Analysis of LTI Circuits with Differential-Integral Equations

MohammadMahdi Shokrzadeh

Electrical Circuits - Dr. Hadi

June 29, 2024

Abstract

This report presents a comprehensive analysis and design methodology for linear time-invariant (LTI) circuits characterized by differential-integral equations. The primary objective is to derive the differential equations that govern the behavior of the circuits, determine their impulse response, and obtain their frequency response. Additionally, the report explores the concept of the minimal differential equation that represents the system. The proposed methods are implemented in MATLAB, and the results are validated through practical examples.

Contents

Preface

This report serves as the preface to a comprehensive analysis and design methodology for linear time-invariant (LTI) circuits characterized by differential-integral equations. The objective of this work is to derive the differential equations that govern the behavior of LTI circuits, determine their impulse response, and obtain their frequency response. Additionally, the report explores the concept of the minimal differential equation that represents the system. The proposed methods are implemented in MATLAB, and the results are validated through practical examples.

The report begins by presenting the theoretical framework for analyzing LTI circuits using differential-integral equations. It explains the process of solving for specific circuit variables using Cramer's rule, which involves calculating determinants of modified matrices. The resulting differential equations provide a direct relationship between the desired circuit variables and the input waveform.

The report then delves into the concept of impulse response, which is the output of a system when the input is a Dirac delta function. It explains how to obtain the impulse response by setting the inputs of the circuit to the Dirac delta function and solving the resulting differential equation. The impulse response characterizes the system's reaction to an instantaneous impulse input.

Next, the report discusses the frequency response of LTI circuits, which describes how the system responds to sinusoidal inputs of varying frequencies. It introduces the steady-state sinusoidal response theorem, which provides the theoretical foundation for determining the frequency response. The amplitude and phase responses of the frequency response are explained, highlighting their significance in understanding the system's behavior.

The report also explores the concept of the minimal differential equation, which represents the lowest-order differential equation that governs the system's behavior. It explains the process of reducing the order of a differential equation by performing model reduction, and converting it back to a differential equation. This ensures that the resulting equation is both accurate and efficient for analysis and simulation.

1 Theoretical Background

Linear Time-Invariant (LTI) circuits are widely used in electrical engineering due to their predictable and stable behavior. The analysis of such circuits often involves solving differential-integral equations, which describe the relationship between input and output signals. In this section, we present the theoretical foundations for modeling LTI circuits using the D-operator, deriving the corresponding differential equations, and obtaining the impulse and frequency responses.

The problem of LTI circuits

Consider an LTI circuit characterized by the equation:

$$A\mathbf{Y} = \mathbf{W}$$

where A is an $n \times n$ matrix whose elements are multiples of D, D^{-1}, and 1. Here, $\mathbf{Y}$ is an $n \times 1$ vector of circuit variables, and $\mathbf{W}$ is an $n \times 1$ vector representing the input waveform.

1.1 D-Operator and Differential-Integral Equations

The D-operator is a mathematical tool used to represent differentiation and integration operations. For a function $y(t)$, the D-operator is defined as:

$$D = \frac{d}{dt}$$

Using this operator, we can express higher-order derivatives and integrals in a compact form. For instance:

$$D^2 y(t) = \frac{d^2 y(t)}{dt^2}, \quad D^{-1} y(t) = \int y(t)\, dt$$

1.2 Solving for Desired Circuit Variable

To solve for a specific circuit variable $y(t)$, located at the M-th entry of $\mathbf{Y}$, we use Cramer's rule. This involves calculating the determinant of A and a modified matrix A_m, where the m-th column is replaced by $\mathbf{W}$:

$$e_m(t) = \frac{\det(A_m)}{\det(A)}$$

This yields a differential equation for $y(t)$ in terms of the input waveform.

1.3 Impulse Response

The impulse response $h(t)$ of a system is the output when the input is a Dirac delta function $\delta(t)$. Therefore, the impulse response is found by setting the inputs of the circuit, such as the voltage of an independent voltage source or the current of a current source, to $\delta(t)$ and then solving the resulting differential equation.

By solving this equation with appropriate initial conditions, we obtain the impulse response $h(t)$, which characterizes the system's reaction to an impulse input.

1.4 Frequency Response

The frequency response $H(j\omega)$ describes how the system responds to sinusoidal inputs of varying frequencies. It is obtained by analyzing the steady-state response of the system to a sinusoidal input. The following theorem provides the theoretical foundation for determining the frequency response:

Theorem (Steady-State Sinusoidal Response)

If all the roots of the characteristic equation

$$\sum_{k=0}^{n} a_k s^k = 0$$

corresponding to the differential equation

$$\sum_{i=0}^{n} a_i y^{(i)}(t) = \sum_{l=0}^{m} b_l w^{(l)}(t), \quad y^{(i)}(0) = Y_i, \quad i = 0, 1, \ldots, n-1$$

are in the open left-hand complex plane, the steady-state sinusoidal response $y(t)$ to the input $w(t) = |A| \cos(\omega t + \angle A) = \Re\{Ae^{j\omega t}\}$ is of the form

$$y(t) = y_p(t) = |B| \cos(\omega t + \angle B), \quad t \geq 0$$

where the input phasor $A = |A|e^{j\angle A}$. The steady-state response phasor $B = |B|e^{j\angle B}$ is the solution of the equation

$$\frac{B}{A} = H(j\omega) = \frac{\sum_{l=0}^{m} b_l (j\omega)^l}{\sum_{i=0}^{n} a_i (j\omega)^i}$$

Here, $H(j\omega)$ is called the frequency response.

Amplitude and Phase Response

The frequency response $H(j\omega)$ is a complex function that can be expressed as:

$$H(j\omega) = |H(j\omega)|e^{j\angle H(j\omega)}$$

The amplitude response $|H(j\omega)|$ and the phase response $\angle H(j\omega)$ provide important insights into the system's behavior.

Amplitude Response: The amplitude response is the magnitude of the frequency response and is given by:

$$|H(j\omega)| = \sqrt{(\Re\{H(j\omega)\})^2 + (\Im\{H(j\omega)\})^2}$$

It is an even function, meaning that:

$$|H(j\omega)| = |H(-j\omega)|$$

Phase Response: The phase response is the argument of the frequency response and is calculated as:

$$\angle H(j\omega) = \tan^{-1}\left(\frac{\Im\{H(j\omega)\}}{\Re\{H(j\omega)\}}\right)$$

It is an odd function, meaning that:

$$\angle H(j\omega) = -\angle H(-j\omega)$$

In summary, the frequency response $H(j\omega)$ shows how the system processes sinusoidal inputs, with the amplitude response indicating the gain and the phase response indicating the phase shift introduced by the system at different frequencies.

1.5 Minimal Differential Equation

The minimal differential equation represents the lowest-order differential equation that governs the system's behavior. Reducing the order of a differential equation involves converting it to state-space form, performing model reduction, and converting it back to a differential equation. This process ensures the resulting equation is both accurate and efficient for analysis and simulation.

2 Solution of Differential Equation Using Cramer's Rule

Given the matrix equation representing the integral-differential equations of our LTI circuit:

$$\mathbf{AY} = \mathbf{W}$$

where $\mathbf{A}$ is an $n \times n$ matrix with elements that are multiples of $\{D, D^{-1}, 1\}$, $\mathbf{Y}$ is an $n \times 1$ vector of circuit variables, and $\mathbf{W}$ is an $n \times 1$ vector representing the input waveform, we aim to solve for a specific variable in the $\mathbf{Y}$ vector.

2.1 Cramer's Rule for Solution

To isolate the desired variable $y_N(t)$ in the N-th entry of $\mathbf{Y}$, we use Cramer's Rule. For a general system:

$$\mathbf{AY} = \mathbf{W}$$

the solution for the i-th element of $\mathbf{Y}$ can be expressed as:

$$y_i(t) = \frac{\det(\mathbf{A}_i)}{\det(\mathbf{A})}$$

where $\mathbf{A}_i$ is the matrix obtained by replacing the i-th column of $\mathbf{A}$ with the vector $\mathbf{W}$.

In our specific case, to solve for the N-th variable $y_N(t)$, we form the modified matrix $\mathbf{A}_N$ by replacing the N-th column of $\mathbf{A}$ with $\mathbf{W}$. The differential equation governing $y_N(t)$ is then given by:

$$\det(\mathbf{A})y_N(t) = \det(\mathbf{A}_N)$$

In the code, this process is done by **solve circuit** function:

```
function eq = solve_circuit(A, W, m)
    [n, ~] = size(A);

    if length(W) ~= n || m > n
        error('Dimensions of A and W must match, and m must
           be within bounds.');
    end

    syms t D is_(t)
    A_det = det(A);
    A_mod = A;
    A_mod(:, m) = W;
    A_mod_det = det(A_mod);

    syms em(t);
    equation = A_det * em(t) == A_mod_det;
    equation = expand(equation);

    eq = collect(equation, em(t));
    %disp("base eq");
    %disp(eq);
    eq_lhs = lhs(eq);
    eq_rhs = rhs(eq);

    [num_lhs, den_lhs] = numden(eq_lhs);
    [num_rhs, den_rhs] = numden(eq_rhs);
    min_power_lhs = min(feval(symengine, 'degree', den_lhs,
       D));
    min_power_rhs = min(feval(symengine, 'degree', den_rhs,
       D));
    min_power = max(min_power_lhs, min_power_rhs);
    if min_power > 0
        eq = eq * D^(min_power);
        eq = expand(eq);
        eq = collect(eq, em(t));
    end
    eq = collect(eq,is_(t));
    disp("Collected D-form equation:");
    disp(eq);
    eq = expand(eq);

end
```

../code/solve_circuit.m

2.2 Conversion to Differential Equation

To convert the operator form to a differential equation, we replace the D-operator with differential operators. The D-operator D corresponds to differentiation with respect to time t:

$$D \to \frac{d}{dt}$$

Similarly, D^{-1} corresponds to the integration operator:

$$D^{-1} \to \int dt$$

By substituting these operators into the equation:

$$\det(\mathbf{A}) y_N(t) = \det(\mathbf{A}_N)$$

we convert the equation from the D-operator form to a standard differential equation involving derivatives of $y_N(t)$ and the input waveform $w(t)$. This differential equation provides a direct relationship between the desired output $y_N(t)$ and the input $w(t)$, allowing for further analysis such as impulse and frequency responses.
The general form of the differential equation derived from our system is:

$$\sum_{k=0}^{n} a_k \frac{d^k y_N(t)}{dt^k} = \sum_{l=0}^{m} b_l \frac{d^l w(t)}{dt^l}$$

where the coefficients a_k and b_l are determined by the elements of $\mathbf{A}$ and $\mathbf{W}$, respectively.

This process is implemented in the MATLAB code, which calculates the determinant of the modified matrix and constructs the differential equation for the desired circuit variable, converting it from D form to the standard differential equation form. It is done by **convert to differential** function:

```
function diff_eq = convert_to_differential(eq)

    eq_lhs = expand(lhs(eq));
    eq_rhs = expand(rhs(eq));

    % Substitutiing D with the differential operator
    if length(children(eq_lhs)) == 1
        terms_lhs = {eq_lhs};
    else
        terms_lhs = children(eq_lhs);
    end
    diff_eq_lhs = 0;
    for term = terms_lhs
        term_temp = term{1};
        diff_eq_lhs = diff_eq_lhs +
           substitute_differential(term_temp);
    end

    if length(children(eq_rhs)) == 1
        terms_rhs = {eq_rhs};
    else
        terms_rhs = children(eq_rhs);
    end

     diff_eq_rhs = 0;
     for term = terms_rhs
         term_tem = term{1};
         diff_eq_rhs = diff_eq_rhs +
            substitute_differential(term_tem);
     end
    diff_eq = diff_eq_lhs == diff_eq_rhs;
end



function result = substitute_differential(term)
    syms D  t em(t) ;
    if has(term, D)
        [coeff, powers] = coeffs(term, D);
        result = 0;
        for i = 1:length(powers)
```

```
            pow = powers(i);
            pow = extract_power(pow);
            coeff_term = coeff(i);
            if pow == 1
                result = result + diff(coeff_term , t);
            else
                result = result + diff(coeff_term, t, pow);
            end
        end
    elseif has(term, 1/D)
        [coeff, powers] = coeffs(term, 1/D);
        result = 0;
        for i = 1:length(powers)
            pow = powers(i);
            coeff_term = coeff(i);
            if pow == -1
                result = result + int(coeff_term * em(t), t);
            else
                result = result + int(coeff_term * em(t), t,
                   -pow);
            end
        end
    else
        result = term;
    end
end
```

../code/convert_to_differential.m

In the next sections, we will analyze the impulse and frequency responses of this differential equation to understand the system's behavior under different input conditions.

3 Impulse Response

The impulse response of a system characterizes its output when subjected to a Dirac delta function as input. For LTI circuits, this response provides insight into the system's behavior and stability.

3.1 Methodology

To determine the impulse response, we set the input of the circuit (voltage or current source) to a Dirac delta function, $\delta(t)$. Mathematically, the Dirac delta function is defined as:

$$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(t)\, dt = 1$$

When the input $w(t)$ is set to $\delta(t)$, the differential equation governing the desired output $y_N(t)$ becomes:

$$\sum_{k=0}^{n} a_k \frac{d^k y_N(t)}{dt^k} = \sum_{l=0}^{m} b_l \frac{d^l \delta(t)}{dt^l}$$

3.2 Calculation

In our specific case, we substitute $\delta(t)$ for the input waveform in the derived differential equation. This substitution transforms the equation to:

$$\sum_{k=0}^{n} a_k \frac{d^k y_N(t)}{dt^k} = \sum_{l=0}^{m} b_l \frac{d^l \delta(t)}{dt^l}$$

To solve this equation, we employ symbolic computation methods to find $y_N(t)$, which represents the system's impulse response $h(t)$. The solution $h(t)$ is then obtained using MATLAB's `dsolve` function, with initial conditions set to zero.

This process is implemented in the MATLAB code, which calculates and plots the impulse response:

```matlab
function h_t = find_impulse_response(diff_eq)
    syms em(t) is_(t) v_s(t) y(t)

    diff_eq = subs(diff_eq, em(t), y(t));


    right_hand_eq = rhs(diff_eq);
    right_hand_eq = subs(right_hand_eq, is_(t), dirac(t));
    right_hand_eq = subs(right_hand_eq, v_s(t), dirac(t));

    final_eq = lhs(diff_eq) == right_hand_eq;
    disp("Final impulse equation:");
    disp(final_eq);
    lastwarn('');

    % h_t = dsolve(final_eq, y(0) == 0, diff(y(0), t) == 0);
    h_t = dsolve(final_eq, y(0) == 0,diff(y(0), t) ==
       0,diff(y(0),t, t) == 0 ,diff(y(0),t, t,t) == 0 );
    [warnMsg, warnId] = lastwarn;
     if ~isempty(warnMsg)
        warning('Could not solve the equation!');
        return;
    end



    free_vars = symvar(h_t);
    free_vars = setdiff(free_vars, t); % Exclude the time
       variable t
    if ~isempty(free_vars)
```

```
        for i = 1:length(free_vars)
            h_t = subs(h_t, free_vars(i), 1);
        end
    end

    disp("Final answer:");
    disp(h_t);

    h_t_func = matlabFunction(h_t, 'Vars', t);

    fplot(h_t_func, [0, 30]);
    title('Impulse Response');
    xlabel('Time (t)');
    ylabel('h(t)');
    ylim([0, 1.1]);

end
```

../code/find_impulse_response.m

3.3 Examples

Consider a simple first-order system with the differential equation:

$$\frac{dy(t)}{dt} + 3y(t) = \delta(t)$$

The impulse response $h(t)$ is found by solving:

$$\frac{dy(t)}{dt} + 3y(t) = \delta(t)$$

with initial condition $y(0) = 0$. Solving this using MATLAB, we get:

$$h(t) = e^{-3t} \times sign(t) \quad \text{for } t \geq 0$$

3.4 Plots

The impulse response can be visualized through a plot of the output $h(t)$ against time t. Here is the plot for the example in the previous section, which is generated by the code:

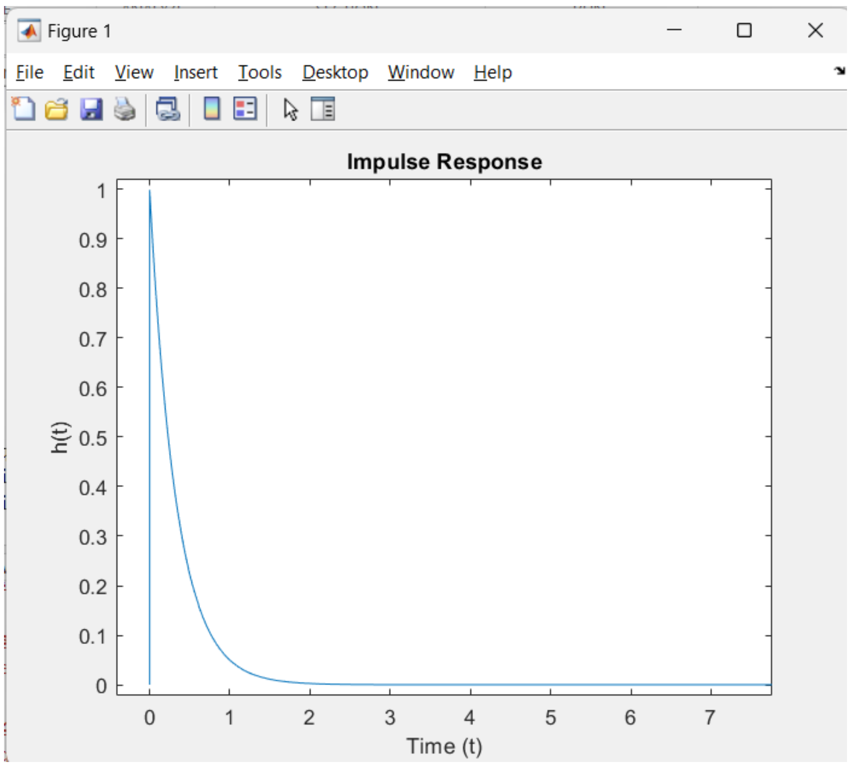


Figure 1: Impulse Response of the System

4 Frequency Response

The frequency response of a system describes how the system responds to sinusoidal inputs at various frequencies. It is a crucial characteristic in understanding the behavior of LTI circuits.

4.1 Calculation

The frequency response $H(j\omega)$ can be computed as the ratio of the polynomial in $j\omega$ formed from the coefficients of the input and output differential equations:

$$H(j\omega) = \frac{\sum_{l=0}^{m} b_l (j\omega)^l}{\sum_{i=0}^{n} a_i (j\omega)^i}$$

The frequency response consists of two components:

- **Amplitude Response** $|H(j\omega)|$: This is an even function, representing the magnitude of the system's response at different frequencies.
- **Phase Response** $\angle H(j\omega)$: This is an odd function, representing the phase shift introduced by the system at different frequencies. It can be calculated as:

$$\angle H(j\omega) = \tan^{-1}\left(\frac{\text{Imaginary part of } H(j\omega)}{\text{Real part of } H(j\omega)}\right)$$

This process is implemented in the MATLAB code, which calculates the frequency response and plots the amplitude and phase responses:

```matlab
function H_jw = find_frequency_response(eq)
    syms em(t) is_(t) v_s(t) D w ;
    eq_lhs = expand(lhs(eq));
    eq_rhs = expand(rhs(eq));
    num =  return_need(eq_rhs);
    dinom = return_need(eq_lhs);

    H_jw = num/dinom;
    H_jw = simplify(H_jw);
    disp("H_jw:");
    disp(H_jw);
    if has(H_jw, t)
        H_jw = subs(H_jw, t, 2*pi/w);
    end
    H_jw_func = matlabFunction(H_jw, 'Vars', w);

    % where till where?
    freq_range = -10:0.1:10;
    H_jw_vals = arrayfun(H_jw_func, freq_range);

    % Amplitude res
    figure;
    subplot(2, 1, 1);
    plot(freq_range, abs(H_jw_vals));
    title('Magnitude Response');
    xlabel('Frequency (rad/s)');
    ylabel('|H(j\omega)|');

    % Phase res
    subplot(2, 1, 2);
    plot(freq_range, unwrap(angle(H_jw_vals)));
    title('Phase Response');
    xlabel('Frequency (rad/s)');
    ylabel('<H(j\omega) (radians)');
end


function R_t = return_need(eq)
    result = 0;
    syms s t em(t) is_(t) v_s(t) w D tem_i
    temp_i = sqrt(-1);
    eq = subs(eq , em(t) , 1);
    eq = subs(eq , v_s(t) , 1);
    eq = subs(eq , is_(t) , 1);

    num = 0;
    terms_lhs = children(eq);
    diff_eq_lhs = 0;
    for term = terms_lhs
        term = term{1};
        if has(term, D)
            [coeff, powers] = coeffs(term, D);
            for i = 1:length(powers)
                pow = powers(i);
                pow = extract_power(pow);
                coeff_term = coeff(i);
                if pow == 1
                    result = result + coeff_term *
                       (w*temp_i);
                else
                    result = result + coeff_term *
                       (w*temp_i)^pow;
                end
            end
        else
           result = result + term;
        end
    end
    R_t = result;
end
```

../code/find_frequency_response.m

4.2 Example

Consider the same first-order system:

$$\frac{dy(t)}{dt} + 3y(t) = w(t)$$

The frequency response $H(j\omega)$ is given by:

$$H(j\omega) = \frac{1}{j\omega + 3}$$

The amplitude response is:

$$|H(j\omega)| = \frac{1}{\sqrt{\omega^2 + 9}}$$

The phase response is:

$$\angle H(j\omega) = \tan^{-1}\left(-\frac{\omega}{3}\right)$$

4.3 Plots

Here are the plots for example in the previous section which is generated by the code:

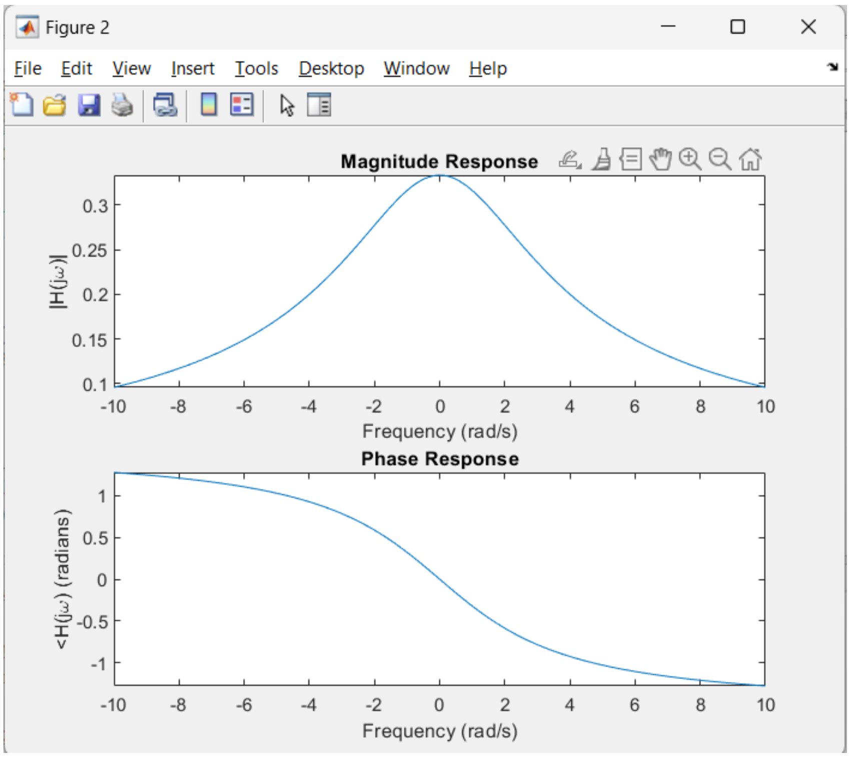


Figure 2: Amplitude and Phase Response of the System

5 Minimal Differential Equation

5.1 Methodology

The minimal differential equation is pivotal in system modeling as it represents the simplest form that captures essential system dynamics without unnecessary complexity. Several methods can be employed to identify this minimal form:

- Balanced Truncation: This method aims to reduce the system's order by preserving the most significant dynamics while minimizing the impact of higher-order modes. It involves computing controllability and observability Gramians to identify the most influential states and then truncating those with minimal contribution.
- State-Space Realization: By transforming a high-order differential equation into a state-space representation, it becomes feasible to manipulate and reduce the model. This approach utilizes matrix computations to convert differential equations into a more manageable form, facilitating subsequent reduction techniques.
- Direct Differentiation: This straightforward technique involves differentiating the original differential equation multiple times until higher-order terms are eliminated. While conceptually simple, it can be computationally intensive and may require symbolic computation tools to handle arbitrary order derivatives effectively.

I chose to use the balanced truncation method to reduce the order of the system and derive the minimal differential equation.

5.2 Calculation

To find the minimal differential equation for our system, we followed these steps:

1. Transfer Function: We first defined the transfer function of the system using the provided coefficients of the numerator and denominator.
2. Model Reduction: Using the 'balred' function in MATLAB, we performe balanced truncation for model reduction.
3. Convert Back to Differential Equation: Then we convert the reduced model back to differential equation form.

Here is the MATLAB code:

```
num = [3,2 , 4];
den = [5, 2 ,2,1 ,1 ];
sys = tf(num, den);

[sys_reduced, ~] = balred(sys, 3);

t = linspace(0, 10, 1000);

u = ones(size(t));
y_original = lsim(sys, u, t);
```

```matlab

y_reduced = lsim(sys_reduced, u, t);

figure;
plot(t, y_original, 'b', 'DisplayName', 'Original System');
hold on;
plot(t, y_reduced, 'r--', 'DisplayName', 'Reduced System');
xlabel('Time (s)');
ylabel('Response');
title('System Response Comparison');
legend;
grid on;


% disp('Reduced Transfer Function:');
% disp(sys_reduced);
syms y(t) u(t)
ode = poly2sym(den_reduced, t)*diff(y(t), t,
   length(den_reduced)-1) == ...
      poly2sym(num_reduced, t)*diff(u(t), t,
         length(num_reduced)-1);
disp('Minimal Differential Equation:');
disp(ode);
```

../code/findMinimalDiffEq.m

1. The code first defines a transfer function using the provided coefficients of the numerator (`num`) and the denominator (`den`).
2. It then reduces the order of the system using the balanced truncation method.
3. The time responses of both the original and reduced systems are computed.
4. These responses are then plotted together for comparison.
5. Finally, the reduced transfer function is converted back into a differential equation.

The coefficients `num` and `den` are derived directly from the system's differential equation. Specifically, `num` contains the coefficients of the derivatives of the input function $u(t)$ in the differential equation, and `den` contains the coefficients of the derivatives of the output function $y(t)$ in the differential equation. Therefore, given a differential equation of the form

$$a_n y^{(n)}(t)+a_{n-1}y^{(n-1)}(t)+\ldots+a_1 y'(t)+a_0 y(t) = b_m u^{(m)}(t)+b_{m-1}u^{(m-1)}(t)+\ldots+b_1 u'(t)+b_0 u(t)$$

we can directly say that

$$\texttt{den} = [a_n, a_{n-1}, \ldots, a_1, a_0]\ ,\ \texttt{num} = [b_m, b_{m-1}, \ldots, b_1, b_0]$$

5.3 Examples and Plots

Consider the following third-order system with the differential equation:

$$y'''(t) + 3y''(t) + 3y'(t) + y(t) = u'(t) + 2u(t)$$

Running the code, we get the reduced differential equation as:

```
diff(y(t), t)*(t + 4320148792823329/9007199254740992) ==
-((2147702724922091*t)/4503599627370496
- 8640297585646655/9007199254740992)*diff(u(t), t)
```

which is equivalent to:

$$\frac{dy(t)}{dt}\left(t + \frac{4320148792823329}{9007199254740992}\right) = -\left(\frac{2147702724922091t}{4503599627370496} - \frac{8640297585646655}{9007199254740992}\right)\frac{du(t)}{dt}$$

The code also shows a plot which compares the time responses of the original and reduced systems.

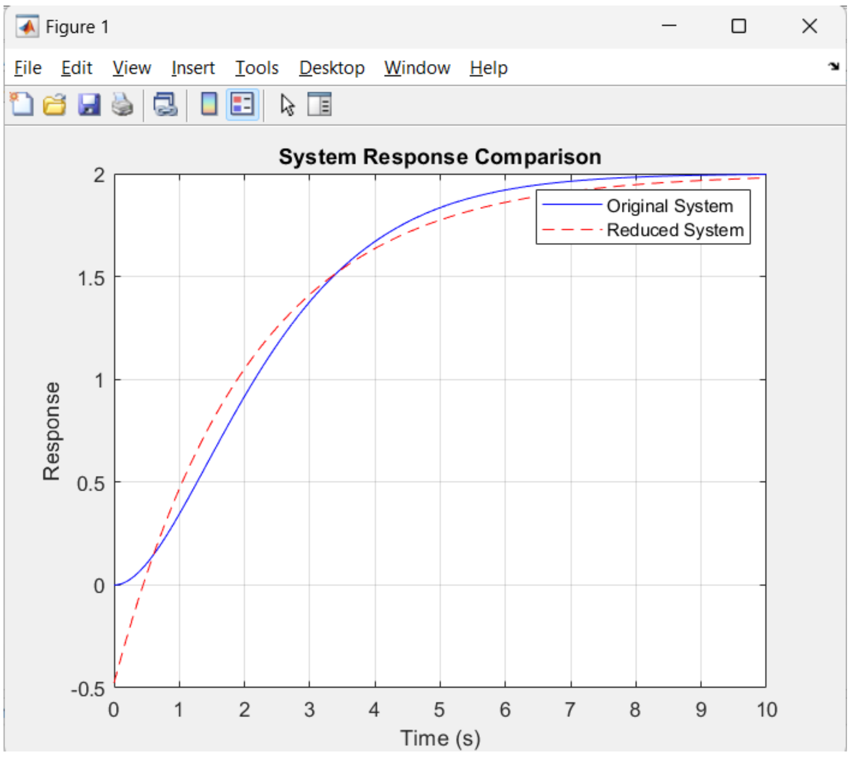


Figure 3: Time Response of Original and Reduced Systems

6 Examples

In this section, we provide examples to illustrate the application of the developed methods. These examples demonstrate the process of obtaining the differential equation, impulse response, and frequency response for various circuits.

6.1 Example 1

Consider the following parallel RLC circuit with an independent current source $i_s(t)$ and a resistor R, inductor L, and capacitor C:

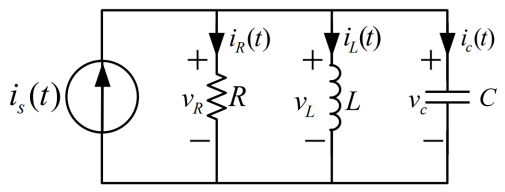


Figure 4: Parallel RLC Circuit

Writing KCL for node will return us the following equation:

$$i_L(0) + \frac{1}{L}\int v_c(t)\,dt + \frac{v_c(t)}{R} + C\frac{dv_c(t)}{dt} = i_s(t)$$

Since impulse response is held under state-zero condition,we will have:

$$\frac{1}{L}\int v_c(t)\,dt + \frac{v_c(t)}{R} + C\frac{dv_c(t)}{dt} = 0$$

Using the D-Operator:

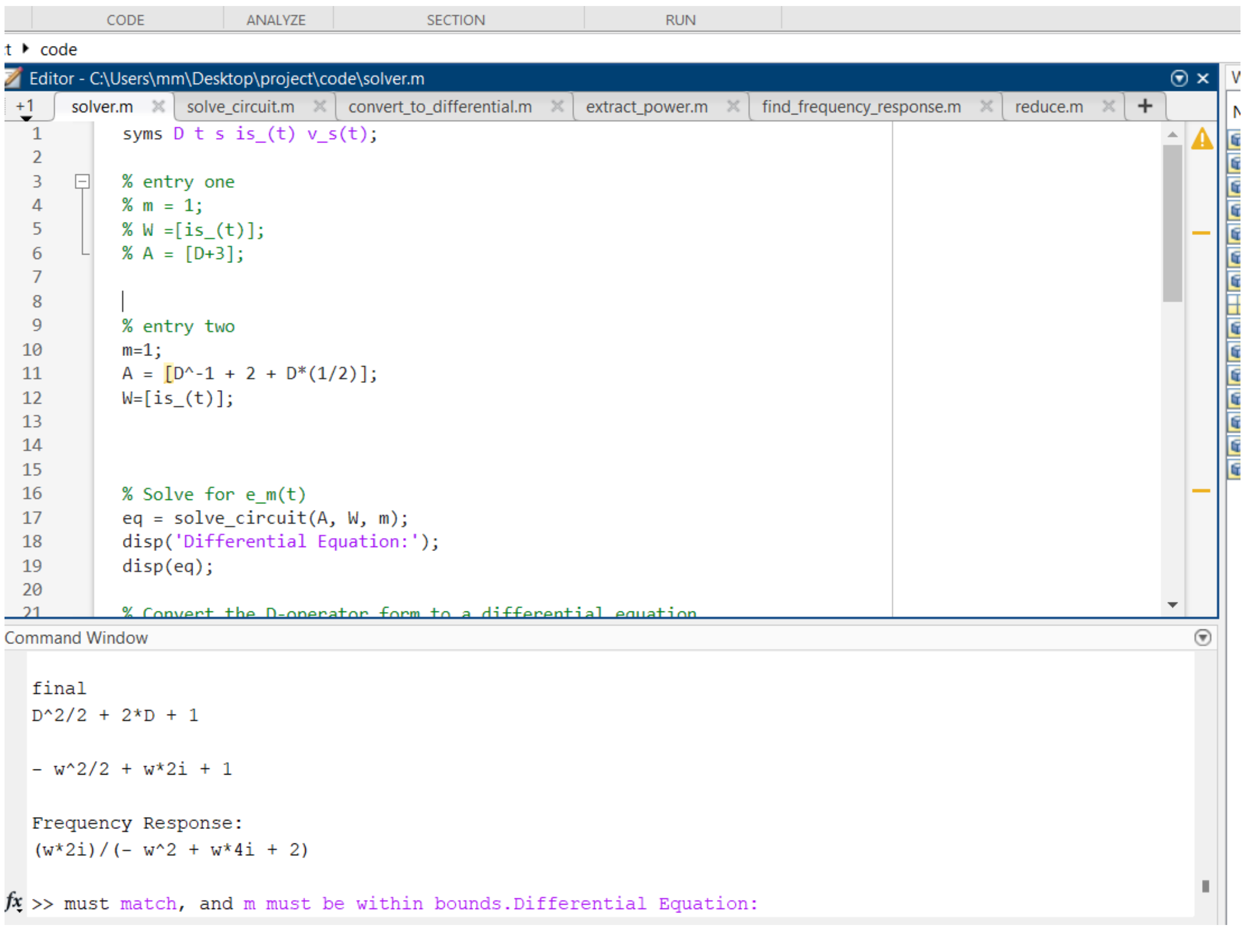
$$\frac{1}{L}D^{-1}v_c(t) + \frac{v_c(t)}{R} + CDv_c(t) = 0$$

Then A and W matrix will be as following, where $m = 1$:

$$A = \begin{bmatrix}\frac{1}{DL} + \frac{1}{R} + CD\end{bmatrix}$$

$$W = \begin{bmatrix}i_s(t)\end{bmatrix}$$

Considering the values of $R = \frac{1}{2}$, $L = 1$, and $C = \frac{1}{2}$, we put entries to the code (solver.m) as following:


CODE
ANALYZE
SECTION
RUN
code
Editor - C:\Users\mm\Desktop\project\code\solver.m
solver.m
solve_circuit.m
convert_to_differential.m
extract_power.m
find_frequency_response.m
reduce.m
Command Window

```
syms D t s is_(t) v_s(t);

% entry one
% m = 1;
% W =[is_(t)];
% A = [D+3];


% entry two
m=1;
A = [D^-1 + 2 + D*(1/2)];
W=[is_(t)];



% Solve for e_m(t)
eq = solve_circuit(A, W, m);
disp('Differential Equation:');
disp(eq);

% Convert the D-operator form to a differential equation
```

```
final
D^2/2 + 2*D + 1

- w^2/2 + w*2i + 1

Frequency Response:
(w*2i)/(- w^2 + w*4i + 2)

>> must match, and m must be within bounds.Differential Equation:
```

Figure 5: entries should be placed in solver.m

Running the code, we will get the impulse response of the circuit as following:

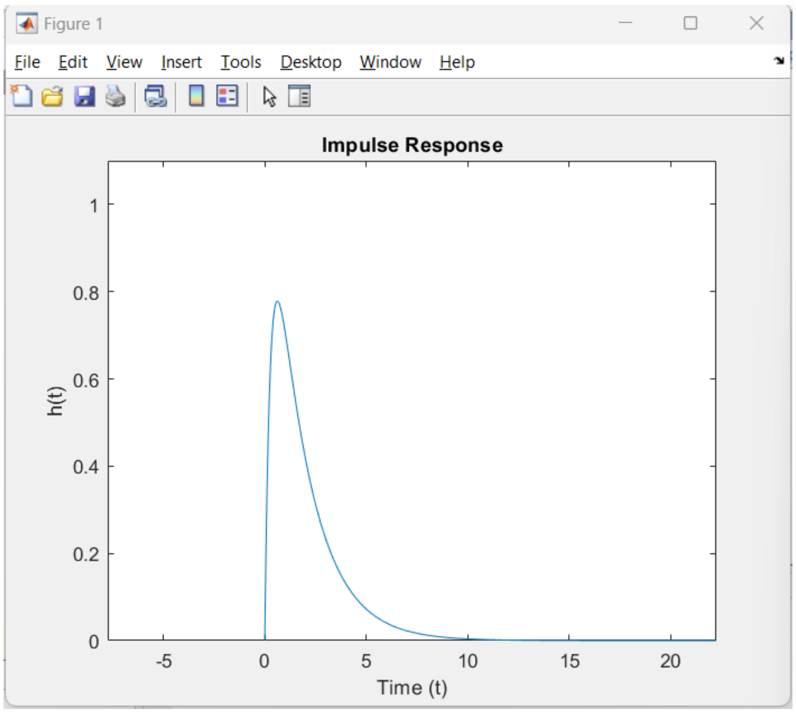


Figure 6: Impulse Response of the Circuit

Since $\alpha > w_0$ and $w_0, \alpha > 0$, the circuit is overdamped and the above result is as expected.

Also the amplitude and phase response of the circuit will be as following:

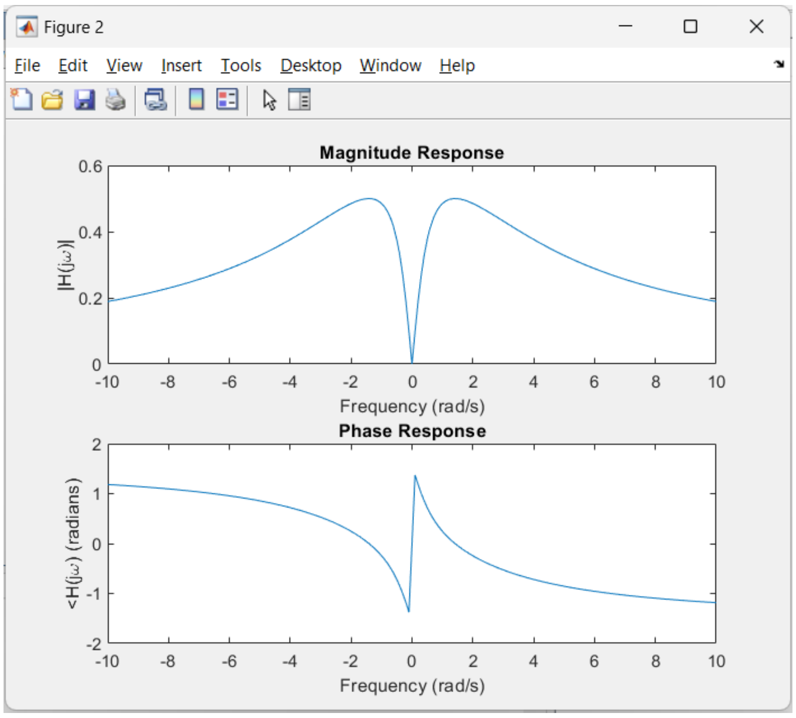


Figure 7: Amplitude and Phase Response of the Circuit, the code outputs the $H(jw)$ as $(w * 2i)/(-w^2 + w * 4i + 2)$

6.2 Example 2

Considering the same RLC circuit of last example, this time we solve solve for following values of R,L and C:

$$R = \frac{1}{2}, L = \frac{1}{4}, C = 1$$

Then, our A matrix will look like:

$$A = \left[4D^{-1} + 2 + 4D\right]$$

Running the code, we will get the impulse response of the circuit as following:

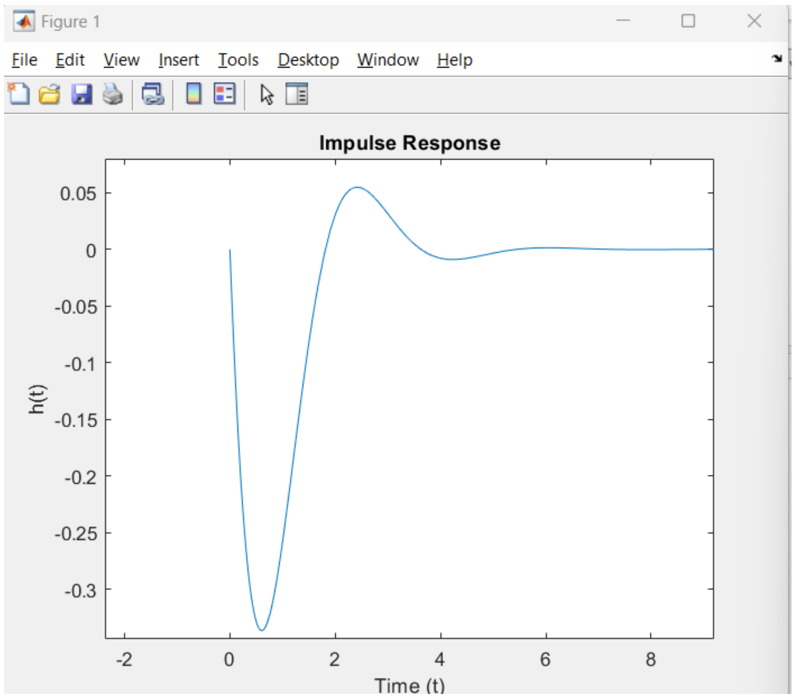


Figure 8: Impulse Response of the Circuit

Since $\alpha < w_0$ and $w_0, \alpha > 0$, the circuit is under damped and the above result is as expected.

The frequency and amplitude of the circuit will be as following:

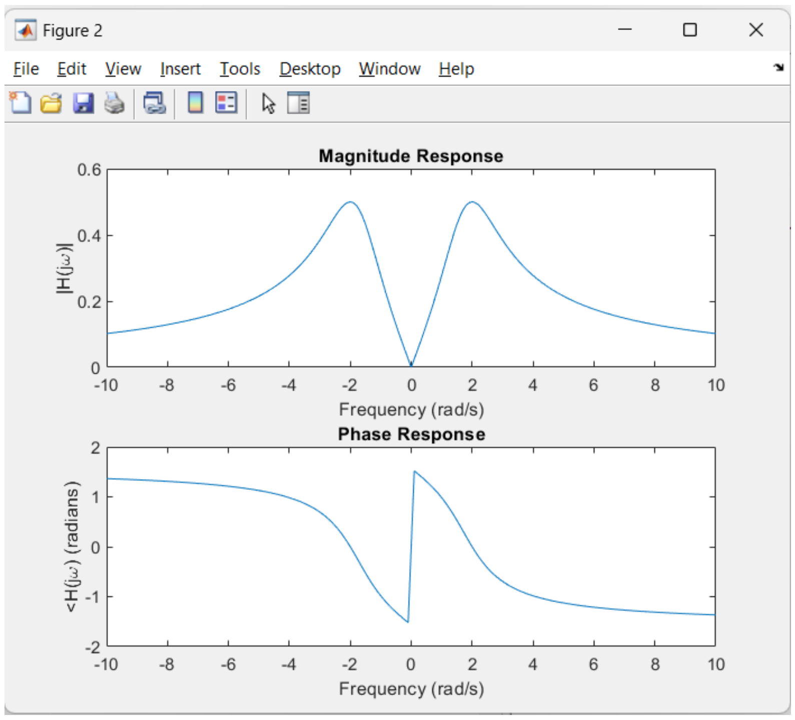


Figure 9: Amplitude and Phase Response of the Circuit, the code outputs the $H(jw)$ as $(w * 1i)/(-w^2 + w * 2i + 4)$

6.3 Example 3

Consider following circuit

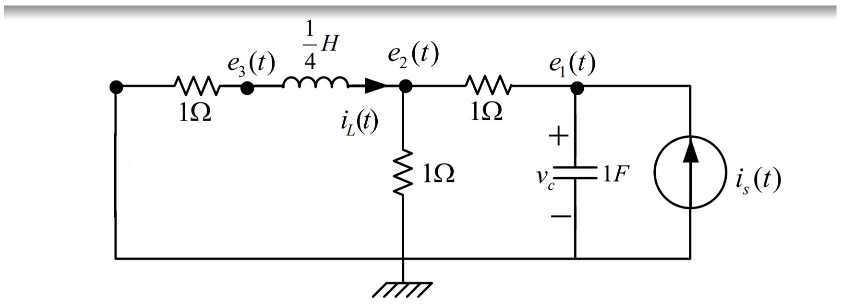


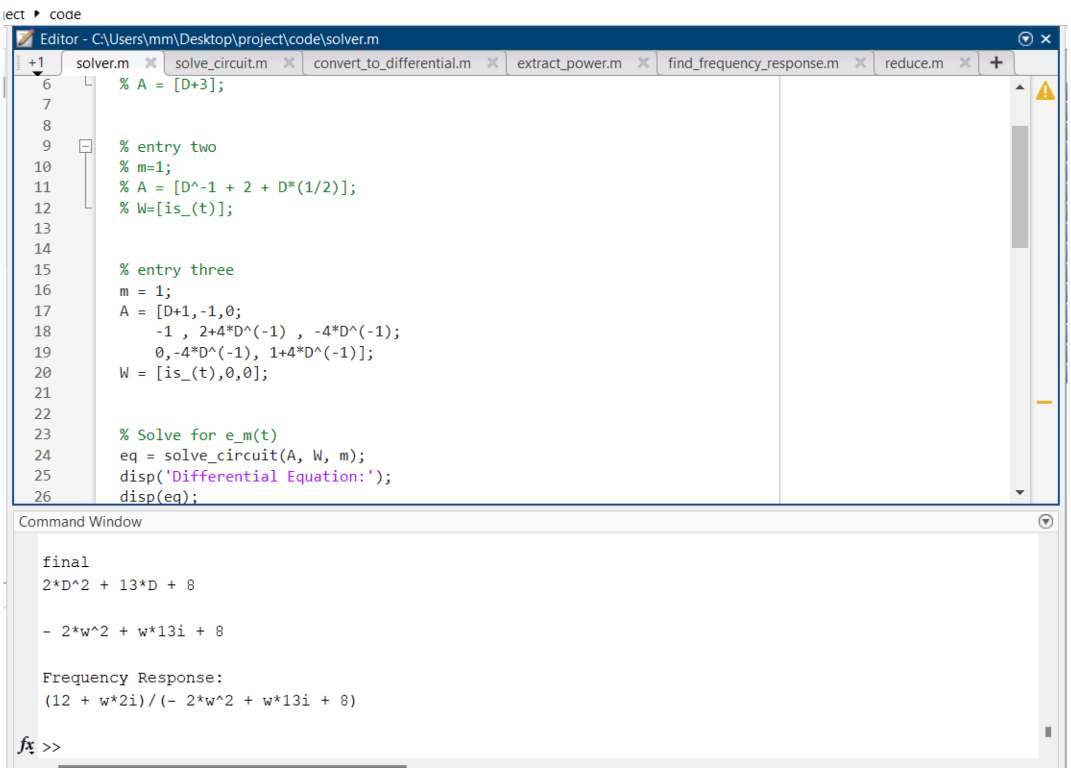
Figure 10: Circuit

Writing KCL for nodes, we will get the following matrixes A and W:

$$A = \begin{bmatrix} D+1 & -1 & 0 \\ -1 & 2+4D^{-1} & -4D^{-1} \\ 0 & -4D^{-1} & 1+4D^{-1} \end{bmatrix}$$

$$W = \begin{bmatrix} i_s(t) \\ 0 \\ 0 \end{bmatrix}$$

So the entries will look like as following in the code:



```
% A = [D+3];


% entry two
% m=1;
% A = [D^-1 + 2 + D*(1/2)];
% W=[is_(t)];


% entry three
m = 1;
A = [D+1,-1,0;
    -1 , 2+4*D^(-1) , -4*D^(-1);
    0,-4*D^(-1), 1+4*D^(-1)];
W = [is_(t),0,0];


% Solve for e_m(t)
eq = solve_circuit(A, W, m);
disp('Differential Equation:');
disp(eq);
```

```
final
2*D^2 + 13*D + 8

- 2*w^2 + w*13i + 8

Frequency Response:
(12 + w*2i)/(- 2*w^2 + w*13i + 8)

>>
```

Figure 11: entries should be placed in solver.m

Running the code, we will get the impulse response of the circuit for $e_1(t)$ as following:

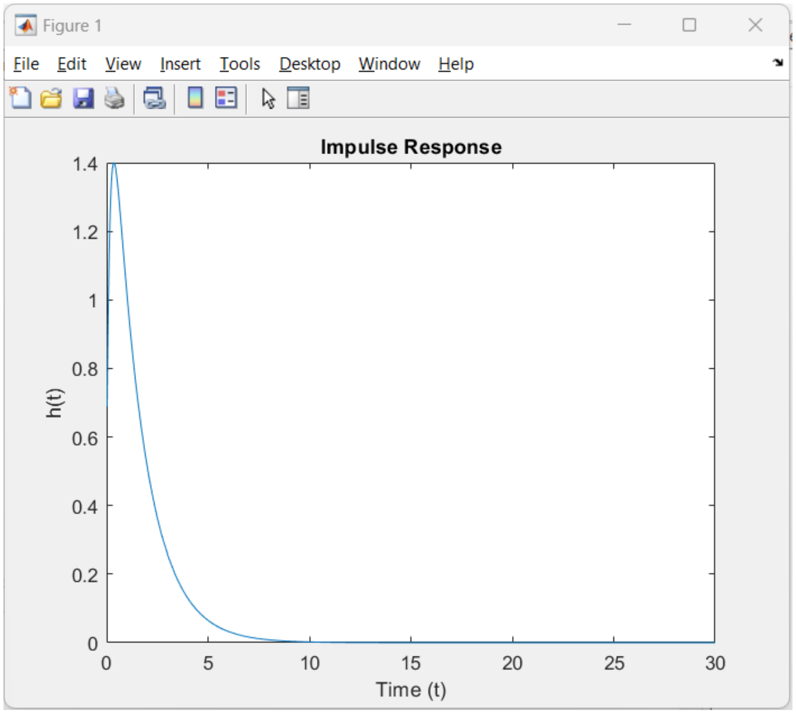


Figure 12: Impulse Response of the Circuit

The code outputs the D-form differential equation as:

$2 * em(t) * D^2 + 13 * em(t) * D + 8 * em(t) == 12 * is(t) + 2 * D * is(t)$

And the standard Matlab form as:

$8 * em(t) + 13 * diff(em(t), t) + 2 * diff(em(t), t, t) == 12 * is(t) + 2 * diff(is(t), t)$

The code also outputs the final impulse equation to be solved as:

$8 * y(t) + 13 * diff(y(t), t) + 2 * diff(y(t), t, t) == 12 * dirac(t) + 2 * dirac(1, t)$

And the final answer for impulse response as:

$exp(t * (105^{(1/2)}/4 - 13/4)) - (3 * exp(-t * (105^{(1/2)}/4 + 13/4)))/2 + exp(t * (105^{(1/2)}/4 - 13/4)) * (heaviside(t)/2 + (2 * 105^{(1/2)} * dirac(t))/105 + (11 * 105^{(1/2)} * heaviside(t))/210) - exp(-t * (105^{(1/2)}/4 + 13/4)) * ((2 * 105^{(1/2)} * dirac(t))/105 - heaviside(t)/2 + (11 * 105^{(1/2)} * heaviside(t))/210)$

Also the amplitude and phase response of the circuit will be as following:

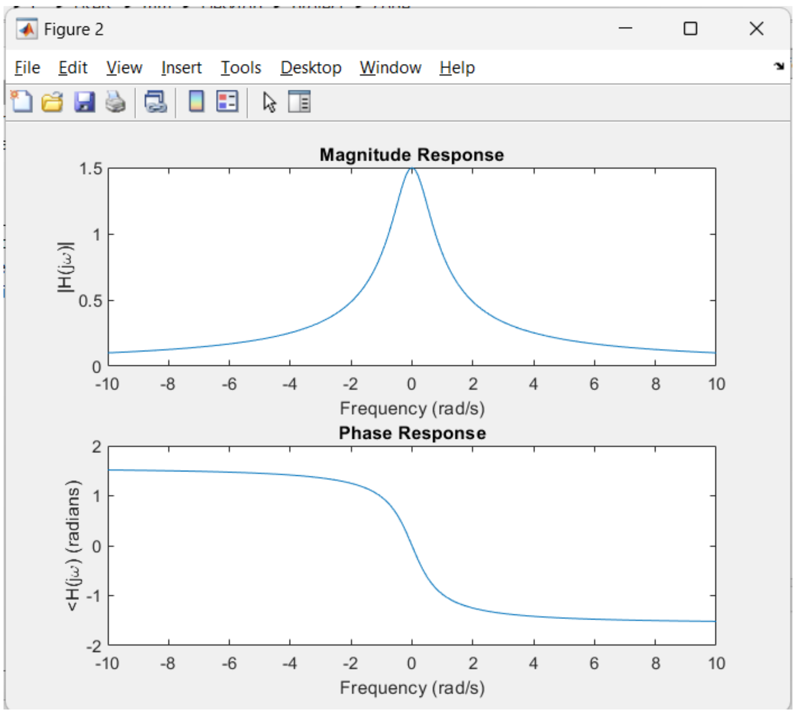


Figure 13: Amplitude and Phase Response of the Circuit, the code outputs the $H(jw)$ as $(12 + w * 2i)/(-2 * w^2 + w * 13i + 8)$

7 Conclusion

In this project, I developed a systematic approach to analyze LTI circuits described by integral-differential equations. The key steps included:

- Deriving the differential equation from the matrix equation using Cramer's rule.
- Converting the differential equation to a standard form.
- Computing the impulse response by setting the input to a Dirac delta function.
- Determining the frequency response using the steady-state sinusoidal response theorem.

The methods were implemented in MATLAB, and the results were verified through examples and plots. This approach provides a comprehensive framework for analyzing and understanding the behavior of LTI circuits.