Assignment 2

$3x_1 + 4x_2 + 3x_3 = 10$
$x_1 + 5x_2 - x_3 = 7$
$6x_1 + 3x_2 + 7x_3 = 15$

1) Solve with Naive Gaussian Elimination

$$\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 1 & 5 & -1 & 7 \\ 6 & 3 & 7 & 15 \end{array}\right]$$

Forward elimination $(n-1)$ steps so 2 steps

Step 1 (zeros below a_{11})

$-\frac{1}{3}R_1 + R_2 = R_2$

$-\frac{6}{3}R_1 + R_3 = R_3$

$$\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 0 & 3\frac{2}{3} & -2 & 3\frac{2}{3} \\ 0 & -5 & 1 & -5 \end{array}\right]$$

Step 2 (zeros below a_{22})

$-\frac{-5}{3\frac{2}{3}}R_2 + R_3 = R_3$

$$\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 0 & 5\frac{2}{3} & -\underline{2} & 3\frac{2}{3} \\ 0 & 0 & 1,\overline{36} & 0 \end{array}\right]$$

Back substitution

$x_3 = \frac{b_3}{a_{33}} \quad x_3 = 0$

$x_2 = \frac{b_2 - (a_{23} \cdot x_3)}{a_{22}} = \frac{3^2/_3 - (0)}{3^2/_3}$

$x_2 = 1$

$x_1 = \frac{b_1 - ((a_{13} \cdot x_3) + (a_{12} \cdot x_2))}{a_{11}}$

$x_1 = \frac{10 - (0 + 4)}{3} = \frac{6}{3}$

$x_1 = 1$

Final matrix

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

$x_1 = 2$
$x_2 = 1$
$x_3 = 0$

Assignment 2 continued

2) $\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 1 & 5 & -1 & 7 \\ 6 & 3 & 7 & 15 \end{array}\right]$ scaling $S = [4, 5, 7]$

column 1 $= [3/4, 1/5, 6/7]$

$R_3 \longleftrightarrow R_1$

$\left[\begin{array}{ccc|c} 6 & 3 & 7 & 15 \\ 1 & 5 & -1 & 7 \\ 3 & 4 & 3 & 10 \end{array}\right]$ Forward elimination step 1

$-\frac{1}{6} R_1 + R_2 \rightarrow R_2$

$-\frac{3}{6} R_1 + R_3 \rightarrow R_3$

$\left[\begin{array}{ccc|c} 6 & 3 & 7 & 15 \\ 0 & 4\frac{1}{2} & -2\frac{1}{6} & 4\frac{1}{2} \\ 0 & 2\frac{1}{2} & -\frac{1}{2} & 2\frac{1}{2} \end{array}\right]$ scaling col 2 $[4.5/5, 2.5/7]$

leave as is

$-\frac{2.5}{4.5} R_2 + R_3 \rightarrow R_3$

$\left[\begin{array}{ccc|c} 6 & 3 & 7 & 15 \\ 0 & 4\frac{1}{2} & -2\frac{1}{6} & 4\frac{1}{2} \\ 0 & 0 & 0.703 & 0 \end{array}\right]$ Back substitution

$x_3 = 0$

$$x_2 = \frac{b_2 - (a_{23} \cdot x_3)}{a_{22}}$$

$$x_2 = \frac{4\frac{1}{2} - 0}{4\frac{1}{2}} \quad x_2 = 1$$

$$x_1 = \frac{b_1 - ((a_{13} \cdot x_3) + (a_{12} \cdot x_2))}{a_{11}}$$

$$x_1 = \frac{15 - (0 + 3)}{6}$$

$$x_1 = 2$$

Final matrix

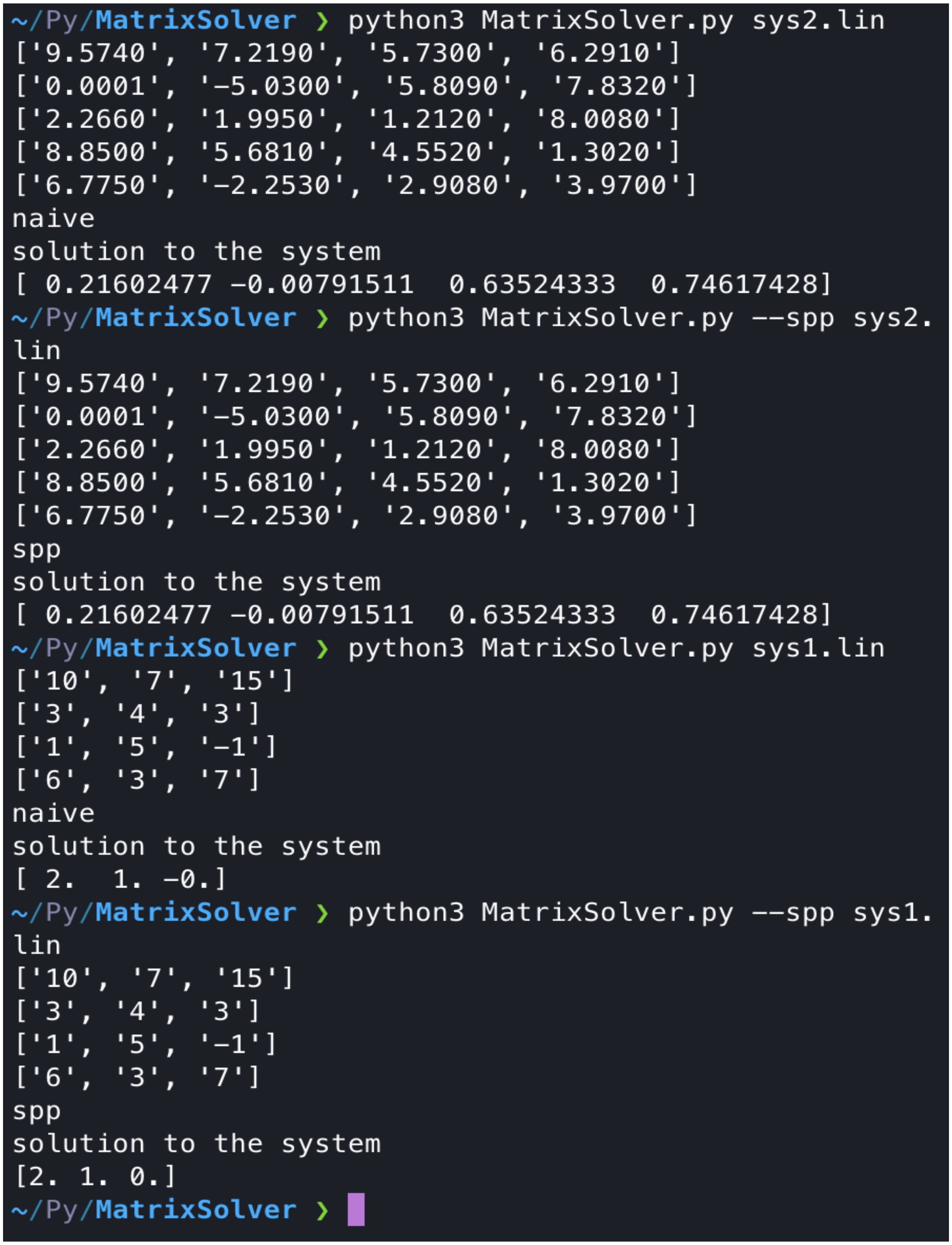
$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right]$

$x_1 = 2$

$x_2 = 1$

$x_3 = 0$

image showing both NGE and SPP work with the outputs. Works writing to files too



```
~/Py/MatrixSolver ❯ python3 MatrixSolver.py sys2.lin
['9.5740', '7.2190', '5.7300', '6.2910']
['0.0001', '-5.0300', '5.8090', '7.8320']
['2.2660', '1.9950', '1.2120', '8.0080']
['8.8500', '5.6810', '4.5520', '1.3020']
['6.7750', '-2.2530', '2.9080', '3.9700']
naive
solution to the system
[ 0.21602477 -0.00791511  0.63524333  0.74617428]
~/Py/MatrixSolver ❯ python3 MatrixSolver.py --spp sys2.
lin
['9.5740', '7.2190', '5.7300', '6.2910']
['0.0001', '-5.0300', '5.8090', '7.8320']
['2.2660', '1.9950', '1.2120', '8.0080']
['8.8500', '5.6810', '4.5520', '1.3020']
['6.7750', '-2.2530', '2.9080', '3.9700']
spp
solution to the system
[ 0.21602477 -0.00791511  0.63524333  0.74617428]
~/Py/MatrixSolver ❯ python3 MatrixSolver.py sys1.lin
['10', '7', '15']
['3', '4', '3']
['1', '5', '-1']
['6', '3', '7']
naive
solution to the system
[ 2.  1. -0.]
~/Py/MatrixSolver ❯ python3 MatrixSolver.py --spp sys1.
lin
['10', '7', '15']
['3', '4', '3']
['1', '5', '-1']
['6', '3', '7']
spp
solution to the system
[2. 1. 0.]
~/Py/MatrixSolver ❯
```

Discussion:

The two different algorithms for solving systems of linear equations have different sets of pros and cons to them. When it comes to the naive Gaussian elimination the perks of it are that it is faster than scaled partial pivot gaussian elimination. It is able to be more memory and time-efficient by using fewer arrays and fewer loops. The downside is that this method can fall victim to division by zero issues sometimes. The scaled partial pivot Gaussian elimination method is good because it protects us from running into division by zero errors. The pivoting prevent that error from happening. This is the trade-off, overall it's better to use the scaled partial pivot method because the time loss should not be that bad, especially compared to the trade-off of the system not getting solved due to division by zero errors.