Week 6 Hand-in

NumIntro 2019

Department of Computer Science
University of Copenhagen

Casper Lisager Frandsen <fsn483@alumni.ku.dk>

Version 2
Due: October 23rd, 08:00

Contents

1. Exercise

(a)

This function has been implemented in the "Q0039.py" file.

(b)

A formal requirement would be that the matrix is invertible. The matrix must also be sqaure, of the shape $(n \text{ x } n)$

(c)

While the power method is quick to compute, inverting a matrix is incredibly slow, on the order of $O(n^3)$. This means that as n increases in the $(n \text{ x } n)$ shaped matrix, the time (and number of operations) it takes to compute the inverse increases by a constant multiplied by n^3.

(d)

To find the condition number, we use the following formula:

$$\kappa(A) = \frac{|\text{max e.v of } A|}{|\text{min e.v of } A|}$$

We calculate the maximum and minimum eigenvalues of A using the power and inverse power methods implemented in the Q-files. With this we get:

$$\begin{aligned} |\text{max e.v of } A| &= 2.0005 \\ |\text{min e.v of } A| &= 0.0005 \\ \kappa(A) &= \frac{2.0005}{0.0005} \\ \kappa(A) &= 4001 \end{aligned}$$

Solving $Ax = b$ for $b = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ we get the vector:

$$x = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

Solving $Ax = b$ for $b = \begin{pmatrix} 2 \\ 2.001 \end{pmatrix}$ we get the vector:

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

This large change in the output makes sense, even with the extremely small change in the input, given our large condition number of 4001.

2. Exercise

(a)

This function has been implemented in the "Q0041.py" file.

(b)

To complete this exercise I have run the "pseudo_inverse_minimal_solution" function with the following inputs for subexercise *(a)*:

$$Ain = np.array([[1, 2, 4], [1, 1, 2]])$$
$$bin = np.array([2, 1])$$

This yields the following results:

$$\begin{bmatrix} 0 & 0.2 & 0.4 \end{bmatrix}$$

To complete this exercise I have run the "pseudo_inverse_minimal_solution" function with the following inputs for subexercise *(b)*:

$$Ain = np.array([[4, 6], [3, -2], [1, 3], [2, 6]])$$
$$bin = np.array([8, -7, 5, 10])$$

This yields the following results:

$$\begin{bmatrix} -1 & 2 \end{bmatrix}$$