Numerical Mathematics I, 2016/2017, Lab session 4

Keywords: iterative methods, linear system, power method

Remarks

- Make a new folder called `NM1_LAB_4` for this lab session, save all your functions in this folder.
- Whenever a new `MATLAB` function is introduced, try figuring out yourself what this function does by typing `help <function>` in the command window.
- Make sure that you have done the preparation before starting the lab session. The answers should be worked out either by pen and paper (readable) or with any text processing software (LaTeX, Word, etc.).

1 Preparation

1.1 Iterative methods

1. Study (Textbook, Section 5.9 - 5.12).
2. Show that an iterative method of the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)}, \quad \mathbf{z}^{(k)} = \mathbf{P}^{-1}\mathbf{r}^{(k)}, \tag{1}$$

leads to the following relationship of the error $\mathbf{e}^{(k)}$

$$\mathbf{e}^{(k+1)} = \mathbf{B}_{\alpha_k}\mathbf{e}^{(k)}, \quad \mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)},$$

where $\mathbf{P}$ is a preconditioner, $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ and $\mathbf{B}_{\alpha_k} = \mathbf{I} - \alpha_k \mathbf{P}^{-1}\mathbf{A}$.

3. Assume $\mathbf{A}$ is a real symmetric and positive definite matrix. Show that minimising the error in the $\mathbf{A}$-norm leads to the dynamic parameter value

$$\alpha_k^1 := \frac{(\mathbf{z}^{(k)})^T\mathbf{r}^{(k)}}{(\mathbf{z}^{(k)})^T\mathbf{A}\mathbf{z}^{(k)}},$$

see also (Textbook, Equation 5.60). Hint: solve $\frac{d}{d\alpha_k}(\mathbf{e}^{(k+1)})^T\mathbf{A}\mathbf{e}^{(k+1)} = 0$ for α_k.

4. The 2-norm of the residual $\mathbf{r}^{(k+1)}$ can also be written as the $\mathbf{C}$-norm of $\mathbf{e}^{(k+1)}$, for which matrix $\mathbf{C}$?
5. Show that minimising the 2-norm of the residual in each step leads to

$$\alpha_k^2 := \frac{(\mathbf{A}\mathbf{z}^{(k)})^T\mathbf{r}^{(k)}}{(\mathbf{A}\mathbf{z}^{(k)})^T(\mathbf{A}\mathbf{z}^{(k)})}.$$

6. Consider Richardson iteration without preconditioning (hence $\mathbf{P} = \mathbf{I}$) on a real symmetric positive definite matrix $\mathbf{A}$, and let

$$\alpha_k = \alpha_{\text{opt}} = \frac{2}{\lambda_{\max} + \lambda_{\min}} \tag{2}$$

be the static parameter, where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalue of $\mathbf{P}^{-1}\mathbf{A}$. See also (Textbook, Equation 5.58).

After checking that it follows from $\mathbf{P} = \mathbf{I}$ that $\mathbf{B}_\alpha$ is symmetric, show that

$$\rho(\mathbf{B}_\alpha) = \|\mathbf{B}_\alpha\|_2 = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}.$$

For the relative residual $\frac{\|\mathbf{r}^{(k)}\|_2}{\|\mathbf{b}\|_2}$ to be smaller than a given tolerance ϵ, show that the expected number of iterations $k_{\min}$ is given by

$$k_{\min} = \frac{\log\left(\frac{\epsilon\|\mathbf{b}\|_2}{\|\mathbf{r}^{(0)}\|_2}\right)}{\log \rho(\mathbf{B}_\alpha)}.$$

Hint: show that $\mathbf{r}^{(k+1)} = \mathbf{B}_\alpha \mathbf{r}^{(k)}$.

1.2 Power iteration

1. Study (Textbook, Section 6.1-6.2).

2. Consider the power iteration

$$\mathbf{x}^{(k+1)} = \mathbf{A}\mathbf{y}^{(k)}, \quad \mathbf{y}^{(k+1)} = \frac{\mathbf{x}^{(k+1)}}{\|\mathbf{x}^{(k+1)}\|_2}, \quad \lambda^{(k)} = (\mathbf{y}^{(k)})^T \mathbf{x}^{(k+1)}.$$

Under what conditions on the matrix $\mathbf{A}$ and the initial guess $\mathbf{x}^{(0)}$ does $\lambda^{(k)} \to \lambda_1$?

3. What are the convergence order and factor when using the power method on a Hermitian matrix? What if the matrix is non Hermitian?

2 Lab experiments

2.1 Iterative methods

Introduction

The Poisson problem in two dimensions is given by the following elliptic partial differential equation

$$\begin{aligned} -\Delta u &= f \qquad \text{for } (x, y) \in \Omega, \\ u &= g \qquad \text{for } (x, y) \in \partial\Omega, \end{aligned}$$

where $\Omega \subset \mathbb{R}^2$. You will learn more about this in a later chapter. For now it is sufficient to know that solving such a partial differential equation using any common type of discretisation (finite difference, finite element, finite volume) method yields a system of equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

We take $f(x, y) = 1$, $g(x, y) = 0$ and $\Omega = [0, 1]^2$. The matrix $\mathbf{A}$ and right-hand side vector $\mathbf{b}$ can be made in MATLAB using

```
A = gallery('poisson', N); b = ones(N^2,1)/(N^2);
```
(3)

where N is the number of subintervals in each direction. You will compare several iterative solution methods on solving this system of equations.

Experiment

Write a MATLAB function `iterMethod` that can solve a system of linear equations using an iterative method as given in (1), see also (Textbook, Equation 5.62). It should satisfy the following requirements

- The function should be able to use a preconditioner $\mathbf{P}$ of $\mathbf{A}$ (specified as an input variable). If no preconditioner is specified*, the preconditioning step should be skipped in the main loop.
- The function should work either with a static parameter $\alpha_k = \alpha$ or with a dynamic parameter α_k^1 or α_k^2.
- The matrix vector multiplication with $\mathbf{A}$ should be done only once per iteration.
- The function should use the residual based *a posteriori* error estimator (Textbook, Section 5.12) for the stopping criterion.
- The function should output the solution $\mathbf{x}$, an integer which is 0 if the method converged, and the convergence history of the 2-norm of the *relative* residual.

The header of your function should be of the following form

```
% Attempts to solve A * x = b, with initial guess x0 using
% an iterative method of the form
%           x^{k+1} = x^{k} + alpha_k P \ r^{k}, r^{k} = b - A x^{k}
% where alpha_k = alpha0 if dynamic = 0
% INPUT
% A         n x n matrix
% b         n x 1 right-hand side
% x0        initial guess
% tol       desired tolerance
% maxIt     maximum number of iterations
% P         preconditioner of A (optional)
% dynamic   0: static, 1: minimise A-norm of error, 2: minimise
%           2-norm of residual
% alpha0    if dynamic = 0, this value is used for alpha_k
% OUTPUT
% x         approximate solution to A * x = b
% flag      if 0 then tolerance is attained
% convHist  relative residual per iteration
function [x, flag, convHist] = iterMethod(A, b, x0, tol, maxIt,...
    P, dynamic, alpha0)
```

*Hence the input `P = []`, this can be verified in MATLAB using `isempty(P)`.

To ensure that your function works properly, test it on the Poisson problem with `N = 10`, `tol = 1E-12` and `maxIt = 10*N^2`. Use $\mathbf{x}^{(0)} = \mathbf{0}$.

For the comparison of the methods you may use the given function `iterCompare`. This function uses your implementation of `iterMethod` to compare several variants of the iterative method (1). Therein the preconditioner $\mathbf{P}$ as well as the type of parameter α is varied. For the preconditioner we consider using: no preconditioner, Jacobi preconditioner and Gauss-Seidel preconditioner. And for the parameter we consider using: static parameter $\alpha_k = \alpha_{\text{opt}}$ (for Gauss-Seidel we can not compute the optimal α_k, so it uses $\alpha_{\text{opt}} = \frac{3}{2}$) and either of the dynamic parameters. In total you will therefore have nine different methods. The results (number of iterations needed per method) are then summarised in one table.

Compare the convergence histories for both dynamic parameters using Gauss-Seidel as preconditioner. Similarly, compare using no preconditioner with using the Jacobi preconditioner when using the optimal value α_{opt}. The given function `iterCompare` will plot these in one figure.

Remark: the solution of this problem can be viewed as a 2-dimensional surface. Such a surface can be plotted in `MATLAB` using

```
surf(reshape(x, N, N)), colormap(gray(50)),
```

where $\mathbf{x}$ is the solution vector.

Optional

Often the full matrix $\mathbf{A}$ is either not available or expensive to compute. For iterative methods it is sufficient in such a situation to have a function that evaluates, for a given input vector $\mathbf{v}$, the matrix-vector product $\mathbf{Av}$. Add this functionality to your `iterMethod` function. Hence the input `A` can either be a matrix, or function handle.

Before entering the main loop you should check what class the input `A` belongs to. You can do this using the built-in `MATLAB` function `isa`. If `A` is a matrix, then every matrix-vector product `A * v` should be computed "normally". If `A` is not a matrix then you should call the function `A` as `A(v)`.

2.2 Power iteration

Write a `MATLAB` function `powerMethod` that computes the extremal eigenvalue λ_1 and corresponding eigenvector $\mathbf{x}_1$ of some input matrix $\mathbf{A}$ using an initial guess $\mathbf{x}^{(0)}$. The header of your function should look like

```
% INPUT
% A             n x n matrix
% x0            n x 1 initial guess
% tol           desired tolerance
% maxIt         maximum number of iterations
% OUTPUT
% lambda        eigenvalue of A largest in magnitude
% x             corresponding eigenvector
% flag          if 0 then tolerance is attained
% lambdaList    list of intermediate eigenvalues
% xList         list of intermediate eigenvectors
% convHist      error estimate per iteration
function [lambda, x, flag, lambdaList, xList, convHist] =...
        powerMethod(A, x0, tol, maxIt)
```

The following error estimate should be used as a stopping criterion,

$$\tilde{e}^{(k)} = \frac{|\lambda^{(k-1)} - \lambda^{(k)}|}{|\lambda^{(k)}|}.$$

Use your function to calculate the extremal eigenvalue of $\mathbf{B}_{\alpha_{\text{opt}}} = \mathbf{I} - \alpha_{\text{opt}}\mathbf{P}^{-1}\mathbf{A}$ from the previous exercise when using the Jacobi preconditioner and when using Gauss-Seidel as preconditioner. Use `N = 10` (hence getting a 100×100 matrix) and a *random* initial guess $\mathbf{x}^{(0)}$. Remember that the power method only converges under certain conditions.

Study the convergence behaviour by plotting the logarithm of the relative error

$$e^{(k)} = \frac{|\lambda_1 - \lambda^{(k)}|}{|\lambda_1|} \tag{4}$$

against the iteration number (use the output variable `lambdaList`). Here λ_1 is the actual largest eigenvalue. You can use `eigs` to compute this eigenvalue.

Optional

Same as the previous optional exercise, add functionality to `powerMethod` such that `A` can either be a matrix or a function that evaluates the matrix-vector product $\mathbf{Av}$. Especially for calculating the extremal eigenvalues of $\mathbf{B}_{\alpha_{\text{opt}}}$ this approach is much more efficient since the preconditioner $\mathbf{P}$ need not be explicitly inverted.

3 Discussion

Remark on sparsity

Define the matrix $\mathbf{A}$ as in (3), but now with $N = 100$. Check the datatype of `A` by typing

```
whos A
```

Now make a new array `fullA` by typing

```
fullA = full(A);
```

Why would you prefer `A` as opposed to `fullA`?

3.1 Iterative methods

1. What quantity does the optimal value found in (2) minimise? Why can this not be applied when the Gauss-Seidel preconditioner is used?

2. For arbitrary $\mathbf{A}$, does it make sense to minimise the $\mathbf{A}$-norm of the error?

   $$\|\mathbf{e}\|_{\mathbf{A}} = \sqrt{\mathbf{e}^T\mathbf{Ae}}$$

   If not, what conditions on $\mathbf{A}$ should be imposed for $\|\mathbf{e}\|_{\mathbf{A}}$ to define a norm?

3. What other quantity can be minimised at each step of the iterative process? Explain why the $\mathbf{A}^T\mathbf{A}$-norm *always* defines a norm if $\mathbf{A}$ is full rank.

4. Let $\alpha_k = \alpha_{\text{opt}}$. Why does the convergence history when using no preconditioner coincide with the convergence history when using the Jacobi preconditioner?

5. Does the number of iterations agree with $k_{\min}$ when using Jacobi with $\alpha_k = \alpha_{\text{opt}}$? Use `eigs` to compute $\rho(\mathbf{B}_{\alpha_{\text{opt}}})$.

6. When $\alpha_k = \alpha_k^1$, does the 2-norm of the (relative) residual monotonically decrease? What if we use $\alpha_k = \alpha_k^2$? Do your experiments confirm this?

3.2 Power iteration

1. The stationary iterative methods (so α_k is constant) from the first problem can be viewed as a power iteration as well, explain this. In which direction will the error $\mathbf{e}^{(k)}$ point?

2. What is the observed convergence factor of the relative error (4)? Does this agree with the theory? (For both matrices.)

3. What happens in exact arithmetic if $\mathbf{x}^{(0)}$ contains no component in the direction of the dominant eigenvector $\mathbf{x}_1$? Is this a problem in practice?

4. *Optional.* If $\mathbf{A}$ is nonsingular and symmetric, show that

$$\tilde{\lambda}^{(k)} := \|\mathbf{A}\mathbf{y}^{(k)}\|$$

is an estimate that converges to $|\lambda_1|$, even if

$$|\lambda_1| = |\lambda_2| > |\lambda_3| > \dots.$$

Test it on the iteration matrix $\mathbf{B}_{\alpha_{\text{opt}}}$ corresponding to the Jacobi preconditioner. Is $\mathbf{y}^{(k)}$ an approximation to an eigenvector?