In [11]:

```
import sympy as sp
from sympy import *
init_printing()
from IPython.display import display, Latex, HTML, Math
import numpy as np
```

Assignment 1

In [2]:

```
# a)

# 1: False
# 2: True
# 3: False
# 4: True
# 5: False
# 6: True
# 7: False
# 8: False
# 9: True
```

In [3]:

```
# b)
# 2 is True
```

In [7]:

```
# c)
# Invertible since it has 4 non-zero eigenvalues
# It is diagonalizable since it has 4 distinct eigenvalues
A = Matrix([2, -1, 3]).T
A.rref()
```

Out[7]:

$$\left(\begin{bmatrix} 1 & -\frac{1}{2} & \frac{3}{2} \end{bmatrix}, \quad (0)\right)$$

In [8]:

```
# d) Dim V = 3
```

Assignment 2

In [9]:

```
a, b, x1, x2, x3 = symbols('a b x1 x2 x3')
```

In [12]:

```
# a)
A = Matrix([[-1, 3, 2], [1, 0, 1], [3, 3, a]])
A[1,0]*A.cofactor(1,0)+A[1,2]*A.cofactor(1,2)
```

Out[12]:

$$-3a + 18$$

In [17]:

```
# b)
B = Matrix([-8, 2, b])
L, U, perm = A.row_join(B).LUdecomposition()
U
```

Out[17]:

$$\begin{bmatrix} -1 & 3 & 2 & -8 \\ 0 & 3 & 3 & -6 \\ 0 & 0 & a-6 & b \end{bmatrix}$$

In [16]:

```
# So A != 6
```

Out[16]:

$$\begin{bmatrix} -1 & 3 & 2 & -8 \\ 0 & 3 & 3 & -6 \\ 0 & 0 & a-6 & b \end{bmatrix}$$

In [18]:

```
# c) a = 6 and b != 0

# d) a = 6 and b = 0
```

Assignment 3

In [3]:

```
A = Matrix([[1, -1, 3], [2, 1, 0], [-1, -5, 9]])
b = Matrix([1, 5, -7])
x, y, z = symbols('x y z')
var = Matrix([x, y, z])
display(Latex("$${}{} = {}$$".format(latex(A), latex(var), latex(b))))
```

$$\begin{bmatrix} 1 & -1 & 3 \\ 2 & 1 & 0 \\ -1 & -5 & 9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ -7 \end{bmatrix}$$

In [4]:

```
# b)
display(A.row_join(b))
L, U, perm = A.row_join(b).LUdecomposition()
display(U)
```

$$\begin{bmatrix} 1 & -1 & 3 & 1 \\ 2 & 1 & 0 & 5 \\ -1 & -5 & 9 & -7 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 3 & 1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

In [5]:

```
# c)
A.row_join(b).rref()
coef = Matrix([2, 1, 0])
display(Latex("$${} = {}+z{}$$".format(latex(var), latex(coef), latex(A.nullspace()[0
])))
```

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} + z \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$

In [24]:

```
# d) A inverse does not exist since it has a free variable.
```

Assignment 4

In [6]:

```
A = Matrix([[1,4], [3, 2]])
```

In [7]:

```
A = Matrix([[1,4], [3, 2]])

# a)

# First we check whether A is diagonalizable

vecs = A.eigenvects()
vecs
```

Out[7]:

$$\left[\left(-2,\ 1,\ \left[\begin{bmatrix} -\frac{4}{3} \\ 1 \end{bmatrix}\right]\right),\ \left(5,\ 1,\ \left[\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right]\right)\right]$$

In [33]:

```
# Since it has distinct eigenvalues, it will be diagonalizable.
d1 = vecs[1][0]
d2 = vecs[0][0]
q1 = vecs[1][2][0]
q2 = vecs[0][2][0]
Q = q1.row_join(q2)
Qinv = Q**-1
D = diag(d1, d2)
Q*D*Qinv
```

Out[33]:

$$\begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix}$$

In [14]:

```
# b)
# A^n = Q*D^n*Qinv
display(Math(r'A^n = Q\times D^n \times Q^{-1}'))
```

$$A^n = Q \times D^n \times Q^{-1}$$

Assignment 5

In [15]:

```
v1 = Matrix([1, 0, 0, 1])
x2 = Matrix([1, 0, 1, 1])
x3 = Matrix([3, 2, 1, 1])
v2 = x2-x2.project(v1)
v3 = x3-x3.project(v2)-x3.project(v1)
# So v1, v2 and v3 will now form an orthogonal basis for W.

# b)
u1 = v1.normalized()
u2 = v2.normalized()
u3 = v3.normalized()
display(U.T*U)
# So u1, u2, and u3 now form an orthonormal basis for W
```

$$\begin{bmatrix} 1 & -1 & 3 & 1 \\ -1 & 10 & -21 & 8 \\ 3 & -21 & 45 & -15 \\ 1 & 8 & -15 & 10 \end{bmatrix}$$

Assignment 6

In [36]:

```
from scipy.io import loadmat
Data = loadmat("TV_Viewing.mat")
Data.keys()
```

Out[36]:

```
dict_keys(['__header__', '__version__', '__globals__', 'X'])
```

In [77]:

```
import numpy as np
Mat = Matrix(Data['X'])
x = np.array(Mat[:,0])
y1 = np.array(Mat[:,1])
y = y1.astype(float)
```

In [67]:

```
# a) Design matrix for linear model

X1 = Matrix([ones(len(x),1)]).row_join(Matrix(x))
XtX = X1.T*X1
Xty = X1.T*Matrix(y)
Mat, _ = XtX.row_join(Xty).rref()
B1 = Mat[:,-1]
display(Latex("$$y_1(x) = {}+{}x$$".format(round(B1[0],2), round(B1[1], 4))))
```

$y_1(x) = 37.28 + 0.9981x$

In [68]:

```
# a) Design matrix for quadratic model

X2 = Matrix([ones(len(x),1)]).row_join(Matrix(x)).row_join(Matrix(x**2))
XtX = X2.T*X2
Xty = X2.T*Matrix(y)
Mat, _ = XtX.row_join(Xty).rref()
B2 = Mat[:,-1]
display(Latex("$$y_2(x) = {}+{}x+{}x^2$$"
              .format(round(B2[0],2), round(B2[1], 4), round(B2[2], 4))))
```

$y_2(x) = 52.75 + 0.2018x + 0.0088x^2$

In [78]:

```
# b)

display(Latex(
    "The error of the linear model is {}, and the error of the quadratic is model {}."
    " We conclude that the quadratic model has the best fit for this specific data."
    .format(round((y-X1*B1).norm(), 2), round((y-X2*B2).norm(), 2))))
```

The error of the linear model is 161.79, and the error of the quadratic is model 157.63. We conclude that the quadratic model has the best fit for this specific data.

Assignment 7

In [79]:

```
A = Matrix([[3, 1, 1], [-1, 3, 1]])
A
```

Out[79]:

$$\begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

In [82]:

```
AtA = A.T*A
vecs = AtA.eigenvects()
vecs
```

Out[82]:

$$\left[\left(0, \quad 1, \quad \left[\begin{bmatrix} -\frac{1}{5} \\ -\frac{2}{5} \\ 1 \end{bmatrix}\right]\right), \quad \left(10, \quad 1, \quad \left[\begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix}\right]\right), \quad \left(12, \quad 1, \quad \left[\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\right]\right)\right]$$

In [88]:

```
s1 = sqrt(vecs[2][0])
s2 = sqrt(vecs[1][0])
v1 = vecs[2][2][0].normalized()
v2 = vecs[1][2][0].normalized()
v3 = vecs[0][2][0].normalized()
u1 = (s1**-1)*A*v1
u2 = (s2**-1)*A*v2
U = u1.row_join(u2)
V = v1.row_join(v2).row_join(v3)
Vt = V.T
S = Matrix(diag(s1, s2)).row_join(zeros(2,1))
U*S*Vt
```

Out[88]:

$$\begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$