==

Gaussian Elimination Algorithm in Python

==

In [10]:

```
def gauss_method(aug_mat):

    # num of rows/unknown variables
    n_row,n_col = aug_mat.shape

    for i in range(n_row):

        # get the divisor from the main diagonal
        divisor = aug_mat[i,i]

        # change the main diagonal all to 1s
        #for each column j in row i:
        for j in range(n_col):
            aug_mat[i,j] /= divisor ## aug_mat[i,j] = aug_mat[i,j]/divisor

        # use the main diagonal to cancel other elements to 0
        #for each row j not equal to i:
        for j in range(n_row):
            if j != i:
                multiplier = -aug_mat[j,i]
                aug_mat[j,:] += multiplier*aug_mat[i,:]

    return aug_mat
```

In [11]:

```
x = np.array([[2,-3,1,-22],[7,9,-3,14],[6,7,2,91]], dtype = float)
x
```

Out[11]:

```
array([[  2.,  -3.,   1., -22.],
       [  7.,   9.,  -3.,  14.],
       [  6.,   7.,   2.,  91.]])
```

In [13]:

Out[13]:

```
array([-3.,  9.,  7.])
```

1. Solve the system of equations using Gaussian Elimination Algorithm

Example 1:

In [9]:

```
import numpy as np

x = np.array([[2,-3,1,-22],[7,9,-3,14],[6,7,2,91]], dtype = float)
x
```

Out[9]:

```
array([[  2.,  -3.,   1., -22.],
       [  7.,   9.,  -3.,  14.],
       [  6.,   7.,   2.,  91.]])
```

In [6]:

```
print(gauss_method(x))
```

```
[[ 1.  0.  0. -4.]
 [ 0.  1.  0. 11.]
 [ 0.  0.  1. 19.]]
```

In [71]:

```
x = -4
y = 11
z = 19
```

Example 2:

In [7]:

```
aug = np.array([[3,-5,5],[7,1,37]],dtype=float)
print(aug)
```

```
[[ 3. -5.  5.]
 [ 7.  1. 37.]]
```

In [8]:

```
print(gauss_method(aug))
```

```
[[1. 0. 5.]
 [0. 1. 2.]]
```

In [74]:

```
x = 5
y = 2
```

Example 3:

In [75]:

```
import numpy as np
a = np.array([[2,-3,  1],
              [7, 9, -3],
              [6, 7,  2]], dtype = float)

b = np.array([[-22], [14], [91]], dtype = float)

aug_mat = np.hstack([a,b])

print(gauss_method(aug_mat))
```

```
[[ 1.  0.  0. -4.]
 [ 0.  1.  0. 11.]
 [ 0.  0.  1. 19.]]
```

In [76]:

```
aug = np.array([[2,-3,  1, -22],
              [7, 9, -3, 14],
              [6, 7,  2, 91]], dtype = float)
```

In [77]:

```
print(gauss_method(aug))
```

```
[[ 1.  0.  0. -4.]
 [ 0.  1.  0. 11.]
 [ 0.  0.  1. 19.]]
```

In [ ]:

```
```

2. Find the inverse of the Matrix using Gaussian Elimination Algorithm

Practice 1 Demo

In [78]:

```
practice1 = np.array([[-1,2,-3],[2,1,0],[4,-2,5]], dtype = float)
print(practice1)
```

```
[[-1.  2. -3.]
 [ 2.  1.  0.]
 [ 4. -2.  5.]]
```

In [79]:

```
I3 = np.eye(3,3)
print(I3)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [81]:
```
practice1_aug = np.hstack([practice1,I3])
print(practice1_aug)
```

```
[[-1.  2. -3.  1.  0.  0.]
 [ 2.  1.  0.  0.  1.  0.]
 [ 4. -2.  5.  0.  0.  1.]]
```

In [82]:
```
print(gauss_method(practice1_aug))
```

```
[[ 1.  0.  0. -5.  4. -3.]
 [ 0.  1.  0. 10. -7.  6.]
 [ 0.  0.  1.  8. -6.  5.]]
```

Practice 2 Demo

In [83]:
```
practice2 = np.array([[1,3],[2,2]], dtype = float)
print(practice2)
```

```
[[1. 3.]
 [2. 2.]]
```

In [84]:
```
I2 = np.eye(2,2)
print(I2)
```

```
[[1. 0.]
 [0. 1.]]
```

In [85]:
```
practice2_aug = np.hstack([practice2,I2])
print(practice2_aug)
```

```
[[1. 3. 1. 0.]
 [2. 2. 0. 1.]]
```

In [86]:
```
print(gauss_method(practice2_aug))
```

```
[[ 1.    0.   -0.5   0.75]
 [-0.    1.    0.5  -0.25]]
```

In [ ]:

In [31]:
```
A = np.array([
    [2,7,1],
    [7,0,-3],
    [1,-3,4]
])
print(A)
```

```
[[ 2  7  1]
 [ 7  0 -3]
 [ 1 -3  4]]
```

In [32]:

```
I = np.eye(3)
print(I)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [33]:

```
A_aug = np.hstack([A,I])
print(A_aug)
```

```
[[ 2.  7.  1.  1.  0.  0.]
 [ 7.  0. -3.  0.  1.  0.]
 [ 1. -3.  4.  0.  0.  1.]]
```

In [34]:

```
print(gauss_method(A_aug))
```

```
[[ 1.          0.          0.          0.03515625  0.12109375  0.0820312
5]
 [-0.          1.          0.          0.12109375 -0.02734375 -0.0507812
5]
 [ 0.          0.          1.          0.08203125 -0.05078125  0.1914062
5]]
```

In [ ]:

```

```

In [2]:

```
import numpy as np
```

In [3]:

```
A = np.array([
 [1,3],
 [2,2]
])
print(A)
```

```
[[1 3]
 [2 2]]
```

In [6]:

```
I2b2 = np.eye(2)
print(I2b2)
```

```
[[1. 0.]
 [0. 1.]]
```

In [8]:

```
aug_A = np.hstack([A,I2b2])
print(aug_A)
```

```
[[1. 3. 1. 0.]
 [2. 2. 0. 1.]]
```

In [11]:

```
gauss_method(aug_A)
```

Out[11]:

```
array([[ 1.  ,  0.  , -0.5 ,  0.75],
       [-0.  ,  1.  ,  0.5 , -0.25]])
```

In [14]:

```
A2 = np.array([
[-1,2,-3],
[2,1,0],
[4,-2,5]
])
print(A2)
```

```
[[-1  2 -3]
 [ 2  1  0]
 [ 4 -2  5]]
```

In [16]:

```
I3b3 = np.eye(3)
print(I3b3)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [18]:

```
aug_A2 = np.hstack([A2,I3b3])
print(aug_A2)
```

```
[[-1.  2. -3.  1.  0.  0.]
 [ 2.  1.  0.  0.  1.  0.]
 [ 4. -2.  5.  0.  0.  1.]]
```

In [19]:

```
gauss_method(aug_A2)
```

Out[19]:

```
array([[ 1.,  0.,  0., -5.,  4., -3.],
       [ 0.,  1.,  0., 10., -7.,  6.],
       [ 0.,  0.,  1.,  8., -6.,  5.]])
```

In [20]:

```
B = np.array([
[2,7,1],
[7,0,-3],
[1,-3,4]
])
print(B)
```

```
[[ 2  7  1]
 [ 7  0 -3]
 [ 1 -3  4]]
```

In [21]:

```
I3b3
```

Out[21]:

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [22]:

```
aug_B = np.hstack([B, I3b3])
print(aug_B)
```

```
[[ 2.  7.  1.  1.  0.  0.]
 [ 7.  0. -3.  0.  1.  0.]
 [ 1. -3.  4.  0.  0.  1.]]
```

In [25]:
```
gauss_method(aug_B)
```

Out[25]:
```
array([[ 1.        ,  0.        ,  0.        ,  0.03515625,  0.12109375,
         0.08203125],
       [ 0.        ,  1.        ,  0.        ,  0.12109375, -0.02734375,
        -0.05078125],
       [ 0.        ,  0.        ,  1.        ,  0.08203125, -0.05078125,
         0.19140625]])
```

In [ ]:
```
 1.        ,  0.        ,  0.        ,  0.03515625,  0.12109375, 0.08203125
 0.        ,  1.        ,  0.        ,  0.12109375, -0.02734375, -0.0507812
 0.        ,  0.        ,  1.        ,  0.08203125, -0.05078125, 0.19140625
```