Progress Report 3 - Standard Function and E3nn Studies

Objective

The goal of this project is to numerically determine the parametric coordinates (ξ, η, ζ) of a point within a trilinear hexahedral finite element, given its physical space coordinates $\mathbf{x}_{\text{phys}} = (x, y, z)$. This process is fundamental in finite element analysis (FEA), where transformations between the reference (parent) domain and the physical domain are essential for interpolation, numerical integration, and applying boundary conditions.

Understanding the Geometry and Mapping

In finite elements, we define elements in a simpler, idealized coordinate system called the *reference domain* or *parent domain*. For 3D hexahedral elements, this domain is the cube $[-1, 1]^3$. Each point (ξ, η, ζ) in this cube can be mapped to a *physical domain*, which represents the actual shape and location of the element in the mesh.

To perform this mapping, we use shape functions based on the trilinear interpolation scheme. Each shape function $N_i(\xi, \eta, \zeta)$ is associated with a node of the hexahedron and satisfies:

- $N_i = 1$ at node i
- $N_i = 0$ at all other nodes

The physical location $\mathbf{x}(\xi, \eta, \zeta)$ is given by:

$$\mathbf{x}(\xi, \eta, \zeta) = \sum_{i=1}^{8} N_i(\xi, \eta, \zeta) \cdot \mathbf{x}_i$$

This is the *forward mapping*. The inverse mapping — finding (ξ, η, ζ) for a given $\mathbf{x}_{\text{phys}}$ — must be done numerically because of the nonlinearity of N_i.

Shape Functions

Each N_i is a product of linear terms:

$$N_1(\xi, \eta, \zeta) = \frac{1}{8}(1 - \xi)(1 - \eta)(1 - \zeta)$$

with analogous forms for the other 7 nodes.

Why We Need the Inverse Mapping

To evaluate integrals using Gauss quadrature or determine interpolation values at specific physical points, we must express those points in the reference domain. Hence, the inverse mapping is necessary.

Newton-Raphson Method

We define the residual function:

$$\mathbf{r}(\xi, \eta, \zeta) = \mathbf{x}_{\text{phys}} - \mathbf{x}(\xi, \eta, \zeta)$$

and seek its root:

$$\mathbf{r}(\xi,\eta,\zeta) = 0 \quad \Leftrightarrow \quad \mathbf{x}(\xi,\eta,\zeta) = \mathbf{x}_{\text{phys}}$$

This nonlinear system is solved using the Newton-Raphson method. At each iteration k:

$$\begin{aligned} \mathbf{x}^{(k)} &= \sum_i N_i(\xi^{(k)},\eta^{(k)},\zeta^{(k)}) \cdot \mathbf{x}_i \\ \mathbf{r}^{(k)} &= \mathbf{x}_{\text{phys}} - \mathbf{x}^{(k)} \\ \Delta\boldsymbol{\xi} &= J^{-1} \cdot \mathbf{r}^{(k)} \\ \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix}^{(k+1)} &= \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix}^{(k)} + \Delta\boldsymbol{\xi} \end{aligned}$$

We repeat until $\|\mathbf{r}^{(k)}\|$ is sufficiently small.

Jacobian Matrix

The Jacobian matrix J describes the derivative of the physical coordinates with respect to the parametric coordinates:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}$$

It is computed as:

$$J = \sum_{i=1}^{8} \nabla N_i(\xi,\eta,\zeta) \otimes \mathbf{x}_i$$

where $\nabla N_i = \left[\frac{\partial N_i}{\partial \xi}, \frac{\partial N_i}{\partial \eta}, \frac{\partial N_i}{\partial \zeta}\right]$, and $\otimes$ is the outer product.

Implementation Summary

- **Step 1: Input validation.** Check shape of input arrays.
- **Step 2: Bounding box check.** Reject points obviously outside the element.
- **Step 3: Newton-Raphson.**
  - Initialize $(\xi,\eta,\zeta) = (0,0,0)$
  - Evaluate shape functions and Jacobian
  - Update until convergence
- **Step 4: Validate solution.** Ensure final $(\xi,\eta,\zeta) \in [-1,1]^3$

Conceptual Summary

- The residual $\mathbf{r} = \mathbf{x}_{\text{phys}} - \mathbf{x}(\xi, \eta, \zeta)$ is driven toward zero.
- The Jacobian gives local transformation info.
- Convergence depends on the elements shape and the initial guess.

This work implements a core capability used throughout finite element software and analysis.