MEEN2450 Ryan Ruby 40848507 Assignment 03

Exercise 1

a) [A]: 3×2 [B]: 3×3 [C]: 3×1
[D]: 2×4 [E]: 3×3 [F]: 2×3 [G]: 1×3

b) Square: [B], [E]
Colum: [C] Row: [G]

c) $a_{12} = 7$ $b_{23} = 7$ d_{32} = undefined
$f_{12} = 0$ $g_{12} = 6$
(row, colum; only 2 rows)

d) See code Output

Exercise 2

$$-2.2x_1 + 20x_2 = 240$$
$$-1x_1 + 8.7x_2 = 87$$

$$\begin{bmatrix} -2.2 & 20 \\ -1 & 8.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 240 \\ 87 \end{bmatrix}$$

$[A]$ $\vec{x}$ $\vec{b}$

a)
b) See Code Output

Exercise 3

$$5x_1 + 1x_2 - 0.5x_3 = 13.5$$
$$-6x_1 - 12x_2 + 4x_3 = -123$$
$$2x_1 + 2x_2 + 10x_3 = -43$$

$$\begin{bmatrix} 5 & 1 & -0.5 \\ -6 & -12 & 4 \\ 2 & 2 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 13.5 \\ -123 \\ -43 \end{bmatrix}$$

$[A]$ $\vec{x}$ $\vec{b}$

a) See Code Output

b) $x_1 = 0.5$ $x_2 = 8$ $x_3 = -6$

$5(0.5) + (8) - 0.5(-6) = 13.5$
$13.5 = 13.5$ ✓

$-6(0.5) - 12(8) + 4(-6) = -123$
$-123 = -123$ ✓

$2(0.5) + 2(8) + 10(-6) = -43$
$-43 = -43$ ✓

```
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.4.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/hoops/OneDrive/Documents/School/ME EN 2450 Numerical Methods/
HW3/MatrixOperations.py', wdir='C:/Users/hoops/OneDrive/Documents/School/ME EN 2450
Numerical Methods/HW3')
Exercise 1
ansA =
[[ 5  8 15]
 [ 8  4 10]
 [ 6  0 10]]
ansB =
[[19 49 25]
 [ 5 14  7]
 [21 42 23]]
ansC =
[[ 3 -2 -1]
 [-6  0  4]
 [-2  0 -2]]
ansD =
[[28 21 49]
 [ 7 14 49]
 [14  0 28]]
ansE =
[[25 13 74]
 [36 25 75]
 [28 12 52]]
ansF =
[[3 6 1]]
ansG =
[[54 76]
 [41 53]
 [28 38]]
ansH =
[[ 9  2]
 [ 4 -1]
 [ 3  7]
 [-6  5]]



Exercise 2
ansA =
0.860000000000001
ansB =
x1 = [404.65116279]
x2 = [56.51162791]
```

```
Exercise 3
Naive Gauss Elimination Steps:

Original Matrix:
[[  5.    1.   -0.5]
 [ -6.  -12.    4. ]
 [  2.    2.   10. ]]

Perform forward elimination:
A =
[[  5.    1.   -0.5]
 [  0.  -12.    4. ]
 [  2.    2.   10. ]]
b =
[[  13.5]
 [-123. ]
 [ -43. ]]

A =
[[  5.    1.   -0.5]
 [  0.  -10.8   4. ]
 [  2.    2.   10. ]]
b =
[[  13.5]
 [-123. ]
 [ -43. ]]

A =
[[  5.    1.   -0.5]
 [  0.  -10.8   3.4]
 [  2.    2.   10. ]]
b =
[[  13.5]
 [-123. ]
 [ -43. ]]

A =
[[  5.    1.   -0.5]
 [  0.  -10.8   3.4]
 [  0.    2.   10. ]]
b =
[[  13.5]
 [-106.8]
 [ -43. ]]

A =
[[  5.    1.   -0.5]
 [  0.  -10.8   3.4]
 [  0.    1.6  10. ]]
b =
[[  13.5]
 [-106.8]
 [ -43. ]]
```

```
A =
[[  5.    1.   -0.5]
 [  0.  -10.8   3.4]
 [  0.    1.6  10.2]]
b =
[[  13.5]
 [-106.8]
 [ -43. ]]

A =
[[  5.    1.   -0.5]
 [  0.  -10.8   3.4]
 [  0.    0.   10.2]]
b =
[[  13.5]
 [-106.8]
 [ -48.4]]

A =
[[  5.          1.         -0.5      ]
 [  0.        -10.8         3.4      ]
 [  0.          0.         10.7037037]]
b =
[[  13.5]
 [-106.8]
 [ -48.4]]


Perform back substitution:
A =
[[  5.          1.         -0.5      ]
 [  0.        -10.8         3.4      ]
 [  0.          0.         10.7037037]]
b =
[[  13.5       ]
 [-106.8       ]
 [ -64.22222222]]

x =
[[ 0.]
 [ 0.]
 [-6.]]

x =
[[ 0.]
 [ 8.]
 [-6.]]

x =
[[ 0.5]
 [ 8. ]
 [-6. ]]

Final x =
[[ 0.5]
```

```
 [ 8. ]
 [-6. ]]
```

In [2]:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Feb 12 16:00:28 2019
HW3
@author: Ryan Dalby
"""

import numpy as np

"""
Performs matrix operations for Exercise 1
"""
print("Exercise 1")
A = np.array([[4,7],[1, 2],[5, 6]])
B = np.array([[4,3,7],[1,2,7],[2,0,4]])
C = np.array([[3],[6],[1]])
D = np.array([[9,4,3,-6], [2,-1,7,5]])
E = np.array([[1,5,8],[7,2,3],[4,0,6]])
F = np.array([[3,0,1], [1,7,3]])
G = np.array([[7,6,4]])


ansA = E + B
ansB = np.matmul(A, F)
ansC = B - E
ansD = 7 * B
ansE = np.matmul(E, B)
ansF = np.transpose(C)
ansG = np.matmul(B, A)
ansH = np.transpose(D)
print("ansA = \n{}\nansB = \n{}\nansC = \n{}\nansD = \n{}\nansE = \n{}\nansF = \n{}\nansG
print("\n\n\n\n")
"""
Solves given system of equations (represented by Mx = b) for Exercise 2
"""
print("Exercise 2")
M2 = np.array([[-2.2,20], [-1, 8.7]])
b2 = np.array([[240],[87]])
ex2Ansa = np.linalg.det(M2)
ex2Ansb = np.linalg.solve(M2, b2)

print("ansA = \n{}\nansB = \nx1 = {}\nx2 = {}".format(ex2Ansa, ex2Ansb[0], ex2Ansb[1]))

print("\n\n\n\n")

"""
Exercise 3
"""
def gaussElimination(A, b):
    """
    Uses naive gauss elimination method without pivoting to determine solutions to Ax = b
```

```
    """
    n = np.shape(A)[0] #assuming nxn matrix

    print("Naive Gauss Elimination Steps:\n")
    print("Original Matrix:")
    print(A,"\n")
    print("Perform forward elimination:")
    #Perform forward elimination
    for k in range(n - 1):
        for i in range(k + 1, n):
            s = A[i,k] / A[k,k]
            for j in range(k, n):
                A[i,j] = A[i,j] - s * A[k,j]
                print("A = \n{}\nb = \n{}\n".format(A,b))
            b[i] = b[i] - s * b[k]

    print()
    print("Perform back substitution:")
    #Perform back substitution
    xSol = np.zeros(shape = (n,1))
    xSol[n-1] = b[n-1] / A[n-1,n-1] #index last element of xSol and set it to the solution
    print("A = \n{}\nb = \n{}\n".format(A,b))
    for i in range(n-1, -1, -1):
        s = 0
        for j in range(i+1, n):
            s = s + A[i,j] * xSol[j]
        xSol[i] = (b[i] - s) / A[i,i]
        print("x = \n{}\n".format(xSol))
    return xSol

print("Exercise 3")
#z = gaussElimination(M2, b2) #Test with last problem
#print(z)

M3 = np.array([[5,1,-.5], [-6,-12,4], [2,2,10]])
b3 = np.array([[13.5], [-123], [-43]])

ansEx3 = gaussElimination(M3, b3)
print("Final x = \n{}".format(ansEx3))
```