Exercise 1, 2:

Naive

$$\left[\begin{array}{cccc} 3 & 4 & 3 & 10 \\ 1 & 5 & -1 & 7 \\ 6 & 3 & 7 & 15 \end{array}\right] \times \frac{1}{3} \quad \times 2$$

$$\left[\begin{array}{cccc} 3 & 4 & 3 & 10 \\ 0 & \frac{11}{3} & -2 & \frac{11}{3} \\ 0 & -5 & 1 & -5 \end{array}\right] \times -\frac{15}{11}$$

$$\left[\begin{array}{cccc} 3 & 4 & 3 & 10 \\ 0 & \frac{11}{3} & -2 & \frac{11}{3} \\ 0 & 0 & -\frac{19}{11} & 0 \end{array}\right]$$

Back substitutio.

$-\frac{19}{11} x_3 = 0$ $\qquad$ $\frac{11}{3} x_2 + 2(0) = \frac{11}{3}$ $\qquad$ $3x_1 + 4(1) + 3(0) = 10$

$\boxed{x_3 = 0}$ $\qquad$ $\boxed{x_2 = 1}$ $\qquad$ $\boxed{x_1 = 2}$

Scaled Partial Pivoting

$$\begin{array}{c|cccc} 3/4 & 3 & 4 & 3 & 10 \\ 1/5 & 1 & 5 & -1 & 7 \\ \boxed{6/7} & 6 & 3 & 7 & 15 \end{array} \quad 1/6 \quad 1/2$$

$$SF = \begin{bmatrix} 4 \\ 5 \\ 7 \end{bmatrix}$$

$$\rightarrow \left[\begin{array}{cccc} 0 & 5/2 & -\frac{1}{2} & 5/2 \\ 0 & 9/2 & -\frac{13}{6} & 9/2 \\ 6 & 3 & 7 & 15 \end{array}\right] \frac{5}{9}$$

$$\left[\begin{array}{cccc} 0 & 0 & \frac{19}{27} & 0 \\ 0 & 9/2 & -\frac{13}{6} & 9/2 \\ 6 & 3 & 7 & 15 \end{array}\right]$$

Back substitution

$\frac{19}{27} x_3 = 0$

$\boxed{x_3 = 0}$

$\frac{9}{2} x_2 = \frac{9}{2}$

$\boxed{x_2 = 1}$

$6x_1 + 3 = 15$

$\boxed{x_1 = 2}$

Exercise 3, 4:

Using my own `solve.bash` script, I generate this output. Your runtime may vary.

```
❯ ./solve.bash
data/sys1.lin solutions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Naive Gaussian Elimination
Single Precision

Solution: 0.21457672 -0.013570835 0.630871 0.745785
Runtime (nanoseconds): 3833
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Naive Gaussian Elimination
Double Precision

Solution: 0.21602476699023043 -0.00791510608732474 0.6352433264885665
0.7461742760893735
Runtime (nanoseconds): 3958
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Scaled Partial Pivoting
Single Precision

Solution: 0.21602473 -0.00791517 0.6352434 0.7461743
Runtime (nanoseconds): 7500
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Scaled Partial Pivoting
Double Precision

Solution: 0.2160247670084124 -0.007915106087778234 0.6352433264931057
0.7461742760857156
Runtime (nanoseconds): 7291
Successfully solved data/sys1.lin. Solutions stored in  data/.
```

The overall precision of all methods were decent. The only solution that was significantly off was the Naive algorithm using single point precision. This is because the naive algorithm performs an addition/subtraction operation on 2 numbers that are sufficiently far apart. This likely causes loss of significance in all cases, but this is amplified by the single point precision.

Runtime analysis

Using the `generate_rand_system.py` script, I am able to generate a random solvable system of linear equations of arbitrary size. I use this to measure how the running time of the naive vs. spp grows as *n* grows.

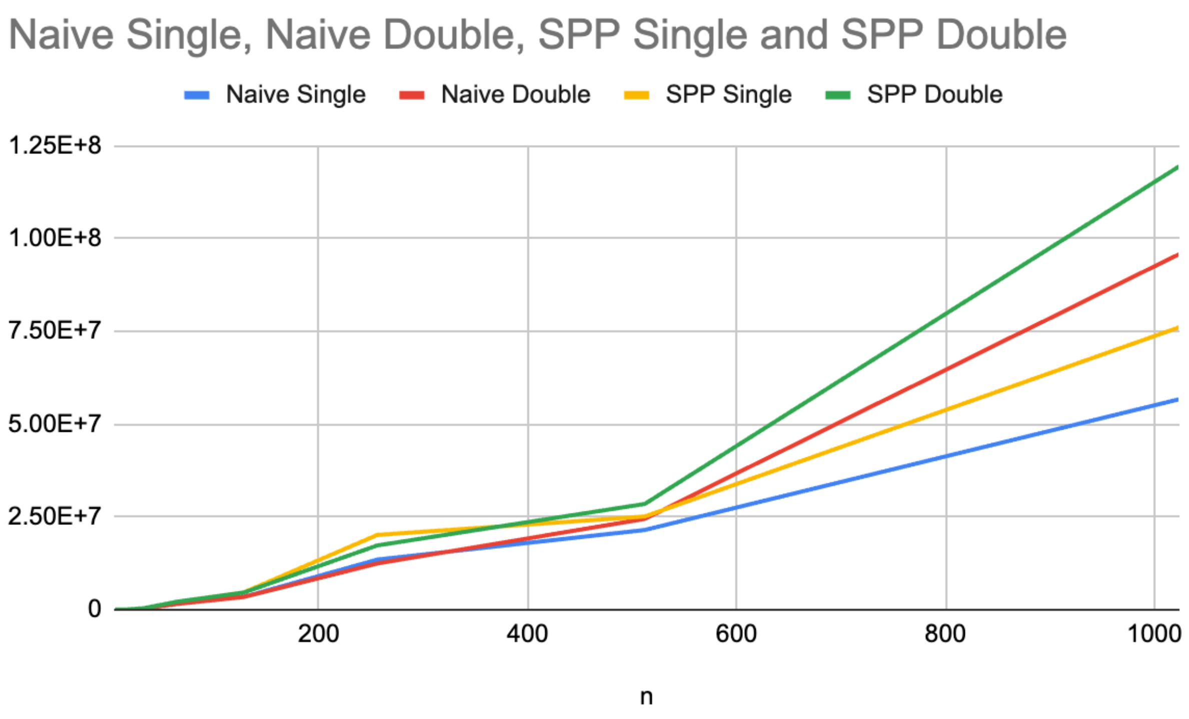
```
❯ python3 generate_rand_system.py 4
❯ ./solve.bash sys_rand | grep "Runtime"
Runtime (nanoseconds): 3959
Runtime (nanoseconds): 3833
Runtime (nanoseconds): 7375
Runtime (nanoseconds): 7333
❯ python3 generate_rand_system.py 8
❯ ./solve.bash sys_rand | grep "Runtime"
Runtime (nanoseconds): 8500
Runtime (nanoseconds): 8584
Runtime (nanoseconds): 17250
Runtime (nanoseconds): 16583
…
```

n	Naive Single	Naive Double	SPP Single	SPP Double
4	3959	3833	7375	7333
8	8500	8584	17250	16583
16	38458	38583	65917	66125
32	251667	252042	380083	380958
64	1609041	1543834	2147583	2152834
128	3431500	3423958	4655167	4614500
256	13516959	12457750	20161791	17329583
512	21484125	24473458	25119458	28495209
1024	56733375	95859750	76124375	119517958

This data clearly shows that the running time of the 4 algorithms can be ranked from fastest to slowest:

1. Naive Single
2. Naive Double
3. SPP Single
4. SPP Double

The data also supports our conclusion that SPP does not affect the Big Oh of our Gaussian Elimination algorithm. Although the running time does increase, the order of growth seems to be unaffected.


Naive Single, Naive Double, SPP Single and SPP Double
Naive Single
Naive Double
SPP Single
SPP Double
1.25E+8
1.00E+8
7.50E+7
5.00E+7
2.50E+7
0
200
400
600
800
1000
n