FIT3139: Applied Questions for Week 4

Question 1

Consider the second-order difference equation with:

$$X_t = 3X_{t-1} - 2X_{t-2}, \qquad X_0 = 0,\ X_1 = 1$$

Write this system as a first-order matrix difference equation. Find the eigenvalues and eigenvectors of the system. Then, using eigen-decomposition, find an explicit solution to the original system.

Question 2

This question is about the Jacobi iteration. In matrix form, the iteration is given by:

$$\vec{x}^{(i)} = D^{-1}b + (-D^{-1}(L+R))\vec{x}^{(i-1)}$$

where D is a matrix that contains the diagonal elements of A, L contains the lower-left sub-diagonal part of A, and R contains the upper-right super-diagonal part.

Derive an element-wise expression for this iteration.

Adapt the code shown in the workshops to use this element-wise description of Jacobi iteration.

Verify your algorithm comparing a few inputs to results from a direct method. Make sure your algorithms handles problems that do not converge.

Test your code on some randomly generated matrices. Use this to estimate the probability of convergence for Jacobi iteration.

Question 3

Derive the Gauss-Seidel iteration from the Jacobi iteration. This idea was discussed in the Workshop last week, please watch the video or review your notes if you need a refresher.

Adapt your code from the previous question to implement the Gauss-Seidel method using an absolute error target.

Verify your algorithm comparing a few inputs to results from a direct method. Make sure your algorithms handles problems that do not converge.

Test your code on some randomly generated matrices. Use this to estimate the probability of convergence for Gauss-Seidel iteration. On average, which method converges more often? In the cases where both methods converge, which one requires fewer iterations on average?

Question 4

Implement and test the Power method to compute the dominant eigenvalue, eigenvector pair for a given matrix.

You can test your method on the following matrix:

$$A = \begin{pmatrix} 1 & 2 & 0 \\ -2 & 1 & 2 \\ 1 & 3 & 1 \end{pmatrix}$$

Starting with $x_0 = (1, 1, 1)$,

It may help to show each iteration, and the normalisation each round by dividing over the maximum element of the vector [1].

What is the dominant eigenvalue found through this method? How does this compare to Numpy's eigendecomposition function (numpy.linalg.eig)?

Given that the rate of convergence of the Power Method is the ratio between the two largest eigenvalues, at what rate will the method converge for the above matrix?

[1] That is, normalising over the infinity norm, $|\boldsymbol{x}|_\infty = \max_i |x_i|$