3. Conjugate Gradient (CG) Algorithm

The objective of this case study is to implement the Conjugate Gradient algorithm to solve a square linear system Ax = b, where A is a symmetric, positive definite, and regular matrix, and the right-hand side vector b is nonzero.

3.1 Fundamentals: The Poisson Problem

The Poisson problem is a common type of partial differential equation. Its standard form is

$-\Delta u(x) = f(x)$, with Dirichlet boundary conditions.

Intuitively, the problem can be viewed as follows: in a square domain, determine the value at each interior point based on the "instructions" provided by the function f(x), while the function values on the edges of the square are fixed at 0.

Step01: Discretize with Finite Difference Method

Computers cannot directly work with continuous functions or differential equations, so the continuous problem must be converted into a discrete one. In this case, we discretize the entire square region by constructing a grid within the domain, with each grid point representing a discrete location. By dividing the domain into a grid and restricting the function values to the grid points, we approximate the original differential equation with algebraic equations.

1. **Interior and Boundary Nodes:**
   The boundary nodes (i.e., the points where i = 0 or i = N, or j = 0 or j = N) have $u(x) = 0$, which are known. Thus, we only need to solve for the values at the interior nodes (those not on the boundary).

2. **Determining the Grid:**

   On the unit square Ω, we discretize the problem into a regular two-dimensional grid. Divide the unit square into N+1 nodes (including the boundary nodes), where the spacing between points is given by $h = 1/N$. For example, when N = 8, you divide the square into 9 equally spaced points (from 0 to 1).

Step02: Approximate the Differential Operator Using the Finite Difference Method

Since the continuous second derivatives cannot be computed directly on a computer, we approximate them using the finite difference method. For two-dimensional problems, the standard five-point finite difference scheme is used to approximate the Laplace operator. The five-point difference formula provides a good approximation of the two-dimensional Laplacian, and its simplicity makes it easy to implement in code.

1. **Five-Point Difference Formula:**

   At each interior node (i, j), the values of the function at the four neighboring nodes are used to approximate the Laplace operator. The formula is:

   $$\Delta u(x_{(i,j)}) \approx (u_(i-1,j_) + u_(i+1,j_) + u_(i,j-1_) + u_(i,j+1_) - 4u_(i,j_))/h^2.$$

   This formula tells us that the second derivative (or "curvature") at a point can be approximated by the values at that point and its immediate neighbors in the up, down, left, and right directions.

2. **Substituting into the Original Equation:**

   The original partial differential equation is

   $$-\Delta u(x) = f(x).$$

   Multiplying by h², it can be rewritten as:

   $$-u_(i-1,j_) - u_(i+1,j_) - u_(i,j-1_) - u_(i,j+1_) + 4u_(i,j_) = h^2 f_(i,j_).$$

   Here,$f_(i,j_)$ represents the value of f(x) at the node(x_i, x_j). In this way, each interior node provides an algebraic equation.

Step03: Construct the Linear System Ay = b

1. **Defining the Unknown Vector y:**
   - The vector y contains the values of u(x) at all the interior nodes, since the boundary values are already known to be 0 and do not need to be solved. Only the interior nodes need to be solved for, and the boundary values are simply "filled in" as zeros.

   - If there are N+1 grid points per dimension, then the number of interior nodes is $(N-1)^2$. That is, y is a vector of length $(N-1)^2$.

2. **Determining the Grid Point Ordering :**

   - Mapping the **Two-Dimensional Points** to a One-Dimensional Vector. Common Method: Lexicographic (Row-Wise) Ordering

   For example, map the 2D point (i, j) to a one-dimensional index k using:

   $k = i + (j-1) * (N-1), \quad$ for $i, j = 1, \ldots, N-1.$

   With this ordering, we can store the values at each interior node sequentially in the vector y.

3. **Constructing the Matrix A:**

   For each interior node (i, j), the discretized equation includes:

   - A diagonal entry (corresponding to $u_(i, j_)$) with a coefficient of 4,

   - Off-diagonal entries (corresponding to the neighboring nodes $u_(i \pm 1, j_)$ and $u_(i, j \pm 1_)$) with coefficients of -1.

   Since many elements of A are zero (each equation involves only the current node and its neighbors), A is a sparse matrix—a significant computational advantage. This efficiency is crucial for later applying iterative methods like the Conjugate Gradient algorithm to solve the system.

4. **Constructing the Right-Hand Side Vector b:**

   For each interior node (i, j), the right-hand side in the finite difference equation is $h^2 f_(i, j_)$.
   Thus, each component of the vector b is
   h^2 multiplied by the value of $f(x)$ at the corresponding node.

If an interior node is adjacent to a boundary, even though the finite difference formula may include values corresponding to boundary nodes, those values are 0 (and thus known) and can be neglected.