Anisotropic Diffusion in curved geometry

JAD Univ. Nice/INRIA Sophia-Antipolis

July 16, 2009

Introduction

We consider the unsteady diffusion problem

$$\frac{\partial u}{\partial t} - \nabla \cdot K \cdot \nabla u = f \quad \text{in } \Omega$$
$$u = 0 \text{ on } \Gamma_D \forall t$$
$$\frac{\partial u}{\partial n} = 0 \text{ on } \Gamma_N \forall t$$

The conductivity, $K = k_{||}\tilde{K} + k_I I$ is a 3 by 3 tensor comprising two parts. The first part is the parallel conductivity given by $k_{||}\tilde{K} = k_{||} bb^T/|b|^2$, where $k_{||}$ is the parallel diffusion coefficient and b is a prescibed magnetic vector-field. The second component, $k_I I$, is the standard isotropic diffusion term. Generally, the diffusion is strongly anisotropic, with $k_{||} >> k_I$.

Triangulation

Definition

For a given set of points V, we call a *triangulation* a set of triangles T, such that:

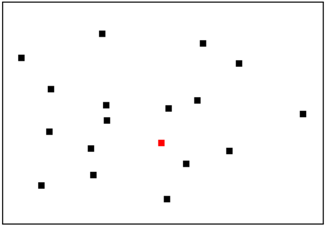
- all points of T constitute V
- none of the triangles in T overlap
- the union of all triangles of T is the convex hull of V
- T crosses V only in nodes.

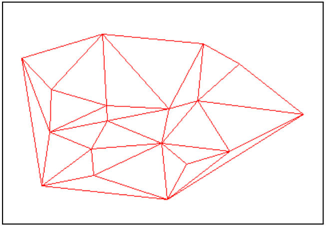
Definition

A *Delaunay triangulation* D of a vertex set V is a graph with the following property: if $u, v \in V$, then $uv \in D \Leftrightarrow \exists$ circle, that passes through u, v that doesn't contain inside any point from V(this is called the Delaunay property of the edge).

From this definition it's obvious that for a given set of points, the Delaunay triangulation is unique, as we can determine for each edge if it's present in the triangulation.

On the pictures below, we have a set of vertices and its Delaunay triangulation.



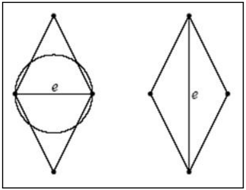


Flipping Algorithm

The Flipping Algorithm starts from any triangualtion of set of points V, and looks for an edge that isn't Delaunay. When it finds it, the edge is removed, creating a quad, and then inserted back as the other diagonal.

Definition

An edge e of triangulation is locally Delaunay if it has the Delaunay property with respect to just to the vertices of two triangles that contain e.



PSLG and Constrained Delaunay triangulation

Definition

A *Planar Straight Line Graph(PSLG)* is a collection of vertices and segments. Segments are edges whose endpoints are vertices in the PSLG, and whose presence in any mesh generated from the PSLG is enforced.

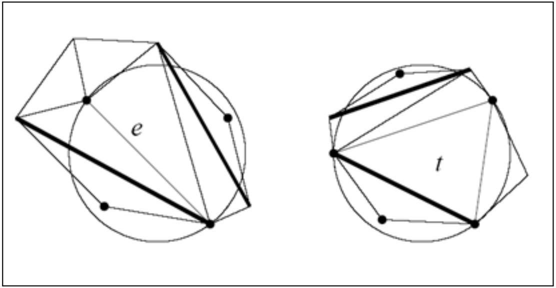
Definition

A *constrained Delaunay triangulation* of a PSLG is similar to a Delaunay triangulation, but each PSLG segment is present as a single edge in the triangulation.

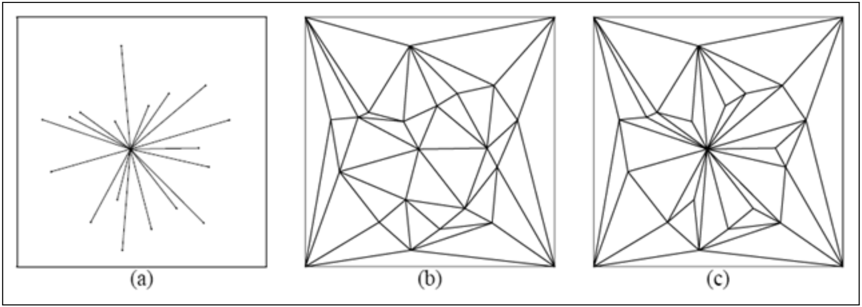
Definition

An edge or a triangle is constrained Delaunay if it doesn't cross/cover a segment of PSLG(except when an edge belongs to PSLG), and its circumcircle doesn't cover any point of PSLG that is visible from the middle of the triangle or the edge(the line joining the middle point and a point of PSLG doesn't cross a segment of PSLG).

On the picture below the edge e and the triangle t are constrained Delaunay. Although their circumcircles cover vertices of PSLG, they aren't visible through the segments.



On the next picture there's (a)–PSLG,(b)–DT,(c)–CDT. As we can see, some edges of CDT are constrained Delaunay but not Delaunay.



Definition

A *conforming Delaunay triangulation(CDT)* of a PSLG is a true Delaunay triangulation in which each PSLG segment may have been subdivided into several edges by insertion of additional vertices, called *Steiner points*. Steiner points are necessary to allow the segments to exist in the mesh while maintaining the Delaunay property. Steiner points are also inserted to meet constraints on the miniumum angle and maximum triangle area.

The latter is a triangulation that a Finite Element method can work with.

Domain generation, based on a list of points and normals

The program uses the second order Bezier splines to approximate the curved segments:

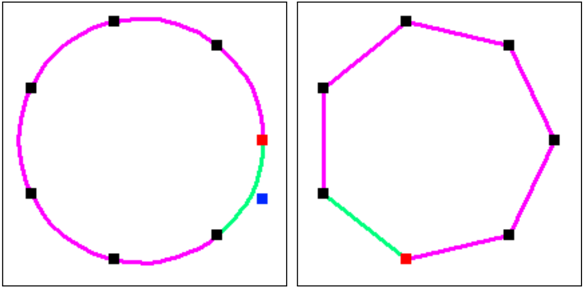
$$s(t) = t^2 p_1 + 2t(1-t)m + (1-t)^2 p_2$$

where p_1 and p_2 are the endpoints and m is the control point.
After little computation, we get the coordinates of control point from normals:

$$x = \frac{x_1 n_{1x} n_{2y} - x_2 n_{2x} n_{1y} + n_{1y} n_{2y}(y_1 - y_2)}{n_{1x} n_{2y} - n_{2x} n_{1y}}$$

$$y = \frac{y_1 n_{1y} n_{2x} - y_2 n_{2y} n_{1x} + n_{1x} n_{2y}(x_1 - y_2)}{n_{1y} n_{2x} - n_{2y} n_{1x}}$$

with Bezier splines, the boundary is C^1



Variational formulation for stationary problem

find $u \in V = \{v \in W_2^1 | v = 0 \text{ on } \Gamma_D\}$ such that

$$\int_\Omega \nabla u \cdot k \cdot \nabla v = \int_\Omega f \cdot v$$

Computing matrices on reference triangle

To compute the elements

$$\int_K \nabla N_i^K \cdot k(x,y) \cdot \nabla N_j^K and \int_K N_i^K N_j^K$$

move to the so-called reference element:

$$\hat{p_1} = (0,0), \quad \hat{p_1} = (1,0) \quad \hat{p_1} = (0,1)$$

The local nodal functions in the reference triangle for $\mathbf{P}_1$ are:

$$\hat{N}_1 = 1 - \xi - \eta, \quad \hat{N}_2 = \xi, \quad \hat{N}_3 = \eta$$

Let us now take the three vertices of a triangle K

$$p_1^K = (x_1, y_1), \quad p_2^K = (x_2, y_2), \quad p_3^K = (x_3, y_3)$$

The following transformation

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

maps the triangle $\hat{K}$ bijectively into K. Call this transformation F_K.

$$F_K(\hat{p}_i) = p_i^K, \quad i = 1, 2, 3$$

It is simple now to see that

$$N_i^K(x, y) = \hat{N}_i(F_K^{-1}(x, y))$$

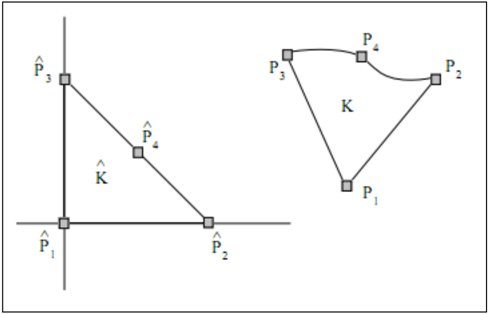
Since computing F_K^{-1} is straightforward from the explicit expression for F_K, this formula gives a simple way of evaluating the functions N_i^K. To evaluate the gradient of N_i^K we have to apply the chain rule:

$$B_K^T(\nabla\phi \circ F_K) = \hat{\nabla}(\phi \circ F_K)$$

B_K^T is the transposed of the matrix of the linear transformation F_K. Taking $\phi = N_i^K$ in this expression, we obtain:

$$\nabla N_i^K = B_K^{-T}\left((\hat{\nabla}\hat{N}_i) \circ F_K^{-1}\right)$$

Isoparametric elements



On the picture above, we have the reference triangle and a deformation of the image triangle.

Let us now call p_4^K to the midpoint of the segment that joins $\hat{p}_2$ and $\hat{p}_3$, that is $\hat{p}_4 = (\frac{1}{2}, \frac{1}{2})$.

Take a fourth point in the physical space, $p_4^K = (x_4, y_4)$ and compute its deviation from the midpoint of p_2^K and p_3^K

$$\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} - \begin{bmatrix} \frac{x_2+x_3}{2} \\ \frac{y_2+y_3}{2} \end{bmatrix}$$

Finally take the transformation $F_K : \hat{K} \to \mathbf{R}^2$ given by

$$F_K(\xi, \eta) = F_K^0(\xi, \eta) + 4\xi\eta \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}$$

$$B_K = DF(\xi, \eta) = B_K^0 + 4 \begin{bmatrix} \eta \\ \xi \end{bmatrix} \begin{bmatrix} \delta_x & \delta_y \end{bmatrix} = \begin{bmatrix} x_2 - x_1 + 4\eta\delta_x & x_3 - x_1 + 4\eta\delta_y \\ y_2 - y_1 + 4\xi\delta_x & y_3 - y_1 + 4\xi\delta_y \end{bmatrix}$$

When p_4^K is not too far from the midpoint of p_2^K and p_3^K, that is, when the deviation (δ_x, δ_y) is not too large, it is possible to prove that the image if $\hat{K}$ under this transformation $K = F_K(\hat{K})$ is mapped bijectively from the reference element and therefore we can construct an inverse to $F_K : \hat{K} \to K$.

Computing the local integrals for isoparametric triangles

$$N_i^K = \hat{N}_i \circ F_K^{-1}$$

Instead of integrating on K, we move to the reference domain:

$$\int_K N_i^K N_j^K = \int_{\hat{K}} |det B_K| \hat{N}_i \hat{N}_j$$

With this strategy, the integral is defined on a plain triangle and we just need to compute the non-constant determinant of

$$B_K = \begin{bmatrix} x_2 - x_1 + 4\eta\delta_x & x_3 - x_1 + 4\eta\delta_y \\ y_2 - y_1 + 4\xi\delta_x & y_3 - y_1 + 4\xi\delta_y \end{bmatrix}$$

on the chosen quadrature points.
Stiffness matrix:

$$\int_{\hat{K}} |det B_K| ((B_K^{-T} \nabla \hat{N}_i)^T \cdot k(x(\xi,\eta), y(\xi,\eta)) \cdot (B_K^{-T} \nabla \hat{N}_i)$$

Formulation of integrals over a triangular area

Since an affine transformation makes it possible to transform any triangle into a standard triangle T with coordinates $\{(0,0), (0,1), (1,0)\}$, we have to consider just the numerical integration on T. The integral of an arbitrary function f over the surface of a triangle T is given by:

$$I = \iint_T f(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_0^1 dx \int_0^{1-x} f(x,y)\,\mathrm{d}y = \int_0^1 dy \int_0^{1-y} f(x,y)\,\mathrm{d}x$$

Now we have to find the value of the integral by a quadrature formula:

$$I = \sum_{m=1}^{N} c_m f(x_m, y_m)$$

where c_m are the weights associated with specifice points (x_m, y_m) and N is the number of pivotal points related to the required precision.

The integral can be transformed into an integral over the surface of the square: $\{(u,v)| \quad 0 \leq u, v \leq 1\}$, by substitution:

$$x = u, y = (1-u)v$$

Then the determinant of the Jacobian and the differential area are:

$$\frac{\partial(x,y)}{\partial(u,v)} = \frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u} = (1)(1-u) - 0(-v) = 1-u$$

and

$$\mathrm{d}x\,\mathrm{d}y = \frac{\partial(x,y)}{\partial(u,v)}\,\mathrm{d}u\,\mathrm{d}v = (1-u)\,\mathrm{d}u\,\mathrm{d}v$$

$$I = \int_0^1 \int_0^{1-x} f(x,y)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1 \int_0^1 f(u,(1-u)v)(1-u)\,\mathrm{d}u\,\mathrm{d}v$$
$$= \int_{-1}^1 \int_{-1}^1 f(\frac{1+\xi}{2}, \frac{(1-\xi)(1+\eta)}{4})(\frac{1-\xi}{8})\,\mathrm{d}\xi\,\mathrm{d}\eta$$

Last equation represents an integral over the surface of a standard 2-square:
$\{(\xi,\eta)| -1 \leq \xi, \eta \leq -1\}$.

$$I = \int_0^1 \int_0^1 f(x(\xi,\eta), y(\xi,\eta))(\frac{1-\xi}{8})\,\mathrm{d}\xi\,\mathrm{d}\eta,$$

$$I = \sum_{i=1}^{n} \sum_{i=1}^{n} (\frac{1-\xi_i}{8}) w_i w_j f(x(\xi,\eta), y(\xi,\eta)),$$

where ξ_i, η_i are Gaussian points in the ξ, η directions, respectively, and w_i and w_j are the corresponding weights.

$$I = \sum_{k=1}^{N=n\times n} c_k f(x_k, y_k),$$

where c_k, x_k and y_k can be obtained from the relations:

$$c_k = \frac{1-\xi_i}{8} w_i w_j, \quad x_k = \frac{1+\xi_i}{2}, \quad y_k = \frac{(1-\xi_i)(1+\eta_i)}{4},$$
$$k, i, j = 1, 2, 3, \ldots, n.$$

Time discretization

There is little difficulty in extending the finite element idealization to situations that are time dependant. By putting

$$
\begin{aligned}
u_h &= \sum N_i a_i = \mathbf{Na} \\
N &= N(x,y,z) \quad a = a(t)
\end{aligned} \tag{1}
$$

for each element, then we get the following *matrix differential equation*:

$$\mathbf{C\dot{a}} + \mathbf{Ka} + \mathbf{f} = \mathbf{0} \tag{2}$$

$$\mathbf{a}(0) = \mathbf{a}_0 \tag{3}$$

in which all the matrices are assembled from element submatrices in the standard manner with **C** being the mass matrix and **K** being the stiffness matrix. To solve the problem we use the SS11 algorithm:

$$\mathbf{a_{n+1}} = \mathbf{a_n} + \mathbf{\Delta t}\alpha \tag{4}$$

$$\alpha = -(\mathbf{C} + \theta\delta t\mathbf{K})^{-1}(f + \mathbf{Ka_n}) \tag{5}$$

θ is chosen to be 0.5, which corresponds to Crank-Nicholson scheme.

Solver

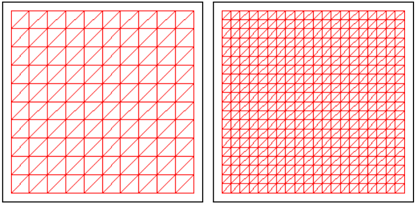
I've implemented a sparse direct solver that uses Cholesky decomposition for symmetric systems.

Following algorithms are used:

- Gibbs algorithm for finding a pseudo-peripheral node.
- Reverse Cuthill-McKee algorithm for finding a symmetrical reordering of rows and columns, that decreases the profile of the system.
- Cholesky decomposition itself, with matrix being stored as an envelope.

Validation

Take known solution to be $u = \sin(\pi x)\sin(\pi y)$ in $\Omega = [0,1]x[0,1]$. To estimate convergence rates we need two meshes with sizes h and $h/2$:



$$||u - u^h||_{L_2} \leq h^m ||u||_{W_2^1}$$

$$||u - u^{h/2}||_{L_2} \leq (\frac{h}{2})^m ||u||_{W_2^1}$$

From here we can find m

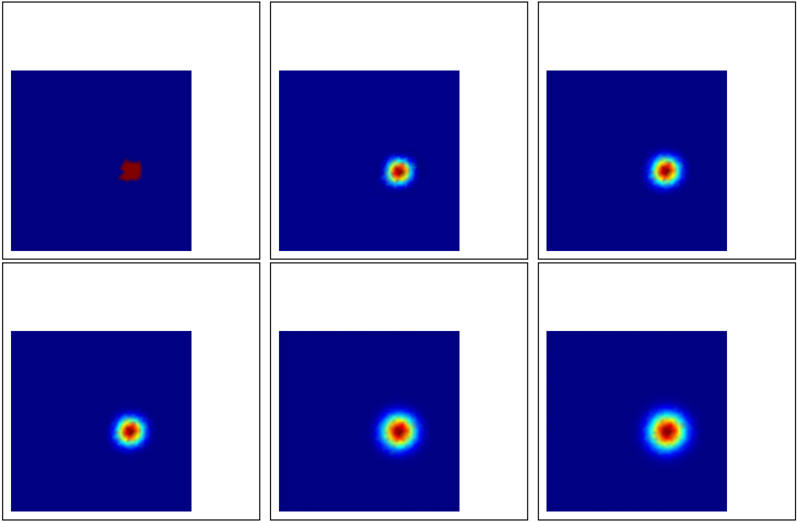
	$h = \frac{1}{10}$	$h = \frac{1}{20}$	rate m
linear	0.00414891	0.00104353	1.9913
quadratic	0.000101269	$1.19623 \cdot 10^{-5}$	3.0816

	$h = \frac{1}{20}$	$h = \frac{1}{40}$	rate my
linear	0.00104353	0.000261274	1.9978
quadratic	$1.19623 \cdot 10^{-5}$	$1.4637 \cdot 10^{-6}$	3.0310

From here we can see that estimated convergence rates match the theoretical ones. The program is working correctly.

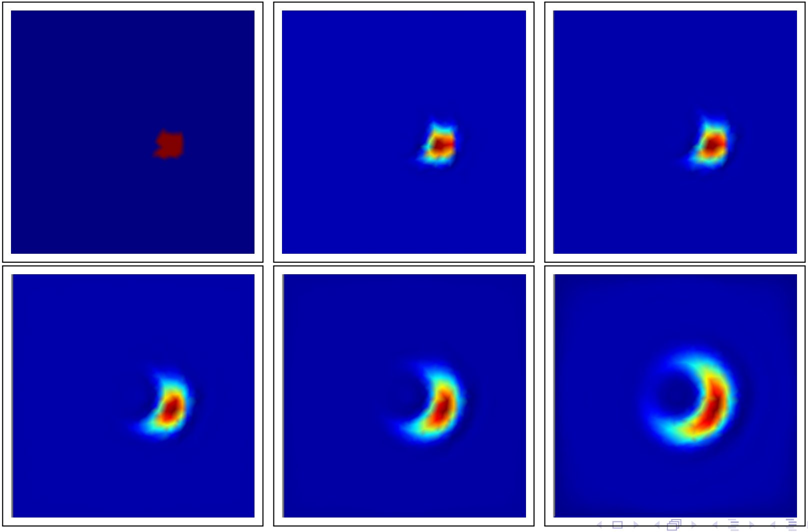
Time evolution example 1, P_1,1710 points, $t \in [0, 0.001]$

For this case consider $k = I, f = 1$



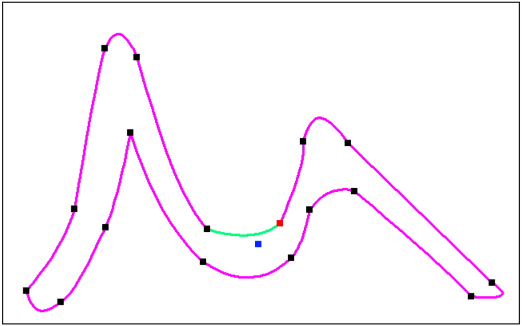
Time evolution example 2, P_1,1710 points, $t \in [0, 0.001]$

For this case consider k to have only the tangential component w.r.t the circles centered at (0.5,0.5).

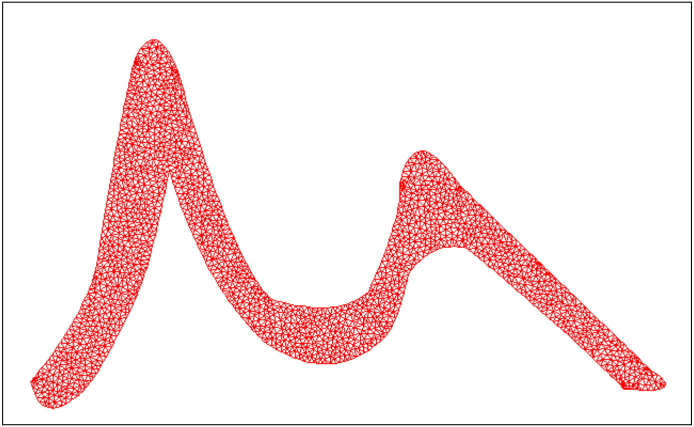


Isoparametric example

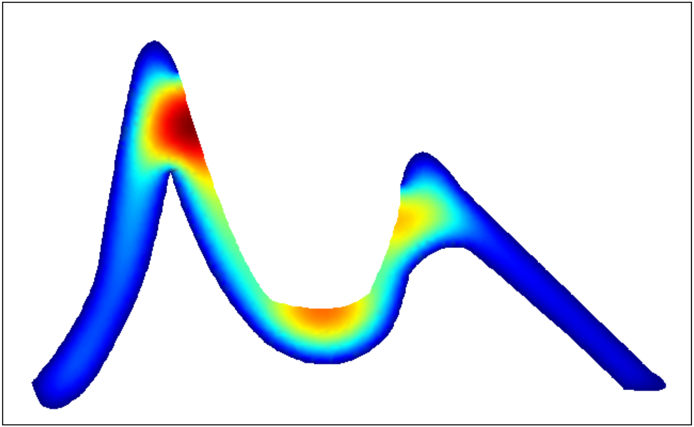
Here is an example of a domain we can model.



This is the triangulation of the domain.



This is how the solution looks like.



Conclusions

During my industrial training I have learned:

- some knowledge on domain triangulation
- some more advanced techniques in FEM
- typesetting in LaTeX

The program I wrote could use some improvements:

- more advanced sparse matrix handling
- try using iterative solvers
- solve platform independance issues