Norwegian University of Science and Technology
Department of Mathematical Sciences

Project part 1

These exercises should be done on a computer using a programming language of your choice, however in the supervision we will give priority to Python or MATLAB. Please submit a short report containing your answers and provide properly commented codes. If you program in Python you can submit a Jupiter-notebook code, containing both codes and report in the same file.

Some of the tasks involve sparse matrices, to obtain efficient code it is preferable to work with sparse matrices using for example the library *sparse* of *scipy* or similar tools in MATLAB. In these exercises are allowed to work with full matrices if you find it easier, but make sure that the comparisons of the various methods in your implementations, e.g. when comparing running times, are fair. Be explicit about these issues in your report.

Exercise 1 Consider the Poisson equation on the unit square $\Omega = [0,1] \times [0,1]$

$$\nabla^2 u(x,y) = f(x,y),$$

where $f(x,y) = 1$ in the interior of the domain, with homogeneous Dirichlet boundary conditions. We will discretize the problem onto a regular $n \times n$ grid. This yields a linear system

$$A\mathbf{u} = \mathbf{b}, \quad \mathbf{b} := \Delta x^2 \cdot \mathbf{f},$$

where $A \in \mathbb{R}^{n^2 \times n^2}$, the vectors[1] $\mathbf{u}, \mathbf{f} \in \mathbb{R}^{n^2}$ have components $u_{i,j} \approx u(x_i, y_j)$ and $f_{i,j} = f(x_i, y_j)$ with $i,j = 1,\ldots,n$ and $x_i = i \cdot \Delta x$, $y_j = j \cdot \Delta y$, and $\Delta x = \Delta y = \frac{1}{n+1}$. $L \in \mathbb{R}^{n^2 \times n^2}$ is the discrete Laplacian obtained applying the five-point finite-difference stencil to approximate the Laplacian operator ∇^2, see [4, ch. 12] for details. This gives the block-tridiagonal matrix L, $n^2 \times n^2$,

$$L = \begin{pmatrix} B & I_n & & \\ I_n & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots & I_n \\ & & & I_n & B \end{pmatrix}, \quad \text{where} \quad B = \begin{pmatrix} -4 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots & 1 \\ & & & 1 & -4 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

and I_n is the $n \times n$ identity matrix. It can be shown that L is a symmetric and positive definite matrix. We will now solve this linear system using some different iteration methods of the type

$$A_1 \mathbf{u}^{(k+1)} = A_2 \mathbf{u}^{(k)} + \mathbf{b}, \tag{1}$$

where $A = A_1 - A_2$ with $\det(A_1) \neq 0$. The way we choose the matrices A_1 and A_2 will identify the iterative method and we will recover three classical iterative methods for linear systems.

Assume A_d, A_l and A_u are the diagonal, minus the strictly lower-triangular and minus the strictly upper-triangular parts of the matrix A such that $A = A_d - A_l - A_u$. We will use the following iterative methods to calculate approximate solutions to the linear system:

[1] These vectors are constructed from the n columns of the grid stacked on top of each other n times.

a) Jacobi: $A_1 = A_d$.

b) Forward Gauss-Seidel: $A_1 = A_d - A_l$.

c) Successive over relaxation: $A_1 = A_d - \omega A_l$, where you can choose the value of ω so to optimise convergence.

See chapter 4.1 in the text book (Y. Saad) for more details about these methods.

Consider the residual vector $\mathbf{r}^k := \mathbf{b} - A\mathbf{u}^k$. Fix $n = 10$, $n^2 = 100$ the size of the linear system.

i) Compare the speed of convergence for the methods. Produce a semi-log plot of the 2-norm of the relative residual $\frac{\|\mathbf{r}^k\|_2}{\|\mathbf{r}^0\|_2}$ versus the number of iterations. Plot all the methods in the same plot for comparison. Try choosing different values of ω in the SOR method, and try to find a value that gives you optimal convergence for this problem.

ii) You should implement a stopping criterion for the three methods. Then measure the relative time each method takes per iteration, and the time needed to guarantee that the norm of the relative residual is below tolerances $TOL = 1e-7$ and $TOL = 1e-14$. Use the function `time.time()` in Python. Make a table containing the results for each method.

iii) This exercise is open-ended. Calculate the spectral radius of $A_1^{-1}A_2$ and see how it relates to the convergence of the method (Hint: Python's `numpy.linalg.eigvals(A)` function returns the eigenvalues of A). Recall that there is a theorem that links the spectral radius of $A_1^{-1}A_2$ to the speed of convergence of the iteration. The theorem states that a spectral radius below 1 is a necessary and sufficient condition for convergence independently on how $\mathbf{u}^0$ is chosen. In fact, the smaller the spectral radius the faster the convergence. Does this theorem reflect what happens in your example? How can you verify this with numerical experiments?

Exercise 2 We will assume that both A and A_1 are symmetric and positive definite. It is a well known fact that the iterative methods of the form (1) can be seen as a gradient descent iteration for minimising the quadratic function

$$E : \mathbb{R}^N \to \mathbb{R}, \qquad E(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T A\mathbf{u} - \mathbf{b}^T\mathbf{u}, \tag{2}$$

[1, 2, 3]. In fact denote with $\langle\cdot,\cdot\rangle$ the Euclidean inner product and with $\langle\cdot,\cdot\rangle_{A_1}$ the inner product defined by

$$\langle\cdot,\cdot\rangle_{A_1} := \langle\cdot, A_1\cdot\rangle,$$

the gradient of E with respect to the Euclidean inner product is $\nabla E(\mathbf{u}) = A\mathbf{u} - \mathbf{b}$ while with respect to the inner product $\langle\cdot,\cdot\rangle_{A_1}$ is[2] $A_1^{-1}\nabla E(\mathbf{u}) = A_1^{-1}((A_1 - A_2)\mathbf{u} - \mathbf{b})$, and the corresponding gradient descent iteration is

$$\mathbf{u}^{k+1} = \mathbf{u}^k - A_1^{-1}\nabla E(\mathbf{u}^k) = \mathbf{u}^k - A_1^{-1}((A_1 - A_2)\mathbf{u}^k - \mathbf{b}) = A_1^{-1}(A_2\mathbf{u}^k + \mathbf{b}).$$

[2]The gradient of E at $\mathbf{u}$ with respect to the inner product $\langle\cdot,\cdot\rangle_{A_1}$ is the vector $\mathrm{grad}E(\mathbf{u})$ satisfying

$$\langle \mathrm{grad}E(\mathbf{u}), \mathbf{v}\rangle_{A_1} = \left.\frac{d}{d\epsilon}\right|_{\epsilon=0} E(\mathbf{u} + \epsilon\mathbf{v}).$$

The definition is of course valid also when $A_1 = I$ and $\langle\cdot,\cdot\rangle_I$ is the usual Euclidean inner product, then one recovers the usual gradient, $\nabla E(\mathbf{u})$. For the function (2) one finds $\mathrm{grad}E(\mathbf{u}) = A_1^{-1}(A\mathbf{u} - \mathbf{b})$.

Gradient descent iterations can be accelerated by applying the following Polyak heavy ball method in two steps:

$$\mathbf{p}^{k+1} = \mathbf{p}^k - hA_1^{-1}(A\mathbf{u}^k - \mathbf{b}) - h\lambda\mathbf{p}^k \quad (3)$$
$$\mathbf{u}^{k+1} = \mathbf{u}^k + h\mathbf{p}^{k+1} \quad (4)$$

with h and λ appropriate parameters and $\mathbf{p}^0 = \mathbf{0}$ and $\mathbf{u}^0$ an initial guess for the solution of the linear system.

The purpose of this part of the project is to produce numerical evidence of whether it can be worthwhile to use accelerated gradient descent iteration to improve convergence of the classical iterative methods of type (1).

- Implement this iteration for the case when $A_1 = A_d$ and $A_2 = A - A_1$ to obtain an "accelerated Jacobi iteration". Try to find optimal values of λ and h that give the fastest convergence.
- To obtain another method where A_1 is symmetric and positive definite consider the following construction
$$A_1 = A_d + \sigma\mathbf{v}\mathbf{v}^T,$$
here α is the largest eigenvalue of $A - A_d$ and $\mathbf{v}$ the corresponding eigenvector. In the case of the discrete Laplacian L there are explicit formulae for σ and $\mathbf{v}$[3]. Implement a fixed point iteration of type (1) and the corresponding accelerated version (3) with this choice of A_1. Recall that one can easily derive an explicit formula for the inverse of A_1 (Sherman–Morrison formula). Use this formula in your implementations.
- Compare the speed of convergence for the methods. Produce a semi-log plot of the 2-norm of the relative residual versus the number of iterations. Plot the four methods in the same plot for comparison.
- So far the methods have been used to solve the discretization of the Poisson equation. It would be interesting to run the iterative methods on a variety of test problems. Construct one or several symmetric positive definite matrices of your choice and re-run the numerical experiments. One way to proceed is to construct the matrix starting from its eigenvalue decomposition:
  - generate a matrix B $n \times n$ symmetric positive definite by generating randomly its eigenvalues (positive and real) and a random orthogonal matrix Q, $n \times n$;
  - form $B = Q\Lambda Q^T$ where Λ is the diagonal matrix having the eigenvalues of B along the diagonal;
  - then construct a matrix with the following structure

$$L = \begin{pmatrix} B & I_n & & & \\ I_n & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & I_n \\ & & & I_n & B \end{pmatrix}, \quad \text{where} \quad B = Q\Lambda Q^T.$$

[3] One can prove that the components of $\mathbf{v}$ are

$$v_{\ell,m} := \sin(\ell\frac{\pi}{n+1})\sin(m\frac{\pi}{n+1},),$$

and $\sigma = 4\cos(\frac{\pi}{n+1})$.

This task is open-ended. The challenge here is both to design an insightful numerical experiment and to run systematic tests aimed at gaining knowledge on the considered iterative methods. Describe clearly the set-up and aims of your experiments, and present your findings and conclusions in the report.

Bibliography

[1] M.T. Chu, *On the continuous realization of iterative processes*, SIAM Rev., 30 (1988), pp. 375-387

[2] M.T. Chu, *Linear algebra algorithms as dynamical systems*, Acta Numer., 17 (2008), pp. 1-86

[3] Y. Miyatake, *On the equivalence between SOR-type methods for linear systems and the discrete gradient methods for gradient systems* J CAM, vol 342, 2018.

[4] J.C. Strikwerda, *Finite difference schemes and partial differential equations*, SIAM, available online: https://epubs.siam.org/doi/abs/10.1137/1.9780898717938