Exercises for Week 11

1 Conjugate directions (by hand)

Prove that if the nonzero vectors $\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots, \boldsymbol{p}_l$ satisfy

$$\boldsymbol{p}_i^T A \boldsymbol{p}_j = 0, \qquad \text{for all } i \neq j,$$

where A is symmetric and positive definite, then these vectors are linearly independent. (This result implies that A has at most n conjugate directions).

2 Conjugate gradient method (in Matlab)

In this exercise, we will implement the conjugate gradient method and apply it to solve the system $A\boldsymbol{x} = \boldsymbol{b}$, which is a finite difference discretization of the Poisson's equation

$$-\Delta u(x_1, x_2) = f(x_1, x_2), \qquad 0 < x_1, x_2 < 1,$$
$$u(x_1, 0) = u(0, x_2) = u(x_1, 1) = u(1, x_2) = 0.$$

The matrix A is a n^2-by-n^2 block tridiagonal (sparse) matrix from discretizing Poisson's equation with the finite-difference scheme on an n-by-n mesh.

1. Implement the conjugate gradient method. You can use the following Matlab template as the start.

```
function [x,stat]=cgm(A,b,x0)

% Solver settings and info
maxit = 100;
tol   = 1.0e-6;

stat.converged = false;          % converged
stat.iter = 0;                   % number of iterations

% Initial iteration
x = x0;
it = 0;
r = b-A*x;
p = r;
norm_r = norm(r);
converged = false;

% Store data for plotting
```

```
stat.X = x;
stat.resd = norm_r;           % norm of residuals

% Main loop of conjugate gradient
while ~converged && (it < maxit)
    it = it+1;

    % TODO -- implement main loop of CG method
    % ==================================


    % ==================================

    % Set the stopping rule
    converged = (norm_r <= tol);

    % Store data for plotting
    stat.X  = [stat.X  x];
    stat.resd  = [stat.resd norm_r];
end

% Prepare return data
if ~converged
     x = [];
end
stat.converged = converged;
stat.iter = it;
```

2. Call Matlab function `gallery` to generate matrix A:

   ```
   A=gallery('poisson',n);
   ```

   Note that it will generate a n^2-by-n^2 matrix. Set the right-hand side $\boldsymbol{b} = [1, 1, \cdots, 1]^T$ and the starting point $\boldsymbol{x}_0 = [0, 0, \cdots, 0]^T$. Try with $n = 5, 10, 20, 30$ and apply the conjugate gradient method to solve the linear system. Use `semilogy` to plot the residual norm as a function of the iteration number. How many iterations did the CG method need to reach the stopping rule?

3. Use Matlab function `condest` to calculate the condition number of the matrix A with different n, and conclude the relation between the condition number and the number of iterations for the CG method.

4. Download the Matlab function `steepestdescent_line` from the same folder as the exercise list in DTU Inside, where the tolerance for $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$ has been set the same as in your CG method. Now, apply the steepest descent method

with backtracking line search to solve the corresponding minimization problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}. \tag{1}$$

In order to use `steepestdescent_line`, you need write a Matlab function to return the objective function value $f(\boldsymbol{x})$ and the gradient $\nabla f(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b}$ with a given $\boldsymbol{x}$.

5. Compare the results from the steepest descent method with backtracking line search and the CG method. Which method converges faster?

3 Nonlinear CG method (by hand)

Prove that when applied to a quadratic function, with exact line searches, the Polak-Ribière formula given by

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|_2^2}$$

reduce to the Fletcher-Reeves formula

$$\beta_{k+1}^{FR} = \frac{\|\nabla f_{k+1}\|_2^2}{\|\nabla f_k\|_2^2}.$$

4 Limited-memory BFGS method (in Matlab)

In this exercise, we apply the limited-memory BFGS method to solve the same quadratic problem as in Exercise 2.

1. Download the Matlab function `LBFGSmethod` from the same folder as the exercise list in DTU Inside, which implements the limited-memory BGFS method. In order to run this function, you need also download the files `backtracking.m` for the backtracking line search and `getHg_lbfgs.m` for L-BFGS two-loop recursion from the same folder.

2. Apply L-BFGS method to solve the quadratic minimization problem in (1) given in Exercise 2 with $n = 10$.

3. Test different m values and plot the residual norm as a function of the iteration number. How does m influence the convergence?

4. Comparing with the CG method and the steepest descent method, what is your conclusion?

5 Relationship between memoryless BFGS and CG method (by hand, optional)

Consider a memoryless BFGS iteration, i.e., in each iteration we reset H_k back to the identity matrix I to obtain the new inverse Hessian approximation as

$$H_{k+1} = \left(I - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}\right)\left(I - \frac{\boldsymbol{y}_k \boldsymbol{s}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}\right) + \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k},$$

where $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{y}_k = \nabla f_{k+1} - \nabla f_k$. If we use an exact line search, i.e.,

$$\alpha_k^* = \arg\min_{\alpha} f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k),$$

where $\boldsymbol{p}_k = -H_k \nabla f_k$, then prove the following two consequences:

1. $\nabla f_{k+1}^T \boldsymbol{p}_k = 0$ for all k;

2. $\boldsymbol{p}_{k+1} = -\nabla f_{k+1} + \frac{\nabla f_{k+1}^T \boldsymbol{y}_k}{\boldsymbol{y}_k^T \boldsymbol{p}_k} \boldsymbol{p}_k$, which is none other than the Hestenes-Stiefel conjugate gradient method.