Technische Universität Berlin **WS 2013/2014**
Fakultät II – Institut f. Mathematik
Instructor: Peschka Handed out: 17.12.
Tutor: Marschall Return during lecture: 08.01.

8. Assignment „Numerische Mathematik für Ingenieure II“

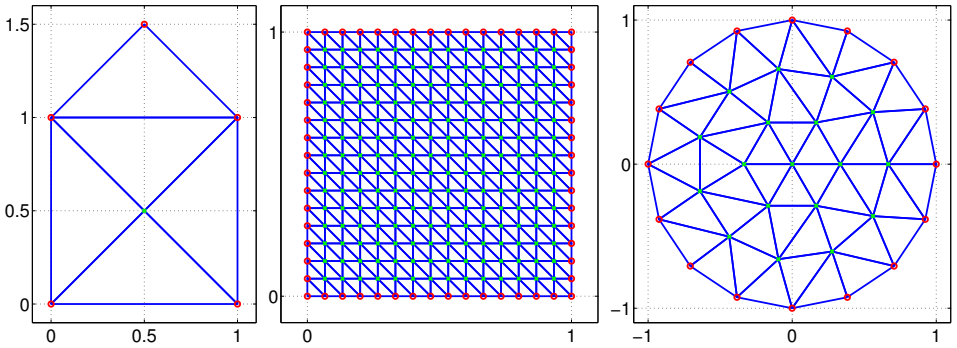
http://www.moses.tu-berlin.de/Mathematik/

Construction of finite elements – PART II

Decomposition: Assume we have a domain Ω with polygonal boundary. First we want to construct a decomposition of Ω in order to construct basis functions for the finite element method.

1. Programming exercise: Element generation in 2D **10 points**

Generate an admissible decomposition of $\Omega_h =$ *"Das Haus vom Nikolaus"* using 5 elements, so that the edges are the lines in the construction of the riddle.



examples: left) "Haus vom Nikolaus", middle) square with $N = 16$, right) disc with $N = 16$

Furthermore generate an admissible decomposition of the square $\Omega_s = (0,1)^2$ and a disc $\Omega_d = \{(x,y) \in \mathbb{R}^2 : x^2 + y^2 < 1\}$. For Ω_d you are supposed to approximate the boundary by a polygon. Illustrations are shown in the picture above, where red dots indicate boundary points and green crosses are interior points. Therefore write MATLAB functions

(a) `function [x,y,npoint,nelement,e2p,id]=generatehaus()`

(b) `function [x,y,npoint,nelement,e2p,id]=generatesquare(N)`

(c) `function [x,y,npoint,nelement,e2p,id]=generatedisc(N)`

which return the x, y-coordinates of points, the number of points, the number of elements (triangles), the element-to-point map `e2p` and a marker of boundary points `id` for $\Omega_h, \Omega_s, \Omega_d$. The marker is a vector of size `npoint` which is zero for interior points and one for a boundary point. Similar to the 1D construction, `e2p` is of size `nelement x 3` and contains the numbers of the vertices of an element, so that `e2p(k,1), e2p(k,2), e2p(k,3)` runs over points of the element `k` anticlockwise.

For a) you are supposed to do this by hand, whereas for b,c) use the MATLAB function `delaunay` which creates a Delaunay triangulation `e2p` using the convex hull of a given set of points. The boundary marker needs to be constructed by hand using logical expressions such as `id(x<eps)=1` for the box. For b) the parameter N specifies the number of points in each direction, whereas for c) N is the number of vertices on the boundary. Please make sure that for c) you distribute the number of inner points somewhat evenly[1] (as for example in the figure above). For b) you might use the MATLAB function `meshgrid`.

(d) Furthermore write a `function []=plotmesh(x,y,e2p,id)` which plots a mesh as above. The plot of the mesh can be done using the MATLAB command `triplot(e2p,x,y)`, whereas you are also supposed to plot boundary points (red circles) and inner points (green crosses). Export the plots to `haus.pdf`, `square.pdf` and `disc.pdf`.

(e) Familiarize yourself with the tool *triangle* by J.R. Shewchuk, which can be downloaded at `http://www.cs.cmu.edu/~quake/triangle.html`. Generate and plot your own triangulation for a domain of your choice[2] (see tutorial on ISIS 2 page).

[1] As $N \to \infty$ the length of all edges shall go to zero and angles should neither be small nor large/obtuse.
[2] Except for the example included in the triangle help.

2. Programming exercise: Computation of transformation **8 points**

We have the reference element (triangle)

$$\Omega^{\text{ref}} = \{(x, y) \in \mathbb{R}^2 : 0 < x < 1 - y \text{ and } y > 0\}$$

with the corner points $\mathbf{x}_1^{\text{ref}} = (0, 0)$, $\mathbf{x}_2^{\text{ref}} = (1, 0)$, $\mathbf{x}_3^{\text{ref}} = (0, 1)$ in anticlockwise sense. Assume the kth triangle consists of points $\mathbf{x}_i \in \mathbb{R}^2$ for $i = 1, 2, 3$ also in anticlockwise sense. For a given `k` the ith point should have the coordinates $x_i =$`x(e2p(k,i))` and $y_i =$`y(e2p(k,i))`.

(a) Compute the unique linear transformation $F : \Omega^{\text{ref}} \to \Omega_k$, which maps the triangle Ω^{r} to the kth element, where in particular $F(x_i^{\text{ref}}, y_i^{\text{ref}}) = (x_i, y_i)$ for $i = 1, 2, 3$.

(b) Compute the Jacobian matrix ∇F, defined as $(\nabla F)_{ij} = \partial_j F_i$, the determinant of the Jacobian matrix $\det(\nabla F)$, and the inverse of the Jacobian $(\nabla F)^{-1}$ as a function of x_i, y_i for $i = 1, 2, 3$.

(c) Write a `function [edet,dFinv]=generatetransformation(k,e2p,x,y)` which for given element `k=1..nelement` computes the determinant of the Jacobian `edet`=$\det(\nabla F)$ and the inverse matrix `dFinv`= $(\nabla F)^{-1}$ using your result from a,b).

(d) Write a `function check=checkorientations(e2p,nelement,x,y)` which checks the sign of the determinants for all elements and returns `check=true` if all have positive sign, and `check=false` otherwise. Verify that you get `check=true` for the meshes created in exercise 1. If `check=false` the points of an element are not in anticlockwise order.

3. Programming exercise: Computation of local matrices **8 points**

On the reference element Ω^{ref} we have the three affine linear basis functions defined by $\phi_i(x_j^{\text{ref}}, y_j^{\text{ref}}) = \delta_{ij}$. So they are $\phi_1(x, y) = 1 - x - y$, $\phi_2(x, y) = x$, $\phi_3(x, y) = y$.

(a) Compute the 3×3 matrices

$$M_{ij} = \int_{\Omega^{\text{ref}}} \phi_i(x, y)\phi_j(x, y)\,\mathrm{d}x\,\mathrm{d}y$$
$$S_{ij} = \int_{\Omega^{\text{ref}}} \nabla\phi_i(x, y) \cdot \nabla\phi_j(x, y)\,\mathrm{d}x\,\mathrm{d}y$$

(b) For a fixed k consider the map $F : \Omega^{\text{ref}} \to \Omega_k$ as defined in the previous exercise. Let $\bar{\phi}_i : \Omega_k \to \mathbb{R}$ defined by $\bar{\phi}_i\big(F(x, y)\big) = \phi_i(x, y)$. We want to compute

$$\bar{M}_{ij} = \int_{\Omega_k} \bar{\phi}_i(x, y)\bar{\phi}_j(x, y)\,\mathrm{d}x\,\mathrm{d}y$$
$$\bar{S}_{ij} = \int_{\Omega_k} \nabla\bar{\phi}_i(x, y) \cdot \nabla\bar{\phi}_j(x, y)\,\mathrm{d}x\,\mathrm{d}y$$

using integration-by-substitution. Therefore verify the following two equations

$$\bar{M}_{ij} = \det(\nabla F)M_{ij}$$
$$\bar{S}_{ij} = \frac{1}{2}\det(\nabla F)(G_{i1}G_{j1} + G_{i2}G_{j2})$$

where

$$G_{i\alpha} = \sum_{j=1}^{2} \frac{\partial\phi_i}{\partial x_j^{\text{ref}}} \frac{\partial x_j^{\text{ref}}}{\partial x_\alpha}$$

or in other words $\bar{S} = \frac{1}{2}\det(\nabla F)GG^\top$. In particular show that

$$\int_{\Omega_k} \mathrm{d}x\,\mathrm{d}y = \frac{1}{2}\det(\nabla F).$$

Why is the expression for S of no big help?

(c) Using the results from (a,b) write two functions `function mloc=localmass(edet)` and `function sloc=localstiff(edet,dFinv)` which for given value of `edet` and `dFinv` compute the element mass matrix $\bar{M}$ and stiffness matrix $\bar{S}$. It might be useful to write a separate function to compute $G \in \mathbb{R}^{3\times 2}$ and then set `sloc=1/2*edet*G*G'`.

4. Programming exercise: Construction of global matrix **10 points**

For the moment we will consider a problem with homogeneous Dirichlet boundary conditions on the whole boundary. Therefore let I be the indices of nodes, which lie on the boundary, i.e. $i \in I$ implies $(x_i, y_i) \in \partial\Omega$. Then we want to distinguish the following two types of functions

$$v(x,y) = \sum_{j=1}^{\texttt{npoint}} \alpha^j w_j(x,y), \qquad v_0(x,y) = \sum_{j=1, j\notin I}^{\texttt{npoint}} \alpha^j w_j(x,y)$$

and the corresponding discrete spaces V_h and V_h^0. In particular we have the relation $V_h^0 \subset V_h$. Obviously we have $\dim V_h =$ `npoint` and $\dim V_h^0 =$ `npoint-nboundary` $<$ `npoint`. With `nboundary` we denote the number of points on the boundary, which is the same as the number of elements in I.

Our variational problem reads

"Find a function $u_h \in V_h^0$ such that $a(u_h, v_h) = f^*(v_h)$ for all test-functions $v_h \in V_h^0$."

with bilinear form

$$a(u_h, v_h) = \int_\Omega \left(\nabla u_h \cdot \nabla v_h + c\, u_h v_h\right) \mathrm{d}\Omega$$

and $c \geq 0$ a given constant. For the given linear form

$$f^*(v_h) = \int_\Omega f(x,y) v_h(x,y)\, \mathrm{d}x\, \mathrm{d}y$$

assume that $f \in V_h$.

(a) Show that for $w_i \in V_h$

$$f^*(w_i) = \sum_{j=1}^{\texttt{npoint}} \hat{M}_{ij} f_j$$

where we have the global mass matrix

$$\hat{M}_{ij} = \langle w_i, w_j \rangle_{L^2(\Omega)} = \int_\Omega w_i(x,y) w_j(x,y) \mathrm{d}x\, \mathrm{d}y$$

for $f_j = f(x_j, y_j)$ and $1 \leq i, j \leq$ `npoint` .

(b) Let $\hat{A}_{ij} = a(w_j, w_i)$ for $w_i, w_j \in V_h$ instead of V_h^0. Show that the Galerkin equation is still equivalent to the variational form

"Find coefficients α^i such that $\sum_{j=1, j\notin I}^{\texttt{npoint}} \hat{A}_{ij}\alpha^j = f_i$ for all $i \notin I$."

For example consider the case $f(x) = 1$. Explain why in MATLAB we get the equivalent reduced system by

```
rhs = M*ones(npoint,1);
rhs = rhs(id==0);
Ar  = A(id==0,id==0);

u         = zeros(npoint,1);
u(id==0) = Ar \ rhs;
trisurf(e2p,x,y,u);
```

(c) Consider the code `elliptic2d.m` which uses your functions to construct the global matrix corresponding to the Galerkin equation. Explain how and why this works. Point out the differences to the 1D implementation.

(d) Solve the problem with $c = 0$ and $f = 1$ on Ω_s and Ω_d and plot the solution. For Ω_d compare with the exact solution on the disc (it is a polynomial of second degree).

(e) Modify `elliptic2d.m` so that you solve the problem with $c = 1$ and $f = 1$ on Ω_d and homogeneous Neumann boundary conditions (natural boundary conditions) and compare with exact solution. Why is the solution with Neumann conditions unique?

(f) Modify `elliptic2d.m` so that you solve the problem with $f = 0$ and inhomogeneous Dirichlet boundary conditions $u = x^2$ on the square by modification of f using $u(x,y) = u_0(x,y) + \bar{u}(x,y)$, where

$$u_0(x_i, y_i) = \begin{cases} x^2 & i \in I, \\ 0 & \text{otherwise.} \end{cases}$$

total sum: 36 points