Numerical Methods II

Problem Set 6

due in class, April 28, 2004

1. (a) Let T_k denote the Chebychev polynomial of order k on the interval $[-1, 1]$,

$$T_k(x) = \cos(k \arccos x)\,. \tag{1}$$

Show that

$$T_k(x) = \tfrac{1}{2}\left(x + \mathrm{i}\sqrt{1-x^2}\right)^k + \tfrac{1}{2}\left(x - \mathrm{i}\sqrt{1-x^2}\right)^k.$$

(b) Show that, with $\kappa = \lambda_{\max}/\lambda_{\min}$,

$$T_k\left(\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right)^{-1} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k.$$

2. (From *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain* by J.R. Shewchuk.)

Suppose you wish to solve $A\boldsymbol{x} = \boldsymbol{b}$ for a symmetric, positive-definite $n \times n$ matrix A. Unfortunately, the trauma of your linear algebra course has caused you to repress all memories of the Conjugate Gradient algorithm. Seeing you in distress, the Good Eigenfairy materializes and grants you a list of d distinct eigenvalues (but not the eigenvectors) of A. However, you do *not* know how many times each eigenvalue is repeated.

Clever person that you are, you mumbled the following algorithm in your sleep this morning:

```
Choose an arbitrary starting point x(1);
for i = 1:d
  r(i) = b - A*x(i);
  Remove an arbitrary eigenvalue from the list and call it l(i);
  x(i+1) = x(i) + r(i)/l(i);
end
```

No eigenvalue is used twice; on termination, the list is empty.

(a) Show that upon termination of this algorithm, $\boldsymbol{x}_{d+1}$ is the solution to $A\boldsymbol{x} = \boldsymbol{b}$.

(b) Although this routine finds the exact solution after d iterations, you would like each intermediate iterate x_i to be as close to the solution as possible. Give a *crude* rule of thumb for how you should choose an eigenvalue from the list on each iteration. (In other words, in what order should the eigenvalues be used?)

(c) What could go terribly wrong with this algorithm if floating point roundoff occurs?

(d) Give a rule of thumb for how you should choose an eigenvalue from the list on each iteration to prevent floating point roundoff error from escalating.

Hint: The answer is not the same as that of question (b).

3. Recall that the linear Conjugate Gradient method for minimizing the quadratic function $f(\boldsymbol{x}) = \frac{1}{2}\,\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}$ can be written

$$\begin{aligned} \boldsymbol{r}_k &= -\boldsymbol{\nabla} f(\boldsymbol{x}_k)\,, \\ \boldsymbol{d}_k &= \begin{cases} \boldsymbol{r}_1 & \text{for } k = 1 \\ \boldsymbol{r}_k + \beta_k\,\boldsymbol{d}_{k-1} & \text{for } k \geq 2\,, \text{ where } \beta_k = -\dfrac{\boldsymbol{r}_k^T A\boldsymbol{d}_{k-1}}{\boldsymbol{d}_{k-1}^T A\boldsymbol{d}_{k-1}}\,, \end{cases} \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha_k\,\boldsymbol{d}_k\,, \text{ where } \alpha_k = \frac{\boldsymbol{d}_k^T\boldsymbol{r}_k}{\boldsymbol{d}_k^T A\boldsymbol{d}_k}\,. \end{aligned}$$

Show that the following two expressions for β_k are exact in the linear case.

(a) $\beta_k^{\mathrm{FR}} = \dfrac{\boldsymbol{r}_k^T\boldsymbol{r}_k}{\boldsymbol{r}_{k-1}^T\boldsymbol{r}_{k-1}}$ (Fletcher–Reeves)

(b) $\beta_k^{\mathrm{PR}} = \dfrac{\boldsymbol{r}_k^T(\boldsymbol{r}_k - \boldsymbol{r}_{k-1})}{\boldsymbol{r}_{k-1}^T\boldsymbol{r}_{k-1}}$ (Polak–Ribière)

Remark: These two formulae are useful for nonlinear problems, as they do not involve the Hessian matrix of f explicitly. In this case, the two expressions may differ. Fletcher–Reeves converges provided the starting point is sufficiently close to the extremum. Polak–Ribière often converges more quickly, but convergence is only guaranteed with the choice $\beta_k = \max\{0, \beta_k^{\mathrm{PR}}\}$.

4. **Project:** Implement a nonlinear Conjugate Gradient method. Note that the parameter α_k cannot be computed explicitly as in the linear case. Instead, you need to implement a one-dimensional "line search" routine!

5. **Project:** Compare the convergence of Fletcher–Reeves vs. Polak–Ribière when using your nonlinear Conjugate Gradient algorithm to find the minimum of the function

$$f(x_1, x_2) = -\frac{1}{1 + x_1^2 + (1 + 10\,\sin^2 x_1)\,x_2^2}\,.$$

with starting point $(10, 10)$.

Extra credit: Does periodic restart of the CG iteration improve convergence? Can you get rid of the requirement to explicitly compute the gradient of f by using a Broyden-type argument?