EX.2.1.2.a, Sauer3

Use Gaussian elimination to solve the systems:

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

EX.2.1.2.a, Sauer3, solution, Langou

Part (a)
Colab URL: `https://colab.research.google.com/drive/15nTh989L747L9p3OHsf6JvBSG72ghGPu`

$$\left(\begin{array}{rrr|r} 2 & -2 & -1 & -2 \\ 4 & 1 & -2 & 1 \\ -2 & 1 & -1 & -3 \end{array}\right) \begin{array}{c} L_1 \leftarrow L_1 - 2L_0 \\ L_2 \leftarrow L_1 + L_0 \\ \rightsquigarrow \end{array} \left(\begin{array}{rrr|r} 2 & -2 & -1 & -2 \\ 0 & 5 & 0 & 5 \\ 0 & -1 & -2 & -5 \end{array}\right) \begin{array}{c} L_2 \leftarrow L_2 + L_1/5 \\ \rightsquigarrow \end{array} \left(\begin{array}{rrr|r} 2 & -2 & -1 & -2 \\ 0 & 5 & 0 & 5 \\ 0 & 0 & -2 & -4 \end{array}\right)$$

Now we are ready for backsubstitution.

Row 3 reads $-2z = -4$, therefore $z = 2$.
Row 2 reads $5y = 5$, therefore $y = 1$.
Row 1 reads $2x - 2y - z = -2$ and so substituing $y = 1$ and $z = 2$, we find $x = 1$. The solution is

$$x = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$$

```python
import numpy as np

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

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(x)

print("\nA x = \n", A@x )
```

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

A x =
 [[-2.]
 [ 1.]
 [-3.]]
```

```
# concatenate A and b to form the augmented matrix Z

Z = np.concatenate((A, b), axis=1)

print("\n",Z)
```

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

```
# perform Gaussian elimination on Z by ``hand``
Z[1,:] = Z[1,:] - 2. * Z[0,:]
Z[2,:] = Z[2,:] + Z[0,:]
print("\n",Z)
```

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

```
Z[2,:] = Z[2,:] + Z[1,:] / 5.
print("\n",Z)
```

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

```
# use our bucket algorithm
#   gaussian_elimination__section_2_1
# to compute the equivalent triangular system obtained
# after Gausssian elimination
print("After Gaussian elimination, triangular system is:\n",\
 gaussian_elimination__section_2_1( A, b )[0])
print("and the right-hand side is:\n",\
 gaussian_elimination__section_2_1( A, b )[1] )
```

```
After Gaussian elimination, triangular system is:
 [[ 2. -2. -1.]
 [ 0.  5.  0.]
 [ 0.  0. -2.]]
and the right-hand side is:
 [[-2.]
 [ 5.]
 [-4.]]
```

```python
# use our bucket algorithms
#   gaussian_elimination__section_2_1
#   and backward_substitution
# for the solve
x = backward_substitution( *gaussian_elimination__section_2_1( A, b ) )
print("\nx=\n", x )
```

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