WI4205 - 2022/23
Applied Finite Elements
Assignment 1.4
Deadline - 23:59, June 20, 2023

Instructions and assessment criteria to keep in mind:

- A submission for Assignment 1.4 is **required for passing** WI4205.
- This is a **group assignment**. Please work in groups of two and submit a joint report.
- The reports need to be **typed in LaTeX or Word**.
- The **deadline** for uploading your solutions to Brightspace is 23:59, June 20, 2023.
- Grading of late submissions will be done as per the **grading protocol posted on Brightspace**; make sure you are familiar with it.
- Provide **clear and motivated answers** to the questions. No/reduced points will be awarded if your solutions are unaccompanied by explanations.

An Implementation of the 2D Finite Element Method (20 points)

In this assignment, you will implement the finite element method for a two-dimensional problem into a computer code. As you will discover, and as explained in class, the overall code structure from the 1D implementation will carry over to this setting, the only difference will be in certain small details. The following important information should be kept in mind:

- You are free to use your programming language of choice.
- When necessary, in all of the following, the size of input/output matrices in the Python scripts is denoted in blue as $[a, b]$.
- **The equations in this document follow the numbering convention from our lectures – i.e., they use one-based indexing for all variables. For Python you will need to shift the indices by one.**

Setup and notation

In this section we set up the weak problem that you will solve in this assignment, and then summarize the finite element discussion from class for your convenience.

Strong and weak problem

In this assignment, you will be asked to solve a Poisson problem which can be stated as follows in strong form: find u such that

$$\begin{aligned} -\nabla \cdot \kappa \nabla u &= f\ , \quad \text{on } \Omega\ , \\ u|_{\Gamma_D} &= 0\ , \\ \nabla u \cdot \mathbf{n}|_{\Gamma_N} &= g\ , \end{aligned} \tag{1}$$

where $\Gamma_D \cap \Gamma_N = \emptyset$ and $\overline{\Gamma_D \cup \Gamma_N} = \partial\Omega$, and $\mathbf{n}$ is the outward facing unit normal to $\partial\Omega$.

A weak form for this problem can be stated as[1]: find $u \in \mathcal{S}$ such that the following is satisfied for all $w \in \mathcal{W}$,

$$\underbrace{\int_\Omega \nabla w \cdot \kappa \nabla u \, dx^1 dx^2}_{=:B(u,w)} = \underbrace{\int_\Omega wf \, dx^1 dx^2 + \int_{\Gamma_N} gw \, ds}_{=:L(w)} \,, \tag{2}$$

where $\mathcal{S} = \mathcal{W} = \{v \in H^1(\Omega) : v|_{\Gamma_D} = 0\}$. This is the problem that you will solve on different geometries using a Galerkin formulation for appropriate choices of discrete spaces $\mathcal{S}_h \subset \mathcal{S}$ and $\mathcal{W}_h \subset \mathcal{W}$: find $u_h \in \mathcal{S}_h$ such that the following holds for all $w_h \in \mathcal{W}_h$,

$$\underbrace{\int_\Omega \nabla w_h \cdot \kappa \nabla u_h \, dx^1 dx^2}_{=B(u_h,w_h)} = \underbrace{\int_\Omega w_h f \, dx^1 dx^2 + \int_{\Gamma_N} gw_h \, ds}_{=L(w_h)} \,, \tag{3}$$

Reference element and reference polynomial space

Denote with $\tilde{\Omega} = [0,1]^2$ the reference element and, given $\mathbf{p} = (p_1, p_2)$, consider $\mathcal{P}_\mathbf{p}$, the space of polynomial functions of bi-degree at most $\mathbf{p}$ on $\tilde{\Omega}$. A basis for this space is $\tilde{N}_i$, $i = 1, \ldots, (p_1+1)(p_2+1)$, defined as

$$\tilde{N}_i(\xi^1, \xi^2) := \tilde{N}_{j_1}(\xi^1)\tilde{N}_{j_2}(\xi^2) \,,$$

where $i = j_1 + (j_2 - 1)(p_1 + 1)$ and $\tilde{N}_{j_1}(\xi^1)$ and $\tilde{N}_{j_2}(\xi^2)$ are the j_1-th and j_2-th univariate Bernstein–Bézier polynomials evaluated at ξ^1 and ξ^2, respectively.

Maps from reference element to domain

Next, consider a domain $\Omega \subset \mathbb{R}^2$ on which we want to solve a finite element problem. Let Ω_h be a decomposition of Ω into *curvilinear quadrilaterals* Ω_i, $i = 1, \ldots, m$. That is, each Ω_i is bounded by 4 curves which can be straight lines or not; see the lecture notes for an example. Let ϕ_i, $i = 1, \ldots, m$, be bijections that map $\tilde{\Omega}$ to Ω_i,

$$\begin{aligned} \phi_i \; &: \; \tilde{\Omega} \to \Omega_i \,, \\ (\xi^1, \xi^2) &\mapsto (x^1, x^2) := \phi_i(\xi^1, \xi^2) \,. \end{aligned}$$

We will asssume that each component of the map is an element of the space $\mathcal{P}_\mathbf{p}$, i.e.,

$$\phi_i \in \mathcal{P}_\mathbf{p} \times \mathcal{P}_\mathbf{p} \,.$$

In other words, for each Ω_i, there exist points (X^1_{ji}, X^2_{ji}), $j = 1, \ldots, (p_1 + 1)(p_2 + 1)$, such that

$$\phi_i(\xi^1, \xi^2) = \left(\underbrace{\sum_{j=1}^{(p_1+1)(p_2+1)} X^1_{ji} \tilde{N}_j(\xi^1, \xi^2)}_{=:x^1(\xi^1,\xi^2)}, \; \underbrace{\sum_{j=1}^{(p_1+1)(p_2+1)} X^2_{ji} \tilde{N}_j(\xi^1, \xi^2)}_{=:x^2(\xi^1,\xi^2)} \right) \,. \tag{4}$$

With these maps, for $(x^1, x^2) \in \Omega_i$, and the assumed invertibility of ϕ_i, we can define the *pushforwards* of the functions $\tilde{N}_i$ from $\tilde{\Omega}$ to Ω_i through the maps ϕ_i in the following equivalent ways,

$$\tilde{N}_{ji}(x^1, x^2) := \tilde{N}_j \circ \phi_i^{-1}(x^1, x^2) \iff \tilde{N}_{ji} \circ \phi_i(\xi^1, \xi^2) := \tilde{N}_j(\xi^1, \xi^2) \,.$$

[1]Make sure that you see how to arrive at this weak form yourself; you will be asked to do something similar in the exam.

Finite element spaces on the domain

We now define the finite element spaces that will be used for solving problems on Ω. Given $k \geq -1$, we define

$$\mathcal{F}(p,k;\Omega_h) := \left\{ f \in C^k(\overline{\Omega}) \mid f \circ \phi_i \in \mathcal{P}_{\mathbf{p}}\ ,\ i = 1,\dots,m \right\} .$$

Let $n := \dim(\mathcal{F}(p,k;\Omega_h))$ and denote a basis spanning $\mathcal{F}(p,k;\Omega_h)$ as N_i, $i = 1,\dots,n$. Therefore, given any $f \in \mathcal{F}(p,k;\Omega_h)$, there exist coefficients $f_i \in \mathbb{R}$ such that

$$f = \sum_{i=1}^{n} f_i N_i\ .$$

Given Ω_i, let $I_i := \{j\ :\ N_j|_{\Omega_i} \neq 0\}$. By definition of the finite element space and for any Ω_i and $j \in I_i$, there exist coefficients $e_{jki} \in \mathbb{R}$, $k = 1,\dots,(p_1+1)(p_2+1)$, such that

$$N_j\big|_{\Omega_i} := \sum_{k=1}^{(p_1+1)(p_2+1)} e_{jki}\tilde{N}_{ki}\ .$$

It is important to note that in general, and unlike in 1D, I_i and I_j will have different cardinalities for $i \neq j$. We will denote the cardinality of I_i with $\#I_i$.

Reference element: evaluation points for quadrature and plotting

In this 2D setting, we will evaluate the reference basis functions $\tilde{N}_i$ on a grid of points distributed on $\tilde{\Omega}$. These points can be for visualizing the solution, or you might be using them for performing numerical quadrature. Given n_{q1}, we will distribute $n_q := n_{q1}^2$ evaluation points on $\tilde{\Omega}$, and we will denote these points with $(\tilde{\xi}_i^1, \tilde{\xi}_i^2) \in \tilde{\Omega}$, $i = 1,\dots,n_q$. These evaluation points are placed in an $n_{q1} \times n_{q1}$ grid on $\tilde{\Omega}$.

To do this, you **need to implement** a function (by generalizing the 1D function provided to you) that can create the following data structure for you:

```
function create_ref_data(n_q1, p, data_kind)
  ⋮
return ref_data
```

where data_kind is a string that is either equal to 'plot' or 'integrate'. It should return a structure ref_data with the following fields:

```
ref_data
├ p                                  [1 × 2]
├ evaluation_points                  [n_q × 2]
├ quadrature_weights                 [n_q × 1]
├ reference_basis                    [(p_1 + 1)(p_2 + 1) × n_q]
└ reference_basis_derivatives        [(p_1 + 1)(p_2 + 1) × n_q × 2]
```

where the i-th row of evaluation_points contains the i-th evaluation point $(\tilde{\xi}_i^1, \tilde{\xi}_i^2) \in \tilde{\Omega}$; the i-th row of quadrature_weights contains the i-th quadrature weight if data_kind is 'integrate' (else it is empty); the (i,j)-th entry of reference_basis contains the value $\tilde{N}_i(\tilde{\xi}_j^1, \tilde{\xi}_j^2)$; and the (i,j,k)-th entry of reference_basis_derivatives contains the value $\frac{\partial \tilde{N}_i(\tilde{\xi}_j^1, \tilde{\xi}_j^2)}{\partial \xi^k}$.

REMARK: As in the 1D setting, when data_kind is set to 'integrate', the reference basis functions should be computed at the quadrature points for a $(n_{q1} \times n_{q1})$-point Gauss quadrature scheme. When data_kind is set to 'plot', the reference basis functions should be computed on $n_{q1} \times n_{q1}$ points uniformly distributed on the $\tilde{\Omega}$.

Map coefficients and finite element space

As is clear, the definition of the finite element spaces is tied to the specific mesh and the geometry that you are working with. Therefore, in this assignment, for a choice of different geometries, you are given all information necessary for building finite element spaces in different files. Specifically, you are given the following geometry files:

star3
star4
star5
distressed_robotD
distressed_robotDN

For each geometry, you **are given** a file <geometry-name>.mat that can be loaded into Python. It contains three pieces of information: the degree **p**, and two structures called fe_geometry and fe_space. They are explained below.

Map coefficients

The structure fe_geometry contains the following fields:

fe_geometry

├ m	$[1 \times 1]$
└ map_coefficients	$[(p_1+1)(p_2+1) \times 2 \times m]$

where the (i,j,k)-th value of map_coefficients contains X^j_{ik}; recall that these are the coefficients that help define the maps from $\tilde{\Omega}$ to elements of Ω_h. Given these coefficients, you can evaluate the maps as well as the derivatives of the maps using Equation (4).

You **need to implement** a function:

```
function create_geometric_map(fe_geometry, ref_data)
  ⋮
return geom_map
```

which returns a structure geom_map with the following fields:

geom_map

├ map	$[n_q \times 2 \times m]$
├ map_derivatives	$[n_q \times 4 \times m]$
└ imap_derivatives	$[n_q \times 4 \times m]$

where the $(i,:,k)$-th row of map contains the values $\phi_k(\tilde{\xi}^1_i, \tilde{\xi}^2_i)$; the $(i,:,k)$-th row of map_derivatives contains the values

$$\left[\frac{\partial \phi_k(\tilde{\xi}^1_i, \tilde{\xi}^2_i)}{\partial \xi^1}\ ,\ \frac{\partial \phi_k(\tilde{\xi}^1_i, \tilde{\xi}^2_i)}{\partial \xi^2}\right]\ ;$$

and the $(i,:,k)$-th row of map_derivatives contains the values

$$\left[\frac{\partial \phi_k^{-1}(x^1(\tilde{\xi}^1_i, \tilde{\xi}^2_i), x^2(\tilde{\xi}^1_i, \tilde{\xi}^2_i))}{\partial x^1}\ ,\ \frac{\partial \phi_k^{-1}(x^1(\tilde{\xi}^1_i, \tilde{\xi}^2_i), x^2(\tilde{\xi}^1_i, \tilde{\xi}^2_i))}{\partial x^2}\right]\ .$$

Finite Element Spaces

The structure fe_space contains the following fields:

fe_space

```
├ n                                              [1 × 1]
├ boundary_bases                                 [1 × n_D]
└ support_and_extraction                         [m × 1 (struct)]
    ├ supported_bases                            [1 × #I_i]
    └ extraction_coefficients                    [#I_i × (p_1 + 1)(p_2 + 1)]
```

Here, n_D is the number of basis functions that are non-zero on Γ_D and boundary_bases contains the indices of these basis functions. In particular, if these indices are $[i_1, \dots, i_{n_D}]$, then for any finite element function $u_h = \sum_{i=1}^{n} u_i N_i$ for $u_i \in \mathbb{R}$,

$$u_h|_{\Gamma_D} = \sum_{\ell=1}^{n_D} u_{i_\ell} N_{i_\ell}|_{\Gamma_D} \; .$$

You are provided with a special kind of finite element space such that the basis functions corresponding to boundary_bases are *linearly independent* when restricted to Γ_D.

Next, support_extraction is a structure with m components. Its i-th component, accessible as support_extraction(i), contains two pieces of information:

- supported_bases contains the indices in I_i;
- if the j-th entry of supported_bases is k, then the (j, ℓ)-th entry of extraction_coefficients contains the coefficient $e_{k\ell i}$.

In other words, That is, if the j-th entry of "support_extraction(i).supported_bases" is k, then the $(j, :)$-th row of "support_extraction(i).extraction_coefficients" contains all coefficients in the following equation,

$$N_k|_{\Omega_i} = \sum_{\ell=1}^{(p_1+1)(p_2+1)} e_{k\ell i} \tilde{N}_{\ell i} \; .$$

Note that you are provided with a special choice of finite element basis functions such that the sum of all coefficients in the $(j, :)$-th row is equal to 1 for any j and i.

Finally, observe that for a finite element function $u_h = \sum_{i=1}^{n} u_i N_i$ for $u_i \in \mathbb{R}$, if the $(i, :)$-th row of supported_bases is $[j_1, \dots, j_{\#I_i}]$, then

$$u_h\big|_{\Omega_i} = \sum_{\ell=1}^{\#I_i} u_{j_\ell} N_{j_\ell}\big|_{\Omega_i} \; .$$

(4 points) **The finite element space**
REMARK: Note that the provide plots are made by evaluating the functions at 20 uniformly spaced points on each Ω_i (including the boundary points of Ω_i).

A. (4 points) Provide pseudo-code for your implementation of create_geometric_map.
In your pseudocode, reference the structures and fields as defined in the assignment. ***Do not*** *submit a screenshot of your* MATLAB *code, it will not qualify as pseudo-code.*

B. (0 points: Do not show these plots, this task is here so that you can verify that your implementation is behaving correctly.)
Choose the geometry star3. Visualize the finite element basis function N_3 on Ω and verify that it corresponds to the provided plots in `star3_N3.png`.

C. (0 points: Do not show these plots, this task is here so that you can verify that your implementation is behaving correctly.)
Choose the geometry star4. Visualize the finite element basis function N_3 on Ω and verify

that it corresponds to the provided plots in `star4_N3.png`.

D. (0 points: Do not show these plots, this task is here so that you can verify that your implementation is behaving correctly.)
Choose the geometry star5. Visualize the finite element basis function N_3 on Ω and verify that it corresponds to the provided plots in `star5_N3.png`.

REMARK: Just like in the 1D assignment, you can verify that the provided set of 2D basis functions also form a partition of unity.

Finite element assembly routine

Consider the Galerkin problem of interest in Equation (3), with

$$\mathcal{S}_h = \mathcal{W}_h := \{v \in \mathcal{F}(p,k;\Omega_h) \mid v|_{\Gamma_D} = 0\}\ .$$

We will additionally assume throughout that $g = 0$ in Equation (3).

Let the finite element solution be $u_h = \sum_{i=1}^n u_i N_i \in \mathcal{S}_h$. Then, the purpose of the finite element problem is to determine the coefficients u_i, $i = 1, \dots, n$, using either the boundary conditions or the linear system implied by the Galerkin problem.

In this assignment, you will perform this task by first assembling the following matrix and vector, respectively,

$$\underbrace{\begin{bmatrix} B(N_1,N_1) & B(N_1,N_2) & \cdots & B(N_1,N_n) \\ B(N_2,N_1) & B(N_2,N_2) & \cdots & B(N_2,N_n) \\ \vdots & \vdots & \ddots & \vdots \\ B(N_n,N_1) & B(N_n,N_2) & \cdots & B(N_n,N_n) \end{bmatrix}}_{=:\mathrm{A}\quad [n\times n]}\ ,\quad \underbrace{\begin{bmatrix} L(N_1) \\ L(N_2) \\ \vdots \\ L(N_n) \end{bmatrix}}_{=:\mathrm{b}\quad [n\times 1]}$$

That is, you **need to implement** a function:

```
function assemble_fe_problem(fe_space, ref_data, geom_map, problem_B, problem_L)
  ⋮
return A, b
```

where, given B and L, `problem_B` and `problem_L` are anonymous functions that (you **need to implement** these) such that

$$B(u_h, w_h) = \int_\Omega \texttt{problem\_B}(\mathbf{x}, u_h(\mathbf{x}), u_{h,x^1}(\mathbf{x}), u_{h,x^2}(\mathbf{x}), w_h(\mathbf{x}), w_{h,x^1}(\mathbf{x}), w_{h,x^2}(\mathbf{x}))\, dx^1 dx^2\ ,$$

$$L(w_h) = \int_\Omega \texttt{problem\_L}(\mathbf{x}, w_h(\mathbf{x}))\, dx^1 dx^2\ .$$

(16 points) **The finite element problem**

E. (4 points) Provide pseudo-code for your implementations of assemble_fe_problem. *In your pseudocode, reference the structures and fields as defined in the assignment.* ***Do not*** *submit a screenshot of your* MATLAB *code, it will not qualify as pseudo-code.*

F. (1 point) Let the indices in fe_space.boundary_bases be $[i_1, \dots, i_{n_D}]$. Argue why $u_h \in \mathcal{S}_h$ implies that

$$u_{i_1} = \cdots = u_{i_{n_D}} = 0\ .$$

G. (1 point) Using the previous problem, specify how you can find an $(n - n_D) \times (n - n_D)$

submatrix $\tilde{\mathrm{A}}$ of A, and an $(n - n_D) \times 1$ subvector $\tilde{\mathrm{b}}$ of b, such that $\tilde{\mathrm{A}}^{-1}\tilde{\mathrm{b}}$ will yield the remaining unknown coefficients u_i.

H. (5 points) Choose the geometry distressed_robotD (for which $\Gamma_N = \emptyset$) and let $\kappa = 1$ and $f = 1$.

- (3.5 points) Compute $\tilde{\mathrm{A}}$ and $\tilde{\mathrm{b}}$ and save them in text files `A.txt` and `b.txt`, respectively (use the same format as the data you are provided with). Submit the files in a zip file named `distressed_robotD.zip`.
- (1.5 point) Compute the finite element solution and plot its values and (piecewise-)derivatives in your report.

I. (5 points) Choose the geometry distressed_robotDN (for which $\Gamma_N \neq \emptyset$) and let $\kappa = 1$ and $f = \sin(x)\sin(y)$.

- (3.5 points) Compute $\tilde{\mathrm{A}}$ and $\tilde{\mathrm{b}}$ and save them in text files `A.txt` and `b.txt`, respectively (use the same format as the data you are provided with). Submit the files in a zip file named `distressed_robotDN.zip`.
- (1.5 point) Compute the finite element solution and plot its values and (piecewise-)derivatives in your report.