CS 475/675 Spring 2015: Assignment 1

Due June 15, 2015, 5:00pm.
Written/analytical work should be submitted to the physical assignment box on 4th floor MC (Q1-4).
MATLAB code and any associated output and commentary should be submitted to the DropBox on LEARN (Q5).
Everyone should do Q1, 2, 4,and 5. As indicated, CS475 does Q3U, CS675 does Q3G.

1. (10 marks) Let $a_{ij}^{(k-1)}$ be the entries of A after $(k-1)$ steps of Gaussian elimination. Suppose $A^{(k-1)}$ is **column** diagonally dominant; i.e.

$$|a_{ii}^{(k-1)}| \quad > \quad \sum_{j \geq k, j \neq i} |a_{ji}^{(k-1)}| \qquad i = k, \ldots, n.$$

   If Gaussian elimination without pivoting is used; i.e.

$$a_{ji}^{(k)} = a_{ji}^{(k-1)} - \frac{a_{jk}^{(k-1)} a_{ki}^{(k-1)}}{a_{kk}^{(k-1)}} \qquad j = k+1, \ldots, n, \;\; i = k+1, \ldots, n,$$

   prove that

$$|a_{ii}^{(k)}| - \sum_{j \geq k+1, j \neq i} |a_{ji}^{(k)}| \quad > \quad 0 \qquad i = k+1, \ldots, n.$$

   i.e. the submatrix $A^{(k)}$ is also column diagonally dominant. (Hint: pay very careful attention to the precise range of the index j. Use the fact that: $\sum_{j \geq k+1, j \neq i} |a_{ji}^{(k)}| = \sum_{j \geq k+1, j \neq i} |a_{ji}^{(k-1)} - \frac{a_{jk}^{(k-1)} a_{ki}^{(k-1)}}{a_{kk}^{(k-1)}}|$ and then apply the triangle inequality.)

2. (15 marks) Let A be a strictly **row** diagonally dominant matrix; i.e.

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|.$$

   (a) Show that Jacobi method converges.
   (Hint: you could use the result that $\rho(J) \leq \|J\|_\infty$ for any matrix J. Recall the definition of infinity norm:

$$\begin{aligned} \|x\|_\infty &= \max_i |x_i|, \\ \|J\|_\infty &= \max_i \sum_{j=1}^{n} |J_{ij}|. \end{aligned}$$

   Consider J = iteration matrix of the Jacobi method.)

   (b) Let G be the iteration matrix of the Gauss-Seidel method; i.e.

$$G = I - (D - L)^{-1} A$$

   where $D - L$ is the lower triangular part of A. Suppose x is any vector with $\|x\|_\infty = 1$, and let $y = Gx$. Show that $\|y\|_\infty < 1$. (Hint: consider $(D - L)y$.)

(c) By definition of matrix norm,

$$\|G\|_\infty \equiv \max_{\|x\|_\infty=1} \|Gx\|_\infty.$$

Use part (b), show that the Gauss-Seidel method converges.

3U. [CS475 students only] (10 marks) Let $\{p^i\}$ be a set of A-orthogonal search direction vectors. 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.

(a) Show that

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

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

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

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

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

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.

3G. [CS675 students only] (10 marks) Let A be a symmetric positive definite matrix. Consider solving $Ax = b$ using conjugate gradient with $x^0 = 0$.

(a) Suppose $b = v_1$ where v_1 is an eigenvector of A; i.e.

$$Av_1 = \lambda_1 v_1,$$

where λ_1 is the corresponding eigenvalue. Verify by direct computation that CG converges in one iteration.

(b) Suppose $b = \sum_{k=1}^{m} v_k$ where v_k are eigenvectors of A corresponding to the eigenvalues λ_k. Assume the eigenvalues are distinct. What is the exact solution of $Ax = b$.

(c) For the right-hand side in part (b), show that CG converges in m iterations. (Hint: note that $r^0 = b$. Consider the subspace that r^0 belongs to in terms of $\{v_k\}$. What are the subspace $span\{r^0, Ar^0, \ldots, A^m r^0\}$ and its dimension?)

4. (10 marks) Consider the least squares problem $Ax = b$ where

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

   (a) Solve the least squares problem using the normal equations.

   (b) Solve the least squares problem using (classical or modified) Gram-Schmidt. Determine the $\hat{Q}$ and $\hat{R}$ factors.

5. (40 marks) In PDE-based image processing, the simple diffusion approach produces images with smeared edges. Employing the total variation regularization approach, images with sharp edges can be restored by solving the following equation repeatedly (for $k \geq 0$):

$$-\alpha \nabla \cdot \tfrac{1}{|\nabla u^k|} \nabla u^{k+1}(x,y) + u^{k+1}(x,y) \;=\; u^0(x,y) \qquad \text{in } \Omega = (0,1) \times (0,1), \tag{1}$$

where $\nabla u = (\partial u/\partial x, \partial u/\partial y)$ and $|\nabla u| = \sqrt{(\partial u/\partial x)^2 + (\partial u/\partial y)^2}$. $\alpha > 0$ is the parameter controlling how much noise is to be removed. Figure 1 shows a reconstructed image after 20 iterations.

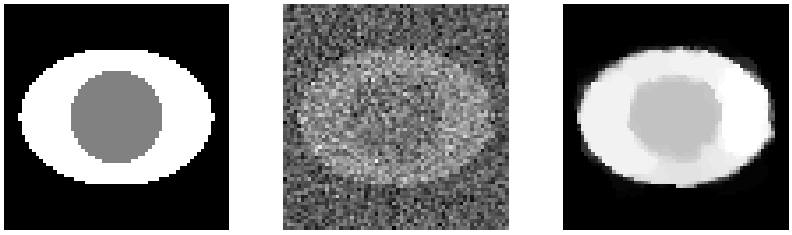

Figure 1: (Left) original, (middle) noisy, and (right) denoised images.

Applying a standard finite difference discretization, at each grid point (x_i, y_j), we have

$$AC u_{i,j}^{k+1} + AW u_{i-1,j}^{k+1} + AE u_{i+1,j}^{k+1} + AS u_{i,j-1}^{k+1} + AN u_{i,j+1}^{k+1} = u_{i,j}^0, \tag{2}$$

where

$$\begin{aligned}
AW &= -\frac{\alpha}{h^2}\left(\frac{1}{2\sqrt{(\frac{u_{i,j}^k - u_{i-1,j}^k}{h})^2 + (\frac{u_{i,j}^k - u_{i,j-1}^k}{h})^2 + \beta}} + \frac{1}{2\sqrt{(\frac{u_{i,j}^k - u_{i-1,j}^k}{h})^2 + (\frac{u_{i-1,j+1}^k - u_{i-1,j}^k}{h})^2 + \beta}}\right) \\
AE &= -\frac{\alpha}{h^2}\left(\frac{1}{2\sqrt{(\frac{u_{i+1,j}^k - u_{i,j}^k}{h})^2 + (\frac{u_{i+1,j}^k - u_{i+1,j-1}^k}{h})^2 + \beta}} + \frac{1}{2\sqrt{(\frac{u_{i+1,j}^k - u_{i,j}^k}{h})^2 + (\frac{u_{i,j+1}^k - u_{i,j}^k}{h})^2 + \beta}}\right) \\
AS &= -\frac{\alpha}{h^2}\left(\frac{1}{2\sqrt{(\frac{u_{i,j}^k - u_{i-1,j}^k}{h})^2 + (\frac{u_{i,j}^k - u_{i,j-1}^k}{h})^2 + \beta}} + \frac{1}{2\sqrt{(\frac{u_{i+1,j-1}^k - u_{i,j-1}^k}{h})^2 + (\frac{u_{i,j}^k - u_{i,j-1}^k}{h})^2 + \beta}}\right) \\
AN &= -\frac{\alpha}{h^2}\left(\frac{1}{2\sqrt{(\frac{u_{i+1,j}^k - u_{i,j}^k}{h})^2 + (\frac{u_{i,j+1}^k - u_{i,j}^k}{h})^2 + \beta}} + \frac{1}{2\sqrt{(\frac{u_{i,j+1}^k - u_{i-1,j+1}^k}{h})^2 + (\frac{u_{i,j+1}^k - u_{i,j}^k}{h})^2 + \beta}}\right) \\
AC &= -(AW + AE + AS + AN) + 1.
\end{aligned}$$

Here h is mesh size, and $\beta = 10^{-6}$ is a constant parameter. Conceptually, the finite difference equation (2) corresponds to solving the linear system:

$$A(u^k)u^{k+1} = u^0, \tag{3}$$

where the coefficient matrix $A(u^k)$ depends on the current values of u^k and its nonzero structure is the same as the standard 5-point stencil 2D Laplacian matrix. (Assume zero boundary conditions.)

To summarize, the algorithm of the image denoising process is:

Given noisy image u^0.
for $k = 0, 1, \ldots, K$
    Solve $A(u^k)\, u^{k+1} = u^0$ for u^{k+1}
end

To solve equation (3), we will apply different iterative methods. Let $u^{k+1,l}$ be the approximate solution of u^{k+1} given by l iterations of an iterative method. Then the denoising algorithm can be written as:

Given noisy image u^0.
for $k = 0, 1, \ldots, K$
    $u^{k+1,0} = u^k$
    for $l = 0, 1, \ldots$, until convergence
        $u^{k+1,l+1} = \text{IterativeMethodStep}(u^{k+1,l}, A(u^k), u^0)$
    end
end

For this assignment, use $K = 10$, and $\alpha = 4 \times 10^{-2}, 3 \times 10^{-2}, 1.5 \times 10^{-2}$, and 1.2×10^{-2} for the image sizes $16 \times 16, 32 \times 32, 64 \times 64$, and 128×128, respectively.

(a) Create two MATLAB functions:

```
A = ImageMatrix(u)
u0 = ImageRHS(X)
```

The input for `ImageMatrix` is an approximate solution u and the output is the image matrix (equation (2)). The input for `ImageRHS` is a noisy image X and the output is the vector representation $u0$ of X. Note that if the image size of X is $m \times m$, then the size of A should be $n \times n$ where $n = m^2$ and the size of $u0$ is $n \times 1$. Noisy images of different sizes can be generated by `set_image` which can be downloaded from the class homepage. Submit a listing of your code.

(b) Implement the iterative methods: Jacobi, Gauss-Seidel, SOR and Conjugate Gradient. Create the following MATLAB functions:

```
[x,iter] = Jacobi(A,b,x_initial,maxiter,tol)
[x,iter] = GS(A,b,x_initial,maxiter,tol)
[x,iter] = SOR(omega,A,b,x_initial,maxiter,tol)
[x,iter] = CG(A,b,x_initial,maxiter,tol)
```

These MATLAB functions take as inputs the sparse matrix A, the right-hand size b, the initial guess $x\_initial$, the maximum number of iterations $maxiter$ and the tolerance tol, and compute the approximate solution x using $iter$ steps of the corresponding iterative method. (SOR has one more input parameter, $omega$.) You may use MATLAB's built-in backslash operator ('\') to perform the triangular solve in the inner loop of Gauss-Seidel and SOR.

The iteration should be stopped either when it reaches the maximum number of iterations or when the residual vector satisfies:

$$\|r^l\|_2 \leq tol\,\|b\|_2.$$

Submit all your code.

(c) Create a MATLAB script, `Denoise`, that solves (3) using different iterative methods. In particular, for each k, set up the matrix A and right-hand side $b = u0$ by calling `ImageMatrix` and `ImageRHS`. Then solve the linear system by calling one of the iterative methods in part (b). (Hint: for easier debugging, you may want to temporarily use MATLAB's backslash operator, `A\b`, for the inner loop/solve first to make sure your outer loop and the two functions, `ImageMatrix` and `ImageRHS`, are done correctly. Then replace `A\b` by an iterative method.)

In this assignment, use a relatively large tolerance, $tol = 10^{-2}$. In many practical scenarios, one usually uses a much smaller tolerance, e.g. 10^{-6} or 10^{-8}. However, for image denoising, the result is not visually distinguishable, and crucially, this can dramatically reduce the number of iterations required. Also, impose a maximum number of iterations, say, 100000. For SOR, determine the optimal $omega$ for each grid size to 2 decimal places by trial-and-error. Report the optimal values.

Record the CPU times and number of iterations. Construct a table of execution times and a table of total number of iterations for solving (1) using Jacobi, Gauss-Seidel, SOR and Conjugate Gradient iterations. Use image sizes 16×16, 32×32, 64×64, (and optionally, 128×128, if time permits). Comment on the results. Convert the output from an array back to a matrix, and use `imshow` or `imagesc` to display the denoised image. Submit: the plots of the denoised images for the grid sizes specified above, the table you constructed, your comments, and the code for `Denoise`.