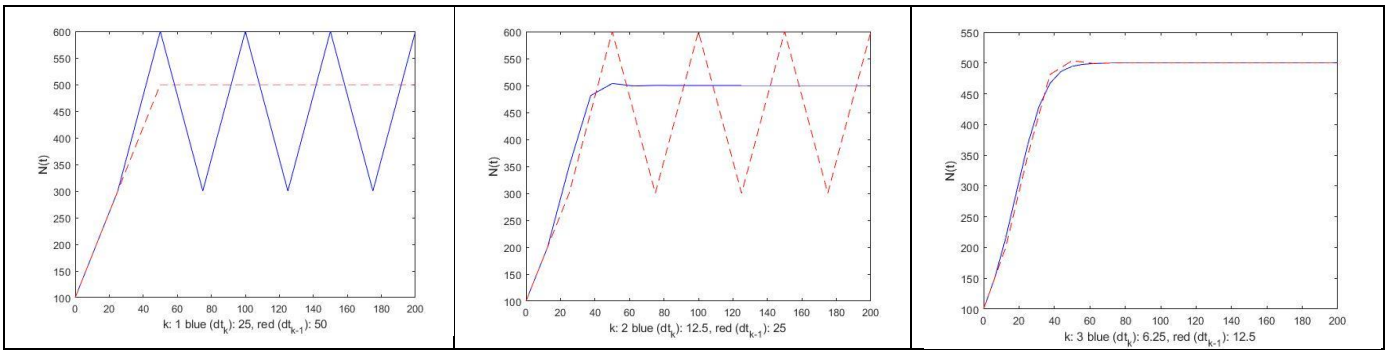
Yevonnael Andrew - 000546755
Assignment (ODE) - BK70A0600

Exercise 4.4: Find an appropriate time step; logistic model

We want to find the numerical solution of the logistic model, with the repeatedly halved timesteps: $\Delta t_k = 2^{-k}\Delta t, k = 0,1, \ldots$ At each iteration, it will plot both last times two steps Δt_k and Δt_{k-1} in the same plot. The program will ask for confirmation from the user whether it needs to continue or to end the iteration.

Extending from *logistic.m*, we need to add a *while* loop scheme with a *break* scenario when the user answer 'n' for no. Finally, these are plot for $f(t) = 0.1 \times \left(1 - \frac{u}{500}\right) \times u$ where $u(0) = 100$. Here are the plots for the first three iterations produced by the code. We can see that this function already converges nicely in the third iteration plot.

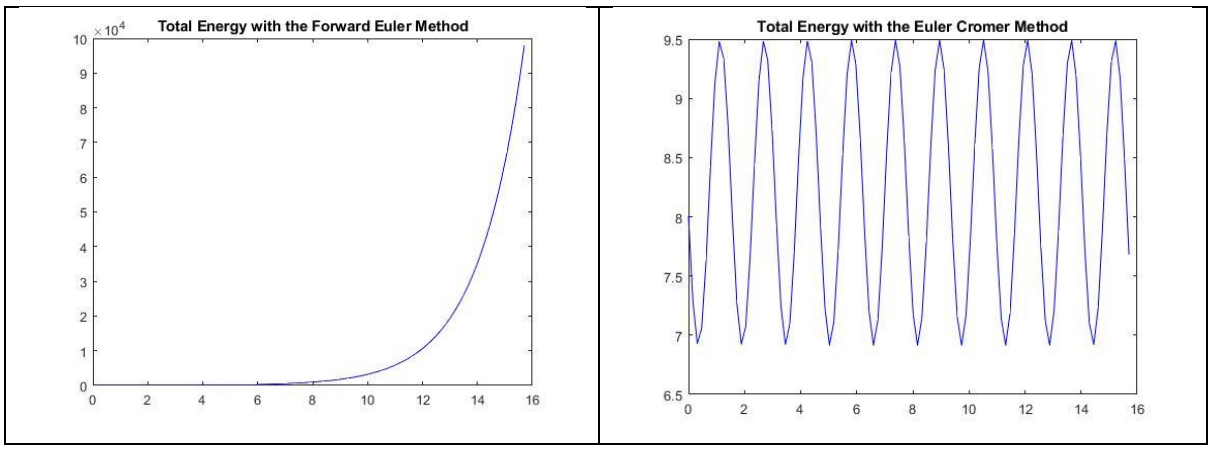


Exercise 4.10: Compute the energy in oscillations

We want to create a function called from the Forward-Euler and the Euler-Cromer solver codes to calculate the potential energy and kinetic energy of an oscillating system and plot the sum of them.

The formula is straightforward, that is $PE = \frac{1}{2}\omega^2 u^2$ and $KE = \frac{1}{2}ku^2$ And $TE = PE + KE$. We put the formula into one file called *osc_energy.* In the solver codes, *osc_FE,* and *osc_EC,* inside the primary function, we first initialize an empty array to store the energy at each timestep (we will calculate and store the amount of total energy at each timestep at the array). Finally, call the function and plot the stored array.

The code produces plots as follows.



As illustrated by the plots, the Forward Euler method shows that the energy is increasing with time. It is intuitively correct because Forward Euler gives growing amplitudes over time. In the Euler-Cromer methods, the energy fluctuates with time, but it does not increase nor decrease with time.

Exercise 4.14: Use of Backward Euler scheme for oscillations

From eq 4.84 and 4.85, we have a system of two algebraic equations for two unknowns:

$$u^n - \Delta t\, v^n = u^{n-1}$$

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

First, we need to solve for u^n and v^n. Rearranging $u^n - \Delta t\, v^n = u^{n-1}$ as $u^n = u^{n-1} + \Delta t\, v^n$ and plug into the second equation, we have $v^n + \Delta t\, \omega^2\, (u^{n-1} + \Delta t\, v^n) = v^{n-1}$. Pulling out v^n we have:

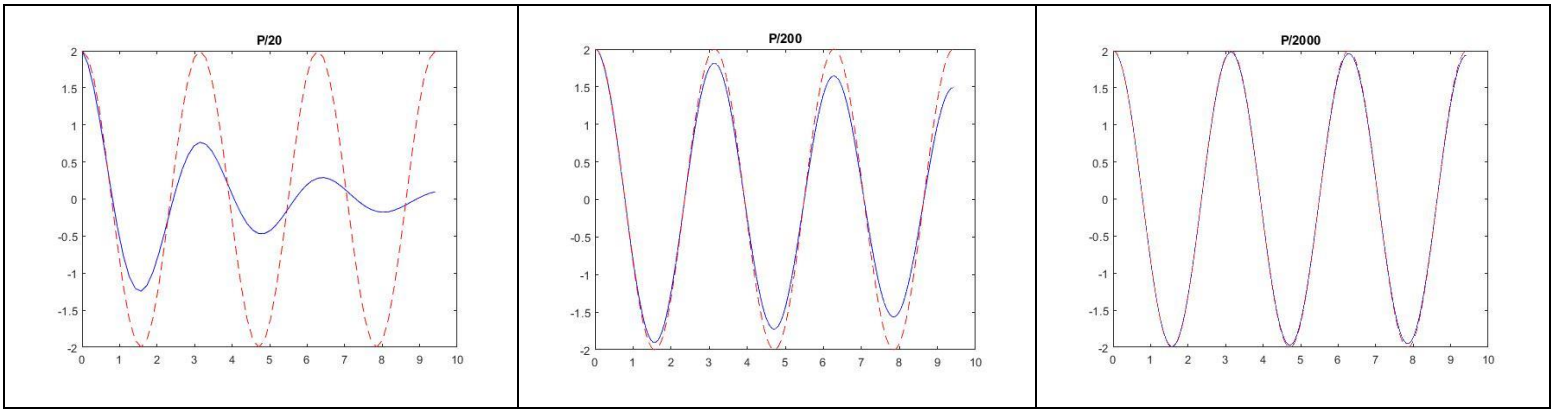
$$v^n + \Delta t\, \omega^2\, u^{n-1} + \omega^2 \Delta t^2\, v^n = v^{n-1}$$

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

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

And for the u^n we have:

$$u^n = u^{n-1} + \Delta t\, v^n$$



We don't need to plug v^n into the u^n algebraically as we can do it automatically in the code.

After we found the formula for both u^n and v^n, we implement them into the code. Finally, using three different dt, P/20, P/200, and P/2000, we obtain the results as follows.

We can see that from the three plots, the result improves significantly as we decrease the dt. In P/2000, which means 2000 time steps per period, we can see that the numerical solution is almost identical to the exact solution.

In Backward Euler, the solution amplitudes are damped. It is the opposite of the Forward Euler, in which the amplitudes are growing.