Master Vietnam-France in HCMC
High Performance computing

TP 0 : Taking matters in hand - A short review of basic Finite Elements functions in a few steps

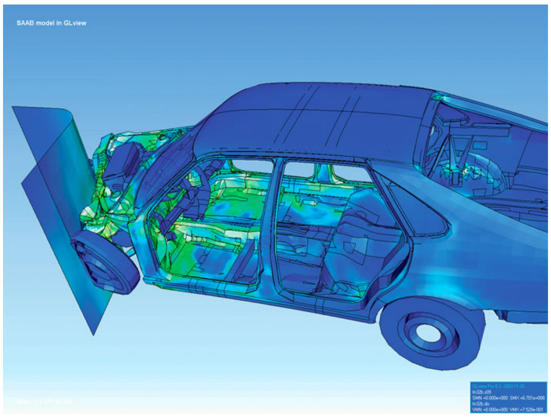


FIGURE 1 – Visualization of how a car deforms in an asymmetrical crash using finite element analysis
(https ://commons.wikimedia.org/w/index.php?curid=641911)

The aim of this session is to recall and use some basic functions of Finite Elements problems without any crash.

The classical Laplace problem is defined in a domain $D \subset \mathbb{R}^2$,

$$\begin{cases} -\nabla\cdot(\mu\nabla u) = -\Delta u = f & \text{in } D \quad (\mu = 1) \\ u = g & \text{on } \partial D \end{cases} \tag{1}$$

Domain D is meshed with a set of triangles $\Omega = \cup_k \Omega_k$, $\Gamma_k = \partial\Omega_k$. μ is a function constant per triangle that characterises the material (in our case $\mu = 1$) Applying a variational formulation and using P_1 Finite Elements, one obtains the following formulation :

$$\forall \phi \in P_1, \int_\Omega \nabla u \nabla \phi \; dx \quad = \int_\Omega f \phi dx \tag{2}$$

Let $V = (\phi_i)_{i=1,N}$ be a a generating family of P_1, the P_1 finite element problem can be defined as

Find $u = \sum_i u_i \phi_i$ such that

$$\forall \phi_j \in V, \;\; \sum_{i=1,N} u_i \int_\Omega \nabla\phi_i \nabla\phi_j \; dx = \int_\Omega f \phi_j dx \quad j = 1, N \tag{3}$$

Let Nnodes be the number of nodes in the mesh. Eq. 3 can be written in matrix form as

$$\text{Find } X = (u_i) \text{ in } \mathbb{R}^2 \text{ , such that } KX = MF$$

with $F = M(f_j)_{j=1,Nnodes}$ and $f_j = f(x_j)$ with x_j the node associated to ϕ_j for all nodes.
K is the stiffness matrix , M is the mass matrix (more details are to be found in FEDforstudents.pdf)

Step 1 = Meshing

Provided :

- Newmesh.m : [N,T,P] = NewMesh(G) : A function generating a finite element mesh. G is an integer designing the object being meshed (triangle, square, space shuttle, ...)
  N is the node array of 2D coordinates
  T is the triangle array : T (triangle number i, 1 :6) where T(i,1 :3) are the 3 node numbers that define the triangle given in trigonometrical order, T(i, 4 :6) gives the type of edge (0 = inner edge, 1 = edge with a Dirichlet boundary condition, 2=edge with a Neumann boundary condition)
  P is a 1d array that associates to each triangle a material characteristic (in our case $P(i) = \mu = 1$)
- RefineMesh.m : A function refining a given Finite Element mesh
- PlotMesh.m : A function that visualizes the mesh

To do

- Understand how they work = Create a mesh (G=6) and see how boundary conditions are taken into account. Visualize it.
- Refine the mesh twice and visualize the 2 refinements.

Step 2 = Solving - Direct solvers - Matlab backslash solver

Provided :

- ComputeElementMassMatrix.m : Me=ComputeElementMassMatrix(t) a function that generates the mass matrix associated to the triangular element described by its three nodal coordinates
  t=[x1 y1 ;x2 y2 ;x3 y3], where the nodes are labelled trigonometrically.
- ComputeElementStiffnessMatrix.m : Ke=ComputeElementStiffnessMatrix(t) a function that generates the stiffness matrix associated to the triangular element described by its three nodal coordinates as above.
- PlotSolution.m : PlotSolution(u,N,T) a function that will plot a vector $u = (u_i)$ defined on all the nodes N defining a triangular mesh of triangles T.
- FEpoisson.m

To do 1

A Matlab script that solves the Laplace equation Eq.3 with Dirichlet boundary equation on a square given by NewMesh(6) refined twice with a given **manufactured solution** and solved with the Matlab backslash operator using the provided functions.

There are two ways of writing the script.

Version 1

The first is to consider the problem associated to the values of u on **all** the nodes.
What do you need ?

- An array of dimension the number of nodes equal to 0 if an inner node and equal to 1 if on the border.

— The sparse mass matrix M of size (Nnodes , Nnodes) assembled with ComputeElementMassMatrix.m
— The sparse stiffness matrix K of size (Nnodes , Nnodes) assembled with ComputeElementStiffnessMatrix.m . For each line associated to a boundary point , all elements should be set to 0 except for the diagonal element set to 1
— The right hand side associated to Eq.3 where each coefficient associated to a boundary point is set to the Dirichlet value.

Look at FEPoisson.m

Version 2

The second is to consider the problem associated to the values of u on all **the inner** nodes.
What do you need ?
Let INodes be the number of inner nodes.

— An array of dimension the number of nodes equal to 0 if an inner node and equal to 1 if on the border.
— The sparse mass matrix M of size (INnodes , INnodes) assembled with ComputeElementMassMatrix.m
— The sparse stiffness matrix K of size (INnodes , INnodes) assembled with ComputeElementStiffnessMatrix.m .
— The right hand side associated to Eq.3 modified by the contribution of Dirichlet nodes.

Look at FEDforstudents.pdf (page 2)

What is a manufactured solution and what is it for :

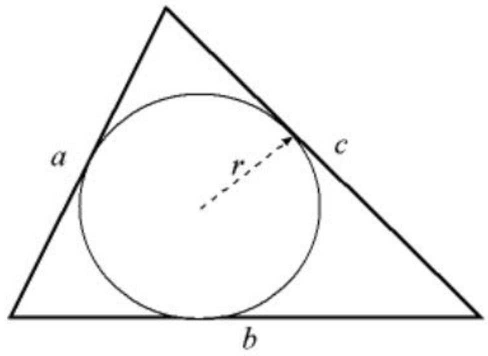
— choose a function $u_m(x, y)$.
— Define $g = u_m$ on ∂D.
— Compute $f = -\Delta u_m$ in D.
— solve Eq. 3 with the defined f and g and compute the error as the L2 norm of $u - u_m$

To do 2 - Validate

A method that is not validated is of no value.

Show that the P_1 method defined above is second order in space. Choose a manufactured solution and check that the result is coherent .

— case 1 : u_m = constant, or u_m linear function. Compute the L2 norm of the error which should be 0 within numerical accuracy (i.e. less than 10^{-13}).
— case 2 : u_m a polynomial of order greater than 2. Check that the L2 norm of the error which is a function of r (r = minimal radius of inscribed circles over all triangles), has the right behaviour, i.e. Log(E) versus Log(r) should have a slope of 2.



$$\textbf{Radius} = \frac{TriangularArea}{k} = \frac{\sqrt{k(k-a)(k-b)(k-c)}}{k}$$

$$\text{where } k = \frac{1}{2}(a+b+c)$$

FIGURE 2 – computation of the inscribed circle radius

To do 3 - Send us your scripts
Create a directory with your first name and second name : for example mkdir Juliette-Ryan.
In this directory, you will put all the scripts developed in **To do 1** and **To do 2** , the main script will be called MyLapSolver.m
Zip this directory and send it by mail to
halpern@math.univ-paris13.fr
ryan@math.univ-paris13.fr

Step 3 = Solving - Direct solvers - LU solver

To do :
Write a Lower Upper (LU) decomposition of $K = LU$ without permutation and 2 functions to solve $LUX = F$ to replace the backslash operator in your script developed in step 2
K is given in Matlab sparse format. Create L and U as banded matrices.

Step 4 = Begin your project

If your project needs a specific mesh, create it.

If not , generate the mesh for the following domains :
— the domain contained by an outer circle and an inner square
— the domain contained by an outer rectangle and an inner square
— the domain contained by an outer rectangle and an inner circle