MT 3802 - Numerical Analysis:

Project on Iterative Methods

Programming language used: Python

Maria Ivanova

Overview

This project examines the basic properties of the three different iterative methods (Jacobi, Gauss-Seidel & Successive Relaxation). Two examples are analysed to discuss and compare the three methods and their rates of convergence.

Implementation & Design

For all three of the methods there are some key arrays which are defined before defining the main functions. These are as follows:

A – matrix a[i][j] given from the examples
b – vector given from the examples
N – number of iterations
x – solution vector
Y – a n array consisting of the diagonal values of A (function used from numpy package)
empty – empty 6x6 arrays for later use
D – diagonal matrix with elements a[i][i]
L – lower triangular matrix with l[i][j]=-a[i][j] for i>j and l[i][j]=0 elsewhere (for-loop going through A is used to fill in one of the empty arrays with the required values for a lower triangular matrix)
U – upper triangular matrix with u[i][j]=-a[i][j] for i<j and u[i][j]=0 elsewhere (for-loop going through A is used to fill in one of the empty arrays with the required values for an upper triangular matrix)
Result – L+U
Rho(1 to 3) - the spectral radiuses, calculated from the formula ρ(B)=max(| λ i|)for 1<=i<=n, where B is the Iteration matrix. The function for eigenvalues is from numpy.linalg.

<u>Jacobi Method</u>

A function 'Jacobi' with parameters A, b, N and x is defined.
It starts with an initial guess for the vector x (in these examples the initial guess will be the (0,0,0,0,0,0)). For every iteration the function calculates the solution vector from the formula given in the lecture notes, namely x(k+1) = D^{-1}*(L+U)*x(k)+D^{-1}*b.

The Iteration matrix is calculated from the formula in the lecture notes : $B_j=D^{-1}*(L+U)$. The matrix multiplication is done using the np.dot function from the numpy package.

The spectral radius for Jacobi is calculated.

Gauss-Seidel Method

A function 'GaussSeidel' with parameters A, b, DLInv, IterMatrixGauss, N and x is defined. It starts with an initial guess for x and for every iteration it calculates the new solution vector from the formula $x(k+1) = (D-L)^{-1}*U* x(k)+(D-L)^{-1}*b$. The variable DLInv is the calculated inverse of the matrix (D-L) and the Iteration matrix IterMatrixGauss is calculated from the formula $Bgs=(D-L)^{-1}*U$. The function np.linalg.inv calculating the inverse of a matrix is used from the numpy package.

The spectral radius for Gauss-Seidel is calculated.

Successive Relaxation Method

The successive relaxation method is used to accelerate the Gauss-Seidel method. In this project it is calculated in a different manner to the other two iteration methods. In order to be able to see the significance of the value of the relaxation parameter a for- loop is used to calculate the iterations for each different value of ω. The loop takes the values of ω in the range from 0 to 2 with a step = 0.05. A function frange is defined since the function range does not work with floats.

Formulas used:

$X(k+1)=(D-\omega L)^{-1}((1-\omega)D+\omega U)x(k)+(D-\omega L)^{-1}\omega b$.

$Bsor=(D-\omega L)^{-1}[(1-\omega)D+\omega U]$

The variable DwL is the matrix $(D - \omega L)$, while DwLInv is it's inverse.

The spectral radius for SOR is calculated.

The function SOR is defined to calculate the solution vector using the optimal value for ω and using the formula from the lecture notes.

Tests

The three algorithms were tested with the linear system from Example 2.3.1 from the lecture notes. All results were equal to these in the lecture notes.

Examples Analysis

Example 1:

Comment: In order to work with the linear system from example 1, the code for the values of A and b needs to be de-hashed.

N=100

X guess = (0,0,0,0,0,0)

As it can be observed, the values for the solution vector are very similar in each case – approx.(1,2,1,2,1,2). The spectral radiuses in each case, however, are quite different.

The spectral radius of Jacobi method equals approximately 0.923 while the one of the Gauss-Seidel method equals approximately 0.282.
The rate of convergence of the successive relaxation method appears to be very sensitive to the value of ω. From the values of ω examined, we can observe a fastest rate of convergence when ω=1.05, a value close to the one in the Gauss-Seidel method, but with spectral radius = approx. 0.152.
From these values, we can observe that the rate of convergence is fastest when using the successive relaxation method and slowest when using the Jacobi method.
Changing N:

In this example the Jacobi method is first most accurate when N=223, we get a solution vector [1.00000001 2.00000004 1.00000002 2.00000002 1.00000004 2.00000003]. The solution vector we get from Gauss-Seidel when N=16 is [1. 2.00000001 1.00000001 2. 0.99999999 1.99999999]. Thus, we need much less iterations with the Gauss-Seidel method than we do for the Jacobi method.

Example 2:

Comment: In order to work with the linear system from example 2, the code for the values of A and b of the previous example needs to be hashed, and the new values need to be de-hashed.
For:
N=100

As it can be observed, the values for the solution vector are very similar in each case – approx.([2. 3. 4. 2. 1. 2.]) The spectral radiuses in each case, however, are quite different.
The spectral radius of Jacobi method equals approximately 0.561 while the one of the Gauss-Seidel method equals approximately 0.314.
The rate of convergence of the successive relaxation method appears to be very sensitive to the value of ω. From the values of ω examined, we can observe a fastest rate of convergence when ω=1.1, a value close to the one in the Gauss-Seidel method, but with spectral radius = 0.1.
From these values, we can observe that the rate of convergence is fastest when using the successive relaxation method and slowest when using the Jacobi method.
Changing N:

In this example the Jacobi matrix is first most accurate when N=36, we get a solution vector: [2. 3. 4. 2. 1. 2.]. The same solution vector we get from Gauss-Seidel when N=19. Here, the Gauss-Seidel method needs less iterations than the Jacobi one, but this time the difference is not as big as in example 1.

Conclusion

From the analysed examples we can make the conclusion that the Jacobi Method has the slowest rage of convergence and it takes the most iterations to converge, while the Gauss-Seidel method had a faster rate of convergence and requires less iterations for greater accuracy. The successive relaxation method can be used to accelerate the Gauss-Seidel method even more, having the fastest rate of convergence.