Computational Linear Algebra: Ax=b (Part IV: Iterative Methods)

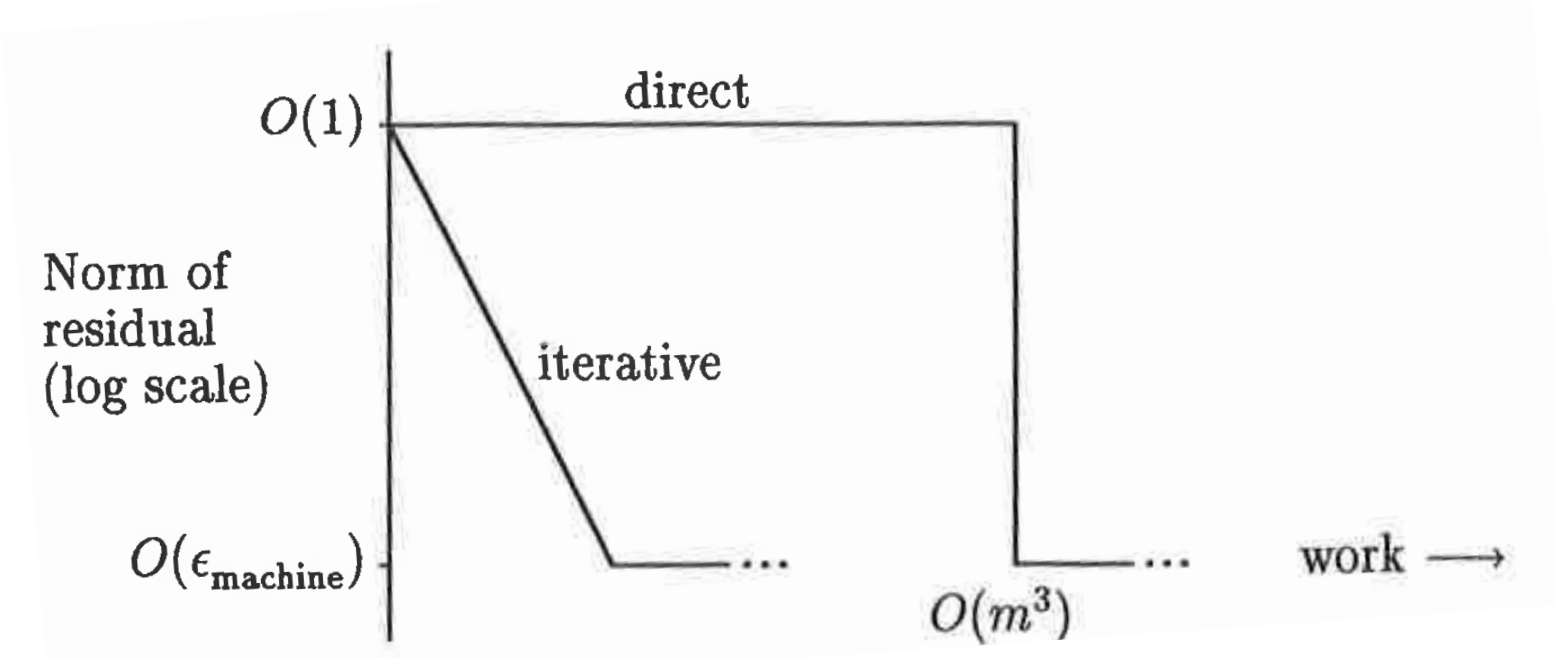
David Shuman

February 22, 2022

Iterative Methods for Solving $Ax = b$ (Section 2.5)

- Gaussian Elimination and its relatives ($A = LU$, $PA = LU$, Cholesky) are direct methods of solving a system of equations
  - In theory these methods go directly to the solution in a finite number of steps: $O(n^3)$
- An iterative method starts with a guess at the answer and we improve our guess at each step
  - We saw iterative methods for root finding (e.g., Newton's Method)
  - We expect the iterative process to converge to the correct answer
  - It may never get there, but it may get to within a certain tolerance in a reasonable number of steps

Convergence: Iterative vs. Direct Methods



- Figure from Trefethen and Bau, 1997, p. 247
- This is for iterative methods under "favorable" conditions

The Jacobi Method

This is a cool trick for solving $A\mathbf{x} = \mathbf{b}$

- Write $A = D + R$, where D is the diagonal and $R = A - D$.

$$E.g., \quad \begin{pmatrix} 10 & 2 & 3 & -2 \\ 1 & 6 & 2 & -1 \\ 2 & 1 & -7 & 3 \\ 3 & -1 & 3 & 9 \end{pmatrix} = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & -7 & 0 \\ 0 & 0 & 0 & 9 \end{pmatrix} + \begin{pmatrix} 0 & 2 & 3 & -2 \\ 1 & 0 & 2 & -1 \\ 2 & 1 & 0 & 3 \\ 3 & -1 & 3 & 0 \end{pmatrix}$$

- Then:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ (D + R)\mathbf{x} &= \mathbf{b} \\ D\mathbf{x} &= \mathbf{b} - R\mathbf{x} \\ \mathbf{x} &= D^{-1}(\mathbf{b} - R\mathbf{x}) \end{aligned}$$

- Key features: D is diagonal so it is easily invertible
- Sometimes $\mathbf{x} \leftarrow D^{-1}(\mathbf{b} - R\mathbf{x})$ converges to the solution

Example

- Let $A = \begin{pmatrix} 5 & 2 \\ 1 & -4 \end{pmatrix}$ and $\mathrm{b} = (16, 10)$.

  Use R to find the exact solution to $A\mathrm{x} = \mathrm{b}$

- Now, starting with $\mathrm{x}_0 = (0, 0)$ together we will iterate

$$\mathrm{x} \leftarrow D^{-1}(\mathrm{b} - R\mathrm{x})$$

  to see if it converges to the correct solution

- Repeat on $A = \begin{pmatrix} 1 & 4 \\ 5 & -2 \end{pmatrix}$ and $\mathrm{b} = (11, 11)$

The Jacobi Algorithm

Look at the R code `jacobi` in `365Functions.r`
Discuss

- $D = \text{diag}(A)$
- $A = D + R$
- the iteration $\mathrm{x}^{(i+1)} = D^{-1}(\mathrm{b} - R\mathrm{x}^{(i)})$ is computed by the statement

  ```
  x = (b - R %*% x)/d
  ```

- The matrix D is never created
- Count operations for the Jacobi method
  - Each iteration is $O(n^2)$ [if R is a dense matrix; much better if R is sparse]

Stopping

When should the iterations stop?

- Our iteration is of the form

$$\mathrm{x}^{(k+1)} \leftarrow D^{-1}(\mathrm{b} - R\mathrm{x}^{(k)})$$

- The iteration is not guaranteed to converge, so you should use a maximum number of iterations
- When it is converging, the stopping condition that is most convenient and widely used in existing code is when the relative residual norm is less than a given tolerance:

$$\text{relative residual norm} : \quad \frac{||\mathrm{b} - A\mathrm{x}^{(k)}||}{||\mathrm{b}||} \leq tol$$

Convergence of Jacobi's Method

- Two questions:
  - When does it converge to the solution?
  - How fast does it converge?
- Note that

$$\mathrm{x} = D^{-1}(\mathrm{b} - R\mathrm{x}) = \underbrace{-D^{-1}R}_{B}\mathrm{x} + \underbrace{D^{-1}\mathrm{b}}_{z} = B\mathrm{x} + \mathrm{z}$$

So we have

$$\mathrm{x}^{(k+1)} = B\mathrm{x}^{(k)} + \mathrm{z} = B\left(\mathrm{x} + \mathrm{e}^{(k)}\right) + \mathrm{z} = B\mathrm{x} + \mathrm{z} + B\mathrm{e}^{(k)} = \mathrm{x} + B\mathrm{e}^{(k)}$$

$$\Rightarrow \mathrm{e}^{(k+1)} = B\mathrm{e}^{(k)} = B^{k+1}\mathrm{e}^{(0)}$$

Now we can write the initial error $\mathrm{e}^{(0)}$ as a linear combination of the eigenvectors of B to get

$$\mathrm{e}^{(k+1)} = B^{k+1}\sum_{i=1}^{n}\alpha_i \mathrm{v}_i = \sum_{i=1}^{n}\alpha_i B^{k+1}\mathrm{v}_i = \sum_{i=1}^{n}\alpha_i \lambda_i^{k+1}\mathrm{v}_i$$

- Necessary and sufficient condition for Jacobi method to converge: eigenvalues of B are all less than 1 in magnitude (i.e., $|\lambda_i| < 1, \forall i$)

Convergence of Jacobi's Method (cont.)

- There is one easy-to-check sufficient (but not necessary) condition: the Jacobi method converges if the matrix A is strictly diagonally dominant, i.e.,

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \qquad \text{in each row } i$$

- Convergence is faster when the eigenvalues of B are all small in magnitude
- If the sufficient condition on diagonal dominance is satisfied, the greater the diagonal dominance, the faster the method converges, in general

Other Iterative Methods

- For these methods, $A = L + D + U$, where L is lower triangular, D is diagonal, and U is upper triangular
  - Note: different than our LU decomposition, which was multiplicative. This notation assumes the three matrices are added
  - With this notation, Jacobi iteration is:

$$x^{(k+1)} = D^{-1}\left(b - Ux^{(k)} - Lx^{(k)}\right)$$

- The Gauss-Seidel Method (also known as the Liebmann Method)
  - Same as Jacobi except that updates are done one row at a time
  - i.e., the computation of $x_i^{(k+1)}$ uses the components of $x^{(k+1)}$ that have already been updated, instead of the old components from $x^{(k)}$:

$$x^{(k+1)} = D^{-1}\left(b - Ux^{(k)} - Lx^{(k+1)}\right)$$

  - Requires less memory than Jacobi, and often converges faster
  - Look at the R code `GaussSeidel` in `365Functions.r`

Other Iterative Methods (cont.)

- Successive Over Relaxation (SOR)
  - A generalization of Gauss-Seidel

$$x^{(k+1)} = (\omega L + D)^{-1} \left(\omega b - [\omega U + (\omega - 1) D] \, x^{(k)} \right)$$

  - When $\omega = 1$, reduces to Gauss-Seidel
- If interested, see Section 2.5 of the textbook for more details on these algorithms

Why Use Iterative Methods?

Some major reasons/situations for using iterative methods:

1. A single step of an iterative method may require only n^2 steps, so it can be faster if it converges to within an acceptable tolerance in fewer than n iterations
2. When you already have a good approximation/initial guess about the solution x
   - For example, if you solve $A\mathrm{x} = b$ to get x_0, and then A or b is slightly tweaked
   - The solution x_0 is a good starting point to begin an iterative method on the tweaked problem
   - This technique is called polishing
3. When the matrix A is sparse
   - Many (usually all but $O(n)$) of the entries of A are equal to 0
   - Matrix-vector multiplication is much more efficient for sparse matrices, so you want to i) preserve the sparse structure of the matrix, and ii) leverage it to make the computations more efficient
   - Gaussian elimination typically fills in a sparse matrix when it does row operations, losing the sparse structure

Sparse Matrices in R

Example 2.25: on laptop

- $n = 1,000$ (8,000,200 bytes dense / 65,352 bytes sparse)
  - Dense matrix, using solve: 0.441 seconds
  - Sparse matrix, using solve: 0.003 seconds
  - Sparse matrix, using Jacobi: 0.014 seconds; (to 4 digits in ∞-norm)
- $n = 10,000$ (800,000,200 bytes / 641,352 bytes sparse)
  - Dense matrix: 339.2 seconds
  - Sparse matrix, using solve: 0.011 seconds
  - Sparse matrix, using Jacobi: 0.042 seconds; (to 4 digits in ∞-norm)
- $n = 100,000$ (6,401,352 bytes sparse)
  - Dense matrix: can't do (not enough memory)
  - Sparse matrix, using solve: 0.109 seconds
  - Sparse matrix, using Jacobi: 0.546 seconds; (to 4 digits in ∞-norm)
- $n = 1,000,000$ (64,001,352 bytes sparse)
  - Dense matrix: can't do (not enough memory)
  - Sparse matrix, using solve: can't do (not enough memory)
  - Sparse matrix, using Jacobi: 5.720 seconds; (to 4 digits in ∞-norm)