The instant finite element method

Sébastien Loisel

September 16, 2014

1 Introduction

This is how I do the 4th year projects/msc projects; this is the "computational half" of the project. There is also a "theory half". The idea is that all the code examples are done interactively in front of the students.

I normally do Section 2 in one hour and then spend several hours doing Section 3.

2 1d Example

We want to solve the differential equation

$$-u''(x) = f(x) \text{ for } x \in (a,b) \text{ with } u(a) = u(b) = 0. \tag{1}$$

This is 1d so we can do an exact solution:

$$u'(x) = -\int f(x)\,dx + C$$

$$u(x) = -\int\int f(x)\,dx\,dx + Cx + D. \tag{2}$$

If we take the concrete example $f(x) = x$, $a = 0$, $b = 1$, we get

$$u(x) = -\frac{1}{6}x^3 + Cx + D.$$

We can find C, D by $u(0) = 0$ and $u(1) = 0$ which gives

$$-\frac{1}{6}0^3 + C0 + D = 0$$

$$-\frac{1}{6}1^3 + C1 + D = 0$$

The solution is $D = 0$ and $C = \frac{1}{6}$ which gives

$$u(x) = -\frac{1}{6}(x^3 - x).$$

This is an exact solution. What if we could not compute the double integral (2)? We need a numerical solution.

2.1 1d piecewise linear functions

We let $x_0 < x_1 < \ldots < x_n$ be a grid of points and

$$\phi_j(x) = \max\left\{0, \min\left\{\frac{(x - x_{j-1})}{(x_j - x_{j-1})}, \frac{(x - x_{j+1})}{(x_j - x_{j+1})}\right\}\right\}. \tag{3}$$

See Fig. 1. We can approximate any function using these basis functions. For example,

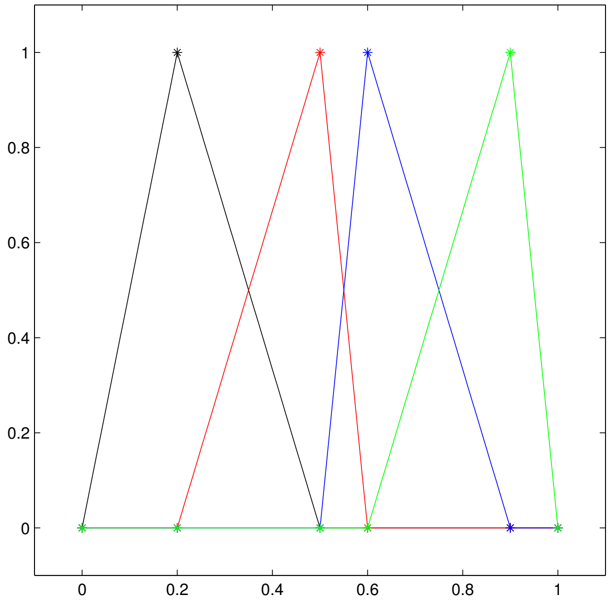


Figure 1: Piecewise linear basis functions $\phi_i(x)$.

here is $u(x) = 2\phi_1(x) + 3\phi_2(x) + 5\phi_3(x) - 1.5\phi_4(x)$, see Fig. 2. The choice of basis

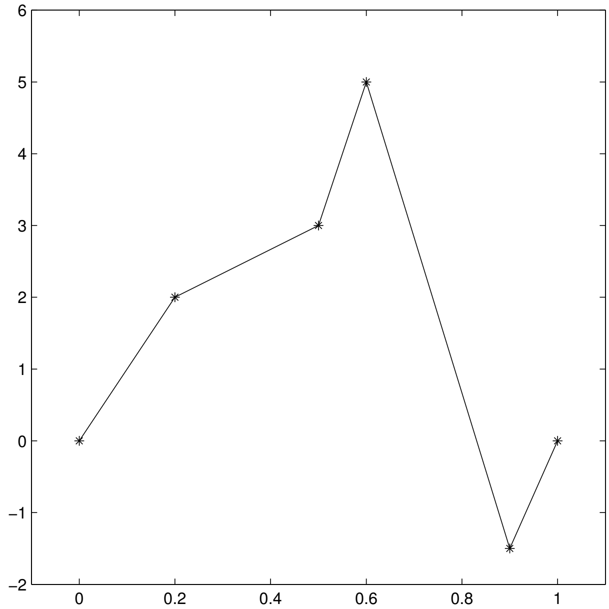


Figure 2: $u(x) = 2\phi_1(x) + 3\phi_2(x) + 5\phi_3(x) - 1.5\phi_4(x)$:

functions (3) is very convenient because

$$\phi_i(x_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

As a result:

Theorem 1. *The piecewise linear basis function that interpolates the points* (x_j, y_j) *is*

$$u(x) = \sum_{j=1}^{n} y_j \phi_j(x).$$

2.2 1d finite element solution

Now returning to (1), we can find a numerical solution as follows. First, we find the **weak formulation**. Let $\phi_i(x)$ be one of the basis functions. Then:

$$\int_a^b -u''(x)\phi_i(x)\,dx = \int_a^b f(x)\phi_i(x)\,dx.$$

We integrate by parts:

$$\int_a^b u'(x)\phi_i'(x)\,dx - u'(x)\phi_i(x)|_a^b = \int_a^b f(x)\phi_i(x)\,dx.$$

Note that $\phi_i(a) = \phi_i(b) = 0$.

$$\int_a^b u'(x)\phi_i'(x)\,dx = \int_a^b f(x)\phi_i(x)\,dx. \tag{4}$$

Equation (4) is the **weak form** of (1).

Next, we make the ansatz that $u(x) = \sum_{j=1}^{n} u_j \phi_j(x)$ and we obtain

$$\sum_{j=1}^{n} \overbrace{\int_a^b \phi_i'(x)\phi_j'(x)\,dx}^{a_{ij}}\, u_j = \overbrace{\int_a^b f(x)\phi_i(x)\,dx}^{F_i}.$$

In Matrix form, this is

$$Au = F.$$

In the next several pages, we show how this can be done in MAPLE.

> $n := 4;$

$$n := 4 \tag{1}$$

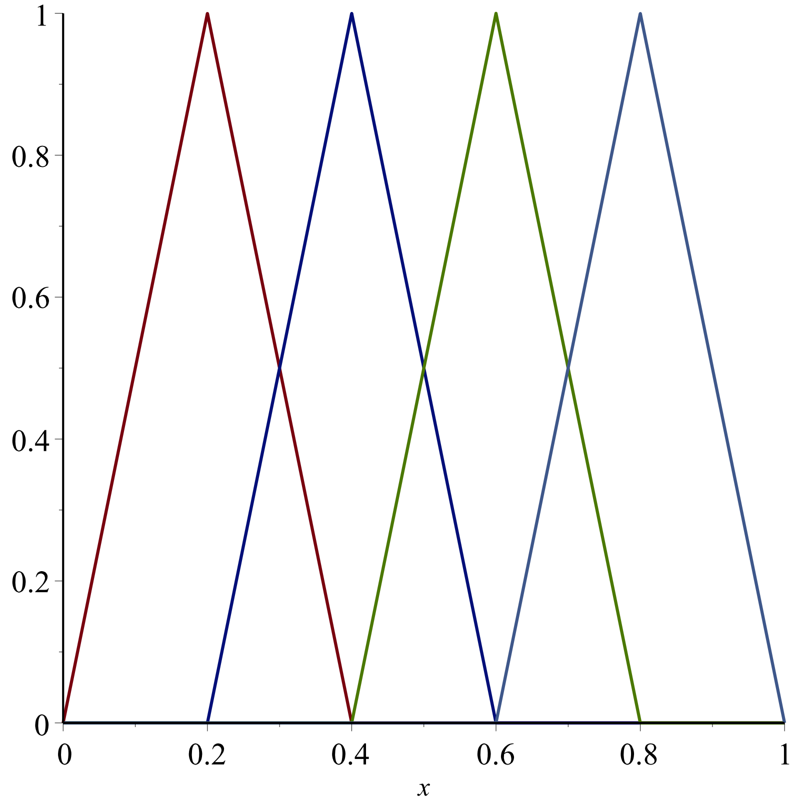
> $xs := \left['\frac{k}{n+1}' \$ k = 0 .. n+1 \right];$

$$xs := \left[0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right] \tag{2}$$

> $\text{phi} := \left['\mathit{convert}\left(\max\left(0, \min\left(\frac{(x - xs[j-1])}{xs[j] - xs[j-1]}, \frac{(x - xs[j+1])}{xs[j] - xs[j+1]} \right)\right), \mathit{piecewise}\right)' \$ j = 2 .. n + 1 \right]$

$$\phi := \left[\begin{cases} 0 & x \le 0 \\ 5\,x & x \le \frac{1}{5} \\ -5\,x + 2 & x \le \frac{2}{5} \\ 0 & \frac{2}{5} < x \end{cases}, \begin{cases} 0 & x \le \frac{1}{5} \\ 5\,x - 1 & x \le \frac{2}{5} \\ -5\,x + 3 & x \le \frac{3}{5} \\ 0 & \frac{3}{5} < x \end{cases}, \begin{cases} 0 & x \le \frac{2}{5} \\ 5\,x - 2 & x \le \frac{3}{5} \\ -5\,x + 4 & x \le \frac{4}{5} \\ 0 & \frac{4}{5} < x \end{cases}, \begin{cases} 0 & x \le \frac{3}{5} \\ 5\,x - 3 & x \le \frac{4}{5} \\ -5\,x + 5 & x \le 1 \\ 0 & 1 < x \end{cases} \right] \tag{3}$$

> $\mathit{plot}(\text{phi}, x = 0 .. 1)$



> $A := Matrix(['['int(diff(\text{phi}[i], x) \cdot diff(\text{phi}[j], x), x = 0 ..1)'\$j = 1 ..n]'\$i = 1 ..n])$

$$A := \begin{bmatrix} 10 & -5 & 0 & 0 \\ -5 & 10 & -5 & 0 \\ 0 & -5 & 10 & -5 \\ 0 & 0 & -5 & 10 \end{bmatrix} \tag{4}$$

> $f := x;$

$$f := x \tag{5}$$

> $F := eval(Vector(['int(f \cdot \text{phi}[j], x = 0 ..1)'\$j = 1 ..n]));$

(6)

$$F := \begin{bmatrix} \frac{1}{25} \\ \frac{2}{25} \\ \frac{3}{25} \\ \frac{4}{25} \end{bmatrix} \tag{6}$$

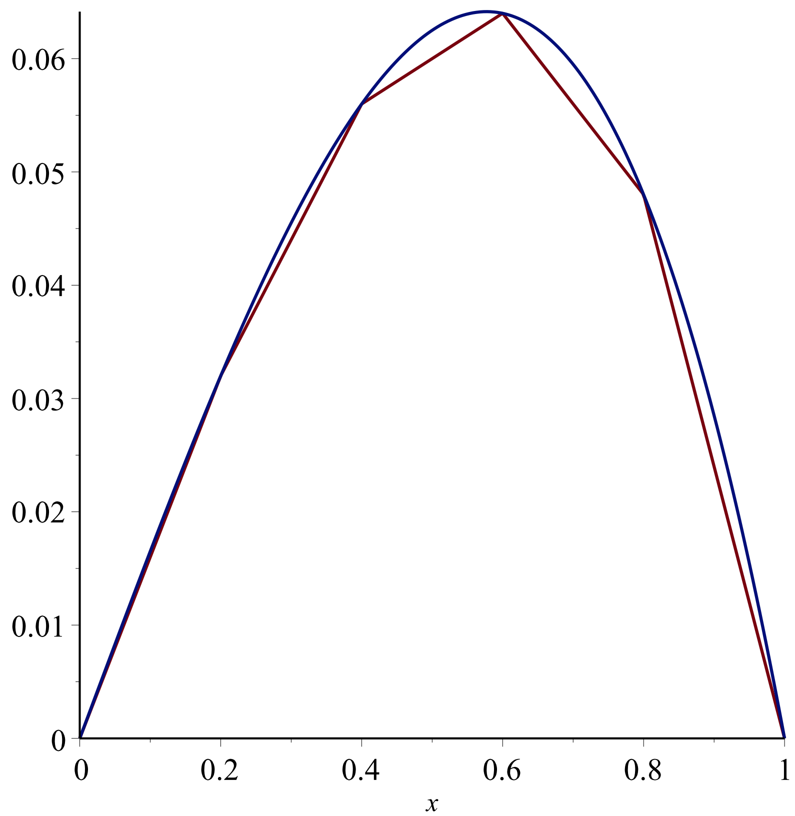
> $us := A^{-1}.F$

$$us := \begin{bmatrix} \frac{4}{125} \\ \frac{7}{125} \\ \frac{8}{125} \\ \frac{6}{125} \end{bmatrix} \tag{7}$$

> $u := simplify(sum('us[j]\cdot \mathrm{phi}[j]', j = 1..n))$

$$u := \begin{cases} 0 & x \le 0 \\ \frac{4}{25}\,x & x \le \frac{1}{5} \\ \frac{3}{25}\,x + \frac{1}{125} & x \le \frac{2}{5} \\ \frac{1}{25}\,x + \frac{1}{25} & x \le \frac{3}{5} \\ -\frac{2}{25}\,x + \frac{14}{125} & x \le \frac{4}{5} \\ -\frac{6}{25}\,x + \frac{6}{25} & x \le 1 \\ 0 & 1 < x \end{cases} \tag{8}$$

> $plot\left(\left[u, -\frac{1}{6}\left(x^3 - x\right)\right], x = 0..1\right)$



0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
x

>

3 2d general case

We are now solving

$$-u_{xx} - u_{yy} = f \text{ in } \Omega \text{ and } u = 0 \text{ on } \partial\Omega. \tag{5}$$

This time we cannot find an exact solution (except for the solutions that Joseph Fourier found in 1806).

3.1 A 2d domain and its triangular mesh

We can define a 2d domain as a union of triangles, see Fig. 3; this will be our **example mesh** going forward.

This triangle has vertices

$$x_1 = [1, 1] \tag{6}$$
$$x_2 = [4, 2] \tag{7}$$
$$x_3 = [2, 2.5] \tag{8}$$
$$x_4 = [1.5, 5] \tag{9}$$

It consists of two triangles $T_1 = (x_1, x_2, x_3)$ and $T_2 = (x_1, x_3, x_4)$. This can all be written in "matrix form" using the following MATLAB code:

```
T = [1 2 3
     1 3 4];
V = [1 1
     4 2
     2 2.5
     1.5 5];
trimesh(T,V(:,1),V(:,2))
```

We can also find edges (x_i, x_j). For example, the triangle $T_1 = (x_1, x_2, x_3)$ has three edges $e_1 = (x_1, x_2)$, $e_2 = (x_2, x_3)$, $e_3 = (x_3, x_1)$. By convention, we always list the vertex with the smaller index first, so in fact $e_3 = (x_1, x_3)$. We can do this in MATLAB:

```
edges = sort([T(:,1) T(:,2)
              T(:,2) T(:,3)
              T(:,3) T(:,1)],2)
```

It is important to be able to distinguish edges that are on the boundary $\partial\Omega$ and edges that are inside Ω. Fortunately this is easy: all the edges that are repeated twice are inside, and all the edges that appear only once are on the boundary. Right now, we have:

```
edges =

     1     2
     1     3
     1     3
     1     4
     2     3
     3     4
```

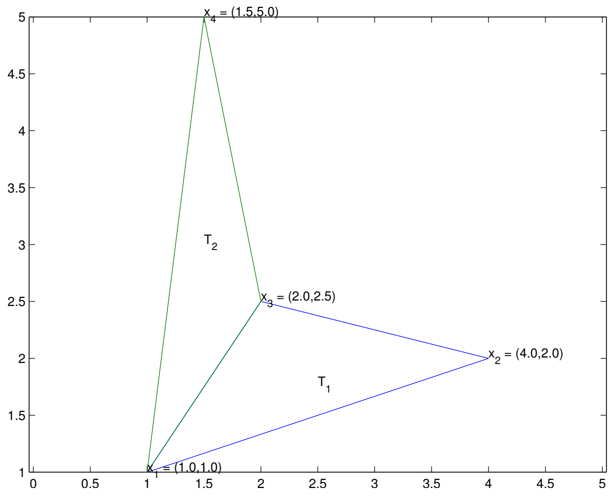


Figure 3: A triangular mesh for a domain.

We see that the edges (x_1, x_2), (x_1, x_4), (x_2, x_3) and (x_3, x_4) are on the boundary, while the edge (x_1, x_3) is in the interior; this is consistent with Fig. 3.

We can count the number of times that each edge is repeated as follows:

```
[edges,~,index] = unique(edges,'rows');
counts = accumarray(index,1);
```

We can then find the boundary edges, boundary vertices and interior vertices as follows:

```
boundaryEdges = edges(counts==1,:);
boundaryVertices = unique(boundaryEdges(:));
I = setdiff(1:n,boundaryVertices); % interiorVertices
```

For our example mesh there are no interior vertices, but this will be more useful later.

3.2 Piecewise linear basis functions

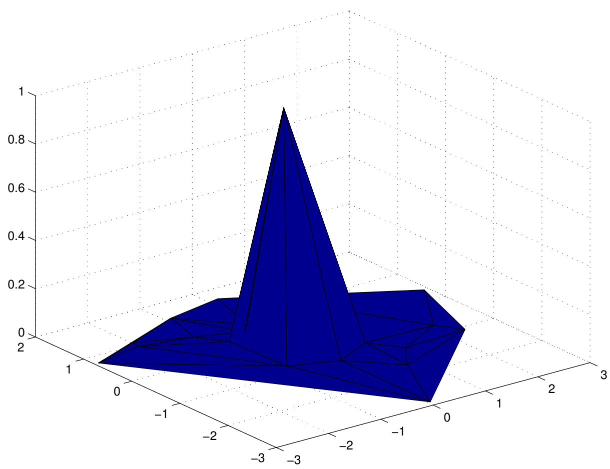


Figure 4: A 2d piecewise linear basis function.

We want piecewise linear basis functions for our **example mesh**, similar to the "hat functions" in the 1d case, see Fig. 4. Using these piecewise linear basis functions, we find that

$$\phi_i(x_j, y_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

As a result:

Theorem 2. *The piecewise linear basis function that interpolates $u(x_j) = z_j$ is*

$$u(x, y) = \sum_{j=1}^{n} z_j \phi_j(x, y).$$

3.3 The reference triangle

Consider the reference triangle $\hat{T}$, consisting of reference vertices $\hat{x}_1 = (0, 0)$, $\hat{x}_2 = (1, 0)$, $\hat{x}_3 = (0, 1)$.

Theorem 3. *The reference basis functions are $\hat{\phi}_1(x, y) = 1 - x - y$; $\hat{\phi}_2(x, y) = x$ and $\hat{\phi}_3(x, y) = y$.*

Proof. We have to check the interpolation conditions (10).

$$\hat{\phi}_1(\hat{x}_1) = \hat{\phi}_1(0, 0) = 1 - 0 - 0 = 1$$

$$\hat{\phi}_1(\hat{x}_2) = \hat{\phi}_1(1, 0) = 1 - 1 - 0 = 0$$
$$\hat{\phi}_1(\hat{x}_3) = \hat{\phi}_1(0, 1) = 1 - 0 - 1 = 0$$
$$\text{etc...}$$
$$\hat{\phi}_3(\hat{x}_3) = \hat{\phi}_2(0, 1) = y = 1$$

as required. □

3.4 Mapping to an arbitrary triangle

Let $T = (x_1, x_2, x_3)$ be an arbitrary triangle. How do we map $\hat{T}$ to T?

Theorem 4. *Let*

$$B = [x_2 - x_1, x_3 - x_1] \text{ and } c = x_1 \tag{11}$$

and define $G(\hat{x}) = B\hat{x} + c$. Then G maps $\hat{T}$ onto T. Furthermore, the basis functions ϕ_i are defined by

$$\phi_i(x) = \hat{\phi}_i(G^{-1}(x)) = \hat{\phi}_i(B^{-1}(x - c)). \tag{12}$$

This is how it fits together (see Fig. 5):

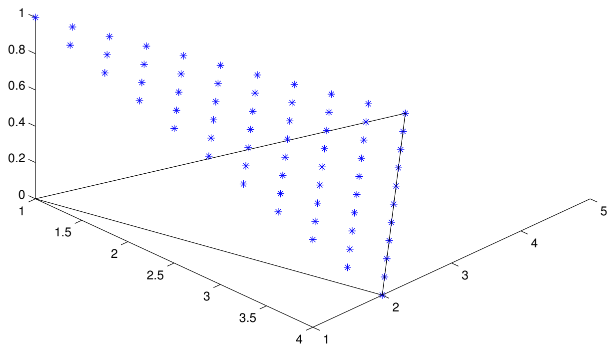


Figure 5: A general triangle and one of the basis functions.

```
% Consider this general triangle:
x1 = [1;1]
x2 = [4;2]
x3 = [2;5]
% The mapping from the reference triangle:
B = [x2-x1 x3-x1]
c = x1
% Let's say we want to plot points on the basis function phi1:
hold on;
for xh = 0:0.1:1
    for yh = 0:0.1:1-xh
        x = B*[xh;yh]+c
        plot3(x(1),x(2),1-xh-yh,'*');
    end
end
% Let's also plot our general triangle.
V = [x1 x2 x3 x1];
plot3(V(1,:),V(2,:),[0 0 0 0],'k-');
hold off;
camorbit(45,-45)
```

3.5 Finite elements in 2d

We now find the variational form of (5). We multiply by one of the test functions $\phi_i(x, y)$ and integrate

$$\int_\Omega (-u_{xx}(x, y) - u_{yy}(x, y))\phi_i(x, y) = \int_\Omega f(x, y)\phi_i(x, y). \tag{13}$$

Note that these are double integrals (surface integrals) over the domain Ω. Let's focus on the term $\int_\Omega -u_{yy}(x, y)\phi_i(x, y)$ and let's "parametrize" the domain Ω. For example, if Ω were the unit disc $x^2 + y^2 < 1$ we would have that

$$\overbrace{-\sqrt{1-x^2}}^{y_{\min}(x)} < y < \overbrace{\sqrt{1-x^2}}^{y_{\max}(x)}.$$

When the domain Ω is not a disc we still assume that Ω is defined by

$$y_{\min}(x) < y < y_{\max}(x).$$

In that case, the double integral $\int_\Omega$ is $\int_{x=x_{\min}}^{x_{\max}} \int_{y=y_{\min}(x)}^{y_{\max}(x)}$. Thus,

$$\begin{aligned}\int_\Omega (-u_{yy}(x,y))\phi_i(x,y) &= \int_{x=x_{\min}}^{x_{\max}} \int_{y=y_{\min}(x)}^{y_{\max}(x)} (-u_{yy}(x,y))\phi_i(x,y)\; dy\; dx \\ &= \int_{x=x_{\min}}^{x_{\max}} \left(\int_{y=y_{\min}(x)}^{y_{\max}(x)} u_y(x,y)\phi_{iy}(x,y)\; dy - u_y(x,y)\phi_i(x,y)\big|_{y=y_{\min}(x)}^{y_{\max}(x)} \right) dx,\end{aligned}$$

where we have integrated by parts. Since $\phi_i = 0$ on the boundary and $y = y_{\min}(x)$ and $y = y_{\max}(x)$ are on the boundary, the boundary term disappears and

$$\int_\Omega (-u_{yy}(x,y))\phi_i(x,y) = \int_\Omega u_y(x,y)\phi_{iy}(x,y).$$

Similarly for the derivatives in x, we find

$$\int_\Omega (-u_{xx}(x,y))\phi_i(x,y) = \int_\Omega u_x(x,y)\phi_{ix}(x,y).$$

Summing produces the **weak formulation**

$$\int_\Omega u_x\phi_{ix} + u_y\phi_{iy} = \int_\Omega f\phi_i$$

Making the ansatz $u = \sum_{j=1}^n u_j\phi_j(x,y)$ leads to the finite element problem

$$Au = F \text{ where } A_{ij} = \int_\Omega \phi_{ix}\phi_{jx} + \phi_{iy}\phi_{jy} \text{ and } F_i = \int_\Omega f\phi_i.$$

The **stiffness matrix** can be written more concisely using the gradient notation:

$$A_{ij} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \text{ where } \nabla v = \begin{bmatrix} v_x \\ v_y \end{bmatrix}.$$

3.6 The discrete gradient

If we restrict a function $\phi_i(x)$ to a given triangle T_k then $\phi_i(x)|_{T_k}$ is a linear polynomial and hence its gradient $\nabla\phi_i(x)|_{T_k}$ is constant. Thus, $\nabla\phi_i(x)$ is a piecewise constant function. This allows us to define "discrete derivative matrices". The entries of the matrix D_q are

$$(D_q)_{ki} = \frac{\partial}{\partial x_q}\phi_i|_{T_k}.$$

How do we compute the discrete derivative matrices? Recall that on triangle T_k, $\phi_i(x) = \hat{\phi}_p(G^{-1}(x))$ where $p \in \{1,2,3\}$ and $\hat{\phi}_p$ is one of the reference basis functions. Applying the chain rule, we find that

$$\nabla\phi_i = B^{-T}\nabla\hat{\phi}_p.$$

However, instead of trying to figure out which p we need as a function of i and k, it is better to go "the other way around". Given the triangle T_k and the reference vertex $p \in \{1,2,3\}$, the corresponding physical vertex is `i = T(k,p)`. This is the MATLAB we need to compute the discrete derivatives:

```
% The gradient of the reference basis functions
grad = [-1 1 0
        -1 0 1];
% The matrices B
B11 = V(T(:,2),1) - V(T(:,1),1);      B12 = V(T(:,2),2) - V(T(:,1),2);
B21 = V(T(:,3),1) - V(T(:,1),1);      B22 = V(T(:,3),2) - V(T(:,1),2);
% The determinants
detB = B11.*B22 - B12.*B21;
% The matrix inverse transposes, call them C = transpose(inv(B)):
C11 =  B22./detB;                     C12 = -B21./detB;
C21 = -B12./detB;                     C22 =  B11./detB;
m = size(T,1);
n = size(V,1);
% We use sparse matrices for efficiency because there are lots of zeros.
D1 = sparse(m,n);
D2 = sparse(m,n);
for p=1:3
    D1 = D1 + sparse(1:size(T,1),T(:,p),C11*grad(1,p) + C12*grad(2,p),m,n);
    D2 = D2 + sparse(1:size(T,1),T(:,p),C21*grad(1,p) + C22*grad(2,p),m,n);
end
```

The preceding code applied to our mesh produces

```
>> full(D1)

ans =

   -0.1429    0.4286   -0.2857         0
   -1.0769         0    1.2308   -0.1538

>> full(D2)

ans =

   -0.5714   -0.2857    0.8571         0
    0.1538         0   -0.4615    0.3077
```

Each row represents one of two triangles, and each column represents one of four vertices, and there are two derivatives matrices D_1 and D_2 corresponding to the two partial derivatives $\partial/\partial x_1$ and $\partial/\partial x_2$. For example,

$$\frac{\partial}{\partial x_1}\phi_3|_{T_2} = 1.2308.$$

3.7 Computing the stiffness matrix

To compute the entries of the stiffness matrix, we have to compute $\int_\Omega \nabla\phi_i \cdot \nabla\phi_j$. However, using the fact that $\Omega = \cup_k T_k$ we can rewrite the integral as

$$\int_\Omega \nabla\phi_i \cdot \nabla\phi_j \; dx \; dy = \sum_k \int_{T_k} \nabla\phi_i \cdot \nabla\phi_j \; dx \; dy$$

Since the $\nabla\phi_i$ are constant on each T_k we obtain

$$\int_\Omega \nabla\phi_i \cdot \nabla\phi_j = \sum_k \nabla\phi_i|_{T_k} \cdot \nabla\phi_j|_{T_k} \int_{T_k} dx\, dy = \sum_k ((D_1)_{ki}(D_1)_{kj} + (D_2)_{ki}(D_2)_{kj})|T_k|,$$

where $|T_k|$ denotes the surface area of the triangle T_k. This surface area is $\frac{1}{2}|\det B_k|$. Letting $W = \frac{1}{2}\operatorname{diag}(|\det B_1|, \ldots, |\det B_m|)$ allows us to write the stiffness matrix in matrix notation:

$$A = D_1^T W D_1 + D_2^T W D_2.$$

In MATLAB, we get:

```
W = spdiags(0.5*abs(detB),0,m,m);
A = D1'*W*D1+D2'*W*D2;
```

This produces the following stiffness matrix in our example:

```
>> full(A)

ans =

    2.5302    0.1786   -3.0549    0.3462
    0.1786    0.4643   -0.6429         0
   -3.0549   -0.6429    4.2363   -0.5385
    0.3462         0   -0.5385    0.1923
```

3.8 The mass matrix

The term $F_i = \int_\Omega f\phi_i$ is difficult to evaluate exactly because $f(x)$ could be any function. We approximate it by numerical integration (the trapezoidal rule for a triangle). On the reference triangle, one has:

$$\int_{\hat{T}} \hat{\phi}_p \hat{f}\, d\hat{x} \approx \frac{1}{6}\left(\hat{\phi}_p(0,0)\hat{f}(0,0) + \hat{\phi}_p(1,0)\hat{f}(1,0) + \hat{\phi}_p(0,1)\hat{f}(0,1)\right) = \frac{1}{6}\hat{f}(\hat{x}_p).$$

Doing a change of variable, with $dx = |\det B_k| d\hat{x}$ produces

$$F_i = \int_\Omega \phi_i f\, dx \approx \overbrace{f(x_i)}^{f_i} \frac{1}{6} \sum_{k \text{ s.t. } x_i \in T_k} |\det B_k|$$

In matrix form:

$$F = Mf \text{ where } M_{ij} = \begin{cases} \frac{1}{6}\sum_{k \text{ s.t. } x_i \in T_k} |\det B_k| & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

The diagonal matrix M is called the **mass matrix**. The MATLAB to do this is:

```
M = accumarray(T(:,1),abs(detB)./6,[n 1]) + ...
    accumarray(T(:,2),abs(detB)./6,[n 1]) + ...
    accumarray(T(:,3),abs(detB)./6,[n 1]);
M = spdiags(M,0,n,n);
```

```
>> full(M)

ans =

    1.1250         0         0         0
         0    0.5833         0         0
         0         0    1.1250         0
         0         0         0    0.5417
```

3.9 Putting it all together

First, we generate a more interesting mesh:

```
rng(0);
V = randn(100,2);
T = delaunay(V);
```

We then assemble the stiffness matrix and the mass matrix as usual, compute the interior vertices and so on. We now pick a forcing, we use $f(x) = 1$ because it's easy. From this forcing, we compute the vector F restricted to the interior vertices, and the stiffness matrix $A0$ restricted to the interior vertices, then solve the PDE:

```
A0 = A(I,I);
F = M(I,:)*ones(n,1);
u0 = A0\F;
u = zeros(n,1);
u(I,1) = u0;
patch('Faces',T,'Vertices',V,'FaceVertexCData',u,'FaceColor','interp');
```

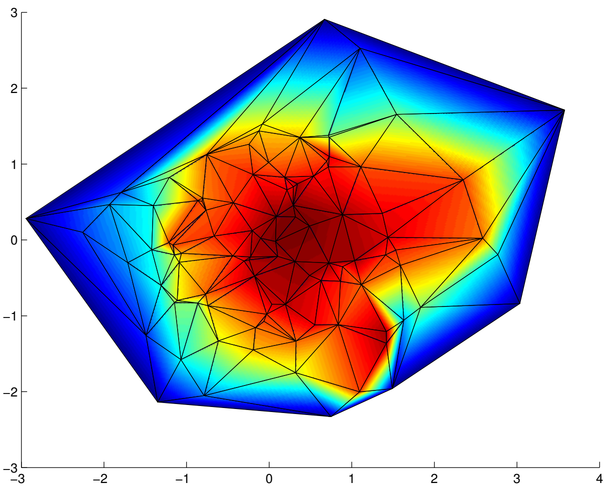


Figure 6: The solution of the heat equation by the Finite Element Method.

The output is in Fig. 6.