Ridge Regression and the Minimum Norm Solution

Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Assume you are given the regularized least squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \tfrac{\delta}{2}\|x\|_2^2.$$

1. Which equation does the solution x_δ of the above least squares problem solve?
2. Assume you are given the following data

z	-1	1
y	2	-1

   which you want to explain by a model $f : \mathbb{R} \to \mathbb{R}$ of the form

   $$f_c(z) = c_1 + c_2 z + c_3 z^2.$$

   Implement a program to solve the regularized least squares problem

   $$\min_{c \in \mathbb{R}^3} \sum_{i=1}^{2} (f_c(z_i) - y_i)^2 + \tfrac{\delta}{2} \sum_{j=1}^{3} c_j^2$$

   to determine appropriate coefficients $x_\delta = (c_1^\delta, c_2^\delta, c_3^\delta)$ for *multiple* $\delta \in (0,1)$. Why is regularization an appropriate approach here?
3. Find a routine to compute the minimum norm least squares solution x^+ and compare it to your solutions x_δ. What do you observe for small δ?
4. Plot the measurements and the fitting polynomial corresponding to x_δ and x^+ into one figure. What do you observe for small δ?

   Hint: Use

   - scipy.linalg.solve to solve a linear system and set the correct flag that informs the function about the positivity of the matrix (see documentation),
   - numpy.linspace to create an array of multiple $\delta \in (0,1)$,
   - the plot routines from previous exercises if you want.

Solution:

```python
import numpy as np
import scipy.linalg as linalg
import matplotlib.pyplot as plt

if __name__ == "__main__":

    # given data
    data = np.array([[-1., 1],
                     [2, -1]])

    # Assembly
    # design matrix
    p = [0,1,2]
    z_i = data[0,:]
    z_i = z_i[np.newaxis].T
    A = z_i**p
```

```python
# right-hand side
b = data[1,:]

# minimum norm least squares solution
x_0 = linalg.lstsq(A,b)[0] # note that lstsq uses the svd

# create an array of delta's in (0,1)
num_delta = 3
Delta = np.linspace(0.0001,1, num_delta, endpoint=False)

# loop over delta's from large to small
print("del \t x_del  \t\t\t\t\t    ||x_del-x0|| ||x|| \t  ||Ax-b||")
for delta in Delta[::-1]:

    # solve regularized least squares problem (reg. normal equation)
    A_delta = A.T@A + delta * np.eye(len(A.T@A))
    x_delta = linalg.solve(A_delta, A.T@b, assume_a='pos')

    print(np.round(delta, 4), "\t",
          np.round(x_delta, 6), "\t",
          np.round(linalg.norm(x_delta-x_0), 6), "\t",
          np.round(linalg.norm(x_delta), 6), "\t ",
          np.round(linalg.norm(A@x_delta - b), 6))

    # Plot measurements and fitting curve
    plt.figure()
    plt.title("Polynomial degree = "+str(max(p))+", delta = "+str(np.round(delta,4)))
    # mesaurements
    plt.plot(z_i, b, 'ro')
    # curve
    X = np.linspace(-2,2, 50)
    plt.plot(X, x_delta[0] + x_delta[1]*X + x_delta[2]*(X**2), 'b')
    plt.plot(X, x_0[0] + x_0[1]*X + x_0[2]*(X**2), 'cyan', linestyle = "--")
    plt.legend(["Samples", "Regularized Sol", "MinNorm Sol"])
    plt.show()
```