Coding Gauss-Jordan Elimination for Calculating Matrix Systems and Matrix Inverse

Kamyab Safaie

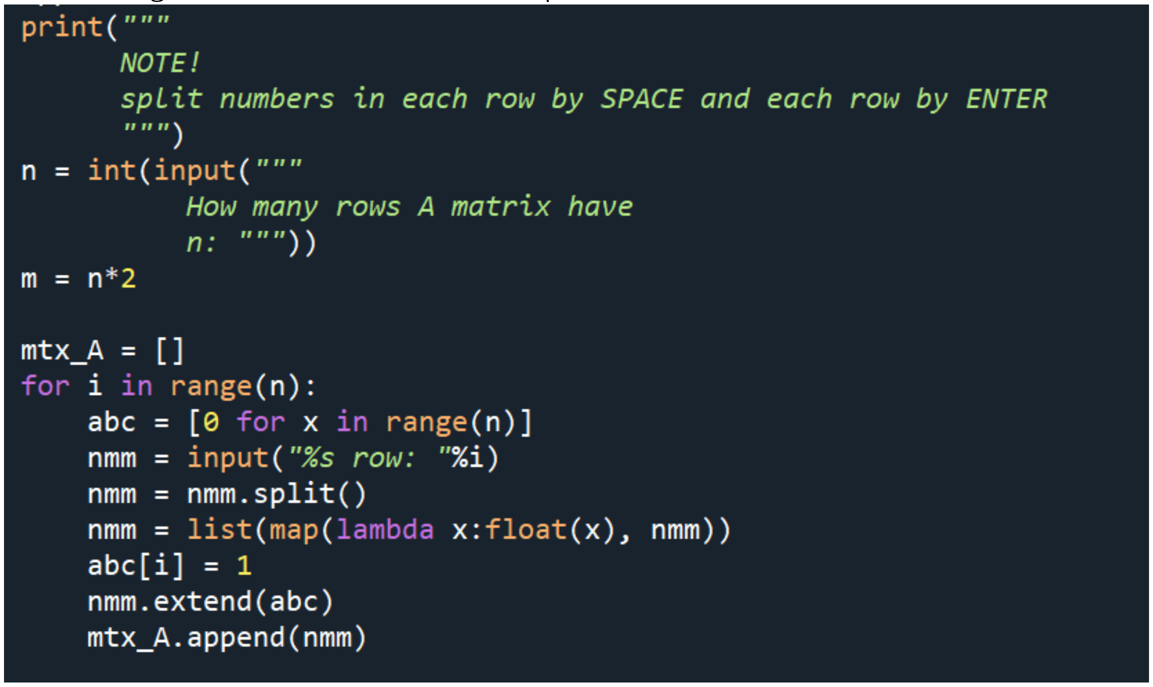
Student of Chemical Engineering at Sharif University of Technology

In this paper I present you my python code for calculating matrix systems using Gauss-Jordan method. The code is segmented and clear but here is more explanation for better understanding. You can also get more information about Gauss-Jordan method in Use of Gauss-Jordan Elimination Method to Balance Chemical Reaction paper by M.V.Deshmukh[1].

NOTE that if the determinant of {A} equal zero the program didn't work.

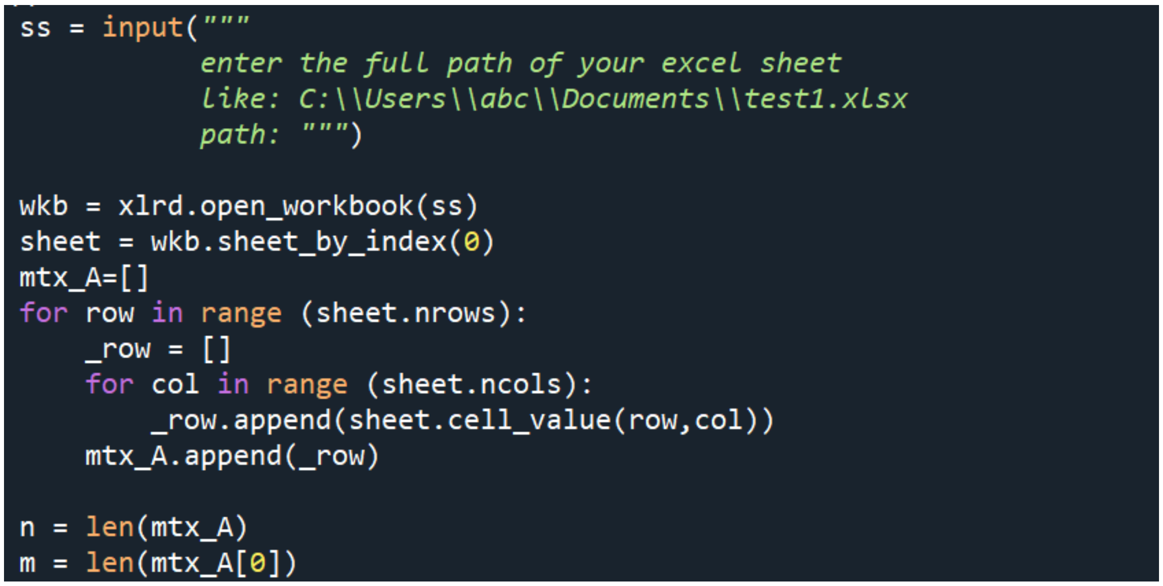
Segments:

- **Inputs:**
  This code can calculate two types of problems, Matrix systems and matrix inverse. Indeed, the algorithms are similar.
  For matrix systems for fascinating the progress, we get {A|B} as input and for matrix inverse we get {A} and then we create {A|I} like the code below:



```
print("""
      NOTE!
      split numbers in each row by SPACE and each row by ENTER
      """)
n = int(input("""
          How many rows A matrix have
          n: """))
m = n*2

mtx_A = []
for i in range(n):
    abc = [0 for x in range(n)]
    nmm = input("%s row: "%i)
    nmm = nmm.split()
    nmm = list(map(lambda x:float(x), nmm))
    abc[i] = 1
    nmm.extend(abc)
    mtx_A.append(nmm)
```

“abc” is a list of zeros with n component. Then by abc[i] = 1 we create i_{th} row of {I} and add (extend) it to i_{th} row of {A} and creating {A|I}. **“nmm”** is also each row of {A} which we split it by space then turn all components to float numbers.
We could get two types of input. For engineering systems, we usually use excel for storing data. So, for reading matrix data from excel we could use **xlrd** module by importing it to the program. The code part is like:



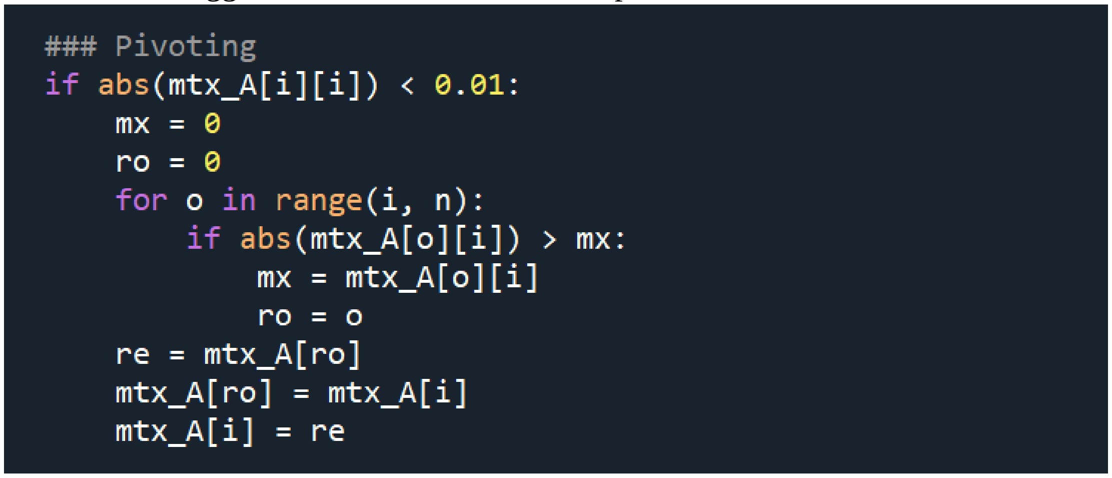
```
ss = input("""
            enter the full path of your excel sheet
            like: C:\\Users\\abc\\Documents\\test1.xlsx
            path: """)

wkb = xlrd.open_workbook(ss)
sheet = wkb.sheet_by_index(0)
mtx_A=[]
for row in range (sheet.nrows):
    _row = []
    for col in range (sheet.ncols):
        _row.append(sheet.cell_value(row,col))
    mtx_A.append(_row)

n = len(mtx_A)
m = len(mtx_A[0])
```

NOTE that you should enter the full path of your excel file like the example above or instead you can save your program in your excel file folder and then you could just enter filename.xlsx for input.

- ## Body

We use the same algorithm for each row so let’s just have a look.

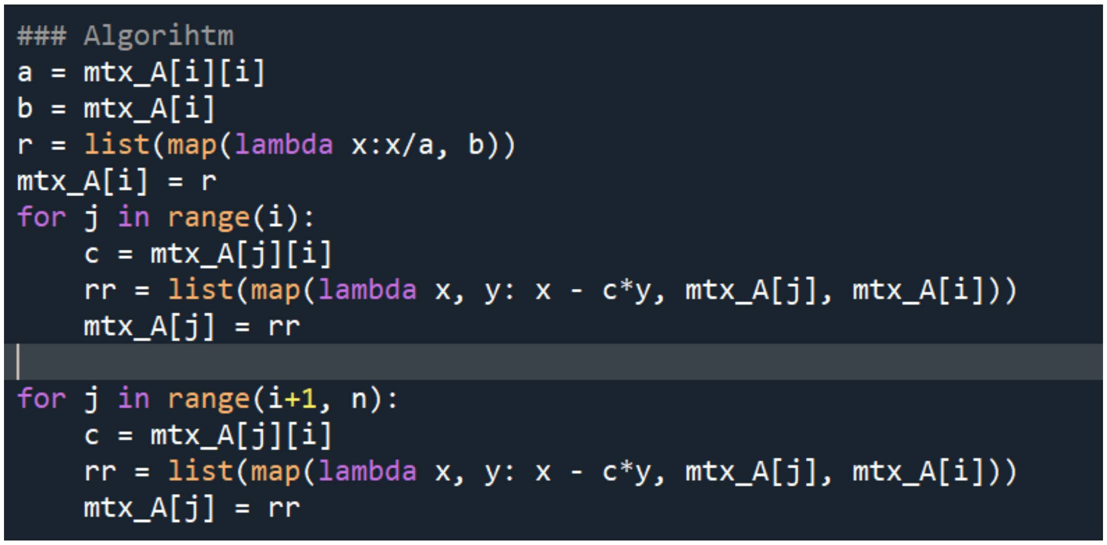
- **Pivoting:** We use pivoting algorithm when we face zero or a very little number in diameter of {A} that effect the algorithm. So, we change that row with a row below which have a biggest number of that column to prevent errors. Code:



```
### Pivoting
if abs(mtx_A[i][i]) < 0.01:
    mx = 0
    ro = 0
    for o in range(i, n):
        if abs(mtx_A[o][i]) > mx:
            mx = mtx_A[o][i]
            ro = o
    re = mtx_A[ro]
    mtx_A[ro] = mtx_A[i]
    mtx_A[i] = re
```

The code is clear enough. “**ro**” is the number of the row which has the largest number of the column. And then we could easily replace two rows.

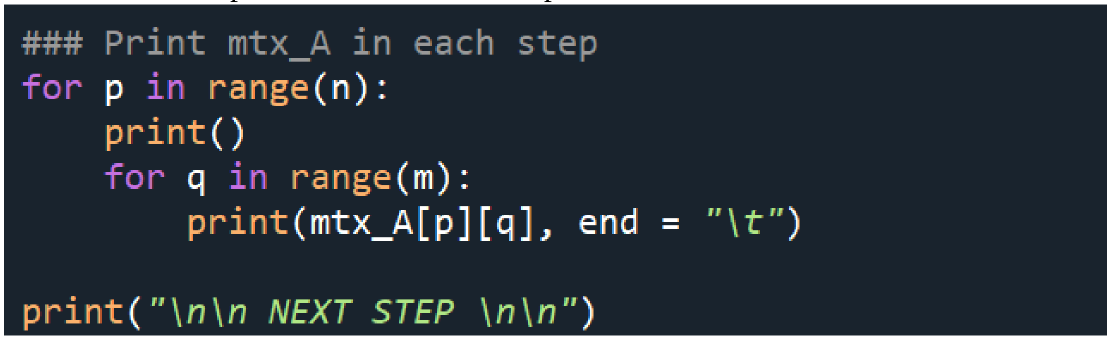
- **Gauss-Jordan Algorithm:** I mentioned a paper earlier to study the Gauss-Jordan elimination. Now let’s see how the code works.



```
### Algorihtm
a = mtx_A[i][i]
b = mtx_A[i]
r = list(map(lambda x:x/a, b))
mtx_A[i] = r
for j in range(i):
    c = mtx_A[j][i]
    rr = list(map(lambda x, y: x - c*y, mtx_A[j], mtx_A[i]))
    mtx_A[j] = rr

for j in range(i+1, n):
    c = mtx_A[j][i]
    rr = list(map(lambda x, y: x - c*y, mtx_A[j], mtx_A[i]))
    mtx_A[j] = rr
```

As you can see **“a”** is i_{th} number of diameter {A} and **“b”** is i_{th} row of {A} so we divided all numbers of **“b”** by **“a”** and then replace it with the i_{th} row in {A}. Then we should make all number of the column zero except the diameter number. In this case could use two loops, one for numbers above and one for numbers below the diameter. The algorithms are the same. We also could use one loop with the condition of “if i==j then continue”. In each step we put a number of the column in **“c”** and then minus that row by the factor **“c*x”** which **“x”** is the i_{th} row of {A}.

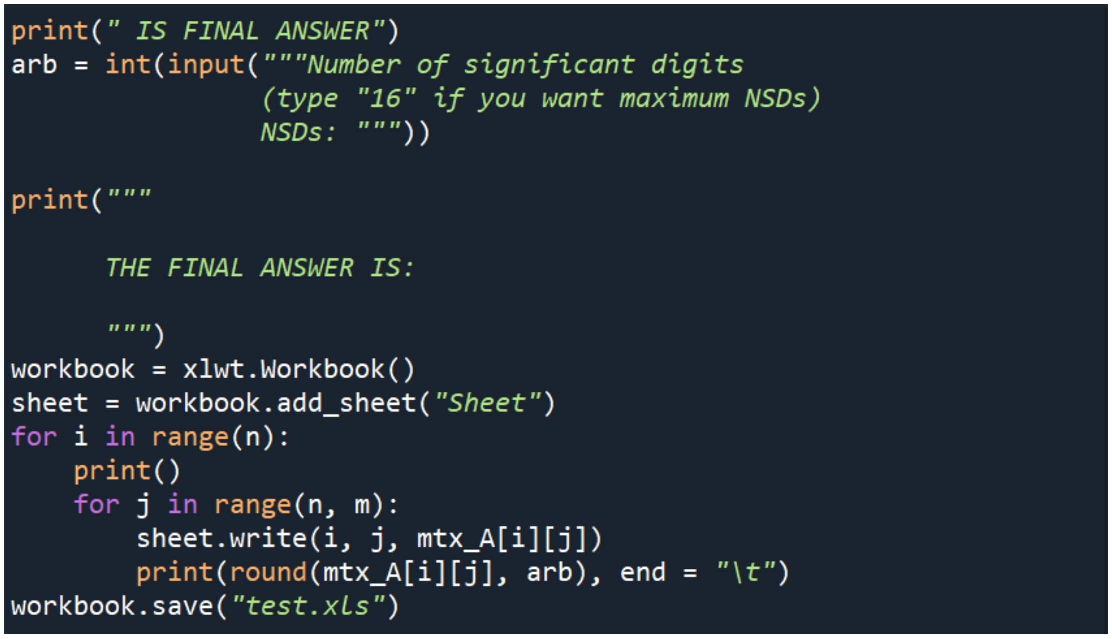
- **Print each step:** After each elimination we could print the {A} to see the changes of {A} throw the process. The code is simple:



```
### Print mtx_A in each step
for p in range(n):
    print()
    for q in range(m):
        print(mtx_A[p][q], end = "\t")

print("\n\n NEXT STEP \n\n")
```

Which **“n”** is number of rows in {A} or {A|B} and **“m”** is number of column in {A|B} for matrix system or {A|I} for matrix inverse which is double **“n”** or **“2*n”**.

- **Print the final answer:** For printing final answer we use the same code like last part but in this case, we just print the answer not the hole {A|B} or {A|I}. And there is an optional case which we could export the final answer to an excel sheet. For this part we should first import **xlwt** to the program then use the code below:



```
print(" IS FINAL ANSWER")
arb = int(input("""Number of significant digits
                (type "16" if you want maximum NSDs)
                NSDs: """))

print("""

      THE FINAL ANSWER IS:

      """)
workbook = xlwt.Workbook()
sheet = workbook.add_sheet("Sheet")
for i in range(n):
    print()
    for j in range(n, m):
        sheet.write(i, j, mtx_A[i][j])
        print(round(mtx_A[i][j], arb), end = "\t")
workbook.save("test.xls")
```

NOTE that you can easily use the same code for exporting each step matrix to an excel sheet but in this case, I prefer to not do that.

- **Resources:** I learn the Gauss-Jordan method from my Numerical Calculation teacher, **Dr.Saadatmand.** But you can learn more by reading the paper I mention earlier **by M.V.Deshmukh** which I zip it with the paper and the code.
  You could also find some really helpful notes for coding from **Stack Overflow** which is a forum that you could find vast of your questions about coding.

Last word: Hope you could find some helpful articles in this paper and always notice that you would reach your goals if you insist, So never stop hard working and believing yourself.

Kamyab Safaie **November 2021**