NETWORK ANALYSIS AND SCIENTIFIC COMPUTING LABORATORY

III Semester: EEE								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
AEEC08	**Core**	**L**	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
		1	0	2	2	30	70	100
Contact Classes: Nil	**Tutorial Classes: Nil**	**Practical Classes: 36**				**Total Classes:36**		
Prerequisite: Electrical Circuits, Linear Algebra and Calculus								

I. COURSE OVERVIEW:

The Network Analysis and Scientific Computing Laboratory is designed to give hands-on experience on virtual instrumentation through digital simulation techniques. These techniques enable the students in examining characteristics of DC and AC circuits, filters, solution of differential equation, generation of three phase and complex wave forms using MATLAB.

II. COURSE OBJECTIVES:

The students will try to learn:

I. Time varying characteristics of series and parallel circuits using MATLAB.
II. Transfer function of electrical circuits using MATLAB.
III. Relations between electrical quantities in complex electrical networks using MATLAB.
IV. The performance of single phase and three phase circuits using Lab View.

III. COURSE OUTCOMES:

CO1	Identify the symbols, tool kits and connections in Simulink environment for computing the quantities associated with electrical circuits
CO2	Examine the transfer function for studying transient response of RL, RC and RLC circuits.
CO3	Analyze the virtual instrumentation (VI) using control loops, arrays, charts and graphs.
CO4	Determine various alternating quantities of single phase and three phase signals generated in MATLAB/ LabVIEW.
CO5	Design the various sensors for measuring electrical and non-electrical quantities through digital simulation.

IV. SYLLABUS:

EXERCISES ON NETWORK ANALYSIS AND SCIENTIFIC COMPUTING LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

Introduction to MATLAB

Identify the symbols, tool kits and connections in Simulink environment for computing the quantities associated with electrical circuits.

MATLAB is a powerful software environment widely used in various fields, including electrical engineering and circuit analysis. It provides a range of symbols, toolkits, and connections related to electrical circuits to assist engineers and researchers in designing, analyzing, and simulating electrical circuits. Here's an introduction to some of the key features and tools MATLAB offers in this regard:

1. Symbolic Math Toolbox: MATLAB's Symbolic Math Toolbox allows you to perform symbolic computations, which are crucial for electrical circuit analysis. You can define symbolic variables for circuit components like resistors, capacitors, and inductors and manipulate algebraic equations symbolically. This is useful for solving circuit equations and obtaining transfer functions.
2. Simulink: Simulink is a graphical environment within MATLAB for modeling, simulating, and analyzing dynamic systems, including electrical circuits. It provides a vast library of pre-built electrical components and blocks, making it easier to create circuit models and simulate their behavior.
3. Control System Toolbox: This toolbox contains tools and functions for analyzing and designing control systems, which are essential for understanding the behavior of electrical circuits. You can use it to design controllers, analyze stability, and simulate the response of electrical systems.
4. Electrical Circuit Analysis Toolbox: There are also third-party toolboxes available for MATLAB, such as the Electrical Circuit Analysis Toolbox, that provide specialized tools for electrical circuit analysis. These toolboxes often include features like circuit simulation, AC/DC analysis, and more.
5. SimPowerSystems (formerly known as SimElectronics): SimPowerSystems is a specialized toolbox in Simulink for modeling and simulating electrical circuits and systems. It includes a wide range of electrical components, such as transformers, generators, and motors, allowing you to create detailed electrical system models.
6. Simscape Electrical: Simscape Electrical is another Simulink toolbox that focuses on modeling and simulating electrical power systems. It includes components like power electronics devices, electrical machines, and power generation and distribution elements.
7. Signal Processing Toolbox: This toolbox is valuable for analyzing signals within electrical circuits. You can use it for tasks like filtering noisy signals, performing Fourier analysis, and extracting important information from sensor data.
8. Data Acquisition Toolbox: If you're working with hardware experiments or data acquisition systems, this toolbox can help you connect MATLAB to external hardware, making it easier to acquire and analyze data from real-world electrical circuits.

9. Instrument Control Toolbox: This toolbox enables you to communicate with and control external instruments like oscilloscopes, signal generators, and mustimeters, which is beneficial for experimental work in electrical engineering.
10. Custom Functions and Scripts: MATLAB allows you to create custom functions and scripts to model and analyze specific aspects of electrical circuits. You can define your own functions for circuit equations, transfer functions, and more.
    Creating syntax programs for electrical circuit analysis in MATLAB often involves using built-in functions, toolboxes, and custom code to define circuits, simulate their behavior, and analyze results. Below are some examples of syntax programs for common tasks in electrical circuit analysis using MATLAB:

Hints

1. Defining Circuit Components and Equations:

```
 % Define symbolic variables
syms V1 V2 R1 R2 C1 L1 s

% Define resistor values
R1 = 100; % Ohms
R2 = 200; % Ohms

% Define capacitor and inductor values
C1 = 0.1; % Farads
L1 = 0.5; % Henrys

% Define Kirchhoff's laws for an RC circuit
Eq1 = V1 - R1 * I1 - V2 == 0;
Eq2 = V2 - R2 * I2 - I1/C1 == 0;

% Define Laplace domain equations
Eq1_Laplace = laplace (Eq1, t, s);
Eq2_Laplace = laplace (Eq2, t, s);
```

2. Solving Circuit Equations:

```
% Solve the Laplace domain equations symbolically
I1_Laplace = solve (Eq1_Laplace, I1);
I2_Laplace = solve (Eq2_Laplace, I2);

% Inverse Laplace transform to get time-domain solutions
I1_time = ilaplace (I1_Laplace, s, t);
I2_time = ilaplace (I2_Laplace, s, t);
```

3. Simulating Circuits in Simulink:

```
% Create a Simulink model for an RLC circuit
model = 'RLC_Circuit';
open_system(new_system(model));

% Add components and connections using Simulink blocks
add_block ('Simulink/Continuous/Resistor', [model '/R1']);
add_block ('Simulink/Continuous/Inductor', [model '/L1']);
add_block ('Simulink/Continuous/Capacitor', [model '/C1']);
add_block ('Simulink/Sources/Step', [model '/Step Voltage']);
```

```
add_line (model, 'Step Voltage/1', 'R1/1');
% Add connections as needed

% Set component values and simulation parameters
set_param ([model '/R1'], 'Resistance', '100');
set_param ([model '/L1'], 'Inductance', '0.5');
set_param ([model '/C1'], 'Capacitance', '0.1');
set_param (model, 'Stop Time', '5');

% Run the simulation
sim(model);
```

4. Analyzing Frequency Response:

```
% Define a transfer function for an RC circuit
num = [1];
den = [R1*C1 1];
sys = tf (num, den);

% Bode plot for frequency response
bode(sys);
```

5. Data Analysis and Plotting:

```
% Import experimental data from a file
data = import data('experimental_data.csv');
time = data (: 1);
voltage = data (: 2);

% Plot voltage-time graph
plot (time, voltage);
xlabel ('Time (s)');
ylabel ('Voltage (V)');
title ('Voltage vs. Time');
```

Try:

a. Circuit Equations:

i. Write down the Kirchhoff's Voltage Law (KVL) equation for a series RC circuit involving a resistor (R) and a capacitor (C). Assume a constant voltage source V_S.

ii. Write down the Kirchhoff's Voltage Law (KVL) equation for a series RL circuit involving a resistor (R) and an inductor (L). Assume a time-varying current source I_S(t).

b. MATLAB Code: Write MATLAB code to:

i. Solve the KVL equation for the RC circuit (from question 2a) symbolically for V_C as a function of time, assuming initial conditions.

ii. Create a Simulink model for the RL circuit (from question 2b) with appropriate components and connections. Set the simulation parameters and run a transient analysis for a specified time duration.

Hint: Save your MATLAB code as a separate script file (.m) for questions

c. Transfer Function:

i. Define the transfer function H(s) for the RC circuit in terms of Laplace variables.

ii. Plot the Bode plot of H(s) to analyze the frequency response of the RC circuit.

d. Data Analysis:

i. Import experimental voltage and time data from a CSV file into MATLAB.
ii. Plot the voltage-time graph and label the axes appropriately.

e. Circuit Toolbox:

Mention at least two MATLAB toolboxes that are useful for electrical circuit analysis. Explain briefly how each toolbox can assist in circuit analysis.

f. If R = 10 Ohms and the current is increased from 0 to 10 A with increments of 2A, write a MATLAB program to generate a table of current, voltage and power dissipation.

2. Transient Response of Series RL, RC and RLC Circuits:

Examine the time varying characteristics of series RL, RC and RLC circuits for given values of R, L and C using MATLAB software.

2.1 Transient Response of Series RL

In this MATLAB program:

i. Circuit parameters (resistance, inductance, and applied voltage) are defined.
ii. The time constant (τ) is calculated as L / R.
iii. A symbolic variable **t** is defined for time.
iv. The differential equation representing the transient response of the RL circuit is defined using **diff** and ==.
v. The **dsolve** function is used to solve the differential equation symbolically.
vi. A time vector (**'t_span'**) is created for the simulation.
vii. IL(t) is numerically evaluated for the given time vector.
viii. The transient response is plotted.

Hints

```
% Define circuit parameters
R = 100; % Resistance in ohms
L = 0.5; % Inductance in henrys
Vin = 10; % Applied voltage in volts

% Calculate the time constant τ
tau = L / R;

% Define the symbolic variable for time
syms t

% Define the differential equation for the current IL(t)
IL = sym('IL(t)');
eqn = L * diff(IL, t) + R * IL == Vin;

% Solve the differential equation
IL_solution = dsolve (eqn, IL);
IL_solution = simplify(IL_solution);

% Create a time vector and evaluate IL(t
time = 0:0.01:2*tau; % Adjust the time span as needed
IL_numeric = double (subs (IL_solution, t, time));

% Plot the transient response
plot (time, IL_numeric);
xlabel ('Time (s)');
ylabel ('Current (A)');
```

```
title ('Transient Response of RL Circuit');
grid on;
```

Try:

a. Circuit Parameters:

Given the following values:

i. Resistor (R): 150 ohms
ii. Inductor (L): 0.3 H
iii. Applied Voltage (V_in): 12 V

Calculate the time constant (τ) of the RL circuit.

b. Transient Analysis:

iv. Write down the first-order differential equation governing the transient response of the RL circuit in terms of the current (I_L) through the inductor.
v. Solve the differential equation symbolically to obtain the expression for I_L(t) as a function of time (t).

c. MATLAB Simulation: Write MATLAB code to:

i. Define the circuit parameters (R, L, and V_in).
ii. Simulate the transient response of the RL circuit using the 'I_{sim}' function. Use a time vector from 0 to 5τ with a suitable time step.
iii. Plot the transient response of the current (I_L) as a function of time. Label the axes appropriately.

d. Time Constant Verification:

i. Calculate the time constant τ numerically from the transient response obtained in question 3c.
ii. Compare the numerically calculated time constant with the value obtained in question 1. Explain any differences or similarities.

e. Analysis:

Based on the transient response plot obtained in question iii, discuss the behavior of the current in the RL circuit as it responds to the voltage step input. Specifically, explain the time constant, the initial current, and the behavior as time progresses.

2.2 Transient Response of Series RC

In this MATLAB code:

i. Circuit parameters such as resistance (R), capacitance (C), and applied voltage (Vin) are defined.
ii. The time constant (τ) is calculated as R * C.
iii. A symbolic variable t is defined for time.
iv. The differential equation representing the transient response of the RC circuit is defined using diff and ==.
v. The dsolve function is used to solve the differential equation symbolically.
vi. A time vector (t_span) is created for the simulation.
vii. VC(t) is numerically evaluated for the given time vector.
viii. The transient response is plotted, showing how the voltage across the capacitor changes over time in response to a step input voltage.

You can adjust the time span in t_span to observe the transient response for the desired duration.

Hints

```
% Define circuit parameters
R = 100; % Resistance in ohms
C = 0.1; % Capacitance in farads
Vin = 5; % Applied voltage in volts

% Calculate the time constant τ
tau = R * C;

% Define the symbolic variable for time
syms t

% Define the differential equation for the voltage VC(t) across the capacitor
VC = sym('VC(t)');
eqn = R * diff (VC, t) + VC == Vin;

% Solve the differential equation symbolically
VC_solution = dsolve (eqn, VC);
VC_solution = simplify(VC_solution);

% Create a time vector
t_span = 0:0.01:5*tau; % Adjust the time span as needed

% Evaluate VC(t) for the given time vector
VC_numeric = double (subs (VC_solution, t, t_span));

% Plot the transient response
plot (t_span, VC_numeric);
xlabel ('Time (s)');
ylabel ('Voltage across Capacitor (V)');
title ('Transient Response of RC Circuit');
grid on;
```

Try:

a. Circuit Parameters:

Given the following values:

i. Resistor (R): 220 ohms
ii. Inductor (L): 0.01 F
iii. Applied Voltage (V_in): 10 V

Calculate the time constant (τ) of the RC circuit.

b. Transient Analysis:

i. Write down the first-order differential equation governing the transient response of the RC circuit in terms of the voltage (V_C) across the capacitor.
ii. Solve the differential equation symbolically to obtain the expression for V_C(t) as a function of time (t).

c. MATLAB Simulation: Write MATLAB code to:

i. Define the circuit parameters (R, C, and V_in).
ii Simulate the transient response of the RC circuit using the 'lsim' function. Use a time vector from 0 to 5τ with a suitable time step.

iii. Plot the transient response of the voltage across the capacitor (V_C) as a function of time. Label the axes appropriately.

d. Time Constant Verification:

i. Calculate the time constant τ numerically from the transient response obtained in question 3c.
ii. Compare the numerically calculated time constant with the value obtained in question 1. Explain any differences or similarities.

e. Analysis:

Based on the transient response plot obtained in question c(iii), discuss the behavior of the current in the RL circuit as it responds to the voltage step input. Specifically, explain the time constant, the initial current, and the behavior as time progresses.

2.3 Transient Response of Series RLC

In this MATLAB code:

I. Circuit parameters such as resistance (R), inductance (L), capacitance (C), and applied voltage (Vin) are defined.
II. The natural frequency (wn) and damping ratio (zeta) are calculated based on R, L, and C.
III. The time constant (τ) is calculated as 1 / (wn * sqrt (1 - zeta^2)).
IV. A symbolic variable **'t'** is defined for time.
V. The differential equation representing the transient response of the RLC circuit is defined using **'diff'** and ==.
VI. The **dsolve** function is used to solve the differential equation symbolically.
VII. A time vector (**'t_span'**) is created for the simulation.
VIII. VC(t) is numerically evaluated for the given time vector.
IX. The transient response is plotted, showing how the voltage across the capacitor changes over time in response to the applied voltage.

You can adjust the time span in t_span as needed to observe the transient response for the desired duration.

Hints

```
% Define circuit parameters
R = 100; % Resistance in ohms
L = 0.2; % Inductance in henrys
C = 0.01; % Capacitance in farads
Vin = 10; % Applied voltage in volts

% Calculate the natural frequency (wn) and damping ratio (zeta)
wn = 1 / sqrt(L * C);
zeta = R / (2 * L * wn);

% Calculate the time constant (τ)
tau = 1 / (wn * (1 - zeta^2) ^0.5);

% Define the symbolic variable for time
syms t

% Define the differential equation for the voltage VC(t) across the capacitor
VC = sym('VC(t)');
eqn = L * diff (VC, t, 2) + R * diff (VC, t) + (1/C) * VC == Vin;

% Solve the differential equation symbolically
```

```
VC_solution = dsolve (eqn, VC);
VC_solution = simplify(VC_solution);

% Create a time vector
t_span = 0:0.001:5*tau; % Adjust the time span as needed

% Evaluate VC(t) for the given time vector
VC_numeric = double (subs (VC_solution, t, t_span));

% Plot the transient response
plot (t_span, VC_numeric);
xlabel ('Time (s)');
ylabel ('Voltage across Capacitor (V)');
title ('Transient Response of RLC Circuit');
grid on;
```

Try

a. Circuit Parameters:

Given the following values:

I. Resistor (R): 220 ohms
ii. Inductor (L): 0.1 H
iii. Capacitor (C): 0.01 F
iv. Applied Voltage (V_in): 12 V

Calculate the natural frequency (wn), damping ratio (zeta), and time constant (τ) of the RLC circuit.

b. Transient Analysis:

i. Write down the second-order differential equation governing the transient response of the
ii. RLC circuit in terms of the voltage (V_C) across the capacitor.
iii. Solve the differential equation symbolically to obtain the expression for V_C(t) as a function of time (t).

c. MATLAB Simulation: Write MATLAB code to:

i. Define the circuit parameters (R, L, C, and V_in).
ii. Simulate the transient response of the RLC circuit using the lsim function. Use a time vector
from 0 to 5τ with a suitable time step.
iii. Plot the transient response of the voltage across the capacitor (V_C) as a function of time. Label the axes appropriately.

d. Time Constant Verification:

i. Calculate the natural frequency (wn), damping ratio (zeta), and time constant (τ) numerically
from the transient response obtained in question 3c.
ii. Compare the numerically calculated values with the values obtained in question 1. Explain
any differences or similarities.

3. Solving Differential Equations

Perform the solution of differential equation which is representing mathematical model of electric circuit using MATLAB software.

Define the Differential Equation: First, define the differential equation that represents the circuit. This equation depends on the specific circuit components and the mathematical model you're using. For instance, consider an RC circuit with a resistor (R) and a capacitor (C) connected in series. The differential equation for the voltage across the capacitor (Vc) can be written as:

$$RC\frac{dV_c}{dt} + V_c - V_{in}$$

where RC is the time constant, $\frac{dV_c}{dt}$ represents the derivative of Vc with respect to time, and Vin is the applied voltage.

Convert to Symbolic Variables**:** In MATLAB, convert the equation into symbolic form using the **'syms'** function and define the symbolic variables. For example:

Hints

```
syms Vc(t) Vin R C
eqn = R * C * diff(Vc(t), t) + Vc(t) == Vin;
```

Solve the Differential Equation: Use the '**dsolve'** function to solve the differential equation symbolically:

```
Vc_solution = dsolve(eqn);
```

Define Initial Conditions: If your circuit has initial conditions (e.g., Vc(0) = 0), you can specify them in the '**dsolve'** function to get a more specific solution.

```
Vc_solution = dsolve(eqn, Vc(0) == 0); % Example initial condition
```

Plot the Solution: Create a time vector and evaluate the symbolic solution for Vc(t) for the given time vector. Then, plot the transient response:

```
t = 0:0.01:5*RC; % Adjust the time span as needed
Vc_numeric = double (subs (Vc_solution, t));
plot (t, Vc_numeric);
xlabel ('Time (s)');
ylabel ('Voltage across Capacitor (V)');
title ('Transient Response of RC Circuit');
```

Try

I. Write down the differential equation representing the voltage across the capacitor (Vc) in the RC circuit. Include all the relevant terms and variables.

II. Define the symbolic variables and the differential equation in MATLAB to represent the RC circuit.

III. Use the **dsolve** function to solve the differential equation symbolically for $V_c(t)$ Assume an initial condition $V_c(0) = 0$

IV. Write MATLAB code to:

Define the circuit parameters:

Resistance R: 100 ohms

Capacitance C: 0.01 F

Applied Voltage V_{in}: 10 V

V. Simulate and plot the transient response of the voltage across the capacitor $V_c(t)$ for a time span of 0 to 5 times the time constant (τ).

VI. Label the axes appropriately and provide a title for the plot.

VII. Calculate the time constant (τ) of the RC circuit using the given values of R and C

VIII. Based on the transient response plot obtained in question v, discuss the behavior of the voltage across the capacitor (Vc) as it responds to the voltage step input. Explain the concept of the time constant (τ) and its significance in the transient response.

4. Transfer Function of Electrical Circuit

Determine the voltage transfer function of series RLC electrical circuit for frequencies from 10Hz to 100KHz using MATLAB software.

To determine the voltage, transfer function of a series RLC electrical circuit for a range of frequencies from 10 Hz to 100 kHz using MATLAB, you can follow these steps

Define Circuit Parameters:

- Resistance (R) in ohms
- Inductance (L) in henrys
- Capacitance (C) in farads

Define Symbolic Variables:

- s as the complex frequency variable

Create the Transfer Function:

The transfer function (H(s)) for a series RLC circuit can be defined as follows:

$$\boldsymbol{H(s) = \frac{V_{out}(s)}{V_{in}(s)} - \frac{1}{LC^2+RCs+1}}$$

Where:

$\boldsymbol{V_{out}(s)}$ is the Laplace-transformed output voltage?

$\boldsymbol{V_{in}(s)}$ is the Laplace-transformed input voltage?

L is the inductance.

C is the capacitance.

R is the resistance.

s is the complex frequency variable.

Calculate and Plot the Frequency Response:

HINTS

```
% Define circuit parameters
R = 100; % Resistance in ohms
L = 0.1; % Inductance in henrys
C = 0.01e-6; % Capacitance in farads (1 uF)
frequencies = logspace (1, 5, 100); % Frequency range from 10 Hz to 100 kHz

% Define the complex frequency variable s
s = j*2*pi*frequencies;

% Calculate the transfer function H(s)
H = 1. / (L*C*s.^2 + R*C*s + 1);

% Calculate magnitude and phase of H(s)
magnitude_H = abs(H);
phase_H = angle(H);

% Plot the Bode plot (magnitude and phase)
subplot (2,1,1);
semilogx (frequencies, 20*log10(magnitude_H));
title ('Bode Plot of RLC Circuit');
xlabel ('Frequency (Hz)');
ylabel ('Magnitude (dB)');
grid on;

subplot (2,1,2);
semilogx (frequencies, rad2deg(phase_H));
xlabel ('Frequency (Hz)');
ylabel ('Phase (degrees)');
grid on;
```

In this code:

- We define the circuit parameters (R, L, and C).
- We create a range of frequencies from 10 Hz to 100 kHz using **'logspace'**.
- We define the complex frequency variable s as $\boldsymbol{j\omega}$, where ω is the angular frequency.
- The transfer function $\boldsymbol{H(s)}$ is calculated for each frequency in the range.
- We calculate the magnitude and phase of $\boldsymbol{H(s)}$
- Finally, we plot the Bode plot, showing the magnitude and phase response of the RLC circuit

as a function of frequency.

Try

I. a series RLC electrical circuit with the following parameters:

Resistance (R): 150 ohms

Inductance (L): 0.2 H

Capacitance (C): 0.01 F

Determine the voltage transfer function (H(s)) of the circuit for a range of frequencies from 10 Hz to 100 kHz.

II. Write down the equation for the voltage transfer function (H(S)) of the series RLC circuit in terms of R, L, C and the complex frequency variable

III. Write MATLAB code to:

a. Define the given circuit parameters (R, L, and C).
b. Generate a range of frequencies from 10 Hz to 100 kHz.
c. Calculate the complex frequency variable s for each frequency in the range.
d. Calculate the voltage transfer function (H(s)) for each frequency using the equation from question II.

IV. Magnitude and Phase Plot: Write MATLAB code to:

a. Calculate the magnitude and phase of H(s) for the entire frequency range.
b. Plot the Bode plot of the magnitude (in decibels) and phase (in degrees) of H(s) as a function of frequency.

V. Based on the Bode plot obtained in question IV(b), analyze and discuss the behavior of the series RLC circuit as the frequency changes from 10 Hz to 100 kHz. Explain how the circuit responds to different frequencies and highlight any resonance frequencies if observed.

5. Transient Response of Parallel RL, RC and RLC Circuits

Examine the time varying characteristics of parallel RL, RC and RLC circuits for given values of R, L and C using MATLAB software.

5.1 Transient Response of Parallel RL

1. **Define Circuit Parameters:**
   - Resistance (R) in ohms
   - Inductance (L) in henrys
   - Applied Voltage (V_{in}) in volts
2. **Define Symbolic Variables:**
   - t as the time variable
   - $I_L(t)$ as the current through the inductor as a function of time
3. **Create the Differential Equation:**

   The differential equation for a parallel RL circuit can be written as:

$$L\frac{dI_L(t)}{dt} + \frac{V_m}{R} = 0$$

   Where:

   - L is the inductance.
   - R is the resistance.

- $I_L(t)$ is the current through the inductor as a function of time.
- V_{in} is the applied voltage.

Hints

- **Solve the Differential Equation:**
  Use the '**dsolve**' function to solve the differential equation symbolically:
  HINT

```
% Define the differential equation
syms t
IL = sym('IL(t)');
eqn = L * diff (IL, t) + Vin / R == 0;

% Solve the differential equation symbolically
IL_solution = dsolve (eqn, IL);
IL_solution = simplify(IL_solution);
```

- **Create a Time Vector and Evaluate the Solution:**
  Create a time vector and evaluate $I_L(t)$ for the given time vector:

```
% Create a time vector
t_span = 0:0.01:5*tau; % Adjust the time span as needed

% Evaluate IL(t) for the given time vector
IL_numeric = double (subs (IL_solution, t, t_span));
```

- **Plot the Transient Response:**

Plot the transient response of the current $I_L(t)$ as a function of time:

```
% Plot the transient response
plot (t_span, IL_numeric);
xlabel ('Time (s)');
ylabel ('Current (A)');
title ('Transient Response of Parallel RL Circuit');
grid on;
```

In this code:

- We define the circuit parameters($\boldsymbol{R, L, and\ V_{in}}$).
- We create a symbolic variable t for time and a symbolic variable $\boldsymbol{I_L(t)}$ for the current through the inductor.
- We define the differential equation and solve it symbolically.
- We create a time vector for the simulation.
- We evaluate the symbolic solution for $\boldsymbol{I_L(t)}$ numerically.
- Finally, we plot the transient response of the current.

Try

1. **Circuit Parameters:**

   Given the following values:
   Resistor (R): 220 ohms
   Inductor (L): 0.1 H
   Applied Voltage (V_in): 12 V
   Calculate the time constant (τ) of the parallel RL circuit.

2. Transient Analysis:

a) Write down the first-order differential equation governing the transient response of the RL circuit in terms of the current $I_L(t)$ through the inductor.
b) Solve the differential equation symbolically to obtain the expression for $I_L(t)$ as a function of time (t).

3. MATLAB Simulation:

Write MATLAB code to:

a) Define the circuit parameters ($\boldsymbol{R, L, and\ V_{in}}$).
b) Simulate the transient response of the RL circuit using the **'lsim'** function. Use a time vector from 0 to 5τ with a suitable time step.
c) Plot the transient response of the current I_L through the inductor as a function of time. Label the axes appropriately.

4. Time Constant Verification:

a) Calculate the time constant (τ) numerically from the transient response obtained in question 3c.
b) Compare the numerically calculated time constant with the value obtained in question 1. Explain any differences or similarities.

5. Analysis:

Based on the transient response plot obtained in question 3c, discuss the behavior of the current (I_L) in the RL circuit as it responds to the voltage step input. Specifically, explain the time constant, the initial current, and the behavior as time progresses.

5.2 Transient Response of Parallel RC

To analyze the transient response of a parallel RC circuit in MATLAB, you can use the circuit's differential equation and solve it numerically. The transient response describes how the circuit behaves as it moves from one steady state to another when an input voltage or current changes suddenly.
Assuming you have a parallel RC circuit with a voltage source connected in parallel to a resistor (R) and a capacitor (C), you can write the differential equation for this circuit as follows:

$$V(t)=I(t)\cdot R+VC(t)$$

Where:

V(t) is the voltage across the circuit as a function of time.

I(t) is the current through the resistor as a function of time.

$V_C(t)$ is the voltage across the capacitor as a function of time.

You can use MATLAB to solve this differential equation and obtain the transient response.

Hints

1. Define the circuit parameters (R and C) and the input voltage waveform (V_in) as functions of time.

```
R = 1; % Resistor value in ohms
C = 0.1; % Capacitor value in farads

% Define the input voltage as a function of time (for example, a step input)
t = 0:0.01:5; % Time vector
V_in = zeros(size(t)); % Initialize the input voltage vector
V_in (t >= 0) = 1; % Set the input voltage to 1V for t >= 0
```

2. Write the differential equation based on the circuit equation and use MATLAB's ode45 function to solve it numerically.

```
% Define the differential equation for the parallel RC circuit
% dVc/dt = (Vin - Vc) / (R * C)
% Where Vc is the voltage across the capacitor as a function of time
dVc_dt = @(t, Vc) (V_in(t) - Vc) / (R * C);

% Initial condition (at t = 0, Vc = 0)
Vc0 = 0;

% Solve the differential equation using ode45
[t, Vc] = ode45(dVc_dt, t, Vc0);
```

3. Plot the transient response of the voltage across the capacitor.

```
% Plot the transient response
figure;
plot (t, Vc);
xlabel ('Time (s)');
ylabel ('Voltage across Capacitor (V)');
title ('Transient Response of Parallel RC Circuit');
grid on;
```

4. To simulate and analyze the transient response of a parallel RC (Resistor-Capacitor) circuit in MATLAB, you can use the following syntax:

```
% Define circuit parameters
R = 220; % Resistance in ohms
C = 0.01 % Capacitance in farads
Vin = 12;% Applied voltage in volts

% Calculate the time constant (tau)
tau = R * C;

% Define the symbolic variable for time
syms t

% Define the differential equation for the voltage VC(t) across the capacitor
VC = sym('VC(t)');
eqn = C * diff(VC, t) + VC / R == Vin;

% Solve the differential equation symbolically
VC_solution = dsolve(eqn, VC);
VC_solution = simplify(VC_solution);

% Create a time vector
```

```
t_span = 0:0.001:5*tau; % Adjust the time span as needed

% Evaluate VC(t) for the given time vector
VC_numeric = double (subs (VC_solution, t, t_span));

% Plot the transient response
plot (t_span, VC_numeric);
xlabel ('Time (s)');
ylabel ('Voltage across Capacitor (V)');
title ('Transient Response of Parallel RC Circuit');
grid on;
```

Try:

1. **Circuit Parameters:**
   Given the following values:
   Resistor (R): 100 ohms
   Capacitance (C): 0.01 F
   Applied Voltage (V_in): 10 V

Calculate the time constant (τ) of the parallel RC circuit.

2. **Transient Analysis:**
   a) Write down the first-order differential equation governing the transient response of the RC circuit in terms of the voltage (Vc) across the capacitor.
   b) Solve the differential equation symbolically to obtain the expression for Vc(t) as a function of time (t).

3. **MATLAB Simulation:**

Write MATLAB code to:

a) Define the circuit parameters (R, C, and V_in).
b) Simulate the transient response of the RC circuit using the **'lsim'** function. Use a time vector from 0 to 5τ with a suitable time step.
c) Plot the transient response of the voltage (Vc) across the capacitor as a function of time. Label the axes appropriately.

4. **Time Constant Verification:**
   a) Calculate the time constant (τ) numerically from the transient response obtained in question 3c.
   b) Compare the numerically calculated time constant with the value obtained in question 1. Explain any differences or similarities.

5.3 Transient Response of Parallel RLC

To calculate the transient response of a parallel RLC circuit in MATLAB, you can use the circuit's differential equation and solve it numerically. Here's an example of how you can do this:
Assuming you have a parallel RLC circuit with the following parameters:
Resistance (R)
Inductance (L)
Capacitance (C)
Voltage source (V)
Initial conditions (initial voltage across the capacitor and initial current through the inductor)

Here's the MATLAB code to calculate and plot the transient response of the parallel RLC circuit:

Hints

```
 % Define circuit parameters
R = 1;       % Resistance (in ohms)
L = 0.5;     % Inductance (in henries)
C = 0.2;     % Capacitance (in farads)
V = 5;       % Voltage source (in volts)

% Define initial conditions
Vc0 = 0;     % Initial voltage across the capacitor (in volts)
Il0 = 0;     % Initial current through the inductor (in amperes)

% Create a time vector
t = 0:0.01:5; % Time from 0 to 5 seconds with a step of 0.01 seconds

% Define the differential equation for the parallel RLC circuit
% d2Vc/dt^2 + (1/(R*C)) * dVc/dt + (1/(L*C)) * Vc = (1/L) * dIl/dt
% d2Il/dt^2 + (1/(R*L)) * dIl/dt + (1/(L*C)) * Il = (1/L) * dVc/dt
% Where Vc is the voltage across the capacitor and Il is the current through the
inductor

% Define a function that returns the derivatives of Vc and Il
dVcdt = @(t, Vc, Il) (1/L) * (V - Vc) - (1/(R*C)) * Vc;
dIldt = @(t, Vc, Il) (1/L) * (Vc - V) - (1/(R*L)) * Il;

% Use the ODE solver to solve the system of differential equations
initial_conditions = [Vc0, Il0];
[t,  y]  =  ode45(@(t,  y)  [dVcdt(t,  y(1),  y(2));  dIldt(t,  y(1),  y(2))],  t,
initial_conditions);

% Extract voltage across the capacitor and current through the inductor
Vc = y(:, 1);
Il = y(:, 2);

% Plot the transient response
subplot(2, 1, 1);
plot(t, Vc);
xlabel('Time (s)');
ylabel('Voltage across Capacitor (V)');
title('Transient Response of Parallel RLC Circuit');
grid on;

subplot(2, 1, 2);
plot(t, Il);
xlabel('Time (s)');
ylabel('Current through Inductor (A)');
grid on;
```

Try

1. Mathematical Modeling:
    a) Consider a parallel RLC circuit with the following parameters:
        - Resistance (R): 2 ohms
        - Inductance (L): 0.5 henries

- Capacitance (C): 0.1 farads
- Voltage source (V): 10 volts

b) Define the initial conditions as follows:
- Initial voltage across the capacitor (Vc0): 0 volts
- Initial current through the inductor (Il0): 0 amperes

c) Write the differential equations that describe the behavior of this parallel RLC circuit over time. Express these equations in terms of voltage across the capacitor (Vc) and current through the inductor (Il).

2. Using MATLAB, simulate and plot the transient response of the parallel RLC circuit for the following conditions:
- Time span: 0 to 5 seconds
- Time step: 0.01 seconds

You should create MATLAB scripts/functions to do the following:

a) Define the circuit parameters and initial conditions as specified in Problem 1.

b) Create a time vector that covers the time span.

c) Implement the differential equations for the parallel RLC circuit.

d) Use MATLAB's ODE solver (ode45) to solve the differential equations.

e) Plot the transient response, showing both the voltage across the capacitor and the current through the inductor on separate graphs.

3. Analysis and Interpretation

a) Analyze the plots obtained in Problem 2. Explain how the voltage across the capacitor and the current through the inductor change over time during the transient response.

b) Calculate and report the time constant (τ) of this parallel RLC circuit based on its parameters (R, L, and C). Discuss the significance of the time constant in relation to the transient response.

6. Generation of Three Phase Wave Form

In this code:

- Parameters such as frequency and amplitude are defined.
- Phase differences and phase sequences are specified in degrees.
- A time vector (t) is created to represent time.
- Phase angles in radians are calculated for each phase based on time and phase differences.
- Sinusoidal waveforms for each phase (Phase A, Phase B, and Phase C) are generated.
- Finally, the waveforms are plotted on separate subplots to visualize the three-phase AC waveform.
- You can adjust the 'frequency', 'amplitude', 'phase_diff_AB', 'phase_diff_BC', and 'phase_diff_CA' values to observe different phase differences and phase sequences in the generated waveform.

Hints

```
% Define parameters
frequency = 50;  % Frequency in Hertz
amplitude = 220;  % Amplitude of each phase in volts

% Define phase differences and phase sequences
phase_diff_AB = 0;   % Phase difference between phases A and B (in degrees)
phase_diff_BC = 120; % Phase difference between phases B and C (in degrees)
phase_diff_CA = 240; % Phase difference between phases C and A (in degrees)

% Create time vector
t = 0:0.0001:0.1;  % Time vector (0 to 0.1 seconds with a step of 0.0001 seconds)

% Calculate phase angles in radians
omega = 2 * pi * frequency;
theta_A = omega * t;
theta_B = omega * t - deg2rad(phase_diff_AB);
theta_C = omega * t - deg2rad(phase_diff_CA);

% Generate sinusoidal waveforms for each phase
phase_A = amplitude * sin(theta_A);
phase_B = amplitude * sin(theta_B);
phase_C = amplitude * sin(theta_C);

% Plot the three-phase waveforms
figure;
subplot (3, 1, 1);
plot (t, phase_A, 'b');
xlabel ('Time (s)');
ylabel ('Phase A Voltage (V)');
title ('Three-Phase AC Waveform');
grid on;

subplot (3, 1, 2);
plot (t, phase_B, 'r');
xlabel ('Time (s)');
ylabel ('Phase B Voltage (V)');
grid on;

subplot (3, 1, 3);
plot (t, phase_C, 'g');
xlabel ('Time (s)');
ylabel ('Phase C Voltage (V)');
grid on;

% Adjust plot settings for better visualization
linkaxes;

% Legend for phase sequence
legend ('Phase A', 'Phase B', 'Phase C');
```

Try

Problem 1: Mathematical Background

Explain the concept of three-phase AC power systems, including the significance of phase differences and

phase sequences. Discuss how the phase differences and sequences impact the operation of electrical systems.

Problem 2: MATLAB Programming

Write MATLAB code to generate three-phase AC waveforms with the following specifications:

- Frequency: 50 Hz
- Amplitude: 220 V

You should generate three phases (Phase A, Phase B, and Phase C) with different phase differences and sequences. Use a time vector with a duration of at least 0.1 seconds and a suitable time step.

a) Implement code to create sinusoidal waveforms for each phase with the specified frequency and amplitude.
b) Generate waveforms for the following scenarios:
   - Scenario 1: Balanced three-phase AC with a phase sequence of ABC and zero phase differences.
   - Scenario 2: Balanced three-phase AC with a phase sequence of BCA and phase differences of 120 degrees between phases.
   - Scenario 3: Unbalanced three-phase AC with a phase sequence of CBA and phase differences of 45 degrees between phases.
c) Plot each scenario's waveforms on separate graphs, clearly labeling the phases and including appropriate axis labels and titles.

Problem 3: Analysis and Discussion

a) Analyze and compare the waveforms generated in Scenario 1, Scenario 2, and Scenario 3. Explain how the phase differences and phase sequences affect the shape and behavior of the three-phase AC waveforms.
b) Discuss the practical applications of different phase sequences in electrical systems. Provide examples of where each phase sequence might be used and why.

Submission Instructions:

1. Provide well-commented MATLAB code for generating the three-phase AC waveforms (Problem 2).
2. Write a clear and concise analysis and discussion of the waveforms and phase sequences (Problem 3) in a separate document.
3. Submit both your MATLAB code and the analysis document as your assignment.

7. Three phase measurements

In this code:

- We generate three-phase sinusoidal waveforms with the specified frequency and amplitude. Phase B and Phase C are 120 degrees out of phase with respect to Phase A, representing a balanced three-phase system.
- RMS voltage for each phase is calculated using the '**rms'** function.
- Phase angles between Phase A and other phases (Phase B and Phase C) are calculated using the **'atan2'** function and converted from radians to degrees.
- Total power is calculated for each phase assuming a known impedance, and then the total power is summed to get the total power in the three-phase system.

- You can adjust the **'frequency'**, **'amplitude'**, and **'impedance'** values to match your specific waveform and system parameters. This code will help you determine essential electrical quantities for a three-phase waveform in MATLAB.

Hints

```
% Define the three-phase waveform data
time = 0:0.0001:0.1;  % Time vector (0 to 0.1 seconds with a step of 0.0001 seconds)
frequency = 50;       % Frequency in Hertz
amplitude = 220;      % Amplitude of each phase in volts

% Generate the three-phase sinusoidal waveforms
omega = 2 * pi * frequency;
phase_A = amplitude * sin(omega * time);
phase_B = amplitude * sin(omega * time - 2*pi/3);  % Phase B is 120 degrees ahead
of Phase A
phase_C = amplitude * sin(omega * time + 2*pi/3);  % Phase C is 120 degrees behind
Phase A

% Calculate RMS voltage for each phase
rms_A = rms(phase_A);
rms_B = rms(phase_B);
rms_C = rms(phase_C);

% Calculate phase angles between Phase A and other phases
phase_angle_B = rad2deg(atan2(imag(phase_B), real(phase_B)));
phase_angle_C = rad2deg(atan2(imag(phase_C), real(phase_C)));

% Calculate total power
power_A = rms_A^2 / impedance;  % Assuming a known impedance
power_B = rms_B^2 / impedance;
power_C = rms_C^2 / impedance;
total_power = power_A + power_B + power_C;

% Display the results
fprintf('RMS Voltage (Phase A): %.2f V\n', rms_A);
fprintf('RMS Voltage (Phase B): %.2f V\n', rms_B);
fprintf('RMS Voltage (Phase C): %.2f V\n', rms_C);
fprintf('Phase Angle (Phase B - Phase A): %.2f degrees\n', phase_angle_B);
fprintf('Phase Angle (Phase C - Phase A): %.2f degrees\n', phase_angle_C);
fprintf('Total Power: %.2f Watts\n', total_power);
```

Try:

Problem 1: Mathematical Background

I. Provide a brief explanation of three-phase electrical systems, including the significance of balanced and unbalanced systems. Discuss key electrical quantities such as RMS voltage, phase angles, and power in the context of three-phase waveforms.

II. Modify the barrel shifter by changing the number of data lines to 16 bit wide to support 16 different functionalities

Problem 2: MATLAB Programming

Write MATLAB code to generate and analyze a three-phase waveform with the following specifications:

- Frequency: 60 Hz

- Amplitude: 220 V (for each phase)
- Duration: 0.2 seconds

a. Generate three sinusoidal waveforms (Phase A, Phase B, and Phase C) with the specified frequency and amplitude. Phase B and Phase C should have phase differences of 120 degrees with respect to Phase A.

b. Calculate and display the following electrical quantities:

- RMS voltage for each phase.
- Phase angles between Phase A and Phase B, and between Phase A and Phase C.
- Total power in the three-phase system. Assume a balanced system and a known impedance.

Problem 3: Analysis and Interpretation

a. Analyze and discuss the results obtained in Problem 2. Explain the significance of RMS voltage, phase angles, and total power in a three-phase system.

b. Consider a scenario where one of the phases (e.g., Phase B) becomes unbalanced and has a different amplitude (e.g., 200 V) compared to the other phases. Modify the MATLAB code to simulate this unbalanced scenario and recalculate the electrical quantities. Discuss how the unbalance affects the system.

Submission Instructions:

1. Provide well-commented MATLAB code for generating and analyzing the three-phase waveform (Problem 2).
2. Write a clear and concise analysis and interpretation of the electrical quantities and the impact of unbalance (Problem 3) in a separate document.
3. Submit both your MATLAB code and the analysis document as your assignment.

8. Virtual instruments (vi) using Labview

Virtual instruments (VIs) in LabVIEW (Laboratory Virtual Instrument Engineering Workbench) are a powerful way to create customized software applications for data acquisition, analysis, and instrument control. VIs are graphical representations of an instrument or a measurement system and can be created using LabVIEW's intuitive programming environment. Here are the steps to create a basic virtual instrument (VI) using LabVIEW:

Hints

```
% Launch LabVIEW
```

Start LabVIEW on your computer.

```
% Open a New VI
```

Click on "File" in the menu bar.

Select "New VI" to create a new VI or "Open VI" if you have an existing VI that you want to work on.

```
% Front Panel Design
```

The front panel is where you design the user interface of your VI. You can add controls (inputs) and indicators (outputs) by using the tools in the "Controls Palette" on the left-hand side of the LabVIEW window.

Drag and drop controls (e.g., buttons, knobs, numeric inputs) onto the front panel to specify user inputs.

Drag and drop indicators (e.g., graphs, numeric displays) onto the front panel to display the results or data.

Arrange and customize the front panel elements to create a user-friendly interface for your VI.

```
% Block Diagram Design
```

The block diagram is where you define the functionality of your VI by connecting wires between the controls and indicators and adding programming logic.

Right-click on the block diagram and use the "Functions Palette" on the left to add functions and structures.

Use wires to connect the controls to functions and indicators to display the results.
Write LabVIEW code to perform the desired operations, calculations, or data processing.

```
% Wiring and Programming
```

Wire controls and indicators together to establish data flow between the front panel and the block diagram. To wire, click and drag from a control to an indicator or to a function's input.

Write LabVIEW code on the block diagram using graphical programming. You can use structures like loops and case structures for more complex operations.

```
% Save the VI
```

Click on "File" and select "Save" to save your VI with a meaningful name and location on your computer.

```
% Run the VI
```

Click the "Run" button on the toolbar to execute your VI. Interact with the controls on the front panel and observe the results on the indicators.

```
% Debug and Test
```

LabVIEW provides debugging tools such as probes, highlight execution, and breakpoints to help you identify and resolve issues in your VI.

```
% Documentation and Deployment
```

Consider documenting your VI with comments and descriptions to make it more understandable for others. You can also create a standalone application or share the VI with others by creating an executable file or a LabVIEW run-time engine package.

Try

Problem 1: Understanding the Existing VI

1. Download the provided LabVIEW VI file (e.g., "Assignment_VI.vi") or use an existing VI if available.
2. Open the VI in LabVIEW.
3. Examine the front panel and block diagram of the VI to understand its functionality and user interface.

Problem 2: Editing and Building the Existing VI

1. Edit the existing VI to achieve the following:
    - Modify the user interface by adding at least one additional control (e.g., a numeric input, a switch, a slider).
    - Modify the VI's functionality by adding LabVIEW code to perform a specific operation (e.g., mathematical calculations, data processing).
2. Build and run the edited VI to ensure it works as intended.

Problem 3: Creating a Sub VI

1. Identify a portion of the LabVIEW code in the edited VI that can be modularized into a separate sub VI. This portion should be a self-contained task or operation.
2. Create a new VI (Sub VI) that performs the identified task or operation. This sub VI should have inputs and outputs as needed.
3. Incorporate the Sub VI into the edited VI by replacing the relevant LabVIEW code with a call to the Sub VI.
4. Build and run the edited VI with the Sub VI to verify that the modularization works correctly.

Problem 4: Documentation and Presentation

1. Document your changes and explain the functionality of the edited VI and the purpose of the Sub VI in a clear and concise report.
2. Create a brief presentation (e.g., slides or a video) demonstrating the edited VI, the Sub VI, and the modifications you made. Explain the benefits of modularization using Sub VIs.

Submission Instructions:

1. Submit the edited LabVIEW VI file (with your modifications).
2. Submit the created Sub VI as a separate LabVIEW file.
3. Submit a report (in PDF or Word format) that includes documentation of your changes and explanations.
4. Provide a link or file for your presentation materials.

9. Generation of Common Wave Forms Using Labview

Generating common waveforms using LabVIEW is relatively straightforward with the built-in functions and structures provided by the LabVIEW programming environment. You can create waveforms such as sine waves, square waves, triangular waves, and saw tooth waves using LabVIEW. Below are the steps

and syntax to generate some of these common waveforms:

Hints

Generating a Sine Wave:

1. Place a while loop on the block diagram to continuously generate the sine wave.
2. Inside the loop, use a Numeric Control to adjust the frequency of the sine wave.
3. Use a Sine Waveform function to generate the sine wave.

```
/**   Generating a Sine Wave:*/

Sine Waveform VI -> Amplitude (constant) = 1.0 -> Frequency (control) = [frequency
control value]
```

Generating a Square Wave:

1. Place a while loop on the block diagram.
2. Inside the loop, use a Numeric Control to adjust the frequency of the square wave.
3. Use a Square Waveform function to generate the square wave.

```
/**   Generating a Square Wave: */

Square Waveform VI -> Amplitude (constant) = 1.0 -> Frequency (control) = [frequency
control value]
```

Generating a Triangular Wave:

1. Place a while loop on the block diagram.
2. Inside the loop, use a Numeric Control to adjust the frequency of the triangular wave.
3. Use a Triangular Waveform function to generate the triangular wave.

```
/**   Generating a Triangulate Wave: */

Triangular Waveform VI -> Amplitude (constant) = 1.0 -> Frequency (control) =
[frequency control value]
```

Generating a Saw tooth Wave:

1. Place a while loop on the block diagram.
2. Inside the loop, use a Numeric Control to adjust the frequency of the saw tooth wave.
3. Use a Saw tooth Waveform function to generate the saw tooth wave.

```
        /**   Generating a Saw Tooth Wave: */

Saw tooth Waveform VI -> Amplitude (constant) = 1.0 -> Frequency (control) =
[frequency control value]
```

Try

1 Display of maximum and minimum values of a sinusoidal signal.

2 Display of modulation of two sinusoidal signals.

10. Frequency measurement using Lissajous figures in Lab View

Frequency measurement using Lissajous figures in LabVIEW involves the use of an oscilloscope or waveform display to analyze the Lissajous pattern formed when two sinusoidal signals with different frequencies are plotted against each other. The frequency measurement can be performed by counting the number of intersections or lobes of the Lissajous figure within a known time period. Below is a step-by-step guide on how to implement frequency measurement using Lissajous figures in LabVIEW:

Hints

```
/** Create a LabVIEW VI*/

Open LabVIEW and create a new VI (Virtual Instrument).

/** Generate Lissajous Figures*/
Generate two sinusoidal signals with different frequencies and amplitudes using
appropriate Signal Generation functions. Use separate Frequency controls for each
signal.
Create two separate Waveform Charts or Graphs on the front panel.
Wire the generated signals to the Waveform Charts or Graphs.

/** Set Up the Timebase */
Add a Numeric Control for the timebase (the time interval over which you want to
measure frequency).
Connect the Numeric Control to a Wait (ms) function to introduce a delay in your
loop.

/** Count Lissajous Lobe Intersections */

Place a While Loop on the block diagram.
Inside the loop, increment a counter (initialized to zero) every time you detect an
intersection or lobe crossing of the Lissajous figure.
Compare the elapsed time (using the Tick Count function) with the desired timebase,
and exit the loop when the desired time has passed.

/** Calculate Frequency */

Outside the loop, calculate the frequency using the following formula:
Frequency (Hz) = Number of Intersections / Timebase (seconds)

/** Display the Frequency */

Create a Numeric Indicator to display the calculated frequency.
Connect the calculated frequency to the Numeric Indicator.

/** Run the VI */

Run the VI and observe the Lissajous figure on the waveform charts.
The calculated frequency will be displayed on the Numeric Indicator when the VI
completes.
```

Try:

1. Display of Lissajious pattern for sinusoidal voltages of same frequencies.
2. Display of Lissajious pattern for sinusoidal voltages of same different frequencies.

11. Structures Using Labview

Analyze the virtual instrumentation (VI) using control loops, arrays, charts and graphs in LabVIEW software

Hints

```
/** Open LabVIEW and create a new VI */

/** Place the following controls on the front panel */
   - Numeric Control for Frequency (Hz)
   - Numeric Control for Amplitude (V)
   - Numeric Control for Duration (seconds)
   - Waveform Chart (for displaying the original signal)
   - Waveform Graph (for displaying the FFT result)

/** Create a FOR loop on the block diagram with the following elements inside it */
   - Numeric Constant (0) wired to the FOR loop iteration terminal
   - Numeric Control for Duration wired to the count terminal of the FOR loop

/** Inside the FOR loop, generate a sinusoidal signal using the Sine Waveform function */
   - Place a Sine Waveform function inside the FOR loop
   - Wire the Frequency control to the "Frequency (Hz)" input of the Sine Waveform function
   - Wire the Amplitude control to the "Amplitude (V)" input of the Sine Waveform function
   - Use the FOR loop iteration terminal to generate a time signal (0, 1 sample period, 2 sample periods, etc.)
   - Wire the time signal to the "Time (s)" input of the Sine Waveform function

/** Wire the output of the Sine Waveform function to a Build Array function to create an array of the generated signal */

/** After the FOR loop, add a Peak Detector VI to detect peaks in the signal */
   - Place a Peak Detector VI on the block diagram
   - Wire the output array to the "Input Array" input of the Peak Detector
   - Set the "Threshold" parameter of the Peak Detector VI to a suitable value

/** Connect the output of the Peak Detector VI to the "Data" input of the Waveform Chart to display the original signal with peaks highlighted */

/** Place a Waveform Chart Clear function outside the FOR loop and wire it to the chart to clear the chart before displaying the new data */

/** Create a WHILE loop with the following elements inside it */
   - A Waveform Graph Clear function to clear the graph for displaying FFT results
   - An FFT function to compute the Fast Fourier Transform of the signal:
     - Wire the output array (from the Peak Detector) to the "Time Domain Signal" input of the FFT function
   - A waveform chart to display the FFT result:
     - Wire the output of the FFT function to the "Amplitude Spectrum" input of the waveform chart

/** Wire the output of the Waveform Chart Clear function inside the WHILE loop to clear the chart before displaying the new data */

/** Run the VI to generate the sinusoidal signal, detect peaks, compute the FFT, and display the results on the charts */
```

```
/** Analyze the displayed results, including peak positions and frequency components
in the FFT result */

This LabVIEW example demonstrates the use of FOR and WHILE loops, charts (waveform
chart and waveform graph), arrays, and analysis VIs (Peak Detector and FFT) in a
signal processing application. Students can modify and extend this example to
explore more complex signal processing and analysis tasks.
```

Try:

1. Obtain VI using For loop and While loop.
2. Obtain VI using charts and arrays.
3. Obtain VI using graphs.

12. Simulation of low pass and high pass filters using digital simulation

To plot the characteristics of low-pass and high-pass filters using MATLAB, you can follow these steps and use the **'freqz'** function for plotting the frequency response:

Hints

```
/**   Design the Filters */

Design low-pass and high-pass filters using the 'fir1' or 'fdesign' functions.
Here's an example using 'fir1':

% Design a low-pass filter
lowpass_order = 50; % Filter order
lowpass_cutoff_frequency = 0.2; % Cutoff frequency (normalized)
lowpass_filter = fir1(lowpass_order, lowpass_cutoff_frequency);

% Design a high-pass filter
highpass_order = 50; % Filter order
highpass_cutoff_frequency = 0.2; % Cutoff frequency (normalized)
highpass_filter = fir1(highpass_order, highpass_cutoff_frequency, 'high');

/** Plot the Frequency Response */

Plot the frequency response of the filters using the 'freqz' function:

% Frequency response of the low-pass filter
figure;
freqz (lowpass_filter, 1, 1024);
title ('Low-Pass Filter Frequency Response');
xlabel ('Normalized Frequency (\pi radians/sample)');
ylabel ('Magnitude (dB)');

% Frequency response of the high-pass filter
figure;
freqz (highpass_filter, 1, 1024);
title ('High-Pass Filter Frequency Response');
xlabel ('Normalized Frequency (\pi radians/sample)');
ylabel ('Magnitude (dB)');
```

In this code:
‘freqz (lowpass_filter, 1, 1024)’ calculates and plots the frequency response of the low-pass filter.
‘freqz (highpass_filter, 1, 1024)’ calculates and plots the frequency response of the high-pass filter.

```
/** Display the Phase Response (Optional) */
```

If you also want to display the phase response, you can modify the **‘freqz’** function as follows:

```
% Phase response of the low-pass filter
figure;
[h, w] = freqz (lowpass_filter, 1, 1024);
plot (w, unwrap(angle(h)));
title ('Low-Pass Filter Phase Response');
xlabel ('Normalized Frequency (\pi radians/sample)');
ylabel ('Phase (radians)');

% Phase response of the high-pass filter
figure;
[h, w] = freqz (highpass_filter, 1, 1024);
plot (w, unwrap(angle(h)));
title ('High-Pass Filter Phase Response');
xlabel ('Normalized Frequency (\pi radians/sample)');
ylabel ('Phase (radians)');
```

Try

Problem 1: Filter Design

1. Design a low-pass filter with the following specifications:
    - Filter Order: 40
    - Cutoff Frequency: 0.2 (normalized frequency)
2. Design a high-pass filter with the following specifications:
    - Filter Order: 30
    - Cutoff Frequency: 0.3 (normalized frequency)

Problem 2: Frequency Response Plot

1. Plot the magnitude frequency response (in dB) of both the low-pass and high-pass filters.
    - Use the **‘freqz’** function to calculate and plot the frequency response.
    - Label the x-axis as "Normalized Frequency (\pi radians/sample)" and the y-axis as "Magnitude (dB)."
2. Plot the phase frequency response (in radians) of both filters.
    - Use the freqz function to calculate and plot the phase response.
    - Label the x-axis as "Normalized Frequency (\pi radians/sample)" and the y-axis as "Phase (radians)."

Problem 3: Filtered Signal

1. Generate a noisy input signal using MATLAB. You can use the **‘randn’** function to generate random noise and the **‘sin’** function to generate a sinusoidal signal.
2. Apply both the low-pass and high-pass filters to the noisy signal to obtain filtered signals.

3. Plot the noisy input signal, the low-pass filtered signal, and the high-pass filtered signal on the same graph.

Problem 4: Frequency Response Analysis

1. Analyze the frequency response plots and explain how the magnitude and phase characteristics of the filters align with their designed specifications
2. Describe the differences in filtering results between the low-pass and high-pass filters. Discuss which frequencies are attenuated and which are passed through.

Problem 5: Submission

1. Prepare a report summarizing your filter design, frequency response analysis, and filtered signal results.
2. Include the MATLAB code used for filter design, frequency response plotting, and signal processing in your report.
3. Provide explanations and interpretations of the results in the report.

13. Sensor Circuit Using LAB View

Designing the electric and electronic circuit of a sensor in LabVIEW involves creating a virtual representation of the circuit using LabVIEW's graphical programming interface. You'll use various components, such as controls, indicators, and wiring, to simulate the behavior of the physical sensor circuit. Below is a simplified example of how you can design a virtual sensor circuit in LabVIEW.

HINTS

```
/** Open LabVIEW and Create a New VI */

        Launch LabVIEW and create a new VI (Virtual Instrument).

/** Design the Front Panel */

        Add controls and indicators to the front panel to represent components in
your sensor circuit. For example:

Numeric Controls for resistors, capacitors, or other components with adjustable
values.
Switches or buttons to simulate the opening or closing of switches.
Numeric Indicators or Waveform Charts to display sensor readings.
You can find these controls and indicators in the "Controls" and "Indicators"
palettes.

/** Create the Block Diagram */

Switch to the block diagram by clicking on the "Block Diagram" tab.

/** Wire Components Together */

Use wiring to connect the controls and indicators to simulate connections in your
sensor circuit.
For example, use wires to connect a voltage source (control) to a resistor
(indicator).

/** Implement Logic */

Use LabVIEW's graphical programming to implement the logic and calculations needed
```

```
for your sensor circuit. You can use functions and structures like loops, case structures, and math functions.

/** Simulate Sensor Behavior */

Implement code that simulates the behavior of the sensor. For instance, if you're simulating a temperature sensor, write code to generate temperature readings based on user inputs or predefined conditions.

/** Display Sensor Output */

Use indicators (such as numeric indicators or waveform charts) to display the simulated sensor readings on the front panel.

/** Run the VI */

Save and run your VI to see the simulated behavior of the sensor circuit.

Adjust control values on the front panel to see how they affect the sensor output.
/** Analyze and Validate */

Analyze the behavior of your virtual sensor circuit to ensure it behaves as expected.

Validate the circuit's response to different inputs and conditions.

/** Document Your Design*/

Document your virtual sensor circuit design, including circuit diagrams, code explanations, and any assumptions you've made.

/** Extend and Customize */

Depending on your needs, you can extend the virtual sensor circuit to include more components, sensors, or complex behavior.
```

Try

Problem 1: Front Panel Design

1. Open LabVIEW and create a new VI.
2. On the front panel, design the user interface to represent an electric and electronic circuit of a sensor. Include the following elements:
    - Numeric Controls for adjusting resistor values (e.g., R1, R2).
    - Toggle Switches to simulate the opening and closing of switches.
    - Numeric Indicators to display voltage, current, or other relevant values.
    - A Waveform Chart to display dynamic sensor data.
    - Any additional elements you consider relevant to your sensor circuit.

Problem 2: Block Diagram Implementation

1. Switch to the block diagram and use LabVIEW's graphical programming to simulate the behavior of the sensor circuit. Implement the following:

- Wiring to connect the components as per the designed circuit.
- Logic for calculating voltages, currents, or sensor readings based on component values and switch positions.
- Simulation of time-dependent behavior if applicable (e.g., transient response).

Problem 3: Simulate Sensor Operation

1. Create a scenario where the sensor circuit interacts with an external environment or input signal. For example, if simulating a temperature sensor, simulate temperature changes over time.
2. Implement code that calculates and updates sensor data based on the interaction with the environment or input signal.

Problem 4: Real-Time Monitoring

1. Configure LabVIEW to provide real-time monitoring of sensor data.
    - Use a loop structure to continuously update sensor readings.
    - Display the sensor data in real-time on the Waveform Chart on the front panel.

Problem 5: Data Analysis and Validation

1. Run the VI and observe the behavior of the virtual sensor circuit.
    - Verify that the circuit responds correctly to changes in component values and switch positions.
    - Analyze the data displayed on the Waveform Chart and ensure it matches the expected behavior.

Problem 6: Report and Documentation

1. Prepare a report that includes the following:
    - A description of the designed sensor circuit.
    - A discussion of how LabVIEW was used to simulate its behavior.
    - Results and observations from running the VI.
    - An analysis of the data collected during the simulation.
2. Include screenshots of the LabVIEW front panel and block diagram in your report.

Problem 7: Presentation

Prepare a brief presentation to demonstrate your virtual sensor circuit and discuss your findings.

14. Sensor Circuit Using LAB View

Measuring the speed of a machine using a proximity sensor in LabVIEW typically involves using a sensor to detect the position of a rotating object and then calculating the speed based on the change in position over time.

HINTS

```
/** Hardware Setup */

Connect your proximity sensor to the appropriate data acquisition hardware (e.g., a DAQ card) and ensure that it is properly configured.

/** Open LabVIEW and Create a New VI */
```

Launch LabVIEW and create a new VI (Virtual Instrument).

/** Front Panel Design */

On the front panel, add the necessary controls and indicators:
        Numeric Indicator for displaying the speed.
        A Start Button to initiate speed measurement.
        A Graph or Chart for visualizing speed data (optional).
        Any other controls or indicators you may need.

/** **Create the Block Diagram** */

Switch to the block diagram by clicking on the "Block Diagram" tab.

/** Initialize Variables */

Use a Numeric control to set the initial position of the machine.

/** **While Loop for Measurement** */

Place a While Loop on the block diagram to continuously measure and update the speed.

Use a Case Structure with the Stop Button to control the loop's execution.

/** Read Proximity Sensor Data */

Inside the While Loop, use DAQ functions or appropriate libraries to read data from the proximity sensor.

Store the sensor's position data in a variable.

/** **Calculate Speed** */

Calculate the speed of the machine by determining the change in position over time (velocity).

You can use LabVIEW's Math functions to perform this calculation.

/** Display Speed */

Display the calculated speed on the Numeric Indicator on the front panel.

/** **Optional: Data Visualization** */

If desired, you can also plot the speed data on a Graph or Chart in real-time to visualize the machine's speed profile.

/** Start and Stop Controls */

Use the Start Button to initiate speed measurement by enabling the While Loop.

Use the Stop Button to stop the measurement by disabling the While Loop.

```
/** Run the VI */

Save and run your VI to measure and display the speed of the machine.
```

This is a simplified example, and in practice, you may need to consider additional factors such as calibration, noise filtering, and error handling. The specific implementation details can vary depending on the type of proximity sensor and data acquisition hardware you are using.

Try

Problem 1: Hardware Setup

Set up a proximity sensor and connect it to a DAQ (Data Acquisition) device or hardware.

Problem 2: LabVIEW VI Design

1. Open LabVIEW and create a new VI.
2. Design the front panel with the following elements:
    - A numeric indicator to display the measured speed (in RPM or other appropriate units).
    - A start button to initiate speed measurement.
    - A stop button to stop the measurement.
    - A chart or graph indicator to display the speed profile (optional but recommended).
    - Any additional controls or indicators you consider necessary.

Problem 3: Block Diagram Implementation

1. Create a While Loop on the block diagram.
2. Use a Case Structure inside the loop to handle the start and stop buttons. The loop should only execute when the start button is pressed and stop when the stop button is pressed.
3. Use DAQ functions or libraries (appropriate for your hardware) to read data from the proximity sensor. You may need to configure the DAQ task or device settings.
4. Calculate the speed of the machine based on the sensor data. Consider the appropriate formula for converting sensor data into speed (e.g., RPM calculation).
5. Display the calculated speed on the numeric indicator on the front panel.
6. If desired, plot the speed data on the chart or graph indicator for real-time visualization.

Problem 4: User Interface

1. Test the VI by running it.
    - Click the start button to initiate speed measurement.
    - Observe the speed reading on the numeric indicator.
    - Use the stop button to halt the measurement.

Problem 5: Data Analysis

1. Analyze the data collected during the measurement.
    - Calculate the average speed.
    - Identify any variations or anomalies in the speed profile.

Problem 6: Report

1. Prepare a report summarizing your LabVIEW program's design and functionality.
    - Include screenshots of the LabVIEW front panel and block diagram.
    - Describe the hardware setup and connection.

Problem 7: Presentation

1. Create a brief presentation to demonstrate your LabVIEW program and discuss your findings.
    - Discuss how the program measures machine speed.

V. TEXT BOOKS:

1 A Chakrabarthy, "Circuit Theory", Dhanpat Rai Publications, 6th edition, 2006.

2 A Sudhakar, Shyammohan S Palli, "Circuits and Networks", Tata McGraw Hill, 4th edition, 2010.

VI. REFERENCES

1. William Hayt, Jack E Kemmerly S.M. Durbin, "Engineering Circuit Analysis", Tata McGraw Hill, 7th edition, 2010.
2. K S Suresh Kumar, "Electric Circuit Analysis", Pearson Education, 1st edition, 2013.
3. Rudrapratap, "Getting started with MATLAB: A Quick Introduction for Scientists and Engineers", OxfordUniversity Press, 1st edition, 19994.