Effects of Mutual Inductance

Colin Turner: 250956100

MME 9621: Computational Methods in Mechanical Engineering

Dr. Mohammad Z. Hossain

April 19, 2024

Problem

1. Consider an RLC circuit with a resistor of 10 Ω, an Inductor of 0.1 Henry, and a capacitor of 1 Faraday. This circuit is powered by an oscillating ac power source. Using Kirchhoff's law, the system can be described as $V(t)_S = A * \cos(\omega t)$, where amplitude is 60 and $w = 2\pi 60$.

$$V_S = V_{R1} + V_{L1} + V_{C1}$$

$$V(t)_S = R_1 \frac{di_1}{dt}(t) + L_1 \frac{d^2 i_1}{di^2}(t) + \frac{i_1(t)}{C_1}$$

A second unpowered RLC circuit with a resistor of 10 Ω, an Inductor of 0.3 H, and a capacitor of 1 F. would be described as,

$$0 = V_{R2} + V_{L2} + V_{C2}$$

$$0 = R_2 \frac{di_2}{dt}(t) + L_2 \frac{d^2 i_2}{di^2}(t) + \frac{i_2(t)}{C_2}$$

The introduction of this second unpowered, closed circuit, RLC circuit within close proximity to the first would create a mutual inductance of 0.09 H and would affect both systems. The Kirchhoff's law including mutual inductance for both circuits can be described as,

1. $V(t)_S = R_1 \frac{di_1}{dt}(t) + L_1 \frac{d^2 i_1}{di^2}(t) + \frac{i_1(t)}{C_1} - M \frac{d^2 i_2}{di^2}(t)$
2. $M \frac{d^2 i_1}{di^2}(t) = R_2 \frac{di_2}{dt}(t) + L_2 \frac{d^2 i_2}{di^2}(t) + \frac{i_2(t)}{C_2}$

Assuming the system has been inactive for a long time and all initial conditions are 0, however assume the first inductor retains a residual current stored of 0.01 amps. Evaluate from t = 0, 0.1.

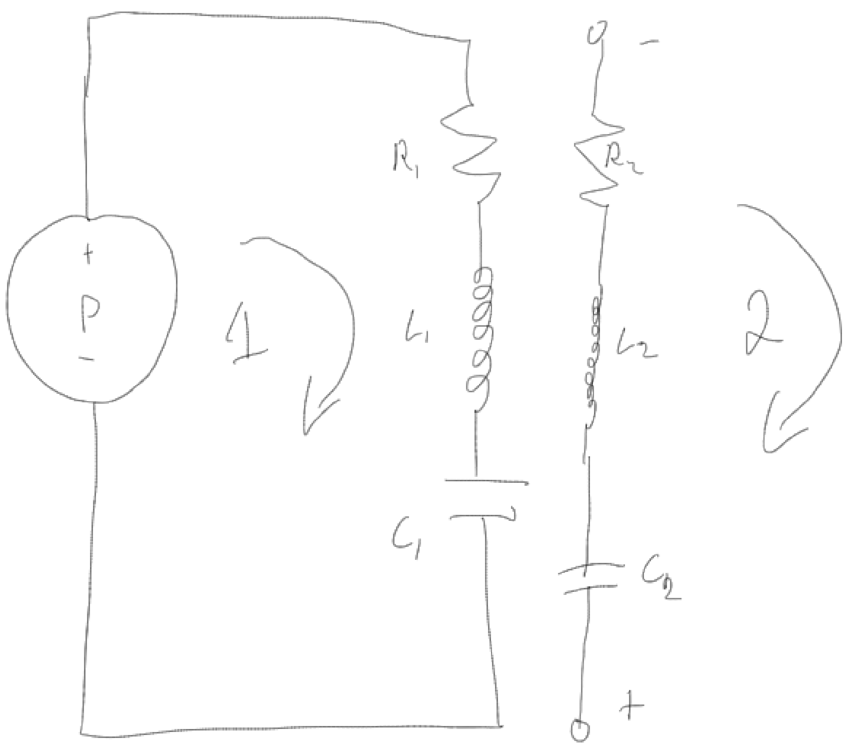


Figure 1: Circuit Diagram

a) Derive the equations by hand using Kirchhoff's Law.

b) Create the system of first order ODE's that describes the interaction of this system.

c) Solve using ODE45, ODE23tb & through RK-4.

d) Obtain a grid independent solution for RK-4.

e) Plot the results of the current, di/dt in both systems & the voltage induced in the second system.

f) Plot the resulting error of the methods. Comment on accuracy, cost of computing and determine which ODE solver works best for this problem.

g) Discuss the features of the MATLAB function & discuss how you process the equations to use in the function.

h) For your code show the detailed mathematical procedure and numerical algorithms & why this method was chosen.

i) Discuss the effect of TOL on cost of computing and overall results.

j) Discuss the physics of the problem by changing parameters.

Solution

Derivation of Equations

The equations presented in the MATLAB code are derived from Kirchhoff's voltage and current laws, which are fundamental principles in circuit analysis. Kirchhoff's voltage law states that the sum of the voltages around any closed loop in a circuit must be zero. Similarly, Kirchhoff's current law states that the sum of currents entering a node (or junction) in a circuit must equal the sum of currents leaving that node. The given system consists of two inductors (with mutual inductance), two resistors, and two capacitors with each component creating a separate RLC Circuit where one is powered by Vs and the other is powered solely off mutual inductance. Applying Kirchhoff's voltage law to each loop in the system results in two coupled, second order, linear, homogeneous differential equations that describe the behavior of the current in each circuit with respect to time. Further processing must take place to prepare the systems for analysis in MATLAB.

1. $V_S = V_{R_1} + V_{L_1} + V_{M_2} + V_{C_1}$

$$V_S = I_1R_1 + L_1\frac{dI_1}{dt} + M\frac{dI_2}{dt} + \frac{1}{C}\int I_c\,dt$$

$$\frac{dV_S}{dt} = R_1\frac{dI_1}{dt} + L_1\frac{d^2I_1}{dt^2} + M\frac{d^2I_2}{dt} + \frac{1}{C}I_1$$

2. $V_{In} = V_{L2} + V_{M_1}$

$$V_{in} = L_2\frac{dI_2}{dt} + M\frac{dI_1}{dt}$$

$$\frac{dV_{in}}{dt} = L_2\frac{d^2I_2}{dt} + M\frac{d^2I_1}{dt}$$

$$\frac{d^2I_2}{dt} = M\frac{d^2I_1}{dt} - \frac{dV_{in}}{dt}$$

Figure 2: Derivation of Equations

System of First Order ODE’s

Breaking the given second-order differential equations into a system of first-order ordinary differential equations (ODEs) is a common practice in numerical analysis, particularly when using MATLAB. This approach facilitates the use of MATLAB's built-in ODE solvers, which are designed to handle systems of first-order ODEs. By converting the equations into first-order form, we transform a single higher-order problem into a set of coupled first-order problems. Additionally, breaking the equations into first-order form often simplifies the implementation of boundary or initial conditions and allows for a more straightforward interpretation of the problem. Overall, breaking down the equations into first-order ODEs enhances the efficiency, accuracy, and ease of analysis when utilizing MATLAB's computational capabilities for solving dynamic systems like the one described in the circuit model. The figure below depicts the process leading to the system of first order equations.

$$\textcircled{1} \quad V_S = V_{R_1} + V_{L_1} + V_{M_2} + V_{C_1}$$

$$V_S = IR_1 + L_1 \frac{dI_1}{dt} + M \frac{dI_2}{dt} + \frac{1}{C_1}\int I_1 \, dt$$

$$V_S = R_1 \frac{dI_1}{dt} + L_1 \frac{d^2 I_1}{dt} + M \frac{dI_2}{dt} + \frac{1}{C_1} I_1$$

$$\frac{d^2 I_1}{dt} = V_S - R_1 \frac{dI_1}{dt} - M \frac{dI_2}{dt^2} - \frac{1}{C_1} I_1$$

$$y = y_1 \qquad \textcircled{1} \; y_2$$

$$y_1' = y_2 \qquad \textcircled{2} \; y_2' = (V_S - R_1 y_2 - M y_4' - \frac{1}{C} y_1)/L_1$$

$$V_{M_1} = V_{R_2} + V_{L_2} + V_{C2}$$

$$M \frac{dI_1}{dt} = IR_2 + L_2 \frac{dI_2}{dt} + \frac{1}{C_2}\int I_2 \, dt$$

$$M \frac{d^2 I_1}{dt} = R_2 \frac{dI_2}{dt} + L_2 \frac{d^2 I_2}{dt} + \frac{1}{C} I_2$$

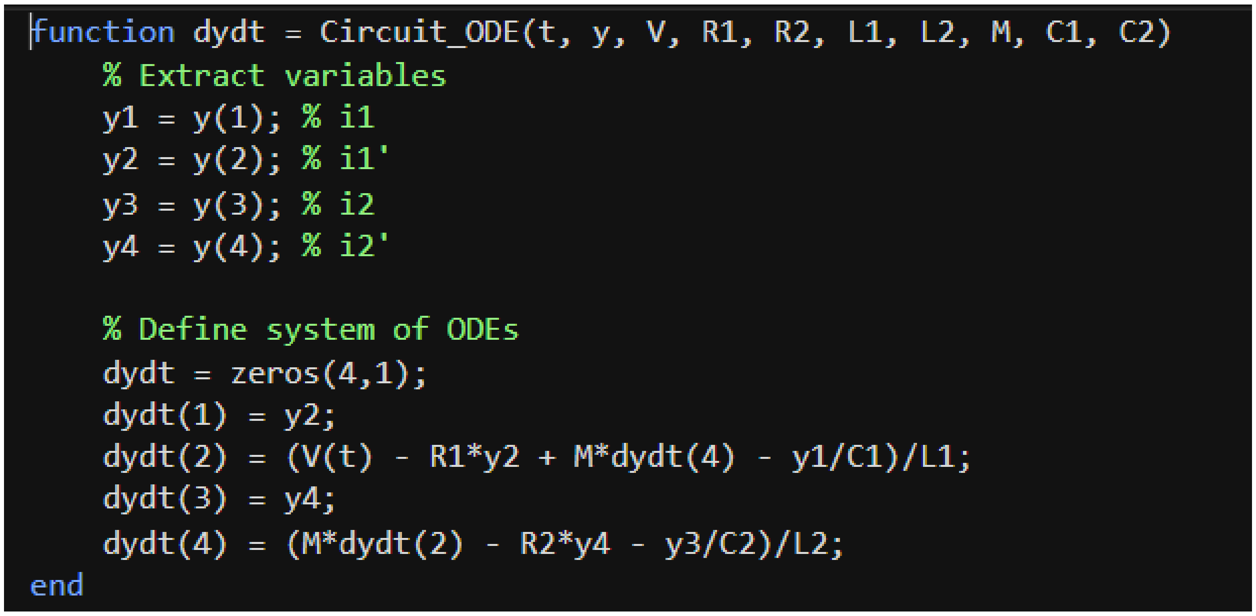
$$\frac{d^2 I}{dt} = \left(M \frac{d^2 I_1}{dt^2} - R_2 \frac{dI_2}{dt} - \frac{1}{C} I_2\right)/L_2$$

$$y = x_1 \qquad \textcircled{1} \; x_2$$

$$y_1' = x_2 \qquad \textcircled{2} \; x_2' = (M y_2' - R_2 x_2 - \frac{1}{C} x_1$$

Figure 3: System of First Order ODE’s

After the system was acquired, the next step was to create a MATLAB function to process the system. The following figure depicts the Circuit ODE function which was used to store the system of equations, y(1) = i1, y(2) = di1/dt, y(3) = i2 & y(4) = di2/dt.

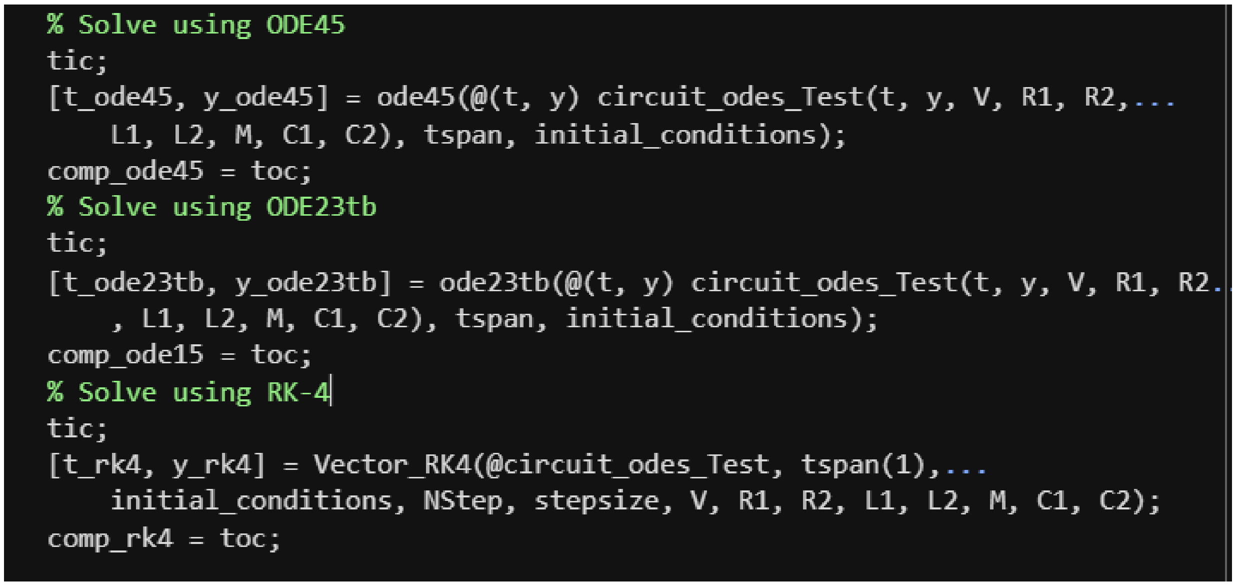


```
function dydt = Circuit_ODE(t, y, V, R1, R2, L1, L2, M, C1, C2)
    % Extract variables
    y1 = y(1); % i1
    y2 = y(2); % i1'
    y3 = y(3); % i2
    y4 = y(4); % i2'

    % Define system of ODEs
    dydt = zeros(4,1);
    dydt(1) = y2;
    dydt(2) = (V(t) - R1*y2 + M*dydt(4) - y1/C1)/L1;
    dydt(3) = y4;
    dydt(4) = (M*dydt(2) - R2*y4 - y3/C2)/L2;
end
```

Figure 4: ODE Function

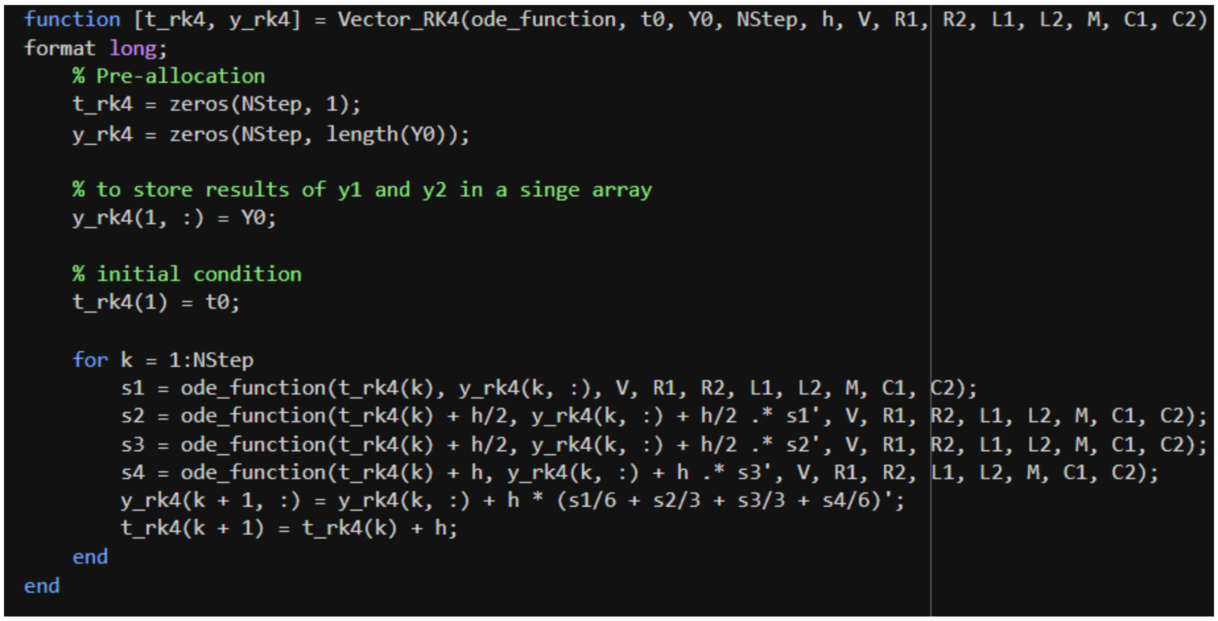
Solve using ode45, ode23tb & RK-4 Method

After creating the ode function the system of equations can be passed through to the following functions for further processing. While debugging the system initially it was assumed that the problem was to result in a stiff problem, however after testing and tunning of the code it was determined that the system was not stiff. Although the system does not appear stiff, further analysis will be conducted further in this report as discrepancies arose in the expected results of each MATLAB solver. The following two figures represent the code implementing each solver and the vectorized fourth order Runge-Kutta method function. Vectorized rk-4 was selected for this process as the system yielded a non stiff system of ordinary differential equations in the format of an initial value problem, which is suited to the capabilities and efficiency of vectorized rk-4.



```
% Solve using ODE45
tic;
[t_ode45, y_ode45] = ode45(@(t, y) circuit_odes_Test(t, y, V, R1, R2,...
    L1, L2, M, C1, C2), tspan, initial_conditions);
comp_ode45 = toc;
% Solve using ODE23tb
tic;
[t_ode23tb, y_ode23tb] = ode23tb(@(t, y) circuit_odes_Test(t, y, V, R1, R2..
    , L1, L2, M, C1, C2), tspan, initial_conditions);
comp_ode15 = toc;
% Solve using RK-4
tic;
[t_rk4, y_rk4] = Vector_RK4(@circuit_odes_Test, tspan(1),...
    initial_conditions, NStep, stepsize, V, R1, R2, L1, L2, M, C1, C2);
comp_rk4 = toc;
```

Figure 5: ODE Solvers

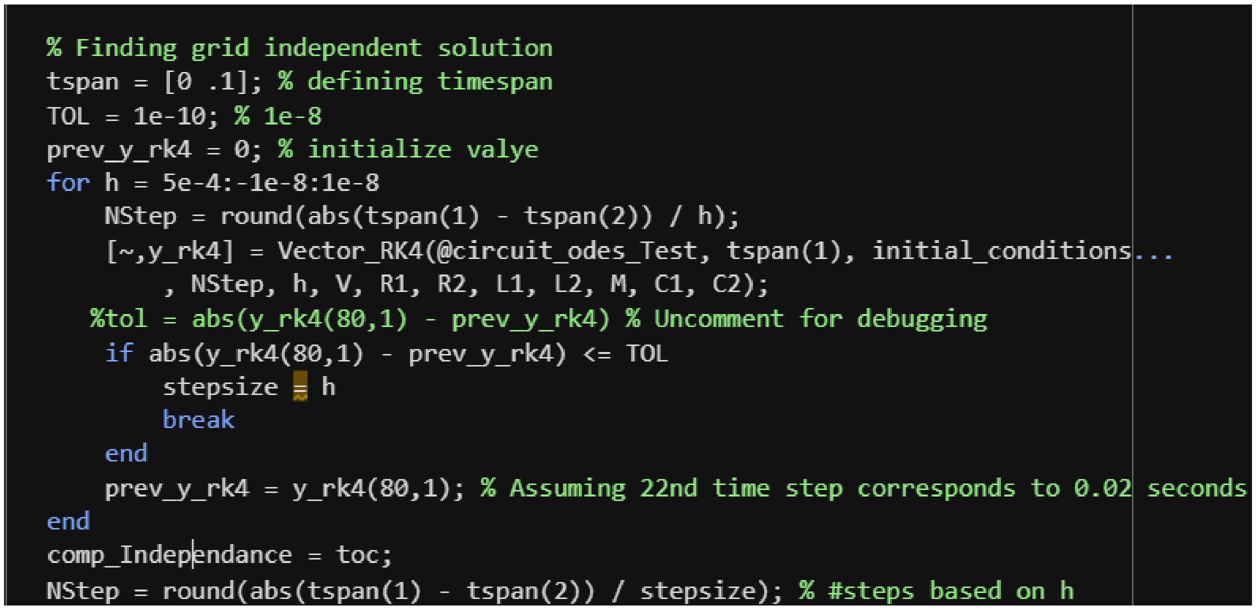


```
function [t_rk4, y_rk4] = Vector_RK4(ode_function, t0, Y0, NStep, h, V, R1, R2, L1, L2, M, C1, C2)
format long;
    % Pre-allocation
    t_rk4 = zeros(NStep, 1);
    y_rk4 = zeros(NStep, length(Y0));

    % to store results of y1 and y2 in a singe array
    y_rk4(1, :) = Y0;

    % initial condition
    t_rk4(1) = t0;

    for k = 1:NStep
        s1 = ode_function(t_rk4(k), y_rk4(k, :), V, R1, R2, L1, L2, M, C1, C2);
        s2 = ode_function(t_rk4(k) + h/2, y_rk4(k, :) + h/2 .* s1', V, R1, R2, L1, L2, M, C1, C2);
        s3 = ode_function(t_rk4(k) + h/2, y_rk4(k, :) + h/2 .* s2', V, R1, R2, L1, L2, M, C1, C2);
        s4 = ode_function(t_rk4(k) + h, y_rk4(k, :) + h .* s3', V, R1, R2, L1, L2, M, C1, C2);
        y_rk4(k + 1, :) = y_rk4(k, :) + h * (s1/6 + s2/3 + s3/3 + s4/6)';
        t_rk4(k + 1) = t_rk4(k) + h;
    end
end
```

Figure 6: Vectorized RK-4 Function

Grid Independence

Grid grid independence refers to the property where the solution of the problem becomes insensitive to changes in the grid or discretization parameters. In the provided code snippet, grid independence is being assessed by iteratively varying the step size (`h`) for solving a system of ordinary differential equations using the rk-4 method. The loop systematically adjusts the step size and computes the solution until Tol is reached. Once the solution becomes invariant to further reductions in the step size, it indicates that the solution is grid independent within the given tolerance. This process ensures that the numerical solution accurately captures the behavior of the underlying physical system. The Tolerance must be found though trial and error as setting it too high results in a loop that never reaches convergence while setting too low results in a system that produces inaccurate results.



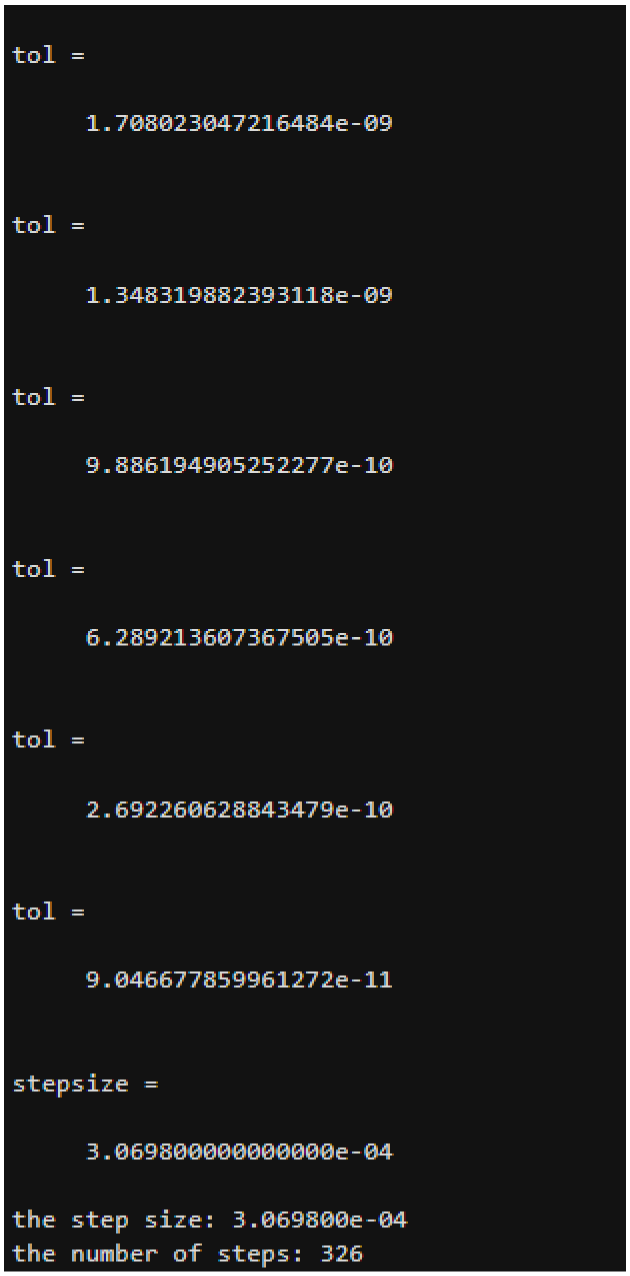
```
% Finding grid independent solution
tspan = [0 .1]; % defining timespan
TOL = 1e-10; % 1e-8
prev_y_rk4 = 0; % initialize valye
for h = 5e-4:-1e-8:1e-8
    NStep = round(abs(tspan(1) - tspan(2)) / h);
    [~,y_rk4] = Vector_RK4(@circuit_odes_Test, tspan(1), initial_conditions...
        , NStep, h, V, R1, R2, L1, L2, M, C1, C2);
   %tol = abs(y_rk4(80,1) - prev_y_rk4) % Uncomment for debugging
    if abs(y_rk4(80,1) - prev_y_rk4) <= TOL
        stepsize = h
        break
    end
    prev_y_rk4 = y_rk4(80,1); % Assuming 22nd time step corresponds to 0.02 seconds
end
comp_Independance = toc;
NStep = round(abs(tspan(1) - tspan(2)) / stepsize); % #steps based on h
```

Figure 7: Grid Independence

The terminal outputs reveal the progression of the debugging process, which was enabled to trace and resolve any errors or inconsistencies in the computational algorithm. This validation process is essential for ensuring the reliability and accuracy of the results.

During this run, a tolerance (Tol) value of 10 was utilized, indicating the acceptable level of error in the solution. The resulting step size, calculated as 3.0698e-4, is the increment by which the simulation progresses at each iteration. This step size is significant as it allows for the execution of 326 adequate steps to be taken, ensuring a thorough exploration of the solution space.

Through meticulously adjusting the step size to achieve the desired number of steps, the computational process can strike a balance between computational efficiency and solution accuracy. This meticulous tuning of parameters is a common practice in numerical simulations to optimize performance while maintaining result quality."



```
tol =

     1.708023047216484e-09


tol =

     1.348319882393118e-09


tol =

     9.886194905252277e-10


tol =

     6.289213607367505e-10


tol =

     2.692260628843479e-10


tol =

     9.046677859961272e-11


stepsize =

     3.069800000000000e-04

the step size: 3.069800e-04
the number of steps: 326
```

Figure 8: Iteration Process

Plotting Results

Through graphical analysis of the solutions, it is evident that the system operates as intended, with all three methods producing nearly identical results. However, upon closer examination of magnified subplots, it becomes apparent that although the system lacks inherent stiffness, the ode45 solver struggles to maintain smooth transitions at the peaks of the waveform. This phenomenon can be attributed to the methodologies employed by each solver in solving the given equation.

Beginning with the rk-4 method, the grid independence step allows us to achieve a step size resulting in highly accurate waveform tracking without compromising computational efficiency. This was accomplished by employing a dummy rk-4 solver and iteratively decreasing the step size until the change in outputs became minimal, falling below the selected tolerance of 1e-10.

Transitioning to the ode45 solver, the rigidity observed at the peaks can be attributed to its use of a single-step explicit Runge-Kutta method. This method, chosen by MATLAB, proves less accurate in documenting the peaks of the waveform compared to the other two methods.

Lastly, considering ode23tb, this method effectively maps the waveform by utilizing an implicit Runge-Kutta method. This approach aids in accurately determining the location of the next point along the waveform. Ode23tb was chosen as the control source for the system over ode15s due to its inherent efficiencies.

The two resulting figures illustrate the outputs of the solvers across varying time domains. In both figures, the current in the circuits is represented, with current in circuit 1 denoted by i1 and current in circuit 2 denoted by i2.

In the first figure, it is apparent that due to the presence of capacitors in the system, there is a slight fluctuation during the startup phase of the device. However, as the system progresses towards steady state, the resulting waveform begins to oscillate with consistent peaks and troughs. Despite the absence of a power source, there exists an induced current in the second closed system, owing to the magnetic forces generated by the inductor. This induced current is always less than that of the primary circuit, in accordance with the principle of conservation of energy.

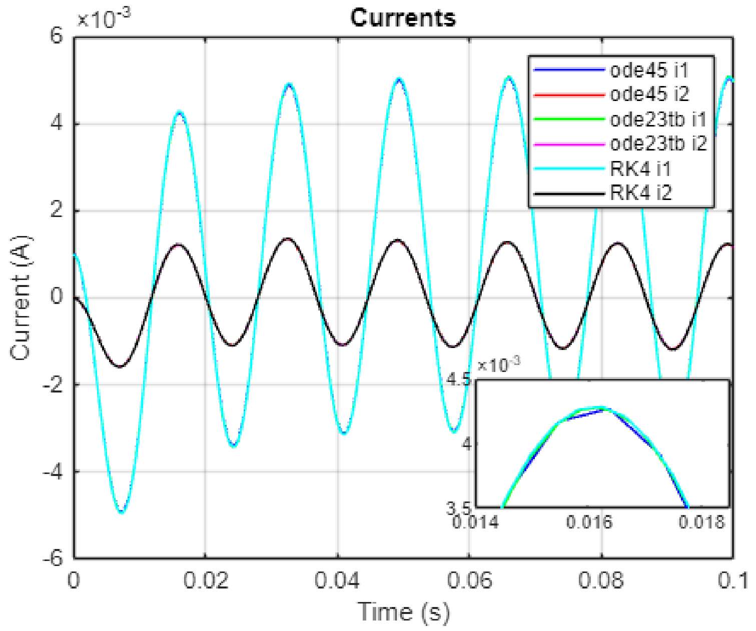


Figure 9: Current in Each Circuit Tol 1e-11

Figure 10 illustrates an extended time interval to adequately demonstrate the system's progression towards steady state.

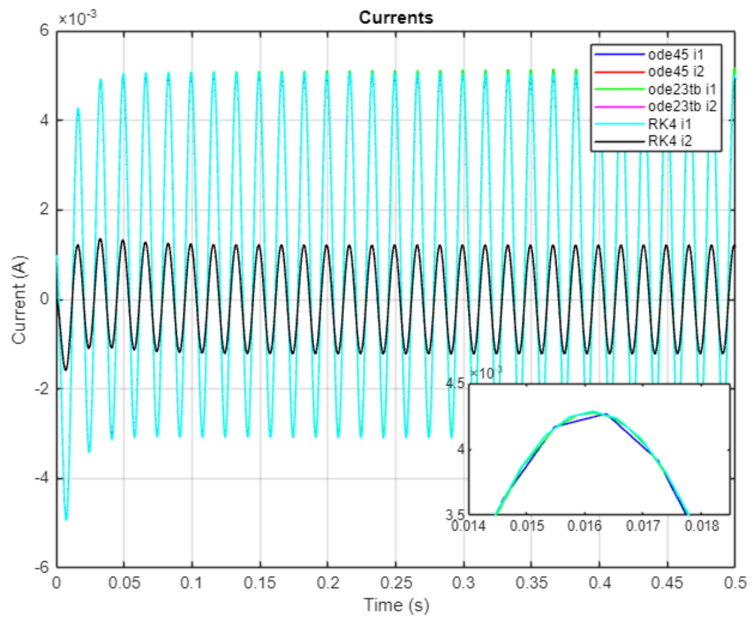


Figure 10: Current in Each Circuit Time Span 0.5

The following figure depicts the rate of current change in both circuits as a result of each of the methods used. Circuit 1 & 2 is denoted by di1/dt & di2/dt respectively.

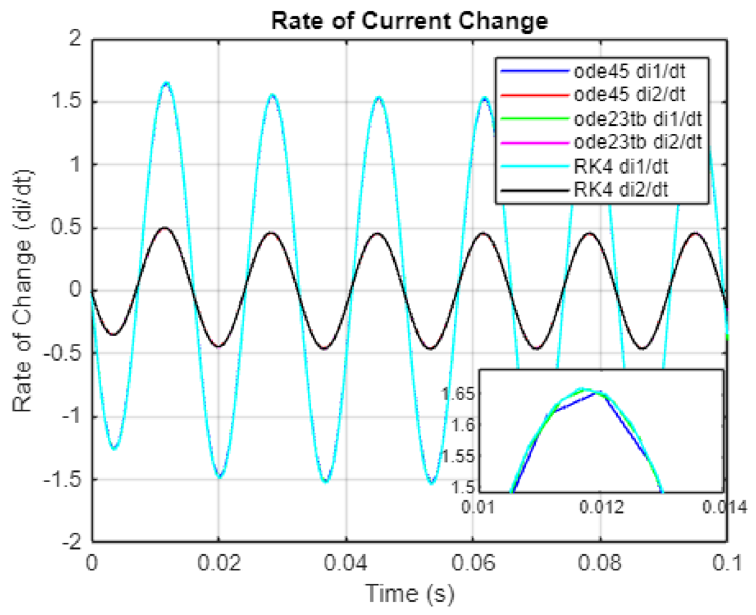


Figure 11: Rate of Current Change Tol 1e-11

Finally, the last output illustrates the voltage induced on the second circuit. This interaction is fundamental to the operation of electrical transformers. Implementing a diode in this system would convert the AC current source into the widely used DC current output. Specifically, the use of a negative restrictive diode would permit only positive current flow, thereby inducing a positive voltage in the system and effectively eliminating the bottom half of the waveform.

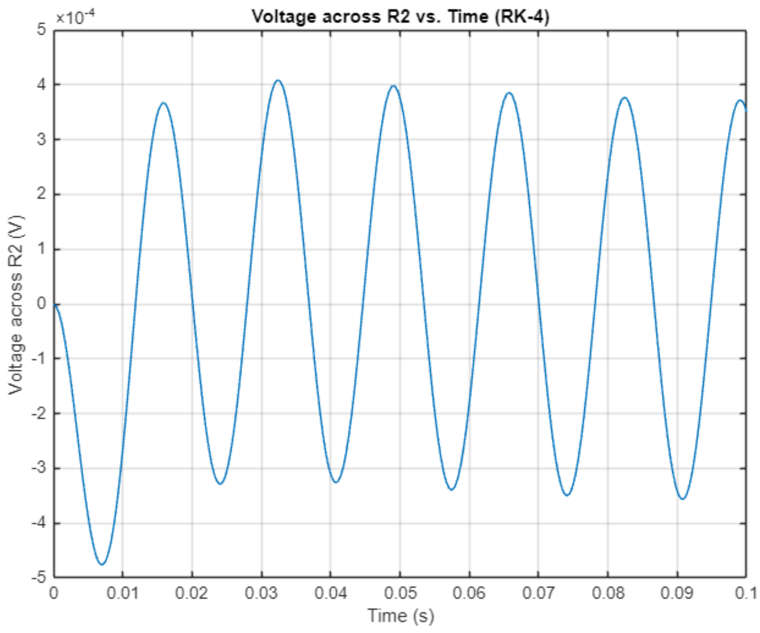


Figure 12: Induced Voltage in R2

Analysis

The following figure illustrates the plots of the resulting absolute error for each process. Given the nature of the system, the most effective approach to track error in a system of ODEs is to measure absolute error relative to previous solutions. As mentioned earlier, ode23tb was selected as the control solver due to visual analysis of the resulting graphs and its production of the smoothest curve along with utilization of a highly effective implicit Runge-Kutta process.

The graphical representation of the error in the system further reinforces the visual inspection of error conducted previously with the magnified subplots. It can be observed from the red line that the rk-4 method produces the least error compared to ode23tb, which aligns with the visual inspection conducted earlier. Furthermore, when comparing ode45 with the other two solvers, it becomes evident that the error significantly increases, as depicted by the blue and green lines. This demonstrates that rk-4 and ode23tb produce very similar results.

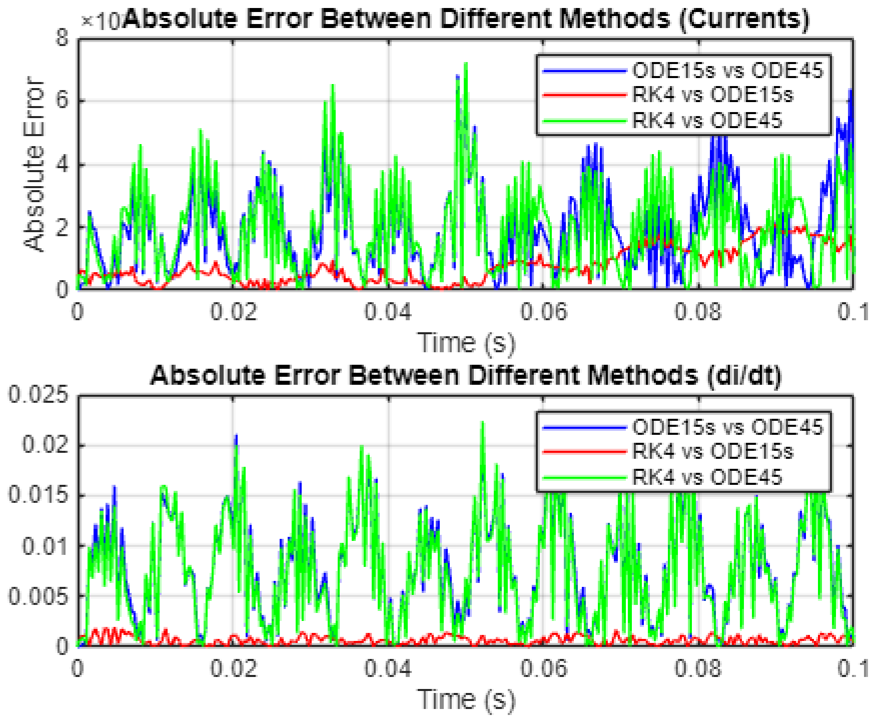


Figure 13: Absolute Error Plot Tol 1e-11

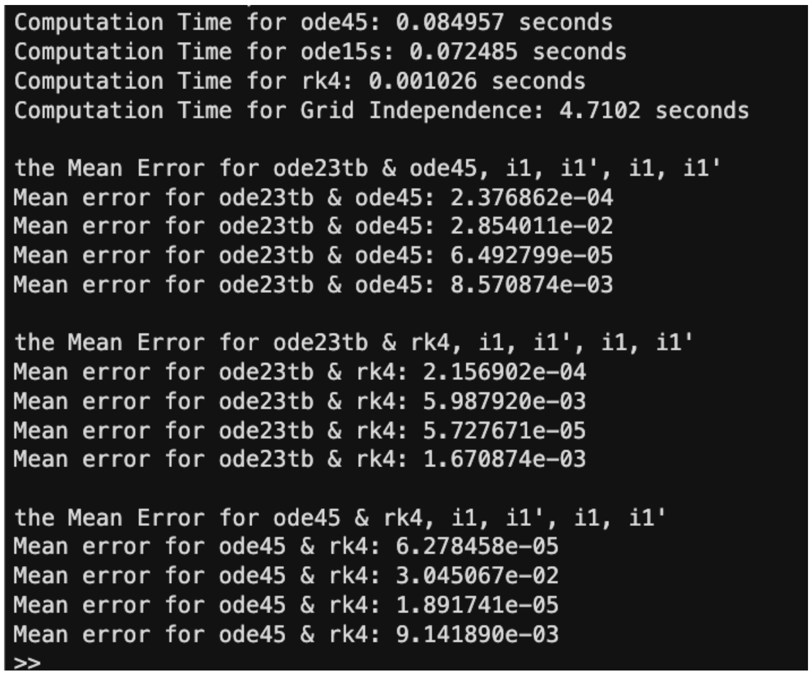
Further analysis was conducted, and calculations of the mean absolute error were performed. This reinforces the observation that although the system does not inherently exhibit signs of stiffness, the ode45 solver falls short of the other two solvers in terms of absolute error.

The limitations of ode45 persist when analyzing the computational time required by the solvers. The figure below depicts the mean error for the solvers alongside their respective computational times. Notably, a tolerance of 1e-11 was initially set, which will later be iteratively adjusted to assess its impact on the system.

With a tolerance of 1e-11, it is evident that while the rk-4 method only requires 1.026e-3 seconds, this value does not account for the time needed to determine grid independence, which is a substantial 4.7 seconds. This highlights that although a tolerance of 1e-11 is effective for accuracy, the associated time cost renders it impractical.

Moving on to the MATLAB solvers, we observe that ode23tb outperforms ode45 by providing a solution 0.01 seconds faster while maintaining superior accuracy. After a comprehensive analysis of all processes utilized, it can be concluded that for this system, ode23tb is the recommended solution as it offers the most accurate results within the shortest timeframe.

This prompts further consideration regarding the applications of ode23tb. Despite the waveform not demonstrating inherent stiffness, ode23tb proves to be more efficient at solving the problem compared to both rk-4 and ode45. With this in mind it is recommended that for this given problem ode23tb be utilized for computation.



```
Computation Time for ode45: 0.084957 seconds
Computation Time for ode15s: 0.072485 seconds
Computation Time for rk4: 0.001026 seconds
Computation Time for Grid Independence: 4.7102 seconds

the Mean Error for ode23tb & ode45, i1, i1', i1, i1'
Mean error for ode23tb & ode45: 2.376862e-04
Mean error for ode23tb & ode45: 2.854011e-02
Mean error for ode23tb & ode45: 6.492799e-05
Mean error for ode23tb & ode45: 8.570874e-03

the Mean Error for ode23tb & rk4, i1, i1', i1, i1'
Mean error for ode23tb & rk4: 2.156902e-04
Mean error for ode23tb & rk4: 5.987920e-03
Mean error for ode23tb & rk4: 5.727671e-05
Mean error for ode23tb & rk4: 1.670874e-03

the Mean Error for ode45 & rk4, i1, i1', i1, i1'
Mean error for ode45 & rk4: 6.278458e-05
Mean error for ode45 & rk4: 3.045067e-02
Mean error for ode45 & rk4: 1.891741e-05
Mean error for ode45 & rk4: 9.141890e-03
>>
```

Figure 14: Computation Time & Mean Error for Tol 1e-11

Effects of Tolerance

Continuing from the previous segment, we will now examine the effects of tolerance on the rk-4 method. As mentioned earlier, while a tolerance of 1e-11 provides an accurate solution, it is deemed too slow for practical use. Increases beyond 1e-11 only serve to further escalate computational time and will therefore not be considered.

Next, we will investigate the impact of tolerances set at 1e-8 and 1e-6 to assess their effects on the system. We'll focus our attention on the red line, representing the rk-4 method, in comparison with ode23tb.

In the following figure, the system is run with a tolerance of 1e-8. It becomes evident that the spikes of inaccuracy are more pronounced compared to the previous run.

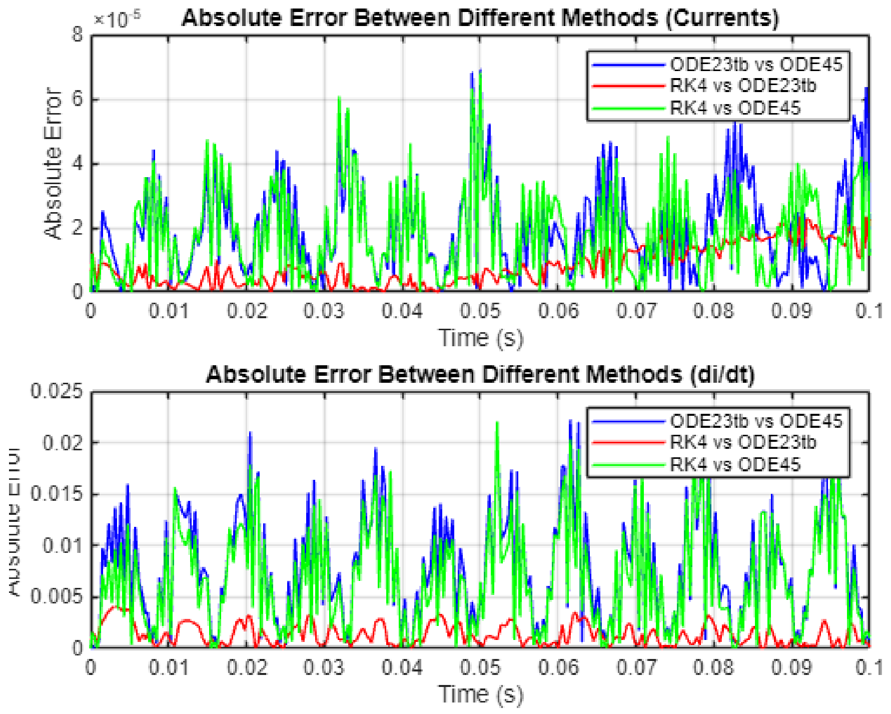


Figure 15: *Absolute Error Plot Tol 1e-8*

Additionally, when reanalysing the rate of change graph, we can see more pronounced edges relating to the cyan line, representing the rk-4 method.

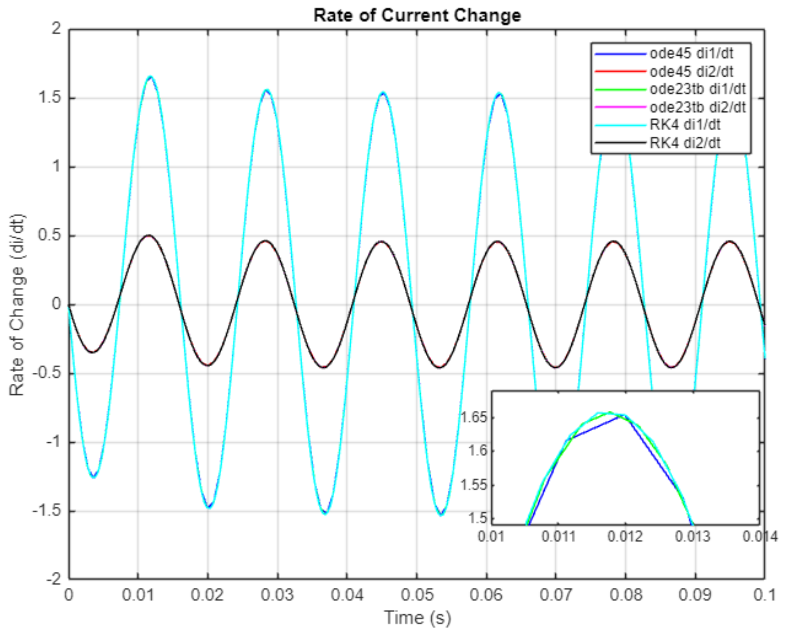
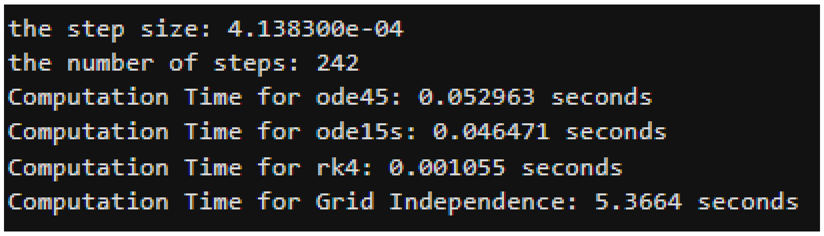


Figure 16: Rate of Current Change Tol 1e-8

Finally, we can observe the computation time for grid independence decreased drastically however, still holds a time longer than 1 second, which can be assumed as unacceptable.



```
the step size: 4.138300e-04
the number of steps: 242
Computation Time for ode45: 0.052963 seconds
Computation Time for ode15s: 0.046471 seconds
Computation Time for rk4: 0.001055 seconds
Computation Time for Grid Independence: 5.3664 seconds
```

Figure 17: Computation Time for Tol 1e-8

Next, we will conduct the analyse again using tolerance of 1e-6, again focusing on the red line. It can be observed that the magnitudes of error are continuing to increase as tolerance decreases.

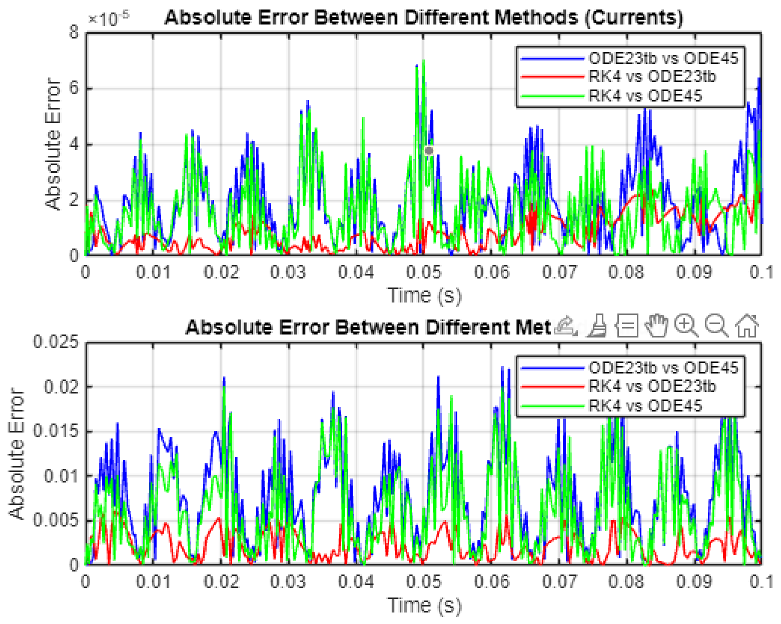


Figure 18: *Absolute Error Plot Tol 1e-6*

Again, focusing on the magnified subplot we can see even more pronounced jagged edges protruding from around the peak further indicating that error is increasing with a decreasing tolerance.

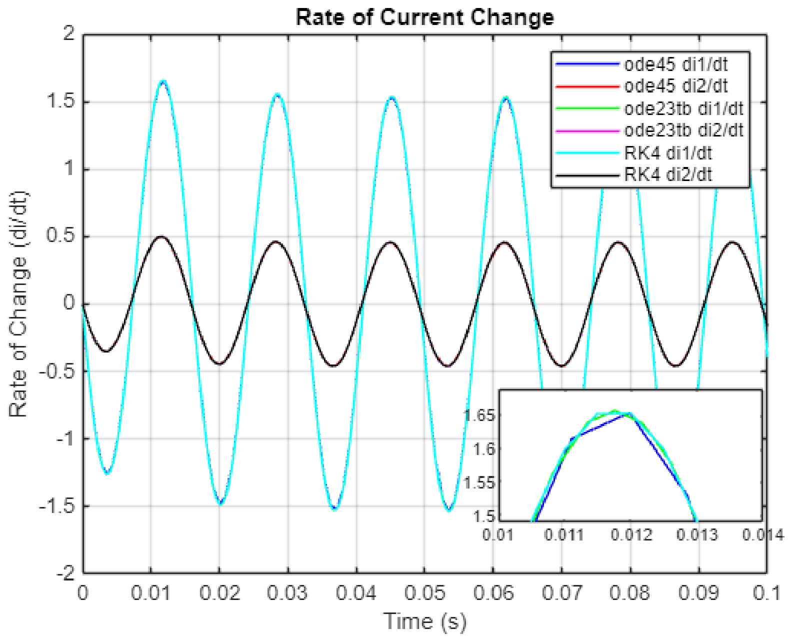


Figure 19: Rate of Current Change Tol 1e-6

However, through examination of the output terminal, a tolerance of 1e-6 finally reaches a computation time under 1 second which is deemed as acceptance for this process.

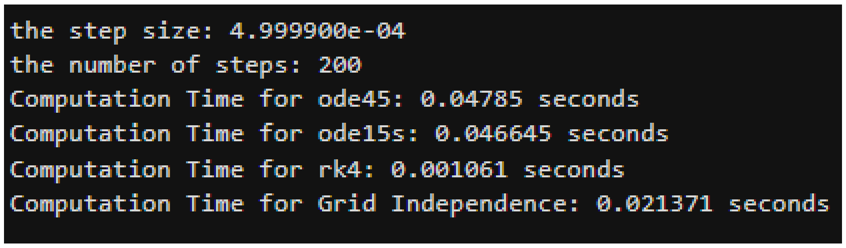

```
the step size: 4.999900e-04
the number of steps: 200
Computation Time for ode45: 0.04785 seconds
Computation Time for ode15s: 0.046645 seconds
Computation Time for rk4: 0.001061 seconds
Computation Time for Grid Independence: 0.021371 seconds
```

Figure 20: Computation Time for Tol 1e-6

To conclude our discussion on tolerance, it must be mentioned that the grid divergence method used pertains to a significant portion of the computational time for rk-4 method, and more efficient means for grid independence would drastically reduce this time and allow for more accurate results. Additionally, it should be noted that if the step size found by tolerance 1e-11 or higher is manually placed in the code, the time significantly reduces as the grid independence loop is no longer required. Overall the ode23tb solver still maintains its place as the preferred solver for this situation.

Discuss the physics of the problem by changing parameters.

To discuss the physics of the problem parameters will be changed to showcase the workings of the system. The parameters to be changed will be: Mutual Inductance, Power Source Frequency and Resistance.

Mutual Inductance

Mutual inductance is a phenomenon in electromagnetism where a changing current in one coil induces a voltage in another coil that is placed nearby. This effect occurs due to the magnetic field produced by the changing current in the first coil, which then links with the turns of wire in the second coil, inducing an electromotive force (EMF) according to Faraday's law of electromagnetic induction.

The magnitude of the induced voltage in the second coil depends on factors such as the rate of change of current in the first coil, the number of turns in each coil, and their relative positions and orientations. Mutual inductance is often denoted by the symbol M and is measured in henries (H). It plays a crucial role in the operation of transformers, motors, generators, and various other electrical devices. This property can be described as $M = \frac{N\alpha}{I}$ where N denotes number of turns in coil, alpha denotes the magnetic flux and I denotes current. The Value of M is the same for both circuits due to the nature of the equation. This property is the focus of this system and as we will see poses significant effects on the system when drastically changed. For this problem initially mutual inductance was selected as 0.09 H.

M = 0

When the value of mutual inductance is 0 H, it indicates that the systems are not sufficiently close to one another, or the magnetic field of the primary system is too weak to influence the second system. Consequently, no activity occurs on the second system, and no induced current is observed. This absence of induction is visually represented in the following two figures, where the value of M was set to zero. It is evident that there is no induced current or voltage detected in the second system.

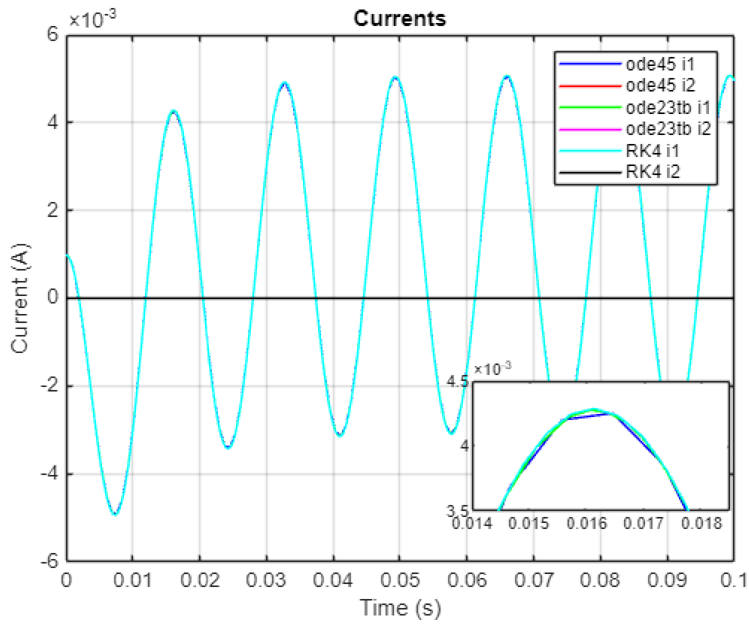


Figure 21: Current in Each Circuit M=0

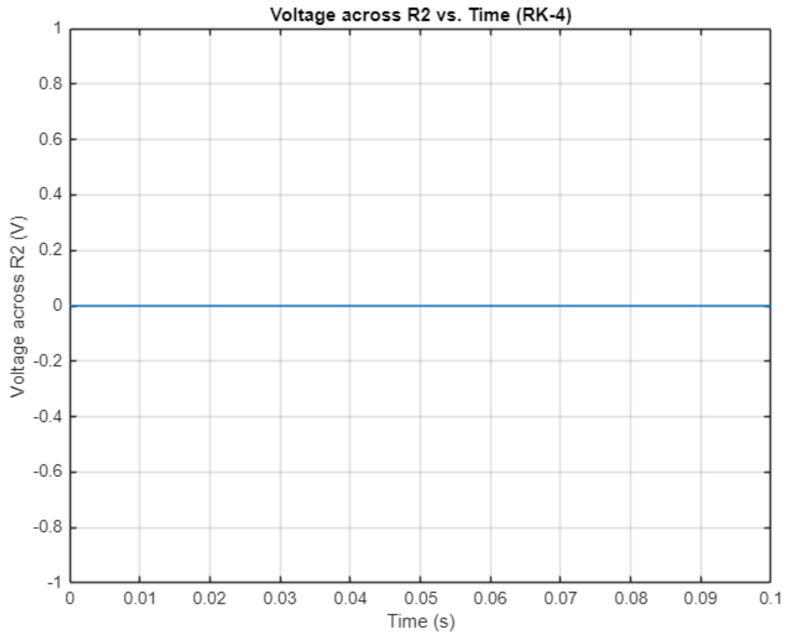


Figure 22: Induced Voltage in R2 M=0

M = 0.2

Additionally, if the systems are brought together or the magnetic field of the first system is increased we will see an increase in induced current and voltage on the second system. This physical property can be observed in the next two figures, where M was set to 0.2 H.

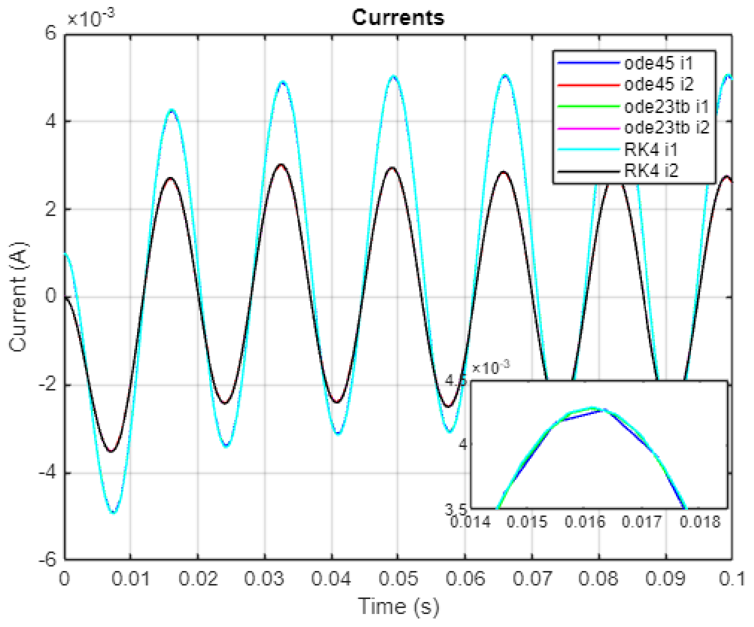


Figure 23: Current in Each Circuit M=0.2

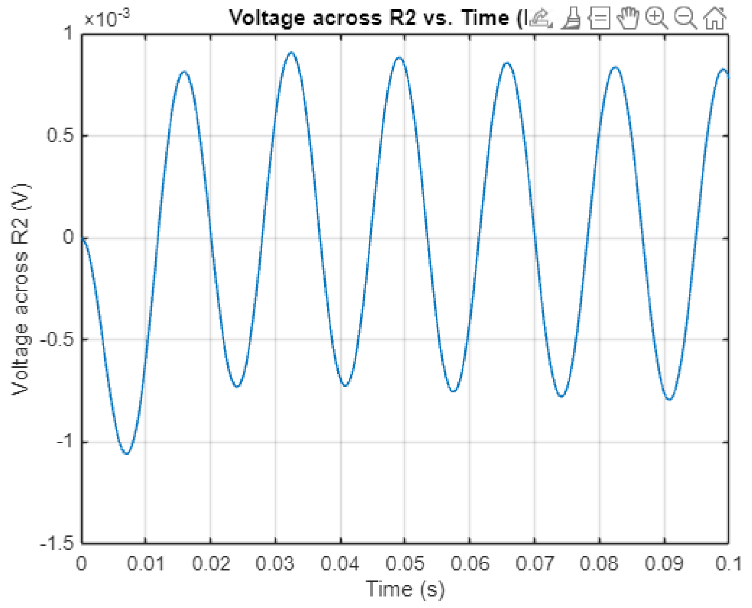


Figure 24: Induced Voltage in R2 M=0.2

Frequency

The frequency of a power source influences the waveform characteristics. Increasing frequency shortens the waveform period, causing cycles to occur more rapidly, while the amplitude may vary, especially near resonance frequencies of systems with capacitors and inductors. Phase shifts can occur, altering the waveform's shape and timing, particularly in reactive components. Higher frequencies can introduce harmonics, multiples of the fundamental frequency, which may distort the waveform and affect power quality. Understanding these effects is vital for designing and analyzing electrical systems operating at varying frequencies. Frequency can be denoted as ω where $V(t)_S = A * \cos(\omega t)$

ω = 80

Increasing the frequency of the power source leads into two resulting effects on the systems. The first being an increase in waveform cycles, and the second being a slight distortion of the waveform, resulting in an increase amplitude and decreased trough, this is most likely due to interactions with the capacitors in the system. The increase in oscillation frequency can be visible in both the current and resulting voltage in the second resistor.

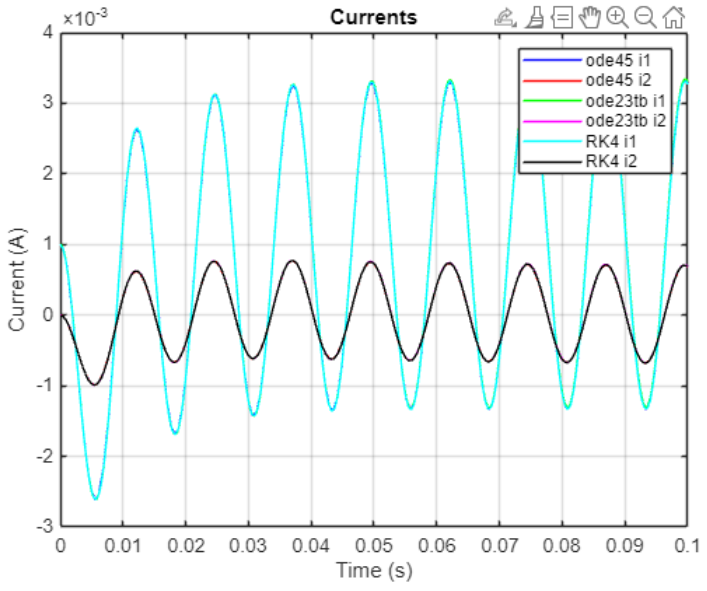


Figure 25: Current in Each Circuit ω=80

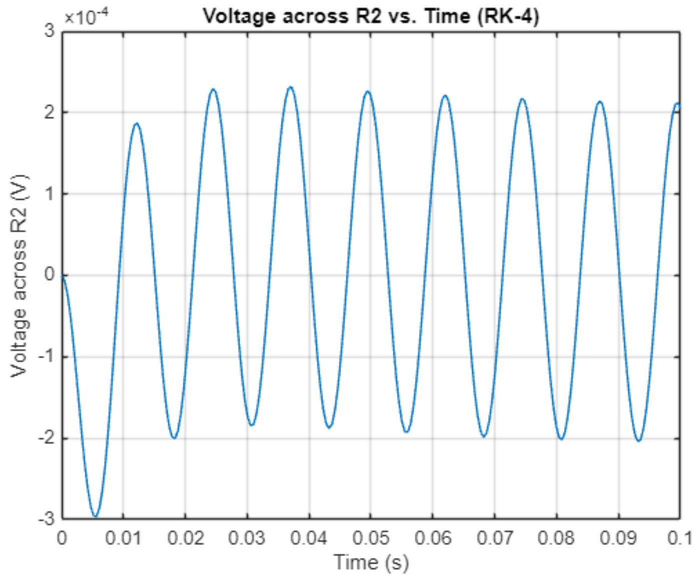


Figure 26: Induced Voltage in R2 ω=40

ω = 40

Decreasing the frequency of the power source has the inverse effect on the system, the frequency of cycles has been decreased, elongating the waveforms, and resulting in a more centered waveform with the amplitude of peak and trough approaching the same absolute value with decreasing frequency. This is most likely due to interactions with the capacitors of the system.

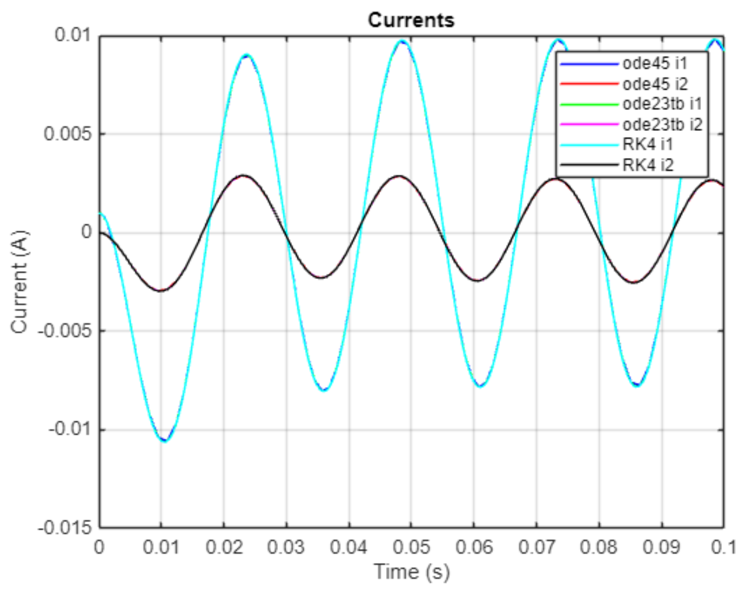


Figure 27: Current in Each Circuit ω=40

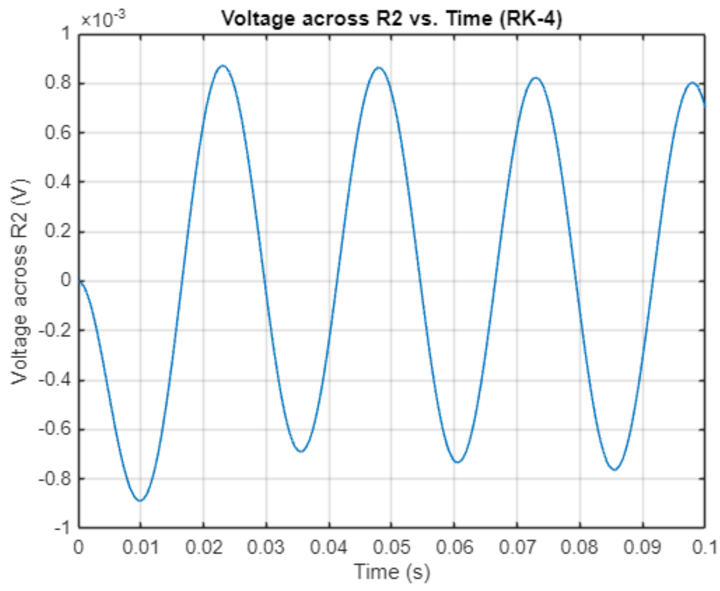


Figure 28: Induced Voltage in R2 ω=40

Resistance

In RLC circuits resistors act as a dampener which aid the circuit in reaching steady state operation. This can be observed through the following three tests. For visual reference the initial resistance of 10 Ω will appear first. Followed by a resistance of 20 Ω and finally a resistance of 10 Ω.

R1 = 10

The initial system the first peak occurs lower than the following peaks. This phenomenon can be countered by fine tuning the resistance to reach a point where the system reaches steady state quickly, increasing the value to large and the system will experience decreasing oscillations until steady state is reached, and lowering the value past optimal set will result in increasing oscillations until steady state is reach.

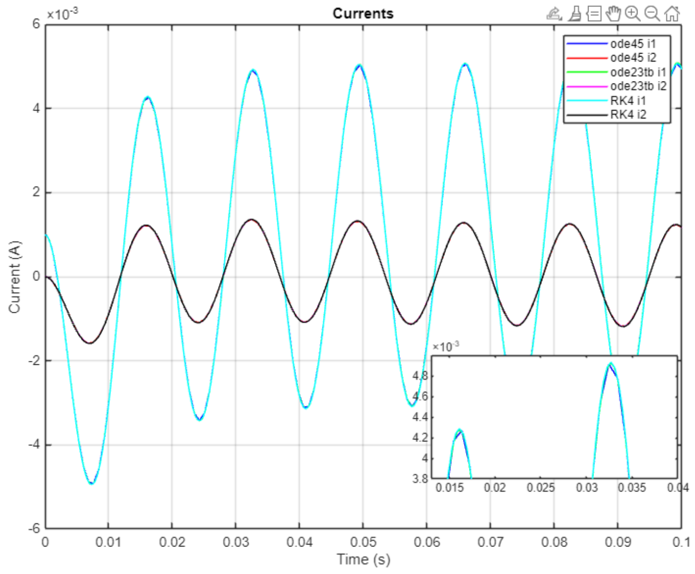


Figure 29: Current in Each Circuit R1=10

R1 = 20

The following figure depicts the use of a 20 Ω resistor and demonstrates the dampening effect of the resistors on system start up. The system requires a further increase in resistance to reach critical dampening, which is described as the maximum value such that the transient response of the circuit reached steady state conditions as quickly as possible without overshoot oscillations.

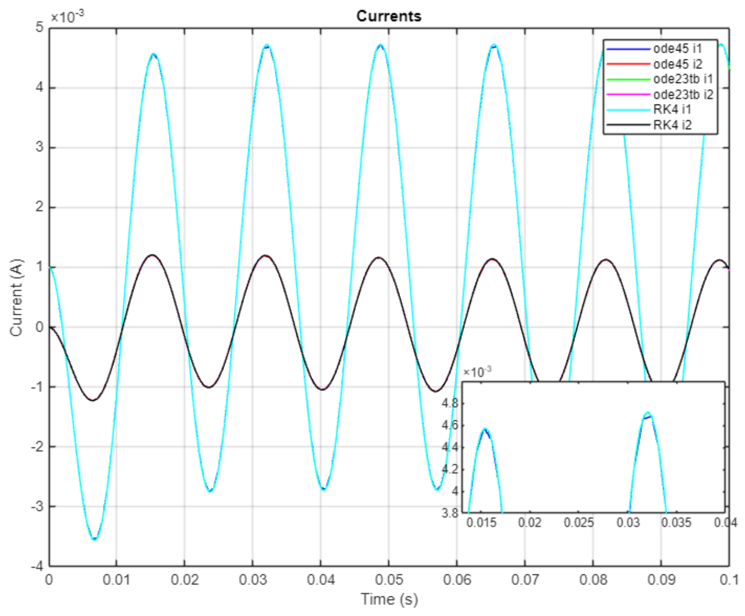


Figure 30: Current in Each Circuit R1=20

R1 = 5

Finally, a resistance of 5 Ω was used and it can be clearly seen that the system takes longer to reach steady state due to the reduced resistance in the system.

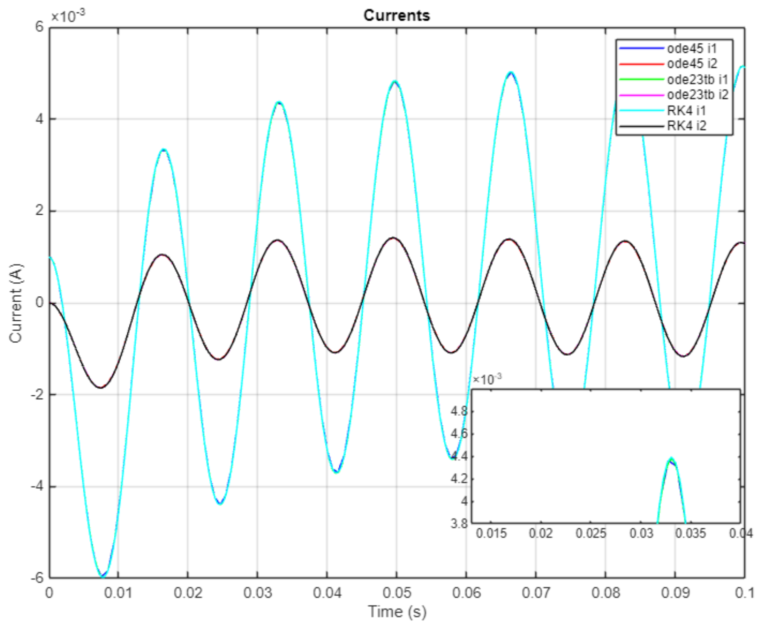


Figure 30: Current in Each Circuit R1=5

To Conclude the parameters of the system all, pertain to crucial aspects of the circuit and changing these values can have profound affects on the resulting waveforms, this can be seen through varying mutual inductance M, frequency ω and R1. The system contains additional parameters which would influence the system in different ways to the previously discussed topics. It should be noted that although Parameters tied to one circuit change, this change ultimately impacts the alternative circuit as well due to the coupled nature of the differential equations. Additionally, this system is by no means perfect, and one example of improvement lies in critical dampening of the system. Which can be found through trial and error in the simulation or through mathematical analysis of the initial system using Laplace transform, however this could pose complex in nature.

Conclusion

In conclusion, our analysis of the system's behavior under various solver methods, tolerances, and parameters has yielded valuable insights. We have observed the efficacy of different numerical solvers, such as rk-4, ode45, and ode23tb, in accurately tracking the system's dynamics. While ode23tb emerged as the preferred solver, offering both accuracy and efficiency, the shortcomings of ode45 became apparent, particularly in its inability to maintain accuracy around the peaks and troughs of the waveform.

Furthermore, our exploration of the impact of mutual inductance on the system demonstrated the critical role it plays in inducing current and voltage in the secondary system. When the mutual inductance was set to zero, no induction occurred, underscoring the dependence of the secondary system's activity on the strength of the magnetic field from the primary system.

Additionally, our investigation into the effects of tolerance on the rk-4 method highlighted the delicate balance between accuracy and computational efficiency. While a tolerance of 1e-11 provided precise results, it proved impractical due to excessive computational time. Conversely, higher tolerances resulted in faster computations but sacrificed accuracy. Implementation of more efficient grid independence loops can yield significantly faster results.

Overall, this study underscores the importance of selecting appropriate numerical methods, tolerances, and system parameters to achieve accurate and efficient simulations in the field of computation method in mechanical engineering. These findings not only enhance our understanding of numerical methods in solving differential equations but also have practical implications for the design and optimization of electrical systems. Finally, it should be stated that the reference in the followed section contain purely sources of inspiration and clarification, all references were utilized to fact check and reconfirm information found in this report.

References

"14.2: Mutual Inductance." *Physics LibreTexts,* Libretexts, 12 Sept. 2022, phys.libretexts.org/Bookshelves/University_Physics/University_Physics_(OpenStax)/Book%3A_University_Physics_II_-_Thermodynamics_Electricity_and_Magnetism_(OpenStax)/14%3A_Inductance/14.02%3A_Mutual_Inductance.

"Analyzing Mutual Inductance in Different Coil Arrangements." *COMSOL*, www.comsol.com/blogs/analyzing-mutual-inductance-in-different-coil-arrangements#:~:text=When%20a%20time%2Dvarying%20current,effect%20is%20called%20mutual%20induction. Accessed 19 Apr. 2024.

"Basic Inductance Principles in Transformers - Technical Articles." *EEPower*, eepower.com/technical-articles/transformer-operating-principles/#. Accessed 19 Apr. 2024.

"L-R-C Circuit." *R*, www.sciencedirect.com/topics/mathematics/l-r-c-circuit#:~:text=the%20following%20table.-,The%20physical%20principle%20needed%20to%20derive%20the%20differential%20equation%20that,t%20)%20impressed%20on%20the%20circuit.&text=L%20d%20I%20d%20t%20%2B%20R%20I%20%2B%201%20C,Q%20%3D%20E%20(%20t%20)%20. Accessed 19 Apr. 2024.

Mutual Inductance, farside.ph.utexas.edu/teaching/em/lectures/node83.html. Accessed 19 Apr. 2024.

Storr, Wayne. "Mutual Inductance of Two Adjacent Inductive Coils." *Basic Electronics Tutorials*, 6 Aug. 2022, www.electronics-tutorials.ws/inductor/mutual-inductance.html.

Code

Main Body

```
clear all; close all; clc;
% Define circuit parameters
R1 = 10; % 10 ohms
R2 = 10; % 10 ohms
L1 = 0.1; % 0.1 H
L2 = 0.3; % 0.1 H (second inductor)
M = 0.09; % 0-0.3 Mutual inductance (H)
C1 = 1; % 0.1 F
C2 = 1; % 0.1 F

% Define time varying power source
V = @(t) -60 * cos(2*pi*60*t);

% Define initial conditions
i10 = .00100; % 0 A
i10_prime = 0; % 0 V
i20 = 0; % 0 Initial current for the second inductor
i20_prime = 0; % 0 V
initial_conditions = [i10; i10_prime; i20; i20_prime];

% Finding grid independent solution
tspan = [0 .1]; % defining timespan
TOL = 1e-8; % 1e-10
prev_y_rk4 = 0; % initialize valye
tic;
for h = 5e-4:-1e-8:1e-8
    NStep = round(abs(tspan(1) - tspan(2)) / h);
    [~,y_rk4] = Vector_RK4(@Circuit_ODE, tspan(1), initial_conditions...
        , NStep, h, V, R1, R2, L1, L2, M, C1, C2);
   %tol = abs(y_rk4(80,1) - prev_y_rk4) % Uncomment for debugging
    if abs(y_rk4(80,1) - prev_y_rk4) <= TOL
        stepsize = h;
        break
    end
    prev_y_rk4 = y_rk4(80,1); % Assuming 22nd time step corresponds to 0.02 seconds
end
comp_Independance = toc;
NStep = round(abs(tspan(1) - tspan(2)) / stepsize); % #steps based on h
% Solve using ODE45
tic;
[t_ode45, y_ode45] = ode45(@(t, y) Circuit_ODE(t, y, V, R1, R2,...
    L1, L2, M, C1, C2), tspan, initial_conditions);
comp_ode45 = toc;
% Solve using ODE23tb
tic;
[t_ode23tb, y_ode23tb] = ode23tb(@(t, y) Circuit_ODE(t, y, V, R1, R2...
    , L1, L2, M, C1, C2), tspan, initial_conditions);
comp_ode15 = toc;
% Solve using RK-4
tic;
[t_rk4, y_rk4] = Vector_RK4(@Circuit_ODE, tspan(1),...
    initial_conditions, NStep, stepsize, V, R1, R2, L1, L2, M, C1, C2);
```

```matlab
comp_rk4 = toc;

% Interpolate ODE45 solution onto ODE15s time grid
y_ode45_interp = interp1(t_ode45, y_ode45, t_ode23tb);
y_rk4_interp = interp1(t_rk4, y_rk4, t_ode23tb);

% Calculate absolute error for ODE15s
abs_error_ode23tb_ode45 = abs(y_ode23tb - y_ode45_interp);

% Calculate absolute error for RK4
abs_error_rk4_ode23tb = abs(y_ode23tb - y_rk4_interp);

% Calculate absolute error between ODE45 and RK4
abs_error_ode45_rk4 = abs(y_ode45_interp - y_rk4_interp);

fprintf("the step size: %e\nthe number of steps: %d\n", stepsize, NStep);
disp(['Computation Time for ode45: ', num2str(comp_ode45), ' seconds']);
disp(['Computation Time for ode15s: ', num2str(comp_ode15), ' seconds']);
disp(['Computation Time for rk4: ', num2str(comp_rk4), ' seconds']);
disp(['Computation Time for Grid Independence: ', num2str(comp_Independance), ' seconds']);
fprintf("\nthe Mean Error for ode23tb & ode45, i1, i1', i1, i1'\n");
fprintf('Mean error for ode23tb & ode45: %e\n', max(abs_error_ode23tb_ode45));
fprintf("\nthe Mean Error for ode23tb & rk4, i1, i1', i1, i1'\n");
fprintf('Mean error for ode23tb & rk4: %e\n', max(abs_error_rk4_ode23tb));
fprintf("\nthe Mean Error for ode45 & rk4, i1, i1', i1, i1'\n");
fprintf('Mean error for ode45 & rk4: %e\n', max(abs_error_ode45_rk4));


% Plot currents
figure;
plot(t_ode45, y_ode45(:,1), 'b', t_ode45, y_ode45(:,3), 'r', ...
    t_ode23tb, y_ode23tb(:,1), 'g', t_ode23tb, y_ode23tb(:,3), 'm', ...
    t_rk4, y_rk4(:,1), 'c', t_rk4, y_rk4(:,3), 'k');
title('Currents');
xlabel('Time (s)');
ylabel('Current (A)');
xlim([tspan(1) tspan(2)]);
legend('ode45 i1', 'ode45 i2', 'ode23tb i1', 'ode23tb i2', 'RK4 i1', 'RK4 i2');
grid on;
% Insert magnified view
left = 0.6; bottom = 0.19; width = 0.3; height = 0.2;
axes('Position',[left, bottom, width, height]);
plot(t_ode45, y_ode45(:,1), 'b', t_ode45, y_ode45(:,3), 'r', ...
    t_ode23tb, y_ode23tb(:,1), 'g', t_ode23tb, y_ode23tb(:,3), 'm', ...
    t_rk4, y_rk4(:,1), 'c', t_rk4, y_rk4(:,3), 'k');
xlim([0.014 0.0185]);
ylim([0.0035 0.0045]);

% Plot rate of current change
figure;
plot(t_ode45, y_ode45(:,2), 'b', t_ode45, y_ode45(:,4), 'r', ...
    t_ode23tb, y_ode23tb(:,2), 'g', t_ode23tb, y_ode23tb(:,4), 'm', ...
    t_rk4, y_rk4(:,2), 'c', t_rk4, y_rk4(:,4), 'k');
title('Rate of Current Change');
xlabel('Time (s)');
ylabel('Rate of Change (di/dt)');
```

```
xlim([tspan(1) tspan(2)]);
legend('ode45 di1/dt', 'ode45 di2/dt', 'ode23tb di1/dt', 'ode23tb di2/dt', 'RK4 di1/dt', 'RK4 di2/dt');
grid on;
% Insert magnified view
left = 0.6; bottom = 0.19; width = 0.3; height = 0.2;
axes('Position',[left, bottom, width, height]);
plot(t_ode45, y_ode45(:,2), 'b', t_ode45, y_ode45(:,4), 'r', ...
    t_ode23tb, y_ode23tb(:,2), 'g', t_ode23tb, y_ode23tb(:,4), 'm', ...
    t_rk4, y_rk4(:,2), 'c', t_rk4, y_rk4(:,4), 'k');
xlim([0.01 0.014]);
ylim([1.49 1.69]);

% Plot absolute error
figure;
subplot(2,1,1);
plot(t_ode23tb, abs_error_ode23tb_ode45(:,1), 'b', t_ode23tb, abs_error_rk4_ode23tb(:,1), 'r', t_ode23tb,
abs_error_ode45_rk4(:,1), 'g');
title('Absolute Error Between Different Methods (Currents)');
xlabel('Time (s)');
ylabel('Absolute Error');
legend('ODE23tb vs ODE45', 'RK4 vs ODE23tb', 'RK4 vs ODE45');
grid on;

subplot(2,1,2);
plot(t_ode23tb, abs_error_ode23tb_ode45(:,2), 'b', t_ode23tb, abs_error_rk4_ode23tb(:,2), 'r', t_ode23tb,
abs_error_ode45_rk4(:,2), 'g');
title('Absolute Error Between Different Methods (di/dt)');
xlabel('Time (s)');
ylabel('Absolute Error');
legend('ODE23tb vs ODE45', 'RK4 vs ODE23tb', 'RK4 vs ODE45');
grid on;

% Calculate voltage across R2 from the RK-4 solution
V_R2_rk4 = L2 * y_rk4(:,3);

% Plot the voltage across R2
figure;
plot(t_rk4, V_R2_rk4);
xlim([tspan(1) tspan(2)]);
xlabel('Time (s)');
ylabel('Voltage across R2 (V)');
title('Voltage across R2 vs. Time (RK-4)');
grid on;
```

Function 1: Circuit ODE

```
function dydt = Circuit_ODE(t, y, V, R1, R2, L1, L2, M, C1, C2)
  % Extract variables
  y1 = y(1); % i1
  y2 = y(2); % i1'
  y3 = y(3); % i2
  y4 = y(4); % i2'

  % Define system of ODEs
  dydt = zeros(4,1);
```

```
    dydt(1) = y2;
    dydt(2) = (V(t) - R1*y2 + M*dydt(4) - y1/C1)/L1;
    dydt(3) = y4;
    dydt(4) = (M*dydt(2) - R2*y4 - y3/C2)/L2;
end
```

Function 2: Vectorized RK-4

```
function [t_rk4, y_rk4] = Vector_RK4(ode_function, t0, Y0, NStep, h, V, R1, R2, L1, L2, M, C1, C2)
format long;
    % Pre-allocation
    t_rk4 = zeros(NStep, 1);
    y_rk4 = zeros(NStep, length(Y0));

    % to store results of y1 and y2 in a singe array
    y_rk4(1, :) = Y0;

    % initial condition
    t_rk4(1) = t0;

    for k = 1:NStep
        s1 = ode_function(t_rk4(k), y_rk4(k, :), V, R1, R2, L1, L2, M, C1, C2);
        s2 = ode_function(t_rk4(k) + h/2, y_rk4(k, :) + h/2 .* s1', V, R1, R2, L1, L2, M, C1, C2);
        s3 = ode_function(t_rk4(k) + h/2, y_rk4(k, :) + h/2 .* s2', V, R1, R2, L1, L2, M, C1, C2);
        s4 = ode_function(t_rk4(k) + h, y_rk4(k, :) + h .* s3', V, R1, R2, L1, L2, M, C1, C2);
        y_rk4(k + 1, :) = y_rk4(k, :) + h * (s1/6 + s2/3 + s3/3 + s4/6)';
        t_rk4(k + 1) = t_rk4(k) + h;
    end
end
```