Report

Module Design

There are four classes in this program: *Main, ODE_solver, Simulator, Testor*. During the implementations of these classes, I import some codes from my homework-4, for example, the class *Vector*. In the following report, I will describe these modules in more detail.

The class *Main* is used to do the validation and simulations according to the instructions of the handout. Thus, this class will generate the corresponding results of the validation and simulations. However, since the limitation of our implementation of the ODE solver, each time we want to do the simulation for a different system, we need to comment current codes first.

The class *ODE_solver* is the core module of this program. There are two fields *rank* and *constructed* in this class. The rank records the desired rank of output. The value can be set by the users, which make this solver to be rank flexible. However, although the rank of the solver can be set by the users, the rank must be equal to the number of functions stored in the *fxDirectory*. Only if this condition is satisfied, the field *constructed* can be set to "True". And only if this field is "True", the ODE solver will solve the ODEs with the input initial conditions. Thus, each time we try to simulate a different system, we need to change the *fxDirectory* first. This solver can solve the ODEs in three different one-step methods: Forward Euler, RK4 and RK34 with time adaptation. In order to make this solver rank flexible, all of these methods can accept vectors as input and output results in a vector. Finally, we implement a method *solve()* which can solve the ODEs with one of these three methods according to the instruction of the users.

As for the class *Simulator,* which is used to store the initial conditions for each time of simulation. The initial conditions will include the initial time, initial values of variables, time step size and how long we should simulate. The method *simulate()* can be used to call the *ODE_solver* to simulate the system with the initial conditions and output the corresponding results.

Finally, the class *Testor* is designed to do the validation for the *ODE_solver*. The strategy of validation is to solve an ODE with known ground truth. Thus, there are three methods in this class. One method compute the ground truth, one method calls the methods in *ODE_solver* and the other method will compute the relative error.

Results Discussion

The results of the task 3: After implementing the rank flexible ODE solver, I did the according to the instructions of the hand out. And the results are included in the folder *Validation*. From the results, it is easy to find that the error is accumulated. And the method of RK34 with time adaptation is the most accurate, while the accuracy of Forward Euler is the lowest.

The results of the task 4: In this task, we used the ODE solver to simulate a simple RC circuit given parameters and initial conditions. The results are included in the folder *RC_simulation*. According to the results, no matter which method we choose, the results will present two oscillation waves as expected. The period cycle of the signal is about 20ns. This can also be an evidence that our simulation is correct since the time constant given here is about 10ns.

Another observation is about the results after increasing the simulation time step size from 0.2ns to 1.0ns. Since we can estimate the error by comparing the results of different methods with different orders of accuracy. Thus, we observe the difference of the plots of Forward Euler and RK4. We found that these two plots will become more different from each other after increasing the time step size. This implies that the error will increase if we simply increase the time step size.

The results of the task 5: In this task, we used the ODE solver to simulate an amplifier equivalent circuit given parameters and initial conditions. The results are included in the folder *Amplifier_simulation*. From the results, it is easy to observe that ΔV_1 is amplified to ΔV_2 and $V_2(t)$ is bounded between 0 and 5V. This observation satisfies the physical laws that this circuit should meet.

As for the observations among different methods and different time step sizes, we can get the results similar to task 4.