R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

(Affiliated to DBATU, Lonere, and M. S.)

Laboratory Manual

Department of Electrical Engineering

Numerical Methods and Computer Programming
(PCEE4060L)
S.Y. B. Tech (Electrical)
[SEM-IV]

Shahada Road, Near Nimzari Naka, Shirpur, Maharashtra 425405 Ph: 02563 259802,
Web: www.rcpit.ac.in

The Shirpur Education Society's

R. C. Patel Institute of Technology, Shirpur

CERTIFICATE

This is to certify that Mr / Miss ____of ***Second Year Electrical Engineering*** *branch, Roll No. _______has performed practical work satisfactorily in the Subject* ***Numerical Methods and Computer Programming****, in the premises of the Department of Electrical Engineering during the academic year 20   - 20*

Date: /   / 20

Place: Shirpur

Signature of the Teacher   *Head of Depart*

Table of Contents

Part-A (Python Programming)

Part-B (MATLAB Programming)

Part-A

Laboratory Report

Experiment No.1

Batch Code: ___________________

Name of Student: -- Roll No--------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] ------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] ------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment 1

Objective: Write a program for finding roots of f(x) by Gauss Elimination Method using Python

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

Theory: Gauss elimination method is used to solve a system of linear equations. Let's recall the definition of these systems of equations. A system of linear equations is a group of linear equations with various unknown factors. As we know, unknown factors exist in multiple equations. Solving a system involves finding the value for the unknown factors to verify all the equations that make up the system. If there is a single solution that means one value for each unknown factor, then we can say that the given system is a consistent independent system. If multiple solutions exist, the system has infinitely many solutions; then we say that it is a consistent dependent system. If there is no solution for unknown factors, and this will happen if there are two or more equations that can't be verified simultaneously, then we say that it's an inconsistent system.

Python Programme

```
# Importing NumPy Library
import numpy as np
import sys

# Reading number of unknowns
n = int(input('Enter number of unknowns: '))

# Making numpy array of n x n+1 size and initializing
# to zero for storing augmented matrix
a = np.zeros((n,n+1))

# Making numpy array of n size and initializing
# to zero for storing solution vector
```

```
x = np.zeros(n)

# Reading augmented matrix coefficients
print('Enter Augmented Matrix Coefficients:')
for i in range(n):
    for j in range(n+1):
        a[i][j] = float(input( 'a['+str(i)+']['+ str(j)+']='))

# Applying Gauss Elimination
for i in range(n):
    if a[i][i] == 0.0:
        sys.exit('Divide by zero detected!')

    for j in range(i+1, n):
        ratio = a[j][i]/a[i][i]

        for k in range(n+1):
            a[j][k] = a[j][k] - ratio * a[i][k]

# Back Substitution
x[n-1] = a[n-1][n]/a[n-1][n-1]

for i in range(n-2,-1,-1):
    x[i] = a[i][n]

    for j in range(i+1,n):
        x[i] = x[i] - a[i][j]*x[j]

    x[i] = x[i]/a[i][i]

# Displaying solution
print('\nRequired solution is: ')
for i in range(n):
    print('X%d = %0.2f' %(i,x[i]), end = '\t')
```

Output of Program:

Conclusion:-

Lab Assignment:-

1. Solve the following equation by Gauss Elimination method;

$$2x + 4y - 6z = -4;\ x + 5y + 3z = 10;\ x + 3y + 2z = 5$$

R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

Laboratory Report

Experiment No.2

Batch Code: ____________________

Name of Student: --- Roll No-------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] -------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] -------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment 2

Objective: Write a program for finding roots of f(x) by Bisection Method using Python

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

Theory: In Mathematics, the bisection method is a straightforward technique to find numerical solutions of an equation with one unknown. Among all the numerical methods, the bisection method is the simplest one to solve the transcendental equation. The bisection method is used to find the roots of a polynomial equation. It separates the interval and subdivides the interval in which the root of the equation lies. The principle behind this method is the intermediate theorem for continuous functions. It works by narrowing the gap between the positive and negative intervals until it closes in on the correct answer. This method narrows the gap by taking the average of the positive and negative intervals. It is a simple method and it is relatively slow. The bisection method is also known as interval halving method, root-finding method, binary search method or dichotomy method.

Python Program:

```
# Defining Function
def f(x):
    return x**3-5*x-9

# Implementing Bisection Method
def bisection(x0,x1,e):
    step = 1
    print('\n\n*** BISECTION METHOD IMPLEMENTATION ***')
    condition = True
    while condition:
        x2 = (x0 + x1)/2
        print('Iteration-%d, x2 = %0.6f and f(x2) = %0.6f' %
(step, x2, f(x2)))
```

```
        if f(x0) * f(x2) < 0:
            x1 = x2
        else:
            x0 = x2

        step = step + 1
        condition = abs(f(x2)) > e

    print('\nRequired Root is : %0.8f' % x2)

# Input Section
x0 = input('First Guess: ')
x1 = input('Second Guess: ')
e = input('Tolerable Error: ')

# Converting input to float
x0 = float(x0)
x1 = float(x1)
e = float(e)

#Note: You can combine above two section like this
# x0 = float(input('First Guess: '))
# x1 = float(input('Second Guess: '))
# e = float(input('Tolerable Error: '))

# Checking Correctness of initial guess values and bisecting
if f(x0) * f(x1) > 0.0:
    print('Given guess values do not bracket the root.')
    print('Try Again with different guess values.')
else:
    bisection(x0,x1,e)
```

Output of Program:

Conclusion:-

Lab Assignment:

1. Determine the positive root of $x - \cos x = 0$ by bisection method

R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

Laboratory Report

Experiment No.3

Batch Code: ____________________

Name of Student: -- Roll No-------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] ------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] ------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment - 3

Objective: Write a program for finding roots of f(x) by False Position Method using Python

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

Theory: Regula Falsi method or the method of false position is a numerical method for solving an equation in one unknown. It is quite similar to bisection method algorithm and is one of the oldest approaches. It was developed because the bisection method converges at a fairly slow speed. In simple terms, the method is the trial and error technique of using test ("false") values for the variable and then adjusting the test value according to the outcome.

Python Program:

```
# Defining Function
def f(x):
    return x**3-5*x-9

# Implementing False Position Method
def falsePosition(x0,x1,e):
    step = 1
    print('\n\n*** FALSE POSITION METHOD IMPLEMENTATION ***')
    condition = True
    while condition:
        x2 = x0 - (x1-x0) * f(x0)/( f(x1) - f(x0) )
        print('Iteration-%d, x2 = %0.6f and f(x2) = %0.6f' %
(step, x2, f(x2)))

        if f(x0) * f(x2) < 0:
            x1 = x2
        else:
            x0 = x2
```

```
        step = step + 1
        condition = abs(f(x2)) > e

    print('\nRequired root is: %0.8f' % x2)

# Input Section
x0 = input('First Guess: ')
x1 = input('Second Guess: ')
e = input('Tolerable Error: ')

# Converting input to float
x0 = float(x0)
x1 = float(x1)
e = float(e)

#Note: You can combine above two section like this
# x0 = float(input('First Guess: '))
# x1 = float(input('Second Guess: '))
# e = float(input('Tolerable Error: '))

# Checking Correctness of initial guess values and false
positioning
if f(x0) * f(x1) > 0.0:
    print('Given guess values do not bracket the root.')
    print('Try Again with different guess values.')
else:

    falsePosition(x0,x1,e)
```

Output of Program:

Conclusion:-

Lab Assignment:

1. Solve the positive root of $x^3 = 2x + 5$ by False Position Method.

Laboratory Report

Experiment No.4

Batch Code: ____________________

Name of Student: -- Roll No-------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] ------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] ------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment – 4

Objective: Write a program for finding roots of f(x) by Secant Method using Python

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

Theory: Secant method is also a recursive method for finding the root for the polynomials by successive approximation. It's similar to the Regular-falsi method but here we don't need to check f(x1)f(x2)<0 again and again after every approximation. In this method, the neighbourhoods roots are approximated by secant line or chord to the function f(x). It's also advantageous of this method that we don't need to differentiate the given function f(x), as we do in Newton-raphson method.

Python Program:

```
# Defining Function
def f(x):
    return x**3 - 5*x - 9

# Implementing Secant Method

def secant(x0,x1,e,N):
    print('\n\n*** SECANT METHOD IMPLEMENTATION ***')
    step = 1
    condition = True
    while condition:
        if f(x0) == f(x1):
            print('Divide by zero error!')
            break

        x2 = x0 - (x1-x0)*f(x0)/( f(x1) - f(x0) )
        print('Iteration-%d, x2 = %0.6f and f(x2) = %0.6f' %
(step, x2, f(x2)))
```

```
        x0 = x1
        x1 = x2
        step = step + 1

        if step > N:
            print('Not Convergent!')
            break

        condition = abs(f(x2)) > e
    print('\n Required root is: %0.8f' % x2)

# Input Section
x0 = input('Enter First Guess: ')
x1 = input('Enter Second Guess: ')
e = input('Tolerable Error: ')
N = input('Maximum Step: ')

# Converting x0 and e to float
x0 = float(x0)
x1 = float(x1)
e = float(e)
# Converting N to integer
N = int(N)

#Note: You can combine above three section like this
# x0 = float(input('Enter First Guess: '))
# x1 = float(input('Enter Second Guess: '))
# e = float(input('Tolerable Error: '))
# N = int(input('Maximum Step: '))

# Starting Secant Method
secant(x0,x1,e,N)
```

Output of Program:

<u>Conclusion</u>:-

Lab Assignment:

1. Write the advantages and disadvantages of Secant method?
2. A real root of the equation f(x) = x3 – 5x + 1 = 0 lies in the interval (0, 1). Perform four iterations of the secant method.

Laboratory Report

Experiment No.5

Batch Code: ____________________

Name of Student: -- Roll No--------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] -------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] -------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment – 5

Objective: Write a program for to generate forward difference table using Python

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

Theory:

Interpolation is the technique of estimating the value of a function for any intermediate value of the independent variable, while the process of computing the value of the function outside the given range is called extrapolation.

Forward Differences: The differences y1 – y0, y2 – y1, y3 – y2, ……, yn – yn–1 when denoted by dy0, dy1, dy2, ……, dyn–1 are respectively, called the first forward differences. Thus, the first forward differences are :

Forward difference table

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0					
		Δy_0				
x_1	y_1		$\Delta^2 y_0$			
$(= x_0 + h)$		Δy_1		$\Delta^3 y_0$		
x_2	y_2		$\Delta^2 y_1$		$\Delta^4 y_0$	
$(= x_0 + 2h)$		Δy_2		$\Delta^3 y_1$		$\Delta^5 y_0$
x_3	y_3		$\Delta^2 y_2$		$\Delta^4 y_1$	
$= (x_0 + 3h)$		Δy_3		$\Delta^3 y_2$		
x_4	y_4		$\Delta^2 y_3$			
$= (x_0 + 4h)$		Δy_4				
x_5	y_5					
$= (x_0 + 5h)$						

Python Program:

```
# Reading number of unknowns
n = int(input('Enter number of data points: '))

# Making numpy array of n & n x n size and initializing
# to zero for storing x and y value along with differences of y
x = np.zeros((n))
y = np.zeros((n,n))


# Reading data points
print('Enter data for x and y: ')
for i in range(n):
    x[i] = float(input( 'x['+str(i)+']='))
    y[i][0] = float(input( 'y['+str(i)+']='))

# Generating forward difference table
for i in range(1,n):
    for j in range(0,n-i):
        y[j][i] = y[j+1][i-1] - y[j][i-1]


print('\nFORWARD DIFFERENCE TABLE\n');

for i in range(0,n):
    print('%0.2f' %(x[i]), end='')
    for j in range(0, n-i):
        print('\t\t%0.2f' %(y[i][j]), end='')
    print()
```

Output of Program:

Conclusion:-

Lab Assignment:

1. Find Solution using Newton's Forward Difference formula

x	1891	1901	1911	1921	1931
y	46	66	81	93	101

Part-B

SHIRPUR EDUCATION SOCIETY

R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

Laboratory Report

Experiment No.1

Batch Code: ____________________

Name of Student: --- Roll No--------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] -------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] -------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment – 1

Objective: Write a program for finding roots of f(x) by Crout's Method using MATLAB

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

MATLAB Program

```
A = [1 3 4 8
    2 1 2 3
    4 3 5 8
    9 2 7 4];

B = [1
    1
    1
    1];

matrixSize=length(A);
Lower=zeros(size(A));
Upper=zeros(size(A));

Lower(:,1)= A(:,1); %Set the first column of L to the frist column of A
Upper(1,:)=A(1,:)/Lower(1,1); % Create the first row of upper, divide by
L(1,1)

Upper(1,1)=1;   % Start the identity matrix

for k=2:matrixSize
for j=2:matrixSize
    for i=j:matrixSize
        Lower(i,j)=A(i,j) - dot(Lower(i,1:j-1),Upper(1:j-1,j));
    end
    Upper(k,j)=(A(k,j)-dot(Lower(k,1:k-1),Upper(1:k-1,j)))/Lower(k,k);
end
end

Upper
Lower

% L * Y = B
Y = zeros(matrixSize, 1);
% BASE CASE, SOLVE THE FIRST ONE
Y(1) = B(1);
for row = 2 : matrixSize   %2 - number or rows
    Y(row) = B(row);
    for col = 1 : row - 1      %1 - row number
        Y(row) = Y(row) - Lower(row, col) * Y(col);
```

```
    end
    Y(row) = Y(row) / Lower(row, row)
end
Y


% U * X = Y
X = zeros(matrixSize, 1);

X(matrixSize) = Y(matrixSize) / Upper(matrixSize,matrixSize);

for row = matrixSize - 1 : -1 : 1   %second to last row - 1
    temp = 0;
    for col = matrixSize : -1 : 1      %number of rows to row
        temp  = temp + Upper(row,col) * X(col);
    end
    X(row) = (Y(row) - temp) / Upper(row,row);
end

X
```

Output of Program:

```
>> LU_Crout

Upper =

    1.0000    3.0000    4.0000    8.0000

         0    1.0000    1.2000    2.6000

         0         0    1.0000    3.0000

         0         0         0    1.0000

Lower =

    1.0000         0         0         0

    2.0000   -5.0000         0         0

    4.0000   -9.0000   -0.2000         0

    9.0000  -25.0000    1.0000   -6.0000


Y =

    1.0000

    0.2000

         0

         0

Y =

    1.0000

    0.2000
```

```
    6.0000
         0
Y =
    1.0000
    0.2000
    6.0000
    1.5000
Y =
    1.0000
    0.2000
    6.0000
    1.5000
X =
   -0.5000
   -5.5000
    1.5000
    1.5000
```

Conclusion:-

Lab Assignment:

1. Solve the following set of equations by Crout's Method

$$2x + 4y + 4z = 12;\ 8x - 3y + 2z = 20;\ 4x + 11y - z = 33$$

R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

Laboratory Report

Experiment No.2

Batch Code: ___________________

Name of Student: -- Roll No--------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] -------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] -------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment – 2

Objective: Write a program for finding roots of f(x) by Gauss Siedel Method using MATLAB

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

Theory:

The **Gauss Seidel** method is an iterative process to solve a square system of (multiple) linear equations. It is also prominently known as **'Liebmann'** method. In any iterative method in numerical analysis, every solution attempt is started with an approximate solution of an equation and iteration is performed until the desired accuracy is obtained. In Gauss-Seidel method, the most recent values are used in successive iterations. The Gauss-Seidel Method allows the user to control round-off error. The Gauss Seidel method is very similar to Jacobi method and is called as the **method of successive displacement**.

MATLAB Program

```
% Gauss-Seidel method

 n=input('Enter number of equations, n:  ');
 A = zeros(n,n+1);
 x1 = zeros(n);
 tol = input('Enter the tolerance, tol: ');
 m = input('Enter maximum number of iterations, m:  ');
A=[4 2 3 8; 3 -5 2 -14; -2 3 8 27];
 x1=[0 0 0];

 k = 1;
 while  k <= m
    err = 0;
    for i = 1 : n
       s = 0;
       for j = 1 : n
          s = s-A(i,j)*x1(j);
       end
       s = (s+A(i,n+1))/A(i,i);
       if abs(s) > err
           err  = abs(s);
       end
       x1(i) = x1(i) + s;
    end

    if err <= tol
```

```
      break;
    else
      k = k+1;
    end
end
fprintf('Solution vector after %d iterations is :\n', k-1);
for i = 1 : n
  fprintf(' %11.8f \n', x1(i));
end
```

Output of Program:

Conclusion:

Lab Assignment:

1. Using the Gauss Siedal Method solve the system of equations correct to three decimal places.

$$x + 2y + z = 0 \; ; \; 3x + y - z = 0 \; ; \; x - y + 4z = 3$$

R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

Laboratory Report

Experiment No.3

Batch Code: ___________________

Name of Student: --- Roll No-------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] -------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] ------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment – 3

Objective: Write a program for solving numerical integration by trapezoidal rule using MATLAB

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

Theory:

Trapezoidal rule is a numerical tool for the solving of definite integral. This rule based on computing the area of trapezium. Trapezoidal rule is applicable for all number of interval whether n is even or odd. The large number of interval give the best result compare than small number of interval.

MATLAB Program

```
% Numerical Analysis Trapezoidal Rule using MATLAB
clear all;
close all;
clc;

f=inline('1/(1+x^2)');

a=input('Enter lower limit of integral=');
b=input('Enter upper limit of integral=');
n=input('Enter number of intervals=');

h=(b-a)/n;

sum=0.0;

for i=1:n-1
    x=a+i*h;
    sum=sum+f(x);
end

trap=h*(f(a)+2*sum+f(b))/2.0;
fprintf('Evaluated Integral =%f',trap);
```

Output of Program:

Conclusion:

Lab Assignment:

1. Evaluate the integral $\int_0^{1.2} e^x \, dx$, taking six intervals by using trapezoidal rule up to three significant figures.

2. Evaluate $\int_0^{12} \frac{dx}{1+x^2}$, by using trapezoidal rule, taking n=6, correct to give significant figures.

R. C. PATEL INSTITUTE OF TECHNOLOGY

An Autonomous Institute

Laboratory Report

Experiment No.4

Batch Code: ___________________

Name of Student: -- Roll No--------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] -------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] -------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment – 4

Objective: Write a program for solving numerical integration by Simpson's 1/3 Rule using MATLAB

Pre-lab: Linear Algebra, Matrix Operations and Crammers rule.

Equipment's needed: Computer with python installed.

Theory:

Simpson 1/3 rule is a numerical integration technique which give the better result than trapezoidal rule. It is applicable when the number of interval n is even. The Simpson 1/3 rule reduce the error than trapezoidal rule. The large number of interval give the best result and reduce error compare than small number of interval. This rule is also based on computing the area of trapezium.

MATLAB Program

```
% Numerical Method Simpson 1/3 Rule using MATLAB
clear all;
close all;
clc;

f=inline('1/(1+x^2)');

a=input('Enter lower limit of integral=');
b=input('Enter upper limit of integral=');
n=input('Enter number of intervals (multiple of 2)=');

h=(b-a)/n;

sum1=0.0;
sum2=0.0;
for i=1:2:n-1
    x=a+i*h;
    sum1=sum1+f(x);
end
for i=2:2:n-2
    x=a+i*h;
    sum2=sum2+f(x);
end

simp=h*(f(a)+4*sum1+2*sum2+f(b))/3;

fprintf('Integrated value is %f',simp)
```

Output of Program:

Conclusion:

Lab Assignment:

1. Evaluate the integral $\int_0^{1.2} e^x \, dx$, taking six intervals by using Simpson's 1/3 rule.

2. Evaluate $\int_0^{12} \frac{dx}{1+x^2}$, by using Simpson's 1/3 rule, taking n=6.

SES
SHIRPUR EDUCATION SOCIETY

R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

Laboratory Report

Experiment No.5

Batch Code: ____________________

Name of Student: --- Roll No-------------------------.

Date of Lab: ----------------------- Date of Submission: ---------

Evaluations

1) Lab Attendance [2] -------------------
2) Observations and Conclusion [2] --------------------
3) Oral [1] -------------------

Overall Marks (5) ------------------------

____

Sign of Practical Teacher

Experiment – 5

<u>Objective</u>: Write a MATLAB program for loop analysis of electric circuits

<u>Pre-lab</u>: Linear Algebra, Matrix Operations and Crammers rule.

<u>Equipment's needed</u>: Computer with python installed.

Theory:

1.In this method, we set up and solve a system of equations in which the unknowns are loop currents. The currents in the various branches of the circuit are easily determined from the loop currents.

The steps in the loop current method are:

- Count the number of loop currents required. Call this number m.
- Choose m independent loop currents, call them I1, I2, . . . , Im and draw them on the circuit diagram.
- Write down Kirchhoff's Voltage Law for each loop. The result, after simplification, is a system of n linear equations in the n unknown loop currents in this form:

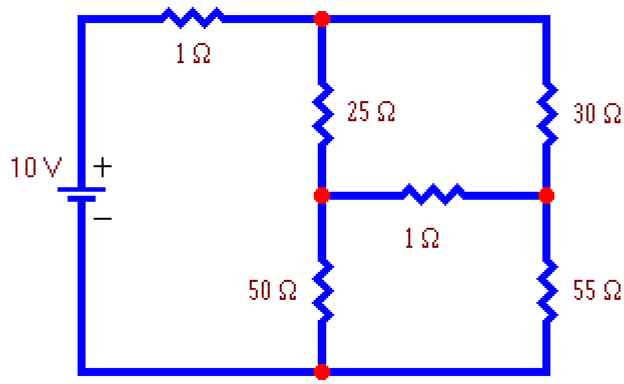
$$\begin{cases} R_{11} \cdot I_1 + R_{12} \cdot I_2 + \cdots + R_{1m} \cdot I_m = V_1 \\ R_{21} \cdot I_1 + R_{22} \cdot I_2 + \cdots + R_{2m} \cdot I_m = V_2 \\ \quad \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots \\ R_{m1} \cdot I_1 + R_{m2} \cdot I_2 + \cdots + R_{mm} \cdot I_m = V_m \end{cases}$$

  where R11, R12, . . . , Rmm and V1, V2, . . . , Vm are constants.

  Alternatively, the system of equations can be gotten (already in simplified form) by using the inspection method.

- Solve the system of equations for the m loop currents I1, I2, . . . , Im using Gaussian elimination or some other method.
- Reconstruct the branch currents from the loop currents.

2. Find the current flowing in each branch of this circuit.



Solution:

1. The number of loop currents required is =

2. We will choose the loop currents shown to the below. In fact these loop currents are mesh currents. **(Draw the circuit with loops showing direction of currents)**

3. Write down Kirchoff's Voltage Law for each loop. The result is the following system of equations: **(Write down THREE simultaneous equations)**

4. Collecting terms this becomes: **(Write down final THREE equations)**

5. Solving the system of equations using Gaussian elimination or some other method gives the following currents, all measured in amperes:

I_1 = **I_2 =** **I_3 =**

6. Reconstructing the branch currents from the loop currents gives the results shown in the picture to the below; **(Redraw the circuit with Answers)**

MATLAB Program:

```
C = [                ]
b= [                 ]'
A = [C b];                  %Augmented Matrix
n= size(A,1);              %number of eqns/variables
x = zeros(n,1);          %variable matrix [x1 x2 ... xn] coulmn
for i=1:n-1
    for j=i+1:n
        m = A(j,i)/A(i,i)
        A(j,:) = A(j,:) - m*A(i,:)
    end
end
x(n) = A(n,n+1)/A(n,n)
for i=n-1:-1:1
    summ = 0
    for j=i+1:n
        summ = summ + A(i,j)*x(j,:)
        x(i,:) = (A(i,n+1) - summ)/A(i,i)
    end
end
```

Program Output:

Conclusion: