Matlab Project

Problem The partial differential equation

$$-u_{xx} = 1, \text{ in } (0,1)$$

with homogeneous Dirichlet boundary conditions, yields after discretizing with centered finite difference a linear system with a tridiagonal matrix,

$$Au = b$$

1 Stationary iteration

we can try to solve the linear system $Ax = b$ *iteratively*, as follows:

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

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

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
```

Exercise. *Modify the Matlab program so that it also uses the Jacobi method.*

2 Conjugate Gradient Method

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

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

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 following method:

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

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
```

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 $a_{k-1} = \frac{(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)})}{(\mathbf{p}^{(k-1)}, A\mathbf{p}^{(k-1)})}$.
  - 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 $b_{k-1} = \frac{(\mathbf{r}^{(k)}, \mathbf{r}^{(k)})}{(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)})}$.
  - Set $\mathbf{p}^{(k)} = \mathbf{r}^{(k)} + b_{k-1}\mathbf{p}^{(k-1)}$

Project. **Implement Jacobi, Gauss-Seidel, and *CG*, and compare the restults. First discreize the PDE with a uniform mesh with (M+2) grid points (i.e., partition (0, 1) into M+1 equally-spaced subintervals). Then apply the centered finite difference to approximate second derivative at each grid point (except two end points where values are given by boundary conditions). Form the linear system with matrix A. Choose M = 500; Then you should have A with size 500x500, and A is tridiagonal. Plot the residual with respect to number of iterations. Choose TOL = 1e-16. You can use semi-log plot or plot exponent of residual with respect to number of iterations. Comment on your results.**