Haritha Murali (hm535)
Ariana Bruno (amb633)
ECE 4960 – Programming Assignment #4: Compact SPICE for ODE Solution

ANALYSIS REPORT

Ordinary Differential Equation Solvers

Overview

This programming assignment focuses on building different types of solvers for finding the solutions to ordinary differential equations (ODEs). Three methods (Forward Euler, standard Runge-Kutta 34 (RK34) and time-adaptive Runge-Kutta (RK34A)) are used to build the solvers.

Overall Analysis

Code Reuse

The program functions were built in an incremental fashion, with suitable tests built into each relevant file. We maximized code reuse by importing functions from previous programming assignments, as indicated in the comments of the programs.

Testing

System Functions

The outputs of each system function were compared to Hacker Practice implementations and to additional MATLAB implementations to verify that identical values were obtained, within a certain tolerance limit.

Solver Functions

All the solvers were first tested on a simple exponential function for which the ground truth is known. Subsequently, they were implemented on the system equations obtained from the Simple RC circuit and the Amplifier circuit. Since the ground truth for these functions are not readily known, a secondary implementation on MATLAB was used to compare values and verify values.

All files for the secondary MATLAB implementation can be found in a sub-folder in the repo.

Ease of Function Calls

Since the solver steps are identical for each system, with the exception of the system function call itself, we decided to build generic solver functions that will be able to take in a void function pointer to any system with the standard function signature given by `(phi , current_values , time , march)`. Future users will be able to use the solvers for any custom defined system as long as it follows the given function signature.

Furthermore, instead of running a different function call for each type of solver, we created a "parent" ODE Solver function that allows the user to pass in the type of solver and the function

to be solved as parameters. For example, when considering the simple RC circuit, and wanting to solve this system's ODE using the Forward Euler method, the function would look like:

```
ODE_Solver(simpleCircuit, slope, current_values , time , march)
```

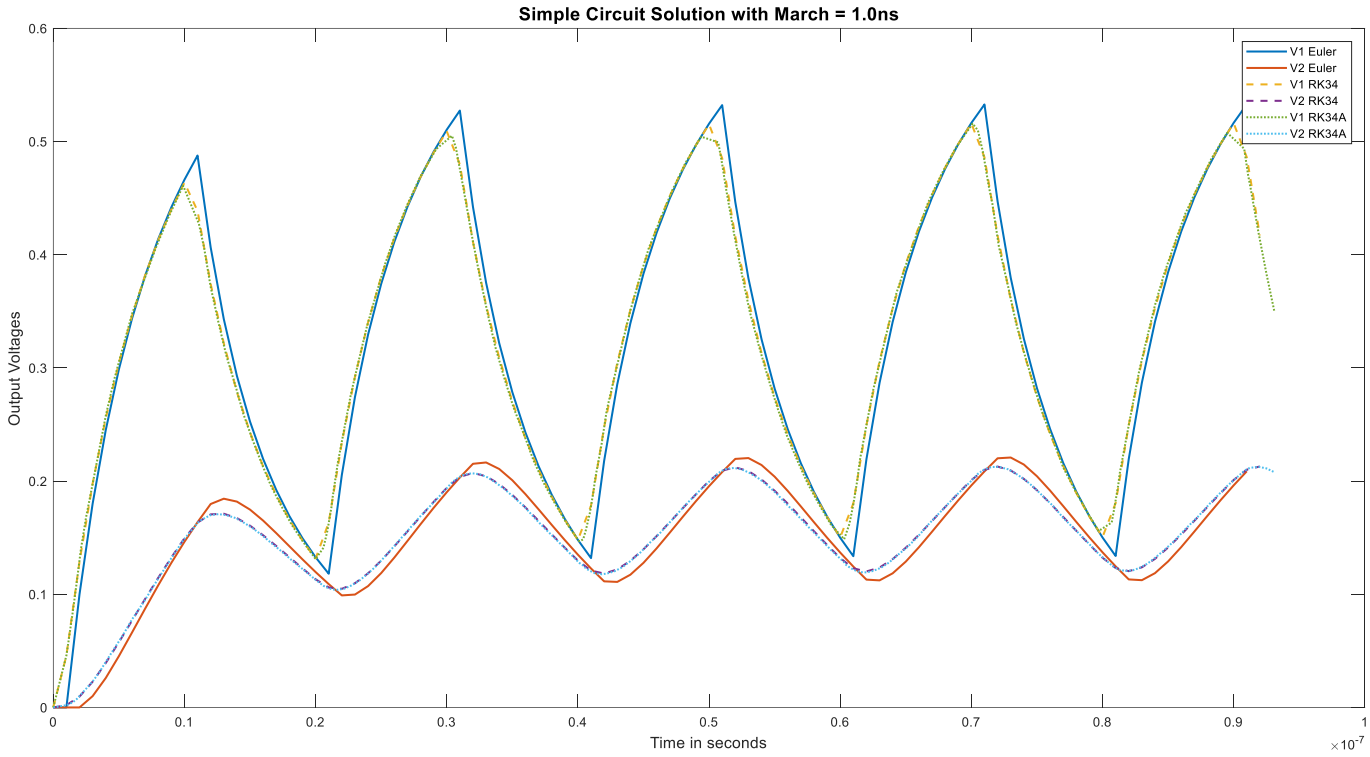
and the `simpleCircuit` function pointer would be defined as such:

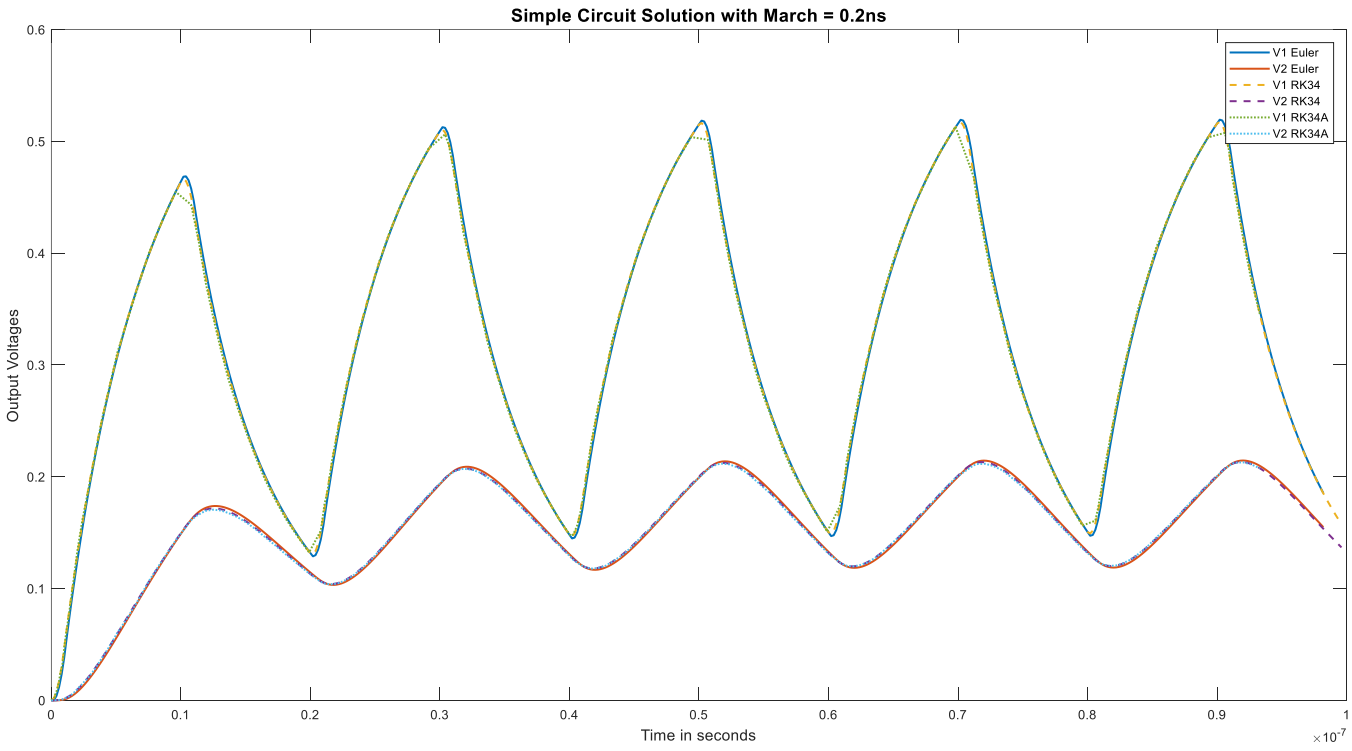
```
(*simpleCircuit)(vector*, vector*, double , double) = simple_RC_circuit
```

where `simple_RC_circuit` would be the originally defined function which takes in the same type of variables as given in the function pointer definition.

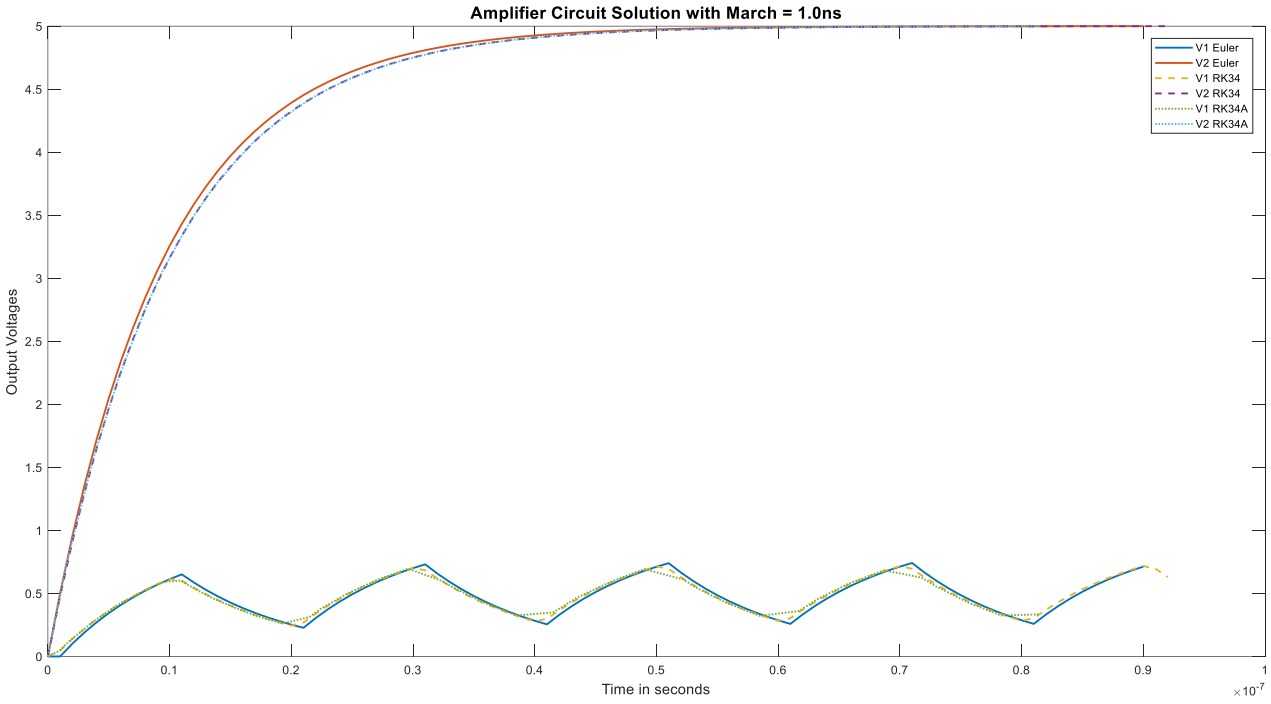
Haritha Murali (hm535)
Ariana Bruno (amb633)
ECE 4960 – Programming Assignment #4: Compact SPICE for ODE Solution

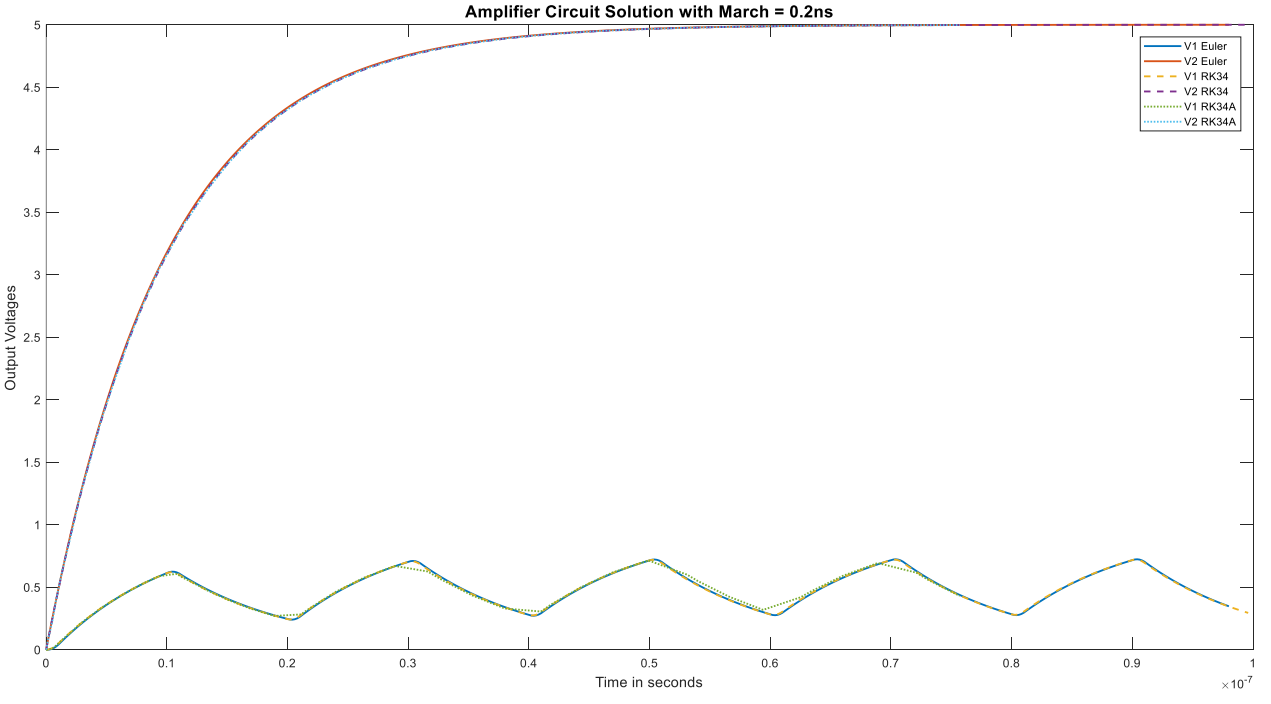
Graphs and Results





Haritha Murali (hm535)
Ariana Bruno (amb633)
ECE 4960 – Programming Assignment #4: Compact SPICE for ODE Solution





Based on the graphs, it can be easily seen that the Forward Euler method seems to be offset from the RK34 results. Based on the test results and overall knowledge about the ODE Solvers, the Forward Euler method would not have as accurate of results as the RK34 method. The difference in accuracy between the two methods, is likely the cause of the shift results seen from the Forward

Euler results. This behavior is seen in both the simple RC and amplifier circuits. However, the offset seems more pronounced in the simple RC circuit. The Forward Euler appeared to improve in accuracy with the smaller step size.

The RK34 results appears to be more stable with the change in step size with not a visibly noticeable improvement in accuracy. The RK34 method also appears to remain stable between the two different step sizes. This makes sense, because the method will adapt the time step to adjust to the performance of the RK34 accuracy. This at least results in potentially fewer steps if not much accuracy is being gained with an initially small time step or will result in more time steps in order to gain better performance accuracy. For instance, for the Amplifier circuit comparing the RK34 and RK34 adaptive with and initial time step of 0.2 ns, the adaptive method results in few steps than the normal RK34 method. In contract when considering the simple RC circuit, with an initial time step of 1 ns, the adaptive RK34 method results in many additional steps compared to the RK34 method without adaptive time.