# CS472 / CM472 / CS672 - Fall 2006: Assignment 3

Instructor: Pascal Poupart    Office: DC2514    Email: `ppoupart@cs.uwaterloo.ca`
Classroom: RCH305    MWF 4:00-5:20    Office Hours: Mon, 10-11
Web Site: www.student.cs.uwaterloo.ca/ ~cs472/

Due: Tuesday, November 14, (at the beginning of class)

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 $k \geq 0$):

$$-\alpha\nabla \cdot \frac{1}{||\nabla u^k||_2}\nabla u^{k+1}(x,y) + u^{k+1}(x,y) \quad = \quad u^0(x,y) \qquad \text{in } \ \Omega = (0,1)\times(0,1), \tag{1}$$

where $\nabla u = (\partial u/\partial x, \partial u/\partial y)$ and $||\nabla u||_2 = \sqrt{(\partial u/\partial x)^2 + (\partial u/\partial y)^2}$. $\alpha > 0$ is the parameter controlling how much noise is to be removed. Figure 1 shows a reconstructed image after 20 iterations.

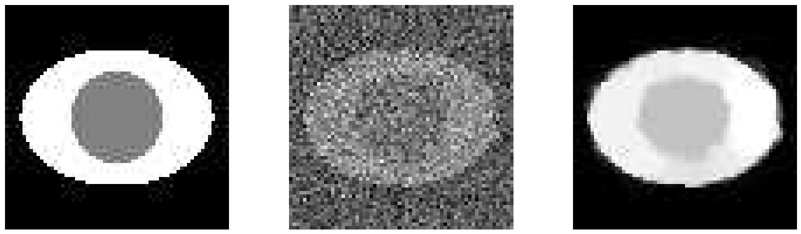
Figure 1: (Left) Original image, (Middle) Noisy image, (Right) Denoised image after 20 iterations.

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

$$ACu^{k+1}_{i,j} + AWu^{k+1}_{i-1,j} + AEu^{k+1}_{i+1,j} + ASu^{k+1}_{i,j-1} + ANu^{k+1}_{i,j+1} = u^0_{i,j}, \tag{2}$$

where

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

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

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:

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

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 condition.)

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 apply different iterative methods. Let $u^{k+1,m}$ be the approximate solution of $u^{k+1}$ given by $m$ 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 $m = 0, 1, \ldots,$ until convergence
        $u^{k+1,m+1} = \text{IterativeMethod}(u^{k+1,m}, A(u^k), u^0)$
    end
end
```

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

You are to implement Gauss-Seidel (GS), successive overrelaxation (SOR), conjugate gradient (CG), and preconditioned conjugate gradient (PCG) using ILU(0) preconditioner. The skeleton *main.c* code is set up to produce the matrix of image denoising and the right-hand side of noisy image as well as the outer iteration of the denoising algorithm. In *itsolver.c*, the code for Jacobi iteration is provided for you.

To speed up the convergence of CG, you are to implement ILU(0), which consists of the symbolic factorization (*ilusym*, given), the numeric incomplete factorization (*ilufac*), and the solve (*ilusol*) phases. For the factor phase, you just need to modify your factor routine from assignment 2. You should be able to reuse your solve routine from assignment 2. The prototypes of these functions can be found in *ilu.c*.

To test convergence of the iterative methods, use the stopping criterion:

$$\|r^k\|_2 \leq 10^{-2}\|b\|_2.$$

(Here, $b = u^0$.) In practice, one usually uses a much smaller tolerance, e.g. $10^{-6}$. However, for image denoising, the result is not distinguishable by eyes; more importantly, the number of iterations can be dramatically reduced. For Jacobi, Gauss-Seidel, and SOR, $r^k$ is not readily available. Since

Jacobi and Gauss-Seidel usually take thousands of iterations to convergence, you may want to do the convergence test every 100 iterations to save up time. For SOR, do the convergence test every 10 iterations. Also, impose the maximum number of iterations, say, 100000.

1. (15 marks) Submit a listing of your code. In order to black box test your code, place your version of *itsolver.c, itsolver.h, ilu.c, ilu.h* in a single directory (nothing else in this directory) with the name

   ```
   your_userid_your_student_id
   ```

   Then zip up this directory using

   ```
   zip -r your_userid_your_student_id your_userid_your_student_id
   ```

   Mail the file

   ```
   your_userid_your_student_id.zip
   ```

   to `cs472@student.cs.uwaterloo.ca`. This code will be linked with *main.o*, etc, by the TA. The total marks for this question are apportioned as: Gauss-Seidel (2 marks), SOR (3 marks), CG (5 marks), PCG (5 marks).

2. (15 marks) Construct a table of execution times and number of iterations for the following solutions methods: Jacobi, GS, SOR, CG, and PCG. For SOR, by trial and error, figure the optimal omega and report the value. For ILU(0), use natural, RCM, and minimum degree orderings. Use image size $16 \times 16$, $32 \times 32$, $64 \times 64$, $128 \times 128$. Comment on the results. Be sure to compile all your code with **make opt** (optimization option) to get the timing results. Use the MATLAB program *vizimage.m* to display the denoised image. Submit the denoised images for the grid sizes specified above.

   (bonus 3 marks) Report also the timings for sparse Gaussian elimination from assignment 2.

3. (10 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 $\rho(M) \leq \|M\|_\infty$ for any matrix $M$. Recall the definition of infinity norm:

   $$\begin{aligned} \|x\|_\infty &= max_i |x_i|, \\ \|M\|_\infty &= max_i \sum_{j=1}^{N} |a_{ij}|. \end{aligned}$$

   (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 $\|y\|_\infty < 1$.

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

4. (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

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

over all search directions.

(a) (5 marks) 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) (2 marks) 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) (3 marks) 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 $\alpha_j$ 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.

5. (10 marks) We want to prove the orthogonality property of the residual vectors; i.e. $r^k \perp \text{span}\{r^0, r^1, \ldots, r^{k-1}\}$. We prove it by induction.

(a) (5 marks) Suppose it holds that $r^i \perp \text{span}\{r^0, r^1, \ldots, r^{i-1}\}$, $i = 1, 2, \ldots, k$. Let $\{p^i\}$ be a set of A-orthogonal search direction vectors, where

$$p^k = r^k + \sum_{i=0}^{k-1} \beta_i p^i.$$

Show that $(r^k, p^k) = (r^k, r^k)$. (Hint: $\text{span}\{r^0, \ldots, r^{k-1}\} = \text{span}\{p^0, \ldots, p^{k-1}\}$)

(b) (5 marks) Consider $r^{k+1} = r^k - \alpha_k A p^k$. Show that

$$\begin{aligned} (r^{k+1}, p^k) &= 0, && \text{and} \\ (r^{k+1}, p^i) &= 0, && i = 0, 1, \ldots, k-1. \end{aligned}$$

Conclude your result.

6. **Graduate Student Question** (10 marks) Redo question 2 using the 3D Laplacian matrix of assignment 2.

**Practice Questions (Do not hand in)**

1. Consider a row diagonally dominant M matrix $A$. Show that $A^{(k)}$ (the submatrix after $k$ steps of an incomplete factorization) is also a diagonally dominant M matrix. Recall that an M matrix has positive diagonals, and non-positive off diagonals.

2. Consider a sparse matrix $A$ which is stored in the $ia-ja$ format by rows and by columns (two separate data structures). It is desired to form $A^TA$. Assume that the maximum number of nonzeros in any row or column of $A$ is $\alpha$. Describe an algorithm (precise pseudo code) to form $A^TA$ with computational complexity $O(\alpha^2 N)$.

3. Consider the matrix with symmetric structure, whose graph is given by

```
22 --- 23 --- 24 --- 19 --- 25
|      |      |      |      |
|      |      |      |      |
18 --- 17 --- 14 --- 13 --- 20
|      |      |      |      |
|      |      |      |      |
16 --- 15 --- 21 --- 12 --- 11
|      |      |      |      |
|      |      |      |      |
6  --- 7  --- 8  --- 9  --- 10
|      |      |      |      |
|      |      |      |      |
1  --- 2  --- 3  --- 4  --- 5
```

   Assume that Gaussian elimination is carried out on this matrix in the order indicated (diagonals used as pivots). Label all possible fill entries $a_{ij}^{(k+1)}$, $k = 1, ..., 20$ for row 21 with the level of fill (make a copy of the graph and label the nodes). Use the reachable set definition of level of fill. Show all your work. What is the meaning of an infinite level of fill?

4. On high performance architectures, it is desirable to to carry out the sparse forward and back solve of an iterative solver as efficiently as possible. Maximum parallelization can be obtained if it is noted that when solving for variable $i$, only some of the previous $(i-1)$ variables are needed. Hence, only these have to be solved at a previous stage. Unknowns can then be partitioned into sets:

   - Set 1: All equations that do not depend on any other one (normally, only one equation, i.e first equation in L).
   - Set 2: All equations that depend only on Set 1.
   - Set i: All equations that depend only on unknowns in Sets $\{1, 2, ..., i-1\}$.

   All unknowns in a given Set (or wavefront) can be solved concurrently, after all unknowns of previous wavefronts have been determined. Given a matrix which has a graph given by the test problem of assignment 2, determine the wavefront sets for L, for and ILU(0) incomplete factorization. Suggest an algorithm for determining the wavefront sets for an arbitrary L.

5. A new iterative algorithm for solution of nonsymmetric systems is the CGS method (unpreconditioned algorithm given below):

```
$r_0 = b - Ax_0$
$q_0 = p_{-1} = 0$
$\rho_{-1} = 1$ ;
    **For**    $n = 0, 1, 2, 3, ...$
        $\rho_n = (r_0, r_n)$ ; $\beta_n = \rho_n / \rho_{n-1}$
        $u_n = r_n + \beta_n q_n$
        $p_n = u_n + \beta_n(q_n + \beta_n p_{n-1})$
        $v_n = Ap_n$
        $\sigma_n = (r_0, v_n)$ ; $\alpha_n = \rho_n / \sigma_n$
        $q_{n+1} = u_n - \alpha_n v_n$
        $r_{n+1} = r_n - \alpha_n A(u_n + q_{n+1})$
        $x_{n+1} = x_n + \alpha_n(u_n + q_{n+1})$
        if converged, then quit
    **End**
```

Convert the above algorithm into a preconditioned CGS method, using the preconditioning matrix $C = (LU)^{-1}$. Use right preconditioning. Your algorithm should not require more than two forward and back solves, and two matrix vector multiplies per iteration.

6. Consider the example matrix given in *main.c* (assignment 3) which is a cube of size $nx \times ny \times nz$ nodes. This is a band matrix. Draw the picture of this band matrix. Label all the distances to the bands (from the diagonal) in terms of $nx$, $ny$, $nz$. Draw the picture of the level 1 ILU for this matrix, and label all the bands (including the level 1 fill) as above.

7. 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.

    $$Av_1 = \lambda_1 v_1.$$

    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 $\lambda_k$. 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 is the subspace $span\{r^0, Ar^0, \dots, A^m r^0\}$?)