Math 5601 Homework 9

Jacob Hauck

November 15, 2023

Problem 1.

Let A be a nonsingular matrix, and let $A^{(2)}$ be the matrix from the lecture slides in the second step of Gaussian elimination. Then there exists $s \geq 2$ such that $a_{2s}^{(2)} \neq 0$.

Proof. Suppose on the contrary. By the Gaussian elimination process, we know that $a_{21}^{(2)} = 0$. If there is no $s \geq 2$ such that $a_{2s}^{(2)} \neq 0$, then the whole second row of $A^{(2)}$ is zero. Hence, expanding by cofactors along the second row, we see that the determinant of $A^{(2)}$ is

$$\det\left(A^{(2)}\right) = 0 \cdot \det(B_1) + 0 \cdot \det(B_2) + \cdots + 0 \cdot \det(B_n) = 0, \tag{1}$$

where B_i is the cofactor corresponding to $a_{2i}^{(2)}$. Then $A^{(2)}$ is singular.

This is a contradiction because $A^{(2)}$ was obtained from A by elementary row operations, and A was nonsingular, and applying row operations to a nonsingular matrix must result in a nonsingular matrix. $\square$

Problem 2.

Let $A = \{a_{ij}\}$, and consider the SOR iteration for solving $Ax = b$:

$$x_i^{(k+1)} = (1-\sigma)x_i^{(k)} + \frac{\sigma}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right], \qquad i = 1, 2, \ldots, n. \tag{2}$$

If L is the lower-triangular part of A and U is the upper-triangular part, and D is the diagonal, so that $A = L + U + D$, then the SOR iteration becomes

$$\begin{aligned}
x^{(k+1)} &= (1-\sigma)x^{(k)} + \sigma D^{-1}\left[b - Lx^{(k+1)} - Ux^{(k)}\right] && (3)\\
\implies (D + \sigma L)x^{(k+1)} &= ((1-\sigma)D - \sigma U)x^{(k)} + \sigma b && (4)\\
\implies x^{(k+1)} &= \left(L + \frac{1}{\sigma}D\right)^{-1}\left(\frac{1}{\sigma}D - D - U\right)x^{(k)} + \left(L + \frac{1}{\sigma}D\right)^{-1} b && (5)\\
&= -\left(L + \frac{1}{\sigma}D\right)^{-1}\left(D - \frac{1}{\sigma}D + U\right)x^{(k)} + \left(L + \frac{1}{\sigma}D\right)^{-1} b. && (6)
\end{aligned}$$

Define $M = L + \frac{1}{\sigma}D$, and $N = -\left(D - \frac{1}{\sigma}D + U\right)$. Then

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b, \tag{7}$$

and

$$A = L + D + U = \left(L + \frac{1}{\sigma}D\right) + \left(D - \frac{1}{\sigma}D + U\right) = M - N. \tag{8}$$

Therefore, SOR is an iterative method that uses the M and N defined above.

Problem 3.

To program the Jacobi and Gauss-Seidel methods, I split the iterative solver into two parts. In the first part, the matrix B and vector c in the iterative form $x^{(k+1)} = Bx^{(k)} + c$ are computed. In the second part, the iteration is performed. The iteration can be done the same way regardless of how B and c are computed, so it is implemented once in the `solve_iterative.m` file. The Jacobi and Gauss-Seidel methods calculate B and c from A and b differently. These calculations are in the `jacobi.m` and `gauss_seidel.m` files. Here is a copy of the code for convenience.

```
function result = solve_iterative(B, c, x0, tol, dist, max_iter)

x = x0;
for i = 1:max_iter
    x_next = B * x + c;
    cauchy_error = dist(x, x_next);

    fprintf(['Iteration %d: x = (%.03e, %.03e, %.03e),', ...
        ' x_next = (%.03e, %.03e, %.03e), Cauchy error = %.05e\n'], ...
        i, x(1), x(2), x(3), x_next(1), x_next(2), x_next(3), ...
        cauchy_error ...
    );

    if cauchy_error < tol
        break;
    end

    x = x_next;
end

result = x;
```

```
function [B, c] = jacobi(A, b)

d = diag(A);
N = A - diag(d);

c = N * b;
B = diag(1 ./ d) * N;
```

```
function [B, c] = gauss_seidel(A, b)

L_plus_D = tril(A);
U = A - L_plus_D;

B = -L_plus_D \ U;
c = L_plus_D \ b;
```

The solution of the given system is $x = 0$. The Jacobi method converges to $x = 0$, but the Gauss-Seidel method does not; instead, it alternates between x_0 and $-x_0$. See `output.txt` for the MATLAB output from the Jacobi and Gauss-Seidel iterations.