Lecture 08: Iterative Methods

June 18, 2025

Outline

Iterative Methods

- The previous lecture concluded our look at direct methods for linear systems.
- These methods are based on factoring the matrix A.
- They solve the system in a known finite sequence of steps, then return the solution.
- In this lecture we begin looking at **iterative** methods for linear systems.
- These methods gradually and iteratively refine a solution.
- They repeat the same steps over and over, then stop only when a desired tolerance is achieved.

Iterative Methods

Possible benefits of iterative methods compared to direct methods:

1. They may be faster and tend require less memory.
2. They may be faster for typically large, sparse, higher-dimensional problems, since they are usually less memory-intensive since no fill-in occurs. LU factorization was $O(n^3)$ in the worst (fully dense) case for $A \in \mathbb{R}^{n \times n}$. For iterative methods the operation count depends on number of non-zeros (nnz), as well as, how many iterations it takes.
3. Another benefit for applications needing only approximate solutions is that one can **"quit early"**. With iterative methods you can increase your error tolerance to obtain a less accurate approximate solution. Ideally, iterative approaches make gradual progress in the solution quality up to the tolerance (or limits of floating point arithmetic). Direct approaches give no solution at all until they complete all operations. Depending on the problem one or the other may get to a (satisfactory) solution first as shown in Figure 1.
4. Exact algorithms such as Gaussian elimination need to alter the matrix A. The splitting algorithms discussed below do not. If we cannot find a good ordering to (significantly) reduce the amount of fill-in, then iterative algorithms should be used (for large problems). The downside of course is that we give up computing an exact (up to machine precision) solution.

Iterative Methods - Termination Criterion

The termination criterion is based on the error $e = x - \hat{x}$ between

1. the (current, approximate) numerical solution $\hat{x}$ and
2. the true solution x.

We terminate computation when $e \approx 0$. However, we do not know the true solution x since that is what we are trying to compute. So a more practical indicator of the error is the norm of the **residual**

$$r = b - A\hat{x}.$$

The residual measures how much the current approximation fails to satisfy $A\hat{x} = b$.

Iterative Methods - Termination Criterion

The residual and error satisfy $Ae = r$ since

$$\begin{aligned} Ax &= b, \\ \Rightarrow Ax - A\hat{x} &= b - A\hat{x}, \\ \Rightarrow A\underbrace{(x - \hat{x})}_{e} &= \underbrace{b - A\hat{x}}_{r}, \\ \Rightarrow Ae &= r. \end{aligned}$$

Iterative Methods - Termination Criterion

Therefore, the matrix condition number (See Stability in Lecture 07) can be used to bound the (relative) size of error e and residual r as

$$\frac{\|e\|}{\|x\|} \leq \kappa(A)\frac{\|r\|}{\|b\|}.$$

From this we see that a small residual can imply a small error, but only if A is **well conditioned** (i.e., κ is small).

Iterative Methods - Termination Criterion

As an example, consider the 1×1 linear system $5x = 300$.

- The true solution is obviously $x = 60$.
- If our current best estimate is $\hat{x} \approx 50$, then the error is $e = x - \hat{x} = 60 - 50 = 10$.
- This is the exact error of how "wrong" $\hat{x}$ is.
- The residual in this case is $r = b - A\hat{x} = 300 - 5(50) = 50$ (i.e., how far is $b - A\hat{x}$ from zero?).
- Notice also that the claim of $Ae = r$ is satisfied since $5(10) = 50$.

Stationary Iterative Methods

Iterative methods start from some (perhaps arbitrary or zero) guess at the solution. Increasingly accurate approximations to the solution are generated by iterating a basic procedure repeatedly.

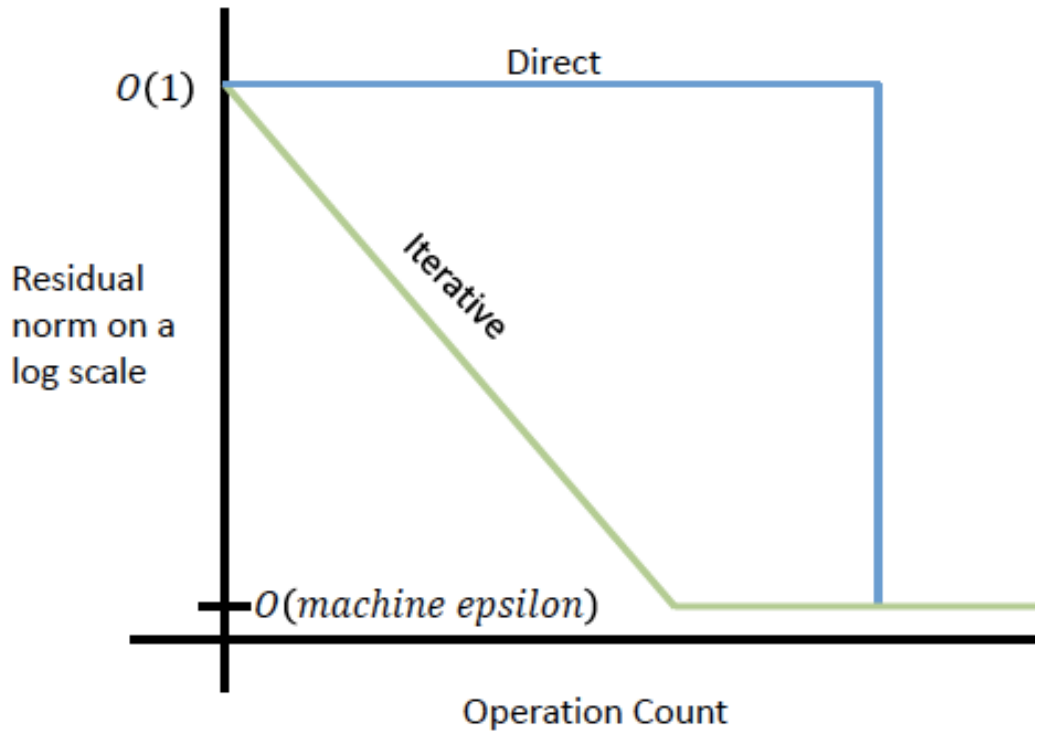

Figure: Comparison of the path to solutions of iterative versus direct methods.

Stationary Iterative Methods

Q & A

1. Do we always know that we will converge towards a solution?
   A: No. We will need to be careful about this point, because convergence is not guaranteed without additional assumptions. We will discuss these convergence assumptions, soon.
2. If an iterative method converges, must it produce a "close enough" answer more quickly than the direct methods that we have studied?
   A: No: Not all iterative methods are created equal, and the meaning of "close enough" completely depends on the choice of tolerance.

Splitting Methods

This section gives describes splitting methods for iteratively solving linear equations. We can rewrite the linear system $Ax = b$ as

$$(M - N)x = b \Leftrightarrow Mx = Nx + b, \qquad \text{where } A = M - N.$$

Important Notes About Notation:

1. We assume that M must be **invertible**.
2. (In each method described below, you should think about what extra constraints this would place on the coefficient matrix, A.)
3. However, we do **not** necessarily compute M^{-1} to solve a system involving M.
4. We abuse notation slightly in what follows, by writing $M^{-1}b$ as shorthand for the solution, x, to the system $Mx = b$.

Iterative Methods

- Then, starting with some initial guess x^0, we can **iteratively** find x by repeatedly solving

$$Mx^{k+1} = (Nx^k + b). \tag{1}$$

- Then we have:

$$\begin{aligned} Mx^{k+1} &= Nx^k + b \\ &= (M - A)x^k + b \\ &= Mx^k - Ax^k + b \\ &= Mx^k + (b - Ax^k) \\ x^{k+1} &= x^k + M^{-1} \underbrace{(b - Ax^k)}_{r^k, \text{ at step } k \ (\hat{x}=x^k)} \end{aligned} \tag{2}$$

Iterative Methods

- For the splitting method (1) to be effective, we need to choose M (hence N) so that
  1. it is easy to solve the linear system (1), i.e., $My = z$ should be easy to solve.
  2. M is close to A, in the sense of having small norm $\|I - M^{-1}A\|$.
- At one extreme we could choose $M = A$, and the iterative procedure (1) will "converge" in one iteration since

$$x^{k+1} = x^k + A^{-1}(b - Ax^k) = x^k + x - x^k = x.$$

- However, using the actual A^{-1} is too expensive (and defeats the purpose).
- The cost would be solving a general linear system $Ax = b$.
- There is a trade-off between the two goals, so we want to take $M \approx A$.

Splitting Methods - Richardson

- Richardson iteration is perhaps the simplest method.
- The choice of M the scaled identity matrix

$$M = \frac{1}{\theta} I,$$

  where $\theta > 0$ is some appropriately chosen constant.
- How to choose θ will be detailed when we discuss the convergence of iterative methods in Section 3.
- We have from (2) that the Richardson iteration is

$$x^{k+1} = x^k + \theta(b - Ax^k).$$

- Or, for a particular i^{th} entry we have $x_i^{k+1} = x_i^k + \theta\left(b_i - \sum_{j=1}^{n} a_{ij} x_j^k\right)$.
- Note that the new value x^{k+1} is a weighted sum of old value x^k and the residual $b - Ax^k$.
- Clearly, each iteration costs $O(\text{nnz}(A))$.
- Note that we need to store 2 separate vectors, x^k and x^{k+1}.

Splitting Methods - Jacobi

The next three methods will rely on the following labelled submatrices of A

- D = main diagonal (all diagonal entries non-zero),
- $-L$ = (strictly) below diagonal,
- $-U$ = (strictly) above diagonal.

$$A = \begin{bmatrix} \ddots & & -U \\ & D & \\ -L & & \ddots \end{bmatrix}$$

- With the choice

$$M = D := \text{diag}(A),$$

we have the Jacobi iteration from (2):

$$x^{k+1} = x^k + D^{-1}(b - Ax^k).$$

Splitting Methods - Jacobi

- Intuitively, we "exactly" solve each row equation independently for the corresponding entry, using the current estimate of vector x^k. For example, consider if row 7 looks like

$$2x_5 - 5x_6 + 10x_7 + 3x_9 = 14.$$

- Then to find x^{k+1} for row index $i = 7$ we set

$$x_7^{k+1} = \frac{1}{10}\left(14 - 2x_5^k + 5x_6^k - 3x_9^k\right),$$

  using the other known (step k) values of x.
- Again, each iteration costs $O(\text{nnz}(A))$ and we must store 2 vectors.

Splitting Methods - Jacobi

- Let us write the Jacobi iteration more explicitly for the ith entry as

$$x_i^{k+1} = x_i^k + \frac{1}{a_{ii}} \left(b_i - \sum_j a_{ij} x_j^k \right),$$

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

Splitting Methods - Jacobi

Motivation for the Jacobi iteration:

- Recall the residual vector $r = b - Ax$.
- Clearly, x is a solution iff $r = 0$.
- Given the current iterate x^k, the Jacobi iteration tries to zero out the i-th residual r_i, **in turn**:

$$\begin{aligned} r_i &= 0 \\ \iff b_i - \sum_{j \neq i} a_{ij} x_j^k - a_{ii} x_i^{k+1} &= 0 \\ \Rightarrow x_i^{k+1} &= \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^k \right). \end{aligned}$$

Splitting Methods - Jacobi

- The Jacobi iteration is easy to implement but unfortunately quite slow.
- However, it is trivially **parallelizable**.
- Given x^k, we can update the components in the next iteration x^{k+1} in parallel.

Splitting Methods - Gauss-Seidel

- The Gauss-Siedel iteration is very similar to the Jacobi iteration.
- The difference is instead of using "old" data, x^k, use "new" x^{k+1} data for entries that have already been updated so far on this pass.
- That is,

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

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

Splitting Methods - Gauss-Seidel

- Recall we are zeroing out the residual at ith step - this gives the first equation.
- If we rearrange the updates we have

$$\sum_{j \leq i} a_{ij} x_j^{k+1} = b - \sum_{j > i} a_{ij} x_j^k,$$

or in matrix form

$$\begin{aligned} (D - L)x^{k+1} &= b + Ux^k \\ \Rightarrow \quad x^{k+1} &= x^k + (D - L)^{-1}(b - Ax^k), \end{aligned}$$

so

$$M = D - L.$$

- Again, each iteration costs $O(\text{nnz}(A))$.
- Gauss-Seidel however only needs to store one vector since you can update/overwrite x^k entries as you go!

Splitting Methods - Backward Gauss-Seidel

- There exist variants of Gauss-Siedel (GS).
- There is nothing special about proceeding to update x^k from top to bottom (i.e ., GS runs over rows from $i = 1, \ldots, n$).
- The reverse ordering gives **backward** Gauss-Seidel.
- This swaps the role of L and U giving

$$x^{k+1} = x^k + (D - U)^{-1}(b - Ax^k),$$

  which corresponds to update the elements of x in the reverse order (i.e., $i = n, n-1, \ldots, 1$).
- Thus

$$M = D - U.$$

Splitting Methods - Symmetric Gauss-Seidel

- Of course, we can also combine (forward) Gauss-Siedel with backward Gauss-Siedel to construct **symmetric** Gauss-Seidel

$$\begin{aligned} x^{k+1/2} &= x^k + (D-L)^{-1}(b - Ax^k), \\ x^{k+1} &= x^{k+1/2} + (D-U)^{-1}(b - Ax^{k+1/2}) \\ &\underbrace{=}_{\text{See Lecture Notes}} x^k + (D-U)^{-1}D(D-L)^{-1}(b - Ax^k). \end{aligned}$$

- So that for symmetric Gauss-Seidel, the matrix M is

$$M = (D-L)D^{-1}(D-U).$$

Splitting Methods - Red-black Gauss-Seidel (Optional)

- Gauss-Siedel usually converges faster than the Jacobi iteration.
- However, GS is an inherently **sequential** algorithm and is harder to parallelize.
- **Red-black Gauss-Seidel** is an update ordering that allows for some parallelization (see Figure 2 left).
- It alternates between sweeps of updating (1) only red nodes and (2) only black nodes.
- We can update all red nodes in parallel since they only use (old) black data, and vice versa.
- One can generalize the red-black GS idea to non-grid structured problems by coloring a graph.
- You ensure no neighbours have same color and then update all same color neighbours simultaneously (see Figure 2 right).
- See for example the application of real time cloth simulation by [Fratarcangeli et al. 2016] in this video.

Splitting Methods - Gauss-Seidel

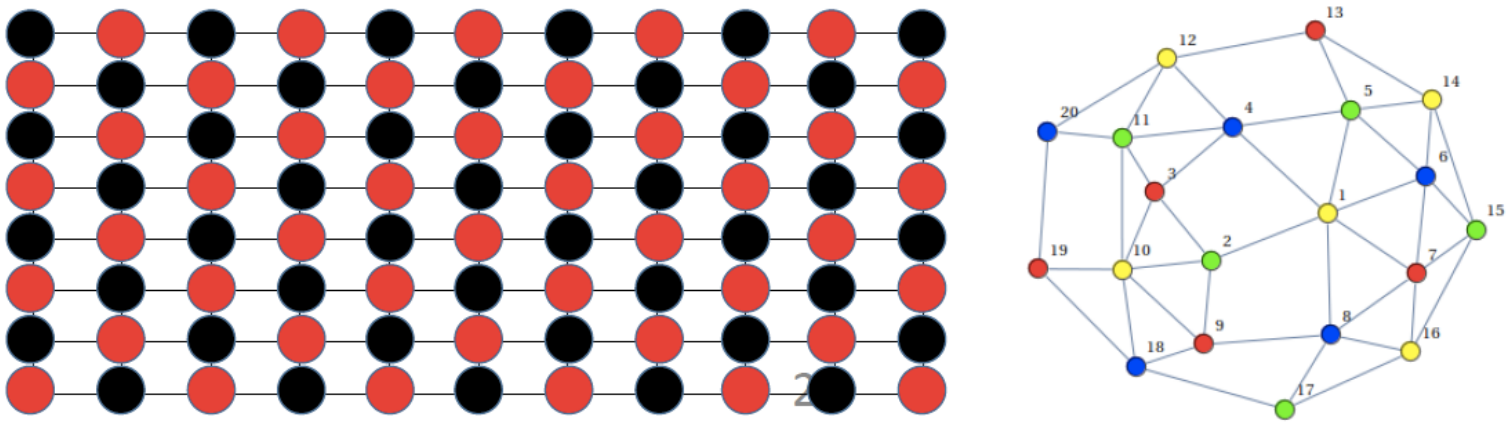

Figure: Example colorings for parallelizing the Gauss-Seidel iteration.

Splitting Methods - Successive Over Relaxation (SOR)

- A common strategy to accelerate fixed-point iterations is averaging.
- For instance, by averaging the current iterate x^k with the GS update according to a **relaxation factor** $\omega > 0$, we obtain SOR:

$$\begin{aligned} x_i^{k+1} &= (1-\omega)x_i^k + \omega(x_i^{k+1})^{\mathtt{GS}} \\ &= (1-\omega)x_i^k + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k\right), \end{aligned}$$

  or in matrix notation

$$x^{k+1} = x^k + \left(\frac{1}{\omega}D - L\right)^{-1}\left(b - Ax^k\right).$$

- Hence, for SOR the matrix M is

$$M = \frac{1}{\omega}D - L.$$

Splitting Methods - Successive Over Relaxation (SOR)

- See the Lecture Notes for a detailed explanation of the setup above.
- For $0 < \omega < 1$ the update is called **under-relaxation**, while
- for $\omega > 1$ the update is called **over-relaxation**.
- When $\omega = 1$, SOR equals Gauss-Seidel.
- For certain choices of $\omega (> 1)$, SOR may converge substantially faster than GS.

Convergence of Splitting Methods

- The key questions involving the convergence of splitting methods are:
  1. under what conditions does the iteration converge to the correct solution?
  2. if it does converge, how quickly does it do so (in terms of number of iterations)?
- These convergence questions depend on the **spectral radius** of A, denoted $\rho(A)$.
- The spectral radius is defined in terms of the eigenvalues of A.

Convergence of Splitting Methods

Definition 3.1

An **eigenvalues** $\lambda \in \mathbb{R}$ *and corresponding eigenvector* $v \in \mathbb{R}^n$ *of* $A \in \mathbb{R}^{n \times n}$ *satisfy*

$$Av = \lambda v \qquad \text{and } v \neq 0.$$

Definition 3.2

The **spectral radius** *of* A *is*

$$\rho(A) = \max_i |\lambda_i|,$$

where λ_i *are the eigenvalues of* A*. In other words,* $\rho(A)$ *is the largest magnitude of an eigenvalue of* A*.*

Convergence of Splitting Methods

- We can rewrite our usual iteration as

$$x^{k+1} = x^k + M^{-1}(b - Ax^k) = (I - M^{-1}A)x^k + M^{-1}b.$$

- We call the matrix $I - M^{-1}A$ the **iteration matrix** for a particular method.
- The following theorem gives a sufficient condition for convergence.

Theorem 1

Let the true solution x^* satisfy $x^* = (I - M^{-1}A)x^* + M^{-1}b$. If $\|I - M^{-1}A\| < 1$ for some induced matrix norm, then the stationary iterative method converges.

- *That is, for any initial guess x^0,*

$$\lim_{k\to\infty} x^k = x^*.$$

Proof.

See Lecture Notes. □

Convergence of Splitting Methods

- Note that an induced matrix norm is defined as $\|A\| := \max_{\|x\|=1} \|Ax\|$.
- A necessary and sufficient theorem for convergence is given next (we will not prove this).

Theorem 2

The iterative method $x^{k+1} = x^k + M^{-1}(b - Ax^k)$ *converges for any* x^0 *and* b *if and only if* $\rho(I - M^{-1}A) < 1$.

- The speed of convergence depends on the size of $\rho(I - M^{-1}A)$ since the error satisfies

$$\|x^{k+1} - x^*\| \leq \rho(I - M^{-1}A)\|x^k - x^*\|.$$

- That is, magnitude of the error scales by $\rho(I - M^{-1}A)$ on each iteration.
- We will call $\rho(I - M^{-1}A)$ the **convergence factor**.
- Smaller ρ (closer to zero) implies faster convergence.
- For even more on convergence see Saad textbook, sections 4.2.1 and 1.8.4.