ALI 2019 Solution ¶

In [1]:
```
import sympy as sp
from sympy import *
init_printing()
from IPython.display import display, Latex, HTML, Math
import numpy as np
import pandas as pd
```

In [2]:
```
v1 = Matrix([2, 0, 0, 0])
x3 = Matrix([0, 0, -1, 0])
x4 = Matrix([0, 0, 0, 1])
v3 = x3 - x3.project(v1)
v4 = x4 - x4.project(v1)- x4.project(v3)
V = v1.row_join(v3).row_join(v4)
u1 = v1.normalized()
u3 = v3.normalized()
u4 = v4.normalized()
U = u1.row_join(u3).row_join(u4)
U
#display(U.T*U)
# Since U.T*U =
```

Out[2]:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Assignment 1

In [2]:
```
# a)
```

FALSE. Two matrices are row equivalent if one can be obtained from the other by a sequence of elementary row operations

In [ ]:
```
# b)
```

FALSE. A linear system must have no solution, unique solutions, or infinitely many solutions. Having no free variables indicates that the system does not have infinitely many solutions, but it does not determine which of the other two cases it might be.

In [ ]:
```
# c)
```

FALSE. The maximum number of pivot positions is number of rows = 5. Since there are 6 columns, there must be a column that does not contain a pivot position (col 6). Hence, x_6 is free and so $A\mathbf{x} = \mathbf{b}$ MUST have infinitely many solutions if it is consistent.

In [ ]:
```
# d)
```

FALSE. A needs 9 linearly independent eigenvectors for A to be diagonalizable. But there exist only 3 + 3 + 2 = 8 linearly independent eigenvectors.

Assignment 2

In [2]:
```
A = Matrix([[4, 8, -2], [-6, 2, 10], [-2, 6, 6]])
b = Matrix([2, 18, 15])
```

In [3]:
```
# a)
A.row_join(b).rref()
```

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

The system is inconsistent which means that **b** is not in the span of the columns of A.

In [26]:
```
# b)
```

From (a) we can conclude that the set is linearly dependent since we do not have pivots in all columns. Since column 3 has no pivots, it follows that $\mathbf{v_3}$ is linearly dependent on $\mathbf{v_1}$ and $\mathbf{v_2}$. We find the dependency relation.

In [4]:
```
A.nullspace()[0][0]
display(Latex("$${}v_1-v_2+{}v_3 = 0$$".format(2*A.nullspace()[0][0],
                                                2*A.nullspace()[0][2])))
print("or")

display(Latex("$${}v_1-v_2 = {}v_3$$".format(2*A.nullspace()[0][0],
                                             -2*A.nullspace()[0][2
])))
print("or")
display(Latex("$${}v_1+{}v_2 = v_3$$".format(-1*A.nullspace()[0][0], -1*A.nullspace()
[0][1])))
```

$3v_1 - v_2 + 2v_3 = 0$

or

$3v_1 - v_2 = -2v_3$

or

$-3/2v_1 + 1/2v_2 = v_3$

Assignment 3

In [12]:
```
# a)
a, b, c, x = symbols('a b c x')
A = Matrix([[a-x, b, c], [1, -x, 0], [0, 1, -x]])
A.det()
display(Latex("$$f(x) = -x^3+ax^2+bx+c$$"))
```

$f(x) = -x^3 + ax^2 + bx + c$

In [40]:
```
# b)
l = symbols('l')
A = Matrix([[2-l, 4], [3,2-l]])
L = solve(A.det(), l)
display(Math(r"\lambda_1 = {} \approx {} \\ \lambda_2 = {} \approx {} ".format(
    latex(L[0]), round(float(L[0]), 3),
    latex(L[1]), round(float(L[1]), 3))))
```

$\lambda_1 = 2 + 2\sqrt{3} \approx 5.464$

$\lambda_2 = -2\sqrt{3} + 2 \approx -1.464$

In [53]:
```
# c) I choose to cofactor expand on the second row since it contains the most zeroes.
A = Matrix([
      [2, 0, 0, 1]
    , [0, 1, 0, 0]
    , [1, 6, 2, 0]
    , [1, 1, -2, 3]
])
display(Math("Det(A) = {}".format(
    A[1,1]*A.cofactor(1,1)
)))
```

$Det(A) = 8$

In [102]:
```
# d)
A = Matrix([[1, 1, 3], [1, 2, 4], [1, 3, a]])
B = Matrix([2, 3, b])
L, U, perm = A.row_join(B).LUdecomposition()
display(U)
display(Math(r'One \:solution: a \neq' + repr(solve(U[2,2])[0]) + r'\:and\: b = \mathbb{R}'))
display(Math(r"Many \:solutions: a = {} \:and\: b = {}".format(solve(U[2,2])[0],solve(U[2,3])[0])))
display(Math(r"No \:solution: a = {} \:and\: b \neq {}".format(solve(U[2,2])[0],solve(U[2,3])[0])))
display(Math('a =' + repr(solve(A.det()-1)[0]) + '\:\:\:when\: Det(A) = 1'))
```

$$\begin{bmatrix} 1 & 1 & 3 & 2 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & a-5 & b-4 \end{bmatrix}$$

$One\ solution : a \neq 5\ and\ b = \mathbb{R}$

$Many\ solutions : a = 5\ and\ b = 4$

$No\ solution : a = 5\ and\ b \neq 4$

$a = 6 \quad when\ Det(A) = 1$

Assignment 4

In [116]:
```
display(Latex("An orthonormal basis for W is"))
GramSchmidt(Matrix([[2, 0, -1, 1]]).nullspace(), True)
```

An orthonormal basis for W is

Out[116]:
$$\left[\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{\sqrt{5}}{5} \\ 0 \\ \frac{2\sqrt{5}}{5} \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{\sqrt{30}}{15} \\ 0 \\ \frac{\sqrt{30}}{30} \\ \frac{\sqrt{30}}{6} \end{bmatrix}\right]$$

Assignment 5

In [2]:
```
A = Matrix([[3, -1, -2], [2, 0, -2], [2, -1, -1]])

# a) We check whether A is diagonalizable

vecs = A.eigenvects()
display(vecs)
display(Latex("Since A has n linearly independent eigenvectors, A is diagnonalizable"
))
```

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

Since A has n linearly independent eigenvectors, A is diagnonalizable

In [74]:
```
# b)

d1 = vecs[1][0]
d2 = vecs[1][0]
d3 = vecs[0][0]
x1 = vecs[1][2][0]
x2 = vecs[1][2][1]
x3 = vecs[0][2][0]
X = x1.row_join(x2).row_join(x3)
D = diag(d1, d2, d3)
Xinv = X**-1
display(Math('XDX^{-1} ='))
display(X*D*Xinv)
display(Math('X^{-1}AX =D:'))
display(Latex("$${}{}{} = {}$$".format(latex(Xinv), latex(A), latex(X), latex(D))))
```

$XDX^{-1} =$

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

$X^{-1}AX = D:$

$$\begin{bmatrix}2 & 0 & -2\\2 & -1 & -1\\-2 & 1 & 2\end{bmatrix}\begin{bmatrix}3 & -1 & -2\\2 & 0 & -2\\2 & -1 & -1\end{bmatrix}\begin{bmatrix}\frac{1}{2} & 1 & 1\\1 & 0 & 1\\0 & 1 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 0\end{bmatrix}$$

In [76]:
```
# c)
display(Math('A^{k} = XD^{k}X^{-1}'))
```

$A^k = XD^kX^{-1}$

Assignment 6

In [77]:
```
df = pd.read_excel('Smoking_and_Cancer.xlsx')
df.head()
```

Out[77]:

	State	Cigarettes sold	lung cancer
0	AL	18.20	17.05
1	AZ	25.82	19.80
2	AR	18.24	15.98
3	CA	28.60	22.07
4	CT	31.10	22.83

In [79]:
```
x = df['Cigarettes sold']
y = df['lung cancer']
```

In [81]:
```
# 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("$$f_1(x) = {}+{}x$$".format(round(B1[0],2), round(B1[1], 4))))
```

$f_1(x) = 6.47 + 0.5291x$

In [82]:
```
# a) Design matrix for quadratic model

X2 = 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("$$f_2(x) = {}x+{}x^2$$"
              .format(round(B2[0],2), round(B2[1], 4))))
```

$f_2(x) = 1.05x + -0.0098x^2$

In [83]:
```
# 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((Matrix(y)-X1*B1).norm(), 2), round((Matrix(y)-X2*B2).norm(), 2))))
```

The error of the linear model is 19.87, and the error of the quadratic is model 19.19. We conclude that the quadratic model has the best fit for this specific data.

In [87]:
```
# c)
display(Math('f_2(50) = ' + repr(round(B2[0]*50+B2[1]*50**2, 2))))
```

$f_2(50) = 27.84$

Assignment 7

In [90]:
```
A = Matrix([[2, 5, 4], [6, 3, 0], [6, 3, 0],[2, 5, 4]])
AtA = A.T*A
vecs = AtA.eigenvects()
vecs
```

Out[90]:
$$\left[\left(0,\quad 1,\quad \left[\begin{bmatrix}\frac{1}{2}\\-1\\1\end{bmatrix}\right]\right),\ \left(36,\quad 1,\quad \left[\begin{bmatrix}-1\\\frac{1}{2}\\1\end{bmatrix}\right]\right),\ \left(144,\quad 1,\quad \left[\begin{bmatrix}2\\2\\1\end{bmatrix}\right]\right)\right]$$

In [114]:
```
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

# We need two more eigenvectors for U

u3 = Matrix([u1.T, u2.T]).nullspace()[0].normalized()
u4 = Matrix([u1.T, u2.T]).nullspace()[1].normalized()

# They will be orthogonal to u1 and u2, but we need to check whether they are mutuall
y orthogonal

display(Math('u_3 \cdot u_4 ={}'.format(u3.dot(u4))))

U = u1.row_join(u2).row_join(u3).row_join(u4)
V = v1.row_join(v2).row_join(v3)
Vt = V.T
S = diag(s1,s2).row_join(zeros(2,1)).col_join(zeros(2,3))

display(Math('U \Sigma V^T = {}{}{}'.format(latex(U), latex(S), latex(Vt))))
display(Latex("Test:"))
display(U*S*Vt)
display(A)
```

$u_3 \cdot u_4 = 0$

$$U\Sigma V^T = \begin{bmatrix}\frac{1}{2} & \frac{1}{2} & 0 & -\frac{\sqrt{2}}{2}\\\frac{1}{2} & -\frac{1}{2} & -\frac{\sqrt{2}}{2} & 0\\\frac{1}{2} & -\frac{1}{2} & \frac{\sqrt{2}}{2} & 0\\\frac{1}{2} & \frac{1}{2} & 0 & \frac{\sqrt{2}}{2}\end{bmatrix}\begin{bmatrix}12 & 0 & 0\\0 & 6 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}\begin{bmatrix}\frac{2}{3} & \frac{2}{3} & \frac{1}{3}\\-\frac{2}{3} & \frac{1}{3} & \frac{2}{3}\\\frac{1}{3} & -\frac{2}{3} & \frac{2}{3}\end{bmatrix}$$

Test:

$$\begin{bmatrix}2 & 5 & 4\\6 & 3 & 0\\6 & 3 & 0\\2 & 5 & 4\end{bmatrix}$$

$$\begin{bmatrix}2 & 5 & 4\\6 & 3 & 0\\6 & 3 & 0\\2 & 5 & 4\end{bmatrix}$$