ECE 4960: Computational and Software Engineering
Programming Assignment 4: Compact SPICE for ODE Solution
Submitted by: Pragya Sharma (ps847)
Github: https://github.com/psharma15/ECE4960/tree/master/Lab4

Programming Environment:

Language: Matlab, R2013a
System Type, OS: 64-bit Windows 10 OS, x64-based Intel® Core™ i7-6700HQ CPU @ 2.6GHz processor
RAM: 8 GB
Model: Acer Aspire VN7-592G

Generic ODE Solvers:

A library of solvers is built to solve ODE. This includes Forward, Backward, and Trapezoidal Euler and RK34.

Backward Euler (and also trapezoidal) have been implemented in two ways, one for linear equations and other for non-linear equations, when x_{i+1} cannot be solved by taking everything to the RHS of the equation. The functions are `backwardEulerSym.m` and `backwardEulerNewton1.m` for linear and non-linear respectively. Newton method is used to solve non-linear equations. Additionally, predictor and corrector based methods have also been implemented(`backwardEulerPC.m`), but the results are not discussed in this report.

RK34 also has different implementations. `rk34.m` is without time adaptation; `rk34Adaptive.m` is with time adaptation for the next step; and `rk34Adaptive2.m` is for both forward and backward time adaptivity.

Solvers – Input and Output:

The solvers are designed to be independent of the ode function, which can be a vector. It takes standard inputs, which are explained in every function. All the solvers need at least the following inputs:

- tmin: Starting time
- tmax: Ending time
- delT: Time step
- x0: Initial solution column vector
- ode: Function handle to the ode function for the problem that needs to be solved.

Additionally, backward and trapezoidal Euler, and RK34 may need following inputs:

- xNew: Function handle to get x_{i+1} on LHS and solve for RHS
- dOde: Function handle to get Jacobian, needed for Newton method
- epsilonR: Normalized error tolerance

The output of all the solvers is the solved x values with time. RK34 may additionally give error estimation and new time adaptive time steps.

Utilizing Solvers:

The solvers have been applied in three settings – validation, linear RC circuit and non-linear transistor circuit. The main files for these are `validateODE.m`, `circuit1ODE.m` and `circuit2ODE.m`.

These scripts define all the input parameters and plots the requied outputs which are discussed in the next section.

Tasks:

1. **Validation**

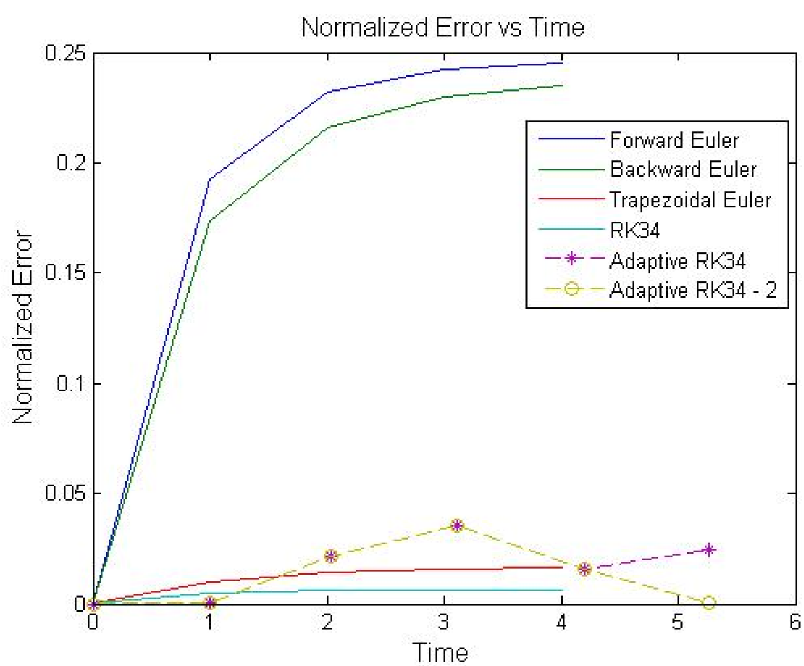
Validation has been performed by checking accuracy of solutions for the ode:

$$\frac{dx}{dt} = 4\,e^{0.8t} - 0.5x$$

Where, the true solution is already known to be:

$$x(t) = \frac{4}{1.3}\left(e^{0.8t} - e^{-0.5t}\right) + 2e^{-0.5t}.$$

The initial solution is $x(0) = 2$.
Since it is a linear ODE, backward and trapezoidal euler algorithms are used with x_{i+1}arranged as: $x_{i+1} = Equation$.



<u>Observation:</u>
Forward euler has maximum error, followed by backward euler. Trapezoidal euler has less error than adaptive RK34 methods, and RK34 without adaptaion has least error. Number of

time steps for adaptive RK34 is same as RK34, and has slightly more error, indicating that the default time step is the best time step for this method in terms of both accuracy and efficiency.

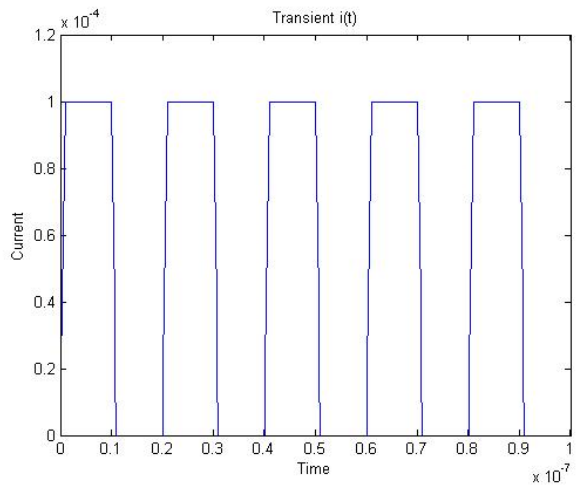
2. Circuit 1: RC circuit

This is the circuit shown in Fig.3 of the lab handout. Following vectorized ode is solved:

$$\frac{d}{dt}\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} -\left(\frac{1}{C_1R_1}+\frac{1}{C_1R_2}\right) & \frac{1}{C_1R_2} \\ \frac{1}{C_2R_2} & -\left(\frac{1}{C_2R_2}+\frac{1}{C_2R_3}\right) \end{pmatrix}\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} + \begin{pmatrix} \frac{i(t)}{C_1} \\ 0 \end{pmatrix}$$

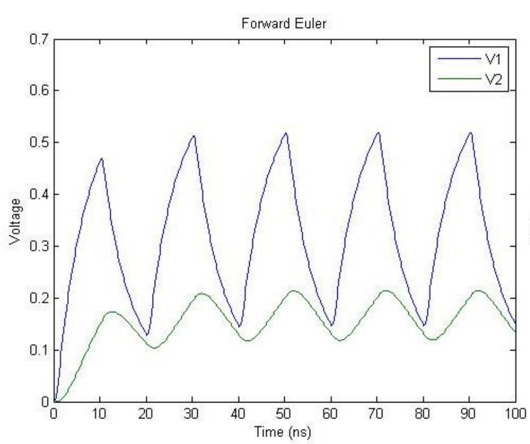
The initial solution is [0;0]. Since it is a linear ODE, backward and trapezoidal euler algorithms are used with x_{i+1}arranged as: $x_{i+1} = Equation$.

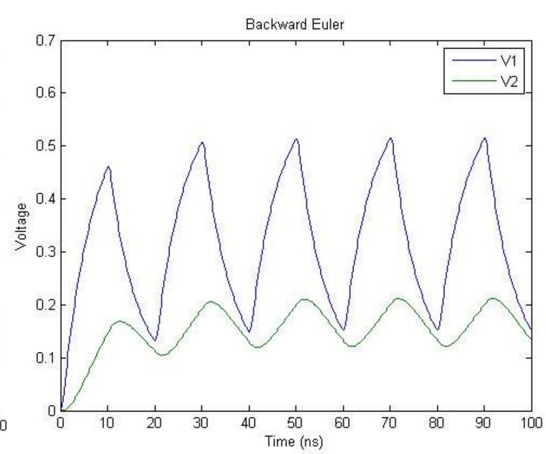
The current plot is as shown:

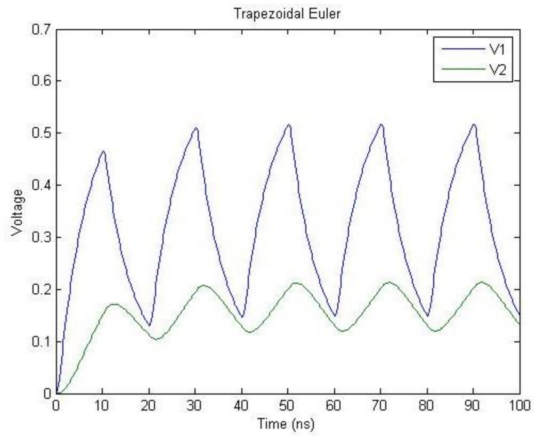


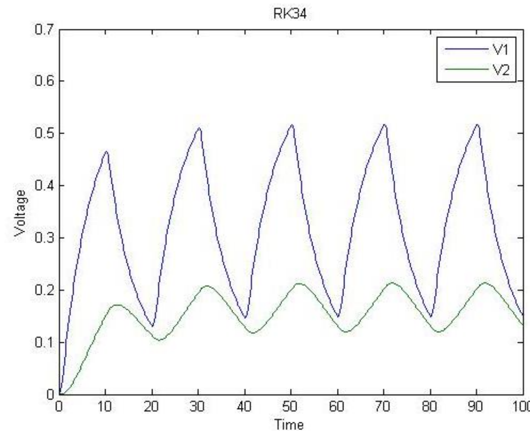
Results and Observations:

ODE solution with $\Delta t = 0.2e^{-9}sec$, ie, 501 Time steps:

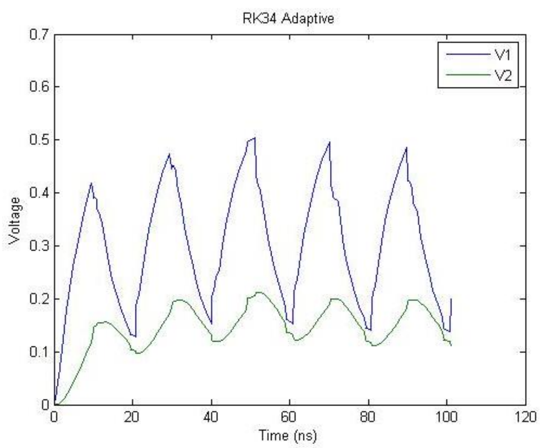


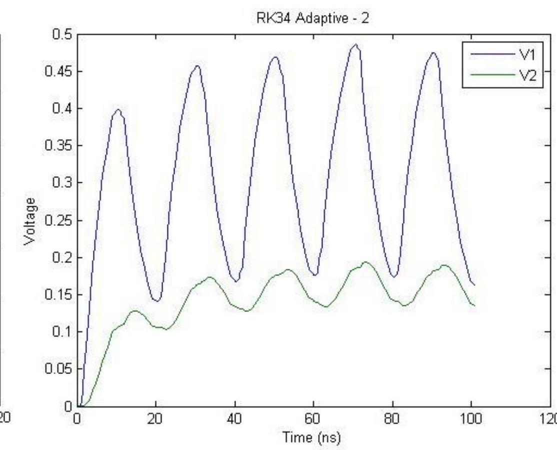






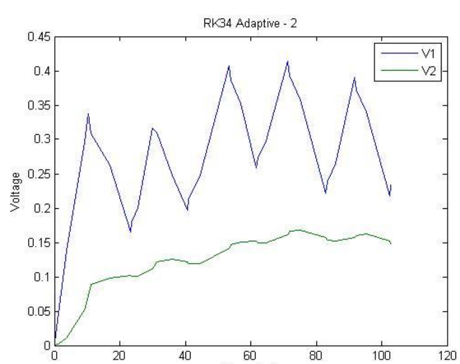
RK34 Adaptive (only next step prediction) is calculated with 189 time steps and RK34 Adaptive 2 (both current and next step prediction) is calculated with 301 time steps:

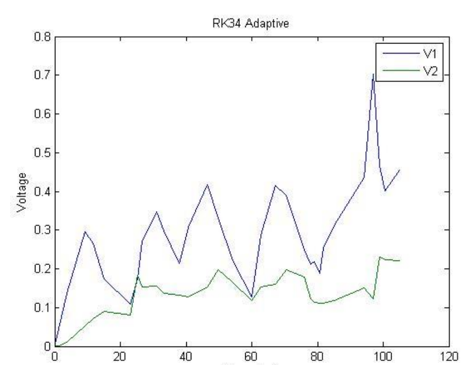


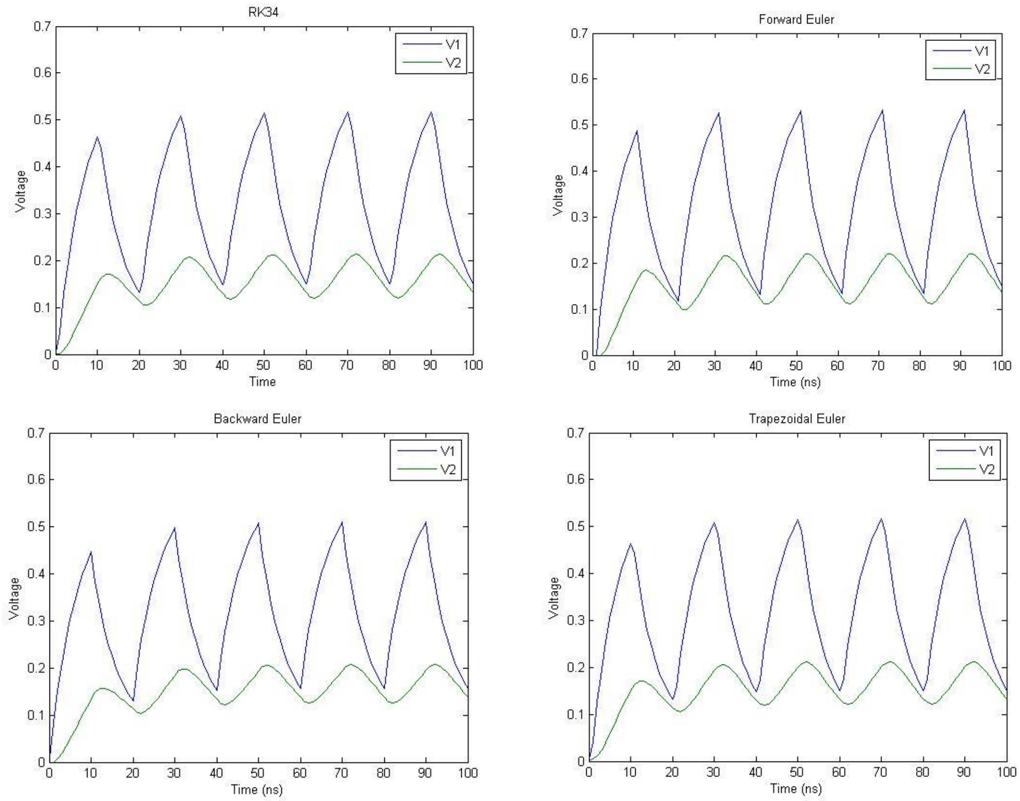


`epsilonR` is 1e-4. True accuracy is unknown, but compared to other solvers, this RK34 solver needs to be further improved, although RK34 Adaptive-2 is giving slightly better results.

With 1ns as the initial delta, results of all the ODEs are very close to the above plots, except adaptive RK34 methods:







3. Circuit 2: Transistor circuit

This is the circuit shown in Fig. 5 of the handout. Following ode is solved:

$$\frac{dV_1}{dt} = f_1(V_1, V_2) = -\frac{1}{R_G C_1} V_1 + \frac{i_{in}(t)}{R_G C_1}$$

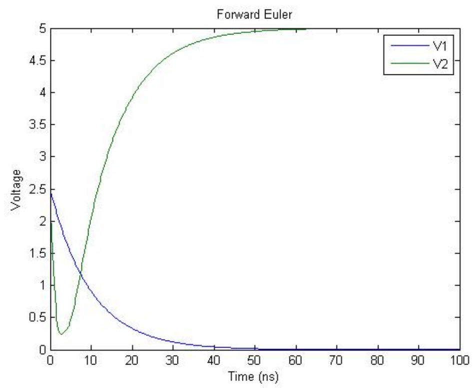
$$\frac{dV_2}{dt} = f_2(V_1, V_2) = -\frac{I_{D,EKV}(V_1, V_2)}{C_2} - \frac{1}{R_L C_2} V_2 + \frac{V_{DD}}{R_L C_2}$$

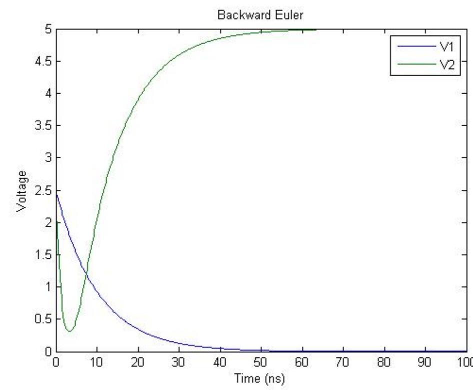
The initial solution is [2.5;2.5]V. This is non-linear, hence backward euler and trapezoidal euler are solved using Newton's method. This method needs Jacobian of the equation to be minimized, which is calculated numerically (centered differentiation).

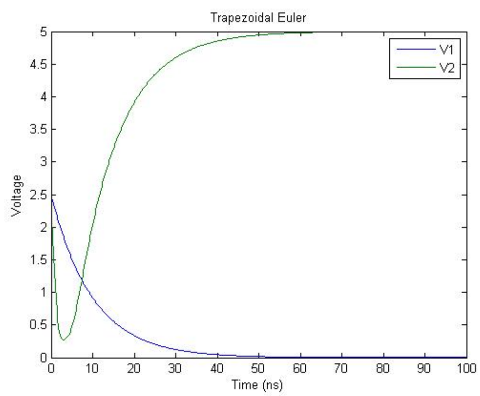
Results and Observations:

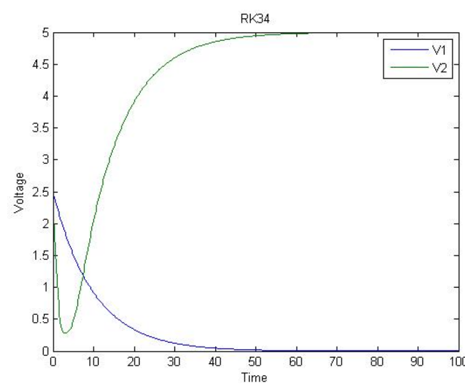
The ODE is solved with time step of 0.2e-9 sec, ie, 501 time steps for time between 0 to 100ns. RK34 Adaptive and Adaptive-2 uses 13 and 12 time steps respectively. Adaptive time methods are able to get the general shape in much less time steps, but care need to be taken to increase the accuracy and smoothness of the plot generated.

Following are the results:

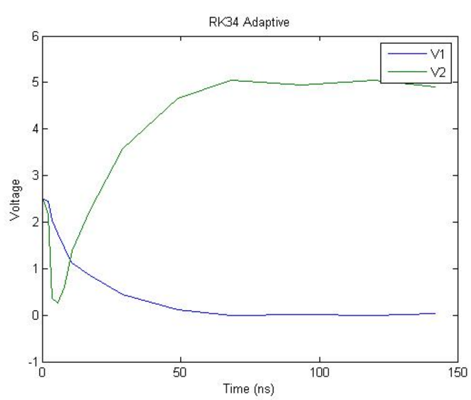


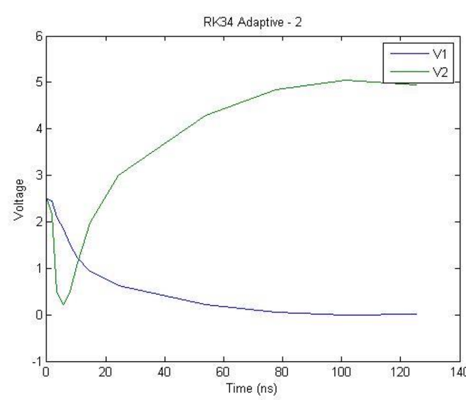






epsilonR is 1e-2:





epsilonR is 1e-3: