CS 475/675 Spring 2025: Crowdmark Assignment 4

Due July 4 at 11:59 pm Eastern.
Submit ALL components of your solutions (written/analytical work, code/scripts, figures, plots, output, etc.) to CrowdMark in PDF form in the section for each question.

Also separately submit a single zip file containing any and all code/scripts you write to the Crowdmark A04 DropBox on LEARN, in runnable format (that is, .m).

For full marks, be sure to show all your work!

1. Let $\{p^i\}$ be a set of A-orthogonal search direction vectors, where A is an SPD matrix. We want to look for x^{k+1} in *all* of these directions. Thus, we write

$$x^{k+1} = x^0 + \sum_{i=0}^{k} \alpha_i p^i.$$

We determine $\{\alpha_i\}$ by minimizing $F(x^{k+1})$, where

$$F(x) \equiv \frac{1}{2} x^T A x - b^T x = \frac{1}{2}(x, x)_A - (b, x),$$

over all search directions.

[4] (a) Show that

$$F(x^{k+1}) = \frac{1}{2}(x^0, x^0)_A + \sum_{i=0}^{k} \alpha_i (x^0, p^i)_A + \frac{1}{2} \sum_{i=0}^{k} \sum_{j=0}^{k} \alpha_i \alpha_j (p^i, p^j)_A - (b, x^0) - \sum_{i=0}^{k} \alpha_i (b, p^i).$$

[2] (b) By using the A-orthogonal property, show that

$$F(x^{k+1}) = \frac{1}{2}(x^0, x^0)_A + \sum_{i=0}^{k} \alpha_i (x^0, p^i)_A + \frac{1}{2} \sum_{i=0}^{k} \alpha_i^2 (p^i, p^i)_A - (b, x^0) - \sum_{i=0}^{k} \alpha_i (b, p^i).$$

[4] (c) To minimize $F(x^{k+1})$, we set $\frac{\partial F}{\partial \alpha_j}(x^{k+1}) = 0$. Show that

$$\alpha_j = \frac{(r^0, p^j)}{(p^j, p^j)_A},$$

where r is the residual. Thus α_j depends only on p^j, not on any other search directions. Once we have minimized in direction p^j, we are done with that direction. In other words, each of the p^j minimizes $F(x^{k+1})$ in a subspace and we **never** have to look in that subspace again.

2. Consider the least squares problem $Ax = b$ where

$$A = \begin{bmatrix} 3 & -3 & -10 \\ 0 & 4 & 16 \\ 4 & -4 & -30 \\ 0 & 3 & 12 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

[5] (a) Solve the least squares problem using the normal equations. Do the computation by hand, and not in Matlab.

[5] (b) Solve the least squares problem using (classical or modified) Gram-Schmidt. Determine the $\hat{Q}$ and $\hat{R}$ factors. Do the computation by hand, and not in Matlab.

[10] 3. Adapt the ideas of Householder QR-factorization to derive a method to instead compute a factorization $A = QL$, where L is **lower** triangular and Q is orthogonal. Assume that A is square and full-rank. Give a text description of how your algorithm works, supported by illustrations and pseudocode. (Hint: Derive a modification of the Householder approach such that $\left(I - 2\frac{vv^T}{v^Tv}\right)x$ is zero everywhere but its **last** component, rather than its first.)

4. In this question, compute each of the specified QR factorizations by hand, and not in Matlab. Show all of your work.

[5] (a) Let $A = \begin{bmatrix} 4 & 2 \\ -8 & -4+9\sqrt{2} \\ 8 & 4+9\sqrt{2} \end{bmatrix}$. Compute a QR factorization for A, in which Q is 3×3 and R is 3×2. Use Householder reflections to do the computation.

[5] (b) Let $A = \begin{bmatrix} 1 & 2 \\ 1 & -2 \\ 0 & 1 \end{bmatrix}$. Compute a QR factorization for A, in which Q is 3×3 and R is 3×2. Use Givens rotations to do the computation.

5. In this question, develop a Matlab function to **efficiently** compute each of the specified QR factorizations. Include a diary which shows running each function to factorize an example matrix A, of size at least 3×2.

[10] (a) Develop a Matlab function, QRHouseholder(A), which uses Householder reflections to compute a QR factorization of A. If A is $m \times n$ $(m \geq n)$, then Q will be $m \times m$ and R will be $m \times n$. You may find it helpful to start with the QRHouseholderTemplate.m template.

[10] (b) Develop a Matlab function, QRGivens(A), which uses Givens rotations to compute a QR factorization of A. If A is $m \times n$ ($m \geq n$), then Q will be $m \times m$ and R will be $m \times n$. You may find it helpful to start with the QRGivensTemplate.m template.

Once again, submit a copy of everything to CrowdMark in PDF/image format, and separately submit your [runnable] source code to the A04 LEARN DropBox in a single ZIP file.