
H L R S
High-Performance
Computing Center
Stuttgart
Solving the 2d Poisson Equation using the FEM and a CG Method

Finite Element Method

Strong Formulation of the Poisson Equation

Goal: Find the unknown function u (e.g., temperature, potential) on a domain Ω, given a source f (e.g., heat source).

This is described by the **strong form**. Find $u : \Omega \to \mathbb{R}$ such that

$$-\Delta u = f \quad \in \Omega \quad \text{(Poisson's Equation)} \tag{1}$$

$$u = 0 \quad \in \partial\Omega \quad \text{(Boundary Condition)} \tag{2}$$

Laplace operator Δ:

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{3}$$

Problem: Requires u to be twice-differentiable. This is too restrictive and difficult to work with computationally.

Notation for the L^2-inner product:

$$(f, g) := \int_\Omega f \cdot g \, \mathrm{d}x \tag{4}$$

[1]

Weak Formulation of the Poisson Equation I

Idea: We „weaken“ the requirement from C^2 to C^1 by testing the equation in an integral sense.

Define function space V for both the solution u and „test functions“ v:

$$V := H_0^1(\Omega), \tag{5}$$

This space V contains all functions that are one time (weakly) differentiable and are zero on the boundary $\partial\Omega$.

Multiply the PDE by a test function $v \in V$ and integrate over Ω:

$$(-\Delta u, v) = (f, v) \quad \forall v \in V \tag{6}$$

Apply integration by parts (Green's theorem) to the left-hand side to move a derivative from u onto v:

$$(\nabla u, \nabla v) - \int_{\partial\Omega} \partial_n u \phi \, \mathrm{d}s = (f, \phi) \quad \forall \phi \in V \tag{7}$$

[1]

Weak Formulation of the Poisson Equation II

Since our test functions $v \in V$ are 0 on the boundary ($v = 0$ on $\partial\Omega$), the boundary integral vanishes.

This gives the **weak formulation**: Find $u \in V$ such that

$$(\nabla u, \nabla v) = (f, v) \quad \forall v \in V \tag{8}$$

This is often written abstractly as $a(u, v) = L(v)$, where:

- $a(u, v) := (\nabla u, \nabla v)$ is the **bilinear form** (handles stiffness).
- $L(v) := (f, v)$ is the **linear form** (handles the source/load).

[1]

Meshing and Function Space I

Problem: The space $V = H_0^1$ is infinite-dimensional.
Solution: We approximate V with a finite-dimensional subspace $V_h \subset V$.

Step 1: Meshing
We discretize the domain Ω into a finite number of simple shapes (elements), like triangles. The collection of these elements is the **mesh**, $\mathcal{T}_h$. The parameter h represents the mesh size (e.g., max triangle width).

[2, 1]

Meshing and Function Space II

Step 2: Define V_h

We define V_h as the space of functions that are:

- Continuous across the whole domain Ω.
- A simple polynomial (e.g., linear) on each triangle $K \in \mathcal{T}_h$.
- Still zero on the boundary $\partial\Omega$.

This is the space of continuous, piecewise-linear functions.

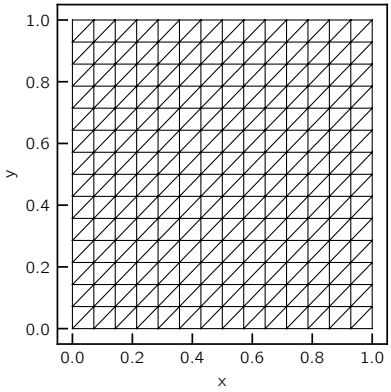

Figure: A mesh $\mathcal{T}_h$ of the unit square.

[2, 1]

Basis Functions (P1 Elements) I

How do we build a basis for V_h? We use **local basis functions** (or **shape functions**) ϕ_i associated with each node N_i of the mesh.

For P1 (piecewise-linear) elements, ϕ_i has a key property:

- ϕ_i is 1 at its „own“ node N_i.
- ϕ_i is 0 at all other nodes N_j ($j \neq i$).
- ϕ_i is linear on each triangle.

This is the „hat function“.

[2, 1]

Basis Functions (P1 Elements) II

Area Coordinates
On a single triangle with nodes $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, these shape functions are identical to the area coordinates L_i:

$$x = L_1 x_1 + L_2 x_2 + L_3 x_3 \tag{9}$$
$$y = L_1 y_1 + L_2 y_2 + L_3 y_3 \tag{10}$$
$$1 = L_1 + L_2 + L_3 \tag{11}$$

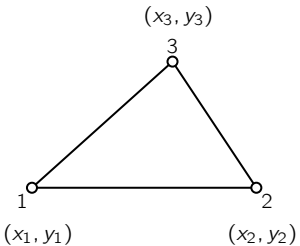

Figure: A single linear triangular element.

[2, 1]

Explicit Formulas

The area coordinates L_i are linear functions of (x, y):

$$L_1 = \frac{a_1 + b_1 x + c_1 y}{2A}, \quad a_1 = x_2 y_3 - x_3 y_2,\ b_1 = y_2 - y_3,\ c_1 = x_3 - x_2 \tag{12}$$

$$L_2 = \frac{a_2 + b_2 x + c_2 y}{2A}, \quad a_2 = x_3 y_1 - x_1 y_3,\ b_2 = y_3 - y_1,\ c_2 = x_1 - x_3 \tag{13}$$

$$L_3 = \frac{a_3 + b_3 x + c_3 y}{2A}, \quad a_3 = x_1 y_2 - x_2 y_1,\ b_3 = y_1 - y_2,\ c_3 = x_2 - x_1 \tag{14}$$

Local Shape Functions

On this linear triangle element, the three local shape functions are just:

$$\phi_1(x, y) = L_1 \quad \phi_2(x, y) = L_2 \quad \phi_3(x, y) = L_3 \tag{15}$$

[2, 1]

Gradients of Shape Functions

Since the shape functions $\phi_i = L_i$ are linear, their gradients are constant on each element:

$$\nabla \phi_i = \begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \end{bmatrix} \tag{16}$$

$$\frac{\partial \phi_1}{\partial x} = \frac{y_2 - y_3}{2A} \tag{17}$$

$$\frac{\partial \phi_2}{\partial x} = \frac{y_3 - y_1}{2A} \tag{18}$$

$$\frac{\partial \phi_3}{\partial x} = \frac{y_1 - y_2}{2A} \tag{19}$$

$$\frac{\partial \phi_1}{\partial y} = \frac{x_3 - x_2}{2A} \tag{20}$$

$$\frac{\partial \phi_2}{\partial y} = \frac{x_1 - x_3}{2A} \tag{21}$$

$$\frac{\partial \phi_3}{\partial y} = \frac{x_2 - x_1}{2A} \tag{22}$$

[2, 1]

The Galerkin Problem (Discrete Form)
We now solve the weak form on the finite-dimensional space V_h:
Find $u_h \in V_h$ such that

$$(\nabla u_h, \nabla v_h) = (f, v_h) \quad \forall v_h \in V_h \tag{23}$$

As the mesh size $h \to 0$, the discrete solution u_h converges to the true solution u.

[2, 1]

Converting into a Linear Equation System I

Any function u_h in our space V_h can be written as a linear combination of its basis functions ϕ_j:

$$u_h(x, y) = \sum_{j=1}^{n} u_j \phi_j(x, y) \tag{24}$$

Here, n is the total number of nodes in the mesh. Because $\phi_j(N_i) = \delta_{ij}$, the unknown coefficients u_j are simply the **unknown values of the solution at the nodes**: $u_j = u_h(N_j)$.

Converting into a Linear Equation System II

Substitute this „ansatz“ into the Galerkin problem:

$$(\nabla u_h, \nabla \phi_h) = (f, \phi_h) \quad \forall \phi_h \in V_h \tag{25}$$

$$\left(\nabla \left(\sum_{j=1}^{n} u_j \phi_j \right), \nabla \phi_h \right) = (f, \phi_h) \quad \forall \phi_h \in V_h \tag{26}$$

$$\sum_{j=1}^{n} u_j \left(\nabla \phi_j, \nabla \phi_h \right) = (f, \phi_h) \quad \forall \phi_h \in V_h \tag{27}$$

This must hold for *all* $\phi_h \in V_h$. It is sufficient to test it against *each basis function* ϕ_i (for $i = 1, \ldots, n$):

$$\sum_{j=1}^{n} u_j \left(\nabla \phi_j, \nabla \phi_i \right) = (f, \phi_i) \quad \forall i \in \{1, \ldots, n\} \tag{28}$$

Converting into a Linear Equation System III

This is a linear system of equations $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{F}$!

$$\underbrace{\begin{bmatrix} (\nabla\phi_1, \nabla\phi_1) & \dots & (\nabla\phi_n, \nabla\phi_1) \\ \vdots & \ddots & \vdots \\ (\nabla\phi_1, \nabla\phi_n) & \dots & (\nabla\phi_n, \nabla\phi_n) \end{bmatrix}}_{\text{Stiffness matrix } \boldsymbol{A}} \underbrace{\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}}_{\boldsymbol{u}} = \underbrace{\begin{bmatrix} (f, \phi_1) \\ \vdots \\ (f, \phi_n) \end{bmatrix}}_{\text{Load vector } \boldsymbol{F}} \tag{29}$$

The entries are:

$$A_{ij} = (\nabla\phi_j, \nabla\phi_i) := \int_\Omega \nabla\phi_j \cdot \nabla\phi_i \, \mathrm{d}x \tag{30}$$

$$F_i = (f, \phi_i) := \int_\Omega f\phi_i \, \mathrm{d}x \tag{31}$$

Our goal is to **solve** this system for the unknown vector of nodal values $\boldsymbol{u}$.

Assembling and Solving the System I

Algorithm 1: Global Assembly

Input : Mesh $\mathcal{T}_h$, Source f

Initialize $\boldsymbol{A} = 0$ (sparse $n \times n$ matrix)
Initialize $\boldsymbol{F} = 0$ ($n \times 1$ vector)
foreach *element* e *in mesh* $\mathcal{T}_h$ **do**
    Initialize $\boldsymbol{A}^e = 0$ (3×3 local matrix)
    Initialize $\boldsymbol{F}^e = 0$ (3×1 local vector)
    Let ϕ_1, ϕ_2, ϕ_3 be the local shape functions on e
    for $i = 1$ **to** 3 **do**
        $F_i^e = \int_e f \phi_i \, \mathrm{d}x$
        for $j = 1$ **to** 3 **do**
            $A_{ij}^e = \int_e \nabla \phi_j \cdot \nabla \phi_i \, \mathrm{d}x$
        end
    end
    for $i = 1$ **to** 3 **do**
        $\boldsymbol{F}[g_i] \mathrel{+}= F_i^e$
        for $j = 1$ **to** 3 **do**
            $\boldsymbol{A}[g_i, g_j] \mathrel{+}= A_{ij}^e$
        end
    end
end
ApplyDirichletBC($\boldsymbol{A}$, $\boldsymbol{F}$)
return $\boldsymbol{A}$, $\boldsymbol{F}$

Assembling and Solving the System II

Structure of the Matrix $\boldsymbol{A}$

- $\boldsymbol{A}$ is a large, **sparse matrix**.
- The entry A_{ij} is non-zero **only if** nodes i and j share an element.
- It is also **symmetric** ($A_{ij} = A_{ji}$) and **positive-definite**.

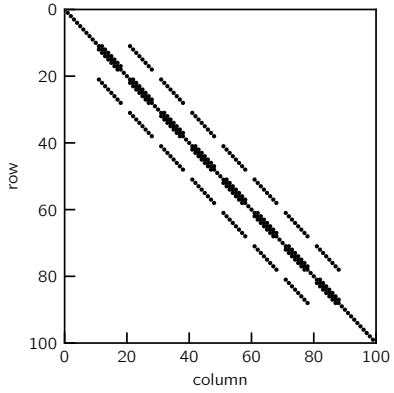

Figure: Sparsity pattern of $\boldsymbol{A}$.

Assembling and Solving the System III

Solving the System

Because $\boldsymbol{A}$ is sparse, symmetric, and positive-definite, we don't use direct solvers (like $\boldsymbol{A}^{-1}$). We use efficient **iterative solvers** like the **Conjugate Gradient (CG) Method**.

Algorithm 2: Conjugate Gradient Method

Input : Matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, initial guess $\boldsymbol{x}_0$
Output: Solution $\boldsymbol{x}$

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$
$\boldsymbol{p}_0 = \boldsymbol{r}_0$
for $k = 0$ **to** *maxit - 1* **do**
  $\alpha_k = (\boldsymbol{r}_k^\mathsf{T}\boldsymbol{r}_k)/(\boldsymbol{p}_k^\mathsf{T}\boldsymbol{A}\boldsymbol{p}_k)$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k$
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k\boldsymbol{A}\boldsymbol{p}_k$
  if $\|\boldsymbol{r}_{k+1}\|_2$ <= *tol* **then**
    return $\boldsymbol{x}_{k+1}$
  $\beta_k = (\boldsymbol{r}_{k+1}^\mathsf{T}\boldsymbol{r}_{k+1})/(\boldsymbol{r}_k^\mathsf{T}\boldsymbol{r}_k)$
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k\boldsymbol{p}_k$
end
return $\boldsymbol{x}_{k+1}$

Solution of the Poisson Equation

After solving $\boldsymbol{Au} = \boldsymbol{F}$ for the vector $\boldsymbol{u}$, we have the value u_j at each node. We can then reconstruct the full solution $u_h = \sum u_j \phi_j$ for visualization.

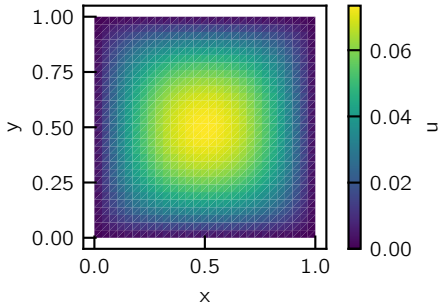

Figure: The computed discrete solution u_h for a source $f = 1$ on the unit square.

Bibliography

Bibliography I

[1] T Wick. *Numerical Methods for Partial Differential Equations*. en. Hannover : Institutionelles Repositorium der Leibniz Universität Hannover, 2020. DOI: 10.15488/9248.

[2] OC Zienkiewicz and RL Taylor. *The finite element method. Volume 1: The basis*. 5th ed. Oxford: Butterworth-Heinemann, 2002. 689 pp. URL: https://www.meil.pw.edu.pl/content/download/58297/306302/file/FEM_Zienkiewicz%20Vol1.pdf (visited on 10/28/2025).