CS319: Scientific Computing (with MATLAB)

Lab 6: Solving Linear Systems, Part 2

Week 10, 2023

Niall Madden (Niall.Madden@UniversityOfGalway.ie)

- **What to do:** Develop functions that implement Jacobi and Gauss-Seidel methods, and a MATLAB live script compare their efficiency.
- **What has this got to do with Scientific Computing?** The solution of linear systems is absolutely central to scientific computing, as is the evaluation and comparison of algorithms.
- **What to upload:** A single live script, that includes the functions at the end.
- **Where to start:** Start by completing Lab 5.

1: Jacobi's method

In Lab 5 you developed implementations of the Jacobi algorithms for solving a linear system of N equations in N unknowns: *find* $x_1, x_2, \ldots, x_N$, *such that*

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N &= b_2 \\ &\vdots \\ a_{N1}x_1 + a_{N2}x_2 + \cdots + a_{NN}x_N &= b_N. \end{aligned}$$

We expressed this as a matrix-vector equation: *Find* x *such that*

$$A\mathrm{x} = \mathrm{b},$$

where A *is a* $N \times N$ *matrix, and* b *and* x *are (column) vectors with* N *entries.*

Jacobi's method is:

- Choose any vector $\mathbf{x}^{(1)}$.
- For $k = 2, 3, \ldots,$ set

$$x_1^{(k)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \cdots - a_{1N}x_N^{(k-1)})$$

$$x_2^{(k)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k-1)} - a_{23}x_3^{(k-1)} - \cdots - a_{2N}x_N^{(k-1)})$$

$$\vdots$$

$$x_N^{(k)} = \frac{1}{a_{NN}}(b_N - a_{N,1}x_1^{(k-1)} - a_{N,2}x_1^{(k-1)} - \cdots - a_{N,N-1}x_{N-1}^{(k-1)})$$

There is also a matrix-version of this iteration. We set D and T to be the matrices

$$d_{ij} = \begin{cases} a_{ii} & i = j \\ 0 & \text{otherwise.} \end{cases} \qquad t_{ij} = \begin{cases} 0 & i = j \\ -a_{ij} & \text{otherwise.} \end{cases}$$

So $A = D - T$. Then *Jacobi's method* can be written neatly in matrix form:

$$x^{(k)} = D^{-1}(b + Tx^{(k-1)}). \tag{1}$$

Jacobi's method is not particularly efficient. Heuristically, you argue that it could be improved as follows. In Jacobi's method, we compute $x_1^{(k)}$ from

$$x_1^{(k)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \cdots - a_{1N}x_N^{(k-1)})$$

We expect that it is a better estimate for x_1 than $x_1^{(k-1)}$.
Next we compute

$$x_2^{(k)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k-1)} - a_{23}x_3^{(k-1)} - \cdots - a_{2N}x_N^{(k-1)})$$

However, here we used the "old" value of $x_1^{(k-1)}$, even though we already know the new, improved, $x_1^{(k+1)}$. That is, we could use

$$x_2^{(k)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k-1)} - \cdots - a_{2N}x_N^{(k-1)})$$

This is called the **Gauss-Seidel** method.

More generally, in Jacobi's method we set

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{N} a_{ij} x_j^{(k-1)}\right).$$

The Gauss-Seidel method uses

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{N} a_{ij} x_j^{(k-1)}\right). \tag{2}$$

In this lab, you'll implement this method, and verify that it is more efficient than the Jacobi method, in the sense that fewer iterations are required to achieve the same level of accuracy.

Writing out (2) in detail, it is

$$x_1^{(k)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \cdots - a_{1N}x_N^{(k-1)})$$
$$x_2^{(k)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k-1)} - \cdots - a_{2N}x_N^{(k-1)})$$
$$\vdots$$
$$x_N^{(k)} = \frac{1}{a_{NN}}(b_N - a_{N1}x_1^{(k)} - \cdots - a_{N,N-1}x_{N-1}^{(k)})$$

In the same way as we did for Jacobi's method, we can write this in a succinct matrix-vector form: we set L and U to be the matrices

$$l_{ij} = \begin{cases} a_{ij} & i \geq j \\ 0 & \text{otherwise.} \end{cases} \qquad u_{ij} = \begin{cases} 0 & i \geq j \\ -a_{ij} & \text{otherwise.} \end{cases}$$

So $A = L - U$. Then the *Gauss-Seidel method* can be written as

$$Lx^{(k+1)} = b + Ux^{(k)}. \tag{3}$$

Note that this involves solving a linear system where L is the coefficient matrix. However, this only involves back-substitution, so is fast.

3: Assignment

You will write a MATLAB live script that tests and compares the Jacobi and Gauss-Seidel methods. The functions for Jacobi and Gauss-Seidel should be at the end of the live script (MATLAB requires that any functions defined in a script must be contains in a section at the end of the script, and that that section can only contain functions).

1. Create a new Live Script. Give it a title, followed by your name, ID number, and email address. Follow that with a table of contents.
2. Add a suitably titled section that sets the integer value N, and then creates a (somewhat random) diagonally dominant $N \times N$ matrix, A. It should also create a non-zero vector x, and a vector $b = Ax$. (See Q1 from Lab 5 on details of diagonally dominant matrices).
3. Write a function implementing the Jacobi method.
   - 3.1 At the **end** of the script, add a suitably titled section. In that section add code for the function that implements the Jacobi method (you can base this on the code provided in Lab 5, or use the matrix version, as you prefer).
   - 3.2 The function should take 4 arguments: a square matrix A, a vector b, a (scalar) `TOL`, and a (scalar) `MaxIts`.

3.3 Jacobi's method to be applied to solving $Ax = b$. At iteration k compute the residual $r(k) = \|b - Ax^{(k)}\|$. A `while` loop should be used to continue iterating until either $r(k) < TOL$ or until `MaxIts` iterations have been computed.

3.4 The function should return both $x^{(k)}$ and r.

4. Write a function that implements the Gauss-Seidel method.
   - 4.1 At with the Jacobi code, it should be in a section at the **end** of the script.
   - 4.2 The function should take 4 arguments: a square matrix A, a vector b, a (scalar) `TOL`, and a (scalar) `MaxIts`.
   - 4.3 The GS method should be applied to solving $Ax = b$. At iteration k compute the residual $r(k) = \|b - Ax^{(k)}\|$. A `while` loop should be used to continue iterating until either $r(k) < TOL$ or until `MaxIts` iterations have been computed.
   - 4.4 The function should return both $x^{(k)}$ and r.
5. Add sections (before the function section) to test each of the Jacobi and GS methods applied to your test problem. For each method, the following should be reported:
   - Number of iterations taken;
   - Residual at the final iteration;
   - The norm of the actual error.

3: Assignment

6. In addition, you should plot both sets of residuals on the same graph? What function is most appropriate for this: `plot()`, `loglog()`, `semilogx()`, `semilogy()`, something else?

Submit the MATLAB live script through the "`Lab 6`" section of the Blackboard module. Usual collaboration rules apply (ask Niall if you need clarification). *Don't forget to include your name, ID number, and University email address in the files!*

Deadline: **5pm, Friday 24 March**