ISOGEOMETRIC METHODS: HOMEWORK # 3

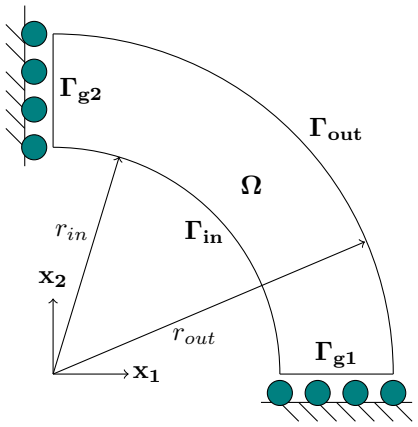

FIGURE 1. Problem Setup

Let $u(x_1, x_2) = (u_1(x_1, x_2),\, u_2(x_1, x_2))^{\mathrm{T}}$ represent the deformation at a point $(x_1, x_2) \in \Omega$. The strain-rate tensor is given by

$$\varepsilon_{ij}(u) = \frac{1}{2}\big(u_{i,j} + u_{j,i}\big)$$

We assume the domain, Ω, is subject to a stress tensor given by

$$\begin{pmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{12}\end{pmatrix} = \underbrace{\frac{E}{1-\nu^2}\begin{pmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{pmatrix}}_{\boldsymbol{D}}\begin{pmatrix}\varepsilon_{11}(u)\\ \varepsilon_{22}(u)\\ 2\varepsilon_{12}(u)\end{pmatrix} := \boldsymbol{D}\,\boldsymbol{\varepsilon}(u)$$

where E is the Young's modulus and ν is the Poisson ratio. The deformation of Ω is determined by the following equations

$$\begin{aligned}
\sigma_{ij,j} &= 0, && \Omega\\
\sigma_{ij}\,\hat{n}_j &= -p\,\hat{n}_i, && \Gamma_{in}\\
\sigma_{ij}\,\hat{n}_j &= 0, && \Gamma_{out}\\
u_1 &= 0, && \Gamma_{g2}\\
u_2 &= 0, && \Gamma_{g1}\\
\sigma_{12} &= 0, && \Gamma_{g1}\cup\Gamma_{g2}
\end{aligned} \tag{1}$$

The exact solution to this problem is radially symmetric. In cylindrical co-ordinates it is given by

$$\begin{aligned}
u_\theta &= 0\\
u_r(r) &= \frac{p}{E}\left(\frac{r_{in}^2 r_{out}^2}{r_{out}^2 - r_{in}^2}\right)\left(\frac{1+\nu}{r} + \frac{1-\nu}{r_{out}^2}\,r\right)
\end{aligned} \tag{2}$$

For this problem we will use $E = p = 1$MPa, $\nu = 0.3$, $r_{in} = 1$mm, $r_{out} = 2$mm

Part 1: Geometry Construction

1. Construct the domain Ω exactly, using quadratic NURBS on the parameter space $[0,1]\times[0,1]$.
2. Insert enough knots to get nine uniform Bézier elements in parameter space (Figure 2)

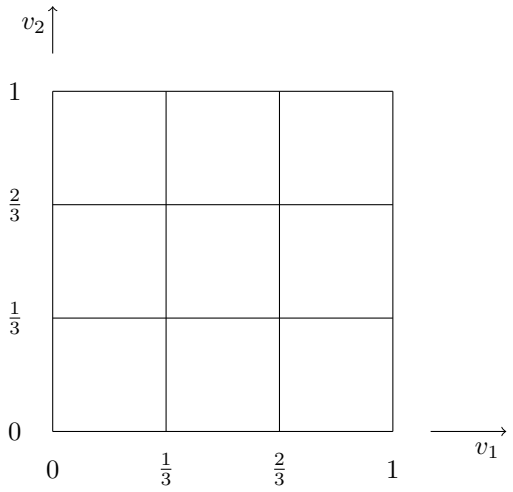

Figure 2. Parameter Space

3. Construct an IEN array for the discretization
4. Derive the element extraction operators $\boldsymbol{C}^e$ for the basis using the code you developed for Homework # 2.

Part 2: Isogeometric Analysis

The weak form of the problem will be

$$\int_\Omega \boldsymbol{\varepsilon}(w)^T\, \boldsymbol{D}\, \boldsymbol{\varepsilon}(u)\, d\Omega = -\int_{\Gamma_{in}} p\, w_i \hat{n}_i\, d\Gamma$$

and the Galerkin form is obtained by replacing w and u with their discrete counterparts w^h and u^h. The next order of business is to convert the Galerkin form into its matrix form $\boldsymbol{K}\,\boldsymbol{d} = \boldsymbol{F}$.

5. Construct an ID array for the problem

If $\aleph$ refers to the set of global control point numbers and $\aleph_{g1}$, $\aleph_{g2}$, $\aleph_{in}$ refer to the set of control point numbers corresponding to the edges Γ_{g1}, Γ_{g2} and Γ_{in} respectively, then the discretizations of our test and trial functions will be

$$w_1^h = R_A, \quad \text{where } A \in \aleph \setminus \aleph_{g2} \qquad\qquad w_2^h = R_C \quad \text{where } C \in \aleph \setminus \aleph_{g1}$$
$$u_1^h = \sum_{B\in\aleph\setminus\aleph_{g2}} d_1^B R_B \qquad\qquad u_2^h = \sum_{E\in\aleph\setminus\aleph_{g1}} d_2^E R_E$$

where R_A is our NURBS basis.

Using the ID array, we can define the column vector $\boldsymbol{d}$ whose P-th component will be d_i^A where $P = ID(A,\, i)$. Likewise the Q-th component of the force vector $\boldsymbol{F}$ will be $-\int_{\Gamma_{in}} p\, R_B \hat{n}_j\, d\Gamma$ where $Q = ID(B,\, j)$.

We are now left with the stiffness matrix $\boldsymbol{K}$. With $\boldsymbol{e}_1 = \begin{pmatrix}1\\0\end{pmatrix}$ and $\boldsymbol{e}_2 = \begin{pmatrix}0\\1\end{pmatrix}$, we can write $w^h = R_A \boldsymbol{e}_i$ and $u^h = \sum_A d_i^A R_A \boldsymbol{e}_i$ where $A \in \aleph \setminus \aleph_{g2}$ for $i = 1$ and $A \in \aleph \setminus \aleph_{g1}$ for $i = 2$.[1] It is then relatively easy to check that $\boldsymbol{\varepsilon}(R_A \boldsymbol{e_i}) = \boldsymbol{B}_A \boldsymbol{e}_i$, where

$$\boldsymbol{B}_A = \begin{bmatrix} R_{A,1} & 0 \\ 0 & R_{A,2} \\ R_{A,2} & R_{A,1} \end{bmatrix}$$

Thus the components of the stiffness matrix will be

$$\boldsymbol{K}_{QP} = \int_\Omega \boldsymbol{e}_j^T \, \boldsymbol{B}_B^T \, \boldsymbol{D} \, \boldsymbol{B}_A \, \boldsymbol{e}_i \, d\Omega$$

where $Q = ID(B,\, j)$ and $P = ID(A,\, i)$.

In order to calculate the components of the element stiffness matrix $\boldsymbol{K}^e$ and the element force vector $\boldsymbol{F}^e$, we simply replace Ω and Γ_{in} with the element Ω^e and Γ_{in}^e as the domain of integration.

For the actual computation of these element components, we will be performing numerical quadrature on the $[-1,\, 1] \times [-1,\, 1]$ parent element using Bernstein polynomials. Our next order of business is to make this possible.

6. Write a function that evaluates the i-th degree p Bernstein basis function and its derivative at a specified point ξ in $[-1,\, 1]$.[2]

$$\left[B, \frac{dB}{d\xi}\right] = Bernstein\_1D(\xi, i, p)$$

7. Write a function that evaluates the (i_1, i_2)-th degree (p_1, p_2) Bernstein basis function and its gradient at a specified point (ξ_1, ξ_2) in $[-1, 1] \times [-1,\, 1]$.

$$\left[B, \frac{dB}{d\boldsymbol{\xi}}\right] = Bernstein\_2D(\xi_1, \xi_2, i_1, i_2, p_1, p_2)$$ [3]

Let $\tilde{\boldsymbol{B}}^e(\boldsymbol{v})$ represent the vector of Bernstein polynomials defined on a given Bézier element $\hat{\Omega}^e$, let $\boldsymbol{B}^+(\boldsymbol{\xi})$ represent the vector of Bernstein polynomials defined on $[-1,\, 1] \times [-1,\, 1]$ and let $\boldsymbol{R}^e$ represent the vector of NURBS basis functions localized to the Bézier element. If we follow the convention used in the Borden paper, our Bezier extraction takes the form

$$\boldsymbol{R}^e(\boldsymbol{v}) = \boldsymbol{W}^e \boldsymbol{C}^e \, \frac{\tilde{\boldsymbol{B}}^e(\boldsymbol{v})}{W^b(\boldsymbol{v})}$$

where $\boldsymbol{W}^e = \text{diag}[\boldsymbol{w}^e]$ is the diagonal matrix of weights for the NURBS with support in the element and $W^b(\boldsymbol{v}) = (\boldsymbol{w}^e)^T \boldsymbol{C}^e \tilde{\boldsymbol{B}}^e(\boldsymbol{v})$.

In order to use the Bernstein polynomials $\boldsymbol{B}^+(\boldsymbol{\xi})$ created in Problem **7.**, we need to pull back from the Bézier element to the parent element $[-1,\, 1] \times [-1,\, 1]$. Let $\phi^e : [-1,\, 1] \times [-1,\, 1] \to \hat{\Omega}^e$ be the affine map from the parent element to the Bézier element, then

$$\tilde{\boldsymbol{B}}^e(\phi^e(\boldsymbol{\xi})) = \boldsymbol{B}^+(\boldsymbol{\xi}) \quad \Longrightarrow \quad \frac{d\tilde{\boldsymbol{B}}^e}{d\boldsymbol{v}}(\phi(\boldsymbol{\xi})) = \frac{d\boldsymbol{B}^+}{d\boldsymbol{\xi}}(\boldsymbol{\xi}) \left(\frac{d\phi^e}{d\boldsymbol{\xi}}(\boldsymbol{\xi})\right)^{-1}$$

8. For a given Bézier element $[a_1,\ b_1] \times [a_2,\, b_2]$ compute $\phi^e(\boldsymbol{\xi})$ and $\frac{d\phi^e}{d\boldsymbol{\xi}}(\boldsymbol{\xi})$.

---

[1] The ID array encodes this.

[2] Recall the formula for defining the derivative of a Bernstein basis function in terms of lower degree Bernstein basis functions

[3] Note that you will eventually have to convert (i_1, i_2) into your local control point number a.

9. Write a function that evaluates $\frac{d\boldsymbol{R}^e}{d\boldsymbol{v}}(\phi^e(\boldsymbol{\xi}))$ in terms of $\boldsymbol{C}^e$, $\boldsymbol{B}^+(\boldsymbol{\xi})$ and $\frac{d\boldsymbol{B}^+}{d\boldsymbol{\xi}}(\boldsymbol{\xi})$ at a point $\boldsymbol{\xi} \in [-1, 1] \times [-1, 1]$.

10. Using

$$\boldsymbol{x}(\phi^e(\boldsymbol{\xi})) = \sum_{a \in \aleph^e} \boldsymbol{P}_a \, R_a(\phi^e(\boldsymbol{\xi}))$$

where $\aleph^e$ denotes the set of local control point numbers for the element, you can now write an element shape function routine that computes the matrix $(\frac{d\boldsymbol{x}}{d\boldsymbol{\xi}}(\phi(\boldsymbol{\xi})))^{-1}$ and the Jacobian $J(\boldsymbol{\xi}) = \det\left(\frac{d\boldsymbol{x}}{d\boldsymbol{\xi}}(\phi(\boldsymbol{\xi}))\right)$ for the point $\boldsymbol{\xi}$ in the parent element.

For some functions $g(\boldsymbol{x})$ and $f(\boldsymbol{x})$ defined on the physical domain Ω^e, integration by quadrature on the parent element will take the form

$$\begin{aligned}\int_{\Omega^e} f(\boldsymbol{x}) \left[\frac{dg}{d\boldsymbol{x}}(\boldsymbol{x})\right] d\Omega &= \int_{-1}^{1}\int_{-1}^{1} f(\boldsymbol{x}(\phi^e(\boldsymbol{\xi}))) \left[\frac{dg}{d\boldsymbol{\xi}}(\boldsymbol{x}(\phi^e(\boldsymbol{\xi}))) \left(\frac{d\boldsymbol{x}}{d\boldsymbol{\xi}}(\phi^e(\boldsymbol{\xi}))\right)^{-1}\right] J(\boldsymbol{\xi})\, d\boldsymbol{\xi} \\ &= \sum_{l=1}^{n_{int}} f(\boldsymbol{x}(\phi^e(\boldsymbol{\xi}_l))) \left[\frac{dg}{d\boldsymbol{\xi}}(\boldsymbol{x}(\phi^e(\boldsymbol{\xi}_l))) \left(\frac{d\boldsymbol{x}}{d\boldsymbol{\xi}}(\phi^e(\boldsymbol{\xi}_l))\right)^{-1}\right] J(\boldsymbol{\xi}_l)\, W_l\end{aligned}$$

where $\boldsymbol{\xi}_l$ and W_l are the quadrature points and weights respectively.

11. Using uniform 2 point Gaussian quadrature for 2D, assemble the stiffness matrix and the force vector and solve for $\boldsymbol{d}$ (The quadrature points and weights for 1D quadrature can be found here[4]). Compare the resulting solution u^h, to the exact solution given in (2).

12. Repeat (11) with 3 point Gauss quadrature.

13. Compare the two solutions obtained in (**11**.) and (**12**.).

BONUS PROBLEM. Create a sequence of refined meshes using knot insertion, and compute the L2 error between the numerical solution and exact solution for each mesh. Plot the L2 error vs. the element size on a log-log scale.

[4]https://en.wikipedia.org/wiki/Gauss-Legendre_quadrature#Definition