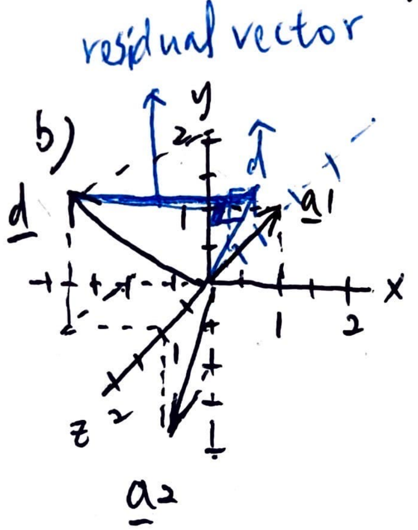
Activity 6

△ Item 1

a) $\underline{A}\underline{w} \simeq \underline{d}$

$$w = (\underline{A}^T\underline{A})^{-1}\underline{A}^T\underline{d}$$

b)



△ Item 3 e)

$$w_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad w_2 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$A = TW^T, \quad x = W\tilde{x}$$

$$\underset{3\times 2}{W} \quad \underset{2\times 1}{\tilde{x}}$$

$$Ax = b = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 2 \end{bmatrix}$$

$$\underset{4\times 3}{\begin{bmatrix} 1 & -1/2 & 1 \\ 0 & 3/2 & 0 \\ 0 & 3/2 & 0 \\ 1 & -1/2 & 1 \end{bmatrix}} \underset{3\times 1}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}$$

$$x_1 - \frac{x_2}{2} + x_3 = 2$$

$$3/2\, x_2 = 1 \rightarrow \boxed{x_2 = \frac{2}{3}}$$

$$x_1 + x_3 = \frac{7}{3}$$

$$\begin{array}{cc} w_1 & w_2 \end{array}$$
$$a\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + b\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 \\ 2/3 \\ x_3 \end{bmatrix}$$

$$a - 2b = 2/3$$

$$2(a+b) = 7/3 \Rightarrow 3a = 2/3 + \frac{7}{3} = 3$$

$$\rightarrow a + b = 7/6$$

$$a = 1$$
$$b = 1/6$$

$$x_1 = 7/6$$

$$x_3 = \frac{7}{3} - \frac{7}{6} = \frac{7}{6}$$

sol. $x = \begin{bmatrix} 7/6 \\ 2/3 \\ 7/6 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -2 \\ 1 & 1 \end{bmatrix} \tilde{x} \begin{bmatrix} a \\ b \end{bmatrix}$

$$\rightarrow 7/6 = a + b$$
$$2/3 = a - 2b$$

In [20]:
```
import numpy as np
# A = np.array([[25,0,1],[20,1,2],[40,1,6]])
# b = np.array([[110],[110],[210]])

# To see rank, use:
# np.linalg.matrix_rank(A)

# To invert a matrix, use:
# np.linalg.inv(A)

# To transpose a matrix, use:
# M.transpose()

# To concatenate matrices by column(0) or by row(1)
# np.concatenate((M1, M2), axis = 0)

# To multiply matrices in Python 3, use:
# A@B
```

In [21]:
```
import numpy as np
A = np.array([[25,15,10,0,1],[20,12,8,1,2],[40,30,10,1,6], [30,15,15,0,3], [35,20,15,2,4
b = np.array([[104],[97],[193],[132],[174]])

print(np.linalg.matrix_rank(A))

# Note: you can use np.hstack() to concatinate vectors, for example np.hstack((A,b))

# Note: you can select all the columns, except the first of a matrix A as:  A[:,1:]
```

```
4
```

In [22]:
```
# Item1 a)
A = np.array([[1, 0],[1, -1],[0, 1]])
d = np.array([[-1],[2],[1]])
# w = (A_T@A)(-1)@A_T@d
A_T = A.transpose()
M1 = A_T @ A #2x3, 3x2
M2 = np.linalg.inv(M1)
M3 = M2 @ A_T # 2x2, 2x3
w = M3 @ d #2x3, 3x1
w
```

Out[22]:
```
array([[ 0.33333333],
       [-0.33333333]])
```

In [23]:
```
d_hat = A @ w
d_hat
```

Out[23]:
```
array([[ 0.33333333],
       [ 0.66666667],
       [-0.33333333]])
```

In [24]:
```
# Item2 a)
A = np.array([[25, 0, 1],[20, 1, 2],[40, 1, 6]])
b = np.array([[110],[110],[210]])
A_rank = np.linalg.matrix_rank(A) # 3
# x = A(-1) @ b
A_inv = np.linalg.inv(A)
x = A_inv @ b
x
```

Out[24]:
```
array([[ 4.25],
       [17.5 ],
       [ 3.75]])
```

In [25]:
```
# Item2 b)
A = np.array([[25, 15, 10, 0, 1],[20, 12, 8, 1, 2],[40, 30, 10, 1, 6],[30, 15, 15, 0, 3]
b = np.array([[104],[97],[193],[132],[174]])
A_b = np.concatenate((A, b), axis = 1)
A_rank = np.linalg.matrix_rank(A) # 4
print(A_rank)
A_b_rank = np.linalg.matrix_rank(A_b) # 4
print(A_b_rank)
# x = A(-1) @ b
# A_inv = np.linalg.inv(A) #--> LinAlgError: Singular matrix
```

```
4
4
```

i) Does an exact solution exist? Why or why not?

Yes, because the rank of A is equal to the rank of A combined with b. Here is 4.

ii) Does a unique solution exist? Why or why not?

No, a unique solution does not exist, because A does not have an inverse matrix.

iii) Suppose you ignore (remove) the total carbohydrates per serving (first column of A). Find a unique solution to the modified least-squares problem and the resulting squared error.

In [26]:
```
# a unique solution
A = np.array([[15, 10, 0, 1],[12, 8, 1, 2],[30, 10, 1, 6],[15, 15, 0, 3],[20, 15, 2, 4]]
b = np.array([[104],[97],[193],[132],[174]])
A_b = np.concatenate((A, b), axis = 1)
A_rank = np.linalg.matrix_rank(A) # 4
print(A_rank)
A_b_rank = np.linalg.matrix_rank(A_b) # 4
print(A_b_rank)
# x = A(-1) @ b --> can't be used, since A is not a squred matrix.
# x = (A_T@A)(-1)@A_T@d
A_T = A.transpose() # 4x5
M1 = A_T @ A #4x5, 5x4
M2 = np.linalg.inv(M1)
M3 = M2 @ A_T # 4x4, 4x5
x = M3 @ b #4x5, 5x1
x
```

```
4
4
```

Out[26]:
```
array([[4.],
       [4.],
       [9.],
       [4.]])
```

In [27]:
```
# resulting squared error.
b_hat = A @ x
print(A)
print(x)
print(b)
print(b_hat)
b_hat = np.array(b_hat, dtype = np.int16)
np.subtract(b, b_hat)
```

```
[[15 10  0  1]
 [12  8  1  2]
 [30 10  1  6]
 [15 15  0  3]
 [20 15  2  4]]
[[4.]
```

```
 [4.]
 [9.]
 [4.]]
[[104]
 [ 97]
 [193]
 [132]
 [174]]
[[104.]
 [ 97.]
 [193.]
 [132.]
 [174.]]
```

Out[27]:
```
array([[0],
       [0],
       [0],
       [0],
       [0]])
```

In [28]:
```python
# Item3 a)
# A = T @ W_T
t1 = np.array([[0.5],[0.5],[0.5],[0.5]])
t2 = np.array([[0.5],[-0.5],[-0.5],[0.5]])
w1 = np.array([[1],[1],[1]])
w2 = np.array([[1],[-2],[1]])
w1_T = w1.transpose()
# print(w1_T)
w2_T = w2.transpose()
# print(w2_T)
# T = np.array([[t1, t2]])
T = np.concatenate((t1, t2), axis = 1)
# print(T)
W_T= np.concatenate((w1_T, w2_T), axis = 0)
# print(W_T)
# W = W_T.transpose()
# print(W)
A = T @ W_T
print(A)
A_rank = np.linalg.matrix_rank(A)
A_rank # 2
```

```
[[ 1.  -0.5  1. ]
 [ 0.   1.5  0. ]
 [ 0.   1.5  0. ]
 [ 1.  -0.5  1. ]]
```

Out[28]: 2

In [29]:
```python
# Item3 b)
T_rank = np.linalg.matrix_rank(T)
T_rank # 2
```

Out[29]: 2

In [30]:
```python
# Item3 c)
A_T = A.transpose()
Q = A_T @ A
# print(Q) # 3x3
Q_rank = np.linalg.matrix_rank(Q)
Q_rank # 2
# Q_inv = np.linalg.inv(Q) #--> LinAlgError: Singular matrix
```

Out[30]: 2

Q is not positive definite because its rank is less than its column number, making it non-invertible.

Item3 d)

The solution can be calculated by inv(A_T @ A) @ A_T @ y, where (A_T @ A) is not invertible.Therefore, there is no solution.

```
In [ ]:
```