EX.4.1.2.b, Sauer3

Find the least squares solutions and RMSE of the following systems:

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \end{bmatrix}.$$

EX.4.1.2.b, Sauer3, solution, Langou

Colab: `https://colab.research.google.com/drive/19z_bgBQjCaILOA_0w3p6_K4Dg8edxQRm`

First we form

$$A^T A = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 3 & 6 \\ 3 & 2 & 2 \\ 6 & 2 & 7 \end{bmatrix}.$$

and

$$A^T b = \begin{bmatrix} 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 3 \\ 11 \end{bmatrix}.$$

Now we solve $(A^T A)x = (A^T b)$. We get the least squares solution as

$$\begin{bmatrix} 7 & 3 & 6 \\ 3 & 2 & 2 \\ 6 & 2 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 3 \\ 11 \end{bmatrix} \iff \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

We can compute the residual

$$r = b - Ax = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

This means that $b \in \text{Range}(A)$. We also have

$$\text{RMSE} = 0.$$

Python helper notebook:

In python, we can either use Numpy's `np.linalg.lstsq`:

```
import numpy as np
from math import sqrt

A = np.array([[1.,0.,1.],[1.,0.,2.],[1.,1.,1.],[2.,1.,1.]])
b = np.array([[2.],[3.],[1.],[2.]])
x = np.linalg.lstsq(A,b,rcond=None)[0]
```

```python
print("x =\n",x)
print("[ Ax, b ]=\n",np.concatenate((A@x,b),axis=1))
r = b - A@x
print("r =\n",r)
print("|| b - Ax ||_2 = ", f"{np.linalg.norm( r, 2 ):.4f}")
print("RMSE = ", f"{np.linalg.norm( r, 2 ) / sqrt( A.shape[0] ) :.4f}")
print("|| A^T ( b - Ax ) ||_oo =",
      f"{np.linalg.norm( A.T@( A@x - b), np.infty ):.1e}" )
```

```
x =
 [[ 1.]
 [-1.]
 [ 1.]]
[ Ax, b ]=
 [[2. 2.]
 [3. 3.]
 [1. 1.]
 [2. 2.]]
r =
 [[ 6.66133815e-16]
 [ 8.88178420e-16]
 [-4.44089210e-16]
 [ 0.00000000e+00]]
|| b - Ax ||_2 =  0.0000
RMSE =  0.0000
|| A^T ( b - Ax ) ||_oo = 2.0e-15
```

or we can do it ourselves using the Normal Equation methods `x = np.linalg.solve( A.T@A, A.T@b )`.