Technische Universität Berlin
Institut für Mathematik
Dr. D. Peschka, A. Selahi

WS 2019/20

Numerical Mathematics II for Engineers

Homework assignment 9-10 : *Finite element mini-project.*

Programming assignments can be solved in MATLAB/Python/Julia.

Deadline : submit until January, Monday 20, 2020.

Goal :

In this mini-project over the next two weeks you learn some theoretical and practical ingredients necessary to set up a variational formulation and to assemble the corresponding FE Galerkin matrix in 1D and 2D. The extension to 3D and beyond is mostly analogous.

1. Exercise : Weak forms **8 (+2) points**

Let $\Omega \subset \mathbb{R}^n$ and $\Gamma \subset \partial\Omega$ and $\Gamma_n = \partial\Omega \setminus \Gamma$. Furthermore let $V \equiv H^1(\Omega)$ the usual Sobolev space and $V_0 \equiv H^1_0(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \Gamma\}$. Consider the following bilinear and linear forms

$$a(u,v) = \int_\Omega a(x)\nabla u(x)\cdot\nabla v(x) + c(x)u(x)v(x)\,\mathrm{d}x \tag{1}$$

$$a(u,v) = \int_\Omega a(x)\nabla u(x)\cdot\nabla v(x)\,\mathrm{d}x + \int_{\Gamma_n} \alpha(x)u(x)v(x)\mathrm{d}s \tag{2}$$

$$a(u,v) = \int_\Omega a(x)\nabla u(x)\cdot\nabla v(x) + [\mathbf{b}(x)\cdot\nabla u(x)]v(x)\,\mathrm{d}x \tag{3}$$

$$a(u,v) = \int_\Omega a(x)\nabla u(x)\cdot\nabla v(x) - [\nabla\cdot(\mathbf{b}(x)v(x))]u(x)\,\mathrm{d}x \tag{4}$$

$$a(\mathbf{u},\mathbf{v}) = \int_\Omega C_{ijkl}(x)\varepsilon_{ij}(\mathbf{u}) : \varepsilon_{kl}(\mathbf{v})\,\mathrm{d}x \tag{5*}$$

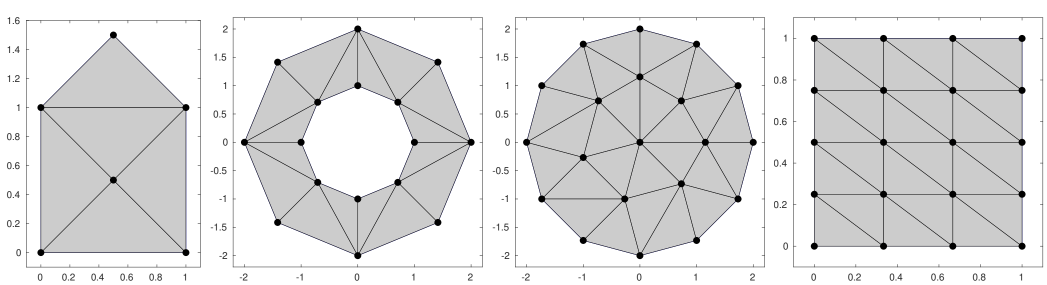
where $\varepsilon_{ij}(\mathbf{w}) = \frac{1}{2}(\partial_i w_j + \partial_j w_i)$ and $\mathbf{w} : \Omega \to \mathbb{R}^n$ with components $w_j \in V_0$. Use

$$f(v) = \int_\Omega f(x)v(x)\,\mathrm{d}x + \int_{\Gamma_n} g(x)v(x)\mathrm{d}s \quad \text{for some} \quad f \in L^2(\Omega).$$

For each variational statement, where we seek $u \in V_0$ such that $a(u,v) = f(v)$ for all $v \in V_0$, assume extra regularity and derive the corresponding elliptic PDE <u>and</u> boundary conditions for u, where (5*) is optional and uses $f(\mathbf{v}) = \int_\Omega f(x)(\nabla\cdot\mathbf{v}(x))\,\mathrm{d}x$.

2. Exercise : Basis functions **6 (+3) points**

Consider the two-dimensional decompositions shown below.



Examples : House, Annulus, Disc, Square

Please turn the page !

a) How many points n_p, faces/edges n_f, elements n_e has each mesh ?

b) How many points & faces are on the boundary $\partial\Omega$?

c) Let $\Gamma = \partial\Omega$. What is the dimension of the finite element discretization $V^h \subset V$ and $V_0^h \subset V_0$ for piecewise linear and for piecewise quadratic basis functions ?

***d)** For a given data structure `e2` as previously discussed for two-dimensional simplex triangulations, write a function `[np,nf,ne]=getnumbers(e2)` which returns n_p, n_f, n_e for arbitrary 2D simplex decompositions. While n_p, n_e are trivial to obtain, the main task is to determine n_f. Test on `haus,mesh1,mesh2,mesh3.ele/poly`.

3. Exercise : FEM – variational form in 1D **15 (+1) points**

In this exercise you write a program to solve the problem

$$a(u,v) = \int_0^1 a(x)u'(x)v'(x) + c(x)u(x)v(x)\,\mathrm{d}x = \int_0^1 f(x)v(x)\,\mathrm{d}x = f(v)$$

by construction of the corresponding Galerkin matrix A and solving $Au = f$. The main task is to build the matrix $(A_h)_{ij} = a(\varphi_j, \varphi_i)$ using the symmetric, bilinear form a. We divide the construction in different steps, which are generalizable to higher dimension.

step 1 : **element generation :** Write a

```
function [ne,np,e1]=generateelements1D(x)
```

which for given array of points `x` with `0=x(1)<x(2)<...<x(end)=1` returns the number of elements `ne` and the number of points `np`. The variable `e1` is the `ne`$\times 2$ matrix for which `i1=e1(k,1)` and `i2=e1(k,2)` are the indices `i1,i2` of the left or right vertex of the element (interval) $\Omega_k =$`(x(i1),x(i2))` and `i2=i1+1`, respectively.

step 2 : **computation of transformation :** Consider the affine linear function T_k, which maps the reference element $\Omega_{\text{ref}} = (0,1)$ the element Ω_k. Write a

```
function [Fdet,Finv]=generatetransformation1D(k,e1,x)
```

which for given element `k` returns `Fdet`$=\det(\nabla T_k)$ and `Finv`$=(\nabla T_k)^{-1}$.

Hint : Since T_k is piecewise affine linear, `Fdet` and `Finv` are just constant scalars.

step 3 : **computation of local matrices :** In order to compute the Galerkin matrix A_h we need the local matrices M, S from the previous assignment. Write the two

```
function mloc=localmass1D(Fdet)
function sloc=localstiff1D(Fdet,Finv)
```

which for given `Fdet` and `Finv`$= G$ compute the element mass and stiffness matrices

$$M = \int_{\Omega_k} \varphi_i(x)\varphi_j(x)\,\mathrm{d}x = \int_{\Omega_{\text{ref}}} \hat\varphi_{\bar\imath}(x)\hat\varphi_{\bar\jmath}(x)|\texttt{Fdet}|\,\mathrm{d}x$$
$$S = \int_{\Omega_k} \varphi_i'(x)\varphi_j'(x)\,\mathrm{d}x = \int_{\Omega_{\text{ref}}} \hat\varphi_{\bar\imath}'(x)GG^T\hat\varphi_{\bar\jmath}'(x)|\texttt{Fdet}|\,\mathrm{d}x$$

with $\hat\varphi_1(x) = 1 - x$ and $\hat\varphi_2(x) = x$ from the previous assignment and $\bar\imath, \bar\jmath = 1, 2$. Note : In 1D G is a number, so is $G = G^T = 1/$`Fdet`.

See next page !

Theory question 1 : For piecewise linear basis functions : Explain how `e1` defines the mapping of indices `i=localtoglobalP11D(k,`$\bar{i}$`)` from a local degree of freedom $\bar{i}$ on the element Ω_k to a global degree of freedom i ?

Theory question 2 : Consider the 3 quadratic basis functions on the reference domain Ω_{red} defined by $\hat{\varphi}_{\bar{i}}(p_{\bar{j}}) = \delta_{\bar{i}\bar{j}}$ for $p_1 = 0$, $p_2 = 1$, $p_3 = 1/2$. Explicitly state the reference basis functions $\hat{\varphi}_{\bar{i}}(x)$ for $\bar{i} = 1, 2, 3$.

Theory question 3 (optional) : For piecewise quadratic basis functions : How can we extend the mapping of local to global degrees of freedom `i=localtoglobalP21D(k,`$\bar{i}$`)` for $\bar{i} = 1, 2, 3$ if we require `localtoglobalP11D(k,`$\bar{i}$`)=localtoglobalP21D(k,`$\bar{i}$`)` for $\bar{i} = 1, 2$?

step 4 : **construction of global matrix :** The construction of the Galerkin matrix for $a = 1$, $c = 0$ is done in the MATLAB code below. Please study the code `elliptic1d.m` and study how it works in detail once you implemented the missing functions by solving the PDE $-u'' = 1$ on $\Omega = (0, 1)$ with $u(0) = u(1) = 0$.

```
%% FEM MATLAB sample code
N    =  32;                          % 1D: number of points
nphi =   2;                          % P1: 2 local basis functions
x = linspace(0,1,N);                 % array of points
[ne,np,e1] = generateelements1D(x);  % generate mesh
localtoglobal1DP1 = e1;              % could it be this simple? why?

%% build matrices
ii = zeros(ne,nphi^2); % sparse i-index
jj = zeros(ne,nphi^2); % sparse j-index
aa = zeros(ne,nphi^2); % entry of Galerkin matrix
bb = zeros(ne,nphi^2); % entry in mass-matrix (to build rhs)

%% build global from local
for k=1:ne             % loop over elements
    [Fdet,Finv] = generatetransformation1D(k,e1,x); % compute trafo

    % build local matrices (mass, stiffness, ...)
    sloc = localstiff1D(Fdet,Finv); % element stiffness matrix
    mloc = localmass1D(Fdet);       % element mass matrix

    % compute i,j indices of the global matrix
    dofs = localtoglobal1DP1(k,:);
    ii( k,: ) = [dofs(1) dofs(2) dofs(1) dofs(2)]; % local-to-global
    jj( k,: ) = [dofs(1) dofs(1) dofs(2) dofs(2)]; % local-to-global

    % compute a(i,j) values of the global matrix
    aa( k,: ) = sloc(:);
    bb( k,: ) = mloc(:);
end
% create sparse matrices
A=sparse(ii(:),jj(:),aa(:));
M=sparse(ii(:),jj(:),bb(:));

% build rhs and take into account Dirichlet bcs, solve, plot
rhs = M*ones(N,1);
u   = zeros(N,1);
u(2:end-1) = A(2:end-1,2:end-1) \ rhs(2:end-1);
plot(x,u,x,x.*(1-x)/2,'r.')
```

a) Modify `elliptic1d.m/py/jl` to `elliptic1dwithc.*` to solve the problem with $a(x) = c(x) = f(x) = 1$ with homogeneous Dirichlet boundary conditions are compare with the exact solution $u = e^{-x}(1 - e^x)(e^x - e)/(1 + e)$.

b) Modify `elliptic1d.m/py/jl` into `elliptic1dinhom.*`, to compute the solution with $f(x) \equiv 1$, $c = 0$ and inhomogeneous Dirichlet boundary conditions $u(0) = 1$ and $u(1) = 0$ by modification of f as explained in the lecture. Compare with the exact solution.

c) Modify `elliptic1d.m/py/jl` into `elliptic1djump.*`, so that you compute a FEM solution of assignment 8, exercise 1. Compare with the exact solution.

Please turn the page!

4. Exercise : FEM – variational form in 2D **8 points**

a) What are the three affine linear reference basis functions $\hat{\varphi}_{\bar{i}}$ for $p_1 = (0,0)$, $p_2 = (1,0)$, $p_3 = (0,1)$ for the reference triangle $\bar{\Omega}_{\text{ref}} = \{(y_1, y_2) \in \mathbb{R}^2 : 0 \leq y_i \leq 1, y_1 + y_2 \leq 1\}$?

b) Based on the map $T_k : \bar{\Omega}_{\text{red}} \to \bar{\Omega}_k$ write the

`function [Fdet,Finv]=generatetransformation2D(k,e2,x,y)`

Validate this function by computing and printing to the screen the area of the house provided with the assignment using `Fdet` only. Write the code in the script `housearea.m/py/jl`. Use the previouly distributed function `readtria` to read a triangle mesh `x,y,np,ne,e2,idp,ide` from a file.

c) Based on the map $T_k : \bar{\Omega}_{\text{red}} \to \bar{\Omega}_k$, compute the following two integrals by change of variables

$$S_{ij} = \int_{\Omega_k} \nabla\varphi_i \cdot \nabla\varphi_j \,\mathrm{d}x, \qquad\qquad M_{ij} = \int_{\Omega_k} \varphi_i \varphi_j \,\mathrm{d}x.$$

and express the result using $G =$`Finv`$= (\nabla T_k)^{-1}$ and `Fdet`$= \det(\nabla T_k)$ and implement the corresponding :

`function mloc=localmass2D(Fdet)`
`function sloc=localstiff2D(Fdet,Finv)`

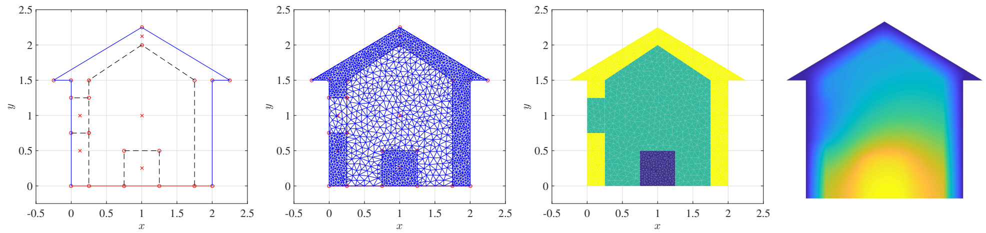
d) The mapping of local to global degrees of freedom is analogous to the previous exercise derived from `e2`. This will result in the assembly of the Galerkin matrix looking like :

```
%% FEM MATLAB sample code
[x,y,np,ne,e2,idp,ide] = readtria('disc'); % read mesh from file
localtoglobal2DP1 = e2;                    % could it be this simple? why?
int               = ~(idp==1);             % select points without Dirichlet bc
nphi = 3;
%% build matrices
ii = zeros(ne,nphi^2); % sparse i-index
jj = zeros(ne,nphi^2); % sparse j-index
aa = zeros(ne,nphi^2); % entry of Galerkin matrix
bb = zeros(ne,nphi^2); % entry in mass-matrix (to build rhs)
%% build global from local
for k=1:ne              % loop over elements
    [Fdet,Finv] = generatetransformation2D(k,e2,x,y); % compute trafo

    % build local matrices (mass, stiffness, ...)
    sloc = localstiff2D(Fdet,Finv); % element stiffness matrix
    mloc = localmass2D(Fdet);       % element mass matrix

    % compute i,j indices of the global matrix
    dofs = localtoglobal2DP1(k,:);
    ii( k,: ) = [dofs([1 2 3 1 2 3 1 2 3])]; % local-to-global
    jj( k,: ) = [dofs([1 1 1 2 2 2 3 3 3])]; % local-to-global

    % compute a(i,j) values of the global matrix
    aa( k,: ) = sloc(:);
    bb( k,: ) = mloc(:);
end
% create sparse matrices
A=sparse(ii(:),jj(:),aa(:));
M=sparse(ii(:),jj(:),bb(:));
% build rhs and take into account Dirichlet bcs, solve, plot
rhs = M*ones(np,1);
u   = zeros(np,1);
u(int) = A(int,int) \ rhs(int);
%% plotting
trisurf(e2,x,y,u)
```

Check this works with your implemented functions on a disc (`disc.ele/poly`) or box `box.ele/poly` domain and modify the `elliptic2d.m/py/jl` to `checksolution2D.*` that prints the error. Compare with an exact solution.

5. Exercise : Heating problem **8 (+2) points**

Consider the house geometry shown below. We provide the mesh files, so that you can load the geometry using the previously provided function `readtria`.



House : geometry, mesh (`house.1.ele/node`), material zones (`ide`), temperature distribution u

This house has different material zones indicated by color in the right panel. The insulating wall $\Omega_{\text{insulator}}$ `ide==3` is yellow with low heat conductivity, the room Ω_{room} `ide==2` is cyan with medium heat conductivity, the heater Ω_{heater} `ide==1` is dark blue with high heat conductivity. The material data (different from the first chapter of the lecture notes) are

$$a_{\text{insulator}} = 0.1, \qquad a_{\text{room}} = 1.5, \qquad a_{\text{heater}} = 4, \qquad f_{\text{heater}} = H \in \mathbb{R}.$$

and $f_{\text{room}} = f_{\text{insulator}} = 0$. All outside walls are at temperature $u = 0$ (say Celsius) and the bottom floor is well insulated $\nu \cdot \nabla u = 0$. The stationary heat flux is defined $\mathbf{q}(x) = -a(x)\nabla u(x)$ and the stationary heat distribution satisfies $\nabla \cdot \mathbf{q} = f$ with the given boundary conditions.

a) State the variational problem and the corresponding space by defining $V_0, a(x), f(x)$ based on the information provided above. How do you need to define `int` to obtain V_0 ?

b) Show that the heat loss is $|\Omega_{\text{room}}|H$ (see lecture notes).

c) Extend the script from exercise 5 to deal with $a(x), f(x)$ which have the different constant values on different elements identified using `ide` for the house geometry shown above.

d) How do you need to choose H to reach an average room temperature

$$u_{\text{room}} = |\Omega_{\text{room}}|^{-1} \int_{\Omega_{\text{room}}} u \, \mathrm{d}x = 19 \quad \text{(say Celsius)},$$

where $|\Omega_{\text{room}}| = \int_{\Omega_{\text{room}}} 1 \, \mathrm{d}x$. Note that in this exercise we entirely neglect physical units. Modify `elliptic2d.m/py/jl` to a script `roomtemperature.*` which computes the solution and plots the final temperature distribution in the house.

e) (optional) How does H and the heat loss reduce if you replace the window $[0, \frac{1}{4}] \times [\frac{3}{4}, \frac{5}{4}]$ by insulation of the same thickness as the yellow part above/below. Implement in `nowindow.*`.

Hint : In order to compute $|\Omega_{\text{home}}|$ it is best to construct a matrix $C_{ij} = \int_\Omega \chi_{\text{room}} \varphi_i \varphi_j \, \mathrm{d}x$ so that $|\Omega_{\text{home}}| = w \cdot Cw$ and $u_{\text{room}} = (w \cdot Cu)/(w \cdot Cw)$ with $w = (1, ..., 1)^\top$ and

$$\chi_{\text{room}}(x) = \begin{cases} 1 & x \in \Omega_{\text{room}} \\ 0 & \text{otherwise} \end{cases}.$$

total sum : 45 (+8) points