Introduction

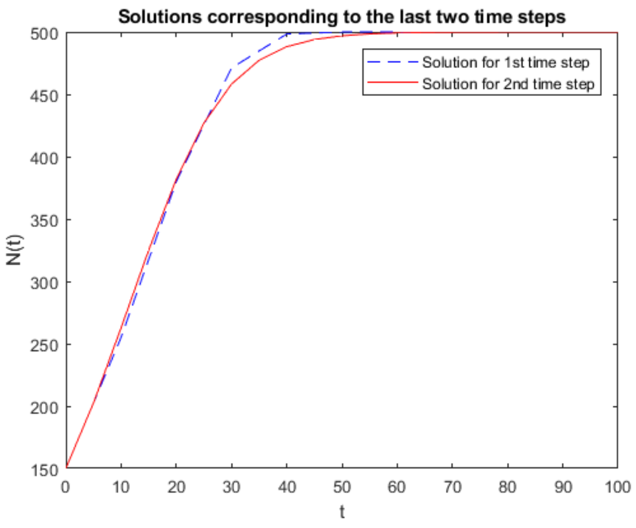
This task was about solving exercises from the book Programming for Computations - MATLAB/Octave. Assignment was completed by MATLAB R2017b. The main topic of this task was Ordinary Differential Equations.

1 Exercise 4.4: Find an appropriate time step; logistic model

This program computes the numerical solution of the logistic equation for a set of repeatedly halved time steps. Plots the solutions corresponding to the last two time steps and displays the used time step. Then it asks the user if the loop is to be continued (Yes or No).

This program *logistic_dt* calls the function *ode_FE(f, U_0, dt, T),* where f – logistic equation, U_0 – initial condition, dt – time step, T – end of the time interval. This is a function for solving any single differential equation u' = f(u, t). Outputs are arrays of *u* and *t*.

At the beginning the user is asked for initial time step, which is then halved every next cycle if user presses *Yes* to continue. A *while* loop and *if* statement is used for repetition and asking the user to launch another cycle. When user presses *No*, the program ends. For this is used function *menu*.



Time step when these two solutions can't be visually distinguished is let's say about 0.07. Solution for initial time step = 10, next time step = 5 ↑

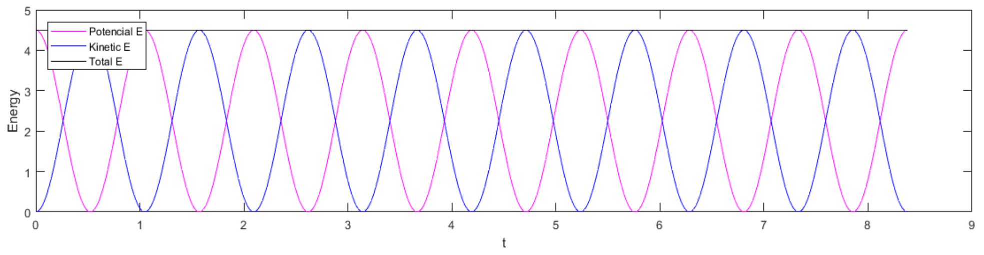
2 Exercise 4.10: Compute the energy in oscillations

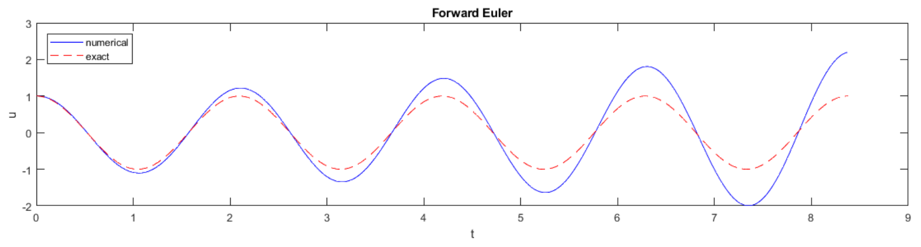
2.a

A function *osc_energy(u, v, omega)* for returning the potential and kinetic energy of an oscillating system was created. Inputs are: u – position coordinate, v – velocity coordinate, omega – natural frequency. Outputs are kinetic and potential energy.

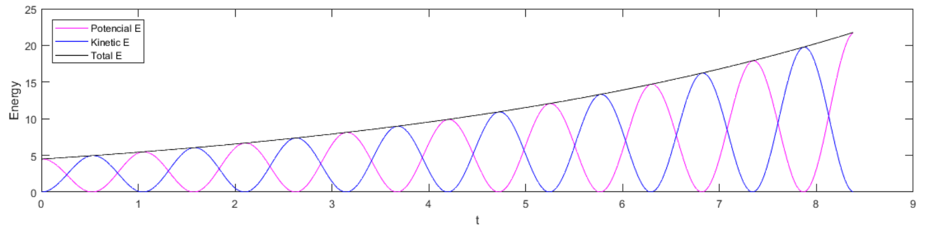
2.b

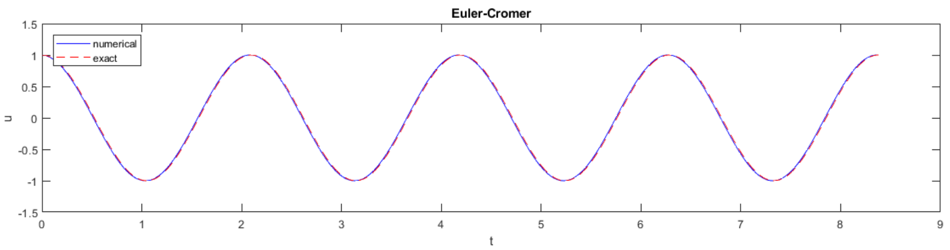
Function *osc_energy* was added to programs *osc_FE_energy* and *osc_EC_energy* which implements Forward Euler and the Euler-Cromer scheme. Then the sum of the kinetic and potential energy is plotted. For exact solution the total energy is constant like this ⟶





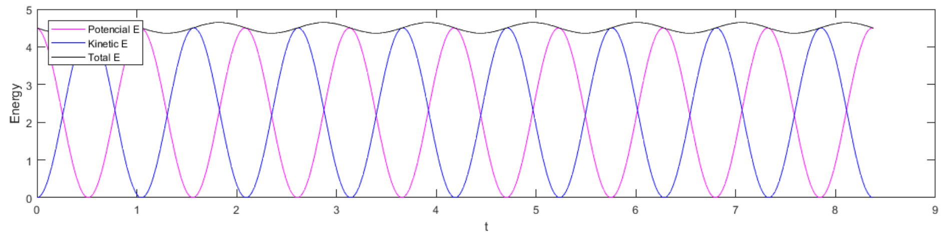
When omega = 3, dt = P/100, X_0 = 1. The energy development for the FE scheme is increasing in time like this ⟶





For EC scheme the energy development oscillates around some constant value of energy →

So, one can conclude that using EC scheme is definitely more accurate.



3 Exercise 4.14: Use a Backward Euler scheme for oscillations

3.a

System of equations was solved for u^n and v^n

$$u^n - \Delta t v^n = u^{(n-1)}$$
$$v^n + \Delta t \omega^2 u^n = v^{(n-1)}$$

$$v^n = v^{(n-1)} - \Delta t \omega^2 u^n$$

$$u^n - \Delta t(v^{(n-1)} - \Delta t \omega^2 u^n) = u^{(n-1)}$$

$$u^n - \Delta t v^{(n-1)} + \Delta t^2 \omega^2 u^n = u^{(n-1)}$$

$$u^n(1 + \Delta t^2 \omega^2) - \Delta t v^{(n-1)} = u^{(n-1)}$$

$$\boldsymbol{u^n = \frac{u^{(n-1)} + \Delta t v^{(n-1)}}{1 + \Delta t^2 \omega^2}}$$

$$v^n = v^{(n-1)} - \Delta t \omega^2 \frac{u^{(n-1)} + \Delta t v^{(n-1)}}{1 + \Delta t^2 \omega^2}$$

$$v^n = \frac{v^{(n-1)} + \Delta t^2 \omega^2 v^{(n-1)}}{1 + \Delta t^2 \omega^2} - \frac{\Delta t \omega^2 u^{(n-1)} - \Delta t^2 \omega^2 v^{(n-1)}}{1 + \Delta t^2 \omega^2}$$

$$\boldsymbol{v^n = \frac{v^{(n-1)} - \Delta t \omega^2 u^{(n-1)}}{1 + \Delta t^2 \omega^2}}$$

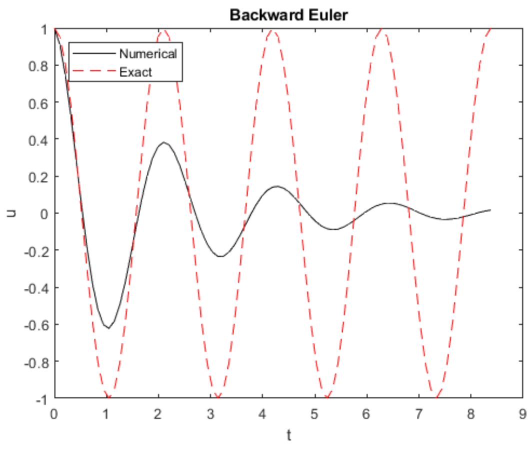
3.b

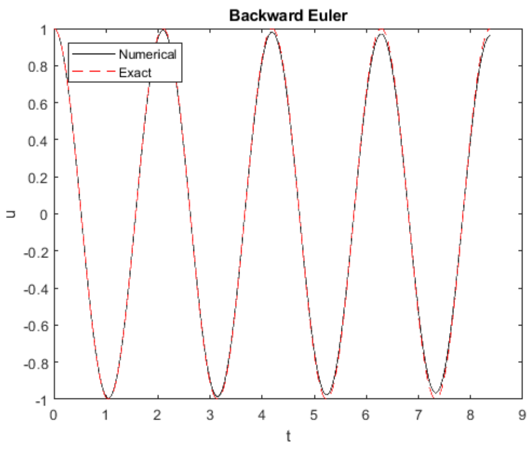
Above formulas are implemented in a program *osc_BE* for computing the entire numerical solution.

3.c

Solution for *Δt* corresponding to 20 and 2000 time steps per period of the oscillations. →

One can observe, that this method leads to decreasing amplitude in time. Not surprising is a property, that lower *Δt* leads to more accurate solutions.





Conclusions

All sources were pushed to GitHub. Link to the repository is: https://github.com/q2493/HW_3_Trusina.git
This was probably the easiest task.