Report of Project 4: Compact SPICE for ODE Solution

Generic ODE Solvers:

Within this project, ODE solvers using Forward Euler, RK4, RK34(with time adaption) methods are designed to simulate the RC and Amplifier circuits. Solvers are designed to be able to take in any arbitrary dimensions of input vector, and able to interface with any ODE functions by indicating the function type. The solvers input form:

`Vector x`: the unknown variables;
`double t`: the starting time point for estimation;
`double h`: the step length;
`String[] fType`: the ODE function type for each unknown variable;

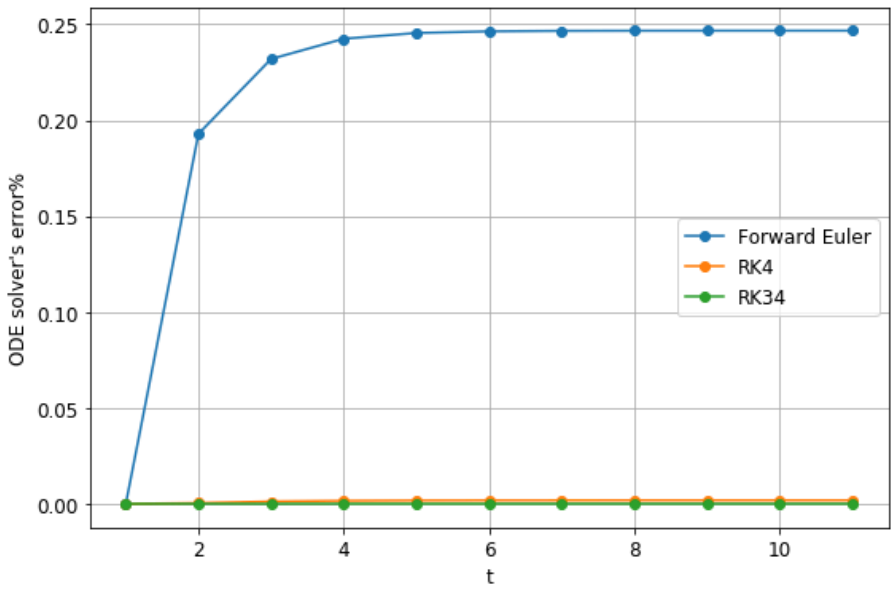
Task 1, 2: Design and Inheritance

The program breaks down into 4 java scripts (Solvers.java, Simulation.java, Test.java, and Main.java) and inherits 2 previous-written helper classes (Vector.java, FileIO.java). The inherited classes' path was built together with this project to make sure they can run interchangeably. Also, the 1-d generic ODE solver from previous Hacker Practice is modified to take input of a Vector.

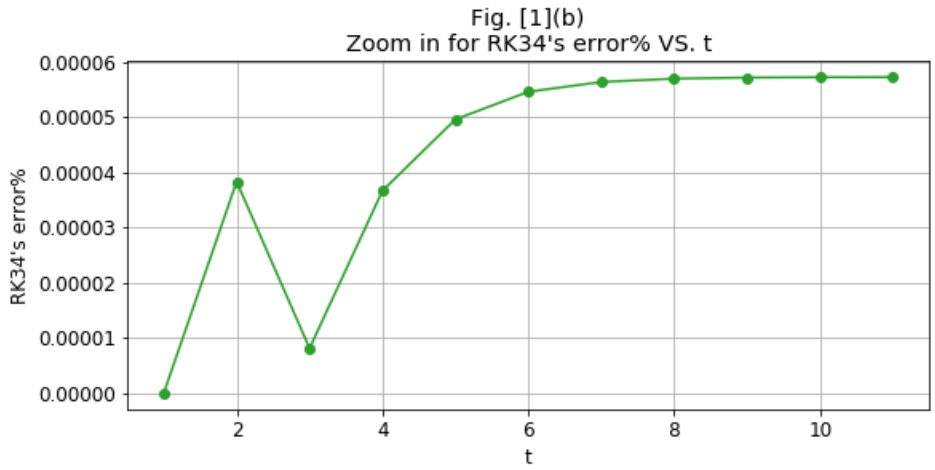
Task 3: Validation for Generic ODE Solvers

The ODE solvers are tested with the example given in previous Hacker Practice in Note 6, page 23, where the ground-truth is given to compute the normalized relative error $|\varepsilon_x|$. Run solvers starting from t = 0s to t = 11s, time step = 1s, and compute $|\varepsilon_x|$ within each time step:

Fig. [1]
ODE solver's error% vs time



Observation: The $|\varepsilon_x|$ of Forward Euler and RK4 methods accumulates with time, and it could be seen that forward Euler has worse accuracy compared to RK34 (with time adaption) and RK4, because Forward Euler is only first-order accurate, while RK4 and RK34 are forth-order accurate. The $|\varepsilon_x|$ of RK34 is too small to tell the trend, so zoom it in as:



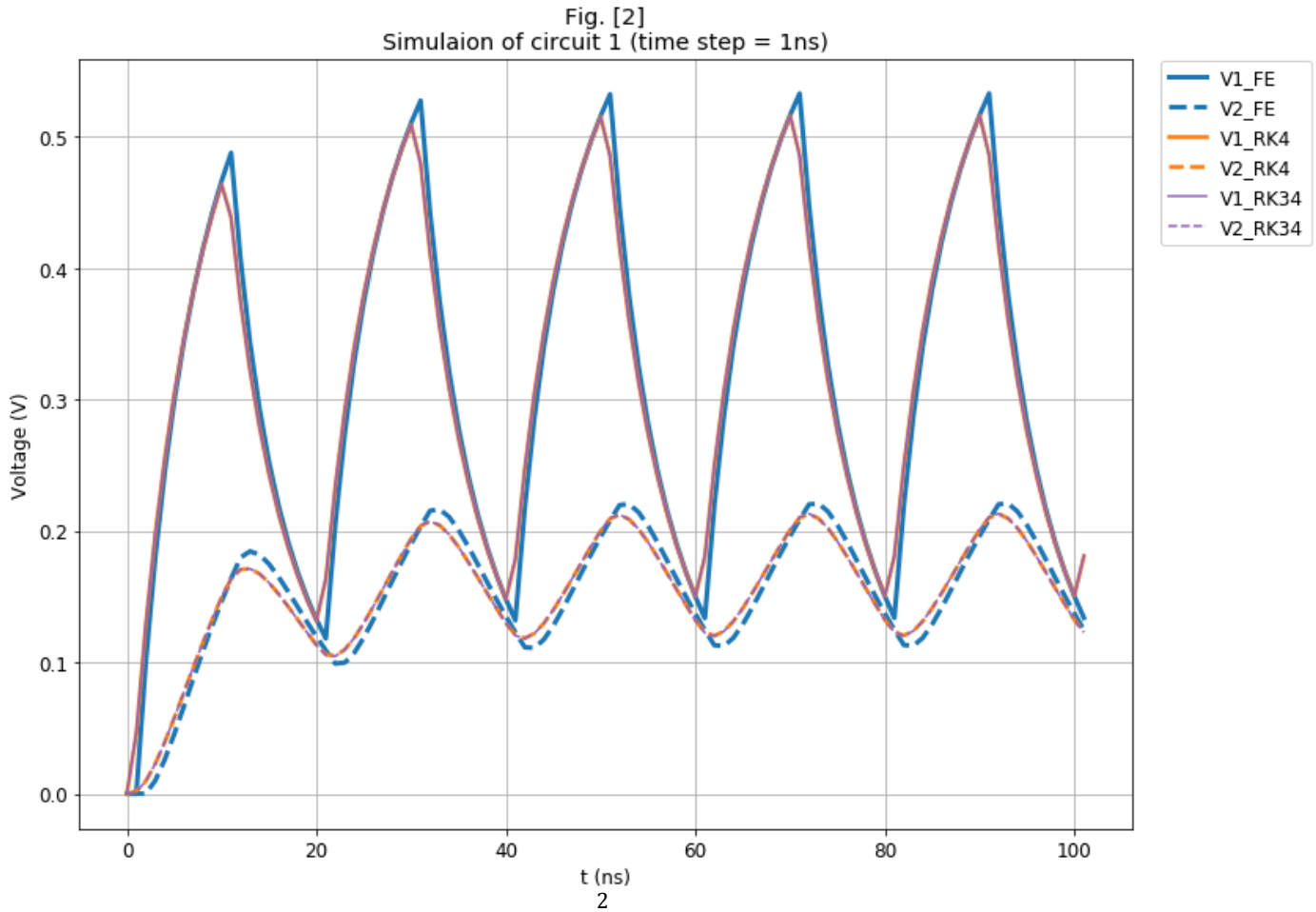
It could be seen that RK34's error as a whole is also accumulated with time.

- **Task 4: Simulation for RC Circuit's V_1, V_2:**

The (V_1, V_2) of RC circuit in handout's Fig. 3 could be expressed as:

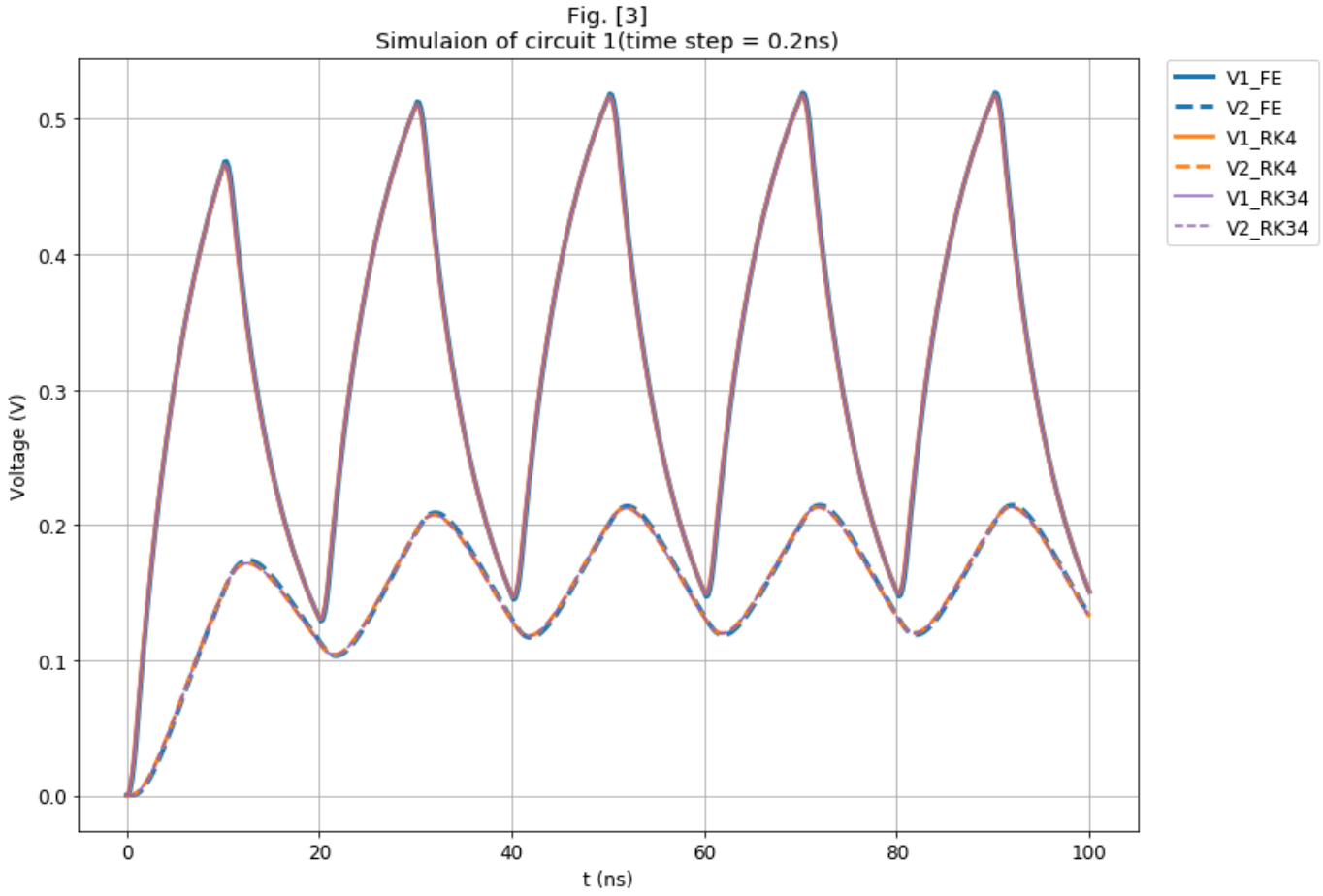
$$\frac{d}{dt}\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} -\left(\frac{1}{C_1R_1}+\frac{1}{C_1R_2}\right) & \frac{1}{C_1R_2} \\ \frac{1}{C_2R_2} & -\left(\frac{1}{C_2R_2}+\frac{1}{C_2R_3}\right) \end{pmatrix}\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} + \begin{pmatrix} \frac{i(t)}{C_1} \\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

The initial $(V_1, V_2) = (0, 0)$. For RK34, the absolute error's tolerance $e_A = 1e-4$, and the relative error's tolerance $e_R = 1e-7$. **When $\boldsymbol{timestep = 1ns}$ ($\boldsymbol{100ns\ in\ total}$):**



Observation: It could be seen that when $timestep = 1ns$, (V_1, V_2) curves generated by RK34 (With time adaption) and RK4 almost overlap with each other. Forward Euler estimates next step based on the rate of current points, therefore when time-step is big, the curve (the blue ones) would seem to be dragged behind than the other two higher-order accurate methods.

When $timestep = 0.2ns$ ($100ns$ in total):



Observation: It could be seen that when time step is finer, Forward Euler's next step estimation is more accurate (based on the known fact that RK4 and RK34 are higher-order accurate methods, and generate more accurate results) than when $timestep = 1ns$. The curves of three methods are smoother when time-step is smaller.

• Task 5: Simulation for Amplifier Circuit's

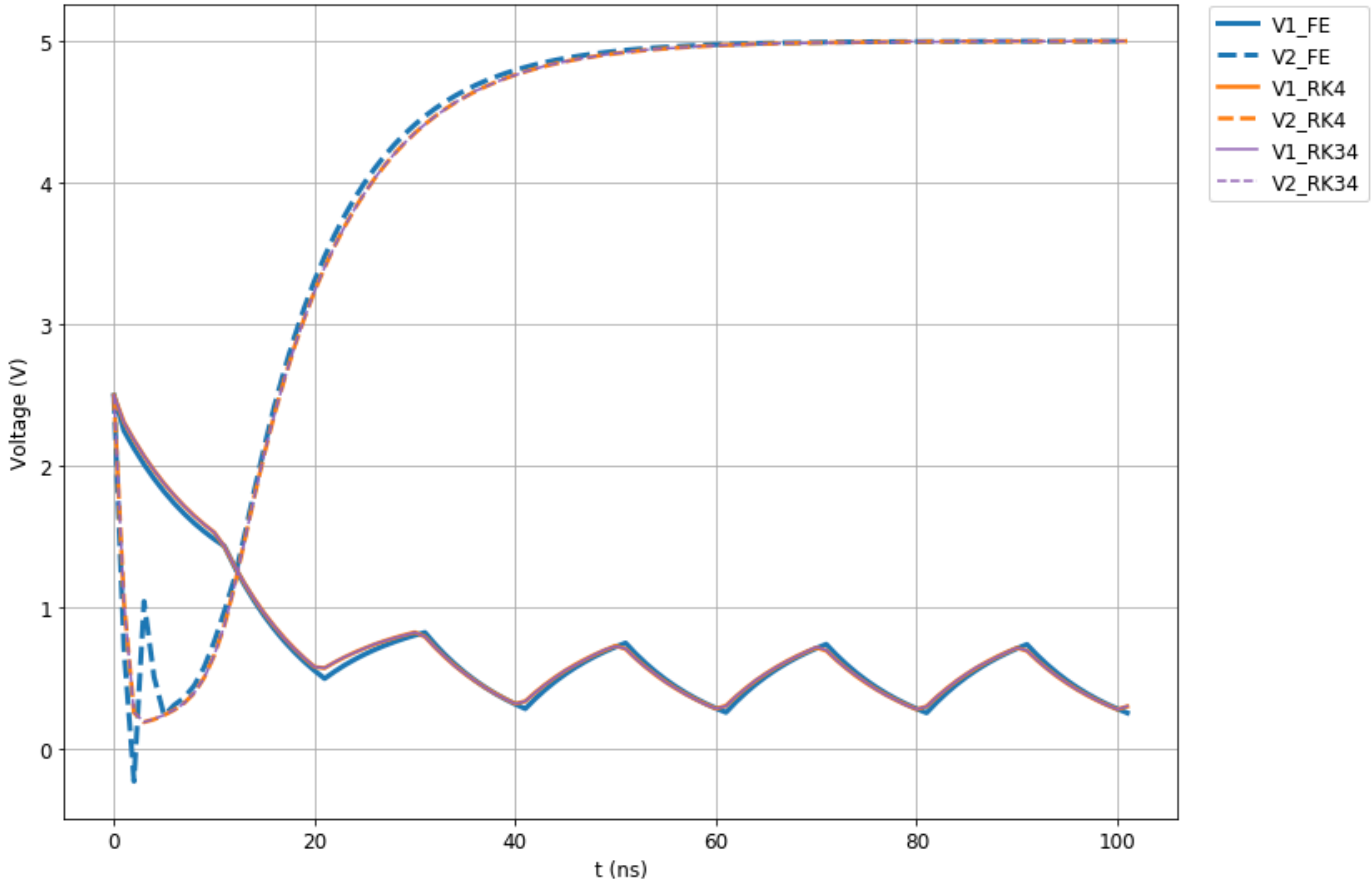
The (V_1, V_2) of Amplifier circuit in handout's Fig. 3 could be expressed as:

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

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

The initial $(V_1, V_2) = (0, 0)$. For RK34, the absolute error's tolerance $e_A = 1e-4$, and the relative error's tolerance $e_R = 1e-7$. **When $timestep = 1ns$ ($100ns$ in total):**

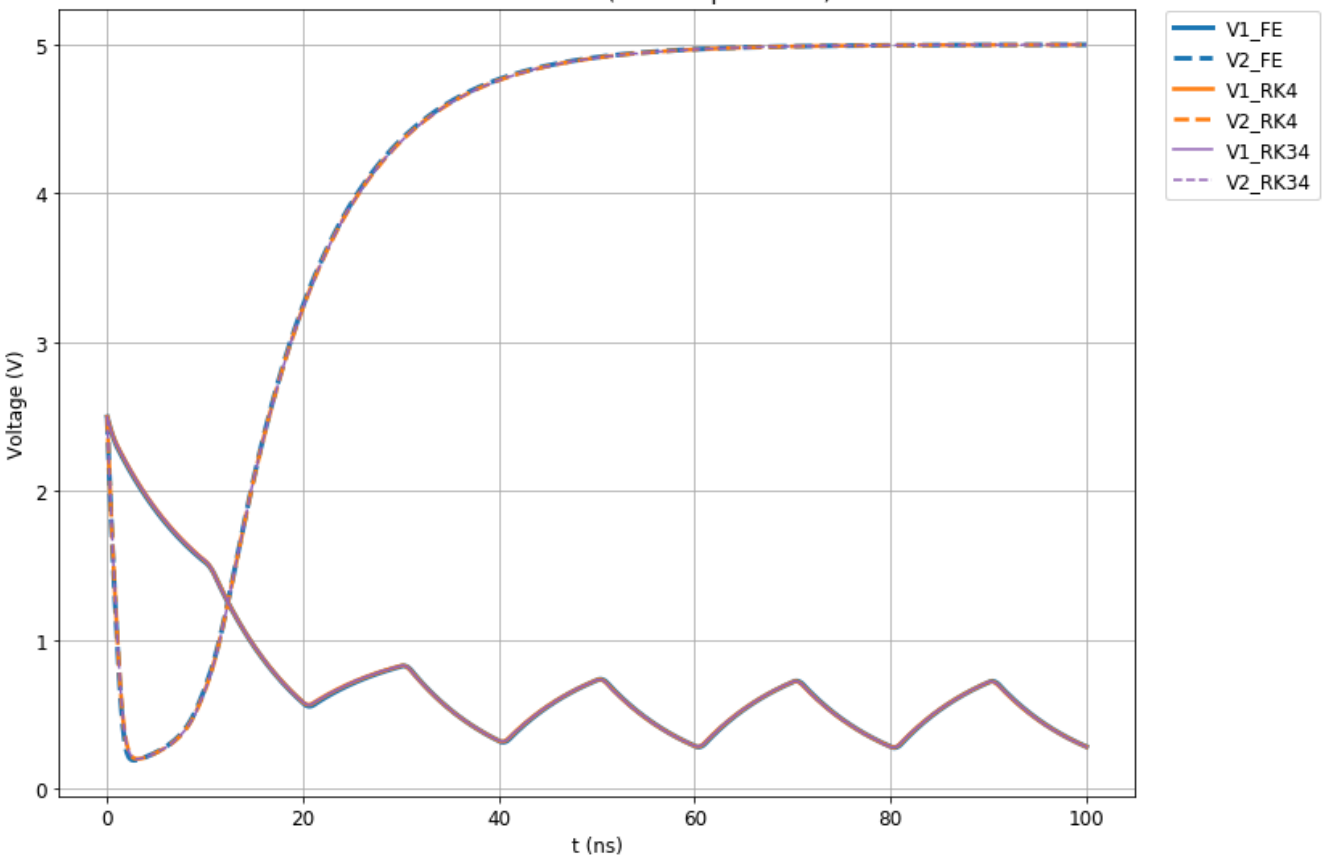
Fig. [4]
Simulaion of circuit 2(time step = 1ns)



Observation: Still, it could be seen when $timestep = 1ns$, (V_1, V_2) curves generated by RK34 (With time adaption) and RK4 almost overlap. At around $t = 1ns$, the Forward Euler (blue curve) oscillates greatly, which could be due to when the gradient of ground-truth changed from negative to positive, Forward Euler method is not able to immediately reflect this change, and would only react to it at next step when the ODE function directs it to change accordingly. Therefore when the gradient of ground-truth (simulated with RK34 or RK4) flips, Forward Euler would overshoot at these flipping points.

When $timestep = 0.2ns$ ($100ns$ *in total*):

Fig. [5]
Simulaion of circuit 2(time step = 0.2ns)



Observation: It could be seen when $timestep = 0.2ns$, three methods' curves almost overlap. The finer time step enables Forward Euler to react sooner to the gradient flips, therefore the oscillation at around $t = 1ns$ disappears, and its curve approaches higher-order methods. The curves of three methods are smoother when time-step is smaller.