MALLA REDDY UNIVERSITY
DEPARTMENT OF MATHEMATICS
MODELLING & CODING ON MATRICES AND DIFFERENTIAL EQUATIONS
(20 MARKS)

Students are instructed to utilize the knowledge they acquired in the course on Matrices and Differential equations, to build mathematical models, and use their understanding of coding.

CODING TO MATHS PROBLEMS (1 X 10 = 10 MARKS)

1. Write Python code to find the Rank, Eigen Values and Eigen vectors of a matrix.

2. Write Python code to solve system of equations using Gauss elimination method.

3. Write Python code to solve the system of equations using LU decomposition method.

4. Write Python code to express the given matrix as a product of lower and upper triangular matrices.

5. Write Python code to find the nature of a quadratic form.

6. Write Python code to solve the second order differential equation with constant coefficients.

MATHEMATICAL MODEL DEVELOPMENT (1 X 10 = 10 MARKS)

1. The perimeter of a triangle is 36 inches. Twice the length of the longest side minus the length of the shortest is 26 inches. The sum of the length of the longest side and twice the sum of the both the other side lengths is 56 inches. Find the sides.

2. The perimeter of a triangle is 30 inches. The shortest side is 4 inches shorter than the longest side. The longest side is 6 inches less than the sum of the other two sides. Find the lengths.

3. A health bar is made from a corn mixture, an egg mixtures and a vegetable mixture. 100 grams of corn mixture contains 25 grams of protein, 30 grams of carbohydrates and 40 grams of fat. The egg mixture contains 40 grams of protein and 20 grams of both carbohydrates and fat. The vegetable mixture contains 20 grams of protein, 10 grams of carbohydrates and 30 grams of fat. How many grams of each mixture should be used to create a health bar that contain 220 grams of protein, 10 grams of carbohydrates and 260 grams of fat.

4. In a T20 match, India need just 6 runs to win with 1 ball left to go in the last over. The last ball was bowled and the batsman at the crease hit it high up. The ball traversed along a path in a vertical plane and the equation of the paths $y = ax^2 + bx + c$ with respect to xy-coordinate system in the vertical plane. And the ball traversed through the points (10,8), (20,16), (40,22). Can you conclude that India won the match? All the distances are measured in meters and the meeting point at the plane of the path with the farthest boundary line is (70,0).

5. A population of insects in a region will grow at a rate that is proportional to their current population. In the absence of any outside factors the population will triple in two weeks time. On any given day there is a net migration into the area of 15 insects and 16 are eaten by the local bird population and 7 die of natural causes. If there are initially 100 insects in the area will the population survive? If not, when do they die out?

6. Construct the population of prey-predator model using the following assumptions.
   a. The prey will grow at a rate that is proportional to its current population if there is no predators
   b. The population of predator will decrease at a rate proportional to its current population if there is no prey.
   c. The number of encounters between predator and prey will be proportional to the product of the populations
   d. Each encounter between the predator and prey will increase the population of the predator and decrease the population of the prey.

ANSWERS

MATHEMATICAL MODEL DEVELOPMENT

1. The perimeter of a triangle is 36 inches. Twice the length of the longest side minus the length of the shortest is 26 inches. The sum of the length of the longest side and twice the sum of the both the other side lengths is 56 inches. Find the sides

Solution: Let x be the longest side.

$$x = y + z = 36$$
$$2x - z = 26$$
$$x + 2y + 2z = 6$$

The solution is $x = 26, y = -7, z = 20$

2. The perimeter of a triangle is 30 inches. The shortest side is 4 inches shorter than the longest side. The longest side is 6 inches less than the sum of the other two sides. Find the lengths.

Solution : Let x be the longest side and z be the shortest side.

$$x + y + z = 30$$
$$x - 4 = z$$
$$x - 6 = y + z$$

The solution is $x = 18, y = -2, z = 14$

3. A health bar is made from a corn mixture, an egg mixtures and a vegetable mixture. 100 grams of corn mixture contains 25 grams of protein, 30 grams of carbohydrates and 40 grams of fat. The egg mixture contains 40 grams of protein and 20 grams of both carbohydrates and fat. The vegetable mixture contains 20 grams of protein, 10 grams of carbohydrates and 30 grams of fat. How many grams of each mixture should be used to create a health bar that contain 220 grams of protein, 10 grams of carbohydrates and 260 grams of fat.

Solution: Let x_1 be the corn mixture, x_2 be the egg mixture and x_3 be the vegetable mixture.

$$25x_1 + 40x_2 + 20x_3 = 220$$
$$30x_1 + 20x_2 + 10x_3 = 180$$
$$40x_1 + 20x_2 + 30x_3 = 260$$

Solution is $x_1 = 4, x_2 = 2, x_3 = 2$

Corn mixture = 4 grams, Egg mixture = 2 grams, Veg mixture = 2 grams

4. In a T20 match, India need just 6 runs to win with 1 ball left to go in the last over. The last ball was bowled and the batsman at the crease hit it high up. The ball traversed along a path in a vertical plane and the equation of the paths $y = ax^2 + bx + c$ with respect to *xy*-coordinate system in the vertical plane. And the ball traversed through the points (10,8), (20,16), (40,22). Can you conclude that India won the match? All the distances are measured in meters and the meeting point at the plane of the path with the farthest boundary line is (70,0).

Solution : The path of the ball is $y = ax^2 + bx + c$ and is traversed through the points A(10,8), B(20,16), C(40,22).

At A, $y = ax^2 + bx + c \Rightarrow 8 = a(100) + b(10) + c \Rightarrow 100a + 10b + c = 8$

At B, $y = ax^2 + bx + c \Rightarrow 16 = a(400) + b(20) + c \Rightarrow 400a + 20b + c = 16$

At C, $y = ax^2 + bx + c \Rightarrow 22 = a(1600) + b(40) + c \Rightarrow 1600a + 40b + c = 22$

Solving we get, a = -1/60=-0.01667, b=1.3 c=-10/3=-3.33333

Hence the equation of path of ball (parabola) is $y = -(1/60)x^2 + 1.3x - (10/3)$

The distance of the boundary is 70, substituting

$$y = -(1/60)4900 + 1.3(70) - (10/3) = 5.99997 = 6$$

So the ball went by six meters high over the boundary line and is not possible for a fielder at the boundary to catch the ball. Hence the ball went for a six and India won the match.

5. A population of insects in a region will grow at a rate that is proportional to their current population. In the absence of any outside factors the population will triple in two weeks time. On any given day there is a net migration into the area of 15 insects and 16 are eaten by the local bird population and 7 die of natural causes. If there are initially 100 insects in the area will the population survive? If not, when do they die out?

Solution: Let *P(t)* represents a population in a given region at any time *t.* Let's start out by looking at the birth rate. We are told that the insects will be born at a rate that is proportional to the current population. This means that the birth rate can be written as rP. Where *r* is a positive constant that will need to be determined. Now, let's take everything into account and get the IVP for this problem.

$$P' = (rP + 15) - (16 + 7), \quad P(0) = 100$$

$$\Rightarrow P' = rP - 8, \qquad P(0) = 100$$

We will use the fact that the population triples in two week time to help us find *r*. In the absence of outside factors the differential equation would become

$$P' = rP, \quad P(0) = 100, \quad P(14) = 300$$

$$\text{Hence,} \qquad P(t) = ce^{rt}$$

Applying the initial condition gives *c* = 100. Now apply the second condition.

$$300 = P(14) = 100e^{14r} \Rightarrow 300 = 100e^{14r}$$

$$3 = e^{14r} \Rightarrow \ln 3 = 14r \Rightarrow r = \frac{\ln 3}{14}$$

Substituting

$$P' - \frac{\ln 3}{14}P = -8, \qquad P(0) = 100$$

This is a fairly simple linear differential equation,

$$IF = e^{\int -\frac{\ln 3}{14}dt} = e^{-\frac{\ln 3}{14}t}$$

$$Pe^{-\frac{\ln 3}{14}t} = \int -8e^{-\frac{\ln 3}{14}t}dt \Rightarrow Pe^{-\frac{\ln 3}{14}t} = -8\left(-\frac{14}{\ln 3}\right)e^{-\frac{\ln 3}{14}t} + c$$

$$\Rightarrow P(t) = \frac{112}{\ln 3} + ce^{\frac{\ln 3}{14}t}$$

Applying the initial condition gives the following.

$$P(t) = \frac{112}{\ln 3} + \left(100 - \frac{112}{\ln 3}\right)e^{\frac{\ln 3}{14}t} = \frac{112}{\ln 3} - 1.94679\ e^{\frac{\ln 3}{14}t}$$

Now, the exponential has a positive exponent and so will go to plus infinity as t increases. Its coefficient, however, is negative and so the whole population will go negative eventually. Clearly, population can't be negative, but in order for the population to go negative it must pass through zero. In other words, eventually all the insects must die. So, they don't survive and we can solve the following to determine when they die out.

$$0 = \frac{112}{\ln 3} - 1.94679\ e^{\frac{\ln 3}{14}t} \Rightarrow t = 50.4415 \text{ days}$$

So, the insects will survive for around 7.2 weeks. Here is a graph of the population during the time in which they survive

6. Construct the population of prey-predator model using the following assumptions.
    a. The prey will grow at a rate that is proportional to its current population if there is no predators
    b. The population of predator will decrease at a rate proportional to its current population if there is no prey.
    c. The number of encounters between predator and prey will be proportional to the product of the populations
    d. Each encounter between the predator and prey will increase the population of the predator and decrease the population of the prey.

Solution : We'll start off by letting x represent the population of the predators and y represent the population of the prey. Now, the first assumption tells us that, in the absence of predators, the prey will grow at a rate of ay where $a > 0$. Likewise the second assumption tells us that, in the absence of prey, the predators will decrease at a rate of $-bx$ where $b > 0$.

Next, the third and fourth assumptions tell us how the population is affected by encounters between predators and prey. So, with each encounter the population of the predators will increase at a rate of αxy and the population of the prey will decrease at a rate of $-\beta xy$ where $\alpha > 0$ and $\beta > 0$. Putting all of this together we arrive at the following system.

$$x' = -bx + \alpha xy = x(\alpha y - b)$$

$$y' = ay - \beta xy = y(a - \beta x)$$

Note that this is a nonlinear system and we've not (nor will we here) discuss how to solve this kind of system. We simply wanted to give a "better" model for some population problems and to point out that not all systems will be nice and simple linear systems.

CODING OF MATHS PROBLEMS - ASSIGNMENT SOLUTIONS

Q1. Write Python code to find the Rank, Eigen Values and Eigen vectors of a matrix

In [1]:
```
import numpy as np  # IMPORTS NUMPY LIBRARY
from numpy.linalg import eig # IMPORTS LINEAR ALGEBRA LIBRARY
my_matrix = np.array([[1, 2, 1], [3, 5, 7], [3, 6, 3]]) # SAVES THE MATRIX AS MY_MATRIX
#my_matrix = np.array([[0, 2], [2, 3]])
#my_matrix = np.array([[2, 2, 4], [1, 3, 5], [2, 3, 4]])
print("Matrix")
for row in my_matrix:   # PRINTS THE ABOVE MATRIX
    print(row)
rank = np.linalg.matrix_rank(my_matrix) # FIND THE RANK OF THE MATRIX USING INBUILT FUNCTION
print("Rank of the given Matrix is : ",rank) # PRINTS THE RANK

w,v=eig(my_matrix) # CALUCULATES THE EIGNEN VALUUES AND EIGEN VECTORS USING INBUILT FUNCTION EIG IN LINALG OF NUMPY
print('E-value:', w) # PRINTS THE EIGEN VALUES
print('E-vector', v) # PRINTS THE EIGEN VECTORS
```

```
Matrix
[1 2 1]
[3 5 7]
[3 6 3]
Rank of the given Matrix is :  2
E-value: [ 1.14462220e+01  8.31146833e-16 -2.44622199e+00]
E-vector [[-0.20471545 -0.90913729  0.22146645]
 [-0.76217835  0.40406102 -0.71381097]
 [-0.61414636  0.10101525  0.66439936]]
```

Q2. Write Python code to solve system of equations using Gauss elimination method.

In [2]:
```
# METHOD 1
import numpy as np  #
def gaussElim(a,b):
    n = len(b)
    # Elimination phase
    for k in range(0,n-1):
        for i in range(k+1,n):
            if a[i,k] != 0.0:
                #if not null define λ
                lam = a [i,k]/a[k,k]
                #we calculate the new row of the matrix
                a[i,k+1:n] = a[i,k+1:n] - lam*a[k,k+1:n]
                #we update vector b
                b[i] = b[i] - lam*b[k]
                # backward substitution
    for k in range(n-1,-1,-1):
        b[k] = (b[k] - np.dot(a[k,k+1:n],b[k+1:n]))/a[k,k]

    return b

#initial coefficients
#a=np.array([[1.0,1.0,1.0],[1.0,-1.0,-1.0],[1.0,-2.0,3.0]])
#b=np.array([1.0,1.0,-5.0])
a=np.array([[25, 40, 20], [35, 20, 10], [40, 20, 30]])
b=np.array([220, 180, 260])
aOrig = a.copy() # save original matrix A
bOrig = b.copy() #save original vector b
x = gaussElim(a,b)
#print A transformed for check
print(aOrig)
print(bOrig)

#print(a)
#print(b)
print("x =\n",x)
```

```
[[25 40 20]
 [35 20 10]
 [40 20 30]]
[220 180 260]
x =
 [3 2 3]
```

In [3]:

```
# METHOD 2
# 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)

# A=np.array([[25, 40, 20], [35, 20, 10], [40, 20, 30]])
#B=np.array([220, 180, 260])

# Reading Augmented matrix coefficients i.e., [A B] 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')
```

```
Enter number of unknowns: 3
Enter Augmented Matrix Coefficients:
a[0][0]=25
a[0][1]=40
a[0][2]=20
a[0][3]=220
a[1][0]=35
a[1][1]=20
a[1][2]=10
a[1][3]=180
a[2][0]=40
a[2][1]=20
a[2][2]=30
a[2][3]=260

Required solution is:
X0 = 3.11	X1 = 1.94	X2 = 3.22
```

Q3. Write Python code to solve the system of equations using LU decomposition method.

In [4]:

```python
import numpy as np
def doolittle(A):

    n = A.shape[0]

    U = np.zeros((n, n), dtype=np.double)
    L = np.eye(n, dtype=np.double)

    for k in range(n):

        U[k, k:] = A[k, k:] - L[k,:k] @ U[:k,k:]
        L[(k+1):,k] = (A[(k+1):,k] - L[(k+1):,:] @ U[:,k]) / U[k, k]

    return L, U

#A = np.array([[1, 4, 5], [6, 8, 22], [32, 5., 5]])
#A=np.array([[2, 1, 1], [1, 3, 2], [3, 1, 2]])
#A=np.array([[3, 2, 7], [2, 3, 1], [3, 4, 1]])
#A=np.array([[7, 3, -1, 2], [3, 8, 1, -4], [-1, 1, 4, -1], [2, -4, -1, 6]] )

#for solution
#A=np.array([[2, -3, 10], [-1, 4, 2], [5, 2, 1]])
#B=np.array([3, 20, -12])

#A=np.array([[1, 3, 8], [1, 4, 3], [1, 3, 4]])
#B=np.array([4, -2, 1])

#A=np.array([[100, 10, 1], [400, 20, 1], [1600, 40, 1]])
#B=np.array([8, 16, 22])

A=np.array([[25, 40, 20], [35, 20, 10], [40, 20, 30]])
B=np.array([220, 180, 260])

L, U = doolittle(A)
#print(L)
#print(U)
n=A.shape[0]
# setup for displaying nicely
print("  Given Matrix \t\t\t   Lower Triangular\t\t  Upper Triangular")

# Displaying the result :
for i in range(n):
    # matrix
    for j in range(n):
        print(A[i][j], end="\t")
    print("", end="\t")

    # Lower
    for j in range(n):
        print(round(L[i][j],2), end="\t")
    print("", end="\t")

    # Upper
    for j in range(n):
        print(round(U[i][j],2), end="\t")
    print("")

print("solution of LB=y : ")
y=np.linalg.solve(L, B)
print(y)

print("the solution of the given system is :")
x=np.linalg.solve(U, y)
print(x)
```

```
  Given Matrix                    Lower Triangular                Upper Triangular
25      40      20              1.0     0.0     0.0             25.0    40.0    20.0
35      20      10              1.4     1.0     0.0             0.0     -36.0   -18.0
40      20      30              1.6     1.22    1.0             0.0     0.0     20.0
solution of LB=y :
[ 220.        -128.          64.44444444]
the solution of the given system is :
[3.11111111 1.94444444 3.22222222]
```

Q4. Write Python code to express the given matrix as a product of lower and upper triangular matrices.

In [5]:

```
import numpy as np
def doolittle(A):

    n = A.shape[0]

    U = np.zeros((n, n), dtype=np.double)
    L = np.eye(n, dtype=np.double)

    for k in range(n):

        U[k, k:] = A[k, k:] - L[k,:k] @ U[:k,k:]
        L[(k+1):,k] = (A[(k+1):,k] - L[(k+1):,:] @ U[:,k]) / U[k, k]

    return L, U

#A = np.array([[1, 4, 5], [6, 8, 22], [32, 5., 5]])
#A=np.array([[2, 1, 1], [1, 3, 2], [3, 1, 2]])
#A=np.array([[3, 2, 7], [2, 3, 1], [3, 4, 1]])
#A=np.array([[7, 3, -1, 2], [3, 8, 1, -4], [-1, 1, 4, -1], [2, -4, -1, 6]] )
A=np.array([[2, -3, 10], [-1, 4, 2], [5, 2, 1]])
B=np.array([3, 20, -12])
L, U = doolittle(A)
#print(L)
#print(U)
n=A.shape[0]
# setw is for displaying nicely
print("  Given Matrix \t\t\t   Lower Triangular\t\t  Upper Triangular")

# Displaying the result :
for i in range(n):
    # matrix
    for j in range(n):
        print(A[i][j], end="\t")
    print("", end="\t")

    # Lower
    for j in range(n):
        print(round(L[i][j],2), end="\t")
    print("", end="\t")

    # Upper
    for j in range(n):
        print(round(U[i][j],2), end="\t")
    print("")
```

```
  Given Matrix                     Lower Triangular               Upper Triangular
2       -3      10              1.0     0.0     0.0             2.0     -3.0    10.0
-1      4       2               -0.5    1.0     0.0             0.0     2.5     7.0
5       2       1               2.5     3.8     1.0             0.0     0.0     -50.6
```

Q5. Write Python code to find the nature of a quadratic form.

In [6]:

```
import numpy as np
from numpy.linalg import eig
pev=0
nev=0
zev=0
#a = np.array([[0, 2], [2, 3]])
#a = np.array([[2, 2, 4], [1, 3, 5], [2, 3, 4]])
#a = np.array([[2, -1, 1], [-1, 2, -1], [1, -1, 2]])
a = np.array([[-3, -1, -1], [-1, -3, 1], [-1, 1, -3]])
w,v=eig(a)
print('Eigen Values are :', w)
#print('E-vector', v)

n=w.shape[0]
for i in range(n):
    if w[i]>0:
        pev=pev+1
    elif w[i]<0:
        nev=nev+1
    else:
        zev=zev+1
print(f"Positive Eigen Values  {pev}")
print(f"Negative Eigen Values  {nev}")
print(f"Eigen Values with zero value  {zev}")

if zev>0:
    if nev==0:
        print ("The Nature is positive semidefinite")
    elif pev==0:
        print ("The Nature is negative semidefinite")
    else:
        print("The Nature is indefinite")
if n==pev:
    print("The Nature is positive definite")
elif n==nev:
    print("The Nature is negative definite")
else:
    print("The Nature is indefinite")
```

```
Eigen Values are : [-4. -1. -4.]
Positive Eigen Values  0
Negative Eigen Values  3
Eigen Values with zero value  0
The Nature is negative definite
```

Q6. Write Python code to solve the second order differential equation with constant coefficients.

In [7]:
```
from IPython.display import display
import math
import sympy as sy

sy.init_printing()  # LaTeX-like pretty printing for IPython

t = sy.Symbol("t", real=True)
m, k = sy.symbols('m k', real=True)  # gives C_1 Exp() + C_2 Exp() solution
# m, k = sy.symbols('m k', positive=True)  # gives C_1 sin() + C_2 cos() sol.
a0, b0 = sy.symbols('a0, b0', real=True)
y = sy.Function('y')

#Eq1 = sy.Eq(4*sy.diff(y(t), t, 2) + 9*y(t))
Eq1 = sy.Eq(sy.diff(y(t),t,3)-3*sy.diff(y(t), t, 2)+4*sy.diff(y(t), t, 1)-2*y(t),sy.exp(t)+sy.cos(t)+t)
print("ODE:")
display(Eq1)

print("Generic solution:")
y_sl0 = sy.dsolve(Eq1, y(t)).rhs  # take only right hand side
display(sy.Eq(y(t), y_sl0))

# Initial conditions:
cnd0 = sy.Eq(y_sl0.subs(t, 0), a0)  # y(0) = a0
cnd1 = sy.Eq(y_sl0.diff(t).subs(t, 0), b0)  # y'(0) = b0

#  Solve for C1, C2:
C1, C2 = sy.symbols("C1, C2")  # generic constants
C1C2_sl = sy.solve([cnd0, cnd1], (C1, C2))

# Substitute back into solution:
y_sl1 = sy.simplify(y_sl0.subs(C1C2_sl))
print("Solution with initial conditions:")
display(sy.Eq(y(t), y_sl1))
```

```
ODE:
```

$$-2y(t) + 4\frac{d}{dt}y(t) - 3\frac{d^2}{dt^2}y(t) + \frac{d^3}{dt^3}y(t) = t + e^t + \cos(t)$$

```
Generic solution:
```

$$y(t) = -\frac{t}{2} + (C_1 + C_2\sin(t) + C_3\cos(t) + t)\,e^t + \frac{3\sin(t)}{10} + \frac{\cos(t)}{10} - 1$$

```
Solution with initial conditions:
```

$$y(t) = -\frac{t}{2} + \left(C_3\cos(t) - C_3 + a_0 + t + \frac{(-10a_0 + 10b_0 - 17)\sin(t)}{10} + \frac{9}{10}\right)e^t + \frac{3\sin(t)}{10} + \frac{\cos(t)}{10} - 1$$