MA519: Numerical Linear Algebra

Lab 3: Iterative solvers

We'll implement and compare some iterative solvers (stationary and non-stationary)

1 The finite difference method

As we did in Lab 2, we'll study the solution of a linear system arising from the discretisation of the Laplace equation. The linear system is generated by the following Matlab code

```
N = 8; % points on 1D mesh
TN = sparse(2:N, 1:N-1, -1, N,N) ...
   + sparse(1:N, 1:N, 2,N,N) ...
   + sparse(1:N-1, 2:N, -1, N,N);
A = kron(speye(N),TN) + kron(TN, speye(N));
m = N^2;
h = 1/N;
b = h^2*ones(m,1);
x = A\b;
```

2 Basic iteration

recall from Lectures 24–28, that we can try to solve the linear system $A\mathbf{x} = \mathbf{b}$ *iteratively*, as follows:

- Choose an initial guess $\mathbf{x}^{(0)}$;
- Choose a splitting A=M-N;
- Iterate as

$$\mathbf{x}^{(k+1)} = M^{-1}N\mathbf{x}^{(k)} + M^{-1}\mathbf{b}. \qquad (1)$$

For example, for Jacobi's method $M = D$, where D is the diagonal of A; for the Gauss-Seidel method, $M = D - E$, where E That is

$$d_{ij} = \begin{cases} a_{ii} & i = j \\ 0 & i \neq j. \end{cases} \quad \text{and} \quad e_{ij} = \begin{cases} -a_{ij} & i > j \\ 0 & i \leqslant j. \end{cases}$$

This code implements Gauss-Seidel in Matlab:

```
D = diag(diag(A));
E = -tril(A,-1);

M = D-E; N = M-A;
k = 0;
x0 = zeros(m,D = diag(diag(A));
r0 = b - A*x0; %residual
while (norm(r0) > TOL)
    k=k+1;
    x1 = M\(N*x0) + M\b;
    x0 = x1;
    r0 = b - A*x0;
end
```

Rather than typing this in, you can download the code from the course website. The program is called `TestGaussSeidel.m`. That program does a few other things, such as displaying norms of the error and residual at each iteration. It takes about 160 iterations to achieve a residual of less than 10^{-10}, so the program displays the result for only every 10^{th} iteration. It can be easier to visualise the residual reduction in a graph, so this is also produced. This is best done on a semi-log plot, as shown below in Figure 1. This also shows the residual reduction for the Jacobi Method. As we can see, it takes roughly twice the number of iterations as the Gauss-Seidel method.

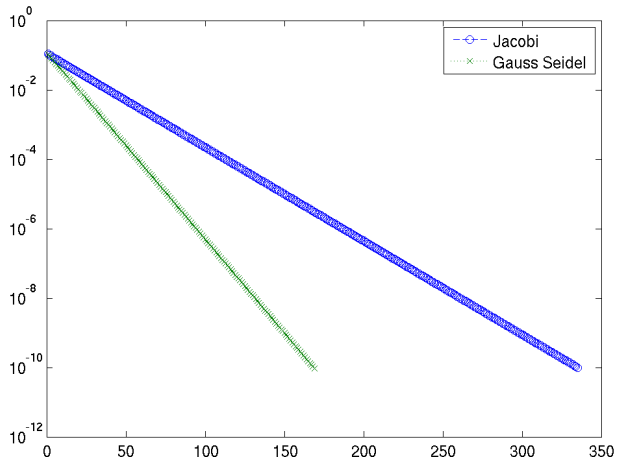

Figure 1: Residual reduciton for Jacobi and Gauss-Seidel methods

Exercise 2.1. *Modify the Matlab program so that it also uses the Jacobi method. verify the results displayed in Figure 1.*

3 Orthomin(1)

The iteration shown in Equation 1 is called *stationary* because M does not depend on k. If we allow the iteration to change at every step, we get a *non-stationary* method. The simplest can be written as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + a_k(\mathbf{b} - A\mathbf{x}^{(k)}). \qquad (2)$$

The goal is to choose a_k in the best way possible. If we do that to minimise the 2-norm of the residual, $\mathbf{r}^{(k+1)}$, we get the *Otthomin(1)* method:

$$a_k = \frac{(\mathbf{r}^{(k)}, A\mathbf{r}^{(k)})}{(A\mathbf{r}^{(k)}, A\mathbf{r}^{(k)})}. \qquad (3)$$

This can be implemented as

```
k = 0;
x0 = zeros(m,1);
r0 = b - A*x0; % residual
while (norm(r0) > TOL)
    k=k+1;
    Ar0 = A*r0;
    a0 = (r0'*Ar0)/(Ar0'*Ar0);
    x1 = x0 + a0*r0;
    x0 = x1;
    r0 = b - A*x0;
    Orth1_Residual(k)=norm(r0);
end
```

You can download this code from `TestOrthomin1.m`.

Exercise 3.1. *Modify the Matlab program so that it implements the* Steepest Descent *algorithm from Lecture 29.*

4 Conjugate Gradient Method

If the matrix A is s.p.d., then a better method to apply is called *Conjugate Gradients* (CG). It works as follows:

- Choose an initial guess, $\mathbf{x}^{(0)}$. Set $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$ and $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$.
- For $k = 1, 2, \ldots,$
  - Compute $A\mathbf{p}^{(k-1)}$
  - Set $a_{k-1} = \dfrac{(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)})}{(\mathbf{p}^{(k-1)}, A\mathbf{p}^{(k-1)})}$.
  - Set $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + a_{k-1}\mathbf{p}^{(k-1)}$.
  - Set $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - a_{k-1}A\mathbf{p}^{(k-1)}$
  - Set $b_{k-1} = \dfrac{(\mathbf{r}^{(k)}, \mathbf{r}^{(k)})}{(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)})}$.
  - Set $\mathbf{p}^{(k)} = \mathbf{r}^{(k)} + b_{k-1}\mathbf{p}^{(k-1)}$

Exercise 4.1. *Implement CG, can compare from Orthomin(1) and Steepest Descent.*