Exercise 8: Steepest descent and conjugate gradient methods

1. **Preconditioned steepest descent method.** Solve the Poisson problem $-\Delta u = f$ on the unit square with homogeneous Dirichlet boundary condition $u = 0$. In this way the stiffness matrix becomes SPD which is needed to use the conjugate gradient method. With the parameter `hmax` of the command `initmesh` with the geometry `'squareg'` and refining sufficiently often, we can generate a mesh with about 1000–3000 nodes.

   Use the command `assempde` to construct the stiffness matrix A. Set the boundary conditions `'squareb1'`.

   (a) Write a code that implements the Steepest Descent method and solve the linear system $A\boldsymbol{x} = \boldsymbol{b}$.

   (b) Extend the code by preconditioning. Proceed like in the simple code `statit` of Lecture 7: the preconditioner can be given as a matrix or as the product of two matrices.

   (c) Use the code developed in (b) to solve $A\boldsymbol{x} = \boldsymbol{b}$ by the Steepest Descent preconditioned by Jacobi and SSOR(ω).

   (d) Modify your Steepest Descent code to admit a preconditioner provided by a function handle. The solution of $M\boldsymbol{z} = \boldsymbol{r}$ is then executed in a function that you provide. Repeat (d) with this implementation.

2. **Preconditioned Conjugate Gradient (PCG) algorithm.**

   Extend your SD codes to implement the Preconditioned Conjugate Gradient (PCG) algorithm. Repeat steps (a) to (d) with the new code(s). The iteration count should be substantially reduced. Discuss the various convergence behaviours.

   Hint: You can estimate the condition number of sparse matrices by the MATLAB command `condest`.

3. **Convergence rates for steepest descent and CG**

   Assume we have a *diagonal* matrix A of order of $N = 100$, the eigenvalues of which are in the range of $[10^{-p}, 1]$, for $p = 1, 2, 3$, respectively. Since all eigenvalues of A are positive, A is SPD.

   (a) For each p, construct the diagonal matrix A with eigenvalues equally distributed in the range $[10^{-p}, 1]$, i.e., $\lambda_i = 10^{-p} + \frac{i(1-10^{-p})}{N-1}$, $i = 0, 1, \ldots, N-1$. Use both steepest descent and CG method (without preconditioning) to solve $A\boldsymbol{x} = \boldsymbol{b}$ with `b = rand(N,1))` and the starting vector `x0 = rand(N,1)`.

   Build a table to compare the iteration steps or plot the residuals for both methods (Keep other parameters such as tolerance the same). Which method converges faster?

   (b) Now let us consider another distribution of eigenvalues in the range $[10^{-p}, 1]$, i.e., $\lambda_i = 10^{-p} + (1 - 10^{-p})\cos(\frac{i\pi}{2(N-1)})$, $i = 0, 1, \ldots, N-1$. Then follow the same procedure as in (a). Compare the iteration steps of (a) and (b). Which method converges faster? Can you explain?

   Hint: In (b), the eigenvalues are clustered close to 1.

Please submit your solution via e-mail to Peter Arbenz (`arbenz@inf.ethz.ch`) by November 16, 2017. (12:00). Please specify the tag **FEM17** in the subject field.