AME50541: Finite Element Methods
Final Project: Due Wednesday, May 8, 2019

In this project, you will enhance the finite element code you developed throughout the semester in the homework assignments into a fully general FEM code capably of handling unstructured meshes, non-homogeneous natural boundary conditions, and nonlinear problems. You will then use your code to solve a number of mechanics problems.

Instructions

- *All starter code can be found on the course website in the final project code distribution. Be sure to carefully read* `notation.m` *for a description of all relevant variables.* Be sure to run `init.m` each time you start MATLAB to add all required directories to your MATLAB path.
- You may assume `ndim` is either 1 or 2 in all coding tasks *without penalty*. That is, you only need your finite element code to work in one and two dimensions to receive full credit. I will award 2% extra credit on each task if your code works correctly for `ndim` is 1, 2, or 3, i.e., your finite element code works in one, two, and three dimensions.
- *Be sure to email all code to the instructor and TA.*

Part 1: (50 points) Since we want our code to be able to handle arbitrarily complex domains, it is necessary to have *simplex* elements available in our FEM code. As we discussed in class, mesh generation with simplex elements (triangles in 2d, tetrahedra in 3d) is a "solved" problem while it is much more difficult to do with hypercube elements (high-quality mesh generation with hypercube elements for arbitrary domains is particularly challenging in 3+ dimensions and remains an open problem).

Part 1.1 First, we need to define the geometry of our element, which is given by the element interior (volume) and its boundary (faces). In addition, nodes are distributed throughout the element based on its degree of polynomial completeness. Simplex elements of order p, i.e., polynomial completeness of degree p, in d-dimension use interpolation functions that include all multinomial terms of order p $\{\xi_1^{\alpha_1} \cdots \xi_d^{\alpha_d} \mid \sum_{i=1}^{d} \alpha_i \leqslant p\}$. Therefore, the number of nodes in a simplex element must be the number of unique multinomials

$$n_v = \sum_{n=0}^{p} \binom{n+d-1}{d-1}. \tag{1}$$

For the special case of $d = 2$ (triangle), we have $n_v = (p+1)(p+2)/2$ nodes, which can easily be verified by referring to the Pascal triangle or the formula in (1).

Since we will use the isoparametric concept to define basis functions, we only need to consider the unit right simplex. We assume the nodes are distributed uniformly throughout the element and number first in the ξ_1-direction, then in the ξ_2-direction, etc., where $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_d)$ are the coordinates in the reference domain (Figure 1). Furthermore, let us number the $n_f = d + 1$ faces of the simplex element as: face i is the face with unit normal $-\boldsymbol{e}_i$ for $i = 1, \ldots, d$, where $\boldsymbol{e}_i \in \mathbb{R}^d$ is the canonical unit vector, and face $d + 1$ is the remaining face.

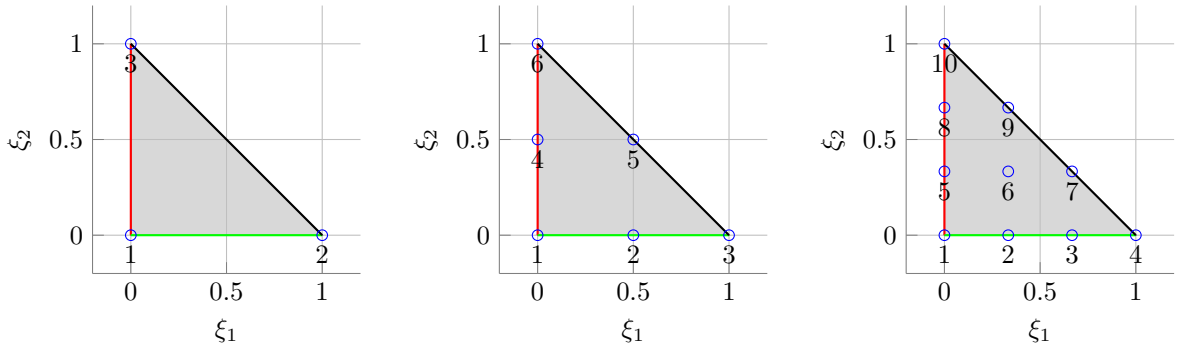

Figure 1: Triangular finite element in reference domain (ξ_1-ξ_2 space) with 3 nodes ($p = 1$) (*left*), 6 nodes ($p = 2$) (*center*), and 10 nodes ($p = 3$) (*right*). The faces are numbered as: face 1 (——), face 2 (——), and face 3 (——).

Tasks for Part 1.1

Write a function that defines the geometry of a simplex element of order p in the reference domain in d spatial dimensions. Your function should define the nodal positions, in the reference domain `zk`, a mapping from the face number to the nodes that lie on that particular face `f2v`, and the unit normal for each face `N`. Your function should have the following signature:

```
function [zk, f2v, N] = create_nodes_bndy_refdom_simp(ndim, porder)
%CREATE_NODES_BNDY_REFDOM_SIMP Create nodal distribution and boundary of
%NDIM-dimensional simplex of order PORDER.
%
%Input arguments
%---------------
%   NDIM, PORDER : See notation.m
%
%Output arguments
%----------------
%   ZK : Array (NDIM, NV) : The nodal positions of the reference simplex
%     element in NDIM-dimensions and polynomial order PORDER.
%
%   F2V, N : See notation.m
```

Read all comments and `notation.m` carefully for instructions regarding the inputs and outputs to the function. For the $d = 2$ case (triangular element), plot all nodes of the element with blue circles. On top of these circles, plot the nodes on face 1 (left edge) with red `x`'s, the nodes of face 2 (bottom edge) with green `□`'s, and the nodes on face 3 (hypotenuse) with black `+`'s.

Part 1.2 Next, we need to define basis functions over the reference domain $\{\psi_i(\boldsymbol{\xi})\}_{i=1}^{n_v}$. For this we will use the direct approach discussed in class. A basis for a simplex element of order p must contain all multinomial terms of order p in d dimensions $\{\xi_1^{\alpha_1} \cdots \xi_d^{\alpha_d} \mid \sum_{i=1}^d \alpha_i \leqslant p\}$, so we can write our n_v basis functions as

$$\psi_i(\boldsymbol{\xi}) = \sum_{k=1}^{n_v} \alpha_{ik} \prod_{j=1}^{d} \xi_j^{\kappa_{jk}} \tag{2}$$

where $\boldsymbol{\kappa}$ is a matrix of natural numbers (including zero) with entries κ_{ij}, $i = 1, \ldots, d$, $j = 1, \ldots, n_v$, such that $\sum_{i=1}^d \kappa_{ij} \leqslant p$ for each $j = 1, \ldots, n_v$ that is used to sweep over all n_v permissible exponents. For example:

- in the special case of $d = 2$ (triangle) and $p = 1$, we have

$$\boldsymbol{\kappa} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \Longrightarrow \quad \psi_i(\xi_1, \xi_2) = \alpha_{i1} + \alpha_{i2}\xi_1 + \alpha_{i3}\xi_2$$

- in the special case of $d = 2$ and $p = 2$, we have

$$\boldsymbol{\kappa} = \begin{bmatrix} 0 & 1 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 2 \end{bmatrix} \quad \Longrightarrow \quad \psi_i(\xi_1, \xi_2) = \alpha_{i1} + \alpha_{i2}\xi_1 + \alpha_{i3}\xi_2 + \alpha_{i4}\xi_1^2 + \alpha_{i5}\xi_1\xi_2 + \alpha_{i6}\xi_2^2$$

- in the special case of $d = 3$ (tetrahedron) and $p = 1$, we have

$$\boldsymbol{\kappa} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \Longrightarrow \quad \psi_i(\xi_1, \xi_2, \xi_3) = \alpha_{i1} + \alpha_{i2}\xi_1 + \alpha_{i3}\xi_2 + \alpha_{i4}\xi_3.$$

For convenience, we introduce the function $\omega_i(\boldsymbol{\xi})$, $i = 1, \ldots, n_v$

$$\omega_i(\boldsymbol{\xi}) = \prod_{s=1}^{d} \xi_s^{\kappa_{si}},$$

so the basis functions can conveniently be expressed as $\psi_i(\boldsymbol{\xi}) = \sum_{k=1}^{n_v} \alpha_{ik}\omega_k(\boldsymbol{\xi})$.

Denote the n_v nodes of the pth order simplex element as $\{\hat{\boldsymbol{\xi}}_i\}_{i=1}^{n_v}$, where $\hat{\boldsymbol{\xi}}_i = (\hat{\xi}_{1i}, \ldots, \hat{\xi}_{di})^T$. The Lagrangian property is

$$\psi_i(\hat{\boldsymbol{\xi}}_j) = \delta_{ij},$$

for $i, j = 1, \ldots, n_v$, which leads to

$$\sum_{k=1}^{n_v} \alpha_{ik}\omega_k(\hat{\boldsymbol{\xi}}_j) = \delta_{ij}$$

once the expression for $\psi_i(\boldsymbol{\xi})$ is used from (2). Let $\hat{V}_{ij} = \omega_j(\hat{\boldsymbol{\xi}}_i) = \prod_{s=1}^{d} \hat{\xi}_{si}^{\kappa_{sj}}$ be the Vandermonde matrix corresponding to the d-dimensional, pth order simplex evaluated at $\{\hat{\boldsymbol{\xi}}_i\}_{i=1}^{n_v}$, then the above constraints can be written in matrix form as $\hat{\boldsymbol{V}}\boldsymbol{\alpha}^T = \boldsymbol{I}_{n_v}$, where $\hat{\boldsymbol{V}}$, $\boldsymbol{\alpha}$ are the matrices with indices $\hat{V}_{ij}$, α_{ij}, respectively, and $\boldsymbol{I}_{n_v}$ is the $n_v \times n_v$ identity matrix. Once we compute the coefficients, $\boldsymbol{\alpha} = \hat{\boldsymbol{V}}^{-T}$, we substitute this expression for the coefficients into (2) and evaluate at new points $\{\tilde{\boldsymbol{\xi}}_i\}_{i=1}^{m}$ where $\tilde{\boldsymbol{\xi}}_i = (\tilde{\xi}_{1i}, \ldots, \tilde{\xi}_{di})$ to give

$$\psi_i(\tilde{\boldsymbol{\xi}}_j) = \sum_{k=1}^{n_v} \alpha_{ik}\omega_k(\tilde{\boldsymbol{\xi}}_j) = \sum_{k=1}^{n_v} \hat{V}_{ki}^{-1}\omega_k(\tilde{\boldsymbol{\xi}}_j) = \sum_{k=1}^{n_v} \hat{V}_{ki}^{-1}\tilde{V}_{jk} \tag{3}$$

where the last expression used the d-dimensional, pth order simplex Vandermonde matrix evaluated at $\{\tilde{\boldsymbol{\xi}}_i\}_{i=1}^{m}$: $\tilde{V}_{ij} = \omega_j(\tilde{\boldsymbol{\xi}}_i) = \prod_{s=1}^{d} \tilde{\xi}_{si}^{\kappa_{sj}}$. Therefore, if we define $Q_{ij} = \psi_i(\tilde{\boldsymbol{\xi}}_j)$, we have

$$\boldsymbol{Q} = \hat{\boldsymbol{V}}^{-T}\tilde{\boldsymbol{V}}^T,$$

where $\boldsymbol{Q}$, $\tilde{\boldsymbol{V}}$ are the matrices with indices Q_{ij}, $\tilde{V}_{ij}$, repsectively.

The partial derivatives of the simplex basis functions are also needed to implement the finite element method. A simple differentiation calculation reveals

$$\frac{\partial \psi_i}{\partial \xi_j}(\boldsymbol{\xi}) = \sum_{k=1}^{n_v} \alpha_{ik} \frac{\partial \omega_k}{\partial \xi_j}(\boldsymbol{\xi}),$$

where the partial derivatives of $\omega_i(\boldsymbol{\xi})$ are

$$\frac{\partial \omega_i}{\partial \xi_j}(\boldsymbol{\xi}) = \begin{cases} 0 & \text{if } \kappa_{ji} = 0 \\ \kappa_{ji}\xi_j^{\kappa_{ji}-1} \displaystyle\prod_{s=1, s\neq j}^{d} \xi_s^{\kappa_{si}} & \text{if } \kappa_{ji} \neq 0. \end{cases}$$

Then, the basis functions evaluated at the points $\{\tilde{\boldsymbol{\xi}}_i\}_{i=1}^m$, take the form

$$\frac{\partial \psi_i}{\partial \xi_j}(\tilde{\boldsymbol{\xi}}_k) = \sum_{l=1}^{n_v} \alpha_{il} \frac{\partial \omega_l}{\partial \xi_j}(\tilde{\boldsymbol{\xi}}_k) = \sum_{l=1}^{n_v} \hat{V}_{li}^{-1} \tilde{W}_{klj},$$

where $\tilde{W}_{ijk}$ contains the partial derivatives of the Vandermonde matrix evaluated at $\{\tilde{\boldsymbol{\xi}}_i\}_{i=1}^{n_v}$, i.e., $W_{ijk} = \frac{\partial \omega_j}{\partial \xi_k}(\tilde{\boldsymbol{\xi}}_i)$.

Tasks for Part 1.2

Your task is to write a function that evaluates the basis functions (and their derivatives) of a pth order, d-dimensional simplex element.

- First you need to implement a function that evaluates the Vandermonde matrix and its derivative corresponding to the d-dimensional, pth order simplex. Your function should have the following signature:

```
function [V, dV] = vander_simp(porder, x)
%VANDER_SIMP Compute NDIM-dimensional Vandermonde matrix of order PORDER
%for a  simplex and its derivative (NDIM determined from shape of X). The
%Vandermonde matrix, V, is the NX x M matrix  of multinomial terms and its
%derivative, dV, is the NX x M x NDIM matrix of the partial derivatives of
%multinomial terms
%
%    V(i, k) = x(1, k)^I(1, i) * ... * x(NDIM, k)^I(NDIM, i),
%
%    dV(i, 1, k) = d(V(i,k))/dx1 = I(1, i)*x(1, k)^(I(1, i)-1) * x(2, k)^I(2, i) * ...
%
%where M is the number of multinomial terms required for polynomial
%completeness (all combinations such that the sum of the exponents is <=
%porder), X are the evaluation points, I is a matrix defining the
%multinomial exponents (completeness requires sum(I, 1) <= porder).
%
%Input arguments
%-----------------
%   PORDER : Polynomial degree of completeness
%
%   X : Array (NDIM, NX) : Points at which to evaluate multinomials in
%     definition of Vandermonde matrix
%
%Output arguments
%------------------
%   V : Array (M, NX) : Vandermonde matrix
%
%   dV : Array (M, NDIM, NX) : Derivative of Vandermonde matrix
```

- Next, you will need to implement a function that evaluates the basis functions and their derivatives for a d-dimensional simplex of order p given the coordinates of the element nodes `xk` and points at which to evaluate the basis `x` (these will eventually be quadrature points). Your function should have the following signature:

```
function [Q, dQ] = eval_interp_simp_lagrange(xk, x)
%EVAL_INTERP_SIMP_LAGRANGE Evaluate interpolation functions for simplex
%using Lagrange polynomials (number of spatial dimensions and polynomial
%degree of completeness determined from the nodes of the element XK).
%
%Input arguments
%-----------------
%   XK : Array (NDIM, NV): Nodes of simplex element.
%
%   X : Array (NDIM, NX) : Points at which to evaluate interpolation
```

```
%     function.
%
%Output arguments
%------------------
%   Q : Array (NV, NX) : Interpolation functions for the
%     simplex element evaluated at each point in X, i.e.,
%     Q(i, k) = phi_i(X(:, k) where phi_i(x) is the ith basis fucntion.
%
%   DQ : Array (NV, NDIM, NX) : Derivative of interpolation
%     functions evaluated at each point in X, i.e.,
%     dQ(i, j, k) = (d(phi_i)/d(x_j))(X(:, k)
```

- Check the correctness of your implementation using known properties of a Lagrangian basis:
  - Lagrangian property: $\psi_i(\hat{\boldsymbol{\xi}}_j) = \delta_{ij}$
  - Partition of unity:

$$\sum_{i=1}^{n_v} \psi_i(\boldsymbol{\xi}) = 1, \qquad \sum_{i=1}^{n_v} \frac{\partial \psi_i}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) = \mathbf{0}$$

    for any $\boldsymbol{\xi} \in \Omega_\square$.

- Also check the derivatives of your basis functions are correct by comparing to a finite difference approximation

$$\frac{\partial \psi_i}{\partial \boldsymbol{\xi}_j}(\boldsymbol{\xi}) \approx \frac{\psi_i(\boldsymbol{\xi} + \epsilon \boldsymbol{e}_j) - \psi_i(\boldsymbol{\xi} - \epsilon \boldsymbol{e}_j)}{2\epsilon},$$

  where ϵ is a small number (but not too small to avoid significant floating point errors), e.g., 10^{-6}, and $\boldsymbol{e}_j \in \mathbb{R}^d$ is the jth canonical unit vector.

- Implement a function that completely defines a simplex element in the reference domain, $\Omega_\square$ and $\Gamma_\square$, as a MATLAB structure. This structure should contain: the nodal coordinates of $\Omega_\square$ ($\{\hat{\boldsymbol{\xi}}_i\}_{i=1}^{n_v}$) and $\Gamma_\square$ ($\{\hat{\boldsymbol{r}}_i\}_{i=1}^{n_v^f}$), the face-to-vertex mapping (`f2v`) and unit normals of each face (from Part 1.1), a quadrature rule for $\Omega_\square$ ($\{(w_k, \tilde{\boldsymbol{\xi}}_i)\}_{i=1}^{n_q}$) and $\Gamma_\square$ ($\{(w_k^f, \tilde{\boldsymbol{r}}_i)\}_{i=1}^{n_q^f}$), and the basis functions evaluated at the appropriate quadrature nodes for $\Omega_\square$ ($\psi_i(\tilde{\boldsymbol{\xi}}_j)$) and $\Gamma_\square$ ($\pi_i(\tilde{\boldsymbol{r}}_j)$). To create the quadrature rule, use the functions `create_quad_onedim_gaussleg` (creates quadrature rule in 1D), `create_quad_hcube_from_onedim` (creates quadrature rule for hypercube from 1D quadrature rule), and `create_quad_simp_from_hcube` (creates quadrature rule for simplex from quadrature rule for hypercube) provided in the project code distribution. To evaluate the basis functions, use the function `eval_interp_simp_lagrange` written in Part 1.2. Your function should have the following signature:

```
function [geom] = create_geom_simp(ndim, porder, nquad_per_dim)
%CREATE_GEOM_SIMP Create structure defining the simplex element.
%
%Input arguments
%----------------
%   NDIM, PORDER : See notation.m
%
%   NQUAD_PER_DIM : number : Number of quadrature nodes per dimension
%
%Output arguments
%-----------------
%   GEOM : See notation.m
```

You may find the function `create_geom_hcube` helpful; it defines the corresponding MATLAB structure defining a hypercube element. Notice that this function uses `eval_interp_hcube_from_onedim` from Homework 3 to define the basis functions.

- Check the quadrature formulas you were provided by computing the following moments of $\Omega_\square$ and $\Gamma_\square$ using quadrature and comparing to known formulas for the area and centroid of simplex and hypercube domains

$$V^{\Omega_\square} = \int_{\Omega_\square} d\boldsymbol{\xi}, \qquad c_i^{\Omega_\square} = \frac{1}{V^{\Omega_\square}} \int_{\Omega_\square} \xi_i \, d\boldsymbol{\xi}, \qquad V^{\Gamma_\square} = \int_{\Gamma_\square} d\boldsymbol{r}, \qquad c_i^{\Gamma_\square} = \frac{1}{V^{\Gamma_\square}} \int_{\Gamma_\square} r_i \, d\boldsymbol{r}, \qquad .$$

  Make sure your quadrature rule has a sufficient number of points to compute the integrals exactly. Only consider the $p = 1$ elements since these quantities do not depend on the degree of completeness of the element. Consider spatial dimensions $d = 1, 2, 3$ (only need to consider $d = 1, 2$ for full credit, $d = 3$ is extra credit) and both simplex and hypercube elements.

Part 1.3 Next we will use the isoparametric concept to define basis functions of elements in the physical domain. Consider an arbitrary d-dimensional simplicial domain, $\Omega_e \subset \mathbb{R}^d$ (physical element), and a d-dimensional regular simplex, $\Omega_\square \subset \mathbb{R}^d$ (referece or parent element) (Figure 2). Define a bijective mapping between the reference and physical domains as

$$\begin{aligned} \mathcal{G}_e : \ \Omega_\square &\to \Omega_e \\ \boldsymbol{\xi} &\mapsto \boldsymbol{x} = \mathcal{G}_e(\boldsymbol{\xi}), \end{aligned}$$

and let $\mathcal{G}_e^{-1} : \Omega_e \to \Omega_\square$ denote the inverse mapping, i.e. $\boldsymbol{\xi} = \mathcal{G}_e^{-1}(\boldsymbol{x})$. In addition, it will prove convenient to introduce the regular $(d-1)$-dimensional simplex, $\Gamma_\square \subset \mathbb{R}^{d-1}$, that will be used as the reference domain for each face of the reference element, $\Omega_\square$. Let

$$\begin{aligned} \mathcal{H}_f : \ \Gamma_\square &\to \partial\Omega_{\square f} \\ \boldsymbol{r} &\mapsto \boldsymbol{\xi} = \mathcal{H}_f(\boldsymbol{r}) \end{aligned}$$

be the bijection that maps $\Gamma_\square$ to the fth face of the d-dimensional reference element $\partial\Omega_{\square f}$ and $\mathcal{H}_f^{-1} : \partial\Omega_{\square f} \to \Gamma_\square$ its inverse, i.e., $\boldsymbol{r} = \mathcal{H}_f^{-1}(\boldsymbol{\xi})$. Finally, we can define a mapping between the $(d-1)$-dimensional reference simplex and fth face of the physical element, $\partial\Omega_{ef}$, by composing these mappings

$$\begin{aligned} \mathcal{F}_{ef} : \ \Gamma_\square &\to \Omega_e \\ \boldsymbol{r} &\mapsto \boldsymbol{x} = \mathcal{G}_e(\mathcal{H}_f(\boldsymbol{r})). \end{aligned}$$

The corresponding inverse mapping is $\mathcal{F}_{ef}^{-1} : \Gamma_\square \to \partial\Omega_{ef}$, i.e., $\boldsymbol{r} = \mathcal{F}_f^{-1}(\mathcal{G}_e^{-1}(\boldsymbol{x}))$.

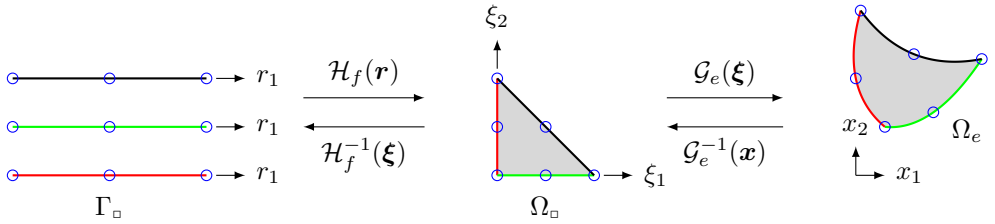

Figure 2: Mapping from $(d-1)$-dimensional reference simplex element ($\Gamma_\square$) to each face of d-dimensional reference simplex element ($\partial\Omega_{\square f}$) ($\boldsymbol{\xi} = \mathcal{H}_f(\boldsymbol{r})$) and the mapping from the d-dimensional reference simplex element ($\Omega_\square$) to the physical element (Ω_e) ($\boldsymbol{x} = \mathcal{G}_e(\boldsymbol{\xi})$).

From this construction, we can define volume and boundary integrals over the physical domain in terms of integrals over the corresponding reference domain using a change of coordinates (volume) and surface parametrization (boundary). Consider the integrals

$$I_v = \int_{\Omega_e} \theta \, dv, \qquad I_s = \int_{\partial\Omega_{ef}} \vartheta \, ds,$$

where $\theta : \Omega_e \to \mathbb{R}$ and $\vartheta : \partial\Omega_{ef} \to \mathbb{R}$. Using the mapping to $\Omega_\square$ for the volume integral and $\Gamma_\square$ for the boundary integral, these integral become

$$\begin{aligned} I_v &= \int_{\Omega_e} \theta \, dv &&= \int_{\Omega_\square} \theta(\mathcal{G}_e(\boldsymbol{\xi})) g_e(\boldsymbol{\xi}) \, d\boldsymbol{\xi} \\ I_s &= \int_{\partial\Omega_{ef}} \vartheta \, ds &&= \int_{\Gamma_\square} \vartheta(\mathcal{F}_{ef}(\boldsymbol{r})) \sigma_{ef}(\boldsymbol{r}) \, d\boldsymbol{r}, \end{aligned} \tag{4}$$

where

$$\begin{aligned} \boldsymbol{G}_e(\boldsymbol{\xi}) &= \frac{\partial \mathcal{G}_e}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}), &\qquad g_e(\boldsymbol{\xi}) &= \det(\boldsymbol{G}_e(\boldsymbol{\xi})) \\ \boldsymbol{F}_{ef}(\boldsymbol{r}) &= \frac{\partial \mathcal{F}_{ef}}{\partial \boldsymbol{r}}(\boldsymbol{r}), &\qquad \sigma_{ef}(\boldsymbol{r}) &= \sqrt{\det\left(\boldsymbol{F}_{ef}(\boldsymbol{r})^T \boldsymbol{F}_{ef}(\boldsymbol{r})\right)}. \end{aligned}$$

Let $\{(w_i, \tilde{\boldsymbol{\xi}}_i)\}_{i=1}^{n_q}$ denote a quadrature rule over $\Omega_\square$, where n_q is the number of quadrature points, $\{w_i\}_{i=1}^{n_q}$ are the quadrature weights, and $\{\tilde{\boldsymbol{\xi}}_i\}_{i=1}^{n_v}$ are the quadrature nodes. Similarly, let $\{(w_i^f, \tilde{\boldsymbol{r}}_i)\}_{i=1}^{n_q^f}$ denote a quadrature rule over $\Gamma_\square$, where n_q^f is the number of quadrature points, $\{w_i^f\}_{i=1}^{n_q^f}$ are the quadrature weights, and $\{\tilde{\boldsymbol{r}}_i\}_{i=1}^{n_v^f}$ are the quadrature nodes. Then, integrals over the reference domains are approximated as

$$\int_{\Omega_\square} \gamma(\boldsymbol{\xi}) \, d\boldsymbol{\xi} \approx \sum_{k=1}^{n_q} w_k \gamma(\tilde{\boldsymbol{\xi}}_k), \qquad \int_{\Gamma_\square} \lambda(\boldsymbol{r}) \, d\boldsymbol{r} \approx \sum_{k=1}^{n_q^f} w_k^f \lambda(\tilde{\boldsymbol{r}}_k).$$

Therefore the integrals in (4) over the physical domains are approximated as

$$I_v \approx \sum_{k=1}^{n_q} \theta(\mathcal{G}_e(\tilde{\boldsymbol{\xi}}_k)) g_e(\tilde{\boldsymbol{\xi}}_k), \qquad I_s \approx \sum_{k=1}^{n_q^f} \vartheta(\mathcal{F}_{ef}(\tilde{\boldsymbol{r}}_k)) \sigma_{ef}(\tilde{\boldsymbol{r}}_k).$$

Next, we define basis functions over the physical element. Let $\{\psi_i\}_{i=1}^{n_v}$ define a basis over the d-dimensional reference element, where $\psi_i : \Omega_\square \to \mathbb{R}$ and n_v is the number of nodes in the element $\Omega_\square$. Furthermore, suppose the basis is Lagrangian with nodes $\{\hat{\boldsymbol{\xi}}_i\}_{i=1}^{n_v}$, i.e., $\psi_i(\hat{\boldsymbol{\xi}}_j) = \delta_{ij}$. Also, let $\{\pi_i\}_{i=1}^{n_v^f}$ define a basis over the $(d-1)$-dimensional reference element, where $\pi_i : \Gamma_\square \to \mathbb{R}$ and n_v^f is the number of nodes in the element $\Gamma_\square$. We also require this basis is Lagrangian with nodes $\{\hat{\boldsymbol{r}}_i\}_{i=1}^{n_v}$, i.e., $\pi_i(\hat{\boldsymbol{r}}_j) = \delta_{ij}$. With these definitions, we define the basis functions over Ω_e as $\{\phi_i^e\}_{i=1}^{n_v}$ and over $\partial\Omega_{ef}$ as $\{\varphi_e^{ef}\}_{i=1}^{n_v^f}$, where $\phi_i^e : \Omega_e \to \mathbb{R}$ and $\varphi_i^{ef} : \partial\Omega_{ef} \to \mathbb{R}$ are defined as

$$\phi_i^e(\boldsymbol{x}) = \psi_i(\mathcal{G}_e^{-1}(\boldsymbol{x})), \qquad \varphi_i^{ef}(\boldsymbol{x}) = \pi_i(\mathcal{F}_{ef}^{-1}(\boldsymbol{x})). \tag{5}$$

From these definitions and the integration formulas in (4), any integrals involving the bases ϕ_i^e or φ_i^{ef} can be written solely in terms of the corresponding reference domain basis $\psi_i(\boldsymbol{\xi})$ and $\pi_i(\boldsymbol{r})$ because the composition of a mapping and its inverse is the identity map. This implies that we only need to evaluate the basis functions associated with the reference domains $\Omega_\square$ and $\Gamma_\square$ at the quadrature nodes associated with those domains. From a simple application of the chain rule of differentiation (first equality) and the inverse function theorem (second equality), the gradient of the physical basis over Ω_e is

$$\frac{\partial \phi_i^e}{\partial x_j}(\boldsymbol{x}) = \sum_{k=1}^{d} \frac{\partial \psi_i}{\partial \xi_k}(\mathcal{G}_e^{-1}(\boldsymbol{x})) \frac{\partial (\mathcal{G}_e^{-1})_k}{\partial x_j}(\boldsymbol{x}) = \sum_{k=1}^{d} \frac{\partial \psi_i}{\partial \xi_k}(\mathcal{G}_e^{-1}(\boldsymbol{x})) \left[\boldsymbol{G}_e^{-1}\right]_{kj} \tag{6}$$

Finally, we define the mappings $\mathcal{G}_e$ and $\mathcal{F}_{ef}$ using the basis functions associated with $\Omega_\square$ and $\Gamma_\square$ (isoparametric)

$$\mathcal{G}_e(\boldsymbol{\xi}) = \sum_{i=1}^{n_v} \hat{\boldsymbol{x}}_i^e \psi_i(\boldsymbol{\xi}), \qquad \mathcal{H}_f(\boldsymbol{r}) = \sum_{i=1}^{n_v^f} \hat{\boldsymbol{\xi}}_i^f \pi_i(\boldsymbol{r}), \qquad \mathcal{F}_{ef}(\boldsymbol{r}) = \sum_{i=1}^{n_v^f} \hat{\boldsymbol{x}}_i^{ef} \pi_i(\boldsymbol{r}), \tag{7}$$

where $\{\hat{\boldsymbol{x}}_i^e\}_{i=1}^{n_v}$ are the nodes associated with the element Ω_e, $\{\hat{\boldsymbol{x}}_i^{ef}\}_{i=1}^{n_v^f}$ are the nodes associated with its face $\partial\Omega_{ef}$, and $\{\hat{\boldsymbol{\xi}}_i^f\}_{i=1}^{n_v^f}$ are the nodes associated with face f of $\Omega_\square$, $\partial\Omega_{\square f}$.

Tasks for Part 1.3
Your task is to write three functions: one that evaluates all relevant isoparametric quantities, one that completely defines a simplex element in the reference domain, and one that completely defines a hypercube element in the reference domain.

- Derive the quantities $\boldsymbol{G}_e(\boldsymbol{\xi})$, $\boldsymbol{F}_{ef}(\boldsymbol{r})$, for the isoparametric mapping given in (7). Your answer should be in terms of the element coordinates $\{\hat{\boldsymbol{x}}_i^e\}_{i=1}^{n_v}$, $\{\hat{\boldsymbol{x}}_i^{ef}\}_{i=1}^{n_v^f}$ and the basis functions $\{\psi_i\}_{i=1}^{n_v}$, $\{\pi_i\}_{i=1}^{n_v^f}$.
- Using the node numbering in Figure 1, write out the isoparametric boundary mapping $\mathcal{H}_f(\boldsymbol{r})$ for each face for the case $d = 2$, $p = 2$. Your answer should be in terms of the nodes of $\Omega_\square$, $\{\boldsymbol{\xi}_i\}_{i=1}^{6}$, and the basis functions associated with $\Gamma_\square$: $\{\pi_i(\boldsymbol{r})\}_{i=1}^{3}$, i.e., the basis functions for a face of the element in their reference domain.
- Implement a function that evaluates all relevant isoparametric quantities. Your function should have the following signature:

```
function [xq, detG, Gi, xqf, sigf] = eval_isoparam_quant(xe, Q, dQdz, Qf, dQfdr, f2v)
%EVAL_ISOPARAM_QUANT Evaluate isoparametric quantities for a single element
%given the nodal coordinates of the element in physical space (XE) and the
%basis functions (and their derivatives) over the element and its faces.
%
%Input arguments
%_______________
%  XE, Q, DQDZ, QF, DQFDR, F2V : See notation.m
%
%Output arguments
%________________
%  XQ, DETG, GI, XQF, SIGF : See notation.m
```

  Check your answer by considering a simple mapping, deriving all isoparametric quantities analytically, and comparing the values returned by your function to their analytical expressions.
- Use these quantities to compute the volume, centroid, and surface area

$$V(\Omega_e) = \int_{\Omega_e} dv, \qquad \boldsymbol{c}(\Omega_e) = \frac{1}{V(\Omega_e)} \int_{\Omega_e} \boldsymbol{x}\, dv, \qquad S(\Omega_e) = \int_{\partial\Omega_e} ds,$$

  of the curved triangle in Figure 2). The nodes of the triangle (in the order given by Figure 1) are $\{(0.0, 0.0), (0.5, 0.15), (1.0, 0.7), (-0.3, 0.5), (0.3, 0.75), (-0.25, 1.2)\}$.

Part 1.4 In this section, we will use a finite element mesh (Figure 3), a collection of finite elements such as the simplex and hypercube elements defined in Part 1.3, to evaluate integrals over complex domains $\Omega \subset \mathbb{R}^d$:

$$I_v = \int_\Omega \theta\, dv, \qquad I_s = \int_{\partial\Omega} \vartheta\, ds. \tag{8}$$

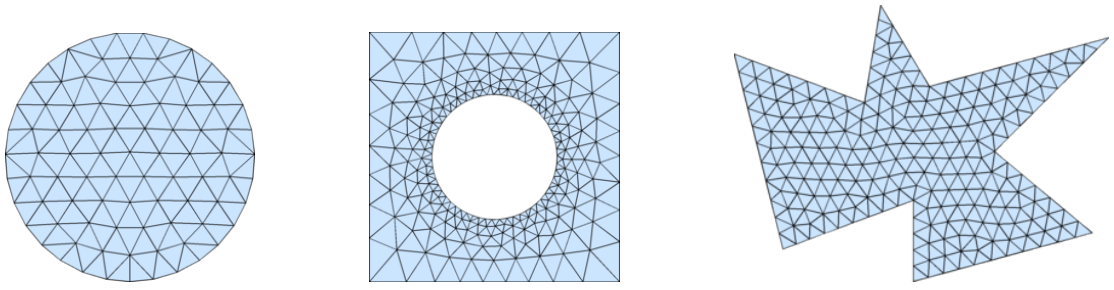

Figure 3: Mesh of a circle, square with circular hole, and arbitrary polygon using triangular elements.

We will describe our unstructured mesh using the arrays: `xcg`, `e2vcg`, and `e2bnd`, defined as

```
%   XCG : 2D array (ndim, nnode) : The position of the nodes in the mesh.
%     The (i, j)—entry is the position of global node j in the ith dimension.
%     The global node numbers are defined by the columns of this matrix, e.g.,
%     the node at xcg(:, j) is the jth node of the mesh.
%
%   E2VCG : 2D array (nnode_per_elem, nelem): The connectivity of the
%     mesh. The (:, e)—entries are the global node numbers of the nodes
%     that comprise element e. The local node numbers of each element are
%     defined by the columns of this matrix, e.g., e2vcg(i, e) is the
%     global node number of the ith local node of element e.
%
%   E2BND: 2D array (nface_per_elem, nelem): The mapping between element
%     boundaries and global boundaries. The (f, e)—entry is the global
%     boundary tag on which the fth face of element e lies. If face f of
%     element e does not touch the global boundary, e2bnd(f, e) is NaN.
```

With this concept of a mesh, the integrals in (8) can be re-written as

$$I_v = \sum_{e=1}^{n_e} \int_{\Omega_e} \theta \, dv, \qquad I_s = \sum_{e=1}^{n_e} \sum_{f=1}^{n_f} \int_{\partial\Omega_{ef} \cap \partial\Omega} \vartheta \, ds = \sum_{e=1}^{n_e} \sum_{f=1}^{n_f} \mathbf{1}_{\{\partial\Omega_{ef} \cap \partial\Omega \neq \varnothing\}} \int_{\partial\Omega_{ef}} \vartheta \, ds,$$

where n_e is the number of elements in the mesh and $\mathbf{1}_{\{\partial\Omega_{ef} \cap \partial\Omega \neq \varnothing\}}$ is the indicator function that takes the value of 0 if $\partial\Omega_{ef} \cap \partial\Omega = \varnothing$ and 1 otherwise. Notice that this will, in general, only be an approximation to an integral since the region covered by the union of all finite elements will not exactly overlap with Ω, except in special cases.

Tasks for Part 1.4

With the isoparametric function and element structures defined in Part 1.3, you can create a structure array `mesh_data` that contains the isoparametric quantities for every element in a mesh defined by `xcg`, `e2vcg`, `e2bnd` using the function `create_mesh_data` provided. This will create all relevant data for each element of your mesh. Using this structure array, we are interested computing the following integrals

$$V(\Omega) = \int_{\Omega} dv, \qquad \boldsymbol{c}(\Omega) = \frac{1}{V(\Omega)} \int_{\Omega} \boldsymbol{x} \, dv, \qquad S(\Omega) = \int_{\partial\Omega} ds,$$

i.e., the volume, centroid, and surface area of Ω.

- Write a function that computes the volume, centroid, and surface area of a domain described by a mesh. Your function should have the following signature:

```
function [V, C, S] = compute_volume_centroid_surfarea(geom, mesh_data)
%COMPUTE_VOLUME_CENTROID_SURFAREA Compute the volume, centroid, and surface
%area of a domain (approximated) by the mesh described by GEOM, MESH_DATA.
%
%Input arguments
%———————————————
%  GEOM, MESH_DATA : See notation.m
%
%Output arguments
%————————————————
%  V : number : Volume of domain
%
%  C : Array (NDIM,) : Centroid of domain
%
%  S : number : Surface area of domain
```

- Consider a domain $\Omega = [0, 1]^d$ for $d = 1, 2, 3$. Create a mesh of this domain with 5^d $p = 2$ hypercube elements using `create_unif_mesh_hcube` and create the corresponding `mesh_data` structure array. Compute the volume, centroid, and surface area of Ω by integrating the appropriate quantities over

the mesh and compare to the known volume, centroid, and surface area of a hypercube. For the case of $d = 2$, use the function `split_quad_mesh_into_tri_mesh` to split quadrilateral elements into triangular ones and repeat the integral calculations. This will serve as a test for parts of your code (quadrature, isoparametric mapping of integrals).

- Compute the volume, centroid, and surface area of the bat signal and ND logo (Figure 4). Use $p = 1$ simplex elements. The mesh for the bat signal and ND logo are provided in `_mesh/_meshes/batman-simp-p1.mat` and `_mesh/_meshes/nd-simp-p1.mat`. Use `visualize_fem.m` to plot the mesh. Plot the centroid of the domain as a sanity check for the centroid computation.

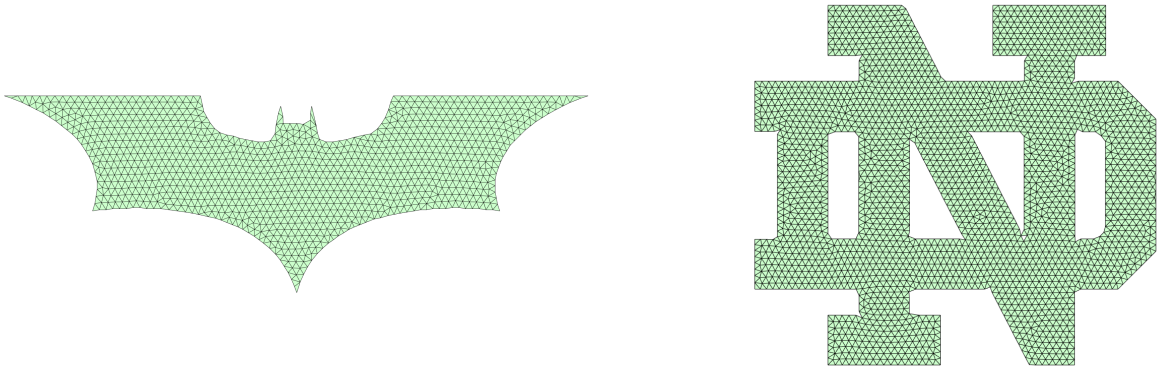

Figure 4: Simplicial mesh ($d = 2$) of the batman symbol and Notre Dame logo.

- What is the volume, centroid, and surface area of the cow (`_mesh/_meshes/cow-simp-p1.mat`), dragon (`_mesh/_meshes/dragon-simp-p1.mat`), and sculpture (`_mesh/_meshes/sculpt10kv-simp-p1.mat`) (Figure 5)? Use $p = 1$ simplex elements and `visualize_fem.m` to plot the mesh. Plot the centroid of the domain for the sculpture mesh as a sanity check for the centroid computation.

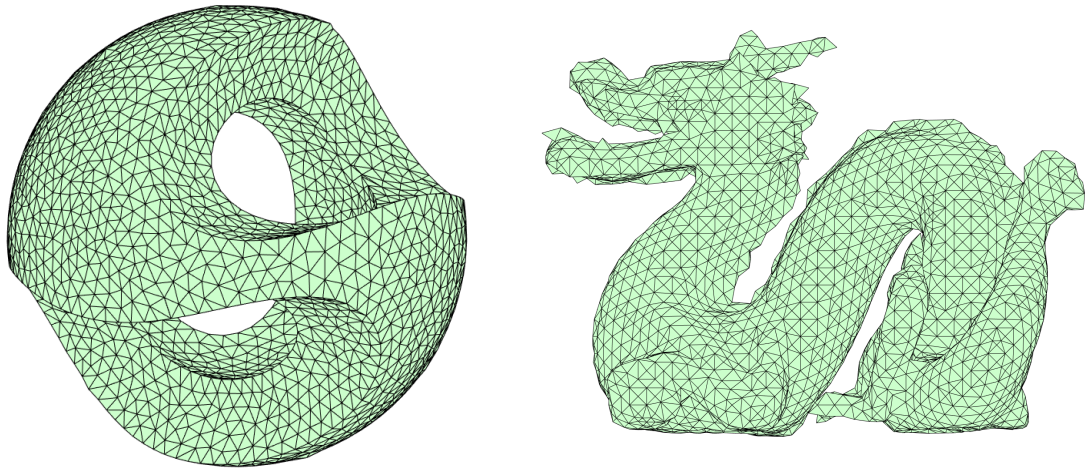

Figure 5: Simplicial mesh ($d = 3$) of a sculpture and dragon.