# PHYS-E0412 Computational Physics :: Homework 10

Due date 26.3. 2019 at 10 am

## Preconditioned Conjugate Gradient Method

The conjugate gradient method is an effective iterative method to solve a linear system of equations $Ax = b$ when $A$ is sparse, symmetric and positive definite. Such systems arise in the discretization of elliptic PDEs. The convergence of the method depends on the condition number $\kappa(A)$ of the matrix $A$. For symmetric, positive definite matrices $\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$.
A standard solution to accelerate the convergence of the conjugate gradient method is to use a preconditioner $P$. A preconditioner is a symmetric, positive definite matrix that is used to transform the original system to $P^{-1}Ax = P^{-1}b$. If $\kappa(P^{-1}A) < \kappa(A)$ a speedup can be expected. Thus, a good preconditioner $P$ is such that it is (spectrally) as close to $A$ as possible while solution of the system $Pz = y$ is relatively inexpensive.
In this homework we study the convergence of the preconditioned conjugate gradient method for discretizations of the elliptic PDE

$$\begin{cases} -\Delta u(x,y) + 2u(x,y) = \exp(-\frac{(x-0.5)^2+(y-0.5)^2}{10}), & (x,y) \in [0,1] \times [0,1] \\ u(x,y) = 0, & \text{on the boundary} \end{cases}$$

a) Implement a discretization of your choice for the above problem. Solvers based on the finite difference method or the finite element method can both be used. (See also Homework 9.) Solve the resulting linear system of equations using the conjugate gradient method without preconditioning. You can use the provided skeleton. What is the condition number of your matrix $A$? (2 p.)

b) Improve the convergence by introducing preconditioners of increasing complexity. In each case report the number of iterations needed to solve the problem and the condition number of the preconditioned system.

   i) Diagonal preconditioner: $P = \text{diag}(A)$.

   ii) Lower-upper preconditioner $P = P_1 P_2$ where $P_1 = L_*$ is the lower triangle of $A$ including diagonal and $P_2 = U_*$ is the upper triangle of $A$ including the diagonal. (Yes, the diagonal gets counted twice.) If you are using matlab check out the functions `tril, triu`.

   iii) Incomplete Cholesky factorization of $A$ without fill-in, IC(0). In the IC(0) preconditioner the matrix $A$ is factored approximately $A \approx \tilde{L}\tilde{L}^T$ where $\tilde{L}$ is an incomplete Cholesky factor of $A$. In the case of no fill-in only non-zero entries of $A$ are included in $\tilde{L}$. For matlab see the function `ichol`.

   Implementing all the above preconditioners can become tedious. Many of them are available as library functions in numerical linear algebra packages. (2 p.)

c) An optimal precondtioner would result in a system where the number of conjugate gradient iterations needed to reach convergence is independent of the size of the matrix $A$. Test if any of the above preconditioners can achieve this for the problem in question. Use a sufficiently wide range of matrix sizes ranging from $O(10)$ to $O(1000)$. (1 p.)