Math 271.1: Exercise 2 (#6)

INSTRUCTION: Tikhonov Regularization for an Ill-posed Example

(a) Compute the direct (unregularized) solutions
(b) the Tikhonov-regularized solutions with fixed parameter λ= 10−3
(c) For both the direct and regularized cases, compute and report

Importing Libraries needed

- numpy for numerical computations
- matplotlib for image display

```
import numpy as np
import matplotlib.pyplot as plt
```

Initializing the 3x3 Hilbert Matrix A and Right-hand vectors b, delta_b and b_pert

- Original: Ax = b
- Perturbed: Ax = b_pert

```
A = np.array([
    [1, 1/2, 1/3],
    [1/2, 1/3, 1/4],
    [1/3, 1/4, 1/5]
])

print("Hilbert Matrix A:\n", A)

b = np.array([1.0000, 0.8000, 0.6000])
delta_b = np.array([0.1, -0.1, 0])
b_pert = b + delta_b

print("\nVector b: ", b)
print("\nPerturbed: ", b_pert)
```

```
Hilbert Matrix A:
 [[1.         0.5        0.33333333]
 [0.5        0.33333333 0.25      ]
 [0.33333333 0.25       0.2       ]]

Vector b:  [1.  0.8 0.6]

Perturbed:  [1.1 0.7 0.6]
```

Compute Condition Number

- $\kappa(A) = ||A|| \cdot ||A^{-1}|| \approx 524$

```
print(f"Condition number: {np.linalg.cond(A):.2e}")
```

```
Condition number: 5.24e+02
```

(a) Direct (Unregularized) Solution

- 1. Solves `Ax = b` using LU decomposition (standard method)
- 2. Solves `Ax = b_pert` with perturbed right-hand side
- 3. Measures **2-norm distance** between solutions
- Shows the instability of the direct solution

```
# Direct solutions
x_direct = np.linalg.solve(A, b)
x_pert_direct = np.linalg.solve(A, b_pert)
delta_direct = np.linalg.norm(x_direct - x_pert_direct, 2)

print(f"\nDirect solution: {x_direct}")
```

```
print(f"Direct perturbed: {x_pert_direct}")
print(f"Δ_direct = {delta_direct:.6f}")
```

```
Direct solution: [-1.8  9.6 -6. ]
Direct perturbed: [  2.7 -13.2  15. ]
Δ_direct = 31.322356
```

(b) Construct Regularized System

- A^TA is even more ill-conditioned than A (condition number squared!). Hence, we need regularization
- Original problem: minimize $||Ax - b||^2$
- Normal equations: $A^TAx = A^Tb$
- These computations set up the regularization

```
# Regularized solutions
lambda_val = 1e-3

# Constructing normal equations: (A^T A + λI)x = A^T b
ATA = A.T @ A
ATb = A.T @ b
ATb_pert = A.T @ b_pert

print(f"\nA^T A:\n{ATA}")
print(f"\nA^T B:\n{ATb}")
print(f"\nA^T B Pert:\n{ATb_pert}")
```

```
A^T A:
[[1.36111111 0.75       0.525     ]
 [0.75       0.42361111 0.3       ]
 [0.525      0.3        0.21361111]]

A^T B:
[1.6        0.91666667 0.65333333]

A^T B Pert:
[1.65       0.93333333 0.66166667]
```

Adding Regularization Term

```
ATA_reg = ATA + lambda_val * np.eye(A.shape[1])

print(ATA_reg)
```

```
[[1.36211111 0.75       0.525     ]
 [0.75       0.42461111 0.3       ]
 [0.525      0.3        0.21461111]]
```

Solving Regularized System

- Regularized solutions are much smaller in magnitude (shrinkage effect)
- Perturbation causes change of only 0.85 (vs 31.32 for direct!)
- Solutions are stable and usable

```
x_lambda = np.linalg.solve(ATA_reg, ATb)
x_pert_lambda = np.linalg.solve(ATA_reg, ATb_pert)
delta_reg = np.linalg.norm(x_lambda - x_pert_lambda, 2)

print(f"\nRegularized solution: {x_lambda}")
print(f"Regularized perturbed: {x_pert_lambda}")
print(f"Δ_reg = {delta_reg:.6f}")
```

```
Regularized solution: [-0.30495693  1.58177583  1.57914923]
Regularized perturbed: [0.19652049 0.99805374 1.20719419]
Δ_reg = 0.854729
```

- Regularized solutions are much smaller in magnitude (shrinkage effect)
- Perturbation causes change of only 0.85 (vs 31.32 for direct!)
- Solutions are stable and usable

Compare Sensitivity

- Small bias for **huge stability** gain:
  - Regularization slightly changes the solution but makes it **97% less sensitive** to noise
- Practical vs theoretical:
  - The direct solution is mathematically exact but numerically useless; regularized solution is approximate but actually works
- Real-world applicability:
  - With measurement noise, only the **regularized solution** provides meaningful results

```
print(f"\nSensitivity reduction: {delta_direct/delta_reg:.2f}x")
print(f"Regularization reduces sensitivity by {(1-delta_reg/delta_direct)*100:.1f}%")
```

```
Sensitivity reduction: 36.65x
Regularization reduces sensitivity by 97.3%
```

References:

[1] Tikhonov Regularization Theory: https://en.wikipedia.org/wiki/Tikhonov_regularization
[2] Hilbert Matrix Properties: https://en.wikipedia.org/wiki/Hilbert_matrix
[3] NumPy Linear Algebra Documentation: https://numpy.org/doc/stable/reference/routines.linalg.html
[4] Elements of Statistical Learning (Ch. 3.4): https://hastie.su.domains/ElemStatLearn/
[5] Interactive Regularization Visualization: https://mlu-explain.github.io/regularization/