EX.2.1.3, Sauer3

Solve by back substitution:

a.

$$\begin{aligned} 3x - 4y + 5z &= 2 \\ 3y - 4z &= -1 \\ 5z &= 5 \end{aligned}$$

b.

$$\begin{aligned} x - 2y + z &= 2 \\ 4y - 3z &= 1 \\ -3z &= 3 \end{aligned}$$

EX.2.1.3, Sauer3, solution, Langou

See Colab `https://colab.research.google.com/drive/1FHXErhF1TkEg-_iF9EHotY2mhI3c49Do`

a. We have $(x, y, z) = (1/3, 1, 1)$.

b. We have $(x, y, z) = (2, -1/2, -1)$.

```
# (a)
#                    3x - 4y + 5z =  2
#                         3y - 4z = -1
#                              5z =  5

A = np.array([
                  [  3., -4.,  5. ],
                  [  0.,  3., -4. ],
                  [  0.,  0.,  5. ]])

b = np.array([
                  [  2.],
                  [ -1.],
                  [  5.]])
```

```
# solve Ax=b with np.linalg.solve
# and check that Ax is b

x = np.linalg.solve(A, b)
print("With np.linalg.solve, we find x = \n", x)

print("\nWe check that Ax is b. Indeed [ Ax, b ] =\n",\
```

```
 np.concatenate((A@x, b), axis=1) )
```

```
With np.linalg.solve, we find x =
 [[0.33333333]
 [1.        ]
 [1.        ]]

We check that Ax is b. Indeed [ Ax, b ] =
 [[ 2.  2.]
 [-1. -1.]
 [ 5.  5.]]
```

```
# use our bucket algorithm
#   backward_substitution
# for the solve
x = backward_substitution( A, b )
print("\nx=\n", x )
```

```
x=
 [[0.33333333]
 [1.        ]
 [1.        ]]
```

```
# (b)
#                     x - 2y +  z =  2
#                         4y - 3z =  1
#                             -3z =  3

A = np.array([
                  [  1., -2.,  1. ],
                  [  0.,  4., -3. ],
                  [  0.,  0., -3. ]])

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

```
# solve Ax=b with np.linalg.solve
# and check that Ax is b

x = np.linalg.solve(A, b)
print("\nWith np.linalg.solve, we find x = \n", x)

print("\nWe check that Ax is b. Indeed [ Ax, b ] =\n",\
 np.concatenate((A@x, b), axis=1) )
```

```
With np.linalg.solve, we find x =
 [[ 2. ]
 [-0.5]
 [-1. ]]
```

```
We check that Ax is b. Indeed [ Ax, b ] =
 [[2. 2.]
 [1. 1.]
 [3. 3.]]
```

```
# use our bucket algorithm
#   backward_substitution
# for the solve
x = backward_substitution( A, b )
print("x=\n", x )
```

```
x=
 [[ 2. ]
 [-0.5]
 [-1. ]]
```