CS 475/675 Spring 2022: MATLAB Assignment 2

Due June 23 at 11:59 pm EDT.
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.

You must also separately submit a single zip file containing any and all code/scripts you write to the MATLAB Assignment 2 DropBox on LEARN, in runnable format (that is, .m).

Everyone does Q1,2,4, and 5. As indicated, CS475 does Q3U, CS675 does Q3G.

You might find these variants of the **triangle inequality** handy for Q1 and/or Q2:

$$|a| - |b| \leq |a + b| \leq |a| + |b|,$$

$$|a - b| \geq ||a| - |b||.$$

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

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 **strictly column** diagonally dominant; i.e.

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

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

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

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

$$\sum_{j \geq k+1, j \neq i} \left|a_{ji}^{(k)}\right| = \sum_{j \geq k+1, j \neq i} \left| a_{ji}^{(k-1)} - \frac{a_{jk}^{(k-1)} a_{ki}^{(k-1)}}{a_{kk}^{(k-1)}} \right|$$

by triangle

$$\leq \sum_{j \geq k+1, j \neq i} \left(\left| a_{ji}^{(k-1)} \right| 1 + \left| \frac{a_{jk}^{(k-1)} a_{ki}^{(k-1)}}{a_{kk}^{(k-1)}} \right| \right)$$

$$< \sum_{j \geq k, j \neq i} \left(\left| a_{ji}^{(k-1)} \right| + \left| a_{ki}^{(k-1)} \right| \right)$$

by $\frac{a_{jk}^{(k-1)}}{a_{kk}^{(k-1)}} < 1$

$$= \sum_{\substack{j \geq k \\ j \neq i}} \left| a_{ji}^{(k-1)} \right|$$

some thing

by diagnal dominand

$$< \left| a_{ii}^{(k)} \right|$$

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

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

(a) As we did in the lectures, let $A = D - L - U$, where D is the diagonal, $-L$ contains only the entries below the diagonal of A, and $-U$ is the entries above the diagonal. Let G be the iteration matrix of the Gauss-Seidel method.

Suppose x and y are vectors such that $y = Gx$. Show that $(D - L)y = Ux$.

(b) Writing out the expression in componentwise form yields

$$a_{jj}y_j + \sum_{i<j} a_{ji}y_i = -\sum_{i>j} a_{ji}x_i. \tag{1}$$

Recall that $\|y\|_\infty = \max_j |y_j|$, and consider the row j corresponding to the largest magnitude entry. Show that the absolute value of the left side of the equality (1) above satisfies

$$\left| a_{jj}y_j + \sum_{i<j} a_{ji}y_i \right| \geq \left(|a_{jj}| - \sum_{i<j} |a_{ji}| \right) \|y\|_\infty.$$

(c) Suppose now that x is a vector with infinity norm of 1. Considering the right side of the earlier equality (1), show

$$\left| \sum_{i>j} a_{ji}x_i \right| \leq \left(\sum_{i>j} |a_{ji}| \right).$$

(d) Parts (b)-(c) together imply that

$$\left(|a_{jj}| - \sum_{i<j} |a_{ji}| \right) \|y\|_\infty \leq \left(\sum_{i>j} |a_{ji}| \right).$$

Recalling that an equivalent definition of the matrix infinity norm is

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

show that Gauss-Seidel converges if the matrix A is row-diagonally dominant, using the fact that $\rho(C) \leq \|C\|_\infty$ for any matrix C.

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

(a) As we did in the lectures, let $A = D - L - U$, where D is the diagonal, $-L$ contains only the entries below the diagonal of A, and $-U$ is the entries above the diagonal. Let G be the iteration matrix of the Gauss-Seidel method.
Suppose x and y are vectors such that $y = Gx$. Show that $(D - L)y = Ux$.

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

$M = D - L$

$(D-L)y = (D-L)Gx = (D-L)(I - (D-L)^{-1}A)x$

$= (D-L)(x - (D-L)^{-1}Ax)$

$= (D-L)x - Ax$

$= (D-L-A)x$

$= Ux$

(b) Writing out the expression in componentwise form yields

$$a_{jj}y_j + \sum_{i<j} a_{ji}y_i = -\sum_{i>j} a_{ji}x_i. \tag{1}$$

Recall that $\|y\|_\infty = \max_j |y_j|$, and consider the row j corresponding to the largest magnitude entry. Show that the absolute value of the left side of the equality (1) above satisfies

$$\left| a_{jj}y_j + \sum_{i<j} a_{ji}y_i \right| \geq \left(|a_{jj}| - \sum_{i<j} |a_{ji}| \right) \|y\|_\infty.$$

$$\left| a_{jj} y_j + \sum_{i<j} a_{ji} y_i \right|$$

since triangular

$$\geq \left| a_{jj} y_j \right| - \sum_{i<j} \left| a_{ji} y_i \right|$$

since y_j is largest in y

$$\geq \left| a_{jj} y_j \right| - \sum_{i<j} \left| a_{ji} y_j \right|$$

since $|y_j|$ is just a number

$$= \left(|a_{jj}| - \sum_{i<j} |a_{ji}| \right) |y_j|$$

(c) Suppose now that x is a vector with infinity norm of 1. Considering the right side of the earlier equality (1), show

$$\left| \sum_{i>j} a_{ji} x_i \right| \leq \left(\sum_{i>j} |a_{ji}| \right).$$

$$\left| \sum_{i<j} a_{ji} x_i \right| \leq \sum_{i<j} |a_{ji}| |x_i| \leq \sum_{i<j} |a_{ji}|$$

by triangular

by $\|x\|_\infty = 1$, $x_i \leq 1$

(d) Parts (b)-(c) together imply that

$$\left(|a_{jj}| - \sum_{i<j} |a_{ji}|\right) \|y\|_\infty \leq \left(\sum_{i>j} |a_{ji}|\right).$$

Recalling that an equivalent definition of the matrix infinity norm is

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

show that Gauss-Seidel converges if the matrix A is row-diagonally dominant, using the fact that $\rho(C) \leq \|C\|_\infty$ for any matrix C.

converge $\leftrightarrow$ $\rho(I - M^{-1}A) < 1$

$\rho(I - M^{-1}A) \leq \| I - M^{-1}A \|_\infty$

consider $\| (I - M^{-1}A)x \|_\infty$

$= \|y\|_\infty \leq \dfrac{\sum_{i>j} |a_{ji}|}{|a_{jj}| - \sum_{i<j} |a_{ji}|}$

since $\sum_{i>j} |a_{ji}| + \sum_{i<j} |a_{ji}| < |a_{jj}|$,

$\sum_{i>j} |a_{ji}| < |a_{jj}| - \sum_{i<j} |a_{ji}|$,

$\dfrac{\sum_{i>j} |a_{ji}|}{|a_{jj}| - \sum_{i<j} |a_{ji}|} < 1$

so $\|y\|_\infty < 1$ $\|G\|_\infty < 1$ as required. so converge

3U. [*CS475 students only*] **(10 marks)** 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.

(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

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

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.

$$F(x^{k+1}) = F\left(x^0 + \sum_{i=0}^{k} \alpha_i p^i\right)$$

$$= \frac{1}{2}\left[\left(x^{0T} + \sum_{i=0}^{k} \alpha_i p^{iT}\right) A \left(x^0 + \sum_{i=0}^{k} \alpha_i p^i\right)\right] - b^T\left(x^0 + \sum_{i=0}^{k} \alpha_i p^i\right)$$

$$= \frac{1}{2}\left[x^{0T} A x^0 + \sum_{i=0}^{k} \alpha_i p^{iT} A x^0 + \sum_{i=0}^{k} \alpha_i x^{0T} A p^i + \sum_{i=0}^{k} \sum_{j=0}^{k} \alpha_i \alpha_j p_i^T A p^j\right] - b^T x^0 - \sum_{i=0}^{k} \alpha_i b^T p^i$$

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

(b) we just show $\sum_{i=0}^{k} \sum_{j=0}^{k} \alpha_i \alpha_j (p^i, p^j)_A = \sum \alpha_i^2 (p^i, p^i)_A$

for $i \neq j$,

$\alpha_i \alpha_j (p^i, p^j)_A = 0$ by A-orthogonality

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

taking derivative over α_i,

$$0 = \frac{dF(x^{k+1})}{d\alpha_i} = (x^0, p^i)_A + \alpha_i (p^i, p^i)_A - (b, p^i)$$

$$\alpha_i = \frac{(x^0, p^i)_A - (b, p^i)}{(p^i, p^i)_A}$$

$$\frac{x^{0T} A p^i - b^T p^i}{p^{iT} p^i}$$

$$= \frac{(x^{0T} A - b^T) p^i}{p^{iT} p^i}$$

$$= \frac{r^0 p^i}{p^{iT} p^i}$$ as reqved

3G. [*CS675 students only*] **(10 marks)** Let A be an SPD 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?)

A^T $\begin{bmatrix} 3 & 0 & 4 & 0 \\ -3 & 4 & -4 & 3 \\ -10 & 16 & -30 & 12 \end{bmatrix}$

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.

(a) $A^TAx = A^Tb = \begin{bmatrix} 7 \\ 0 \\ -12 \end{bmatrix}$

$$A^TA = \begin{bmatrix} 25 & -25 & -150 \\ -25 & 50 & 250 \\ -150 & -250 & 1400 \end{bmatrix}$$

we do Cholesky on A^TA

$L = ($ $a_{11} = 5$

$a_{21} = -5$

$a_{31} = -30$

$a_{22} = a_{22} - a_{21} * a_{21} = 50 - 5^2 = 25$

$a_{32} = a_{32} - a_{21} * a_{31} = 250 - (-5)(-30) = 100$

$a_{33} = a_{33} - a_{31}^2 = 1400 - 30^2 = 500$

$\begin{bmatrix} 5 & & \\ -5 & 25 & \\ -30 & 100 & 500 \end{bmatrix}$

$k=2$ $\quad a_{22}=5$

$a_{32}=20$

$a_{33}=a_{33}-a_{32}^2=100$

$$\begin{bmatrix} 5 & & \\ -5 & 5 & \\ -30 & 20 & 100 \end{bmatrix}$$

$k=3$ $\quad a_{33}=\sqrt{100}=10$

$$\begin{bmatrix} 5 & & \\ -5 & 5 & \\ -30 & 20 & 10 \end{bmatrix}$$

G

G^T

$$\begin{pmatrix} 5 & -5 & -30 \\ & 5 & 20 \\ & & 10 \end{pmatrix}$$

$$GG^T = A^TA$$

$$GG^TX = \begin{bmatrix} 7 \\ 0 \\ 12 \end{bmatrix}$$

$$\begin{bmatrix} 5 & & \\ -5 & 5 & \\ -30 & 20 & 10 \end{bmatrix} \begin{pmatrix} 5 & -5 & -30 \\ & 5 & 20 \\ & & 10 \end{pmatrix} \begin{bmatrix} x \\ \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 12 \end{bmatrix}$$

G $\qquad$ G^T

$$\underbrace{\begin{bmatrix} 5 & & \\ -5 & 5 & \\ -30 & 20 & 10 \end{bmatrix}}_{G} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{bmatrix} 7 \\ 0 \\ 12 \end{bmatrix}$$

$c_1 = \frac{7}{5}$

$$X = \begin{bmatrix} 7 \\ 0 \\ 12 \end{bmatrix} - \begin{bmatrix} 7 \\ -7 \\ -42 \end{bmatrix} = \begin{bmatrix} 0 \\ 7 \\ 54 \end{bmatrix}$$

$c_2 = \frac{7}{5}$

$$X = \begin{bmatrix} 0 \\ 7 \\ 54 \end{bmatrix} - \begin{bmatrix} 0 \\ 7 \\ 28 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 26 \end{bmatrix}$$

$$c_3 = \frac{26}{10}$$

$$C = \begin{bmatrix} \frac{7}{5} \\ \frac{7}{5} \\ \frac{13}{5} \end{bmatrix}$$

$$\underbrace{\begin{pmatrix} 5 & -5 & -30 \\ & 5 & 20 \\ & & 10 \end{pmatrix}}_{G^T} X = \begin{bmatrix} \frac{7}{5} \\ \frac{7}{5} \\ \frac{13}{5} \end{bmatrix}$$

$$X_3 = \frac{13}{50}$$

$$y = \begin{bmatrix} 1 \\ \frac{1}{5} \\ \frac{2}{5} \end{bmatrix} - \begin{bmatrix} \frac{-39}{5} \\ \frac{26}{5} \end{bmatrix} = \begin{bmatrix} \frac{46}{5} \\ \frac{-19}{5} \\ 0 \end{bmatrix}$$

$$x_2 = \frac{-19}{25}; \quad y = \begin{bmatrix} \frac{46}{5} \\ x \\ x \end{bmatrix} - \begin{bmatrix} \frac{19}{5} \\ x \\ x \end{bmatrix} = \begin{bmatrix} \frac{27}{5} \\ x \\ x \end{bmatrix}$$

$$x_1 = \frac{27}{25}$$

$$x = \begin{bmatrix} \frac{27}{25} \\ \frac{-19}{25} \\ \frac{13}{50} \end{bmatrix}$$

(5)

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

$$v_1 = \begin{matrix} 3 \\ 0 \\ 4 \\ 0 \end{matrix} \quad r_{11} = \frac{1}{5} \quad q_1 = \begin{bmatrix} \frac{3}{5} \\ 0 \\ \frac{4}{5} \\ 0 \end{bmatrix}$$

$$v_2 = \begin{matrix} -3 \\ 4 \\ -4 \\ 3 \end{matrix} \qquad r_{12} = q_1^T \cdot a_2 = -\frac{9}{5} - \frac{16}{5} = -5$$

$$v_2 = \begin{matrix} -3 \\ 4 \\ -4 \\ 3 \end{matrix} + 5 \cdot \begin{bmatrix} \frac{3}{5} \\ 0 \\ \frac{4}{5} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 0 \\ 3 \end{bmatrix}$$

$$r_{22} = 5 \qquad q_2 = \begin{bmatrix} 0 \\ \frac{4}{5} \\ 0 \\ \frac{3}{5} \end{bmatrix}$$

$$v_3 = \begin{bmatrix} -10 \\ 16 \\ -30 \\ 12 \end{bmatrix} \qquad r_{13} = \frac{3}{5}\cdot(-10) + \frac{4}{5}\cdot(-30) = -30$$

$$v_3 = \begin{bmatrix} -10 \\ 16 \\ -30 \\ 12 \end{bmatrix} + (+30)\cdot\begin{bmatrix} \frac{3}{5} \\ 0 \\ \frac{4}{5} \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 8 \\ 16 \\ -6 \\ 12 \end{bmatrix}$$

$$r_{23} = \frac{4}{5}\cdot 16 + \frac{3}{5}\cdot 12 = 20$$

$$v_3 = \begin{bmatrix} 8 \\ 16 \\ -6 \\ 12 \end{bmatrix} - 20\cdot\begin{bmatrix} 0 \\ \frac{4}{5} \\ 0 \\ \frac{3}{5} \end{bmatrix}$$

$$= \begin{bmatrix} 8 \\ 0 \\ -6 \\ 0 \end{bmatrix}$$

$$r_{33} = 10$$

$$q_3 = \begin{bmatrix} \frac{4}{5} \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{3}{5} \\ 0 \\ \frac{4}{5} \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ \frac{4}{5} \\ 0 \\ \frac{3}{5} \end{bmatrix} \quad \begin{bmatrix} \frac{4}{5} \\ 0 \\ -\frac{3}{5} \\ 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} \frac{3}{5} & 0 & \frac{4}{5} \\ 0 & \frac{4}{5} & 0 \\ \frac{4}{5} & 0 & -\frac{3}{5} \\ 0 & \frac{3}{5} & 0 \end{bmatrix}$$

$QR = A$

$$R = Q^{-1}A = Q^{T}A = \begin{bmatrix} \frac{3}{5} & 0 & \frac{4}{5} & 0 \\ 0 & \frac{4}{5} & 0 & \frac{3}{5} \\ \frac{4}{5} & 0 & -\frac{3}{5} & 0 \end{bmatrix} \begin{bmatrix} 3 & -3 & -10 \\ 0 & 4 & 16 \\ 4 & -4 & -30 \\ 0 & 3 & 12 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & -5 & -30 \\ 0 & 5 & 20 \\ 0 & 0 & 10 \end{bmatrix}$$

$$QRX = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$$RX = Q^{T} \cdot \begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \frac{7}{5} \\ \frac{7}{5} \\ \frac{1}{5} \end{bmatrix} = b$$

$$X_3 = \frac{1}{50} \qquad b = \begin{bmatrix} \frac{7}{5} \\ \frac{7}{5} \\ x \end{bmatrix} - \begin{bmatrix} -\frac{3}{5} \\ \frac{2}{5} \\ x \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ x \end{bmatrix}$$

$$X_2 = \frac{1}{5} \qquad b = \begin{bmatrix} 2 \\ x \\ x \end{bmatrix} - \begin{bmatrix} -1 \\ x \\ x \end{bmatrix} = \begin{bmatrix} 3 \\ x \\ x \end{bmatrix}$$

$$X_1 = \frac{3}{5}$$

$$X = \begin{bmatrix} \frac{3}{5} \\ -\frac{1}{5} \\ -\frac{1}{50} \end{bmatrix}$$

5. **(40 marks)** In PDE-based image denoising, using a simple Laplacian/diffusion approach produces overly-smoothed images that fail to preserve important edges in the data. Approaches based on "total variation regularization" tend to fare much better. This can be expressed by solving the following equation, repeatedly (for $k \geq 0$):

$$-\alpha \nabla \cdot \frac{1}{|\nabla u^k|} \nabla u^{k+1}(x,y) + u^{k+1}(x,y) \quad = \quad u^0(x,y) \qquad \text{in } \Omega = (0,1) \times (0,1), \tag{2}$$

where we will use $\nabla u = (\partial u/\partial x, \partial u/\partial y)$ and $|\nabla u| = \sqrt{(\partial u/\partial x)^2 + (\partial u/\partial y)^2}$. The parameter $\alpha > 0$ controls the degree of smoothing to be applied. Figure 1 shows an example of an input image, the same image corrupted with noise, and a denoised image using the total variation approach.

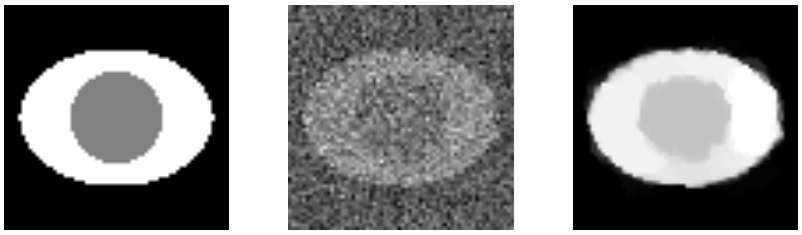

Figure 1: (Left) original, (middle) noisy, and (right) denoised images.

One possible finite difference discretization of the PDE described above gives the following equation at each grid point (x_i, y_j):

$$AC u_{i,j}^{k+1} + AW u_{i-1,j}^{k+1} + AE u_{i+1,j}^{k+1} + AS u_{i,j-1}^{k+1} + AN u_{i,j+1}^{k+1} = u_{i,j}^0, \tag{3}$$

where

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

Note that the coefficients above depend on the data at step k rather than $k+1$. Here h is grid cell size (width), and $\beta = 10^{-6}$ is a constant parameter to circumvent numerical issues (division by zero). The finite difference equations can be formulated into a linear system of the form

$$A(u^k) u^{k+1} = u^0, \tag{4}$$

where the coefficient matrix $A(u^k)$ depends on the current values of u^k. The matrix's sparsity pattern of nonzeros is the same as the standard 5-point stencil 2D Laplacian matrix. You can assume zero boundary conditions.

Our basic algorithm for image denoising process will be:

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

where K is some specified number of iterations of denoising to apply. To actually solve equation (4) inside the loop, we will apply several of the iterative methods we have seen in class. 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 question, we will use $K = 8$ denoising iterations. For α, we will use $\alpha = 6.4 \times 10^{-2}, 3.2 \times 10^{-2}$, 1.6×10^{-2}, and 8×10^{-3} for image sizes of $16 \times 16, 32 \times 32$, 64×64, and 128×128, respectively.

(a) Create two MATLAB functions:

```
A = FormMatrix(u, alpha)
u0 = FormRHS(X)
```

The input for `FormMatrix` is an approximate solution vector u (corresponding to the u^k terms in the coefficients for equation (2)) and the output is the matrix corresponding to equation (2). Be sure to make use of Matlab's sparse matrix functionality! The input for `FormRHS` 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 the provided Matlab file `set_image`, which takes an integer specifying the resolution along one dimension. (You may assume the image is always square.)

(b) Implement the following iterative methods: Jacobi, Gauss-Seidel, SOR, and Conjugate Gradient, by creating 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 functions take as inputs the sparse matrix A, the right-hand side b, the initial guess *x_initial*, the maximum number of iterations *maxiter*, and the tolerance *tol*, and computes 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 necessary triangular solves in the inner loops of Gauss-Seidel and SOR. For Jacobi, you may explicitly construct and use D^{-1} (i.e., breaking our usual rule of not inverting matrices).

The solvers should be stopped either when they reach the maximum number of solver iterations or when the residual vector satisfies:

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

(c) Create a MATLAB script, `Denoise`, that solves the system (3) to perform denoising, using different iterative methods. In particular, for each iteration k, set up the matrix A and right-hand side $b = u0$ by calling `FormMatrix` and `FormRHS`. Then solve the linear system by calling one of the iterative methods in part (b). (Tip: for easier debugging, you may want to temporarily use MATLAB's backslash operator, `A\b`, for the solve first to make sure your outer loop and the two functions, `FormMatrix` and `FormRHS`, are correct, i.e., get the overall denoising working before implementing your own linear solvers. Then, replace `A\b` with the various iterative methods.)

In this assignment, use a relatively large solver tolerance, $tol = 10^{-2}$. Impose a maximum of 20000 solver iterations (note: this is separate from the outer/denoising iterations). For SOR, determine the optimal *omega* (requiring the fewest *total* iterations for the whole denoising process) for each grid size up to 2 decimal places by trial-and-error (by hand or by code) for the three image sizes specified below. Report the optimal values you found.

Record the CPU times and number of iterations. Construct a table of execution times and a table of *total* number of iterations for the denoising process when using Jacobi, Gauss-Seidel, SOR (with your optimal omega values), and Conjugate Gradient iterations. Use image sizes 16×16, 32×32, and 64×64 (and *optionally*, 128×128, if you would like and time permits). Comment on the results. Convert the output vector back to a matrix (2D image), and use `imagesc` with a grayscale color map to display the denoised image.

Submit:

- Your code for `FormMatrix, FormRHS, Jacobi, GS, SOR, CG` and `Denoise`;
- Your list of optimal *omega* values.
- The plots of the denoised images for the grid sizes specified above;
- Your table of CPU times and iteration counts;
- Comments on your observations.

Once again, a copy of your code should be submitted to CrowdMark along with the rest of your answers in PDF/image format, and separately submit your [runnable] source code to the A2 LEARN DropBox in a single ZIP file.