Cooking beef - and other applications of FEM in the kitchen

Programming project in TMA4220

by Kjetil André Johannessen and Trond Kvamsdal
TMA4220 - Numerical solution of partial differential equations using the finite element method

1 Gaussian quadrature

At the heart of every finite element code, lies the evaluation of an integral. This integral may be of varying complexity depending on the problem at hand, and many of these integrals does not even have a known analytical solution. Some integrals are *possible* to solve analytically, but of such computational complexity that it is impractical to do so. As such, one often refers to numerical integration schemes to do the core integration. One popular integration scheme is the *Gaussian quadrature*.

In one dimension the gauss quadrature takes the form

$$\int_{-1}^{1} g(z)dz \approx \sum_{q=1}^{N_q} \rho_q g(z_q),$$

where N_q is the number of integration points, z_q are the Gaussian quadrature points and ρ_q are the associated Gaussian weights.

This extends to higher dimensions by

$$\int_{\hat{\Omega}} g(\boldsymbol{z})d\boldsymbol{z} \approx \sum_{q=1}^{N_q} \rho_q g(\boldsymbol{z}_q),$$

and specifying the vector quadrature points $\boldsymbol{z}_q$ as well as integrating over a suitable reference domain $\hat{\Omega}$ (i.e. squares or triangles in 2D, tetrahedra or cubes in 3D).

a) 1D quadrature

Write a matlab function `I = quadrature1D(a,b,Nq,g)`. With the following arguments:

`I`	$\in \mathbb{R}$	value of the integral
`a`	$\in \mathbb{R}$	integration start
`b`	$\in \mathbb{R}$	integration end
`Nq`	$\in [1,4]$	number of integration points
`g`	$: \mathbb{R} \to \mathbb{R}$	function pointer*

verify that the function evaluates correctly by comparing with the analytical solution of the integral

$$\int_{1}^{2} e^x \, dx$$

N_q	z_q	ρ_q
1-point-rule	0	2
2-point-rule	$-\sqrt{1/3}$	1
	$\sqrt{1/3}$	1
	$-\sqrt{3/5}$	5/9
3-point-rule	0	8/9
	$\sqrt{3/5}$	5/9
	$-\sqrt{\frac{3+2\sqrt{6/5}}{7}}$	$\frac{18-\sqrt{30}}{36}$
4-point-rule	$-\sqrt{\frac{3-2\sqrt{6/5}}{7}}$	$\frac{18+\sqrt{30}}{36}$
	$\sqrt{\frac{3-2\sqrt{6/5}}{7}}$	$\frac{18+\sqrt{30}}{36}$
	$\sqrt{\frac{3+2\sqrt{6/5}}{7}}$	$\frac{18-\sqrt{30}}{36}$

Table 1: 1D gauss quadrature

b) 2D quadrature

Using all numerical quadratures, it is important to first map the function to the referance domain. In one dimension, this is the interval $\zeta \in [-1, 1]$. In higher dimensions, we often map to barycentric coordinates (or area coordinates as they are known in 2D). The gauss points are then given as triplets in this coordinate system. The area coordinates are defined by

$$\begin{aligned} \zeta_1 &= \frac{A_1}{A} \\ \zeta_2 &= \frac{A_2}{A} \\ \zeta_3 &= \frac{A_3}{A} \end{aligned}$$

where A_1, A_2 and A_3 are the area of the triangles depicted in figure 1 and A is the total area of the triangle. Note that these do not form a linear independent basis as $\zeta_1 + \zeta_2 + \zeta_3 = 1$.

N_q	$(\zeta_1, \zeta_2, \zeta_3)$	ρ
1-point rule	$(1/3, 1/3, 1/3)$	1
	$(1/2, 1/2, 0)$	1/3
3-point rule	$(1/2, 0, 1/2)$	1/3
	$(0, 1/2, 1/2)$	1/3
	$(1/3, 1/3, 1/3)$	-9/16
4-point rule	$(3/5, 1/5, 1/5)$	25/48
	$(1/5, 3/5, 1/5)$	25/48
	$(1/5, 1/5, 3/5)$	25/48

Table 2: 2D gauss quadrature

Write a matlab function `I = quadrature2D(p1,p2,p3,Nq,g)`. With the following arguments:

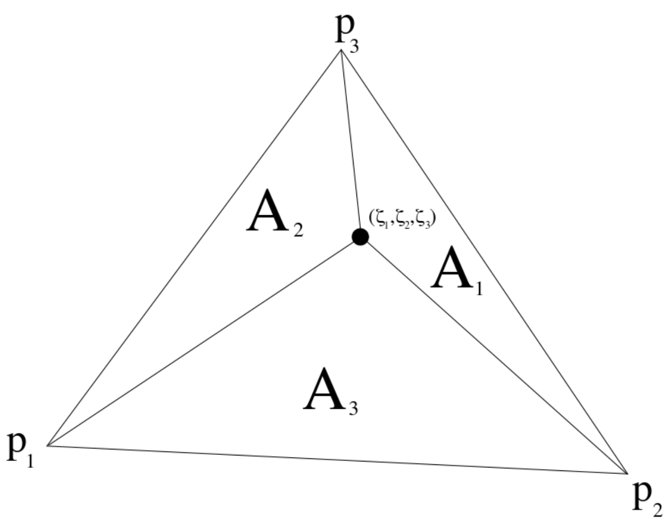

Figure 1: Barycentric coordinates in two dimensions

`I`	$\in \mathbb{R}$	value of the integral
`p1`	$\in \mathbb{R}^2$	first corner point of the triangle
`p2`	$\in \mathbb{R}^2$	second corner point of the triangle
`p3`	$\in \mathbb{R}^2$	third corner point of the triangle
`Nq`	$\in \{1, 3, 4\}$	number of integration points
`g`	$: \mathbb{R}^2 \to \mathbb{R}$	function pointer*

verify that the function evaluates correctly by comparing with the analytical solution of the integral

$$\iint\limits_{\Omega} \log(x + y)\, dx\, dy,$$

where Ω is the triangle defined by the corner points $(1, 0)$, $(3, 1)$ and $(3, 2)$.

c) 3D quadrature

The extension of the barycentric coordinates to 3dimensions and tetrahedral elements, should be straightforward. The integration schemes can be found in the following table

Write a matlab function `I = quadrature3D(p1,p2,p3,p4,Nq,g)`. With the following arguments:

N_q	$(\zeta_1, \zeta_2, \zeta_3, \zeta_4)$	ρ
1-point rule	$(1/4, 1/4, 1/4, 1/4)$	1
4-point rule	(0.5854102, 0.1381966, 0.1381966, 0.1381966)	0.25
	(0.1381966, 0.5854102, 0.1381966, 0.1381966)	0.25
	(0.1381966, 0.1381966, 0.5854102, 0.1381966)	0.25
	(0.1381966, 0.1381966, 0.1381966, 0.5854102)	0.25
5-point rule	$(1/4, 1/4, 1/4, 1/4)$	-4/5
	$(1/2, 1/6, 1/6, 1/6)$	9/20
	$(1/6, 1/2, 1/6, 1/6)$	9/20
	$(1/6, 1/6, 1/2, 1/6)$	9/20
	$(1/6, 1/6, 1/6, 1/2)$	9/20

Table 3: 3D gauss quadrature

`I`	$\in \mathbb{R}$	value of the integral
`p1`	$\in \mathbb{R}^3$	first corner point of the tetrahedron
`p2`	$\in \mathbb{R}^3$	second corner point of the tetrahedron
`p3`	$\in \mathbb{R}^3$	third corner point of the tetrahedron
`p4`	$\in \mathbb{R}^3$	fourth corner point of the tetrahedron
`Nq`	$\in \{1, 4, 5\}$	number of integration points
`g`	$: \mathbb{R}^3 \to \mathbb{R}$	function pointer*

verify that the function evaluates correctly by comparing with the analytical solution of the integral

$$\iiint\limits_{\Omega} e^x \, dx \, dy \, dz,$$

where Ω is the tetrahedron defined by the corner points $(0,0,0)$, $(0,2,0)$, $(0,0,2)$ and $(2,0,0)$.

(*) A function pointer in `matlab` is a variable which represents a function instead of the usual numerical values. In its simplest form it is declared as

```
f = @(x) x^2 + 1
```

which would cause the *variable* `f` to contain a *pointer* to the function $f(x) = x^2 + 1$. The function can then be evaluated using one of two methods

```
y = f(4);
y = feval(f,4);
```

both of which should yield the same result `y = 17`. A function may take in several arguments, i.e. $f(x, y) = x^2 + y^2$ may be declared as

```
f = @(x,y) x^2 + y^2
```

again the evaluation of the function is straightforward

```
y = f(2,2);
y = feval(f,2,2);
```

Provided that the actual function body is capable of vector or matrix operations, then the input arguments may be of vector or matrix form. The syntax remains unchanged by this. You may also use variables in the function declaration, i.e.

```
a = 2;
f = @(x) x*a
```

will result in a function `f` which is doubling its input argument (even if `a` is changed at a later point).

2 Poisson in 2 dimensions

We are going to solve the two-dimensional Poisson problem, given by

$$\begin{aligned} \nabla^2 u(x,y) &= -f(x,y) \\ u(x,y)|_{r=1} &= 0, \end{aligned} \tag{1}$$

with f given in polar coordinates as

$$f(r,\theta) = -8\pi\cos\left(2\pi r^2\right) + 16\pi^2 r^2 \sin\left(2\pi r^2\right)$$

on the domain Ω given by the unit disk, i.e. $\Omega = \left\{(x,y) \, : \, x^2 + y^2 \leq 1\right\}$.

a) Analytical solution

Verify that the following expression is in fact a solution to the problem (1)

$$u(x,y) = \sin\left(2\pi(x^2 + y^2)\right). \tag{2}$$

b) Weak formulation

Show that the problem can be rewritten as

$$a(u,v) = l(v), \quad \forall v \in X.$$

with the bilinear functional a and the linear functional l given by

$$\begin{aligned} a(u,v) &= \iint\limits_{\Omega} \nabla u \cdot \nabla v \, dx \, dy, \\ l(v) &= \iint\limits_{\Omega} f v \, dx \, dy. \end{aligned}$$

What is the definition of the space X?

c) Galerkin projection

Instead of searching for a solution u in the entire space X we are going to be looking for a solution in a much smaller space $X_h \subset X$. Let Ω be discretized into K triangles such that our computational domain is the union of all of these $\Omega = \cup_{k=1}^{K} T_h^k$. Each triangle T_h^k is then defined by its three corner *nodes* $\boldsymbol{x}_i$. For each of these nodes there corresponds one basis function. The space X_h is then defined by

$$X_h = \left\{ v \in X \, : \, v|_{T_h^k} \in \mathbb{P}_1(T_h^k), 1 \leq k \leq K \right\}$$

for which the basis functions $\{\varphi_i\}_{i=1}^{n}$ satisfy

$$X_h = \text{span}\{\varphi_i\}_{i=1}^{n} \qquad \varphi_j(\boldsymbol{x}_i) = \delta_{ij} \tag{3}$$

and δ_{ij} is the Kronecker delta. By searching for a solution $u_h \in X_h$, it is then possible to write this as a weighted sum of the basis functions, i.e. $u_h = \sum_{i=1}^{n} u_h^i \varphi_i(x,y)$.

Show that the problem "Find $u_h \in X_h$ such that $a(u_h, v) = l(v) \quad \forall v \in X_h$" is equivalent to the following problem

Find $\boldsymbol{u}$ such that

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f} \tag{4}$$

with

$$\begin{aligned}
\boldsymbol{A} &= [A_{ij}] = [a(\varphi_i, \varphi_j)] \\
\boldsymbol{u} &= [u_h^i] \\
\boldsymbol{f} &= [f_i] = [l(\varphi_i)].
\end{aligned}$$

d) Implementation

We are now going to actually solve the system (4). First we are going to take a look at the triangulation $\{T_h^k\}$. From the webpage `http://www.math.ntnu.no/emner/tma4220/2010h/` you may download the mesh generators. For organization purposes you might want to keep them in a separate directory and see the matlab `addpath` command.

The function `getDisk` is generating the unit disk Ω. Plot at least three meshes of different sizes using the mesh generated from this function. You may want to check the matlab function `trimesh` or `triplot`.

e) Stiffness matrix

Build the stiffness matrix $\boldsymbol{A}$. You may choose if you perform the integration analytically or by Gaussian quadrature.

The matrix $\boldsymbol{A}$ should now be singular. Verify this in your code and explain why this is the case.

f) Right hand side

Build the right hand side vector $\boldsymbol{f}$ in the same manner as $\boldsymbol{A}$. Here you might need to resort to Gaussian quadrature.

g) Boundary conditions

Implement the homogeneous dirichlet boundary conditions. Describe what method you used for this and how you did it.

h) Verification

Solve the system (4) and verify that you are getting (approximately) the same result as the analytical solution (2).

3 Cooking beef

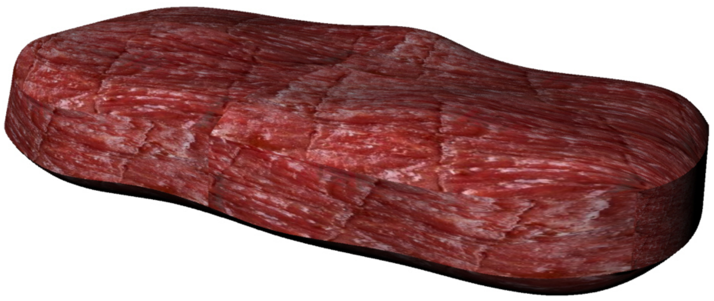

Figure 2: The beef which is to be studied

a) The Poisson in 3d

We are now going to solve the problem

$$\begin{aligned} \nabla^2 u &= f \\ u|_{\partial\Omega} &= 0 \end{aligned} \tag{5}$$

in three dimensions, meaning that we are looking for a solution $u(x, y, z)$.

Generate a mesh, using the function `getBeef` from the downloaded mesh generators. This will give you three variables which will describe the nodal points, the tetrahedral elements and the index of the boundary nodes. These should be familiar from task 2 as the only difference is that spatial coordinates have one more component, as well as the elements require one more index to describe. Note that the function `getBeef` takes three input arguments. This is since the beef is stored as a parametric volume described by three parametric variables (ξ_1, ξ_2, ξ_3). You will be asked to specify the tessellation in each of these directions separately, see figure 3 for details.

Modify your code from task 2 to deal with tetrahedral elements in three dimensions. Use the following f

$$f(x, y, z) = \frac{1}{x^2 + y^2 + z^2}$$

and homogeneous Dirichlet boundary conditions ($u^D = 0$).

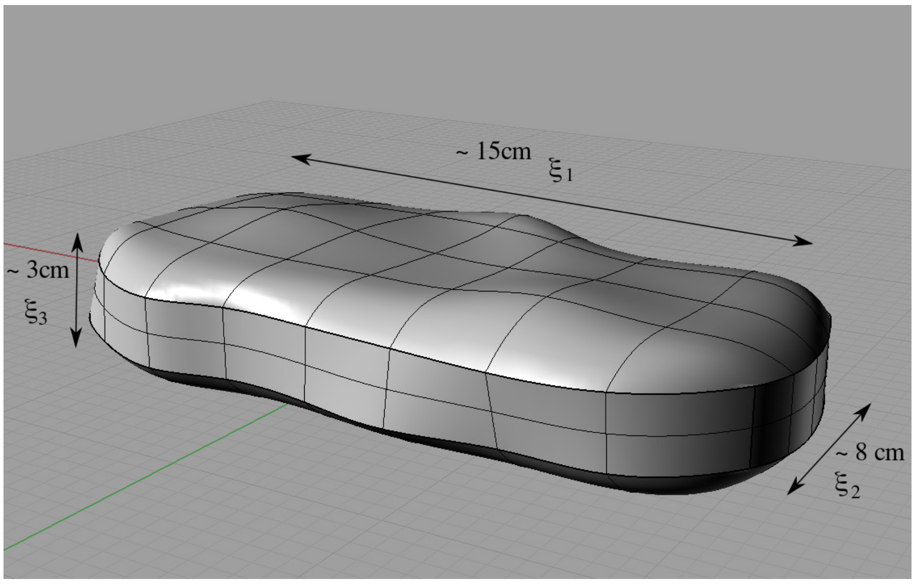

Figure 3: Computational domain Ω

b) Volume visualization

Plot the domain Ω (i.e. the beef). Note that you will not be required to plot every element, as most will be hidden on the inside of the domain. See the matlab function `TriRep` for functionality relating to this.

Plot your solution using isosurfaces. Note that the matlab function `isosurface` requires your data to be structured, which it currently is not. You will have to post process the data to get it on the desired form. Read up on `TriScatteredInterp` for this.

You are by no means limited to the above functions for plotting. Feel free to experiment using different techniques or functions.

c) The heat equation

The heat equation reads

$$\begin{aligned} \frac{\partial u}{\partial t} &= \alpha \nabla^2 u \\ u(t,x,y,z)|_{\partial\Omega} &= u^D \\ u(t,x,y,z)|_{t=0} &= u_0(x,y,z) \end{aligned} \tag{6}$$

where α is an positive constant defined by

$$\alpha = \frac{\kappa}{c_p \rho}$$

with κ^{**} being the thermal conductivity, ρ^{**} the mass density and c_p^{**} the specific heat capacity of the material.

We are going to *semidiscretize* the system by projecting the spatial variables to a finite element subspace X_h. Multiply (6) by a test function v and integrate over the domain Ω to get

$$\iiint\limits_{\Omega} \frac{\partial u}{\partial t} v \, dV = \iiint\limits_{\Omega} \alpha \nabla u \nabla v \, dV$$

Note that we have only semidiscretized the system, and as such our unknown u is given as a linear combination of the *spatial* basis functions, and continuous in time, i.e.

$$u_h(x, y, z, t) = \sum_{i=1}^{n} u_h^i(t) \varphi_i(x, y, z).$$

The variational form of the problem then reads: Find $u_h \in X_h^D$ such that

$$\begin{aligned} &\iiint\limits_{\Omega} \tfrac{\partial u}{\partial t} v \, dV = \iiint\limits_{\Omega} \alpha \nabla u \nabla v \, dV, \quad \forall v \in X_h \\ \Rightarrow \quad &\sum_i \iiint\limits_{\Omega} \varphi_i \varphi_j dV \tfrac{\partial u_h^i}{\partial t} = \sum_i \iiint\limits_{\Omega} \alpha \nabla \varphi_i \nabla \varphi_j dV \, u_h^i \quad \forall j \end{aligned}$$

which in turn can be written as the linear system

$$\boldsymbol{M} \frac{\partial \boldsymbol{u}}{\partial t}(t) = \boldsymbol{A} \boldsymbol{u}(t) \tag{7}$$

which is an ordinary differential equation (ODE) with the matrices defined as

$$\begin{aligned} \boldsymbol{A} = [A_{ij}] &= \iiint\limits_{\Omega} \alpha \nabla \varphi_i \nabla \varphi_j \, dV \\ \boldsymbol{M} = [M_{ij}] &= \iiint\limits_{\Omega} \varphi_i \varphi_j \, dV. \end{aligned}$$

Construct the matrix $\boldsymbol{A}$ and $\boldsymbol{M}$ as defined above.

d) Time integration

The system (7) is an ODE, which should be familiar from previous courses. Very briefly an ODE is an equation on the form

$$\frac{\partial y}{\partial t} = f(t, y)$$

where y may be a vector. The simplest ODE solver available is Eulers method

$$y_{n+1} = y_n + h f(t_n, y_n).$$

More sophisticated include the improved eulers methods

$$y_{n+1} = y_n + \frac{h}{2} \left(f(t_n, y_n) + f(t_{n+1}, y_n + h f(t_n, y_n)) \right)$$

or the implicit trapezoid rule

$$y_{n+1} = y_n + \frac{h}{2} \left(f(t_n, y_n) + f(t_{n+1}, y_{n+1}) \right)$$

and the famous Runge Kutta methods.

Choose an ODE scheme (based on your previous experience and expertize) and implement your time integration. Why did you choose the solver you did?

e) Experimentation

The boundary conditions are the physical variables which we have control over. The initial condition $u(t, x, y, z)|_{t=0}$ is the beef as it is prior to any cooking. A proper choice here would be room temperature, say 20°C.

The actual cooking will be a product of the dirichlet boundary conditions. Frying the beef on a pan will result in a high (how high?) temperature on the bottom and room temperature on the other sides of the beef. What should be done to turn the beef and fry the other side? When should we turn it? Cooking it in an oven would result in a uniform boundary conditions on *all* sides of say 225°C. How long will it have to stay in? Is it a good idea to keep it in room temperature after cooking (and how does this change the boundary conditions)? More exotic cooking techniques include wrapping it in plastic and putting it in a water bath (not boiling) for some time, and only frying it on a pan for seconds prior to serving.

Experiment around by cooking it in a number of ways using different boundary conditions. The optimality criterion is left up to the student. How well is your optimal beef cooked?

(**) Physical proprties of meat

It is hard to generalize too much on the physical properties of the beef as they are dependant on a number of variables outside the scope of this task. Not only are they dependant on the meat composition (i.e. what primal cut it is derived from), but it is also dependant on the temperature. Try and find good approximations for these numbers. A start may be the work of Pan and Singh ("Physical and Thermal Properties of Ground Beef During Cooking") which suggests that the density ρ is in the range 1.006 to 1.033 g/cm^3 and the thermal conductivity κ in the range 0.35 to 0.41 W/m·K. The specific heat capacity is not mentioned in the abstract, but may be commented on in the actual article for those that get their hands on the entire document.

Unconfirmed sources list the specific heat capacity c_p of meat as 3.973 J/kg·K. You may use these values, or better yet: find more reliable, documented values.