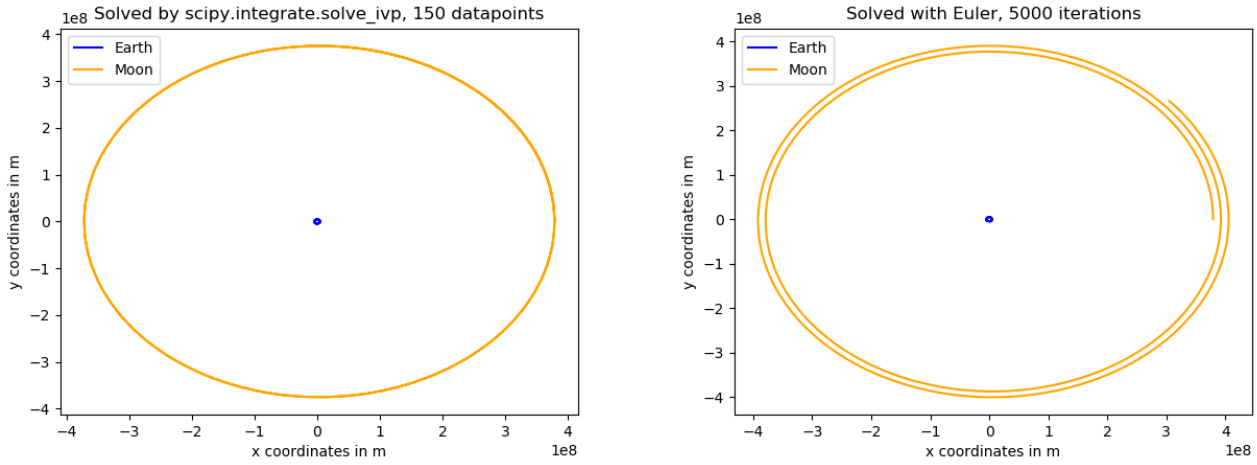

Figure 1: Positions of Earth and Moon in two body problem. Left: scipy's solve_ivp, right: Euler

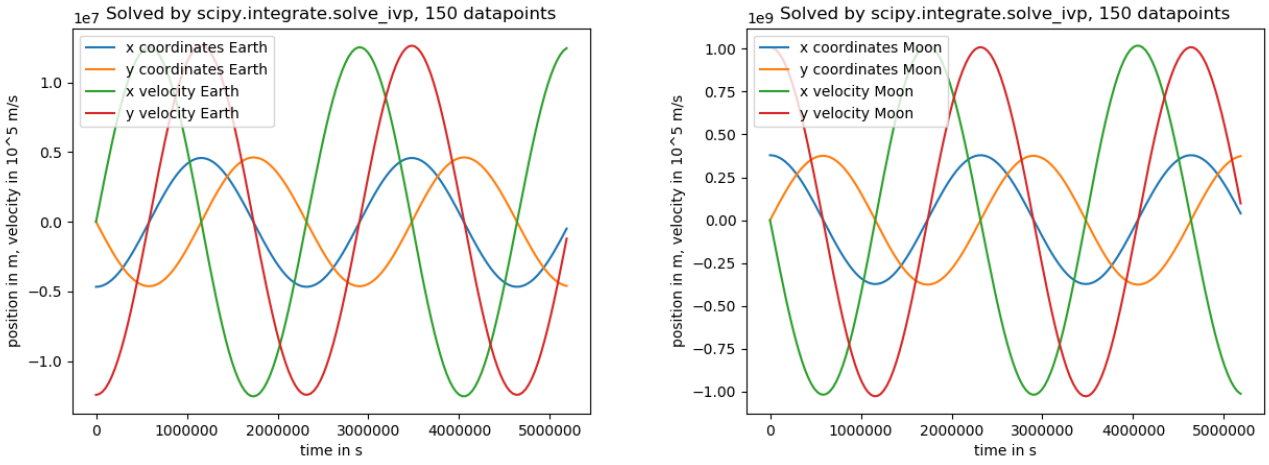

Figure 2: Time series in two body problem. Left: Earth, right: Moon

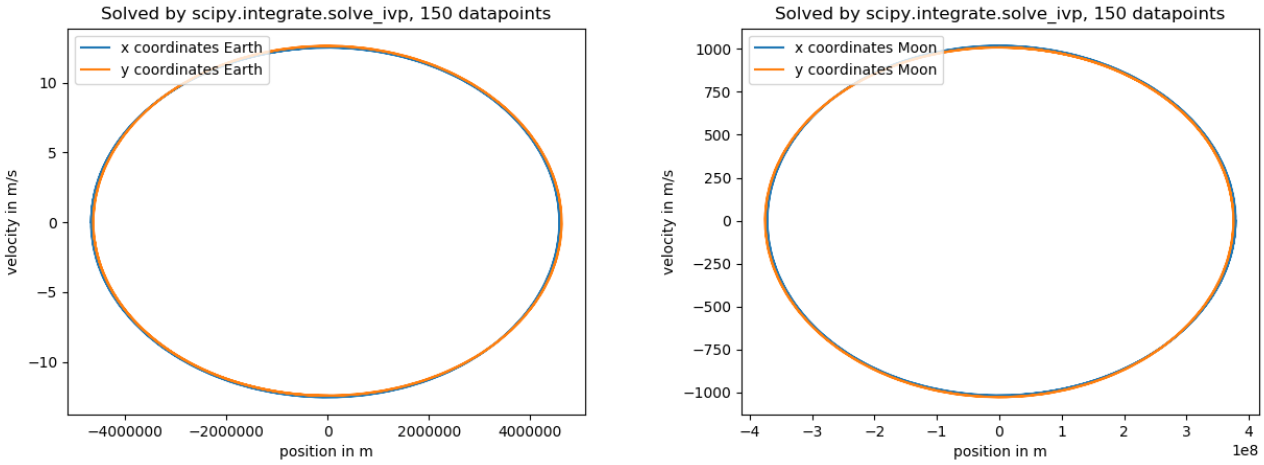

Figure 3: Phase plot in two body problem. Left: Earth, right: Moon

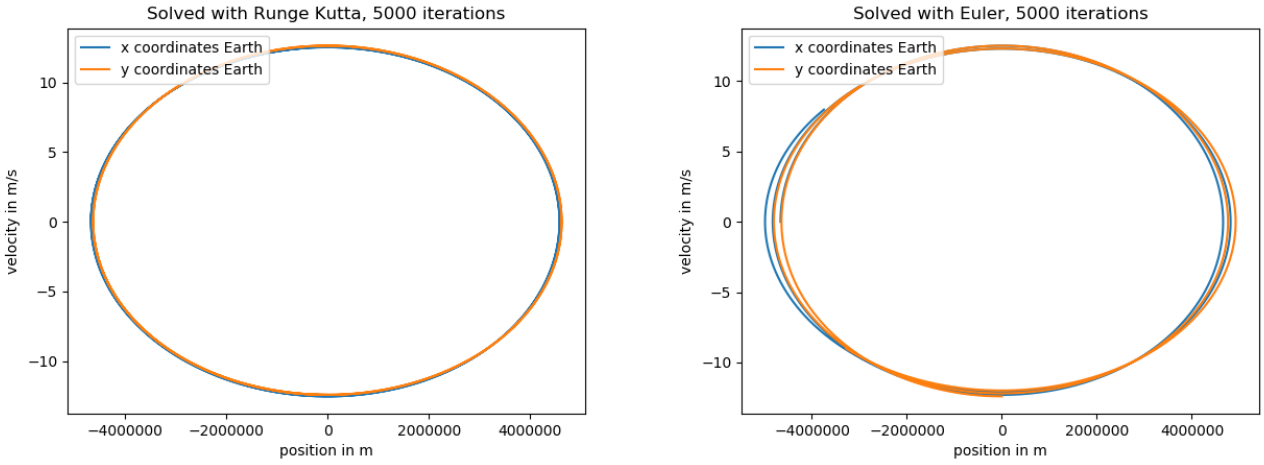

Figure 4: Phase plot of Earth in two body problem. Left: Runge Kutta, right: Euler

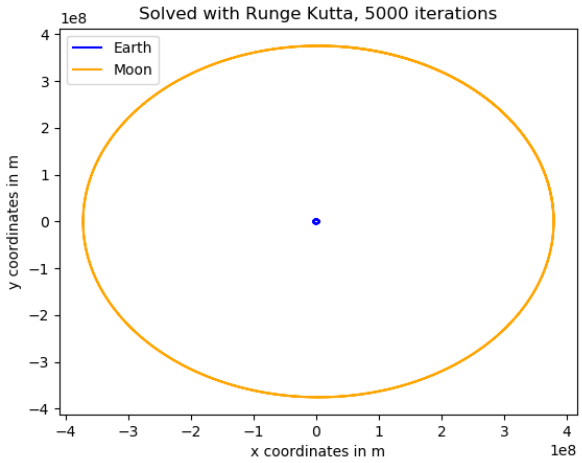

Figure 5: Positions of two body problem, solved by Runge Kutta

2.1c: Computation of center of mass

To plot the time series of the center of mass, we calculate its position with the follow equation:

$$\mathbf{r_C} = \frac{\mathbf{r_M} m_M + \mathbf{r_E} m_E}{m_M + m_E}$$

The coordinates $\mathbf{r_M}$ and $\mathbf{r_E}$ are in the center of mass' coordinate system. Plotting $\mathbf{r_C}$ results in the following graph:

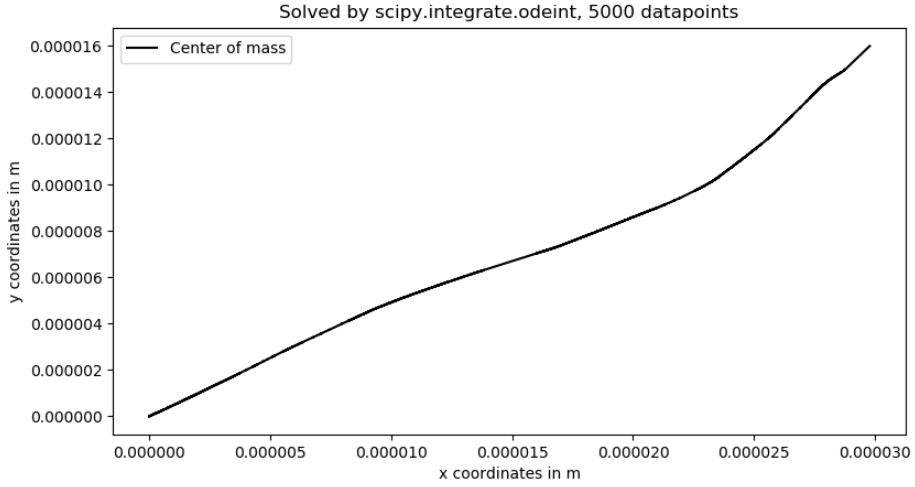

Figure 6: Center of mass in two body problem

In an isolated system like ours, we would expect the center of mass to always stay at the same place. However, as we can see in the graph, the position in our simulation is moving, although it's only at a scale of 10^{-5}. We believe these movements to be a result of the approximate nature of numerical solutions.

2.2a: Differential equations of high tides

Since the tides are supposed to remain in a constant distance to Earth, the constraints can be formulated as followed: $\mathbf{r_{Fi}^2} = x_{Fi}^2 + y_{Fi}^2 = r_E^2$. Using the generalized coordinate φ of the polar coordinate system, the constraints can be implemented easily and we substitute:

$$x = r_E \cos\varphi + x_E, \dot{x} = -r_E\dot{\varphi}\sin\varphi + \dot{x}_E$$
$$y = r_E \sin\varphi + y_E, \dot{y} = r_E\dot{\varphi}\cos\varphi + \dot{y}_E$$

With Lagrangian mechanics it's easy to find the equation of motion for φ. First, the kinetic energy T and the potential V need to be formulated:

$$T = \frac{1}{4}m_O\dot{\mathbf{r}}^2$$
$$= \frac{1}{4}m_O(r_E^2\dot{\varphi}\sin^2\varphi - 2r_E\dot{\varphi}\dot{x}_E\sin\varphi + \dot{x}_E^2 + r_E^2\dot{\varphi}^2\cos^2\varphi + 2r_E\dot{\varphi}\dot{y}_E\cos\varphi + \dot{y}_E^2)$$
$$= \frac{1}{4}m_O[r_E^2\dot{\varphi}^2 + 2r_E\dot{\varphi}(\dot{y}_E\cos\varphi - \dot{x}_E\sin\varphi) + \dot{x}_E^2 + \dot{y}_E^2]$$

$$V = -\frac{1}{2}Gm_Om_M\frac{1}{|\mathbf{r}-\mathbf{r_M}|} = -\frac{1}{2}Gm_Om_M\frac{1}{\sqrt{(r_E\cos\varphi + x_E - x_M)^2 + (r_E\sin\varphi + y_E - y_M)^2}}$$

The Lagrangian is defined by

$$L = T - V$$

Using the Euler-Lagrange equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{\varphi}} = \frac{1}{2}m_O[r_E^2\ddot{\varphi} + r_E(\ddot{y}_E\cos\varphi - \dot{y}_E\sin\varphi\dot{\varphi} - \ddot{x}_E\sin\varphi - \dot{x}_E\dot{\varphi}\cos\varphi)]$$
$$\frac{\partial L}{\partial\varphi} = -\frac{1}{2}m_Or_E\dot{\varphi}(\dot{y}_E\sin\varphi + \dot{x}_E\cos\varphi) - G\frac{m_O}{2}m_Mr_E\frac{(r_E\cos\varphi + x_E - x_M)(-\sin\varphi) + (r_E\sin\varphi + y_E - y_M)\cos\varphi}{|\mathbf{r}-\mathbf{r_M}|^3}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{\varphi}} = \frac{\partial L}{\partial\varphi}$$
$$\Leftrightarrow$$
$$\frac{1}{2}m_Or_E^2\ddot{\varphi} = \frac{1}{2}m_Or_E(\ddot{x}_E\sin\varphi - \ddot{y}_E\cos\varphi)$$
$$-\frac{1}{2}m_Om_MGr_E\frac{(r_E\cos\varphi + x_E - x_M)(-\sin\varphi) + (r_E\sin\varphi + y_E - y_M)\cos\varphi}{|\mathbf{r}-\mathbf{r_M}|^3}$$
$$\Leftrightarrow$$

$$\ddot{\varphi} = \frac{1}{r_E}(\ddot{x}_E\sin\varphi - \ddot{y}_E\cos\varphi) - \frac{1}{r_E}m_MG\frac{(r_E\sin\varphi + y_E - y_M)\cos\varphi - (r_E\cos\varphi + x_E - x_M)\sin\varphi}{|\mathbf{r}-\mathbf{r_M}|^3} \quad (1)$$

The differential equation (1) applies to both φ_{F1} and φ_{F2}. As stated in the exercise sheet, we can ignore the influence of the high tides on each other. The interaction of the high tides with the Earth is negligible as well, since the oceans' mass is small compared to the mass of the Earth. Additionally, we expect one high tide to be always on the exact other side of Earth than the second high tide. Thus, their gravitational influences on Earth cancel each other out. The differential equations for Earth and Moon are the following:

$$\ddot{\mathbf{r}}_\mathbf{E} = -Gm_M\frac{(\mathbf{r_E}-\mathbf{r_M})}{|\mathbf{r_E}-\mathbf{r_M}|^3}$$
$$\ddot{\mathbf{r}}_\mathbf{M} = -G\left[m_E\frac{(\mathbf{r_M}-\mathbf{r_E})}{|\mathbf{r_M}-\mathbf{r_E}|^3} + \frac{1}{2}m_O\left(\frac{\mathbf{r_M} - r_E\cos\varphi_1 - \mathbf{r_E}}{|\mathbf{r_M}-\mathbf{r_{F1}}|^3} + \frac{\mathbf{r_M} - r_E\cos\varphi_2 - \mathbf{r_E}}{|\mathbf{r_M}-\mathbf{r_{F2}}|^3}\right)\right]$$

Just like the differential equation of the two body problem, the differential equations in this chapter aren't numerically stable either. We therefore used the same "Radau" algorithm with solve_ivp which is designed for stiff problems. The solutions are not stable because a minor change of the initial conditions results in a totally different behaviour of the movement of the 4-body-system.

2.2c: Plotting and animating solutions

Like in exercise 2.1b, we used matplotlib to generate 2D plots of the problems' solutions. For animations, we didn't use PyOpenGL this time because the high tides' movement can't be visualized correctly from an Earth-Moon system's outer perspective. Instead, we used matplotlib's animation functionality. To improve the visibility of the tides' movement in the animation, we shrank the distance of the Moon to the Earth to 1/14th of the actual distance.

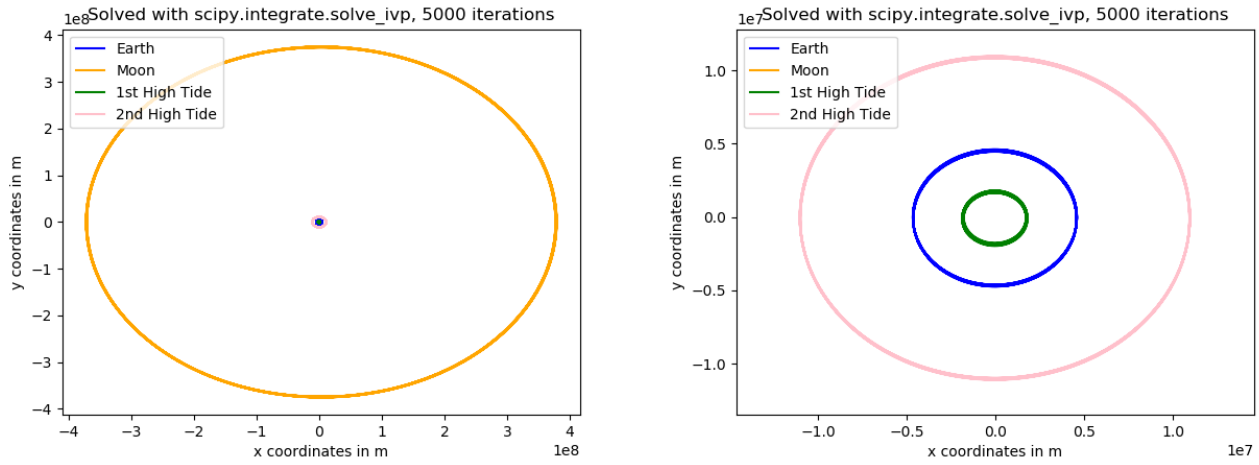

Figure 7: Positions of Earth and Moon in simple four body problem. Left: total view, right: zoomed

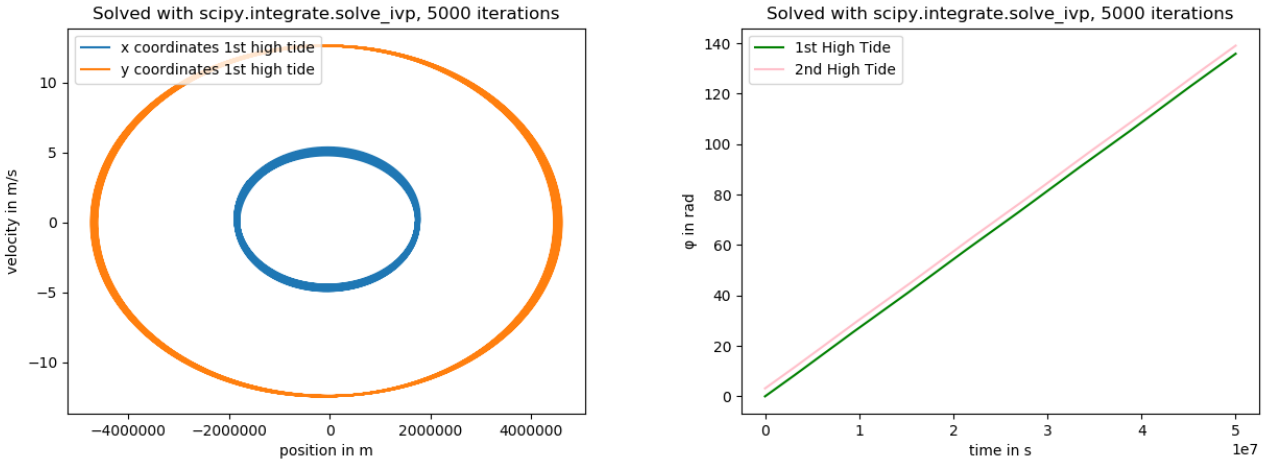

Figure 8: Simple four body problem. Left: phase plot high tide, right: angles

2.3a: Differential equations with intrinsic rotation and friction

In addition to exercise 2.2a, we now take into account the intrinsic rotation of Earth and the friction generated between the Earth's surface and the high tides. We first take a look at Earth's torque:

$$M_E = I_E \ddot{\varphi}_E = \frac{2}{5} m_E R_E^2 \ddot{\varphi}_E$$

$$M_E = hF_{ges} = R_E^3 k \frac{m_O}{2} \sum_{i=1}^{2} |\omega_{i,rel}|\omega_{i,rel} = R_E^3 k \frac{m_O}{2} \left[|\dot{\varphi}_1 - \dot{\varphi}_E|(\dot{\varphi}_1 - \dot{\varphi}_E) + |\dot{\varphi}_2 - \dot{\varphi}_E|(\dot{\varphi}_2 - \dot{\varphi}_E)\right]$$

$$\Leftrightarrow$$

$$\ddot{\varphi}_E = \frac{5}{4} R_E k \frac{m_O}{m_E} \left[|\dot{\varphi}_1 - \dot{\varphi}_E|(\dot{\varphi}_1 - \dot{\varphi}_E) + |\dot{\varphi}_2 - \dot{\varphi}_E|(\dot{\varphi}_2 - \dot{\varphi}_E)\right]$$

The torque on the high tides is given by

$$M_i = I_i \ddot{\varphi}_i = \frac{m_O}{2} R_E^2 \ddot{\varphi}_i$$

$$M_i = hF_i = -R_E^3 k \frac{m_O}{2} |\omega_{i,rel}|\omega_{i,rel} = -R_E^3 k \frac{m_O}{2} |\dot{\varphi}_i - \dot{\varphi}_E|(\dot{\varphi}_i - \dot{\varphi}_E)$$

$$\Leftrightarrow$$

$$\ddot{\varphi}_i = -R_E k |\dot{\varphi}_i - \dot{\varphi}_E|(\dot{\varphi}_i - \dot{\varphi}_E)$$

2.3b: Plotting and animating solutions

The movement of the system is best to be seen in the animation. However, figure (11, right) clearly depicts the reduction of Earths' rotational energy. At the same time, the angular velocity of the high tides is rising slowly (13) due to the conservation of momentum.

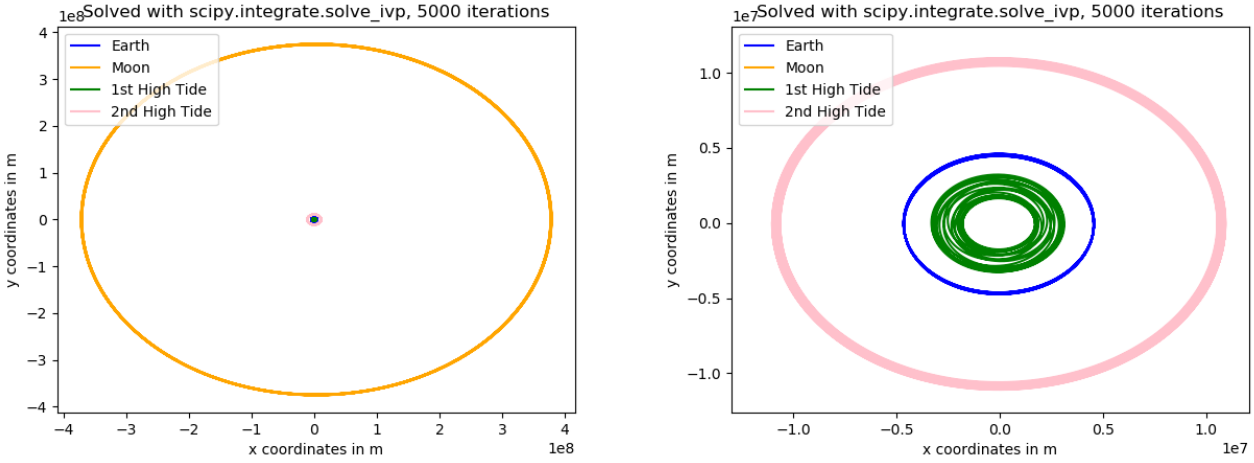

Figure 9: Positions of Earth and Moon in complex four body problem. Left: total view, right: zoomed

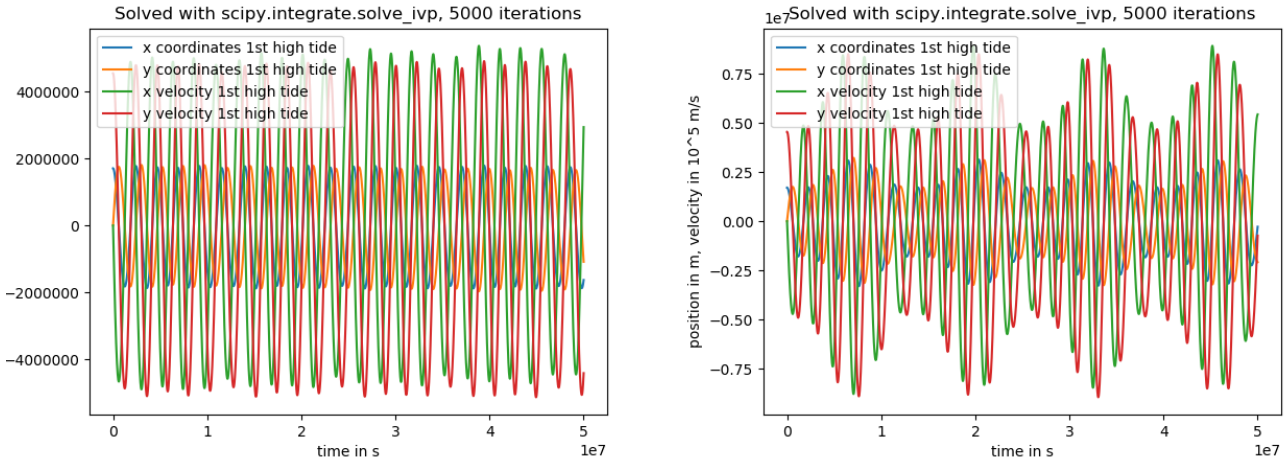

Figure 10: Time series of 1st high tide. Left: simple 4 body problem, right: complex 4 body problem

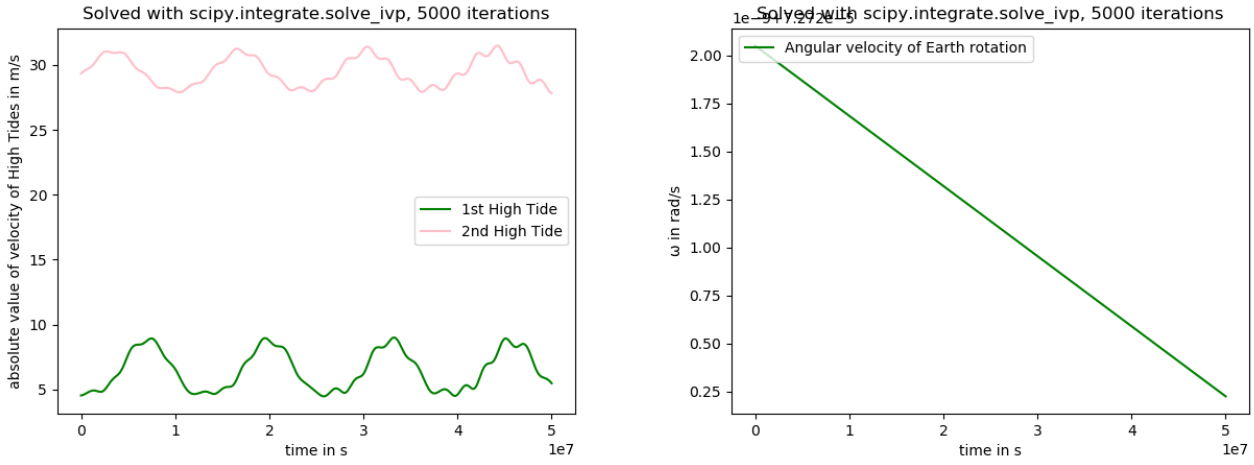

Figure 11: Complex four body problem. Left: absolute high tide velocities, right: velocity of intrinsic rotation

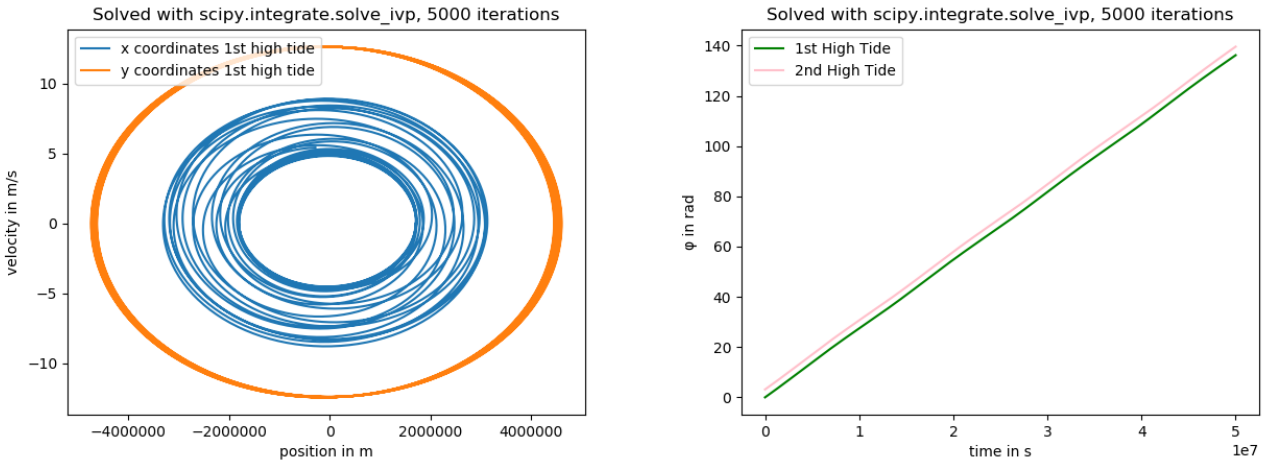

Figure 12: Complex four body problem. Left: phase plot high tide, right: angles

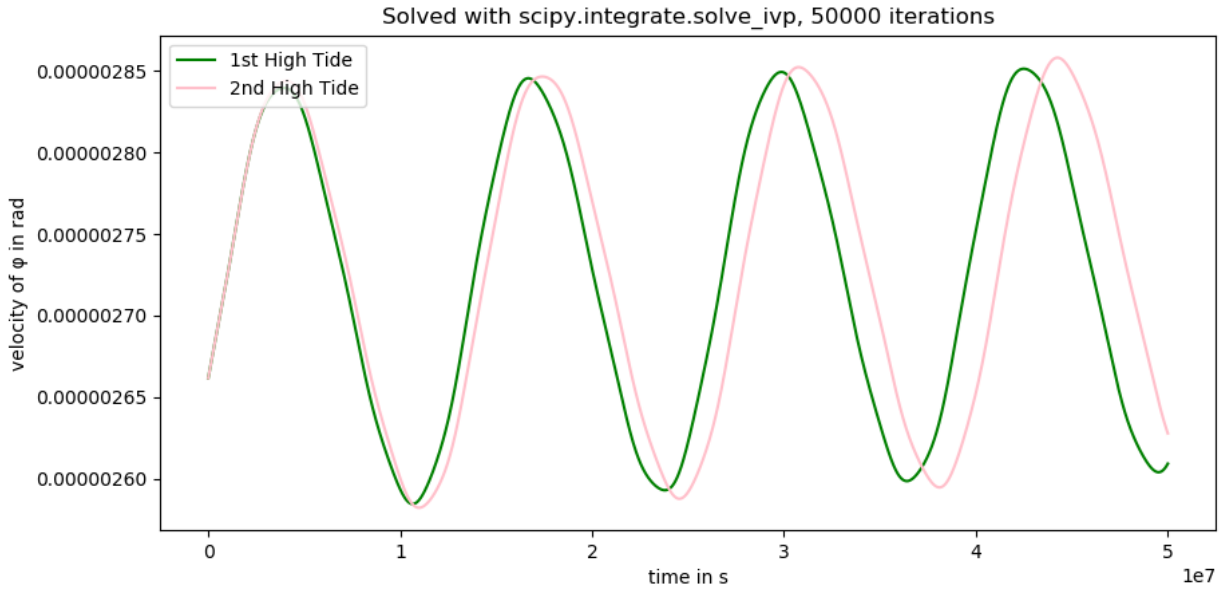

Figure 13: Angular velocity of complex four body problem

2.3c: Fitting of oceans' mass

To respect the settling time of the system, we started our measurements after 1 year. We used the following equation, with $\dot{\varphi_{t_1}}$ and $\dot{\varphi_f}$ being the angular velocity after 1 and 101 years, respectively:

$$\tau_c = 2\pi(\frac{1}{\dot{\varphi_f}} - \frac{1}{\dot{\varphi_{t_1}}})$$

We extrapolated $\dot{\varphi_f}$ by assuming the angular velocity to change linearly. In the end, we found the actual mass to be around 21,411 Gt, which is only 0.0015% of the original mass.

Obviously, the fitted mass of the oceans is way smaller than the actual mass. An explanation could be that not all water on Earth is moving but mostly surface and coast water. Additionally, in our simulation, we did not consider the Earth to be a sphere and therefore did not take the reduction of friction close to the poles into account. Finally, the oceans aren't distributed evenly and due to the continents they cannot move freely which also impacts the system.

3.2: Differential equations with n bodies

When simulating not only 2 but any amount $N \in \mathbb{N}$ of tide particles, we need to generalize the equations from exercise 2.3a:

$$\ddot{\varphi}_E = \frac{5}{2} R_E k \frac{m_O}{N * m_E} \sum_{i=1}^{N} |\dot{\varphi}_i - \dot{\varphi}_E|(\dot{\varphi}_i - \dot{\varphi}_E)$$

$$\ddot{\varphi}_i = -R_E k |\dot{\varphi}_i - \dot{\varphi}_E|(\dot{\varphi}_i - \dot{\varphi}_E)$$

$$\ddot{\mathbf{r}}_\mathbf{M} = -G \left[m_E \frac{(\mathbf{r_M} - \mathbf{r_E})}{|\mathbf{r_M} - \mathbf{r_E}|^3} + \frac{1}{N} m_O \left(\sum_{i=1}^{N} \frac{x_M - R_E \cos\varphi_i - x_E}{|\mathbf{r_M} - \mathbf{r_{Fi}}|^3)} \right) \right]$$

The forming of two high tides can be observed in the animation.

References

[1] Python Software Foundation. *Python. a programming language that lets you work quickly and integrate systems more effectively*. URL: `https://www.python.org/` (visited on 06/29/2020).

[2] NumPy developers. *NumPy. the fundamental package for scientific computing with Python*. URL: `https://numpy.org/` (visited on 07/14/2020).

[3] The Matplotlib development team. *Matplotlib. a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms*. URL: `https://matplotlib.org/` (visited on 07/14/2020).

[4] PyOpenGL developers. *PyOpenGL. the most common cross platform Python binding to OpenGL and related APIs*. URL: `http://pyopengl.sourceforge.net/` (visited on 07/14/2020).

[5] pygame developers. *pygame. a set of Python modules designed for writing video games*. URL: `https://www.pygame.org/wiki/about` (visited on 07/14/2020).

[6] EuiJun Jeong (elbanic). *SolarSystem. Simple Solar System with Python and PyGame*. URL: `https://github.com/elbanic/SolarSystem/` (visited on 07/24/2020).