MA2501 Numerical methods

Mandatory problem set 2

Practical information

- **Handout**: Monday the 24th of April
- **Deadline**: Friday the 12th of May, no later than 6pm.
- **Guidance**:
  - The lecturer will be in office 1354 of "Sentralbygg II" during regular lecture hours and scheduled exercise guidance hours. Additionally, from 2.15pm to 4pm every day apart from Wednesday the 26th of April. Questions may be sent by email to `<bardsk@math.ntnu.no>` and will be answered as far as the questions are reasonable.
  - The exercise assistant's schedule will be listed on the course homepage.

  All email sent to the course staff *must* be marked 'MA2501'.
- **Size of groups**: At most three students may constitute a group.

Introduction

A plate made from a material of homogeneous heat conduction properties is exposed to a uniformly distributed external heat source while the edges of the plate are kept at 0° C. The heat source has been switched on for a long time, so the temperature distribution within the plate is now steady (i.e. does not change with time). We can then show that the temperature $u(x, y)$ in the plate will satisfy a partial differential equation with boundary conditions of the form

$$\begin{aligned} -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) &= 1, \quad \text{within the plate} \\ u(x, y) &= 0, \quad \text{on the boundary} \end{aligned} \tag{1}$$

We will later in the course see how to restate the problem (1) into a problem more suitable to resolution by means of an electronic computer. For now, however, knowing that the restated problem is a linear system of equations,

$$A\mathbf{u} = \mathbf{f}, \tag{2}$$

is sufficient. The solution $\mathbf{u}$ is the temperature values in a finite number of *nodal points* within the plate. This problem set studies and compares various iterative methods for resolving the system (2).

To reduce the amount of programming required in this problem set, a programme which sets up the linear system (i.e. creates A and $\mathbf{f}$), solves the system by means of Gaussian elimination, and plots the solution is available on the course homepage. The programme, called `driver`, uses the auxillary functions `setup` and `plot_solution`. These functions may be downloaded from the course homepage. `driver` is typically used as follows

```
>> R = 'B'; n = 32;
>> driver(R, n)
```

The parameter `n`, an integer strictly greater than zero, designates the level of accuracy in the solution. Increasing `n` produces a more accurate numerical solution at the cost of increasing the amount of computation required to determine the solution. Specifically, the dimension of (2) increases quickly when increasing `n`. The parameter `R` is a *name* of a specifically shaped plate. Using `R = 'B'` means a square plate with a hole in the shape of a butterfly cut out in the middle of the plate. Other possible values are listed in the function `setup`. We recommend starting from `driver` when writing your own code.

We also remark that the matrix A, irrespective of the specific choices made for `R` and `n`, is symmetric and positive definite (SPD). Moreover, the matrix is *sparse* meaning that only very few elements of A are different from zero—in this case at most 5 elements per row. The MATLAB function `spy` allows visual inspection of which matricial elements are zero and which are not.

The report

Every group will produce an independent report from the work. The size of the report should be no more than $\boldsymbol{3 + 2n}$ pages with $\boldsymbol{n}$ being the number of students in the group. The limitation does *not* include print-out of MATLAB code. You may discuss different approaches to solving the problems with other groups, but the discussion should not infringe on the independence of your work. You are not to copy other people's MATLAB code and every member of the group should be able to defend the content of the report.

Please cite references if gleaning material from the internet or texts other than Kincaid and Cheney.

The report should be marked with a ***group number*** and the ***names*** of ***all*** members of the group. The group number will be assigned by the lecturer once the group has been formed.

Problem 1 – Classical iterative methods

Necessary background material for this problem is the text on iterative methods in Chapter **8.2** of Cheney & Kincaid.

1) Briefly describe the method known as *Successive Overrelaxation* (SOR). Give reasons why SOR might be useful for the numerical resolution of the lineary system of equations (2).

2) Implement SOR and test the implementation on the linear system of equations (2).

3) How does the number of SOR iterations vary with the relaxation parameter ω? How does the "optimal" relaxation parameter vary with the dimension of (2) and how sensitive is the number of SOR iterations to the optimal value of ω?

4) Suggest a strategy for exploiting the sparsity structure of the matrix A. You do not have to implement this strategy.

5) Decompose the matrix A of (2) as

$$A = D - C_L - C_U$$

with $D = \operatorname{diag}(A)$, $C_L = -\operatorname{tril}(A, -1)$, and $C_U = \operatorname{triu}(A, 1)$. The *symmetric successive overrelaxation* method (SSOR) is given in matricial form as

$$\begin{aligned}(D - \omega C_L)\mathbf{x}^{(k+1/2)} &= \omega(C_U\mathbf{x}^{(k)} + \mathbf{b}) + (1-\omega)D\mathbf{x}^{(k)}\\ (D - \omega C_U)\mathbf{x}^{(k+1)} &= \omega(C_L\mathbf{x}^{(k+1/2)} + \mathbf{b}) + (1-\omega)D\mathbf{x}^{(k+1/2)},\end{aligned}$$

for $k = 0, 1, 2, \ldots$. Give an intuitive description of this algorithm.

State the algorithm in component form and implement it. Investigate how the performance, measured both in terms of the number of iterations *and* computational time, of SSOR compares with the corresponding results of SOR. Finally, study SSOR's dependence on the relaxation parameter.

Problem 2 – conjugate gradients

Let A be a fixed symmetric, positive definite matrix of dimension $n \times n$ and $\mathbf{b}$ a fixed vector of dimension n. Define the function $f : \mathbb{R}^n \to \mathbb{R}$ as

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^{\mathsf{T}}A\mathbf{x} - \mathbf{x}^{\mathsf{T}}\mathbf{b}. \tag{3}$$

1) Prove that f has a unique critical point $\mathbf{x}_*$ given as the solution to the linear system of equations $A\mathbf{x}_* = \mathbf{b}$ and that $f(\mathbf{x}_*) < f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$ when $\mathbf{x} \neq \mathbf{x}_*$.

Hint: Write $\mathbf{x} = \mathbf{x}_* - \mathbf{e}$ with $\mathbf{e} \neq \mathbf{0}$.

2) Let $\mathbf{x} \in \mathbb{R}^n$ be arbitrary. Argue that the "residual" $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ is the direction in which f decays most quickly from $\mathbf{x}$.

3) Let $\{\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(n-1)}\}$ be a set of *conjugate* (known too under the name of *A-orthogonal*) vectors, i.e.

$${\mathbf{p}^{(i)}}^{\mathsf{T}}A\mathbf{p}^{(j)} = 0$$

whenever $i \neq j$. Let $\mathbf{x}^{(0)} \in \mathbb{R}^n$ be arbitrary and define a sequence of vectors $\mathbf{x}^{(k+1)}$ by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{p}^{(k)}.$$

Determine α_k such that $\mathbf{e}^{(k+1)} = \mathbf{x}_* - \mathbf{x}^{(k+1)}$ becomes A-orthogonal to $\mathbf{p}^{(k)}$. Give reasons why this in exact arithmetic (i.e. in the absence of roundoff errors) means $\mathbf{x}^{(n)} = \mathbf{x}_*$.

The method of *conjugate gradients* is a clever way of iteratively constructing the conjugate *search directions* $\mathbf{p}^{(k)}$.

Using various polynomial identities and the properties of a certain class of vector spaces called *Krylov spaces*, this method can be stated as

$$\begin{aligned}\alpha_k &= \frac{{\mathbf{r}^{(k)}}^{\mathsf{T}}\mathbf{r}^{(k)}}{{\mathbf{p}^{(k)}}^{\mathsf{T}}A\mathbf{p}^{(k)}}\\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k\mathbf{p}^{(k)}\\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}\\ \beta_{k+1} &= \frac{{\mathbf{r}^{(k+1)}}^{\mathsf{T}}\mathbf{r}^{(k+1)}}{{\mathbf{r}^{(k)}}^{\mathsf{T}}\mathbf{r}^{(k)}}\\ \mathbf{p}^{(k+1)} &= \mathbf{r}^{(k+1)} + \beta_{k+1}\mathbf{p}^{(k)}\end{aligned}$$

for all $k \geq 0$ with $\mathbf{p}^{(0)} = \mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$. How much storage (memory) is needed to implement this algorithm if we only care about the final result $\mathbf{x}_*$?

Historically, this method was introduced around 1950 as an alternative to the method of Gaussian elimination of constructing the exact solution to a linear system. In other words, the method was originally viewed as a way of constructing $\mathbf{x}^{(n)}$. However, the method was not really used in practice due to significantly higher cost compared to Gaussian elimination when run in this mode. On the other hand, the method received renewed interest as an *iterative* method for $A\mathbf{x}_* = \mathbf{b}$ approximately 20 years later when it was discovered that $\mathbf{x}^{(k)} \approx \mathbf{x}_*$ even for $k \ll n$. The method of conjugate gradients thus became a useful tool for solving large linear systems of equations of the type arising in the numerical resolution of partial differential equations such as (1).

MATLAB has a built-in function, `pcg`, which implements this method. `pcg` uses a stopping criterion of the form

$$\|\mathbf{r}^{(k)}\|_2 \leq \varepsilon \|\mathbf{r}^{(0)}\|_2 \tag{4}$$

in which the *tolerance* ε kan be specified by the user but uses the default value of 10^{-6} if unspecified. Use `pcg` to resolve the linear system of equations (2). How does the number of "CG" iterations vary with the dimension of the system?

It can be shown that the error $\mathbf{e}^{(k)} = \mathbf{x}_* - \mathbf{x}^{(k)}$ satisfies

$$\|\mathbf{e}^{(k)}\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \cdot \|\mathbf{e}^{(0)}\|_A$$

in which κ is the condition number of A and

$$\|\mathbf{v}\|_A^2 = \mathbf{v}^\mathsf{T} A \mathbf{v}$$

is known as the *A-norm* of a positive definite matrix. How does this corroborate your observations for the number of CG iterations?

Hint: How does the number of CG iterations required to satisfy the condition $\|\mathbf{e}^{(k)}\|_A \leq \varepsilon \|\mathbf{e}^{(0)}\|_A$ depend on κ?