Finite Elements

In the last chapter we learned about the Laplacian, and the part it plays in the Poisson and heat equations that we can use for smoothing. But how do we get this Laplacian implemented on a computer?

We have already met the *finite difference* discretization method when it came to computing derivatives on regular grids, such as for Poisson surface reconstruction, or for averaging in 1D. For two-dimensional meshes, finite differences are not as useful. In this chapter we will learn the basics of the *finite element method* (FEM), a way to discretize partial differential equations on polylines and triangle meshes.

This will only be a superficial introduction to the finite element method. To learn a bit more about FEM, its mathematics, and its backgrounds, I highly recommend the standard book by Braess [1].

Function spaces

In the finite difference philosophy, we approximated a continuous function by sampling it at a variety of discrete grid points, thus turning the infinite-dimensional space of all functions into a finite space of n numbers at grid vertices.

The finite element approach is different – we approximate an infinite-dimensional function space with a finite-dimensional function space.

Definition 1: Continuous function space

The space of continuous functions $f : \Omega \to \mathbb{R}$ that can be differentiated continuously k times is called $C^k(\Omega)$.

The space of continuous functions on Ω is called $C^0(\Omega)$.

The space of smooth (infinitely differentiable) functions on Ω is called $C^\infty(\Omega)$.

One of the most important properties of $C^k(\Omega)$ is that it is a vector space.

Theorem 1: Function spaces are vector spaces

The space $C^k(\Omega)$ is a vector space over the real numbers $\mathbb{R}$.

This can be proven by defining addition and scalar multiplication as you would think,

$$(f+g)(x) = f(x) + g(x) \qquad \forall f, g \in C^k(\Omega)\,, \tag{1}$$

$$(\alpha f)(x) = \alpha f(x) \qquad \forall f \in C^k(\Omega)\,, \tag{2}$$

Additions of C^k functions and scalar multiples of them are still k times differentiable, so C^k is a vector space. ■

"Something to think about" 1: Linear maps

Vector spaces are the domain of linear algebra, and the main tool of linear algebra is the linear map. What is an example of a linear map on $C^k(\Omega)$?

As it turns out $C^k(\Omega)$ is actually not that easy to approximate on surface meshes if $k > 0$. We will have to try to approximate a slightly different space.

Definition 2: Continuous a.e. function space

A function $f : \Omega \to \mathbb{R}$ is said to have a property *almost everywhere* if the set of points $X \subseteq \Omega$ where the property does not hold has zero volume, i.e.,

$$\int_X 1 \, dx = 0 \,. \tag{3}$$

The space of functions that are $C^k(\Omega)$ almost everywhere (and, if applicable, $C^{k-1}(\Omega)$ everywhere) is called $C^k_{\text{ae}}(\Omega)$.[1]

Figure 1 illustrates the properties of C^k, as well as the differences between C^k and C^k_{ae}.

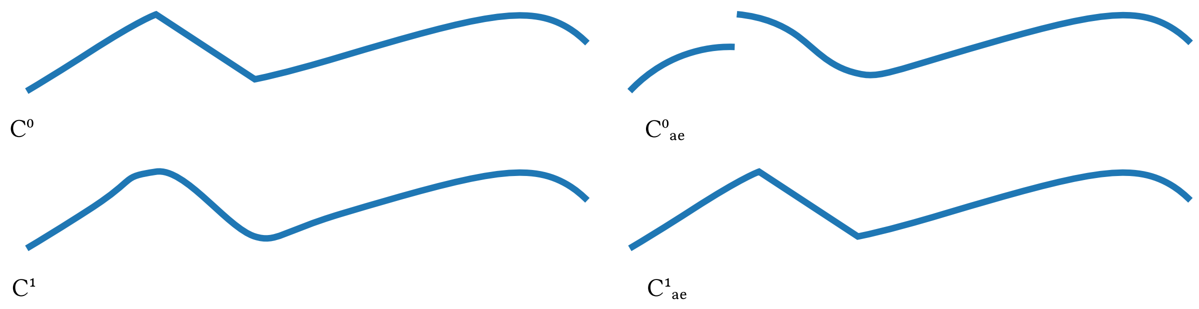

Figure 1: C^k functions are k times continuously differentiable. C^k_{ae} functions are k times continuously differentiable almost everywhere, and $k-1$ times continuously differentiable everywhere.

Finding the right finite-dimensional subspace

The space C^k_{ae} is still infinite-dimensional. In linear algebra terms, there is no *finite basis* for it. The finite element discretizes functions on Ω by picking a particular finite-dimensional subspace of C^k_{ae} that can be represented on a computer. We do this by dividing the surface Ω into finitely many *elements*. These are the data structures we already know:

- one-dimensional polylines, where the elements are the straight line segments between vertices; and
- two-dimensional triangle meshes, where the elements are the flat triangular faces.

These elements are now used to define finite-dimensional function spaces.

[1]This symbol is nonstandard. Do not expect to find it elsewhere, and do not use the symbol C^k_{ae} outside this course. Be sure to specify "C^k almost everywhere."

Definition 3: Discrete function spaces

Let F be the set of elements of the polyline or triangle mesh Ω. (These are straight line segments on a polyline, and flat triangles on a triangle mesh.)

Then the space $P_0(\Omega)$ is defined as the set of functions that are constant on each element $f \in F$:

$$P_0(\Omega) = \left\{u : \Omega \to \mathbb{R} \,|\, u|_f \;\text{ constant } \forall f \in F\right\}. \tag{4}$$

$P_0(\Omega)$ is also called the space of piecewise constant functions.

Then the space $P_1(\Omega)$ is defined as the set of continuous functions that are linear on each element $f \in F$:

$$P_1(\Omega) = \left\{u \in C^0(\Omega) \,|\, u|_f \;\text{ linear } \forall f \in F\right\}. \tag{5}$$

$P_1(\Omega)$ is also called the space of piecewise linear functions.[2]

Figure 2 shows examples of these finite element function spaces P_0, P_1.

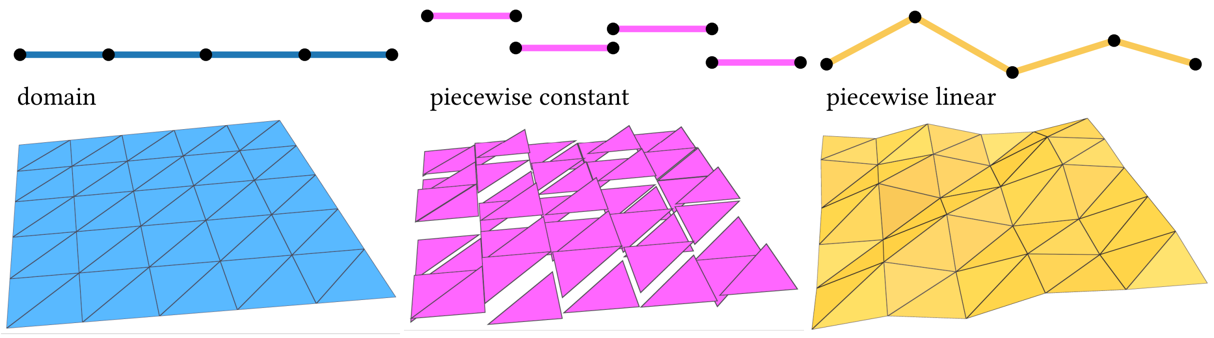

Figure 2: Piecewise continuous and piecewise linear function spaces both on 1D polylines as well as on 2D triangle meshes.

These spaces P_0 and P_1 are finite-dimensional: the number of degrees of freedom corresponds to the number of faces (P_0) or vertices (P_1). They are also naturally a subspace of their respective C^k_{ae} space:

$$P_k \subseteq C^k_{\text{ae}} \qquad k = 1, 2 \tag{6}$$

"Something to think about" 2: Are the P_k vector spaces?

Try to prove that the P_k really are vector spaces by showing they are closed under addition and scalar multiplication.

The piecewise constant space P_0 and the piecewise linear space P_1 are the workhorse of the finite element method. In this chapter we will learn how to use P_1 to discretize the Poisson equation. The functions in P_1 are sometimes also called *hat dunctions*, because when all vertices are set to 0 except one, the function looks like a little hat.

Discretizing the weak form

Let us return to the weak form of the Poisson equation from last chapter. We look for $u \in C^1_{\text{ae}}$ such that

[2]That is what the space is called, but be careful: We actually mean the space of continuous everywhere, and piecewise linear functions.

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in C^1_{\mathrm{ae}}(\Omega) \,. \tag{7}$$

Here the reason for our use of $C^1_{\mathrm{ae}}(\Omega)$ should become clear: it makes it immediately possible to just substitute the function space for $P_1(\Omega)$. Since $P_1(\Omega)$ is a fully-fledged subspace of $C^1_{\mathrm{ae}}(\Omega)$, the equation remains almost exactly the same with only the spaces changing. We now look for $u \in P_1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f \, v \, dx \qquad \forall v \in P_1(\Omega) \,. \tag{8}$$

Since (8) is linear in v, if we have a basis of the vector space $P_1(\Omega)$, we can replace the test $\forall v \in P_1(\Omega)$ with simply testing for all basis functions. Since $P_1(\Omega)$ is finite-dimensional, we know it has a finite basis. Let us call that basis $\eta_1, ..., \eta_n$. Then:

$$\int_\Omega \nabla u \cdot \nabla \eta_i \, dx = \int_\Omega f \, \eta_i \, dx \qquad i = 1, ..., n \,. \tag{9}$$

By that same notion, once we have a basis we can write $u \in P_1(\Omega)$ using this basis:

$$u(x) = \sum_{i=1}^n u_i \eta_i(x) \,, \tag{10}$$

where $u_1, ..., u_n \in \mathbb{R}$ are the scalar coefficients. This allows us to transform (9) into an equation entirely in the basis functions:

$$\sum_{j=1}^n u_j \int_\Omega \nabla \eta_j \cdot \nabla \eta_i \, dx = \sum_{j=1}^n f_j \int_\Omega \eta_j \, \eta_i \, dx \qquad i = 1, ..., n \,, \tag{11}$$

assuming $f(x) = \sum_{i=1}^n f_i \eta_i(x)$ in a similar way.

(11) is now a simple linear system that can be written in matrix form:

$$\begin{aligned} & \boldsymbol{L}\boldsymbol{u} = \boldsymbol{M}\boldsymbol{f} \,, \\ & L_{ij} = \int_\Omega \nabla \eta_i \cdot \nabla \eta_j \, dx, \quad M_{ij} = \int_\Omega \eta_i \eta_j \, dx \qquad \text{matrices} \\ & \boldsymbol{u} = \begin{pmatrix} u_1 \\ \cdots \\ u_n \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} f_1 \\ \cdots \\ f_n \end{pmatrix} . \end{aligned} \tag{12}$$

We know how to invert matrices, so it's straightforward to solve (12) for $\boldsymbol{u}$. This is the core of our FEM approach.

"Something to think about" 3: Boundary conditions

Is (12) really solvable just like that? On a related note, which boundary condition did we just discretize? How do we discretize another one?

The basis functions in detail

We used a basis $\eta_1, ..., \eta_n$ just now to write the weak Poisson equation in matrix form. But what are these basis functions? The function φ_i is 1 at the vertex i, it is 0 at all other vertices, and linear on each face. Figure 3 shows an example for both polylines and triangle meshes.

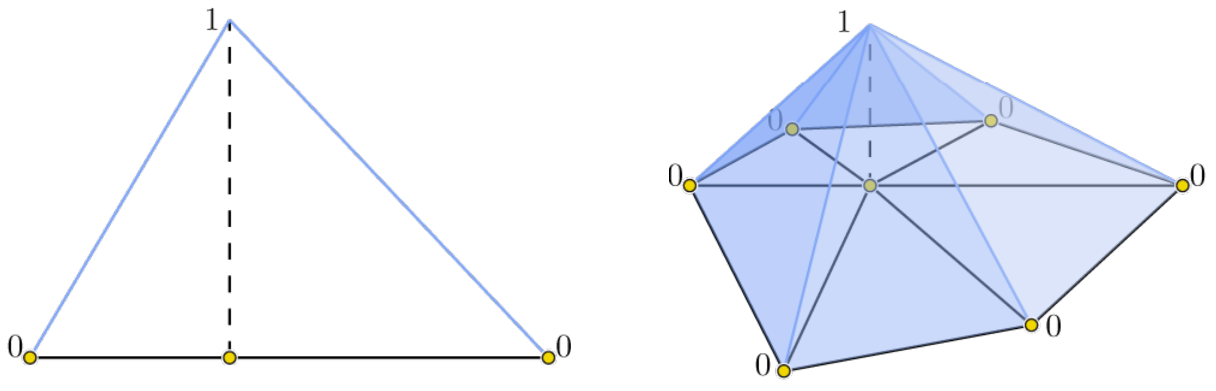

Figure 3: Hat functions on a polyline *(left)* and a triangle mesh *(right)*. Hat functions are 1 at their vertex, 0 at all other vertices, and piecewise linear on all faces.

Figure 3 also makes it clear why the functions in $P_1(\Omega)$ are called the *hat functions*. They can be written explicitly as follows. For polylines, let $v_1, \ldots, v_n \in V$ be the vertices of the polyline. Then

$$\varphi_i(x) = \begin{cases} 1 & x = v_i \\ \frac{\operatorname{length}(x, v_j)}{\operatorname{length}(v_i, v_j)} & x \in \text{line segment } (v_i, v_j) \\ 0 & \text{else} \end{cases}, \qquad (13)$$

where $\operatorname{length}(a, b)$ is the length of the line segment spanned by a and b.

For triangle meshes, let $v_1, \ldots, v_n \in V$ be the vertices of the mesh. Then

$$\varphi_i(x) = \begin{cases} 1 & x = v_i \\ \frac{\operatorname{area}(x, v_j, v_k)}{\operatorname{area}(v_i, v_j, v_k)} & x \in \text{triangle } (v_i, v_j, v_k) \\ 0 & \text{else} \end{cases}, \qquad (14)$$

where $\operatorname{area}(a, b, c)$ is the area of the triangle spanned by a, b and c.

Using these basis functions, we can now compute the entries of the matrices L and M from (12).

The cotangent Laplacian

We will compute the entries for the triangle mesh Laplacian here (try the polyline Laplacian as an exercise). Since the φ_i are linear functions, their gradients are constant vectors. The gradient vectors are shown in Figure 4.

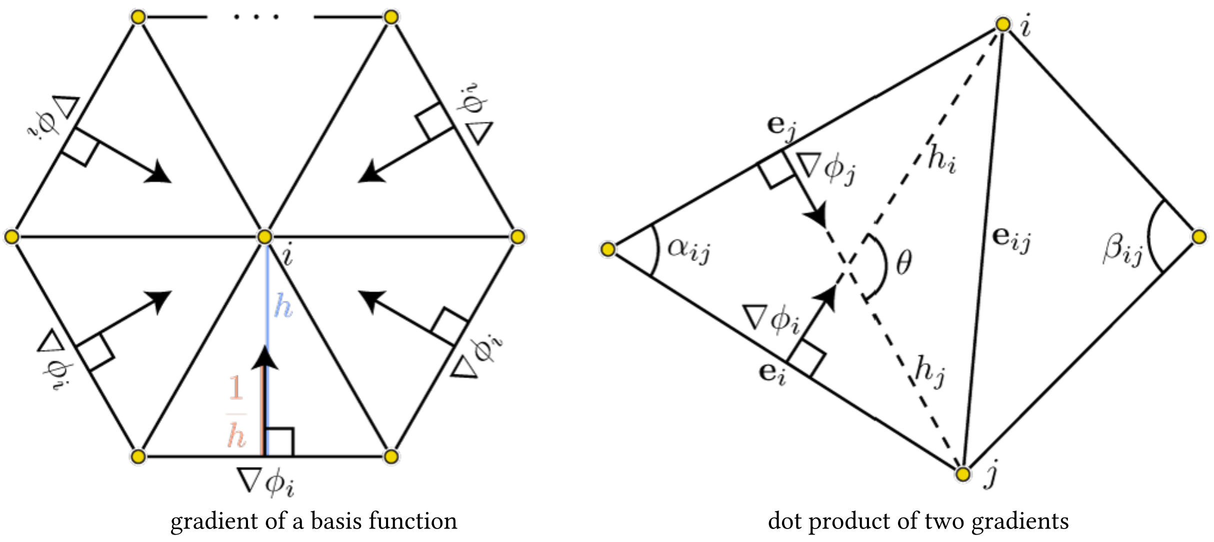

Figure 4: The gradient of a single basis function *(left)*, and the dot product of two gradient functions *(right)*.

We can see from Figure 4 that the gradient of φ_i in a triangle is orthogonal to the edge opposite i. Since the function increases linearly from 0 to 1, this means the magnitude of the gradient must be exactly one divided by the *altitude* h of the triangle, the orthogonal distance between a vertex and its opposing edge, which can be describe in terms of the edge length $\|\boldsymbol{e}_i\|$ and triangle area A:

$$\|\nabla\varphi_i\| = \frac{1}{h} = \frac{\|\boldsymbol{e}_i\|}{2A} \tag{15}$$

The dot product of two vectors $\boldsymbol{v}$, $\boldsymbol{w}$ can be described by their magnitudes and the cosine of the angle θ between them: $\boldsymbol{v}\cdot\boldsymbol{w} = \|v\|\|w\|\cos\theta$. Thus,

$$\nabla\varphi_i \cdot \nabla\varphi_j = \frac{\|\boldsymbol{e}_i\|\|\boldsymbol{e}_j\|}{4A^2}\cos\theta\,. \tag{16}$$

Using basic trigonometry and the right side of Figure 4, $\cos\theta = -\cos\alpha_{ij}$. Since the altitudes intersect their base edges at a right angle, we can use the definition of sin to get $\sin\alpha_{ij} = \frac{h_j}{\|\boldsymbol{e}_i\|}$. This gives us

$$\nabla\varphi_i \cdot \nabla\varphi_j = -\frac{h_j\|\boldsymbol{e}_j\|}{4A^2}\frac{\cos\alpha_{ij}}{\sin\alpha_{ij}}\,. \tag{17}$$

By the definition of the altitude, $h_j = \frac{2A}{\|\boldsymbol{e}_j\|}$, and by the definition of the cotangent, $\frac{\cos\theta}{\sin\theta} = \cot\theta$. This gives us

$$\nabla\varphi_i \cdot \nabla\varphi_j = -\frac{1}{2A}\cot\alpha_{ij}\,. \tag{18}$$

Integrating this (constant) expression over the triangle means multiplying it by the area. So the contribution of a single triangle of area A to the matrix entry L_{ij} is

$$-\frac{1}{2}\cot\alpha_{ij}\,. \tag{19}$$

This is why the matrix L is sometimes called the *cotangent matrix*. The entries of the cotangent matrix are:

$$L_{ij} = \begin{cases} -\frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right) & i \neq j,\, \alpha_{ij} \text{ and } \beta_{ij} \text{ angles opposite the edge } (i,j) \\ -\sum_{k \text{ neighbors of } i} L_{ik} & i = j \\ 0 & \text{else} \end{cases} . \quad (20)$$

For boundary edges, β_{ij} does not exist, so we set it to zero.

The mass matrix

To compute M_{ij} we should really compute the integral $\int_\Omega \eta_i \eta_j \, dx$. We will not do this: Computing this integral leads to a matrix that has entries on the off-diagonal, and this will make the matrix M more difficult to invert. For many applications, we can get away with simply *lumping* the mass matrix. This means summing all contributions of the matrix for each row, and just putting them on the diagonal. For triangle meshes, this is (do the polyline version as an exercise):

$$M_{ij} = \begin{cases} \frac{1}{3}\sum_{\text{triangle } t \text{ contains } i} \operatorname{area}(t) & i = j \\ 0 & i \neq j \end{cases} . \quad (21)$$

We have now computed the matrices from (12), and can solve the Poisson equation on a computer.

“Something to think about” 4: Heat equation

$\boldsymbol{L}\boldsymbol{u} = \boldsymbol{M}\boldsymbol{f}$ discretizes the Poisson equation. How would you discretize the *heat equation* $\frac{\partial u}{\partial x} = -\rho \Delta u$?

Discretize the equation in time using finite differences, and then use the matrices L and M. You have a choice here: either forward in time, or backward in time.

Discretizing the energy form

While our discretization from the last section was derived for the weak form of the Poisson equation, we can use it seamlessly for the energy form as well. Recall the energy form:

$$E(u) = \frac{1}{2}\int_\Omega \|\nabla u\|^2 \, dx \, . \quad (22)$$

We can substitute our definition of u in the finite element space, $u(x) = \sum_{i=1}^n u_i \eta_i(x)$, directly into (22):

$$E(u) = \frac{1}{2}\sum_{ij} u_i u_j \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \, dx \, . \quad (23)$$

Using the finite element matrix L and the vector of degrees of freedom $\boldsymbol{u}$, the energy form is

$$E(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{L}\boldsymbol{u} \, , \quad (24)$$

which can be solved with a variety of black-box quadratic programming routines-

Acknowledgements

The sketches in Figure 3 and Figure 4 were originally created by Alec Jacobson for his geometry processing course.

Bibliography

[1] Dietrich Braess. 2007. *Finite Elements -- Theory, Fast Solvers, and Applications in Solid Mechanics* (3rd ed.).