Test title

Linus Falk

October 31, 2022

In this following problems we are examining the performance (iteration and run-time) of 4 algortihms for solving system of linear equations implemented in MATLAB: Jacobi method (jacobi.m), Gauss-Seidel method (gs.m), LU decomposition (Doolittle algorithm) (myownLU.m) and Conjugate gradient method (CG.m). The timing was done using the MATLAB script: Project_2B.m

Problem B1

In this problem we are solving a small 4x4 system of lineare equations

$$\mathbf{A} = \begin{bmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 6 \\ 25 \\ -11 \\ 15 \end{bmatrix} \tag{1}$$

Method	**Iteration**	**Time**
Jacobi	11	0.004164
Gauss-Seidel	6	0.004300
CG	9	0.000377
myownLU		0.009256
Ab		0.000056
Matlab LU		0.000062

Table 1: Problem B1

We can observe that the MATLAB backslash function is the fastest of the selection of algorithms. This is because the backslash function is not only one algorithm but instead work by first examining the matrix and select method that is best suited for the problem. The MATLAB functions are also much better designed than the implementation of algorithms we constructed using the hardware much more efficiently.

Problem B2

In this problem we are examining the run-time and number of iteration it takes to solve a randomly generated system of equations. A and b were generated with random numbers between 0 and 1. To the diagonal of A is extra "weight" added by changing the parameter w that adds w to all the diagonal entries of

A. The test was then conducted for different sizes N of the systems 100,500 and 1000 and with different weights: 1,5,10 and 100.

The convergence condition for any iterative method is when the spectral radius of the iteration matrix is less than 1 to guarantee convergence. The spectral radius of a square matrix is the maxiumum of the absolute values of the eigenvalues of the matrix. Spectral radius of the iterative matrix:

$$\rho(D^{-1}(L+U)) < 1 \tag{2}$$

A condition for convergence is also that the matrix A is strictly diagonally dominant. This condition is sufficient for convergence but not necessary. Diagonal row dominant means:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \tag{3}$$

Method	**Iterations**	**100**	**500**	**1000**
Jacobi	NaN/NaN/NaN			
Gauss-Seidel	NaN/NaN/NaN			
CG	NaN/NaN/NaN			
myownLU				
Ab		0.00023	0.01010	0.02241
MATLAB LU		0.00021	0.00351	0.02076

Table 2: Problem B2, w = 1

Method	**Iterations**	**100**	**500**	**1000**
Jacobi	NaN/NaN/NaN			
Gauss-Seidel	NaN/NaN/NaN			
CG	NaN/NaN/NaN			
myownLU				
Ab		.00033	0.00982	0.01947
MATLAB LU		0.00020	0.00936	0.01541

Table 3: Problem B2, w = 5

Method	**Iterations**	**100**	**500**	**1000**
Jacobi	NaN/NaN/NaN			
Gauss-Seidel	NaN/NaN/NaN			
CG	NaN/NaN/NaN			
myownLU				
Ab		0.00024	0.009006	0.02136
MATLAB LU		0.00036	0.00426	0.01410

Table 4: Problem B2, w = 10

Method	**Iterations**	**100**	**500**	**1000**
Jacobi	NaN/NaN/NaN			
Gauss-Seidel	NaN/NaN/NaN			
CG	NaN/NaN/NaN			
myownLU				
Ab		0.00030	0.00724	0.02376
MATLAB LU		0.00025	0.00534	0.01787

Table 5: Problem B2, w = 100

N	ρ (w = 1)	ρ (w = 5)	ρ (w = 10)	ρ (w = 100)
100	33.42	9.15	4.76	0.49
500	171.81	45.53	23.26	2.48
1000	346.61	91.06	47.63	47.63

Table 6: Spectral radius of iterative matrix

When the spectral radius of the iterative matrix becomes less than 1 (and the matrix becomes Diagonal dominant row wise) at w=100 and N=100 we can observe that the Jacobi methods starts to work, this is because it guaranteed to solve systems with spectral radius < 1.

Gauss-Siedel method is guaranteed to converge when the matrix is Diagonally dominant or for symmetric positive matrices. We can therefore not be certain that this method works for any other cases except for the same case that Jacobi method worked (N=100 and w = 100). Even though the it converges in many other cases we can see that the number of iterations are very high and that it would not be efficient to use it, and we cant be certain that it allways converges.

The Conjugate gradient method shows similar behaviour as Gauss-Siedel, we can only prove that it converges for positive definite matrices so even if it converges it is not certain it would always do that and the number of iterations is often to high to be useful.

Problem B3

In this section is a large sparse matrix solved using the methods from earlier. When trying to solve this system some of the methods we can't really us: e.g: myownLU was excluded, because *LU* factorization mostly don't exists for random matrixes. They exists for symmetric positive definite matrixes so it wasn't a problem in Part B1.

Even though the Conjugate Gradient method is very useful for large sparse matrices like this, it became clear that the improvement of each iteration for $\alpha = 0$, and 0.00001 was very slow and the test was only finished with $\alpha = 0.1$ and 0.0001. This is Because the improvement of each iteration of the Conjugate Gradient method depends on the condition number. The Conjugate gradient method would finish in the other cases also if we let it run and if the round off errors wouldn't become to big.

For the other two methods (Jacobi and Gauss-Seidel) it was also clear that the condition number had an effect on speed of convergence, making it unpractical to time them for the ill conditioned systems where $\alpha = 0, 0.001$ and 0.00001.

- for $\alpha = 0.1$ cond(A) ≈ 41
- for $\alpha = 0.001$ cond(A) ≈ 4000
- for $\alpha = 0.00001$ cond(A) ≈ 39600
- for $\alpha = 0$ cond(A) ≈ 4053700

Method	**Iteration**	a = 0	a = 0.1	a = 0.001	a = 0.0001
Jacobi	NaN/NaN/NaN/NaN				
Gauss-Seidel	NaN/NaN/NaN/NaN				
CG	NaN/NaN/NaN/NaN				
myownLU					
Ab		0.0003	0.0003	0.0002	0.0004
Matlab LU		0.0014	0.0016	0.0014	0.0017

Table 7: Result Problem B3

The result once again also highlight that even though Conjugate gradient method is a good method for this kind of problem are the MATLAB functions adaptive and much more efficiently constructed.