Sorbonne Université

Advanced Numerical Algorithms

Practical 2 : - Iterative methods

Anatole Vercelloni

Teacher : Stef Graillat

Janvier 2023

Table des matières

The goal of this practical is to benchmark iterative methods. There is a lot of things to do so I decide to focus just on one matrix to well analysis the different methods.

So let $n \in \mathbb{N}$ and

$$A_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & n \end{pmatrix} \tag{1}$$

I choose this matrix because it is Diagonally dominant and PSD which is important for the hypothesis of some method (conjugate gradient). And additionally, it is very sparse. So it will be interesting to compare also with direct solver

I implemented the iterative methods in matlab and I get all these codes :

1 Jacobi's method

First of all, the Jacobi's method :

```
function x=Jacobi(A,y,xo,itmax, eps)
    n = length(y);
    x = xo;
    xold = x;
    tic
    for it=1:itmax
        for i=1:n
        x(i) = (y(i)-A(i,[1:i-1,i+1:n])*xold([1:i-1,i+1:n]))/A(i,i);
        end
         if abs(xold - x) < eps
                break;
         end
    xold=x;
    end
    toc
end
```

I tested this method with a diagonal matrix of size going from 1 to 1000 and obtain the following result for the time

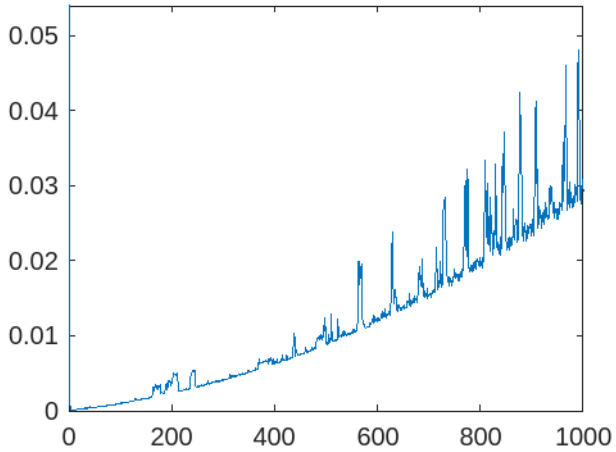

FIGURE 1 – Jacobi's method time in function of n

2 Gauss-Seidel

Then the Gauss-Seidel method which is a Jacobi method but use the more recent computed term

```
function [x, t]=GS(A,y,xo,itmax)
    n = length(y);
    x = xo;
    tic
    for it=1:itmax
        xold = x;
        for i=1:n
            x(i)=(y(i)-A(i,1:i-1)*x(1:i-1)-A(i,i+1:n)*x(i+1:n))/A(i,i);
        end
         if abs(xold - x) < eps
                break;
         end
    end
    t = toc;
end
```

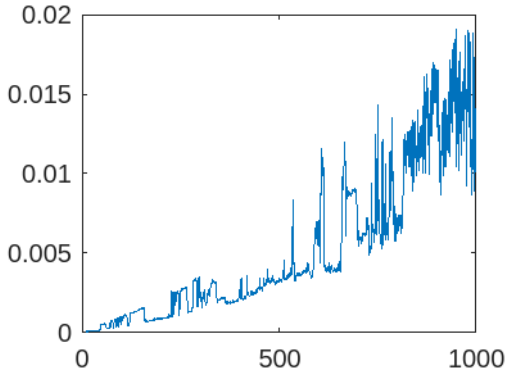

Figure 2 – Gauss-Seidel's method time in function of n

3 Successive over relaxation method

This method is an improvement of the previous one with an $\omega \in]0, 2[$ which is the over relaxation coefficient.

```
function [X,t] = SOR( A ,B, w, X0, eps )
    [na , ~ ] = size (A);
    tol = eps*ones(na,1);
    k= 1;
    xkold = X0;
    xk = X0;
    err= 10;
    tic
    while sum(abs(err) >= tol) ~= zeros(na,1)
        for i = 1:na
            s1 = 0;
            s2 = 0;
            for j=1:i-1
                s1 = s1 + A(i,j)*xk(j);
            end
            for j=i+1:na
                s2 = s2 + A(i,j)*xkold(j);
            end
            xk ( i  ) = w/A(i,i)*(B(i)- s1 - s2) + (1 - w)*xkold(i);
            err = xk( :) - xkold( :);
            xkold = xk;

        end
        k = k+1;
    end
    t = toc;
    disp(k)
    X = xk;
end
```

The best value of ω depend of the structure of the matrix so I did some measure to find the better one for An.

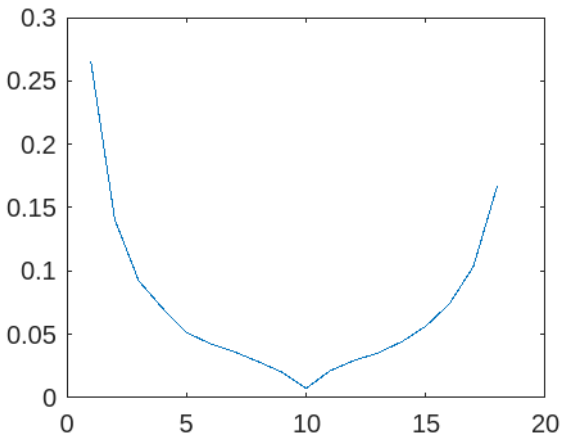

FIGURE 3 – SOR method time in function of ω (n=1000)

We can observe that in this case, the better ω is 1 which the same as doing the G.S. method. We can still plot the result that we get :

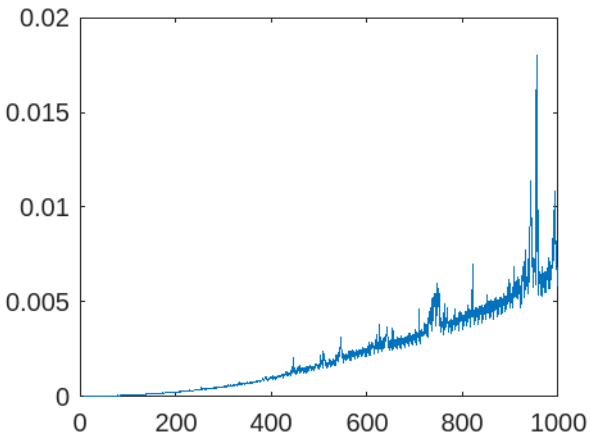

FIGURE 4 – SOR method time in function of n (ω =1)

4 conjugate gradient

The last method that we studied is the conjugate method which is a method based on Krylov subspace.

```
function [x, t] = conjgrad(A, b, x0)
    r = b - A * x0;
    p = r;
    rsold = r' * r;
    x = x0;
    tic
    for i = 1:length(b)
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        if sqrt(rsnew) < 1e-10
            break
        end
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
    end
    t = toc;
end
```

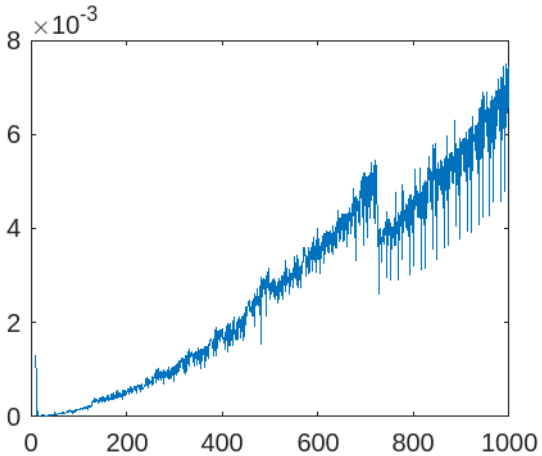

FIGURE 5 – conjugate gradient method time in function of n

5 Conclusion

I also for the experiment did the same computation with the operator \ and get this result :

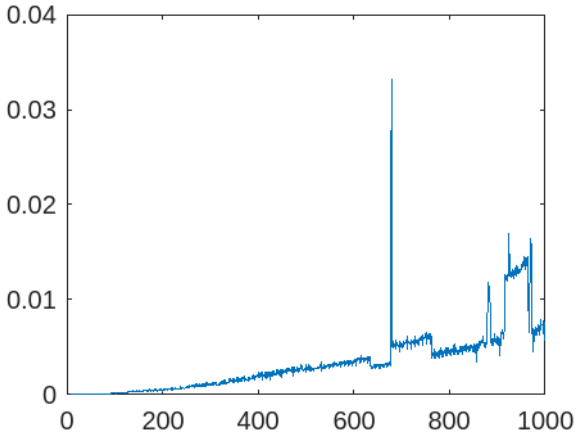

Figure 6 – \ method time in function of n

we got better time for the conjugate gradient than for \, but the \ is more linear. In general, we got quadratics results for all the method but more elaborate the method is, better is the computational time. At the end, we can say that there is no a better method but we have to choose which is the more appropriate to our problem depending of the matrix (SPD, sparse, ..) but also on the accuracy claimed (direct or iterative methode).