Report on Worksheet 1: Integrators

David Artukovic (st184489@stud.uni-stuttgart.de),
Dimitrij Gies (st186750@stud.uni-stuttgart.de)

November 10, 2024

Institute for Computational Physics, University of Stuttgart

Contents

1 Canonball

1.1 Simulating a cannonball

The exercise 2.1 was to simulate the trajectory of a cannonball in 2D until it hits the ground. The simulation was implemented according to the simple Euler scheme, where the propagation of the position $\mathbf{x}(t) = (x, y)^{\mathrm{T}}$ and velocity $\mathbf{v}(t)$ from time t to time $t + \Delta t$ is performed with the following equations:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t \tag{1}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \frac{\mathbf{F}(t)}{m}\Delta t \tag{2}$$

The only force acting on the cannonball was due to gravity $\mathbf{F}(t) = (0, -mg)^{\mathrm{T}}$, where m is the mass of the cannonball and $g = 9,81\frac{\mathrm{kg}}{\mathrm{m}}$ the gravity constant. The starting conditions at $t = 0$ were $m = 2.0$ kg at a position of $\mathbf{x}(0) = \mathbf{0}$ and a velocity of $\mathbf{v}(0) = (60, 60)\frac{\mathrm{kg}}{\mathrm{s}}$. Δt was chosen to be 0.1 s according to the worksheet script.

1.1.1 Euler simulation code

The simulation was performed with three python functions. The first was the calculation of the gravity force array acting on the cannonball:

```python
def force(mass, gravity):
    return np.array([0, -mass * gravity])
```

The second function calculated new position `x` and velocity `v` arrays for a new time step `dt` from the gravity force array `f`:

```python
def step_euler(x, v, dt, mass, gravity, f):
    x = x + v * dt
    v = v + f / mass * dt
    return x, v
```

To calculate the cannonball trajectory the `step_euler` function was performed until the y position was ≤ 0 with the following `run` function:

```python
def run(x, v, dt, mass, gravity):
    trajectory = [x.copy()]
    for timestep in range(int(10e4)):
        x, v = step_euler(x, v, dt, mass, gravity, force(mass, gravity))
        if x[1] >= 0:
            trajectory.append(x.copy())
        else:
            break
    return np.array(trajectory)
```

1.1.2 Cannonball trajectory calculated from Euler scheme

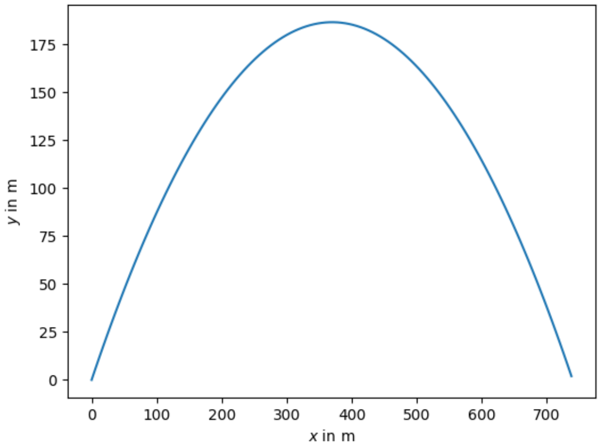

Figure 1: trajectory of the cannonball

The shape of the trajectory as depicted in Figure 1 is not dependent on the mass of the cannonball. This can be derived by either using different masses in the code, or by inserting the definition of force $\mathbf{F}(t) = m \cdot \mathbf{a}(t)$ with the cannonball acceleration vector $\mathbf{a}(t)$ into equation (2), which yields the following mass independent velocity function:

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t$$

1.2 Influence of friction and wind

For exercise 2.2 aerodynamic friction was introduced as a non-conservative force of the form $F_{\text{fric}}(\mathbf{v}) = -\gamma(\mathbf{v} - \mathbf{v}_0)$, where $\gamma = 0.1\ \frac{\text{kg}}{\text{s}}$ was the assumed friction coefficient. The force was aligned in the negative x direction, analogous to wind blowing parallel to the ground with a wind speed v_w ($\mathbf{v}_0 = (v_w, 0)^{\text{T}}\ \frac{\text{m}}{\text{s}}$). This type of aerodynamic model is known as a *viscous flow model* and can be described more generally by the Stokes law for a sphere and Reynolds numbers < 1 (laminar flow)[1]. As spheres in the size and speed range of the cannonball usually produce Reynolds numbers $>> 1$[2] in air and therefore turbulent flow[3], this model is not directly applicable to a real cannonball case.

1.2.1 Wind resistance simulation code

To simulate wind resistance as described above, the `force` function was modified by the `gamma` and `v_0` parameters:

```
def force(mass, gravity, v, gamma, v_0):
    return np.array([0, -mass * gravity]) - gamma * (v - v_0)
```

The `run` function was also extended by the `gamma` and `v_0` parameters and the invocation of the force inside the run function has been adapted accordingly:

```
def run(x, v, dt, mass, gravity, gamma, v_0):
    ...
    for timestep in range(int(10e4)):
        force_xy = force(mass, gravity, v, gamma, v_0)
        x, v = ex_2_1.step_euler(x, v, dt, mass, gravity, force_xy)
        ...
```

1.2.2 Cannonball trajectories with wind resistance

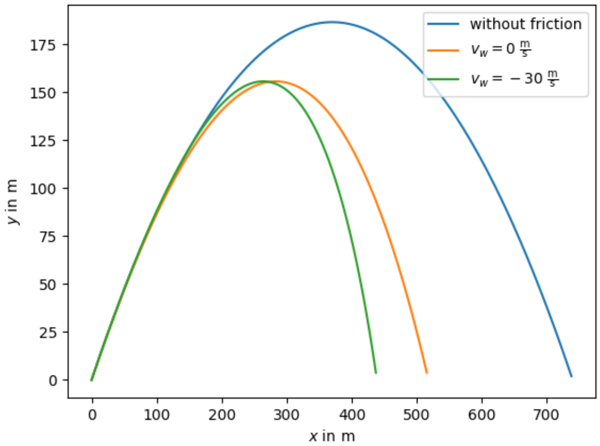

Figure 2: trajectory comparison of the cannonball calculated with the model without wind resistance and the model with resistance at $0\ \frac{\mathrm{m}}{\mathrm{s}}$ and $-30\ \frac{\mathrm{m}}{\mathrm{s}}$ wind speeds.

Figure 2 shows the trajectory of the cannonball at different wind speeds (x-direction). As can be seen, the sole introduction of the *viscous flow model* decreases the arc of the trajectory and the cannonball lands at an earlier x-position compared to the arc of the previous model.

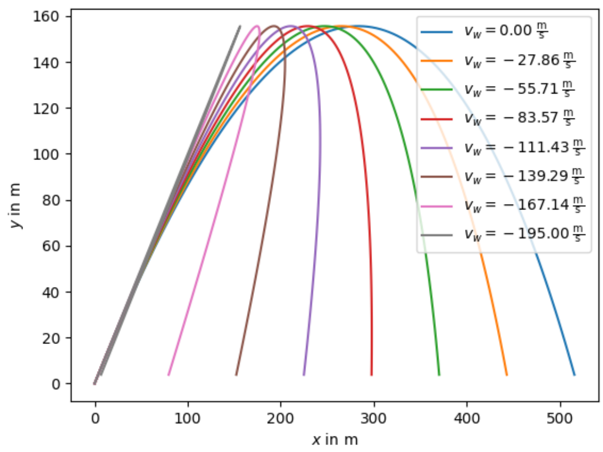

Figure 3: trajectory of the cannonball at different wind speeds.

Figure 3 shows the trajectory of the cannonball at different wind speeds. By increasing the wind speed to around $v_w = -195\ \frac{\text{m}}{\text{s}}$ the cannonball falls back to the starting position.

2 Solar System

2.1 Simulating the solar system with the Euler scheme

The simulation of the trajectories of selected planets of the solar system can be observed in figures 4 and 5:

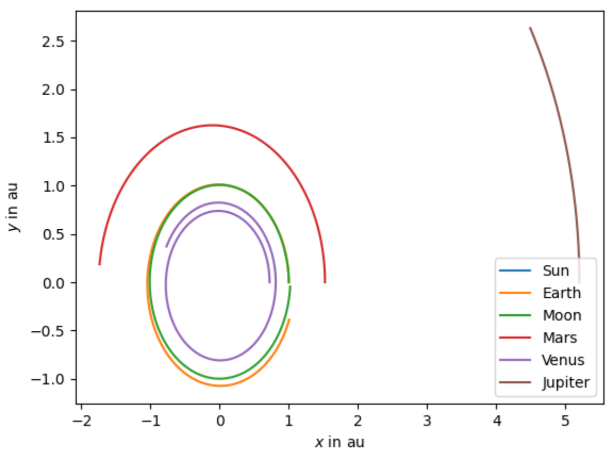

Figure 4: trajectories with t=0.0001

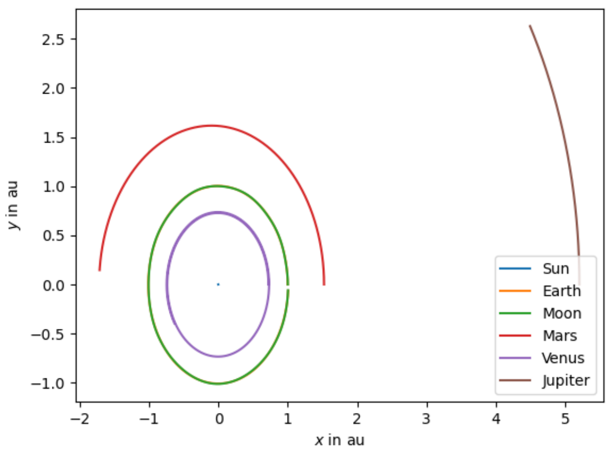

Figure 5: trajectories with t=10^{-5}

In the case of figure 4 the moon and the earth "lose" each other, indicating that a time step of $t = 0.0001$ could be too big to simulate a whole astronomic year. On the other hand, figure 5 shows the expected trajectories for the particles. Particularly, the moon and earth follow each other closely, delivering in total a satisfactory result.

Computationally a large number of particles would be noticed in the calculation of the force matrix. It would grow with a magnitude of $\mathcal{O}(n^2)$.

2.2 Integrators

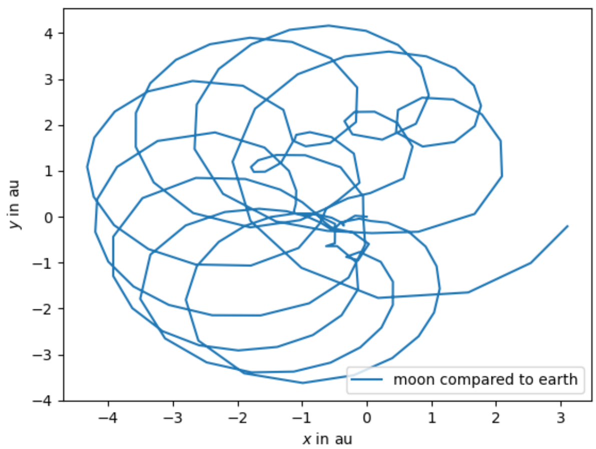

Figure 6: trajectory of the moon compared to earth for symplectic Euler algorithm

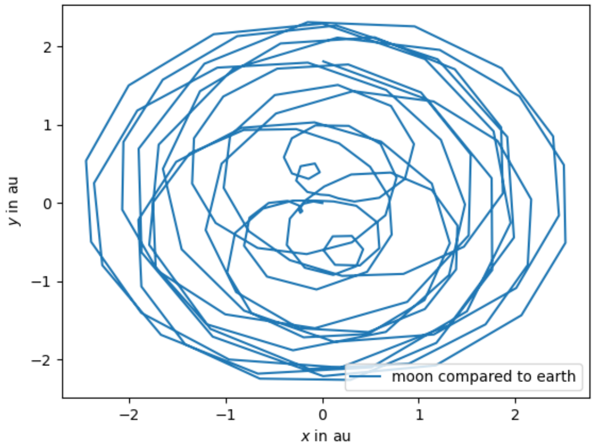

Figure 7: trajectory of the moon compared to earth for velocity Verlet algorithm

Both integrators show an unsatisfying trajectory of the moon compared to the earth for a time span of 20 years and $t = 0.01$.

2.2.1 Derivation of the position update

$$\begin{aligned} x(t+\Delta t) &= \sum_{k=0}^{2} \frac{x^k(t)}{k!}(t+\Delta t - t)^k + \mathcal{O}((\Delta t)^4) \\ &= x(t) + x'(t)\Delta t + \frac{1}{2}x''(t)(\Delta t)^2 + \mathcal{O}((\Delta t)^4) \\ &= x(t) + v(t)\Delta t + \frac{1}{2}a(t)(\Delta t)^2 + \mathcal{O}((\Delta t)^4) \end{aligned}$$

2.2.2 Derivation of the velocity update

Start with helper Taylor Expansion

$$\frac{dv(t+\Delta t)}{dt} = \frac{v(t)}{dt} + \frac{d^2v(t)}{d^2t}\Delta t \tag{3}$$

$$\iff a(t+\Delta t) = a(t) + \frac{d^2v(t)}{d^2t}\Delta t \tag{4}$$

$$\iff \frac{a(t+\Delta t) - a(t)}{\Delta t} = \frac{d^2v(t)}{d^2t} \tag{5}$$

Plug in expression 5 into 8:

$$v(t+\Delta t) = \sum_{k=0}^{2} \frac{v^k(t)}{k!}(t+\Delta t - t)^k + \mathcal{O}((\Delta t)^4) \tag{6}$$

$$= v(t) + v'(t)\Delta t + \frac{1}{2}v'(t)(\Delta t)^2 + \mathcal{O}((\Delta t)^4) \tag{7}$$

$$= v(t) + a(t)\Delta t + \frac{1}{2}\frac{d^2v(t)}{d^2t}(\Delta t)^2 + \mathcal{O}((\Delta t)^4) \tag{8}$$

$$= v(t) + \frac{a(t+\Delta t) + a(t)}{2}(\Delta t) + \mathcal{O}((\Delta t)^4) \tag{9}$$

2.2.3 Equivalence of velocity Verlet and standard Verlet algorithm

$$x(t+2\Delta t) = x(t+\Delta t) + v(t+\Delta t)\Delta t + \frac{1}{2}a(t+\Delta t)(\Delta t)^2 + \mathcal{O}((\Delta t)^4) \tag{10}$$

$$x(t) = x(t+\Delta t) - v(t)\Delta t - \frac{1}{2}a(t)(\Delta t)^2 - \mathcal{O}((\Delta t)^4) \tag{11}$$

Add 10 and 11:

$$x(t+2\Delta t) + x(t) = 2x(t+\Delta t) + [v(t+\Delta t) - v(t)]\Delta t + \frac{1}{2}a(t+\Delta t)(\Delta t)^2 - \frac{1}{2}a(t)(\Delta t)^2 \tag{12}$$

Plug in velocity from velocity Verlet algorithm:

$$x(t+2\Delta t) + x(t) = 2x(t+\Delta t) + a(t+\Delta t)(\Delta t)^2 + \mathcal{O}((\Delta t)^4) \tag{13}$$

$$\text{Set } t = t^* - \Delta t \tag{14}$$

$$\rightarrow x(t^*+\Delta t) = 2x(t^*) - x(t^*-\Delta t) + a(t^*)(\Delta t)^2 + \mathcal{O}((\Delta t)^4), \tag{15}$$

which is the standard Verlet algorithm at time point t^*.

2.2.4 Problem with Verlet Algorithm

The problem to implement a simulation based on this equation is caused by the initialization step as $x(t-\Delta t)$ is not known.

2.3 Long-term stability

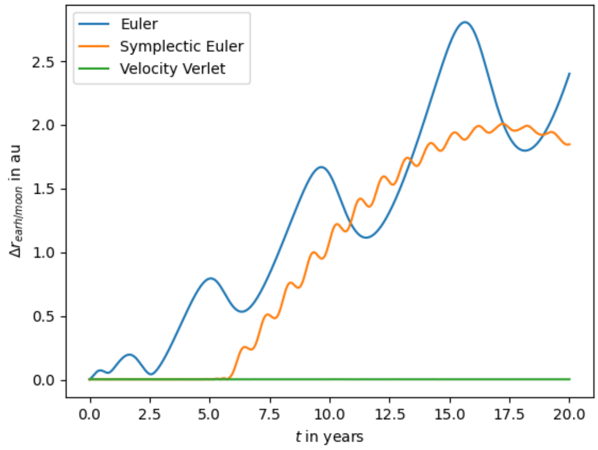

Figure 8: long-term stability plot for the *Euler*, *symplectic Euler* and *Velocity Verlet* algorithm. The distance between the earth and moon are shown as a function of the simulation time. A time step of $\Delta t = 0.01$ years was used.

To test the long-term stability of the simulations the distance between the earth and moon were plotted in Figure 8 against the simulation time. The simulations were run with a time step of $\Delta t = 0.01$ years for a total of 20 years.

Only the *Velocity Verlet* algorithm produced acceptable results as the distances did not change unpredictably for the simulation timeframe. The *non-symplectic Euler* algorithm was unstable from the start, while the *symplectic Euler* algorithm spiraled out of control at about 5 years. The latter could be due to the fact that the errors for the long Δt compiled over the time period, pushing the Venus out of its orbit and causing the sudden increase in distance (cf. Figure 9). Decreasing the time step to $\Delta t = 0.001$ years resulted in stable trajectories of the planets.

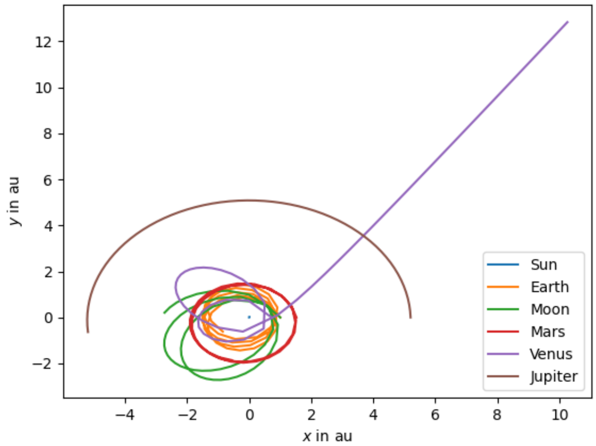

Figure 9: trajectories of planets for the *symplectic Euler* algorithm. A time step of $\Delta t = 0.01$ years was used with a total simulation time of 8 years.

3 References

[1] Stokes's Law, January 2024. URL `https://phys.libretexts.org/Courses/Prince_Georges_Community_College/General_Physics_I%3A_Classical_Mechanics/52%3A_Fluid_Dynamics/52.08%3A_Stokess_Law`.

[2] Rod Cross. Sports ball aerodynamics. URL `http://www.physics.usyd.edu.au/~cross/TRAJECTORIES/Sports%20Balls.pdf`.

[3] Flow Past a Sphere at High Reynolds Numbers, July 2019. URL `https://geo.libretexts.org/Bookshelves/Sedimentology/Introduction_to_Fluid_Motions_and_Sediment_Transport_(Southard)/03%3A_Flow_Past_a_Sphere_II_-_Stokes'_Law_The_Bernoulli_Equation_Turbulence_Boundary_Layers_Flow_Separation/3.08%3A_Flow_Past_a_Sphere_at_High_Reynolds_Numbers`.