A gentle introduction to the Finite Element Method

Francisco–Javier Sayas

2008

An introduction

If you haven't been hiding under a stone during your studies of engineering, mathematics or physics, it is very likely that you have already heard about the Finite Element Method. Maybe you even know some theoretical and practical aspects and have played a bit with some FEM software package. What you are going to find here is a detailed and mathematically biased introduction to several aspects of the Finite Element Method. This is not however a course on the Analysis of the method. It is just a demonstration of how it works, written as applied mathematicians usually write it. There is going to be mathematics involved, but not lists of theorems and proofs. We are also going from the most particular cases towards useful generalizations, from example to theory.

An aspect where this course differs from most of the many introductory books on finite elements is the fact that I am going to begin directly with the two–dimensional case. I've just sketched the one dimensional case in an appendix. Many people think that the one–dimensional case is a better way of introducing the method, but I have an inner feeling that the method losses richness in that very simple situation, so I prefer going directly to the plane.

The course is divided into five lessons and is thought to be read in that order. We cover the following subjects (but not in this order):

- triangular finite elements,
- finite elements on parallelograms and quadrilaterals,,
- adaptation to curved boundaries (isoparametric finite elements),
- three dimensional finite elements,
- assembly of the finite element method,
- some special techniques such as static condensation or mass lumping,
- eigenvalues of the associated matrices,
- approximation of evolution problems (heat and wave equations).

It is going to be one hundred pages with many figures and many ideas repeated over and over, so that you can read it with ease. These notes have evolved during the decade I have been teaching finite elements to mixed audiences of mathematicians, physicists and engineers. The tone is definitely colloquial. I could just claim that these are my classnotes and that's what I'm like[1]. There's much more than that. First, I believe in doing your best at being entertaining when teaching. At least that's what I try. Behind that there is a deeper philosophical point: take your work (and your life) seriously but, please, don't take yourself too seriously.

I also believe that people should be duly introduced when they meet. All this naming old time mathematicians and scientists only by their last names looks to me too much like the Army. Or worse, high school![2] I think you have already been properly introduced to the great Leonhard Euler, David Hilbert, Carl Friedrich Gauss, Pierre Simon Laplace and George Green. If you haven't so far, consider it done here. This is not about history. It's just good manners. Do you see what I mean by being colloquial?

Anyway, this is not about having fun[3], but since we are at it, let us try to have a good time **while learning**. If you take your time to read these notes with care and try the exercises at the end of each lesson, I can assure that you will have made a significant step in your scientific *persona*. Enjoy!

[1]To the very common comment *every person has his/her ways*, the best answer I've heard is *Oh, God, no! We have good manners for that.*

[2]In my high school, boys were called by their last names. I was Sayas all over. On the other hand, girls were called by their first names.

[3]Unfortunately too many professional mathematicians advocate fun or beauty as their main motivations to do their job. It is so much better to have a scientific vocation than this aristocratic detachment from work...

Lesson 1

Linear triangular elements

1 The model problem

All along this course we will be working with a simple model boundary value problem, which will allow us to put the emphasis on the numerical method rather than on the intricacies of the problem itself. For some of the exercises and in forthcoming lessons we will complicate things a little bit.

In this initial section there is going to be a lot of new stuff. Take your time to read it carefully, because we will be using this material during the entire course.

1.1 The physical domain

The first thing we have to describe is the geometry (the physical setting of the problem). You have a sketch of it in Figure 1.1.

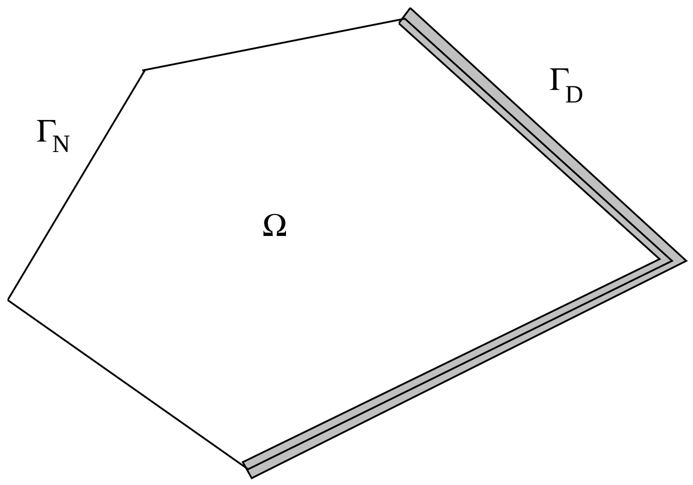


Figure 1.1: The domain Ω and the Dirichlet and Neumann boundaries

We are thus given a polygon in the plane $\mathbb{R}^2$. We call this polygon Ω. Its boundary is a closed polygonal curve Γ. (There is not much difference if we suppose that there is one or more holes inside Ω, in which case the boundary is composed by more than one polygonal curve).

The boundary of the polygon, Γ. is divided into two parts, that cover the whole of Γ and do not overlap:

- the Dirichlet boundary Γ_D,
- the Neumann boundary Γ_N.

You can think in more mechanical terms as follows: the Dirichlet boundary is where displacements are given as data; the Neumann boundary is where normal stresses are given as data.

Each of these two parts is composed by full sides of the polygon. This is not much of a restriction if you admit the angle of 180 degrees as separating two sides, that is, if you want to divide a side of the boundary into parts belonging to Γ_D and Γ_N, you just have to consider that the side is composed of several smaller sides with a connecting angle of 180 degrees.

1.2 The problem, written in strong form

In the domain we will have an elliptic partial differential equation of second order and on the boundary we will impose conditions on the solution: boundary conditions or boundary values. Just to unify notations (you may be used to different ways of writing this), we will always write the Laplace operator, or Laplacian, as follows

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

By the way, sometimes it will be more convenient to call the space variables (x_1, x_2) rather than (x, y), so expect mixed notations.

The boundary value problem is then

$$\left[\begin{array}{ll} -\Delta u + c\,u = f, & \text{in } \Omega, \\ u = g_0, & \text{on } \Gamma_D, \\ \partial_n u = g_1, & \text{on } \Gamma_N. \end{array}\right.$$

There are new many things here, so let's go step by step:

- The unknown is a (scalar valued) function u defined on the domain Ω.
- c is a non–negative constant value. In principle we will consider two values $c = 1$ and $c = 0$. The constant c is put there to make clear two different terms when we go on to see the numerical approximation of the problem. By the way, this equation is usually called a **reaction–diffusion equation**. The diffusion term is given by $-\Delta u$ and the reaction term, when $c > 0$, is $c\,u$.
- f is a given function on Ω. It corresponds to source terms in the equation. It can be considered as a surface density of forces.

- There are two functions g_0 and g_1 given on the two different parts of the boundary. They will play very different roles in our formulation. As a general rule, we will demand that g_0 is a continuous function, whereas g_1 will be allowed to be discontinuous.

- The symbol ∂_n denotes the exterior normal derivative, that is,

$$\partial_n u = \nabla u \cdot \mathbf{n},$$

  where $\mathbf{n}$ is the unit normal vector on points of Γ pointing always outwards and ∇u is, obviously, the gradient of u.

We are not going to bother about regularity issues here. If you see a derivative, admit that it exists and go on. We will reach a point where everything is correctly formulated. And that moment we will make hypotheses more precise. If you are a mathematician and are already getting nervous, calm down and believe that I know what I'm talking about. Being extra rigorous is not what is important at this precise time and place.

1.3 Green's Theorem

The approach to solve this problem above with the Finite Element Method is based upon writing it in a completely different form, which is sometimes called **weak or variational form**. At the beginning it can look confusing to see all this if you are not used to advanced mathematics in continuum mechanics or physics. We are just going to show here how the formulation is obtained and what it looks like at the end. You might be already bored in search of matrices and something more tangible! Don't rush! If you get familiarized with formulations and with the notations mathematicians given to frame the finite element method, many doors will be open to you in terms of being able to read a large body of literature that will be closed to you if you stick to what you already know.

The most important theorem in this process or reformulating the problem is Green's Theorem, one of the most popular results of Vector Calculus. Sometimes it is also called Green's First Formula (there's a popular second one and a less known third one). The theorem states that

$$\int_\Omega (\Delta u)\, v + \int_\Omega \nabla u \cdot \nabla v = \int_\Gamma (\partial_n u)\, v.$$

Note that there are two types of integrals in this formula. Both integrals in the left–hand side are domain integrals in Ω, whereas the integral in the right–hand side is a line integral on the boundary Γ. By the way, the result is also true in three dimensions. In that case, domain integrals are volume integrals and boundary integrals are surface integrals. The dot between the gradients denotes simply the Euclidean product of vectors, so

$$\nabla u \cdot \nabla v = \frac{\partial u}{\partial x_1}\frac{\partial v}{\partial x_1} + \frac{\partial u}{\partial x_2}\frac{\partial v}{\partial x_2}$$

Remark. This theorem is in fact a simple consequence of the Divergence Theorem:

$$\int_\Omega (\operatorname{div}\mathbf{p})\,v + \int_\Omega \mathbf{p}\cdot\nabla v = \int_\Gamma (\mathbf{p}\cdot\mathbf{n})\,v.$$

Here $\operatorname{div}\mathbf{p}$ is the divergence of the vector field $\mathbf{p}$, that is, if $\mathbf{p} = (p_1, p_2)$

$$\operatorname{div}\mathbf{p} = \frac{\partial p_1}{\partial x_1} + \frac{\partial p_2}{\partial x_2}.$$

If you take $\mathbf{p} = \nabla u$ you obtain Green's Theorem. □

1.4 The problem, written in weak form

The departure point for the weak or variational formulation is Green's Theorem. Here it is again

$$\int_\Omega (\Delta u)\,v + \int_\Omega \nabla u\cdot\nabla v = \int_\Gamma (\partial_n u)\,v = \int_{\Gamma_D} (\partial_n u)\,v + \int_{\Gamma_N} (\partial_n u)\,v.$$

Note that we have parted the integral on Γ as the sum of the integrals over the two sub–boundaries, the Dirichlet and the Neumann boundary. You may be wondering what v is in this context. In fact, it is nothing but a test. Wait for comments on this as the section progresses.

Now we substitute what we know in this formula: we know that $\Delta u = f - c\,u$ in Ω and that $\partial_n u = g_1$ on Γ_N. Therefore, after some reordering

$$\int_\Omega \nabla u\cdot\nabla v + c\int_\Omega u\,v = \int_\Omega f\,v + \int_{\Gamma_N} g_1\,v + \int_{\Gamma_D} (\partial_n u)\,v.$$

Note now that I've written all occurrences of u on the left hand side of the equation except for one I have left on the right. In fact we don't know the value of $\partial_n u$ on that part of the boundary. So what we will do is imposing that v cancels in that part, that is,

$$v = 0, \qquad \text{on } \Gamma_D.$$

Therefore

$$\int_\Omega \nabla u\cdot\nabla v + c\int_\Omega u\,v = \int_\Omega f\,v + \int_{\Gamma_N} g_1\,v, \qquad \text{if } v = 0 \text{ on } \Gamma_D.$$

Notice now three things:

- We have not imposed yet the Dirichlet boundary condition ($u = g_0$ on Γ_D). Nevertheless, we have imposed a similar one to the function v, but in a homogeneous way.

- As written now, data (f and g_1) are in the right–hand side and coefficients of the equation (the only one we have is c) are in the left–hand side.

- The expression on the left–hand side is linear in both u and v. It is a bilinear form of the variables u and v. The expression on the right–hand side is linear in v.

Without specifying spaces where u and v are, the weak formulation can be written as follows:

$$\left[\begin{array}{l} \text{find } u \text{ such that} \\ u = g_0, \qquad \text{on } \Gamma_D, \\ \displaystyle\int_\Omega \nabla u \cdot \nabla v + c \int_\Omega u\, v = \int_\Omega f\, v + \int_{\Gamma_N} g_1\, v, \qquad \text{for all } v, \text{ such that } v = 0 \text{ on } \Gamma_D. \end{array}\right.$$

Note how the two boundary conditions appear in very different places of this formulation:

- The Dirichlet condition (given displacements) is imposed apart from the formulation and involves imposing it homogeneously to the testing function v. It is called an **essential boundary condition**.
- The Neumann condition (given normal stresses) appears inside the formulation. It is called a **natural boundary condition**.

Being essential or natural is not inherently tied to the boundary condition: it is related to the role of the boundary condition in the formulation. So when you hear (or say) essential boundary condition, you mean a boundary condition that is imposed apart from the formulation, whereas a natural boundary condition appears inside the formulation. *For this weak formulation of a second order elliptic equation we have*

Dirichlet=essential   Neumann=natural

What is v? At this point, you might (you should) be wondering what is v in the formulation. In the jargon of weak formulations, v is called a test function. It tests the equation that is satisfied by u. The main idea is that instead of looking at the equation as something satisfied point–by–point in the domain Ω, you have an averaged version of the equation. Then v plays the role of a weight function, something you use to average the equation. In many contexts (books on mechanics, engineering or physics) v is called a virtual displacement (or virtual work, or virtual whatever is pertinent), emphasizing the fact that v is not the unknown of the system, but something that only exists virtually to write down the problem. The weak formulation is, in that context, a principle of virtual displacements (principle of virtual work, etc). □

1.5 Delimiting spaces

We have reached a point where we should be a little more specific on where we are looking for u and where v belongs. The first space we need is the space of square–integrable functions

$$L^2(\Omega) = \left\{ f : \Omega \to \mathbb{R} \,\middle|\, \int_\Omega |f|^2 < \infty \right\}.$$

A fully precise definition of this space requires either the introduction of the Lebesgue integral or applying some limiting ideas. If you know what this is all about, good for you! If you don't, go on: for most functions you know you will always be able to check whether they belong to this space or not by computing or estimating the integral and seeing if it is finite or not.

The second space is one of the wide family of Sobolev spaces:

$$H^1(\Omega) = \Big\{ u \in L^2(\Omega) \,\Big|\, \tfrac{\partial u}{\partial x_1}, \tfrac{\partial u}{\partial x_2} \in L^2(\Omega) \Big\}.$$

There is a norm related to this space

$$\|u\|_{1,\Omega} = \left(\int_\Omega |\nabla u|^2 + \int_\Omega |u|^2 \right)^{1/2} = \left(\int_\Omega \left| \frac{\partial u}{\partial x_1} \right|^2 + \int_\Omega \left| \frac{\partial u}{\partial x_2} \right|^2 + \int_\Omega |u|^2 \right)^{1/2}.$$

Sometimes this norm is called the energy norm and functions that have this norm finite (that is, functions in $H^1(\Omega)$) are called functions of finite energy. The concept of energy is however related to the particular problem, so it's better to get used to have the space and its norm clearly written down and think of belonging to this space as a type of admissibility condition.

A particular subset of this space will be of interest for us:

$$H^1_{\Gamma_D}(\Omega) = \{ v \in H^1(\Omega) \,|\, v = 0, \quad \text{on } \Gamma_D \}.$$

Note that $H^1_{\Gamma_D}(\Omega)$ is a subspace of $H^1(\Omega)$, that is, linear combinations of elements of $H^1_{\Gamma_D}(\Omega)$ belong to the same space.

The Mathematics behind. An even half–trained mathematician should be wondering what do we mean by the partial derivatives in the definition of $H^1(\Omega)$, since one cannot think of taking the gradient of an arbitrary function of $L^2(\Omega)$, or at least to taking the gradient and finding something reasonable. What we mean by restriction to Γ_D in the definition of $H^1_{\Gamma_D}(\Omega)$ is not clear either, since elements or $L^2(\Omega)$ are not really functions, but classes of functions, where values of the function on particular points or even on lines are not relevant. To make this completely precise there are several ways:

- Define a weak derivative for elements of $L^2(\Omega)$ and what we understand by saying that that derivative is again in $L^2(\Omega)$. Then you move to give a meaning to that restriction of a function in $H^1(\Omega)$ to one part of its boundary.

- Go the whole nine yards and take time to browse a book on distribution theory and Sobolev spaces. It takes a while but you end up with a pretty good intuition of what this all is about.

- Take the short way. You first consider the space of functions

$$\mathcal{C}^1(\overline{\Omega}) = \Big\{ u \in \mathcal{C}(\overline{\Omega}) \,\Big|\, \tfrac{\partial u}{\partial x_1}, \tfrac{\partial u}{\partial x_2} \in \mathcal{C}(\overline{\Omega}) \Big\},$$

which is simple to define, and then you close it with the norm $\|\cdot\|_{1,\Omega}$. To do that you have to know what closing or completing a space is (it's something similar to what you do to define real numbers from rational numbers). Then you have to prove that restricting to Γ_D still makes sense after this completion procedure.

My recommendation at this point is to simply go on. If you are a mathematician you can take later on some time with a good simple book on elliptic PDEs and will see that it is not that complicated. If you are a physicist or an engineer you will probably not need to understand all the details of this. There's going to be a very important result in the next section that you will have to remember and that's almost all. Nevertheless, if you keep on doing research related to finite elements, you should really know something more about this. In due time you will have to find any of the dozens of books on Partial Differential Equations for Scientists and Engineers, and read the details, which will however not be given in the excruciating detail of PDE books for mathematicians. But this is only an opinion. $\square$

1.6 The weak form again

With the spaces defined above we can finally write our problem in a proper and fully rigorous way:

$$\left[\begin{array}{l} \text{find } u \in H^1(\Omega), \text{ such that} \\ u = g_0, \qquad \text{on } \Gamma_D, \\ \displaystyle\int_\Omega \nabla u \cdot \nabla v + c\int_\Omega u\,v = \int_\Omega f\,v + \int_{\Gamma_N} g_1\,v, \qquad \forall v \in H^1_{\Gamma_D}(\Omega) \end{array}\right.$$

Let me recall that the condition on the general test function $v \in H^1_{\Gamma_D}(\Omega)$ is the same as

$$v \in H^1(\Omega), \qquad \text{such that } v = 0, \quad \text{on } \Gamma_D,$$

that is, v is in the same space as the unknown u but satisfies a homogeneous version of the essential boundary condition.

The data are in the following spaces

$$f \in L^2(\Omega), \qquad g_1 \in L^2(\Gamma_N), \qquad g_0 \in H^{1/2}(\Gamma_D).$$

We have already spoken of the first of these spaces. The space $L^2(\Gamma_N)$ is essentially the same idea, with line integrals on Γ_N instead of domain integrals on Ω. The last space looks more mysterious: it is simply the space of restrictions to Γ_D of functions of $H^1(\Omega)$, that is, $g_0 \in H^{1/2}(\Gamma_D)$ means that there exists at least a function $u_0 \in H^1(\Omega)$ such that $u_0 = g_0$ on Γ_D. In fact, all other functions satisfying this condition (in particular our solution u) belong to

$$u_0 + H^1_{\Gamma_D}(\Omega) = \{u_0 + v \,|\, v \in H^1_{\Gamma_D}(\Omega)\} = \{w \in H^1(\Omega) \,|\, w = g_0, \quad \text{on } \Gamma_D\}$$

(can you see why?). Unlike $H^1_{\Gamma_D}(\Omega)$, this set is not a subspace of $H^1(\Omega)$. The only exception is the trivial case, when $g_0 = 0$, since the set becomes $H^1_{\Gamma_D}(\Omega)$.

That g_0 belongs to $H^{1/2}(\Gamma_D)$ means simply that we are not looking for the solution on the empty set. I cannot give you here a simple and convincing explanation on the name of this space. Sorry for that.

2 The space of continuous linear finite elements

It's taken a while, but we are there! *Numerics start here.* We are now going to discretize all the elements appearing in this problem: the physical domain, the function spaces and the variational/weak formulation.

We are going to do it step by step. At the end of this section you will have the simplest example of a space of finite element functions (or simply finite elements). Many mathematicians call these elements Courant elements, because Richard Courant introduced them several decades ago with theoretical more than numerical intentions. In the jargon of the business we call them triangular Lagrange finite elements of order one, or simply linear finite elements, or for short (because using initials and short names helps speaking faster and looking more dynamic) $\mathbb{P}_1$ elements.

2.1 Linear functions on a triangle

First of all, let us think for a moment about linear functions. A linear function[1] of two variables is the same as a polynomial function of degree at most one

$$p(x_1, x_2) = a_0 + a_1\, x_1 + a_2\, x_2.$$

The set of these functions is denoted $\mathbb{P}_1$. Everybody knows that a linear function is uniquely determined by its values on three different non–aligned points, that is, on the vertices of a (non–degenerate) triangle.

Let us then take an arbitrary non–degenerate triangle, that we call K. You might prefer calling the triangle T, as many people do. However, later on (in Lesson 3) the triangle will stop being a triangle and will become something else, maybe a quadrilateral, and then the meaning of the initial T will be lost. We draw it as in Figure 1.2, marking its three vertices. With this we mean that *a function*

$$p \in \mathbb{P}_1 = \{a_0 + a_1\, x_1 + a_2\, x_2 \,|\, a_0, a_1, a_2 \in \mathbb{R}\}$$

is uniquely determined by its values on these points. Uniquely determined means two things: (a) there is only one function with given values on the vertices; (b) there is in fact one function, that is, the values on the vertices are arbitrary. We can take any values we want and will have an element of $\mathbb{P}_1$ with these values on the vertices. Graphically it is just hanging a flat (linear) function from three non–aligned points.

Thus, a function $p \in \mathbb{P}_1$ can be determined

- either from its three defining coefficients (a_0, a_1, a_2)

[1]For Spanish speakers: please note that in Spanish we call these functions affine and never linear. This distinction is not always done in mathematical English usage.

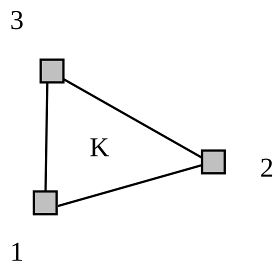


Figure 1.2: A triangle and its three vertices

- or from its values on the three vertices of a triangle K.

Both possibilities state that the space $\mathbb{P}_1$ is a vector space of dimension three. While the first choice (coefficients) gives us a simple expression of the function, the second is more useful for many tasks, in particular for drawing the function. The three values of the function on the vertices will be called the **local degrees of freedom**.

There is another important property that will be extremely useful in the sequel: *the value of $p \in \mathbb{P}_1$ on the edge that joins two vertices of the triangle depends only on the values of p on this two vertices.* In other words, the value of $p \in \mathbb{P}_1$ on an edge is uniquely determined by the degrees of freedom associated to the edge, namely, the values of p on the two vertices that lie on that edge.

2.2 Triangulations

So far we have functions on a single triangle. Now we go for partitions of the domain into triangles. A triangulation of Ω is a subdivision of this domain into triangles. Triangles must cover all Ω but no more and must fulfill the following rule:

> *If two triangles have some intersection, it is either on common vertex or a common full edge. In particular, two different triangles do not overlap.*

Figure 1.3 shows two forbidden configurations. See Figure 1.5 to see how a triangulation looks like. There is another rule, related to the partition of Γ into Γ_D and Γ_N:

> *The triangulation must respect the partition of the boundary into Dirichlet and Neumann boundaries.*

This means that an edge of a triangle that lies on Γ cannot be part Dirichlet and part Neumann. Therefore if there is a transition from Dirichlet to Neumann boundaries, there must be a vertex of a triangle in that transition point. Note that this situation has to be taken into account only when there is a transition from Dirichlet to Neumann conditions inside a side of the polygon Ω.

The set of the triangles (that is, the list thereof) will be generally denoted $\mathcal{T}_h$. The subindex h makes reference to the diameter of the triangulation, defined as *the length of the longest edge of all triangles*, that is, the longest distance between vertices of the triangulation.

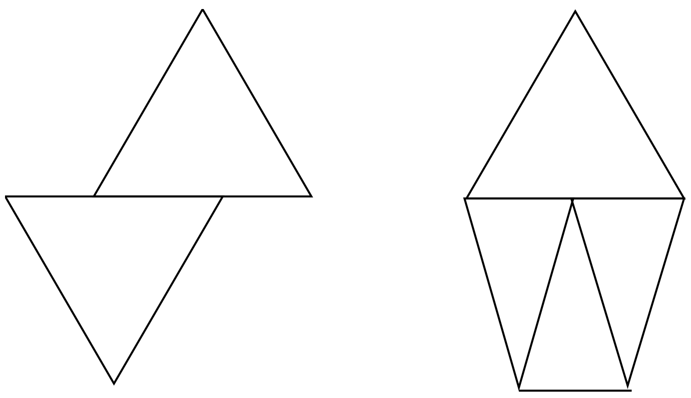


Figure 1.3: Situations not admitted in triangulations. In the second one we see the appearance of what is called a hanging node.

2.3 Piecewise linear functions on a triangulation

We now turn our attention to functions defined on the whole of the polygon Ω that has been triangulated as shown before.

Consider first two triangles sharing a common edge, say K and K' (see Figure 1.6). We take values at the four vertices of this figure and build a function that belongs to $\mathbb{P}_1$ on each of the triangles and has the required values on the vertices. Obviously we can define a unique function with this property. Moreover, since the value on the common edge depends only on the values on the two common vertices, the resulting function is continuous.

We can do this triangle by triangle. We end up with a function that is linear on each triangle and globally continuous. The space of such functions is

$$V_h = \Big\{ u_h \in \mathcal{C}(\overline{\Omega}) \, \Big| \, u_h|_K \in \mathbb{P}_1, \quad \forall K \in \mathcal{T}_h \Big\}.$$

If we fix values on the set of vertices of the triangulation $\mathcal{T}_h$, there exists a unique $u_h \in V_h$ with those values on the vertices. Therefore an element of V_h is uniquely determined by its values on the set of vertices of the triangulation. The values on the vertices of the whole triangulation are the degrees of freedom that determine an element of V_h. In this context we will call **nodes** to the vertices in their role as points where we take values. (Note that in forthcoming lessons there will be other nodes in addition to vertices).

Elements of the space V_h are called linear finite element functions or simply $\mathbb{P}_1$ finite elements.

Let us take now a numbering of the set of nodes (that is, vertices) of the triangulation. At this moment any numbering goes[2]. In Figure 1.7 we have a numbering of the nodes of the triangulation of our model domain. The vertices will be generically denoted $\mathbf{p}_i$ with i varying from one to the number of vertices, which we call N.

[2]And in many instances this will be so to the end of the discretization process. Using one numbering or another has a great influence on the shape of the linear section we will obtain in Section 3, but this shape is relevant only for some choices of the method to solve the corresponding linear system

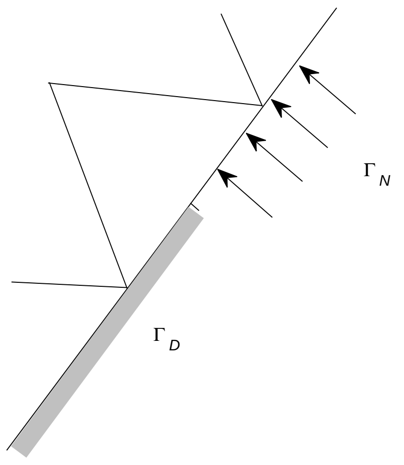


Figure 1.4: A forbidden transition of Dirichlet to Neumann boundary conditions happening inside an edge. Graphical notation for Dirichlet a Neumann boundaries as shown in many Mechanics books are give in the graph

Because of what we have explained above, if we fix one node (vertex) and associate the value one to this node and zero to all others, there exists a unique function $\varphi_i \in V_h$ that has these values, that is,

$$\varphi_i(\mathbf{p}_j) = \delta_{ij} = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases}$$

The aspect of one of these functions is shown in Figure 1.8.

Notice that if a triangle K has not $\mathbf{p}_i$ as one of its vertices, φ_i vanishes all over K, since the value of φ_i on the three vertices of K is zero. Therefore, the support of φ_i (the closure of the set of points where φ_i is not zero) is the same as the union of triangles that share $\mathbf{p}_i$ as vertex. In Figure 1.9 you can see the type of supports you can find.

There is even more. Take $u_h \in V_h$. It is simple to see that

$$u_h = \sum_{j=1}^N u_h(\mathbf{p}_j)\varphi_j.$$

Why? Let me explain. Take the function $\sum_{j=1}^N u_h(\mathbf{p}_j)\varphi_j$ and evaluate it in $\mathbf{p}_i$: you obtain

$$\sum_{j=1}^N u_h(\mathbf{p}_j)\varphi_j(\mathbf{p}_i) = \sum_{j=1}^N u_h(\mathbf{p}_j)\delta_{ji} = u_h(\mathbf{p}_i).$$

Therefore, this function has exactly the same nodal values as u_h and must be u_h. The fact that two functions of V_h with the same nodal values are the same function is the linear independence of the nodal functions $\{\varphi_i\}$. What we have proved is the fact that $\{\varphi_i \,|\, i = 1, \ldots, N\}$ is a basis of V_h and therefore

$$\dim V_h = N = \#\{\text{vertices}\}.$$

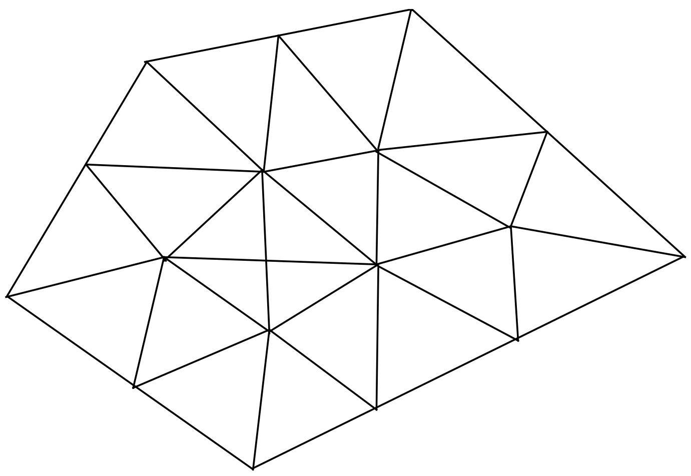


Figure 1.5: A triangulation of Ω

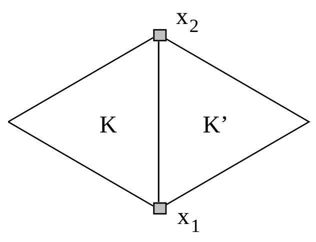


Figure 1.6: Two triangles with a common edge

There is a particularly interesting aspect of this basis of V_h that makes it especial. In general if you have a basis of V_h you know that you can decompose elements of V_h as a unique linear combination of the elements of the basis, that is,

$$u_h = \sum_{j=1}^{N} u_j \, \varphi_j$$

is a general element of V_h. With this basis, the coefficients are precisely the values of u_h on the nodes, that is, $u_j = u_h(\mathbf{p}_j)$. Hence, the coefficients of u_h in this basis are something more than coefficients: there are values of the function on points.

An important result. As you can see, when defining the space V_h we have just glued together $\mathbb{P}_1$ functions on triangles. Thanks to the way we have made the triangulation and to the way we chose the local degrees of freedom, what we obtained was a continuous function. One can think, is this so important? Could I take something discontinuous? At this level, the answer is a very load and clear NO! The reason is the following result that allows us to know whether certain functions are in $H^1(\Omega)$ or not.

Theorem. *Let u_h be a function defined on a triangulation of Ω such that*

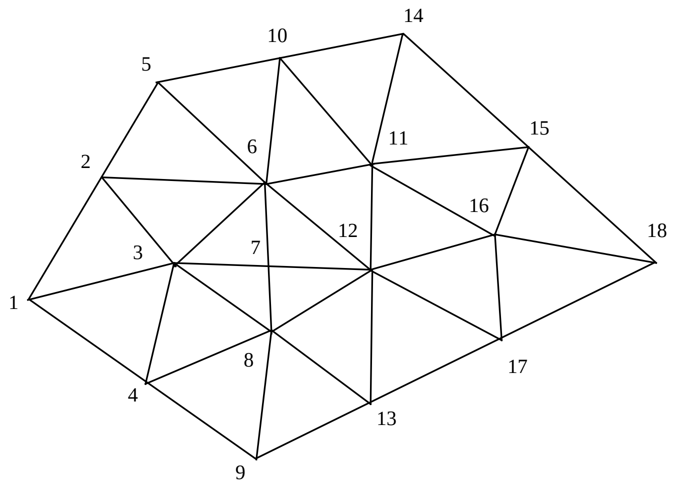


Figure 1.7: Global numbering of nodes

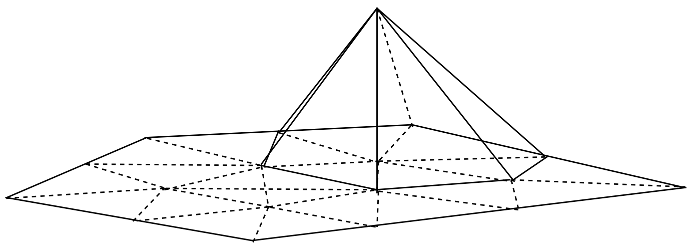


Figure 1.8: The graph of a nodal basis function: it looks like a camping tent.

restricted to each triangle it is a polynomial (or smooth) function. Then

$$u_h \in H^1(\Omega) \qquad \Longleftrightarrow \qquad u_h \textit{ is continuous.}$$

There is certain intuition to be had on why this result is true. If you take a derivative of a piecewise smooth function, you obtain Dirac distributions along the lines where there are discontinuities. Dirac distributions are not functions and it does not make sense to see if the are square–integrable or not. Therefore, if there are discontinuities, the function fails to have a square–integrable gradient. □

2.4 Dirichlet nodes

So far we have taken into account the discrete version of the domain Ω but not the partition of its boundary Γ into Dirichlet and Neumann sides. We first need some terminology. A Dirichlet edge is an edge of a triangle that lies on Γ_D. Similarly a **Neumann edge** is an edge of a triangle that is contained in Γ_N. The vertices of the Dirichlet edges are called **Dirichlet nodes**. Notice that the doubt may arise in transitions from the Dirichlet to the Neumann part of the boundary. If a node belongs to both Γ_N and Γ_D it is a Dirichlet node.

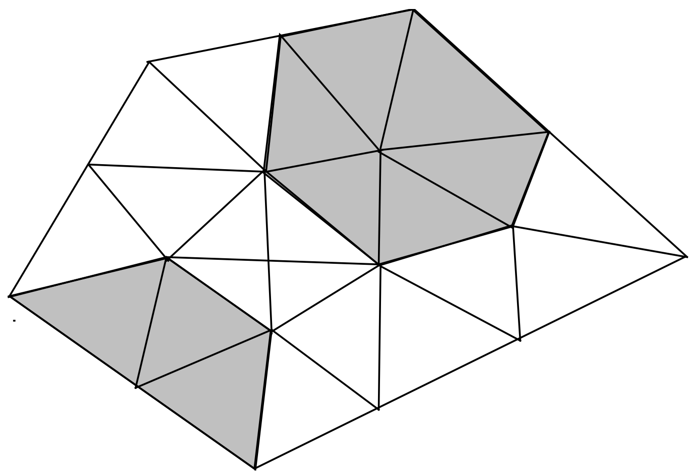


Figure 1.9: Supports of two nodal basis functions

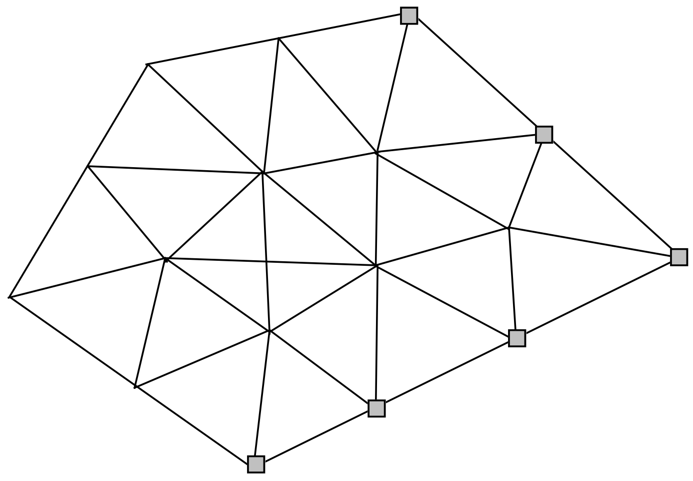


Figure 1.10: Dirichlet nodes corresponding to the domain as depicted in Figure 1.1

In truth, in parallel to what happens with how the Dirichlet and Neumann boundary conditions are treated in the weak formulation, we will inherit two different discrete entities:

- Dirichlet nodes, and
- Neumann edges.

Let us now recall the space

$$H^1_{\Gamma_D}(\Omega) = \{v \in H^1(\Omega) \,|\, v = 0 \quad \text{on } \Gamma_D\}.$$

We might be interested in the space

$$V_h^{\Gamma_D} = V_h \cap H^1_{\Gamma_D}(\Omega) = \{v_h \in V_h \,|\, v_h = 0, \quad \text{on } \Gamma_D\}.$$

Recall now the demand we did on the triangulation to respect the partition of Γ into Dirichlet and Neumann parts. Because of this, $v_h \in V_h$ vanishes on Γ_D if and only if it vanishes on the Dirichlet edges. Again, since values of piecewise linear functions on edges are determined by the values on the corresponding vertices, we have

$v_h \in V_h$ vanishes on Γ_D if and only if it vanishes on all Dirichlet nodes.

The good news is the fact that we can easily construct a basis of $V_h^{\Gamma_D}$. We simply eliminate the elements of the nodal basis corresponding to Dirichlet nodes. To see that recall that when we write $v_h \in V_h$ as a linear combination of elements of the nodal basis, what we have is actually

$$v_h = \sum_{j=1}^{N} v_h(\mathbf{p}_j)\varphi_j.$$

Therefore $v_h = 0$ on Γ_D if and only if the coefficients corresponding to nodal functions of Dirichlet nodes vanish. To write this more efficiently we will employ two lists, Dir and Ind (as in *independent* or free nodes), to number separately Dirichlet and non–Dirichlet (independent/free) nodes. It is not necessary to number first one type of nodes and then the other, although sometimes it helps to visualize things to assume that we first numbered the free nodes and then the Dirichlet nodes.[3] With our model triangulation numbered as in Figure 1.7 and with the Dirichlet nodes marked in 1.10, the lists are

$$\begin{aligned} \text{Dir} &= \{9, 13, 14, 15, 17, 18\}, \\ \text{Ind} &= \{1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 16\}. \end{aligned}$$

With these lists, an element of V_h can be written as

$$u_h = \sum_{j \in \text{Ind}} u_j \varphi_j + \sum_{j \in \text{Dir}} u_j \varphi_j, \qquad u_j = u_h(\mathbf{p}_j)$$

and an element of $V_h^{\Gamma_D}$ is of the form

$$v_h = \sum_{j \in \text{Ind}} v_j \varphi_j.$$

Finally, this proves that

$$\text{dim} V_h^{\Gamma_D} = \#\text{Ind} = \#\{\text{nodes}\} - \#\{\text{Dirichlet nodes}\}.$$

[3]The reason for not doing this is merely practical. The triangulation is done without taking into account which parts of the boundary are Dirichlet and which are Neumann. As we will see in the next Lesson, the numbering of the nodes is inherent to the way the triangulation is given. In many practical problems we play with the boundary conditions for the same domain and it is not convenient to renumber the vertices each time.

3 The finite element method

3.1 The discrete variational problem

After almost fifteen pages of introducing things we can finally arrive to a numerical approximation of our initial problem. Recall that we wrote the problem in the following form

$$\left[\begin{array}{l} \text{find } u \in H^1(\Omega), \text{ such that} \\ u = g_0, \qquad \text{on } \Gamma_D, \\ \displaystyle\int_\Omega \nabla u \cdot \nabla v + c \int_\Omega u\, v = \int_\Omega f\, v + \int_{\Gamma_N} g_1\, v, \qquad \forall v \in H^1_{\Gamma_D}(\Omega). \end{array}\right.$$

The finite element method (with linear finite elements on triangles) consists of the following discrete version of the preceding weak formulation:

$$\left[\begin{array}{l} \text{find } u_h \in V_h, \text{ such that} \\ u_h(\mathbf{p}) = g_0(\mathbf{p}), \qquad \text{for all Dirichlet node } \mathbf{p}, \\ \displaystyle\int_\Omega \nabla u_h \cdot \nabla v_h + c \int_\Omega u_h\, v_h = \int_\Omega f\, v_h + \int_{\Gamma_N} g_1\, v_h, \qquad \forall v_h \in V_h^{\Gamma_D}. \end{array}\right.$$

As you can easily see we have done three substitutions:

- We look for the unknown in the space V_h instead of on the whole Sobolev space. This means that we have reduced the problem to computing u_h in the vertices of the triangulation (in the nodes) and we are left with a finite number of unknowns.
- We have substituted the Dirichlet condition by fixing the values of the unknowns on Dirichlet nodes. This fact reduces the number of unknowns of the system to the number of free nodes.[4]
- Finally, we have reduced the testing space from $H^1_{\Gamma_D}(\Omega)$ to its discrete subspace $V_h^{\Gamma_D}$. We will show right now that this reduces the infinite number of tests of the weak formulation to a finite number of linear equations.

3.2 The associated system

We write again the discrete problem, specifying the numbering of Dirichlet nodes in the discrete Dirichlet condition:

$$\left[\begin{array}{l} \text{find } u_h \in V_h, \text{ such that} \\ u_h(\mathbf{p}_j) = g_0(\mathbf{p}_j), \qquad \forall j \in \text{Dir}, \\ \displaystyle\int_\Omega \nabla u_h \cdot \nabla v_h + c \int_\Omega u_h\, v_h = \int_\Omega f\, v_h + \int_{\Gamma_N} g_1\, v_h, \qquad \forall v_h \in V_h^{\Gamma_D}. \end{array}\right.$$

[4]This way of substituting the Dirichlet condition by a sort of interpolated Dirichlet condition is neither the only nor the best way of doing this approximation, but it is definitely the simplest, so we will keep it like this for the time being.

Our next claim is the following: the discrete equations

$$\int_\Omega \nabla u_h \cdot \nabla v_h + c \int_\Omega u_h\, v_h = \int_\Omega f\, v_h + \int_{\Gamma_N} g_1\, v_h, \qquad \forall v_h \in V_h^{\Gamma_D}$$

are equivalent to the following set of equations

$$\int_\Omega \nabla u_h \cdot \nabla \varphi_i + c \int_\Omega u_h\, \varphi_i = \int_\Omega f\, \varphi_i + \int_{\Gamma_N} g_1\, \varphi_i, \qquad \forall i \in \mathrm{Ind}.$$

Obviously this second group of equations is a (small) part of the original one: it is enough to take $v_h = \varphi_i \in V_h^{\Gamma_D}$. However, because of the linearity of the first expression in v_h, if we have the second for all φ_i, we have the equation for all possible linear combinations of these functions, that is for all $v_h \in V_h^{\Gamma_D}$. Recapitulating, the method is equivalent to this set of N equations to determine the function u_h:

$$\left[\begin{array}{l} \text{find } u_h \in V_h, \text{ such that} \\ u_h(\mathbf{p}_j) = g_0(\mathbf{p}_j), \qquad \forall j \in \mathrm{Dir}, \\ \displaystyle\int_\Omega \nabla u_h \cdot \nabla \varphi_i + c \int_\Omega u_h\, \varphi_i = \int_\Omega f\, \varphi_i + \int_{\Gamma_N} g_1\, \varphi_i, \qquad \forall i \in \mathrm{Ind}. \end{array}\right.$$

To arrive to a linear system, we have first to write u_h in terms of the nodal basis functions

$$u_h = \sum_{j\in\mathrm{Ind}} u_j \varphi_j + \sum_{j\in\mathrm{Dir}} u_j \varphi_j.$$

Then we substitute the discrete Dirichlet condition in this expression

$$u_h = \sum_{j\in\mathrm{Ind}} u_j \varphi_j + \sum_{j\in\mathrm{Dir}} g_0(\mathbf{p}_j) \varphi_j.$$

Finally we plug this expression in the discrete variational equation

$$\int_\Omega \nabla u_h \cdot \nabla \varphi_i + c \int_\Omega u_h\, \varphi_i = \int_\Omega f\, \varphi_i + \int_{\Gamma_N} g_1\, \varphi_i,$$

apply linearity, noticing that

$$\nabla u_h = \sum_{j\in\mathrm{Ind}} u_j \nabla\varphi_j + \sum_{j\in\mathrm{Dir}} g_0(\mathbf{p}_j) \nabla\varphi_j$$

and move to the right–hand side what we already know (the Dirichlet data)

$$\begin{aligned} \sum_{j\in\mathrm{Ind}} \Big(\int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i + c \int_\Omega \varphi_j \varphi_j \Big) u_j \;=\; & \int_\Omega f\, \varphi_i + \int_{\Gamma_N} g_1\, \varphi_i \\ & - \sum_{j\in\mathrm{Dir}} \Big(\int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i + c \int_\Omega \varphi_j \varphi_j \Big) g_0(\mathbf{p}_j). \end{aligned}$$

This is a linear system with as many equations as unknowns, namely with $\#\mathrm{Ind} = \dim V_h^{\Gamma_D}$ equations and unknowns. The unknowns are in fact the nodal values of u_h on the free (non–Dirichlet) vertices of the triangulation. After solving this linear system, the formula for u_h lets us recover the function everywhere, not only on nodes.

Remark Unlike the finite difference method, the finite element method gives as a result a function defined on the whole domain and not a set of point values. Reconstruction of the function from computed quantities is in the essence of the method and cannot be counted as a posprocessing of nodal values. $\Box$

3.3 Mass and stiffness

There are two matrices in the system above. Both of them participate in the final matrix and parts of them go to build the right hand side. First we have the **stiffness matrix**

$$W_{ij} = \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i$$

and second the **mass matrix**

$$M_{ij} = \int_\Omega \varphi_j\, \varphi_i.$$

Both matrices are defined for $i, j = 1, \ldots, N$ (although parts of these matrices won't be used). Both matrices are symmetric. The mass matrix $\mathbf{M}$ is positive definite. The stiffness matrix is positive semidefinite and in fact almost positive definite: if we eliminate take any index i and erase the $i-$th row and the $i-$th column of $\mathbf{W}$, the resulting matrix is positive definite.

The system can be easily written in terms of these matrices, using the vector

$$b_i = \int_\Omega f\, \varphi_i + \int_{\Gamma_N} g_1\, \varphi_i, \qquad i \in \mathrm{Ind},$$

to obtain

$$\sum_{j\in\mathrm{Ind}} \Big(W_{ij} + c\, M_{ij} \Big) u_j = b_i - \sum_{j\in\mathrm{Dir}} \Big(W_{ij} + c\, M_{ij} \Big) g_0(\mathbf{p}_j), \qquad i \in \mathrm{Ind}.$$

Note that this is clearly a square symmetric system. If $c = 0$ (then the original equation is the Poisson equation $-\Delta u = f$ and no reaction term appears), only the stiffness matrix appears. Therefore, stiffness comes from diffusion. Likewise mass proceeds from reaction.

The matrix is positive definite except in one special situation: when $c = 0$ and there are no Dirichlet conditions (i.e., $\Gamma_D = \varnothing$, i.e., $\mathrm{Ind} = \{1, \ldots, N\}$ and $V_h^{\Gamma_D} = V_h$). For the pure Neumann problem for the Laplace operator there are some minor solvability issues similar to the occurrence of rigid motions in mechanical problems. Let us ignore this minor complication for now.

Now look again at the figure showing the supports of nodal basis functions (we copy it right here for convenience) and look at the mass matrix

$$M_{ij} = \int_\Omega \varphi_j\, \varphi_i.$$

If the supports of φ_i and φ_j have no intersecting area, the integral defining M_{ij} vanishes. In fact, since the product of φ_i and φ_j is a non–negative function, $M_{ij} = 0$ if and only

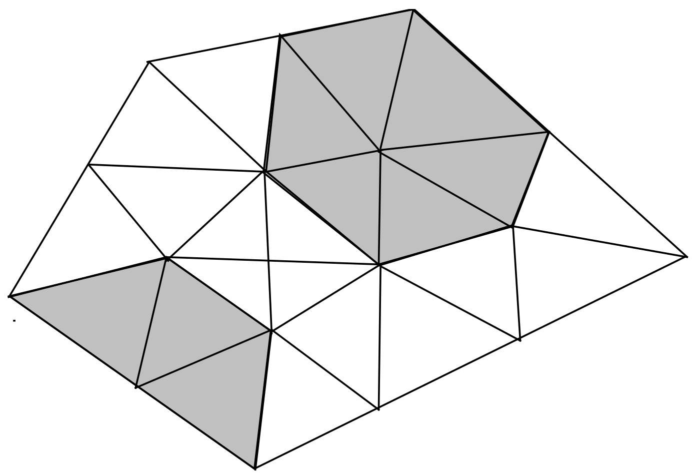


Figure 1.11: Supports of two nodal basis functions

if the area of the intersection of the supports is zero[5]. This happens whenever $\mathbf{p}_i$ and $\mathbf{p}_j$ are not vertices of the same triangle.

We say that two nodes are **adjacent** *if they belong to the same triangle.*

In the case of the stiffness matrix we have a similar (maybe weaker result): if the nodes i and j are not adjacent, then $W_{ij} = 0$.

This fact makes that the mass and stiffness matrices display a great sparsity character. Given a row i, there are only non–zero entries on positions related to adjacent nodes to the $i-$th node.

Going back to the system

$$\sum_{j\in\mathrm{Ind}} \Big(W_{ij} + c\,M_{ij}\Big) u_j = b_i - \sum_{j\in\mathrm{Dir}} \Big(W_{ij} + c\,M_{ij}\Big) g_0(\mathbf{p}_j), \qquad i \in \mathrm{Ind},$$

let us remark some simple facts:

- As it is now, all data appear in the right–hand side of the system (Neumann data and source terms are in the vector $\mathbf{b}$, Dirichlet data appear multipying columns of the stiffness–plus–mass matrix).
- Of the full matrices $\mathbf{W}$ and $\mathbf{M}$ we discard rows corresponding to Dirichlet nodes (Dir indices), since no testing is done with the corresponding basis functions. The columns corresponding to these indices are now eliminated however: they are sent to the right hand side multiplied by the values of the unknown in the Dirichlet nodes, which are known.

[5]By definition the support of a function includes the boundary of the set where the function is non–zero. Therefore, it is possible that the intersection is one edge. The integral is still zero.

4 Exercises

E1.1. third type of boundary condition

Let us consider our usual polygon Ω and the boundary value problem

$$\left[\begin{array}{ll} -\Delta u + u = f, & \text{in } \Omega, \\ \partial_n u + k\, u = g, \text{on } \Gamma. & \end{array}\right.$$

Here k is a positive parameter. This type of boundary condition is usually called a boundary condition of the third kind (first being Dirichlet and second Neumann) or a Robin (or Fourier) boundary condition.

1. Write down the weak formulation for this problem. Note that the condition is natural and there will not be essential boundary condition in the resulting formulation.
2. Write down in detail (as in Sections 3.2/ 3.3) the linear system that has to be solved when we apply the finite element method to this problem. Check that there is a new matrix that can be seen as a boundary–mass matrix. How many non–zero entries has each row of this new matrix?

If we take ε very small and the following slightly modified version of the boundary condition

$$\varepsilon \partial_n u + u = g_0, \qquad \text{on } \Gamma$$

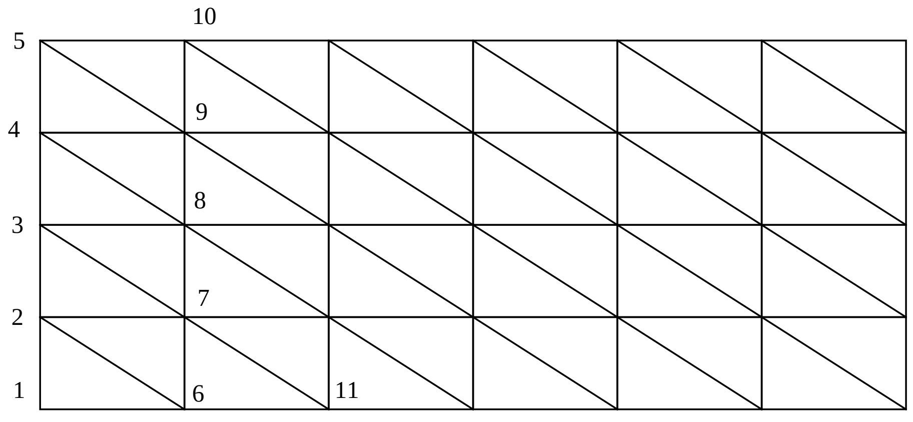
(take $k = \varepsilon^{-1}$ and $g = \varepsilon^{-1} g_0$), we are enforcing the Dirichlet condition in an approximate way. This is done in some commercial packages.

E1.2. A particular domain

Consider the boundary problem of Section 1 on the domain given in the next figure and the following specification for Γ_N and Γ_N

the left and upper sides have Dirichlet conditions

and where numbering is done as shown. Let $\mathbf{A} = \mathbf{W} + \mathbf{M}$ be the matrix associated to the system obtained by discretizing with the $\mathbb{P}_1$ finite element method



1. Write the index sets Dir and Ind.
2. Write which elements of the 12th row of $\mathbf{A}$ are non–zero.
3. Identify on the figure the support of the nodal basis function φ_{13}.
4. What's the size of the system that has to be solved?
5. We call the profile of the matrix $\mathbf{A}$ to the following vector:

$$m(i) = \inf\{j \mid a_{ij} \neq 0\}, \qquad i = 1, \dots, \#\{\text{nodos}\}$$

   that is, $m(i)$ indicates the column number where the first non–zero entry of the ith row is. Compute the profile of $\mathbf{W} + \mathbf{M}$ (without eliminating Dirichlet rows and columns). Draw the form of the matrix using the profile.
6. In the preceding graph mark which rows and columns will be modified by introduction of Dirichlet conditions. Compute the profile of the reduced matrix (without Dirichlet rows and columns).
7. What happens if we number nodes horizontally?

Lesson 2

Theoretical and practical notions

1 Assembly

The first lesson left us with a linear system to solve in order to approximate the boundary value problem with the finite element method. There is however the trick question on how to compute all the integrals that appear in the matrix and right–hand side of the system. This is done by a clever process called **assembly** of the system, another of the many good deeds of the finite element method that has made it so extremely popular (as in popular among scientists and engineers, of course) in the last decades.

At this moment we need the polygonal domain Ω and:

- a triangulation $\mathcal{T}_h$,
- a numbering of the nodes $\{\mathbf{p}_i\}$ (nodes are the vertices of the triangles),
- the set of the nodal basis functions $\{\varphi_i\}$.

In this section, nNod will be the global number of nodes.

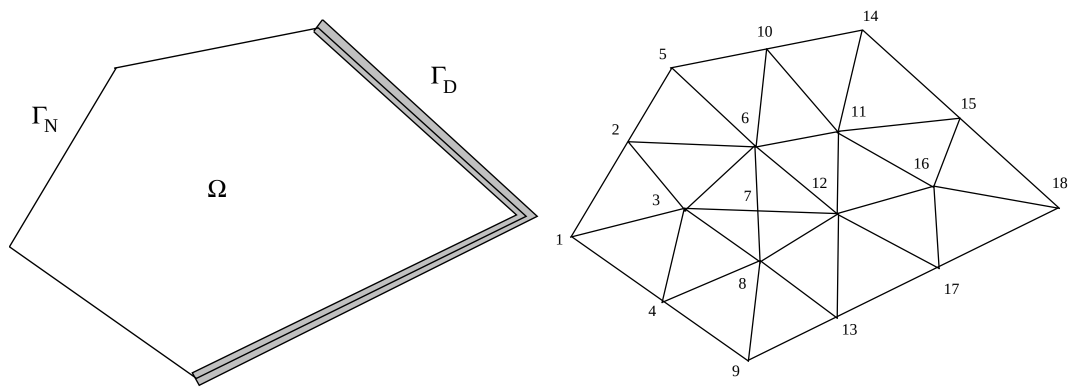


Figure 2.1: Geometry of the problem and triangulation

1.1 The mass and stiffness matrices

We are going to center our attention in the efficient construction of the stiffness matrix

$$w_{ij} = \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i$$

and of the mass matrix

$$m_{ij} = \int_\Omega \varphi_j\, \varphi_i.$$

Integrals over Ω can be decomposed as the sum of integrals over the different triangles

$$w_{ij} = \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i = \sum_K \int_K \nabla\varphi_j \cdot \nabla\varphi_i = \sum_K w_{ij}^K.$$

On each triangle we are going to define three local nodal basis functions. First assign a number to each of the three vertices of a triangle K:

$$\mathbf{p}_1^K, \quad \mathbf{p}_2^K, \quad \mathbf{p}_3^K.$$

Then consider the functions

$$N_1^K, \quad N_2^K, \quad N_3^K \quad \in \mathbb{P}_1$$

that satisfy

$$N_\alpha^K(\mathbf{p}_\beta^K) = \delta_{\alpha\beta}, \qquad \alpha, \beta = 1, 2, 3.$$

It is simple to see that the nodal basis function φ_i restricted to the triangle K is either zero (this happens when $\mathbf{p}_i$ is not one of the three vertices of K) or one of the N_α^K functions. More precisely, let n_α be the global number of the local node with number α in the triangle K. This means that

$$N_\alpha^K = \varphi_{n_\alpha}, \qquad \text{on the triangle } K.$$

We can now compute the 3×3 matrix $\mathbf{K}^K$

$$k_{\alpha\beta}^K = \int_K \nabla N_\beta^K \cdot \nabla N_\alpha^K, \qquad \alpha, \beta = 1, 2, 3.$$

This is due to be simple, since the functions N_α^K are polynomials of degree one (unlike the functions φ_i that are only piecewise polynomials). We will show later on strategies to do this computation. Note at this moment that computation of this matrix depends only on the triangle K and does not take into account any other element of the triangulation.

Therefore

$$k_{\alpha\beta}^K = w_{n_\alpha n_\beta}^K$$

All other elements of the matrix $\mathbf{W}^K$ are zero. Recall again that $\mathbf{W}^K$ is a nNon $\times$ nNod matrix and that

$$\mathbf{W} = \sum_K \mathbf{W}^K.$$

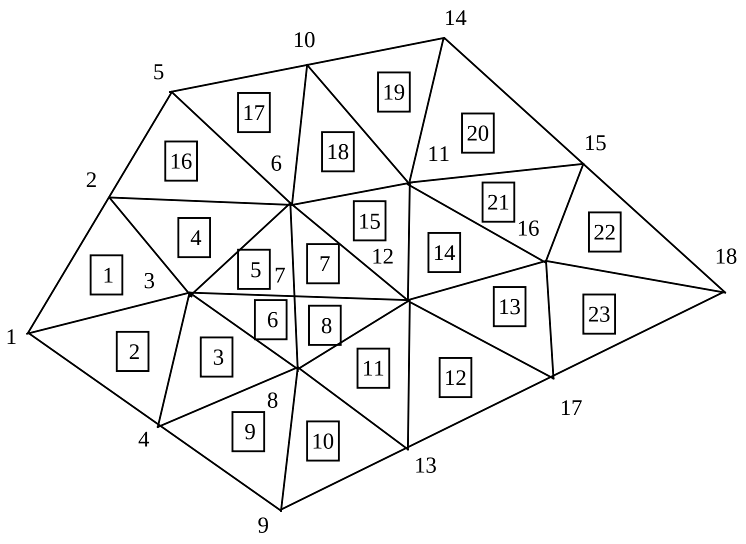
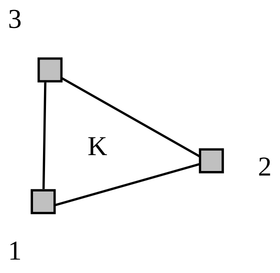


Figure 2.2: A numbering of the triangles



local		global
1	$\leftrightarrow$	12
2	$\leftrightarrow$	16
3	$\leftrightarrow$	11

Figure 2.3: The 14th triangle and their vertex numberings

The assembly process requires then a given numbering of triangles as shown in Figure 2.2. The order of this numbering is only used to do the computations but does not modify the shape of the final result.

The process to assemble the mass matrix is the same. Effective assembly of the mass and stiffness matrices can be done at the same time. Instead of computing separately the matrix $\mathbf{K}^K$ and a similar one for the mass matrix, we can directly try to compute the 3×3 matrix with elements

$$\int_K \nabla N_\beta^K \cdot \nabla N_\alpha^K + c \int_K N_\beta^K \, N_\alpha^K, \qquad \alpha, \beta = 1, 2, 3.$$

1.2 The reference element

To compute the elements

$$\int_K \nabla N_\beta^K \cdot \nabla N_\alpha^K \qquad \text{and} \qquad \int_K N_\beta^K \, N_\alpha^K$$

we need: (a) either and effective way of evaluating the functions N_α^K and their gradients; (b) or a closed form for the resulting integrals. Both possibilities are done usually by moving to the so–called reference element.

For triangles, the reference element is the triangle with vertices

$$\widehat{\mathbf{p}}_1 = (0,0), \qquad \widehat{\mathbf{p}}_2 = (1,0), \qquad \widehat{\mathbf{p}}_3 = (0,1).$$

To distinguish variables in the reference element and in a general triangle (in this context

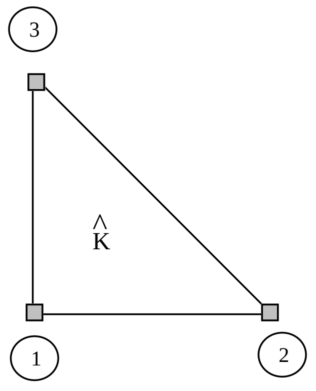


Figure 2.4: The reference element

we say a physical element) it is customary to use the variables (ξ, η) in the reference element and (x, y) in the physical element. In the mathematical literature for FEM it is also usual to hat variables in the reference element, so that $(\hat{x}, \hat{y})$ would be used to denote coordinates in the reference configuration.

An unimportant detail. Some people prefer to use a different reference triangle, with the same shape but with vertices on $(-1,-1)$, $(1,-1)$ and $(-1,1)$. Some details of the forthcoming computations have to be adapted if this choice is taken. $\square$

The local nodal functions in the reference triangles are three $\mathbb{P}_1$ functions satisfying

$$\widehat{N}_\alpha(\widehat{\mathbf{p}}_\beta) = \delta_{\alpha\beta}, \qquad \alpha, \beta = 1, 2, 3.$$

These functions are precisely

$$\widehat{N}_1 = 1 - \xi - \eta, \qquad \widehat{N}_2 = \xi, \qquad \widehat{N}_3 = \eta$$

or, if you prefer hatting variables (this is the last time we will write both expressions)

$$\widehat{N}_1 = 1 - \widehat{x} - \widehat{y}, \qquad \widehat{N}_2 = \widehat{x}, \qquad \widehat{N}_3 = \widehat{y}$$

Let us now take the three vertices of a triangle K

$$\mathbf{p}_1^K = (x_1, y_1), \qquad \mathbf{p}_2^K = (x_2, y_2), \qquad \mathbf{p}_3^K = (x_3, y_3).$$

The following affine transformation

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \underbrace{\begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}}_{\mathbf{B}_K} \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \\ &= \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} (1 - \xi - \eta) + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \xi + \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} \eta \end{aligned}$$

maps the triangle $\widehat{K}$ bijectively into K. In fact, if we call F_K to this transformation

$$F_K(\widehat{\mathbf{p}}_\alpha) = \mathbf{p}_\alpha^K, \qquad \alpha = 1, 2, 3.$$

Notice that the second expression we have written for the transformation gives it in terms of the nodal basis functions in the reference domain. You can think of it as a coincidence. In a way it is: the coincidence stems from the fact that the type of functions we are using for finite elements is the same as the functions needed to transform linearly triangles in the plane.

It is simple now to prove that

$$\widehat{N}_\alpha = N_\alpha^K \circ F_K, \qquad \alpha = 1, 2, 3,$$

or, what is the same

$$N_\alpha^K = \widehat{N}_\alpha \circ F_K^{-1}, \qquad \alpha = 1, 2, 3.$$

The $\circ$ symbol is used for composition. In the last expression, what we have is

$$N_\alpha^K(x, y) = \widehat{N}_\alpha(F_K^{-1}(x, y)).$$

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_α^K.

To evaluate the gradient of N_α^K we have to be more careful, since we have to apply the chain rule. Let us denote briefly gradients as

$$\nabla = \begin{bmatrix} \partial_x \\ \partial_y \end{bmatrix}, \qquad \widehat{\nabla} = \begin{bmatrix} \partial_\xi \\ \partial_\eta \end{bmatrix}$$

(note that we are writing gradients as column vectors). The following formula is the result of applying the chain rule

$$\mathbf{B}_K^\top \big(\nabla\phi \circ F_K\big) = \widehat{\nabla}(\phi \circ F_K).$$

$\mathbf{B}_K^\top$ is the transposed of the matrix of the linear transformation F_K. Taking $\phi = N_\alpha^K$ in this expression and moving things a little, we obtain a formula for the gradient of the local basis functions

$$\nabla N_\alpha^K = \mathbf{B}_K^{-\top} \Big((\widehat{\nabla} \widehat{N}_\alpha) \circ F_K^{-1} \Big).$$

The expression may look complicated but it is very simple to use. If we want to compute the value of the gradient of N_α^K at a point $(x, y) \in K$, we first compute the transformed point $(\xi, \eta) = F_K^{-1}(x, y)$ in the reference triangle, evaluate the gradient of $\widehat{N}_\alpha$ at this point and then multiply it by the matrix $\mathbf{B}_K^{-\top}$, which is the transposed of the inverse of $\mathbf{B}_K$, i.e.,

$$\mathbf{B}_K^{-\top} = \frac{1}{\det \mathbf{B}_K} \left[\begin{array}{cc} y_3 - y_1 & -(y_2 - y_1) \\ -(x_2 - x_1) & x_2 - x_1 \end{array} \right]$$

with

$$\det \mathbf{B}_K = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)$$

(remember that $|\det \mathbf{B}_K| = 2 \,\mathrm{area}\, K$). In fact, for this very elementary method, the gradients of the three basis functions on the reference element are constant vectors

$$\widehat{\nabla} \widehat{N}_1 = \left[\begin{array}{c} -1 \\ -1 \end{array} \right], \qquad \widehat{\nabla} \widehat{N}_2 = \left[\begin{array}{c} 1 \\ 0 \end{array} \right], \qquad \widehat{\nabla} \widehat{N}_3 = \left[\begin{array}{c} 0 \\ 1 \end{array} \right],$$

so computation of the constant vectors ∇N_α^K is very simple, and we don't even have to use the inverse transformation F_K^{-1} for the gradients. We do, however, to evaluate N_α^K.

1.3 Computing with quadrature rules

Depending on the complications of the problem (we are dealing with a very simple model problem), all the computations can be carried out to the reference element or we can try to do things directly on the physical triangle K. Let us mention here two popular quadrature rules for triangles: the three point rule with the vertices

$$\int_K \phi \approx \frac{\mathrm{area}\, K}{3} \Big(\phi(\mathbf{p}_1^K) + \phi(\mathbf{p}_2^K) + \phi(\mathbf{p}_3^K) \Big)$$

and the midpoints approximation

$$\int_K \phi \approx \frac{\mathrm{area}\, K}{3} \Big(\phi(\mathbf{m}_1^K) + \phi(\mathbf{m}_2^K) + \phi(\mathbf{m}_3^K) \Big),$$

where $\mathbf{m}_\alpha^K$ are the midpoints of the edges of K. If ϕ is a polynomial of degree one, the first formula gives the exact value. The second formula is even better: if ϕ is a polynomial of degree two, the midpoints formula is exact.

In the very simple case of $\mathbb{P}_1$ elements, we have ∇N_α^K constant and therefore

$$\int_K \nabla N_\beta^K \cdot \nabla N_\alpha^K = (\mathrm{area}\, K) \, \nabla N_\beta^K \cdot \nabla N_\alpha^K,$$

and this computation is very simple. For the mass matrix, we note that $N_\beta^K \, N_\alpha^K$ is a polynomial of degree two and therefore, the midpoints formula gives the exact value of the integrals

$$\int_K N_\beta^K \, N_\alpha^K$$

with just three evaluations of the functions.

1.4 Doing everything on the reference element

This section gives another idea on how to compute the local mass and stiffness matrices. You can skip it without losing continuity and go to Section 1.5. The change of variables applied to the integral of the local mass matrix gives

$$\int_K N_\beta^K \, N_\alpha^K = |\det \mathbf{B}_K| \int_{\widehat{K}} \widehat{N}_\beta \widehat{N}_\alpha.$$

Therefore everything is done once we have the 3×3 matrix

$$\widehat{\mathbf{K}}_0 = \left[\int_{\widehat{K}} \widehat{N}_\beta \widehat{N}_\alpha \right]_{\alpha,\beta} = \tfrac{1}{24} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

For derivatives, we have to be more careful

$$\begin{aligned} \int_K \nabla N_\beta^K \cdot \nabla N_\alpha^K &= |\det \mathbf{B}_K| \int_{\widehat{K}} \big(\nabla N_\beta^K \circ F_K\big) \cdot \big(\nabla N_\alpha^K \circ F_K\big) = \\ &= |\det \mathbf{B}_K| \int_{\widehat{K}} \Big(\mathbf{B}_K^{-\top} \widehat{\nabla} \widehat{N}_\beta\Big) \cdot \Big(\mathbf{B}_K^{-\top} \widehat{\nabla} \widehat{N}_\alpha\Big) = \\ &= |\det \mathbf{B}_K| \int_{\widehat{K}} \mathbf{C}_K \widehat{\nabla} \widehat{N}_\beta \cdot \widehat{\nabla} \widehat{N}_\alpha \end{aligned}$$

where

$$\mathbf{C}_K = \mathbf{B}_K^{-1} \mathbf{B}_K^{-\top} = \begin{bmatrix} c_{11}^K & c_{12}^K \\ c_{12}^K & c_{22}^K \end{bmatrix}$$

is a symmetric 2×2 matrix that depends only on the triangle. If we compute the following 3×3 matrices in the reference element

$$\begin{aligned} \widehat{\mathbf{K}}_{\xi\xi} &= \left[\int_{\widehat{K}} \partial_\xi \widehat{N}_\beta \, \partial_\xi \widehat{N}_\alpha \right]_{\alpha,\beta} = \tfrac{1}{2} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \widehat{\mathbf{K}}_{\eta\eta} &= \left[\int_{\widehat{K}} \partial_\eta \widehat{N}_\beta \, \partial_\eta \widehat{N}_\alpha \right]_{\alpha,\beta} = \tfrac{1}{2} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \\ \widehat{\mathbf{K}}_{\xi\eta} &= \left[\int_{\widehat{K}} \partial_\xi \widehat{N}_\beta \, \partial_\eta \widehat{N}_\alpha \right]_{\alpha,\beta} = \tfrac{1}{2} \begin{bmatrix} 1 & 0 & -1 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

we have

$$\left[\int_K \nabla N_\beta^K \cdot \nabla N_\alpha^K \right]_{\alpha,\beta} = |\det \mathbf{B}_K| \Big(c_{11}^K \widehat{\mathbf{K}}_{\xi\xi} + c_{22}^K \widehat{\mathbf{K}}_{\eta\eta} + c_{12}^K (\widehat{\mathbf{K}}_{\xi\eta} + \widehat{\mathbf{K}}_{\xi\eta}^\top) \Big).$$

1.5 Right–hand sides

Construction of the right–hand side of the linear system requires the computation of two vectors:

$$\int_\Omega f\,\varphi_i, \qquad \int_{\Gamma_N} g_1\,\varphi_i.$$

In principle, this has to be done for indices of free nodes ($i \in \mathrm{Ind}$), but in practice what is done is to compute them for all i and then discard the elements corresponding to Dirichlet nodes.

The surface forces (sources terms) can be treated in a similar way to the stiffness and mass matrices:

$$\int_\Omega f\,\varphi_i = \sum_K \int_K f\,\varphi_i.$$

For each triangle K we compute the vector

$$\int_K f\,N_\alpha^K, \qquad \alpha = 1,2,3$$

and then add these elements in the positions (n_1, n_2, n_3) of the full vector. This process can be done at the same time as the matrix assembly, since it goes triangle by triangle. For the $\mathbb{P}_1$ element, the following extremely simple approximation is enough:

$$\begin{aligned}\int_K f\,N_\alpha^K &\approx \tfrac{1}{3}\sum_{\beta=1}^3 f(\mathbf{p}_\beta^K)\int_K N_\alpha^K = \frac{|\det \mathbf{B}_K|}{3}\sum_{\beta=1}^3 f(\mathbf{p}_\beta^K)\int_{\widehat{K}} \widehat{N}_\alpha \\ &= \frac{|\det \mathbf{B}_K|}{18}\sum_{\beta=1}^3 f(\mathbf{p}_\beta^K).\end{aligned}$$

Note that three integrals related to the element K are approximated by the same number. What we have done is approximating f by a function that is constant over each triangle: the constant value on the triangle is the average of the values on its vertices. Otherwise, we can try a quadrature rule to approximate the integrals. It is important at this stage to note that the choice of an adequate quadrature rule has to take into account two facts:

- it has to be precise enough not to lose the good properties of the finite element method, but
- it has to be simple enough not to be wasting efforts in computing with high precision a quantity that is only needed with some precision.

In principle, we could think of using a very precise rule to compute the integrals as exactly as possible. This is overdoing it and forgetting one of the most important principles of well–understood scientific computing: errors from different sources have to be balanced. It doesn't make much sense to spend time in computing exactly a quantity when that number is to be used in the middle of many approximate computations.

The presence of Neumann boundary conditions imposes the computation of the following integrals

$$\int_{\Gamma_N} g_1\,\varphi_i.$$

This process is made separately to the ones of computing domain integrals for the matrices and the source terms. First of all we have to decompose the Neumann boundary in the set of edges that lie on it (for that we will need a numbering of the Neumann edges):

$$\int_{\Gamma_N} g_1\,\varphi_i = \sum_L \int_L g_1\,\varphi_i.$$

Note first that unless $\mathbf{p}_i$ is on the Neumann boundary, this integral vanishes.

Next, for each edge consider the two vertices that delimit it: $\mathbf{p}_1^L$ and $\mathbf{p}_2^L$. As we had with triangular elements, we will need the relation between the extremal points of each Neumann edge and the global numbering. If

$$\mathbf{p}_1^L = (x_1, y_1), \qquad \mathbf{p}_2^L = (x_2, y_2),$$

the function

$$[0,1] \ni t \longmapsto \phi_L(t) = (1-t)\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + t\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

is a parameterization of the segment L. We now consider the following two functions

$$\psi_1 = 1 - t, \qquad \psi_2 = t.$$

They are just the nodal basis functions on the reference element $[0,1]$ for the space of linear polynomials in one dimension. It is simple to see that

$$(\varphi_i \circ \phi_L)(t) = \begin{cases} \psi_1(t), & \text{if } \mathbf{p}_i = \mathbf{p}_1^L, \\ \psi_2(t), & \text{if } \mathbf{p}_i = \mathbf{p}_2^L, \\ 0, & \text{otherwise.} \end{cases}$$

The integrals to be computed are

$$\int_L g_1 \varphi_{n_\alpha} = \operatorname{length} L \int_0^1 (g_1 \circ \phi_L)(t)\psi_\alpha(t)\mathrm{d}t, \qquad \alpha = 1, 2$$

(as before n_α denotes the global index for the local node α). We can the use numerical quadrature for this line integral. Alternatively we can approximate

$$\int_L g_1\,\varphi_{n_\alpha} \approx \tfrac{1}{2}\left(g_1(\mathbf{p}_1^L) + g_1(\mathbf{p}_2^L)\right)\int_L \varphi_i = \frac{\operatorname{length} L}{4}\left(g_1(\mathbf{p}_1^L) + g_1(\mathbf{p}_2^L)\right), \qquad \alpha = 1, 2.$$

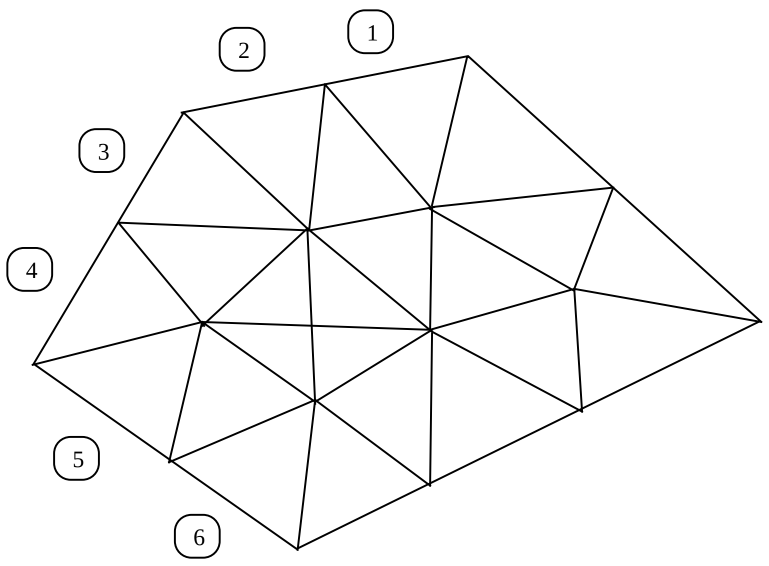
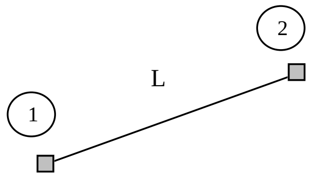


Figure 2.5: A numbering of Neumann edges/elements



local		global
1	$\leftrightarrow$	5
2	$\leftrightarrow$	10

Figure 2.6: The 2nd Neumann edges and its numberings. For this edge, $n_1 = 5$ and $n_2 = 10$.

2 A taste of the theory

2.1 Abstract frame

Because many of the ideas that we will develop on and on in this course are quite independent from the particular problem, let us rewrite everything in a little more abstract form. We have two spaces

$$V = H^1(\Omega), \qquad V_0 = H^1_{\Gamma_D}(\Omega),$$

a bilinear form (related only to the partial differential operator)

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v + c \int_\Omega u\, v$$

and a linear form where most of the data appear

$$\ell(v) = \int_\Omega f\, v + \int_{\Gamma_N} g_1\, v.$$

Finally there is a linear operator γ that serves us to describe the essential conditions: for us γu is the value of u on the boundary Γ_D. Notice that

$$V_0 = \{v \in V \,|\, \gamma v = 0\}.$$

The problem admits then this simple form

$$\left[\begin{array}{l} \text{find } u \in V, \text{ such that} \\[1ex] \gamma u = g_0, \\[1ex] a(u,v) = \ell(v), \quad \forall v \in V_0 \end{array}\right. .$$

Only when $g_0 = 0$ (or when there's no Γ_D and the whole boundary is a Neumann boundary), the problem reduces to an even simpler one

$$\left[\begin{array}{l} \text{find } u \in V_0 \text{ such that} \\[1ex] a(u,v) = \ell(v), \quad \forall v \in V_0. \end{array}\right.$$

Therefore, when the essential condition is homogeneous (or when there is no essential condition), the set where we look for u and the test space are the same. In other cases, the restriction imposed to the tests v is the homogeneous version of the essential condition.

2.2 Well–posedness

Let us recall that the natural norm in our space $V = H^1(\Omega)$ was

$$\|u\| = \|u\|_{1,\Omega} = \left(\int_\Omega |\nabla u|^2 + \int_\Omega |u|^2 \right)^{1/2}.$$

There are several conditions that ensure the well–posedness of the problem

$$\left[\begin{array}{l} \text{find } u \in V, \text{ such that} \\[1ex] \gamma u = g_0, \\[1ex] a(u,v) = \ell(v), \quad \forall v \in V_0, \end{array}\right.$$

or of its homogeneous version ($g_0 = 0$)

$$\left[\begin{array}{l} \text{find } u \in V_0 \text{ such that} \\[1ex] a(u,v) = \ell(v), \quad \forall v \in V_0. \end{array}\right.$$

Well–posedness means existence and uniqueness of solution and continuity of the solution with respect to the data.

Let us first list the properties that are satisfied in all the situations we are addressing in this course:

- V is a Hilbert space (a vector space, with an inner product so that the space is complete with respect to the associate norm)[1],
- V_0 is a closed subspace of V,
- the bilinear form a is continuous in V, that is, there exists $M > 0$ such that
$$|a(u,v)| \leq M\|u\|\,\|v\|, \qquad \forall u, v \in V,$$
- the linear form ℓ is continuous
$$|\ell(v)| \leq C_\ell \|v\|, \qquad \forall v \in V.$$

As already mentioned, all of these properties are satisfied in our case. In fact
$$C_\ell^2 \leq \int_\Omega |f|^2 + C_\Omega \int_{\Gamma_N} |g_1|^2.$$
There is a last property, called **ellipticity**, which reads: there exists $\alpha > 0$ such that
$$a(v,v) \geq \alpha \|v\|^2, \qquad \forall v \in V_0.$$
Note that the property is only demanded on the set V_0. In our case it is not satisfied in all situations. In fact, it is satisfied in all but one case:

- if $c > 0$ the property is satisfied with α depending only on c,
- if $c = 0$ and $\text{length}\,\Gamma_D > 0$, the property is satisfied with α depending on Ω and on the partition of the boundary in Dirichlet and Neumann parts.

If all the properties mentioned above hold, then the problem
$$\left[\begin{array}{l} \text{find } u \in V_0 \text{ such that} \\ a(u,v) = \ell(v), \quad \forall v \in V_0, \end{array}\right.$$
has a unique solution and
$$\|u\| \leq C_\ell/\alpha.$$
If $g_0 \neq 0$ then the problem
$$\left[\begin{array}{l} \text{find } u \in V, \text{ such that} \\ \gamma u = g_0, \\ a(u,v) = \ell(v), \quad \forall v \in V_0 \end{array}\right.$$
has a unique solution if there exists a $u_0 \in V$ such that $\gamma u_0 = g_0$. In that case, the continuity of the solution with respect to the data has a more complicated expression
$$\|u\| \leq C_\ell/\alpha + (M/\alpha + 1) \inf \Big\{ \|u_0\| \,\Big|\, \gamma u_0 = g, \Big\}.$$

[1]Maybe this sentence looks too hard. You should know what a vector space and also what an inner (or scalar) product is. When you have an inner product, you have an associated norm and with it a concept of convergence of sequences of elements of V. Completeness is a property that ensures that all Cauchy sequences have a limit. In essence, it means that convergence has to happen inside the space. We cannot have a sequence of elements of V converging to something that is not in V.

Remark. For the pure Neumann problem with $c = 0$

$$\left[\begin{array}{ll} -\Delta u = f, & \text{in } \Omega, \\ \partial_n u = g_1, & \text{on } \Gamma, \end{array}\right.$$

we cannot verify the conditions to prove existence and uniqueness. In fact existence and uniqueness is not completely guaranteed. First of all, because of the divergence theorem we must have

$$\int_\Omega \Delta u = \int_\Omega \mathrm{div}(\nabla u) = \int_\Gamma \partial_n u$$

and therefore the data have to satisfy the compatibility condition

$$\int_\Omega f + \int_\Gamma g_1 = 0.$$

If this condition is satisfied, there is more that one solution, since constant functions satisfy the homogeneous equation

$$\left[\begin{array}{ll} -\Delta u = 0, & \text{in } \Omega, \\ \partial_n u = 0, & \text{on } \Gamma. \end{array}\right.$$

□

2.3 Galerkin methods

A Galerkin method for the problem

$$\left[\begin{array}{l} \text{find } u \in V_0 \text{ such that} \\ a(u, v) = \ell(v), \quad \forall v \in V_0, \end{array}\right.$$

consists of the choice of a finite dimensional space

$$V_h^0 \subset V_0$$

and on the consideration of the discrete problem

$$\left[\begin{array}{l} \text{find } u_h \in V_h^0 \text{ such that} \\ a(u_h, v_h) = \ell(v_h), \quad \forall v_h \in V_h^0. \end{array}\right.$$

The $\mathbb{P}_1$ finite element method for the reaction–diffusion problem with homogeneous Dirichlet conditions is therefore an example of Galerkin method[2].

The Galerkin equations are equivalent to a linear system. Let us do here the detailed argument, although you will see that we already did exactly this in Section 3 of the previous lesson.

[2]Galerkin comes from Boris Galerkin. A good pronunciation of the word would be something more like *Galyorkin*, with emphasis on the *lyor* syllable. Most English speakers pronounce it however as if it was an English word

First we need a basis of V_h^0: $\{\varphi_i \,|\, i \in \mathrm{Ind}\}$. The index set Ind is now anything you want to number the finite basis of the set. In general we would number form one to the dimension of the space, but in our model problem the numbering proceeds from eliminating some indices from a wider numbering. Then we notice that the abstract set of equations

$$a(u_h, v_h) = \ell(v_h), \quad \forall v_h \in V_h^0$$

is equivalent to

$$a(u_h, \varphi_i) = \ell(\varphi_i), \quad \forall i \in \mathrm{Ind}.$$

Finally, we decompose

$$u_h = \sum_{j \in \mathrm{Ind}} u_j \varphi_j$$

and substitute this expression above to obtain the linear system

$$\sum_{j \in \mathrm{Ind}} a(\varphi_j, \varphi_i) u_j = \ell(\varphi_i), \qquad i \in \mathrm{Ind}.$$

There are as many unknowns as there are equations here. Note that in the general case, the values u_j are not nodal values, since an arbitrary basis of a linear space has nothing to do with nodes or evaluations of functions.

If the hypotheses of Section 2.2 hold, this system has a unique solution. Furthermore we have the following result, which is popularly referred to as Céa's Lemma[3]:

$$\|u - u_h\| \le \frac{M}{\alpha} \inf \Big\{ \|u - v_h\| \,\Big|\, v_h \in V_h^0 \Big\}.$$

The result might not seem to say much at first sight. There are however some aspects that have to be remarked here:

- The result gives an upper bound of the error between the exact solution u and the approximate solution u_h (the finite element solution) and this error bound is measured in the *energy* norm and not in any other one.

- The term
$$\inf \Big\{ \|u - v_h\| \,\Big|\, v_h \in V_h^0 \Big\}$$
is just an approximation error, completely unrelated to the original problem. It measures how well the (unknown) exact solution can be approximated by elements of the space where we are looking for the solution. Because of how this term is estimated in particular situations (in FEM, for instance) many people call this an interpolation error. We will see a bit of this on the following section. This approximation error is measured also in the energy norm, of course.

- The only other constants in the inequality depend on the problem, but not on data. Note however that complicated solutions (solutions that vary a lot, or that have large gradients, or anything you can think of as difficult to grasp with a simple

[3]Céa, as in Jean Céa. French. Do you best with the pronunciation of the name.

approximation) will not necessarily be approximated as well as simple smooth solutions. Since we do not know the solution (by definition, it is the unknown), how can we have an idea of this error? The answer is the lifetime work of numerical analysts and people who do scientific computation. Just three ideas:

– for simple smooth solutions, numerical analysis shows usually how error behaves quite precisely, which gives us a hint of the best possible behavior of our method;
– PDE theory sometimes helps in understanding where things can go wrong and we can do some effort in concentrating approximation in that area;
– finally, there is a whole new (new as in only twenty years old or so) branch of computational knowledge related to error estimation and adaptivity, allowing you to improve your computations with information you harvest from the already performed computations.

The theoretical frame for the case with non–homogeneous Dirichlet conditions is somewhat more delicate, because we have to go someplace more abstract to write correctly the approximation of the condition

$$u = g_0, \qquad \text{on } \Gamma_D$$

by

$$u_h(\mathbf{p}) = g_0(\mathbf{p}), \qquad \forall \mathbf{p} \text{ Dirichlet node},$$

without making any use of the particularities of the finite element space $\mathbb{P}_1$. This can be done in several ways, and we are not going to detail them. Particularized to FEM the result will look like this

$$\|u - u_h\| \le (1 + \frac{M}{\alpha}) \inf \Big\{ \|u - v_h\| \,\Big|\, v_h \in V_h, \quad v_h(\mathbf{p}) = g_0(\mathbf{p}), \quad \forall \mathbf{p} \text{ Dirichlet node} \Big\}.$$

Note that the approximation error in the right–hand side includes the imposition of the discrete essential boundary condition.

2.4 Convergence of the $\mathbb{P}_1$ finite element method

How does all of this work for the $\mathbb{P}_1$ finite element? Let us go back to the case with homogeneous boundary conditions. As mentioned, the error can be bounded as

$$\|u - u_h\|_{1,\Omega} \le \frac{M}{\alpha} \inf \Big\{ \|u - v_h\|_{1,\Omega} \,\Big|\, v_h \in V_h^0 \Big\}.$$

Let us emphasize again that the norm for measuring the error is imposed by the problem (see Section 2.1). Assume now that u is a well–behaved function. For example, that it is continuous. The we can construct a function $\pi_h u$ by taking nodal values of u on the vertices of the triangulation and creating with them an element of V_h. This is, obviously, interpolation in V_h, that is, interpolation with continuous piecewise linear functions. Because of the Dirichlet boundary condition u vanishes on Dirichlet nodes, and so does consequently $\pi_h u$. Therefore $\pi_h u \in V_h^0$ and we can use the bound

$$\|u - u_h\|_{1,\Omega} \le \frac{M}{\alpha} \|u - \pi_h u\|_{1,\Omega}.$$

We have therefore bounded the error of the finite element method by the error of interpolation of the exact solution in the finite element space. A nice thing about this interpolation process is the fact that it is done triangle–by–triangle, so actually, the global error for interpolation is the sum of the errors that we have done element–by–element.

In basic courses on numerical methods you will have seen that it is possible to estimate the error of interpolation without knowing the solution, but that this bound of the error is proportional to some quantity depending on a high order derivative of the function that is interpolated. You will have seen this in one space dimension. In several space dimensions, it is a bit more difficult but not so much. The result is the following: there exists a constant C that depends on the minimum angle of the triangulation such that

$$\|u - \pi_h u\|_{1,\Omega} \le C h \Big(\int_\Omega |\partial_{xx} u|^2 + |\partial_{xy} u|^2 + |\partial_{yy} u|^2 \Big)^{1/2},$$

where h is the size of the longest edge of the triangulation. The expression on the right–hand side is an example of a Sobolev seminorm. It is denoted usually as

$$|u|_{2,\Omega} = \Big(\int_\Omega |\partial_{xx} u|^2 + |\partial_{xy} u|^2 + |\partial_{yy} u|^2 \Big)^{1/2}.$$

The whole bound is

$$\|u - u_h\|_{1,\Omega} \le C' h |u|_{2,\Omega}$$

with the constant C' depending on the coefficients of the problem, on the geometry of the physical setting and on the smallest angle. If the triangles are very flat (the ratio between the longest edge and the inradius[4] is very small), the constant gets to be very large.

First of all, lest us remark that the error bound requires the second derivatives of the solution to be square–integrable, which is not always the case. Second, note that if u is a polynomial of degree one, this error bound is zero and u_h is exactly u. You can use this as a way of constructing exact solutions to validate your own coding of the method. Third, the fact that the bound is proportional to h makes the method a **method of order one**. This means that if you make the longest edge half its size, you should only expect the error to be divided by two. Note that the argument on error decrease is done on the bound, since the error itself is unknown. In fact the error could decrease much faster, but in principle you should not expect this to happen.

3 Quadratic elements

Its very low order makes the $\mathbb{P}_1$ method not very attractive. Just to expect having an additional digit in precision you should have edges ten times shorter, which amounts to

[4]Inradius is the geometric term for the radius of the inscribed circumference.

increasing dramatically the number of unknowns. Instead, it is often recommended to use a higher order method, which is exactly what we are going to do right now.

3.1 Local and global descriptions

Let us consider the space of polynomials in two variables with degree at most two

$$\mathbb{P}_2 = \Big\{ a_0 + a_1\, x + a_2\, y + a_2\, x^2 + a_4\, y^2 + a_5\, x\, y \;\Big|\; a_0, \ldots, a_5 \in \mathbb{R} \Big\}.$$

An element of $\mathbb{P}_2$ is determined by six independent parameters (the quantities a_i), that is, the space $\mathbb{P}_2$ has dimension equal to six. Let us take a triangle K and let us mark six points as **nodes**:

- the three vertices of the triangle,
- the midpoints of the three edges.

The following result is easy to prove: *a function in $\mathbb{P}_2$ is uniquely determined by its values on the six nodes of the triangle.* Take now two points $\mathbf{p}_1$ and $\mathbf{p}_2$. The function

$$[0,1] \ni t \longmapsto (1-t)\,\mathbf{p}_1 + t\,\mathbf{p}_2$$

parameterizes linearly the segment between these two points. If $p \in \mathbb{P}_2$, then a simple computation shows that

$$p((1-t)\mathbf{p}_1 + t\,\mathbf{p}_2) \in \mathbb{P}_2(t) = \Big\{ b_0 + b_1\, t + b_2\, t^2 \;\Big|\; b_0, b_1, b_2 \in \mathbb{R} \Big\},$$

that is, seen on any segment (on any straight line actually) an element of $\mathbb{P}_2$ is a parabolic function, which, as everyone knows, is determined by three different points. Therefore *the value of a function in $\mathbb{P}_2$ on an edge of the triangle is uniquely determined by its three values on the nodes that lie on that edge* (two vertices and one midpoint).

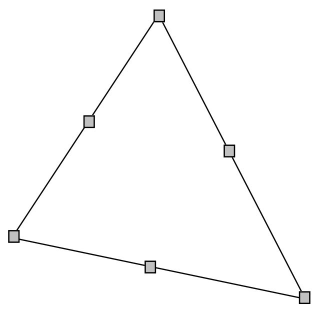


Figure 2.7: the nodes of a $\mathbb{P}_2$ triangle

Because of this last property, we can glue together two $\mathbb{P}_2$ triangles as we did in the $\mathbb{P}_1$ case. Take a triangulation in the usual conditions, fix values of a function in all the nodes (vertices and midpoints) and on each triangle construct the only function in $\mathbb{P}_2$

that matches the given values. The resulting function is continuous. In fact it is a general element of the space

$$V_h = \Big\{ u_h \in \mathcal{C}(\overline{\Omega}) \,\Big|\, u_h|_K \in \mathbb{P}_2, \quad \forall K \in \mathcal{T}_h \Big\}.$$

All the arguments exposed in Lesson 1 hold also here. The dimension of this space is

$$\dim V_h = \#\{\text{vertices}\} + \#\{\text{edges}\},$$

since there is one midpoint per edge.

As before, we give a global numbering to the set of nodes (vertices and midpoints of edges): $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ and construct the functions $\varphi_i \in V_h$ satisfying

$$\varphi_i(\mathbf{p}_j) = \delta_{ij} = \left\{ \begin{array}{ll} 1, & j = i, \\ 0, & j \neq i. \end{array} \right.$$

For the same reasons as in the $\mathbb{P}_1$ case, these functions constitute a basis of V_h and any function of this space can be expressed as

$$u_h = \sum_{j=1}^{N} u_h(\mathbf{p}_j) \varphi_j.$$

There are two types of basis functions now:

- those associated to vertices, whose support is the set of triangles surrounding the vertex,
- those associated to midpoints, whose support is the set of two triangles (only one if the edge is on the boundary) that share the edge.

Take the usual triangulation and make yourself some drawing of the form of the supports of the nodal basis functions.

The concept of a Dirichlet node is the same: it is any node on a Dirichlet edge, Dirichlet edges being edges on the Dirichlet boundary Γ_D. The following result is then a straightforward consequence of the fact that value on edges is determined by degrees of freedom on edges:

$v_h \in V_h$ vanishes on Γ_D if and only if it vanishes on all Dirichlet nodes.

Therefore, it is very simple to construct a basis of

$$V_h^{\Gamma_D} = V_h \cap H^1_{\Gamma_D}(\Omega) = \{ v_h \in V_h \,|\, v_h = 0, \quad \text{on } \Gamma_D \}$$

by simply ignoring nodal basis functions φ_i associated to Dirichlet nodes. Can you notice that I am copy–pasting formulas from Lesson 1?

Very important. The whole of Section 3 in Lesson 1 can be read with these adapted concepts. There's nothing new at all, but the different concepts of local spaces and nodes. *You should have a detailed look again at that section* to convince yourself that this is so. In particular pay attention to mass and stiffness matrices and note that the number of adjacent nodes for each node is increased with respect to $\mathbb{P}_1$ triangles (we will explore this in an exercise). $\square$

3.2 The reference element

If we want to implement the $\mathbb{P}_2$ we need to compute the usual integrals for the mass and stiffness matrices (an also, of course, the right–hand sides, that include the influence of data). For that, we need a way of evaluating the nodal basis functions on each triangle.

Since the argument is, again, exactly the same as for the $\mathbb{P}_1$ element, let us work now in the opposite sense. In the reference triangle we mark the six nodes as shown in Figure 2.8. As usual (ξ, η) are the coordinates in the reference configuration.

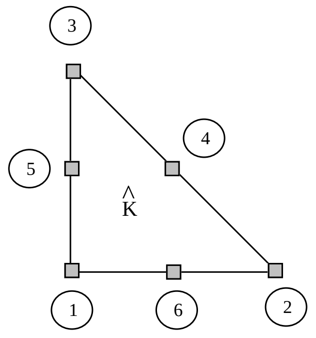


Figure 2.8: The $\mathbb{P}_2$ reference triangle

Each of the functions

$$\widehat{N}_1 = (1-\xi-\eta)(1-2\xi-2\eta), \qquad \widehat{N}_2 = \xi(2\xi-1), \qquad \widehat{N}_3 = \eta(2\eta-1)$$

$$\widehat{N}_4 = 4\xi\eta, \qquad \widehat{N}_5 = 4\eta(1-\xi-\eta), \qquad \widehat{N}_6 = 4\xi(1-\xi-\eta)$$

takes the unit value on the corresponding node (the one numbered with the subindex) and vanishes in all other five nodes.

Let's do again some copy–pasting. The functions

$$N_\alpha^K = \widehat{N}_\alpha \circ F_K^{-1}, \qquad \alpha = 1, \ldots, 6$$

have the same property as the $\widehat{N}_\alpha$ functions, only on the triangle K, that is mapped from $\widehat{K}$ via the linear transformation F_K. These functions are polynomials of degree two (do you see why?) and therefore

$$N_\alpha^K = \varphi_{n_\alpha}, \qquad \text{in } K$$

where n_α is the global index corresponding to the local node α in K. Here is again the formula for the gradient

$$\nabla N_\alpha^K = \mathbf{B}_K^{-\top}\Big((\widehat{\nabla}\widehat{N}_\alpha)\circ F_K^{-1}\Big).$$

Note that now $\widehat{\nabla}\widehat{N}_\alpha$ is not constant, so the inverse transformation F_K^{-1} is needed also to evaluate the gradient.

We then compute the local matrices, which are 6×6 matrices,

$$\int_K \nabla N_\beta^K \cdot \nabla N_\alpha^K \qquad \text{and} \qquad \int_K N_\beta^K\, N_\alpha^K$$

put the elements in the global positions

$$\int_K \nabla\varphi_{n_\beta}\cdot\nabla\varphi_{n_\beta} \qquad \text{and} \qquad \int_K \varphi_{n_\beta}\,\varphi_{n_\alpha}$$

and add the contributions of all triangles to assemble the full stiffness and mass matrices.

3.3 Convergence

The general error bound

$$\|u-u_h\|_{1,\Omega} \le (1+\frac{M}{\alpha})\inf\Big\{\|u-v_h\|_{1,\Omega}\,\Big|\, v_h\in V_h,\ v_h(\mathbf{p})=g_0(\mathbf{p}),\ \forall\mathbf{p}\text{ Dirichlet node}\Big\}.$$

still holds here. In the case of homogeneous Dirichlet conditions, we can use the same arguments as in the preceding section to obtain a full bound like

$$\|u-u_h\|_{1,\Omega} \le Ch^2|u|_{3,\Omega},$$

where:

- the constant C depends on the PDE operator, on the geometry and on the smallest angle (becoming worse as the triangles become flatter)
- the new Sobolev seminorm $|u|_{3,\Omega}$ uses the third order partial derivatives of u.

The result is valid only when this last seminorm is finite, which is much more to require than what we had at the beginning. Note that the **order two** in energy norm ($H^1(\Omega)$ norm) is good news, since using smaller triangles really pays off and the gain of precision is due to be much faster than in the $\mathbb{P}_1$ case. In the final exercise of this section we will explore what's the price to be paid (there's no free lunch, you know).

4 Cubic elements and static condensation

4.1 The $\mathbb{P}_3$ element

Can we do better than order two? The answer is yes, and besides, it is easy to do better. We will just give some hints on the order three case, because a new thing appears and we really want to deal with new ideas instead of doing the same thing over and over. Look first at Figure 2.9. There are ten nodes in the triangle:

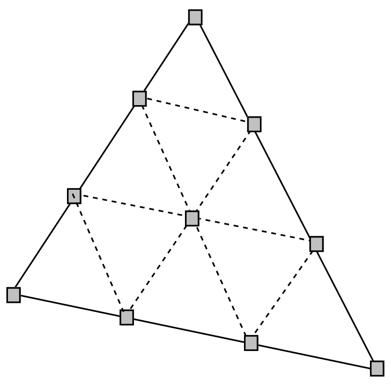


Figure 2.9: The $\mathbb{P}_3$ triangle

- the three vertices,
- two points per side, at relative distances 1/3 and 2/3 from the vertices,
- the barycenter, which is computed by averaging the coordinates of the three vertices

$$\tfrac{1}{3}\mathbf{v}_1^K + \tfrac{1}{3}\mathbf{v}_2^K + \tfrac{1}{3}\mathbf{v}_3^K.$$

Note also that each edge has four nodes on it. The local space is that of polynomials of degree up to three $\mathbb{P}_3$. Instead of writing a general element of this space, let us list the monomials that are used:

$$\begin{array}{llll} 1 & & & \\ x & y & & \\ x^2 & xy & y^2 & \\ x^3 & x^2y & xy^2 & y^3 \end{array}$$

Count them. Ten monomials (i.e., ten coefficients) and ten nodes. Well, that's a surprise! Two other surprises:

- *a function in $\mathbb{P}_3$ is uniquely determined by its values on the ten nodes of the triangle,*
- *the value of a function in $\mathbb{P}_3$ on an edge of the triangle is uniquely determined by its four values on the nodes that lie on that edge.*

Note that a $\mathbb{P}_3$ function restricted to a segment (straight line) is a cubic function of one variable.

We are almost done here. We can construct the spaces V_h, the nodal basis functions φ_i, the subspace $V_h^{\Gamma_D}$ by eliminating Dirichlet nodes, etc. The dimension of V_h is

$$\#\{\text{vertices}\} + 2\,\#\{\text{edges}\} + \#\{\text{triangles}\}.$$

4.2 Static condensation

There is however a new entity here, and that's the very isolated interior node. I say isolated since that node is only adjacent to the other nodes of the same triangle. This has some consequences at the practical level that we are going to explore right now.

Let φ_i be the nodal basis function associated to a node that is the barycenter of the triangle K. Then $\operatorname{supp}\varphi_i = K$. Therefore

$$a(\varphi_j,\varphi_i) = \int_K \nabla\varphi_j\cdot\nabla\varphi_i + c\int_K \varphi_j\,\varphi_i, \qquad \forall j,$$

and

$$\ell(\varphi_i) = \int_K f\,\varphi_i$$

(do you see why there is no Neumann term here?) which means that once we have gone through the element K in the assembly process we will have done the $i-$th row of the system, with no contributions from other elements. The idea of **static condensation** is simple: get rid of that equation and unknown in the same process of assembly.

Let us consider that the $0-$th node is the barycenter of K. Let $\mathbf{K}^K$ and $\mathbf{b}^K$ be the local matrix and right–hand side contributions from the triangle K

$$k^K_{\alpha\beta} = \int_K \nabla N^K_\beta\cdot\nabla N^K_\alpha + c\int_K N^K_\beta\,N^K_\alpha, \qquad b^K_\alpha = \int_K f\,N^K_\alpha, \qquad \alpha,\beta = 0,\dots,9.$$

Now we decompose the matrix and the vector into blocks, separating the contribution from the interior node from all others:

$$\begin{bmatrix} \mathbf{K}^K_{00} & \mathbf{K}^K_{01} \\ \mathbf{K}^K_{10} & \mathbf{K}^K_{11}\end{bmatrix}, \qquad \begin{bmatrix} \mathbf{b}^K_0 \\ \mathbf{b}^K_1\end{bmatrix},$$

with

$$\mathbf{K}^K_{00} = \begin{bmatrix} k^K_{0,0}\end{bmatrix}, \qquad \mathbf{K}^K_{01} = \begin{bmatrix} k^K_{0,1} & \dots & k^K_{0,9}\end{bmatrix}, \qquad \mathbf{b}^K_0 = \begin{bmatrix} b^K_0\end{bmatrix}$$

$$\mathbf{K}^K_{10} = \begin{bmatrix} k^K_{1,0} \\ \vdots \\ k^K_{9,0}\end{bmatrix}, \qquad \mathbf{K}^K_{11} = \begin{bmatrix} k^K_{1,1} & \dots & k^K_{1,9} \\ \vdots & & \vdots \\ k^K_{9,1} & \dots & k^K_{9,9}\end{bmatrix}, \qquad \mathbf{b}^K_1 = \begin{bmatrix} b^K_1 \\ \vdots \\ b^K_9\end{bmatrix}.$$

You will be wondering why are we calling matrices to blocks 1×1 (scalars) and 1×9 or 9×1 (row or column vectors). The reason is twofold: first, the role these scalars and vectors are playing are the ones of blocks in a matrix so we'd better use block notation, independent of their shape; second, we will thus be able to recycle all that comes right now for more complicated situations.

The (ten) equations related to the nodes of the element K are

$$\begin{aligned}&\mathbf{K}^K_{00}\mathbf{u}^K_0 + \mathbf{K}^K_{01}\mathbf{u}^K_1 = \mathbf{b}^K_0,\\ &\mathbf{K}^K_{10}\mathbf{u}^K_0 + (\mathbf{K}^K_{11}+\mathbf{A})\mathbf{u}^K_1 + \mathbf{B}\mathbf{u}_{\mathrm{other}} = \mathbf{b}^K_1 + \mathbf{b}.\end{aligned}$$

The unknowns are separated in the same blocks (1 and 9) and are denoted with local numbering, that is $\mathbf{u}^K_0$ is the unknown associate to the barycenter of K and $\mathbf{u}^K_1$ is the column vector of the nine unknowns associated to all the other nodes on K.

- The matrix $\mathbf{A}$ includes all contributions from other elements to nodes of K. It will be added in the assembly process when we go through these elements.
- The block $\mathbf{B}$ includes all contributions from other triangles to other unknowns (generically written as $\mathbf{u}_{\rm other}$), that is, unknowns on nodes that are not on K but are adjacent to those on K.
- Finally, $\mathbf{b}$ includes all contributions from other triangles and possibly also from Neumann edges, to the right–hand side.

Now we can write $\mathbf{u}_0^K$ (which, in this case, is just the unknown corresponding to the barycenter of K) as

$$\mathbf{u}_0^K = \left(\mathbf{K}_{00}^K\right)^{-1}\mathbf{b}_0^K - \left(\mathbf{K}_{00}^K\right)^{-1}\mathbf{K}_{01}^K\mathbf{u}_1^K$$

and substitute this expression in the block of the remaining equations for the triangle K (the non–interior unknowns), obtaining

$$\left(\mathbf{K}_{11}^K - \mathbf{K}_{10}^K\left(\mathbf{K}_{00}^K\right)^{-1}\mathbf{K}_{01}^K + \mathbf{A}\right)\mathbf{u}_1^K + \mathbf{B}\mathbf{u}_{\rm other} = \mathbf{b}_1^K - \mathbf{K}_{10}^K\left(\mathbf{K}_{00}^K\right)^{-1}\mathbf{b}_0^K + \mathbf{b}$$

This means that instead of assembling the full (10×10) block from K and its corresponding right–hand side, we can forget about the interior nodes (just one) on condition of assembling

$$\mathbf{K}_{11}^K - \mathbf{K}_{10}^K\left(\mathbf{K}_{00}^K\right)^{-1}\mathbf{K}_{01}^K, \qquad \mathbf{b}_1^K - \mathbf{K}_{10}^K\left(\mathbf{K}_{00}^K\right)^{-1}\mathbf{b}_0^K$$

instead of the original matrix. Once we have solved the system, the interior variables are solved using the local equations

$$\mathbf{K}_{00}^K\mathbf{u}_0^K + \mathbf{K}_{01}^K\mathbf{u}_1^K = \mathbf{b}_0^K,$$

that work element–by–element.

Remark. This is a method for implementing the $\mathbb{P}_3$ FEM in a way that the information of the interior nodes is incorporated to the assembly process directly without having to use the corresponding unknown. This doesn't mean that the node is not there. We only compute it separately after having added its contribution to assembly directly. So don't confuse this, which is nothing else than an implementation trick, with some finite elements (in the class of the so–called exotic elements) that avoid interior nodes. $\square$

Maybe I've left you wondering about that strange Algebra in the assembly process and it somehow rings a bell. It should. Write the extended matrix

$$\left[\begin{array}{cc|c} \mathbf{K}_{00}^K & \mathbf{K}_{01}^K & \mathbf{b}_0^K \\ \mathbf{K}_{10}^K & \mathbf{K}_{11}^K & \mathbf{b}_1^K \end{array}\right]$$

and apply Gaussian block elimination (the $\mathbf{K}_{00}^K$ block is just 1×1, so this is just Gauss elimination) you obtain

$$\left[\begin{array}{cc|c} \mathbf{K}_{00}^K & \mathbf{K}_{01}^K & \mathbf{b}_0^K \\ \mathbf{0} & \mathbf{K}_{11}^K - \mathbf{K}_{10}^K\left(\mathbf{K}_{00}^K\right)^{-1}\mathbf{K}_{01}^K & \mathbf{b}_1^K - \mathbf{K}_{10}^K\left(\mathbf{K}_{00}^K\right)^{-1}\mathbf{b}_0^K \end{array}\right].$$

Da, daaaa! There they are. The blocks you wanted. Again, our diagonal block was a scalar, so this was easy. What would have happened if it was a matrix? Do you have to compute that inverse and apply all that Algebra? No, you don't. Gauss block elimination is a nice way of writing the result of Gauss elimination. The point is you apply row elimination to create all those zeros, with no row changes and without trying to create any other zeros. Blocks of the form

$$\mathbf{K}_{11}^{K} - \mathbf{K}_{10}^{K} \left(\mathbf{K}_{00}^{K}\right)^{-1} \mathbf{K}_{01}^{K}$$

are called Schur complements. If the original matrix is symmetric and positive definite, they are still symmetric and positive definite.

4.3 Convergence, $\mathbb{P}_4$ and higher

We haven't mentioned convergence of the $\mathbb{P}_3$ method yet. In the best possible conditions, this is a method of order three in the $H^1(\Omega)$ Sobolev norm:

$$\|u - u_h\|_{1,\Omega} \le C h^3 |u|_{4,\Omega}$$

(can you guess what's in $|u|_{4,\Omega}$?). These best possible conditions include the fact that triangles do not become too flat, since the constant C becomes worse and worse as triangles get flatter and flatter. Note that if you apply static condensation to the $\mathbb{P}_3$ you complicate the assembly process but you end up with a system of order

$$\#\{\text{vertices}\} + 2\,\#\{\text{edges}\}$$

(minus the number of Dirichlet nodes), which is smaller than the one you obtain without condensation. There is an additional advantage of applying condensation. With the usual information of a grid generator (you will have to read the Appendix for that) you can easily construct a coherent numbering including vertices and edges, which works for $\mathbb{P}_2$ elements. Going from $\mathbb{P}_2$ to $\mathbb{P}_3$ means that you have to double the number of unknowns per edge (which is easy) and add the triangles. The numbering of triangles becomes then relevant. It is not, I insist, for the assembly process. If you apply static condensation, you avoid the unknowns related to barycenter and the numbering of vertices–and–edges is enough for the $\mathbb{P}_3$ element.

The $\mathbb{P}_4$ element is constructed easily following these lines:

- You divide each edge into five equally sized pieces. Then you join these new points on different sides with lines that run parallel to the edges. With that you have created a grid of 15 nodes: three vertices, three points per edge, three interior points, placed on the intersections of the interior lines.
- The space is $\mathbb{P}_4$, which has dimension 15. Everything goes on as usual.
- The three interior nodes can be treated with static condensation: the $\mathbf{K}_{00}^{K}$ blocks are now 3×3 blocks. With this you reduce in three times the number of triangles the size of the global system to be solved without affecting convergence.

- Order of the method is.... four! (That was easy)

It is possible to create $\mathbb{P}_k$ methods for arbitrary k. You will find people around that will assert that these methods are useless or just of theoretical interest. Be warned: maybe they find them useless, but some other people work with really high order methods and find many advantages in them[5]. However, if you go from $\mathbb{P}_4$ upwards, you implement the method in a very different way. Nodal bases are not the best choice in that case and there is a different way of constructing node–free bases. For that you will have to go to specialized literature or take my *Advanced Finite Element* course: it's the first lesson of that course actually.

5 Exercises

E2.1. Basis function for the $\mathbb{P}_2$ element

Try to sketch the form of the nodal basis functions for a $\mathbb{P}_2$ finite element space (similar as Figure 1.8). Note that there are two different types of functions, those associated to vertices and those associated to midpoints of edges.

E2.2. The plane elasticity system

The problem of plane deformations in linear elasticity can be reduced to the variational problem:

$$\left[\begin{array}{l} \text{find } u_1, u_2 \in H^1(\Omega) \text{ such that} \\ u_1 = g_x, \quad u_2 = g_y, \qquad \text{on } \Gamma_D, \\ \displaystyle\int_\Omega \Big((\lambda+2\mu)\frac{\partial u_1}{\partial x} + \lambda\frac{\partial u_2}{\partial y}\Big)\frac{\partial v}{\partial x} + \mu\Big(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\Big)\frac{\partial v}{\partial y} = \int_\Omega v\, f_x + \int_{\Gamma_N} v\, t_x, \quad \forall v \in H^1_{\Gamma_D}(\Omega), \\ \displaystyle\int_\Omega \mu\Big(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\Big)\frac{\partial v}{\partial x} + \Big(\lambda\frac{\partial u_1}{\partial x} + (\lambda+2\mu)\frac{\partial u_2}{\partial y}\Big)\frac{\partial v}{\partial y} = \int_\Omega v\, f_y + \int_{\Gamma_N} v\, t_y, \quad \forall v \in H^1_{\Gamma_D}(\Omega), \end{array}\right.$$

where:

- Ω is the plane section of the cylindrical solid
- Γ_D is the part of the boundary of Ω where we know displacements $g_0 = (g_x, g_y)$
- Γ_N is the part of the boundary where we know normal stresses $t = (t_x, t_y)$
- $f = (f_x, f_y)$ are the volume forces
- λ and $\mu = G$ are the Lamé parameters

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}$$

[5]Be always prepared to find opinionated people in the scientific computing community. Sometimes they are right, sometimes they are partially right, sometimes they are plain wrong

- $H^1_{\Gamma_D}(\Omega) = \{v \in H^1(\Omega) \mid v|_{\Gamma_D} = 0\}$.

We are given a triangulation $\mathcal{T}_h$, the associated $\mathbb{P}_1$ nodal basis functions (φ_i), etc. We call Ind and Dir to the usual index sets. We approximate the pair (u_1, u_2) by the discrete functions

$$u_h^1 = \sum_j u_j^1 \varphi_j, \qquad u_h^2 = \sum_j u_j^2 \varphi_j$$

Alternating tests with the two variational equations, and grouping both unknowns on the same node (u_j^1, u_j^2) prove that the resulting finite element system can be written in the form

$$\sum_{j \in \mathrm{Ind}} A_{ij} \left[\begin{array}{c} u_j^1 \\ u_j^2 \end{array} \right] = F_i + T_i - \sum_{j \in \mathrm{Dir}} A_{ij} \left[\begin{array}{c} g_j^1 \\ g_j^2 \end{array} \right], \qquad i \in \mathrm{Ind}\,.$$

where A_{ij} are 2×2 matrices. What's the dimension of the system? Prove that $A_{ij}^\top = A_{ji}$ and deduce that the system is symmetric.

Foreword. For reasons that are not so easy to explain as many people think, $\mathbb{P}_1$ elements are never used in elasticity problems because their performance is rather bad. Note that in what you have done here $\mathbb{P}_1$ or $\mathbb{P}_k$ is all the same, so you can be applying this to $\mathbb{P}_2$ elements, which work well for this problem.

E2.3. Comparison of $\mathbb{P}_1$ and $\mathbb{P}_2$

Consider the simple triangulation depicted in Figure 2.10

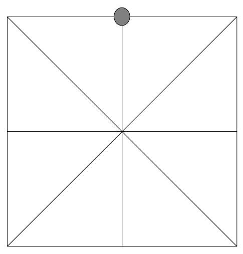


Figure 2.10: A simple triangulation with a marked node

1. If you consider the Neumann problem there (no Dirichlet nodes), how many unknowns are there in the system corresponding to the $\mathbb{P}_2$ method?
2. What are the adjacent nodes to the node that is marked on the figure?
3. A red refinement of a triangle consists of taking the midpoints of the edges and joining them to create four triangles per triangle (see Figure 2.11). If you apply a red refinement to all the elements of the triangulation above and the apply the $\mathbb{P}_1$ element, how many unknowns do you have in the system? Which nodes are adjacent to the same marked nodes in this new triangulation for the $\mathbb{P}_1$ method?

4. **Discussion.** The error of the $\mathbb{P}_2$ method is bounded by something times h^2. The error of the $\mathbb{P}_1$ method on the uniform red refinement is something else times $h/2$. The constant (the unspecified something) for each case is different. In principle, when the triangulation is fine enough h^2 wins over $h/2$ (it is smaller). With the same number of unknowns one method is better than the other. Where's the difference?

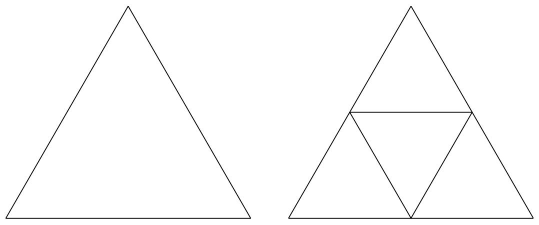


Figure 2.11: A red refinement of a triangle

Lesson 3

New classes of elements

1 The lowest order element on parallelograms

Sometimes dividing a domain into triangles is not the best idea. Some domains, such as rectangles, are much better subdivided into smaller rectangles. Also sometimes triangulations become really messy. For instance Figure 3.1 shows a typical triangular grid of a rectangle as produced by the PDE Toolbox of Matlab. Because of the way these triangulations are produced, working from the boundary to the interior and avoiding very acute angles, they display a very disorganized and non–symmetric pattern. If your problem favors directions, maybe this is not the best way to begin your discretization.

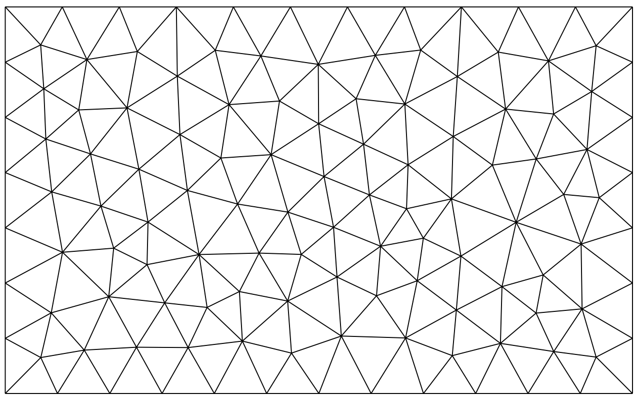


Figure 3.1: A typical triangular grid of a rectangle.

We are going to introduce here finite elements on rectangles and parallelograms. These elements share many features with finite elements on triangles, but there are plenty of novelties. To learn about finite elements on arbitrary quadrilaterals (trapezes and trapezoids) you will have to wait to Lesson 4. They constitute a different species and have to be studied later on to grasp their difficulties.

1.1 The reference space

First of all we need a new polynomial space, which we are going to introduce in reference variables,

$$\mathbb{Q}_1 = \Big\{ a_0 + a_1\xi + a_2\eta + a_3\xi\,\eta \,\Big|\, a_0, a_1, a_2, a_3 \in \mathbb{R} \Big\}.$$

These are polynomials on two variables that are of degree at most one in each variable separately. Note that this space contains $\mathbb{P}_1$. The reference square $\widehat{K}$ is going to be the one with vertices on

$$\widehat{\mathbf{p}}_1 = (-1,-1), \qquad \widehat{\mathbf{p}}_2 = (1,-1), \qquad \widehat{\mathbf{p}}_3 = (1,1), \qquad \widehat{\mathbf{p}}_4 = (-1,1),$$

that is $\widehat{K} = [-1,1] \times [-1,1]$. Note that many books prefer to take the unit square $[0,1] \times [0,1]$ as reference element. Some details change if this choice is made[1]. This is not so important. Note that I've chosen to number the vertices in rotating order. Whether we do this in this way (rotating clockwise or counter–clockwise is immaterial) or in a different way is relevant and we have to be very careful with this. Unlike what happens with triangles, here we really have to know what points are vertices of each edge and we need to fix an order to say that.

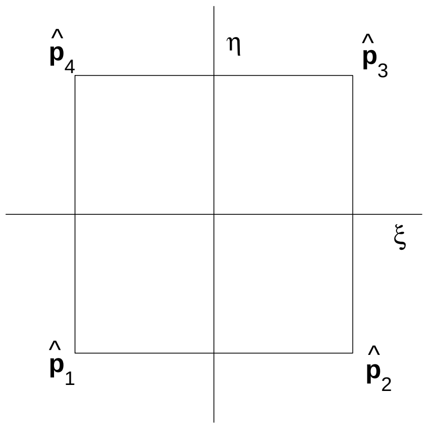


Figure 3.2: The reference square.

Restricted to a horizontal line (η constant) or to a vertical line (ξ constant), functions of $\mathbb{Q}_1$ are polynomials of degree one, that is, seen on horizontal or vertical lines, functions of $\mathbb{Q}_1$ are linear functions. They are however not linear functions (flat plane functions), because of the crossed product $\xi\,\eta$.

Two simple observations, in the line of what we have been doing for triangles:

- the value of an element of $\mathbb{Q}_1$ is uniquely determined by its values on the four vertices of $\widehat{K}$,
- the value of an element of $\mathbb{Q}_1$ on each of the four sides of $\widehat{K}$ is uniquely determined by the value on the extreme points of that side.

[1]In a way, I'm mixing choices in this course, because I chose the unit reference triangle in a form and the reference square in another form.

As usual, we can construct functions $\widehat{N}_\alpha \in \mathbb{Q}_1$ such that

$$\widehat{N}_\alpha(\widehat{\mathbf{p}}_\beta) = \delta_{\alpha\beta}, \qquad \alpha, \beta = 1, \dots, 4.$$

These functions are

$$\widehat{N}_1 = \tfrac{1}{4}(1-\xi)(1-\eta), \qquad \widehat{N}_2 = \tfrac{1}{4}(1+\xi)(1-\eta),$$

$$\widehat{N}_3 = \tfrac{1}{4}(1+\xi)(1+\eta), \qquad \widehat{N}_4 = \tfrac{1}{4}(1-\xi)(1+\eta).$$

The nice joint formula $\frac{1}{4}(1\pm\xi)(1\pm\eta)$ for the whole set justifies the choice of this reference square over $[0,1]\times[0,1]$.

1.2 The local spaces

Take now a parallelogram K and write its four vertices in rotating order (clockwise or counter–clockwise, it doesn't matter)

$$\mathbf{p}_\alpha^K = (x_\alpha, y_\alpha), \qquad \alpha = 1, \dots, 4.$$

Consider now a linear map that transforms $\widehat{K}$ into K. For instance, this one does the job:

$$\begin{bmatrix} x \\ y \end{bmatrix} = -\frac{1}{2}(\xi+\eta)\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \frac{1}{2}(1+\xi)\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \frac{1}{2}(1+\eta)\begin{bmatrix} x_4 \\ y_4 \end{bmatrix}.$$

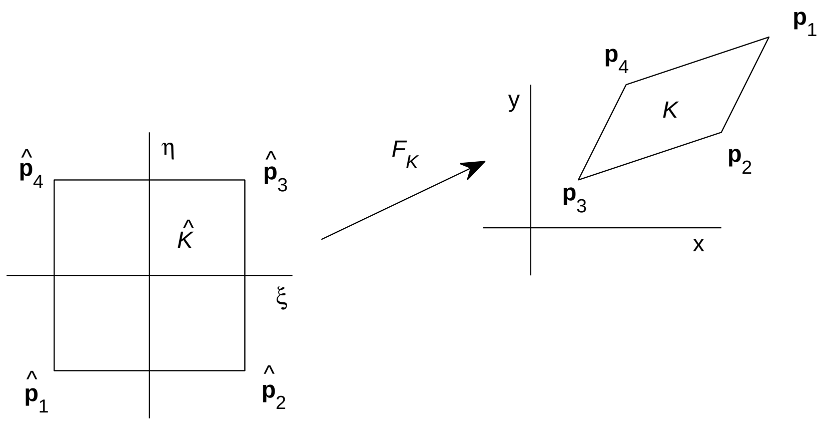


Figure 3.3: The reference square $\widehat{K}$ is mapped to the physical domain K. Note that vertices are notated in rotating order, even if the sense is different.

If we had taken before the reference triangle with vertices on $\widehat{\mathbf{p}}_1$, $\widehat{\mathbf{p}}_2$ and $\widehat{\mathbf{p}}_4$, the $\mathbb{P}_1$ basis functions we would had found would have been

$$-\tfrac{1}{2}(\xi+\eta), \qquad \tfrac{1}{2}(1+\xi), \qquad \tfrac{1}{2}(1+\eta).$$

What we are doing is mapping this triangle into the triangle with vertices $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_4$. The additional point $\widehat{\mathbf{p}}_3$ is mapped automatically to $\mathbf{p}_3$, because K is a parallelogram and linear transformations preserve parallelism. Let's not worry about the explicit formula for the transformation. We'll call it $F_K : \widehat{K} \to K$ and write simply

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{B}_K \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \mathbf{b}_K.$$

or $(x, y) = F_K(\xi, \eta)$. We finally get to the local polynomial space

$$\begin{aligned} \mathbb{Q}_1(K) &= \{q : K \to \mathbb{R} \,|\, q \circ F_K \in \mathbb{Q}_1\} \\ &= \{\widehat{q} \circ F_K^{-1} \,|\, \widehat{q} \in \mathbb{Q}_1\}. \end{aligned}$$

Note that the space is defined by transforming the space $\mathbb{Q}_1$ on the reference element to the physical element K. In a way, that happened also with the $\mathbb{P}_k$, only with the simplicity that in that case

$$\mathbb{P}_k(K) = \mathbb{P}_k$$

and the space in physical and reference variables was the same.

Before giving properties of $\mathbb{Q}_1(K)$ (we need things like local degrees of freedom, the possibility of gluing together different elements, et cetera), let's have a look at the functions in this space. A function in $\mathbb{Q}_1$ is of the form

$$\widehat{q} = a_0 + a_1 \xi + a_2 \eta + a_3 \xi \eta.$$

The reference variables can be written in terms of the physical variables by inverting the transformation F_K. We obtain something of the form:

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} a\,x + b\,y + e \\ c\,x + d\,y + f \end{bmatrix}$$

(the actual numbers $a, \ldots, f$ are not important). Therefore

$$\begin{aligned} \widehat{q} \circ F_K^{-1} &= a_0 + a_1(a\,x + b\,y + e) + a_2(c\,x + d\,y + f) + a_3(a\,x + b\,y + e)(c\,x + d\,y + f) \\ &= b_0 + b_1\,x + b_2\,y + a_3(a\,c\,x^2 + b\,d\,y^2 + (a\,d + b\,c)\,x\,y), \end{aligned}$$

which means that functions in $\mathbb{Q}_1(K)$ have a linear part plus a term of degree two that depends on the element K (see how the coefficients of F_K^{-1} are there). Actually, it looks like the space depends on the transformation chosen to map K from $\widehat{K}$, but that's not so. The following list of facts is of easy verification:

- The space $\mathbb{Q}_1(K)$ depends only on K, not on the concrete transformation F_K we have chosen. This is an important property, that means that we have not to worry about the way in which we ordered the vertices of the parallelogram K. We only need a list in rotating order and nothing else.

- If K is a rectangle with sides parallel to the cartesian axes (and in fact, only in this case), the space $\mathbb{Q}_1(K)$ is simply $\mathbb{Q}_1$.

- In all cases
$$\mathbb{P}_1 \subset \mathbb{Q}_1(K) \subset \mathbb{P}_2,$$
so $\mathbb{Q}_1(K)$ contains all polynomials of degree at most one and is a space of polynomials of degree at most four. The first part of this property is what will give order of convergence to the finite element method using this space.

- The space $\mathbb{Q}_1(K)$ has dimension four. The functions
$$N_\alpha^K = \widehat{N}_\alpha \circ F_K^{-1}, \qquad \alpha = 1, \ldots, 4$$
form a basis of this space. In fact
$$N_\alpha^K(\mathbf{p}_\beta^K) = \delta_{\alpha\beta}, \qquad \alpha, \beta = 1, \ldots, 4.$$

- Restricted to any of the sides of K, a function of $\mathbb{Q}_1(K)$ is a linear function of one variable, that is, if $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are two consecutive vertices of K and $q \in \mathbb{Q}_1(K)$, then
$$t \ni [0,1] \longmapsto q((1-t)\mathbf{p}_i + t\,\mathbf{p}_{i+1}) \in \mathbb{P}_1(t) = \{a + b\,t \,|\, a, b \in \mathbb{R}\}.$$

From the last two bullet points of this list we easily recover the needed properties to construct finite element spaces. First

> *a function of $\mathbb{Q}_1(K)$ is uniquely determined by its values on the four vertices of K*

and

> *the form a function of $\mathbb{Q}_1(K)$ restricted to an edge is independent of the shape of K and is uniquely determined by its values on the two vertices of this side.*

You might have noticed that the second property looks longer than usual. It has to be like that. What we assert there is not only that the function on an edge (side) depends only on the values on the two vertices that lie on that edge, but also that the type of function itself does not depend on where are the other two vertices. Restricted to one of the sides we always have a linear function.

1.3 The $\mathbb{Q}_1$ finite element method

We are done locally. Now we have to divide the domain into parallelograms and glue the local spaces together. Note here the fact that not all domains can be decomposed in parallelograms, but that we are speaking of something else than rectangles and similar domains.

A partition of a domain in parallelograms (we will call **elements** to the parallelograms) has also to respect the rules we gave to partitions with triangles:

- two different elements can meet only on a common vertex or a full edge of both, and

- the partition has to respect the division of the boundary in Dirichlet and Neumann parts.

Recall that the last property is used only when there is a transition point from Γ_D to Γ_N somewhere inside a side of Ω. The properties are exactly the same as those demanded to triangulations. In fact, there is a tradition to calling simply elements to the constitutive figures (triangles or parallelograms) and triangulation to the partition, even if it is a 'parallelogramization' (that's an ugly word!), a tradition we are going to honor.

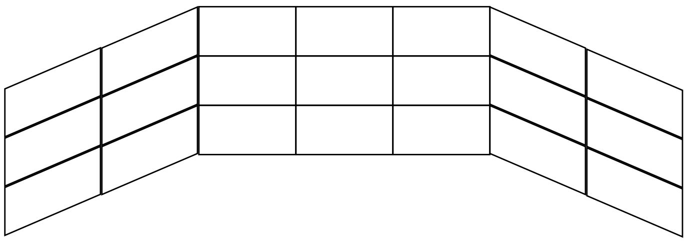


Figure 3.4: A 'triangulation' made of parallelograms

If we now fix values on the nodes (and now **nodes** are vertices of the elements again), we can construct a unique function on each K such that it belongs to the local space $\mathbb{Q}_1(K)$ and matches the values on the vertices. Because of the second local property, the functions on the edges do not depend really on the particular space $\mathbb{Q}_1(K)$ (i.e., on the shape of the element). They are always linear and fixed by the values on the corresponding two vertices. Therefore, what we obtain is a globally continuous function, an element of

$$V_h = \Big\{ u_h \in \mathcal{C}(\overline{\Omega}) \,\Big|\, u_h|_K \in \mathbb{Q}_1(K), \quad \forall K \in \mathcal{T}_h \Big\}.$$

We have done this enough times so that you already now what would come here if we just bothered to rewrite all the details:

- First, we note that the space

$$V_h^{\Gamma_D} = V_h \cap H^1_{\Gamma_D}(\Omega) = \{ v_h \in V_h \,|\, v_h = 0, \quad \text{on } \Gamma_D \}$$

  is the same as the space of elements of V_h that are zero on Dirichlet nodes.

- Then we number nodes and define the nodal basis functions, to obtain a basis of V_h with functions that have small support (four elements at most in a triangulation like the one of Figure 3.4, although there could be more with some special displays of parallelograms). Ignoring functions related to Dirichlet nodes we obtain a basis of $V_h^{\Gamma_D}$.

- We go on and copy the whole of Section 3 in Lesson 1. We have a finite element method, a linear system, mass and stiffness matrices,...

- In the assembly process we notice that, restricted to an element K, a nodal basis function φ_i is either the zero function or one of the four N_α^K. Computing local 4×4 matrices and assembling them in the usual fashion, we can construct effectively the matrix of the system. The same thing applies to the right–hand side. Whenever we want to evaluate N_α^K or its gradient, we have the usual formulas. Note that in this case, the gradients are not constant.

Are we there yet? Almost. We were forgetting about the order. The process is the same one. For the homogeneous Dirichlet problem we obtain

$$\|u - u_h\|_{1,\Omega} \le Ch|u|_{2,\Omega}.$$

The constant depends on the domain Ω (as well as on the division of Γ into Dirichlet and Neumann parts) and also on a parameter that measures the maximum flatness of elements.

Unlike in the case of triangles, flatness of elements is not given by extreme acuteness of angles, but can happen with elongated rectangles. Now, the measurement of flatness has to be the ratio between the maximum distance between points of an element and the radius of the largest circumference we can inscribe in K. This measurement of flatness is also valid for triangles and is the one that is given usually in textbooks. As a general rule, in this type of error analysis, elements cannot become too flat.

You will notice that, at least in theory, performance of $\mathbb{P}_1$ and $\mathbb{Q}_1$ elements seems to be very similar. Linear elements on triangles can be adapted to more complicated geometries and make assembly a bit simpler. In some cases (in particular in many important test cases in mechanics) using rectangles as elements reflects better the inherent geometrical features of the problem and is to be advised. In a forthcoming exercise we will observe that $\mathbb{Q}_1$ elements are just a bit stiffer (more rigid) than $\mathbb{P}_1$ elements.

1.4 Combination of $\mathbb{P}_1$ and $\mathbb{Q}_1$ elements

You might be thinking... if I cannot divide (triangulate) my polygon Ω with parallelograms, am I completely done with the whole $\mathbb{Q}_1$ stuff? Is that it? First of all, let me mention that you will have to wait to the next lesson to see how to construct elements on general quadrilaterals, elements that are, by the way, much more complicated to use. Anyhow, there's another possibility, which I personally find one of the most beautiful proofs of the great versatility of finite element methods and of the great idea that assembly is. Wait for it.

Let me recall something we just saw. In the global finite element space for the $\mathbb{Q}_1$ method

$$V_h = \Big\{ u_h \in \mathcal{C}(\overline{\Omega}) \,\Big|\, u_h|_K \in \mathbb{Q}_1(K), \quad \forall K \in \mathcal{T}_h \Big\},$$

the local space depends on the particular element. You could think that this makes the method complicated. What is complicated is the explanation of the method. The assembly process does not see this difference of local space since it sends evaluations of the local basis functions to the reference domain.

We can think of domains that admit a triangulation made of triangles and rectangles, such as the one of Figure 3.5. The rectangular pieces of the domain are perfect for a

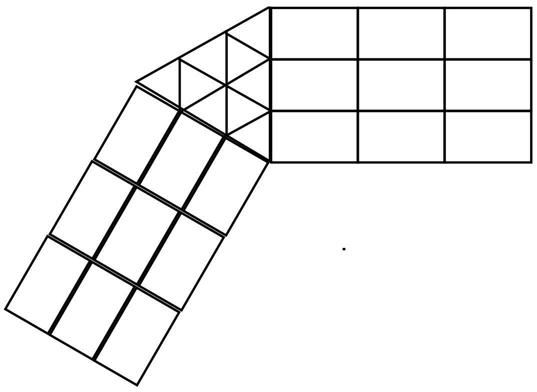


Figure 3.5: A triangulation made of triangles and rectangles.

division in rectangles, but the connecting piece seems to demand the use of triangles, so we decide to use both.

An element of this new type of triangulation can be either a triangle or a parallelogram. The triangulation has to fulfill the usual requirements: intersections can only happen in common vertices or common edges and Dirichlet–Neumann partition of the boundary has to be respected. The local space will depend on the type of element:

$$\mathbb{P}(K) = \left[\begin{array}{ll} \mathbb{P}_1, & \text{if } K \text{ is a triangle,} \\ \mathbb{Q}_1(K), & \text{if } K \text{ is a parallelogram.} \end{array} \right.$$

Nodes are vertices, as usual, and the global space

$$V_h = \Big\{ u_h \in \mathcal{C}(\overline{\Omega}) \,\Big|\, u_h|_K \in \mathbb{P}(K), \quad \forall K \in \mathcal{T}_h \Big\}$$

is easily defined by gluing elements because of the following facts:

> *a function of $\mathbb{P}(K)$ is uniquely determined by its values on the nodes (three or four) of K*

and

> *the form a function of $\mathbb{P}(K)$ restricted to an edge is independent of the type of K and is uniquely determined by its values on the two vertices of this side.*

Seen on edges, all these discrete functions are linear, so we can glue triangles with parallelograms of any shape, as we were able to glue different parallelograms together.

Other than this, life goes on as usual. In the process of assembly is where we use whether an element is a parallelogram or a rectangle: the reference domain is different depending on which and local matrices have different sizes (3×3 for triangles, 4×4 for parallelograms). This looks more complicated but you have to think in terms of the grid generator. If it gives you triangles and rectangles, either they are given in a different list (and you assemble first ones and the the other) or it gives you information about the

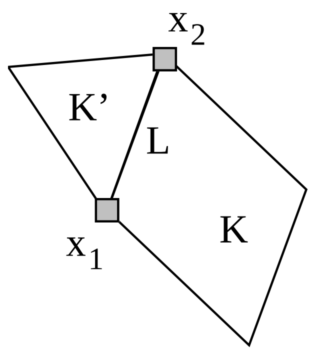


Figure 3.6: A triangle and a parallelogram sharing an edge.

type of geometry of each element, which you obviously learn by only looking at the list of vertices of the element.

What about error? Let's stick again to the case of homogeneous Dirichlet condition. Céa's lemma still applies

$$\|u - u_h\| \le \frac{M}{\alpha} \inf \left\{ \|u - v_h\| \,\Big|\, v_h \in V_h^0 \right\}$$

and the right–hand side is an approximation error which can be bounded locally, element by element. Hence, the error of the method can be bounded by the error of approximating with linear functions on triangles and with $\mathbb{Q}_1(K)$ functions on parallelograms. In both cases, we have an h–type error. Order one.

2 Higher order methods on parallelograms

Once here, we should make a fast review of the novelties of introducing the $\mathbb{Q}_1$ method. Note that it took its time, compared to how simple it was to introduce the $\mathbb{P}_2$ elements once we had done everything on the $\mathbb{P}_1$ very clear. The novelties were: (a) there is a new reference element and therefore a new concept of triangulation, plus (b) the local space depends on the particular element, but (c) restricted to edges the local spaces do not depend on the element. That was basically all of it. Let us go for higher order.

2.1 The $\mathbb{Q}_2$ elements

The space $\mathbb{Q}_2$ uses all linear combinations of these monomials

$$1, \quad \xi, \quad \eta, \quad \xi^2, \quad \xi\eta, \quad \eta^2, \quad \xi^2\eta, \quad \xi\eta^2, \quad \xi^2\eta^2.$$

There is nine of them. (We will need nine nodes in the reference domain). Looking carefully you'll see that $\mathbb{Q}_2$ is the space of polynomials in the variables ξ and η that have degree at most two in each variable separately. It includes therefore all polynomials of degree two but goes up to polynomials of degree four.

There's a nice table that will simplify your life in remembering these spaces. It serves to compare $\mathbb{P}_2$ (the order two space for triangles) with $\mathbb{Q}_2$ (the order two space for squares)

$$\begin{array}{lll} \eta^2 & & \\ \eta & \xi\eta & \\ 1 & \xi & \xi^2 \end{array} \qquad\qquad \begin{array}{lll} \eta^2 & \xi\eta^2 & \xi^2\eta^2 \\ \eta & \xi\eta & \xi^2\eta \\ 1 & \xi & \xi^2 \end{array}$$

(You will have to recognize that that's clever). We now consider nine points (nodes) on the reference square:

- the four vertices,
- the midpoints of the four sides,
- the center (the origin).

The two relevant properties here are the usual ones: (a) a function of $\mathbb{Q}_2$ is uniquely determined by its values on the nine nodes; (b) restricted to one of the sides of $\widehat{K}$, a function of $\mathbb{Q}_2$ is a polynomial of degree two in one variable and is therefore determined by its values on the three nodes that lie on the edge.

Note that if you use a linear map F_K to transform $\widehat{K}$ into the parallelogram K, midpoints of edges are mapped onto midpoints of edges and the center is mapped onto the 'center' of the parallelogram (the point where both diagonals meet, or where the lines joining midpoints of opposite sides meet). See Figure 3.7 for a sketch of this.

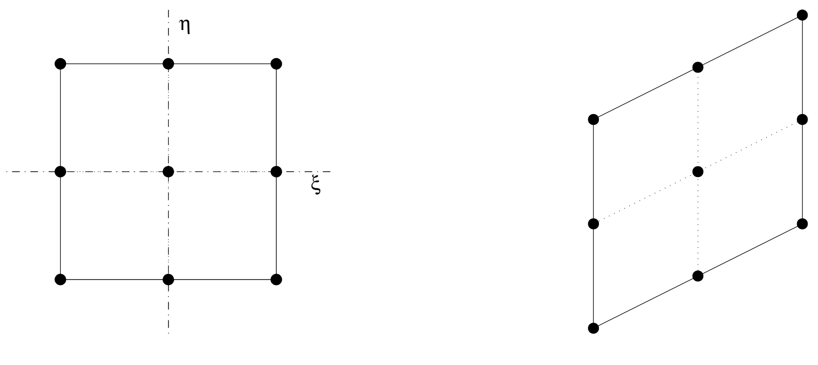


Figure 3.7: The $\mathbb{Q}_2$ reference element on the left. A parallelogram with the $\mathbb{Q}_2(K)$ nodes marked on the right.

Then we create the local space

$$\mathbb{Q}_2(K) = \{q : K \to \mathbb{R} \,|\, q \circ F_K \in \mathbb{Q}_2\} = \{\widehat{q} \circ F_K^{-1} \,|\, \widehat{q} \in \mathbb{Q}_2\},$$

and observe again that:

- this 9−dimensional space depends on K but not on the particular transformation,

- $\mathbb{Q}_2(K) = \mathbb{Q}_2$ when K is a rectangle in the horizontal–vertical direction, but in general
$$\mathbb{P}_2 \subset \mathbb{Q}_2(K) \subset \mathbb{P}_4,$$
- we can construct nine nodal basis functions on $\widehat{K}$, $\{\widehat{N}_\alpha \,|\, \alpha = 1, \ldots, 9\}$ and transform them to
$$N_\alpha^K = \widehat{N}_\alpha \circ F_K^{-1}$$
to obtain a nodal basis of $\mathbb{Q}_1(K)$,
- the two nodal properties that hold on the reference square still hold on K; in particular, restriction of an element of $\mathbb{Q}_1(K)$ to one of the four sides is a polynomial of degree at most two, and is independent on the shape of K.

From here on, everything is just a copy of the $\mathbb{Q}_1$ case: global spaces, global nodal basis functions, restriction of those to elements giving the local nodal functions, Dirichlet nodes, etc. Note that the interior node is just that: interior. Therefore you can apply static condensation to this node. In $\mathbb{P}_k$ we had to wait to $k = 3$ to obtain an interior node.

The fact that the polynomial degree increases is something you cannot take too lightly. For instance, when computing the local mass matrices
$$\int_K N_\beta^K \, N_\alpha^K,$$
you have to compute an integral of a polynomial of degree eight.

Order of the method (in the best possible conditions) is two in the $H^1(\Omega)$ Sobolev norm. The method can be simultaneously used with $\mathbb{P}_2$ elements over triangles for triangulations that combine parallelograms and triangles. Do you see? That was really fast.

3 Three dimensional domains

3.1 Elements on tetrahedra

What we did at the beginning of Lesson 1 about formulating a boundary value problem in a weak form can be easily done for three dimensional domains Ω. Integrals over Ω become volume integrals. Integrals over Γ are now surface integrals. If Ω is a polyhedral domain, it is possible (although not easy) to divide it into tetrahedra. Triangulations with tetrahedra[2] have to follow the usual rules: (a) two different elements can intersect only on a common vertex, a common edge or a common face; (b) the Dirichlet/Neumann division of the boundary has to be respected by the discretized geometry. This last point is much more tricky than before. If each face of the boundary of Ω is entirely included either on the Dirichlet boundary or on the Neumann boundary, everything is simple and condition

[2]We are in three dimensions, but we keep on calling these things triangulations. For these ones some people prefer the very specific neololgism tetrahedrizations.

(b) reduces to nothing. When there are transitions inside a face, these transitions have to be along straight lines or polygonal lines. Otherwise, the method introduces another kind of error, as the discrete geometry is not able to describe precisely the exact geometry. This is similar to what happens in curved boundaries, a problem that we will explore briefly in the following lesson.

An element of

$$\mathbb{P}_1 = \Big\{ a_0 + a_1\, x + a_2\, y + a_3\, z \,\Big|\, a_0, a_1, a_2, a_3 \in \mathbb{R} \Big\}$$

is uniquely determined by its values on the four vertices of a general non–degenerate tetrahedron. See Figure 3.8 for a sketch of the tetrahedral finite element. More over, seen on each of the faces of the tetrahedron such a function is a linear function of two variables and seen on each of the six edges it is a linear function of a single variable. Therefore: the value of the function on each face is determined by the three degrees of freedom (nodal values) that lie on that face and the value on each edge is determined by its values on the two associated nodes.

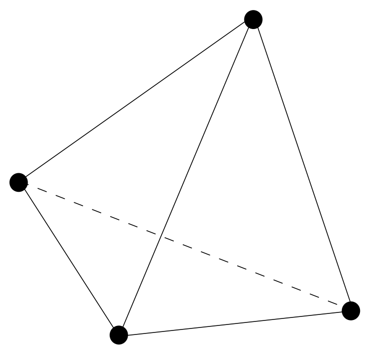


Figure 3.8: A tetrahedron and its four $\mathbb{P}_1$ degrees of freedom

With this in hand we can do our business as usual. Nothing is really changed by going to the three dimensional case. The reference element is usually taken as the tetrahedron with vertices on $(0,0,0)$, $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. Fortunately, the order of the local numbering of vertices is irrelevant, since all permutations give valid numberings.

The price to pay for this simplicity is the fact that tetrahedra are much more strange animals than they look at first sight. In particular it is not that simple to fathom how to divide a given tetrahedron into pieces that are not too deformed. Look at what happens (Figure 3.10) when you cut the four corners of a regular tetrahedron. Inside you obtain a regular octahedron that can be easily divided into two pyramids with square basis, each of which can be divided into two similar tetrahedra. The resulting interior four tetrahedra are not regular anymore. There are more ways of doing this kind of things. My point here is that tetrahedra are easy but not so easy.

Local dimension of the space is four. When you glue the corresponding $\mathbb{P}_1$ elements to create a finite element space the full dimension is the number of vertices. Dirichlet nodes are defined as usual (nodes on the Dirichlet boundary, or vertices of faces that are on the Dirichlet boundary). Order is one.

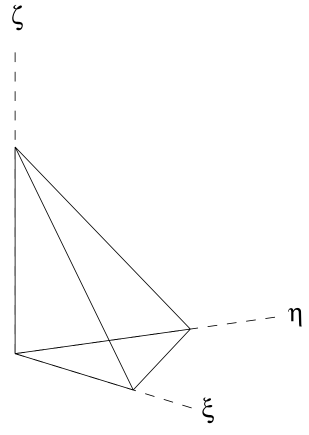


Figure 3.9: The reference tetrahedron, as seen from behind (sort of). The reference variables are ξ, η and ζ (some authors prefer z for the third one)

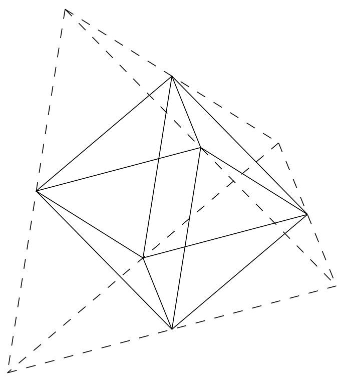


Figure 3.10: When you cut the four corners of a regular tetrahedron you end up with a regular octahedron

It is not difficult to define $\mathbb{P}_k$ elements on the tetrahedron for any k. Note that the local dimensions of the $\mathbb{P}_k$ spaces (as well as the number of necessary nodes) increase now much faster than in the two dimensional cases, because there are many more new monomials. The space $\mathbb{P}_k$ uses all monomials of the form

$$x^{i_1} y^{i_2} z^{i_3}, \qquad i_1, i_2, i_3 \geq 0, \quad i_1 + i_2 + i_3 \leq k.$$

For instance

$$\dim \mathbb{P}_2 = 9, \qquad \dim \mathbb{P}_3 = 19.$$

There's a formula for this but we will not give it here.

It is simple to give the nodes in the reference domain. For the $\mathbb{P}_k$ element, they are just the points with coordinates

$$\left(\tfrac{i_1}{k}, \tfrac{i_2}{k}, \tfrac{i_3}{k}\right), \qquad i_1, i_2, i_3 \geq 0, \quad i_1 + i_2 + i_3 \leq k.$$

We have to wait to $k = 4$ to obtain an interior node that we can condense statically.

3.2 Elements on parallelepipeds

The $\mathbb{Q}_k(K)$ elements are very naturally defined on parallelepipeds. The reference element is the cube $[-1, 1] \times [-1, 1] \times [-1, 1]$ or also $[0, 1] \times [0, 1] \times [0, 1]$, depending on personal preferences. The reference space $\mathbb{Q}_k$ is the one of all linear combinations of monomials

$$\xi^{i_1}\eta^{i_2}\zeta^{i_3}, \qquad 0 \le i_1, i_2, i_3 \le k$$

and has therefore dimension $(k + 1)^3$. Nodes are easily found by subdividing uniformly the cube into equally sized smaller cubes. Interior nodes appear already with $k = 2$. The local spaces on parallelepipeds (the image of a cube under a linear transformation) are the new spaces $\mathbb{Q}_k(K)$ defined as usual.

One has to be extra careful here in giving always vertices in a coherent order, so that we don't try to map the figure incorrectly from the reference element. That is the price to pay for the geometrical simplicity. The increase of the polynomial degree is also a non–minor issue: for $\mathbb{Q}_1(K)$ elements we have polynomials of degree three!

4 Exercises

E3.1. Comparison of $\mathbb{P}_1$ and $\mathbb{Q}_1$

Consider a square domain Ω and two triangulations of it as the ones given in Figure 3.11. In the first triangulation we consider a $\mathbb{P}_1$ method for the usual equation, only with Neumann conditions. In the second partition we consider a $\mathbb{Q}_1$ method.

Check that we have the same number of unknowns in both cases. Draw the form of the mass–plus–stiffness matrices in both cases. Check that the $\mathbb{Q}_1$ has in principle more non–zero elements, since there are pairs of adjacent nodes that are not in the triangular mesh.

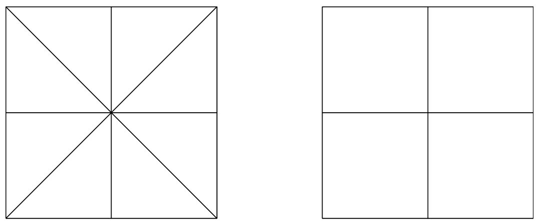


Figure 3.11: A triangle mesh and a parallelogram mesh of a square

E3.2. The $\mathbb{Q}_3$ element in the plane

What would be the nodes and the polynomial space for a generalization of the $\mathbb{Q}_k$ type elements to $k = 3$? How many interior nodes do you obtain?

E3.4. Elements on prisms

The reference prism with triangular basis can be taken for instance as the set of points (ξ, η, ζ) with

$$0 \leq \xi, \eta, \qquad \xi + \eta \leq 1, \qquad 0 \leq \zeta \leq 1.$$

In the two plane variables it works like a triangle. In the vertical variables it works like a parallelogram. Propose a correct polynomial space in this reference configuration so that the six vertices are valid nodes for a finite element using prisms.

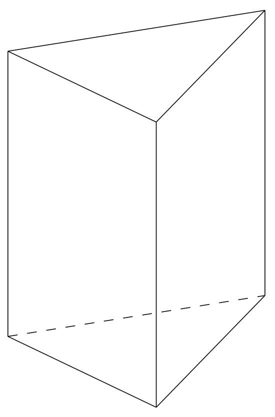


Figure 3.12: The reference prism

Lesson 4

More advanced questions

In this lesson we are going to have a fast look at several different questions related to how the Finite Element Method is used (or adapted) in different situations. The section on eigenvalues is of particular importance, since we will be using it for the stability analysis of evolution problems.

1 Isoparametric elements

So far we have only dealt with polygonal domains. You will agree that in many instances boundaries are due to be curved, so we will have to take into account that fact.

First of all when creating a triangulation, you are substituting your real curved domain by a polygonal approximation. Your grid generator is going to take care of the following detail: *all boundary nodes of the triangulation have to be placed on the real boundary.* This means in particular that if you need smaller triangles, you cannot obtain them by simply subdividing your existing grid and you definitely have to call back your grid generator to give you new vertices that are on the boundary.

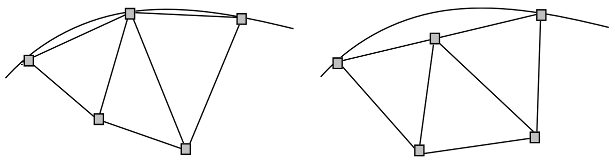


Figure 4.1: A correct and an incorrect triangulation of a curved domain.

You might think, well that's it then, isn't it? You have your triangles and you apply your triangular finite element scheme. The answer is yes if you are going to apply the $\mathbb{P}_1$ method.

Note for a moment that functions on $H^1(\Omega)$ are defined on Ω and functions of V_h on the approximated polygon. Therefore the discrete space V_h is not a subspace of $H^1(\Omega)$. However, an error analysis is still possible. What this error shows is that the error produced by the geometry approximation beats the error of the method if we try to do $\mathbb{P}_2$ elements or higher, so it doesn't pay off to use high order elements if the approximation to the boundary is so rough.

Let us see how to mend this for the $\mathbb{P}_2$ approximation. Note that this is not a purely theoretical question and that part of what we are going to learn here will be used to define finite elements on general quadrilaterals.

1.1 Deformed triangles

As usual, take $\widehat{K}$ to be the reference triangle and K^0 a triangle with vertices

$$\mathbf{p}_\alpha^K = (x_\alpha, y_\alpha), \qquad \alpha = 1, 2, 3.$$

With these points we construct the linear map $F_K^0 : \widehat{K} \to K^0$

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \\ &= (1 - \xi - \eta) \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \xi \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \eta \begin{bmatrix} x_3 \\ y_3 \end{bmatrix}. \end{aligned}$$

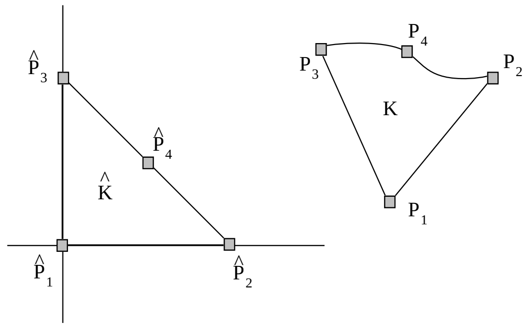


Figure 4.2: The reference triangle and a deformation of the image triangle

Let us now call $\mathbf{p}_4^K$ to the midpoint of the segment that joins $\widehat{\mathbf{p}_2}$ and $\widehat{\mathbf{p}}_3$, that is,

$$\widehat{\mathbf{p}}_4 = (\tfrac{1}{2}, \tfrac{1}{2}).$$

Take a fourth point in the physical space, $\mathbf{p}_4^K = (x_4, y_4)$, and compute its deviation from the midpoint of $\mathbf{p}_2^K$ and $\mathbf{p}_3^K$

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

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

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

Note that this is a linear transformation plus a correction term. The transformation F_K satisfies the following properties, all of them of easy verification:

- It sends the chosen points in the reference domain to the ones in the physical space

$$F_K(\widehat{\mathbf{p}}_\alpha) = \mathbf{p}^K_\alpha, \qquad \alpha = 1, \dots, 4.$$

- If $\xi = 0$, then

$$F_K(0,t) = F^0_K(0,t).$$

  This means that the image of the vertical edge in reference coordinates is the segment joining $\mathbf{p}^K_1$ and $\mathbf{p}^K_3$, covered at constant velocity, as if we were using the linear transformation. The same thing happens to the horizontal side of $\widehat{K}$.

- If $\mathbf{p}^K_4$ is aligned with $\mathbf{p}^K_2$ and $\mathbf{p}^K_3$, then the image of the edge that joins $\widehat{\mathbf{p}}_2$ and $\widehat{\mathbf{p}}_3$ is the segment that joins $\mathbf{p}^K_2$ and $\mathbf{p}^K_3$. However, this segment is parameterized at constant velocity only when $\mathbf{p}^K_4$ is the midpoint of $\mathbf{p}^K_2$ and $\mathbf{p}^K_3$ (in that case $\delta_x = \delta_y = 0$ and we have only the linear term in the transformation F_K).

- The Jacobian matrix of F_K is not constant:

$$\mathbf{B}_K = \mathrm{D}F(\xi,\eta) = \mathbf{B}^0_K + 4\begin{bmatrix} \eta \\ \xi \end{bmatrix}\begin{bmatrix} \delta_x & \delta_y \end{bmatrix} = \begin{bmatrix} x_2 - x_1 + 4\eta\delta_x & x_3 - x_1 + 4\eta\delta_y \\ y_2 - y_1 + 4\xi\delta_x & y_3 - y_1 + 4\xi\delta_y \end{bmatrix}$$

When $\mathbf{p}^K_4$ is not too far from the midpoint of $\mathbf{p}^K_2$ and $\mathbf{p}^K_3$, that is, when the deviation (δ_x, δ_y) is not too large, it is possible to prove that the image of $\widehat{K}$ under this transformation $K = F_K(\widehat{K})$ is mapped bijectively from the reference element and therefore we can construct an inverse to

$$F_K : \widehat{K} \to K.$$

1.2 Local spaces

Now we have the physical element, K, which is defined as the image of $\widehat{K}$ by the transformation F_K, so we have gone one step further from the beginning, as now the physical element is only defined from the reference element. With this element in hand we define the local space by transforming $\mathbb{P}_2$ on reference variables (as we did with all $\mathbb{Q}_k(K)$ spaces):

$$\mathbb{P}_2(K) = \{p : K \to \mathbb{R} \,|\, p \circ F_K \in \mathbb{P}_2\} = \{\widehat{p} \circ F_K^{-1} \,|\, \widehat{p} \in \mathbb{P}_2\}.$$

The degrees of freedom are placed in the following six nodes:

- the three vertices,
- the midpoints of the two straight sides,
- the point $\mathbf{p}^K_4$.

We do not have an explicit expression of how elements of $\mathbb{P}_2(K)$ are, but we know that if $\widehat{N}_\alpha$ are the six nodal basis functions of the $\mathbb{P}_2$ reference element, then the functions

$$N^K_\alpha = \widehat{N}_\alpha \circ F_K^{-1}$$

form a basis of $\mathbb{P}_2(K)$. The following properties are simple to prove:

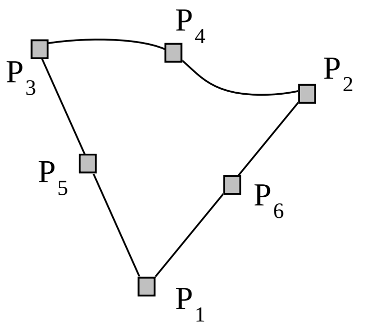


Figure 4.3: The local nodes in an isoparametric $\mathbb{P}_2$ triangle

- A function in $\mathbb{P}_2(K)$ is uniquely determined by the values on the six nodes on K.
- Restricted to any of the two straight sides of K, a function in $\mathbb{P}_2(K)$ is a polynomial of degree two in one variable (that is, the form of the function does not depend on the geometry of the element) and is therefore uniquely determined by its values on the three nodes that lie on that side.
- The value of a function in $\mathbb{P}_2(K)$ on the curved edge of K is uniquely determined by its value on the three nodes that lie on that edge.

The first property allows us to use the six nodes as local degrees of freedom. The second one allows as to glue $\mathbb{P}_2(K)$ on curved triangles with $\mathbb{P}_2$ elements on straight triangles, since the values on the straight sides are just polynomials.

If K is a usual straight triangle and we take $\mathbf{p}_4^K$ to be the midpoint of the corresponding edge, then F_K is a linear map and $\mathbb{P}_2(K) = \mathbb{P}_2$.

1.3 Finite element spaces with isoparametric triangles

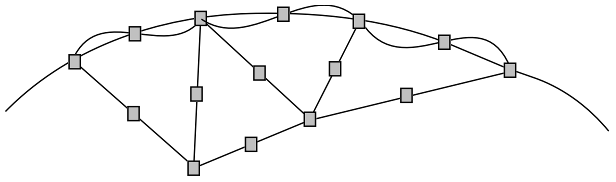


Figure 4.4: A triangulation using isoparametric elements

Let us then begin with an approximate triangulation of a curved domain following the rules:

- Intersection of two different triangles can only happen in a common vertex or edge.
- There must be a vertex placed on each transition point from Dirichlet to Neumann boundaries.

- Triangles with an edge on the of the approximating polygon can have only one edge on this boundary and both vertices have to be on the exact boundary Γ.

Look again at Figure 4.1 to see what we mean. Not only we want boundary triangles to hang from the real boundary, but we want to avoid a triangle to have two edges on the boundary[1].

The second part of the triangulation process consists of choosing a point on the exact boundary for each boundary edge. This point should be close to the midpoint of the straight edge that approximates the real curved boundary. We use this new point to construct an isoparametric triangle with the same vertices for each boundary triangle.

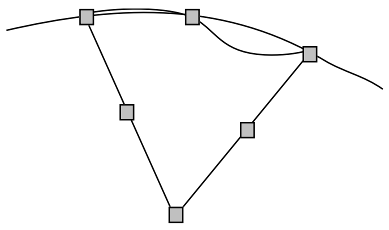


Figure 4.5: Substituting a straight triangle on the boundary by a isoparametric triangle.

When we write the equations of the finite element method using these local spaces, we must have in mind that the union of all triangles (curved on the boundary, straight otherwise) is not the original domain Ω, but an approximation of it, which we will call Ω_h. We will still call **Dirichlet nodes** to nodes on the Dirichlet boundary, remarking that these nodes are in the correct boundary Γ, so we will be able to read data on them when needed. The full finite element space is

$$V_h = \Big\{ u_h \in \mathcal{C}(\overline{\Omega}) \,\Big|\, u_h|_K \in \mathbb{P}_2(K), \quad \forall K \in \mathcal{T}_h \Big\},$$

and the subspace with homogeneous Dirichlet boundary conditions is

$$V_h^{\Gamma_D} = \{ v_h \in V_h \,|\, v_h(\mathbf{p}) = 0, \, \forall \mathbf{p} \,\text{Dirichlet node}\}.$$

Note that functions of $V_h^{\Gamma_D}$ are not really zero on the Dirichlet boundary Γ_D but on the curved approximation of that boundary, an approximation hanging from the vertices of the initial triangulation and from the additional point per edge that was used to create the isoparametric elements. It will not come out as a surprise, since the process of gluing spaces is the same as what we did with $\mathbb{P}_2$ elements, that the dimension of V_h is the number of nodes (that is the number of vertices plus the number of edges) and the dimension of $V_h^{\Gamma_D}$ is the number of non–Dirichlet edges. A nodal basis can be constructed as usual. The restriction to elements of nodal basis functions will be again the local basis functions, themselves defined as the transformed local nodal functions on the reference element.

[1]Many grid generators, even for polygonal domains, avoid putting two edges of the same triangle on the boundary. There is a simple reason for that: if two edges of a triangle are in a homogeneous Dirichlet boundary and we are using $\mathbb{P}_1$ elements, the function vanishes in the whole triangle, which is a poor result.

The discrete bilinear form is $a_h : V_h \times V_h \to \mathbb{R}$

$$a_h(u_h, v_h) = \int_{\Omega_h} \nabla u_h \cdot \nabla v_h + \int_{\Omega_h} u_h \, v_h,$$

and the linear form is $\ell_h : V_h \to \mathbb{R}$

$$\ell_h(v_h) = \int_{\Omega_h} f \, v_h + \int_{\Gamma_N^h} g \, v_h.$$

With them we obtain the numerical method

$$\left[\begin{array}{ll} \text{find } u_h \in V_h \text{such that} & \\ u_h(\mathbf{p}_i) = g_0(\mathbf{p}_i), & \forall i \in \mathrm{Dir}, \\ a_h(u_h, \varphi_i) = \ell_h(\varphi_i), & \forall i \in \mathrm{Ind}\,. \end{array}\right.$$

Note that the bilinear form poses no problem whatsoever. The fact that we are working on the approximate domain is sort of invisible to the assembly process: we will go element by element transforming from the reference configuration and after having added all terms we will have computed an integral over Ω_h instead of Ω. More on this at the end of this section.

The issue of the data functions is a little more delicate. When we want to compute

$$\int_K f \, \varphi_i \qquad \text{or, in fact,} \qquad \int_K f \, N_\alpha^K$$

for one of the curved domains K it is perfectly possible that the source function is not defined in parts of K. Look at Figure 4.5 and see how there is a small piece of the discrete geometry that lies outside the domain. Several theoretically sound possibilities can be proposed to mend this. In practice, and since you are due to use quadrature formulas for this integrals, just avoid using quadrature points in those areas.

The situation for the Neumann conditions (given normal derivative) is even more complicated and I have been deliberately vague in writing

$$\int_{\Gamma_N^h} g \, \varphi_i$$

without specifying what I mean by Γ_N^h. The fact is g_1 is defined in the exact Neumann boundary and φ_i in its approximation, so the integral is just a way of speaking. Assembly of this term will be done edge–by–edge. For each edge integral we could just try to use a quadrature formula that evaluates only on the three common points between the exact and the discrete geometry or think of something more clever. Let's not do this right now. I just wanted you to see that complications arise very easily.

Even when computing the local integrals

$$\int_K \nabla N_\beta^K \cdot \nabla N_\alpha^K \qquad \int_K N_\beta^K N_\alpha^K$$

for K isoparametric we still have to be careful. Let us begin with the easy one, the mass matrix. We have a formula for the local basis functions

$$N_\alpha^K = \widehat{N}_\alpha \circ F_K^{-1}.$$

If we want to evaluate N_α^K in a point of K, say (x, y), we need to compute $F_K^{-1}(x, y)$. This is the same as solving the non–linear system

$$\begin{array}{rcl} x &=& (x_2 - x_1)\,\xi + (x_3 - x_1)\,\eta + x_1 + 4\xi\eta\delta_x \\ y &=& (y_2 - y_1)\,\xi + (y_3 - y_1)\,\eta + y_1 + 4\xi\eta\delta_y. \end{array}$$

It is only a 2×2 system and equations are quadratic, but it is still a non–linear system and you will need Newton's method or something similar to get an approximate solution. Of course we know the exact solution for six points (the six nodes), since they are mapped back to the six nodes of the reference domain, so using these points is for free. It looks like we are done, but you have still to notice that the integral is happening over a very strange domain for which we don't have quadrature formulas. What is the wise thing to do? Move everything back to the reference domain:

$$\int_K N_\beta^K N_\alpha^K = \int_{\widehat{K}} |\det \mathbf{B}_K| \widehat{N}_\beta \, \widehat{N}_\alpha^K.$$

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

$$\mathbf{B}_K = \left[\begin{array}{cc} x_2 - x_1 + 4\eta\delta_x & x_3 - x_1 + 4\eta\delta_y \\ y_2 - y_1 + 4\xi\delta_x & y_3 - y_1 + 4\xi\delta_y \end{array} \right]$$

on the chosen quadrature points. The stiffness matrix is more challenging. Instead of trying to work the integral on the curved domains (with the complication of having to invert F_K every time we need an evaluation), what we do is go backwards to the reference domain and write

$$\int_{\widehat{K}} |\det \mathbf{B}_K| \, (\mathbf{C}_K \nabla \widehat{N}_\alpha \cdot \nabla \widehat{N}_\beta),$$

where

$$\mathbf{C}_K = \mathbf{B}_K^{-1} \mathbf{B}_K^{-\top}$$

(we did this in Lesson 2) is a non–constant matrix that requires inversion of $\mathbf{B}_K$ every time an evaluation is needed.

The whole thing looks more complicated than it is, because there are many aspects to take care of at the same time. The lesson you have to learn here is that evaluating anything (a basis function, its gradient, etc) has a price so you should try to balance a sufficiently precise approximation (exact computation is not possible any longer) of the integrals with taking as few quadrature points as possible.

2 Elements on quadrilaterals

Going back to the case of polygonal domains, we might be still more interested in using grids of quadrilaterals type than triangular grids. It can be a question of your geometry being described in a simpler form by using quadrilaterals, a preference for $\mathbb{Q}_k$ elements or a physical motivation to prioritize directions in the discrete level[2]. Whatever your reasons are, here is a way of defining finite elements of quadrilaterals that are not parallelograms. By the way, many people say **quads**, which is a nice shortening. I'm not going to write it again.

The construction is reminiscent of that of isoparametric elements. We begin with the reference square $[-1,1]\times[-1,1]$ and recall the four $\mathbb{Q}_1$ basis functions

$$\frac{1}{4}(1\pm\xi)(1\pm\eta)$$

(it is easy to check which one corresponds to each vertex). Now we take a general convex quadrilateral[3] and take its four vertices in rotating order: $\mathbf{p}_1^K,\ldots,\mathbf{p}_4^K$.

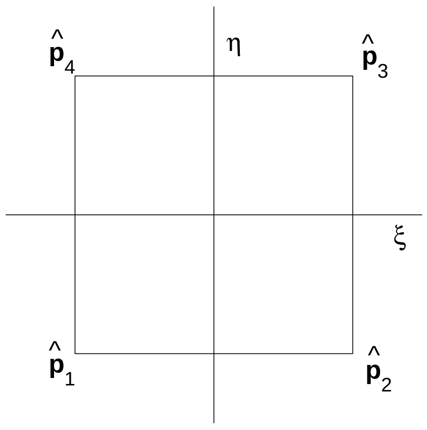


Figure 4.6: The reference square again.

Since the functions $\widehat{N}_\alpha$ satisfy

$$\widehat{N}_\alpha(\widehat{\mathbf{p}}_\beta)=\delta_{\alpha\beta},\qquad \alpha,\beta=1,\ldots,4,$$

it is clear that the map $F_K:\widehat{K}\to\mathbb{R}^2$

$$\begin{bmatrix} x \\ y \end{bmatrix}=\widehat{N}_1(\xi,\eta)\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}+\widehat{N}_2(\xi,\eta)\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}+\widehat{N}_3(\xi,\eta)\begin{bmatrix} x_3 \\ y_3 \end{bmatrix}+\widehat{N}_4(\xi,\eta)\begin{bmatrix} x_4 \\ y_4 \end{bmatrix},$$

sends vertices to vertices, that is

$$F_K(\widehat{\mathbf{p}}_\alpha)=\mathbf{p}_\alpha^K,\qquad \alpha=1,\ldots,4.$$

[2]In any case, remember that from a quadrilateral grid you can always obtain a triangular one doubling the number of elements and with a little less stiffness (remember the exercise in Lesson 2).

[3]Do not waste your time with non–convex quadrilaterals. For that it's better to use pairs of triangles

Moreover, the restriction of F_K to one of the four sides of $\widehat{K}$ is mapped at constant velocity to the corresponding edge of K. Obviously, by continuity, the interior of $\widehat{K}$ is mapped to the interior of K.

The map F_K in fact transforms a uniform Cartesian into something very similar in K, as shown in Figure 4.7. Computation of F_K^{-1} is therefore very simple on points of this special grid in the quadrilateral.

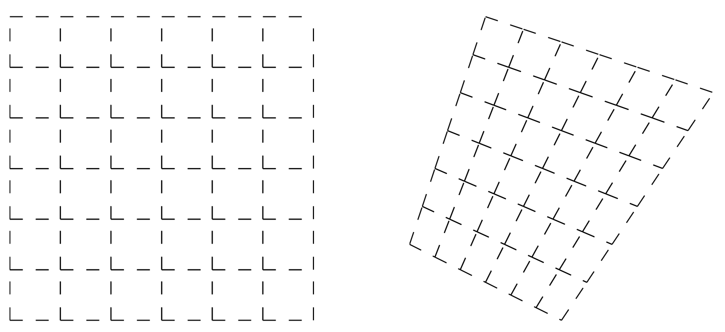


Figure 4.7: The image by the bilinear map of a Cartesian grid in the reference square.

With this transformation in hand we can define the local spaces

$$\mathbb{Q}_k(K) = \Big\{ q : K \to \mathbb{R} \,\Big|\, q \circ F_K \in \mathbb{Q}_k \Big\}.$$

I think you can already imagine what the local nodes are, how we can glue elements on different quadrilaterals and so on. If K is a parallelogram, F_K is a linear map and we obtain the usual $\mathbb{Q}_k$ spaces, which are composed of polynomial functions. In other cases, elements of the space are functions of the form $\widehat{q} \circ F_K^{-1}$ with $\widehat{q}$ a polynomial. The inverse map F_K^{-1} is not linear anymore. Now it is rational function. Therefore, elements of $\mathbb{Q}_k(K)$ are not polynomials any longer, which is not really relevant, since we are only going to use the basis functions, which we obtain by transforming from the reference element.

Just a fast list of facts:

- The spaces $\mathbb{Q}_k(K)$ depend on the quadrilateral and not on the order we have given the vertices to construct the transformation.
- The image of the $\mathbb{Q}_k$ nodes by F_K are valid degrees of freedom in $\mathbb{Q}_k(K)$.
- Restricted to the four sides, functions of $Q_k(K)$ are just polynomials of degree up to k in one variable. Therefore, the type of functions is independent of the shape of the quadrilateral and the values on sides is determined by the values on nodes that are on the side.

Thanks to these properties we can easily construct finite element spaces on quadrilateral grids (composed of convex quadrilaterals). Parallelograms are a particular case of these elements, so we can use all types of quadrilaterals together . Therefore, we can combine these elements with triangular elements of the same degree: for instance $\mathbb{Q}_1(K)$ elements on quadrilaterals with $\mathbb{P}_1$ elements on triangles.

3 Mass lumping

Let us just here add some comment about the mass matrix in the finite element method. For the usual $\mathbb{P}_k$ and $\mathbb{Q}_k$, we should expect the mass matrix

$$\int_\Omega \varphi_j\,\varphi_i$$

to be well–conditioned. Recall that the mass matrix is symmetric and positive definite. The spectral condition number of this type of matrices is the ratio between its largest and its smallest eigenvalue (all of them are real and positive) and good conditioning means that this ratio is not large. In particular it means that if

$$u_h = \sum_j u_j \varphi_j$$

then the constants in the inequality

$$C_1 \sum_j |u_j|^2 \le \int_\Omega |u_h|^2 \le C_2 \sum_j |u_j|^2$$

are of the same size and thus and the Euclidean norm of the vector of coefficients represents faithfully the $L^2(\Omega)-$norm of the function up to a scaling factor. In its turn, good conditioning means that the use of the most common iterative methods for systems with symmetric positive definite matrices (such as Conjugate Gradient) is going to converge quickly.

However, sometimes it seems convenient to substitute the mass matrix by an even simpler matrix. In the next lesson we will see a situation where this seems justified. Substitution of the mass matrix by a diagonal matrix is called mass lumping, since it lumps mass on the nodes instead of distributing it along pairs of nodes. We are going to explain this process for the $\mathbb{P}_1$ case.

Recall briefly the three–vertex quadrature rule on triangles (we mentioned it in the lesson on assembly)

$$\int_K \phi \approx \frac{\operatorname{area} K}{3} \sum_{\alpha=1}^{3} \phi(\mathbf{p}_\alpha^K),$$

where $\mathbf{p}_\alpha^K$ are the three vertices of K. This formula integrates exactly all polynomials of degree one. However, it introduces error when applied to a polynomial of degree two. In particular, the approximation

$$\int_K N_\beta^K N_\alpha^K \approx \frac{\text{area}}{K} \sum_{\gamma=1}^{3} N_\alpha^K(\mathbf{p}_\gamma^K) N_\beta^K(\mathbf{p}_\gamma^K) = \frac{\operatorname{area} K}{3} \delta_{\alpha\beta}$$

is not exact. (We have accumulated as many as three indices in the last expression. Do you see why the result holds? Note that local basis functions are one on a single vertex and zero on the other two). If we apply this approximation at the assembly process for the mass matrix, we are substituting the 3×3 local mass matrices by a 3×3 diagonal matrix. Adding up all contributions we are approximating

$$\int_\Omega \varphi_j \varphi_i \approx 0, \qquad i \neq j$$

and

$$\begin{aligned} \int_\Omega |\varphi_i|^2 &= \sum_K \int_K |\varphi_i|^2 \approx \sum \Big\{ \frac{\operatorname{area} K}{3} \,\Big|\, K \text{ such that } \mathbf{p}_i \in K \Big\} \\ &= \tfrac{1}{3} \operatorname{area} \big(\operatorname{supp} \varphi_i\big). \end{aligned}$$

Once again, the support of φ_i is the set of triangles that surround the node $\mathbf{p}_i$.

In an exercise at the end of this lesson we will see a very, very simple way of computing the lumped mass matrix once the exact mass matrix has been computed.

4 The discrete eigenvalues

While the mass matrix is well conditioned, the stiffness matrix is not. And it has to be so, because it is trying to approximate an intrisically ill–conditioned problem. We are going to have a look at this. Note that this section is really important to understand the stability analysis of the application of FEM methods for evolution problems, so take your time to understand what's being told here. Note that the results here are considerably deeper than what we have been using so far.

4.1 The Dirichlet eigenvalues of the Laplace operator

For simplicity, let us concentrate our efforts in problems only with Dirichlet conditions. In fact, with homogeneous Dirichlet conditions. Instead of trying to solve a boundary value problem, we are going to study an eigenvalue problem: find numbers λ such that there exists non–zero u satisfying

$$\left[\begin{array}{ll} -\Delta u = \lambda u, & \text{in } \Omega, \\ u = 0, & \text{on } \Gamma. \end{array} \right.$$

Note two things. First of all, $u = 0$ is not an interesting solution since it always satisfies the conditions, no matter what λ is. Second, boundary conditions in eigenvalue problems have to be zero. If you have two different solutions u for the same λ, any linear combination of them is another solution. The set of **eigenfunctions** (that's u) for a given **eigenvalue** (that's λ) is a subspace of ... (wait for it).

In this problem $\Gamma_D = \Gamma$. The space $H^1_\Gamma(\Omega)$ is given a different name. This one

$$H^1_0(\Omega) = \Big\{ u \in H^1(\Omega) \,\Big|\, u = 0, \quad \text{on } \Gamma \Big\}.$$

The set of eigenfunctions for a given eigenvalue is a subspace of $H_0^1(\Omega)$. Therefore, also of $H^1(\Omega)$ and of $L^2(\Omega)$, which are bigger and bigger spaces. Substituting the definition of eigenfunction inside Green's formula

$$\int_\Omega \Delta u\, v + \int_\Omega \nabla u \cdot \nabla v = \int_\Gamma (\partial_n u)\, v$$

and proceeding as usual, we obtain

$$-\lambda \int_\Omega u\, v + \int_\Omega \nabla u \cdot \nabla v = \int_\Gamma (\partial_n u)\, v = 0, \qquad \text{if } v = 0 \text{ on } \Gamma.$$

So we arrived easily to the weak formulation of the eigenvalue problem:

$$\left[\begin{array}{l} \text{find } \lambda \text{ such that there exists } 0 \neq u \in H_0^1(\Omega) \text{ satisfying} \\ \displaystyle\int_\Omega \nabla u \cdot \nabla v = \lambda \int_\Omega u\, v, \qquad \forall v \in H_0^1(\Omega). \end{array}\right.$$

Do we know how many eigenvalues are going to appear? Yes, we do. Infinitely many. But among those infinitely many, not so many, since we will be able to count them. I am going to try and break up the theoretical result in many pieces so that you really grasp what's in here:

- All eigenvalues are real and positive.
- They can be numbered and they diverge to infinity. There is therefore no accumulation point of eigenvalues. In other words, if you choose a finite interval, there is only a finite number of eigenvalues in it.
- Two eigenfunctions corresponding to two different eigenvalues are $L^2(\Omega)-$orthogonal. In more detail, assume that

  $$\left[\begin{array}{ll} -\Delta u = \lambda u, & \text{in } \Omega, \\ u = 0, & \text{on } \Gamma, \end{array}\right. \qquad \text{and} \qquad \left[\begin{array}{ll} -\Delta v = \mu v, & \text{in } \Omega, \\ v = 0, & \text{on } \Gamma, \end{array}\right.$$

  with $\lambda \neq \mu$. Then

  $$\int_\Omega u\, v = 0.$$

- The multiplicity of each eigenvalue is finite, that is, if λ is an eigenvalue, there is only a finite number of linearly independent eigenfunctions associated to it.

Let's try to put all these properties together. For each eigenvalue we take a set of linearly independent eigenfunctions. Using the Gram–Schmidt orthogonalization method, we can make them mutually $L^2(\Omega)-$orthogonal and with unit square integral. Instead of numbering the different eigenvalues, we take k copies of each eigenvalue with multiplicity k. Then we can order all the eigenvalues in increasing order and we obtain a list

$$0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \leq \ldots, \qquad \lambda_n \to \infty$$

and associate to each eigenvalue an eigenfunction

$$\left[\begin{array}{l} -\Delta\phi_n = \lambda_n \phi_n, \qquad \text{in } \Omega, \\ \phi_n = 0, \qquad \text{on } \Gamma, \end{array}\right.$$

so that

$$\int_\Omega \phi_n\, \phi_m = \delta_{nm}$$

and we have taken all possible eigenvalues and (linearly independent) eigenfunctions. Note again that eigenfunctions for different eigenvalues are per se orthogonal and that we enforce orthogonality of eigenfunctions of the same eigenvalue by an orthogonalization process.

There is an additional property that needed this kind of numbering of eigenvalues and eigenfunctions to be properly introduced:

- The sequence of eigenfunctions we have just obtained is a complete orthogonal set in $L^2(\Omega)$, which means that if $u \in L^2(\Omega)$, then

  $$u = \sum_{j=1}^\infty u_j\, \phi_j, \qquad u_j = \int_\Omega u\, \phi_j,$$

  with convergence of the series in the norm of $L^2(\Omega)$, i.e.,

  $$\int_\Omega \Big| u - \sum_{j=1}^n u_j \phi_j \Big|^2 \overset{n\to\infty}{\longmapsto} 0.$$

4.2 The discrete Dirichlet eigenvalues

Assume now that Ω is a polygon and take $V_h \subset H^1(\Omega)$, any of our choices of finite element spaces. By eliminating the Dirichlet nodes we obtain a basis of the space

$$V_h^0 = V_h \cap H_0^1(\Omega) = \{u_h \,|\, u_h = 0, \quad \text{on } \Gamma\}.$$

We now substitute the problem

$$\left[\begin{array}{l} \text{find } \lambda \text{ such that there exists } 0 \neq u \in H_0^1(\Omega) \text{ satisfying} \\ \displaystyle\int_\Omega \nabla u \cdot \nabla v = \lambda \int_\Omega u\, v, \qquad \forall v \in H_0^1(\Omega), \end{array}\right.$$

by its finite element approximation

$$\left[\begin{array}{l} \text{find } \lambda_h \text{ such that there exists } 0 \neq u_h \in V_h^0 \text{ satisfying} \\ \displaystyle\int_\Omega \nabla u_h \cdot \nabla v_h = \lambda_h \int_\Omega u_h\, v_h \qquad \forall v_h \in V_h^0. \end{array}\right.$$

Consider the matrices $\mathbf{W}$ and $\mathbf{M}$

$$w_{ij} = \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i, \qquad m_{ij} = \int_\Omega \varphi_j \varphi_i, \qquad i, j \in \text{Ind}.$$

Note that these are just the parts of the stiffness and mass matrices related to non–Dirichlet nodes. Let $N = \#\mathrm{Ind}$ be the number of non–Dirichlet problems. The discrete eigenvalue problem is equivalent to this other problem

$$\left[\begin{array}{l} \text{find } \lambda_h \text{ such that there exists } \mathbf{0} \neq \mathbf{u} \in \mathbb{R}^N \text{ satisfying} \\ \mathbf{W}\mathbf{u} = \lambda_h \mathbf{M}\mathbf{u}. \end{array}\right.$$

This last problem is a generalized eigenvalue problem for matrices. Let me condense the main properties of this problem for you. Recall that $N = N(h)$ is the dimension of the problem.

- (Generalized) eigenvalues are real and positive.
- Eigenvectors corresponding to different eigenvalues are orthogonal with respect to $\mathbf{M}$: if
$$\mathbf{W}\mathbf{u} = \lambda_h \mathbf{M}\mathbf{u}, \qquad \mathbf{W}\mathbf{v} = \mu_h \mathbf{M}\mathbf{v}$$
with $\lambda_h \neq \mu_h$, then
$$\mathbf{u} \cdot (\mathbf{M}\mathbf{v}) = 0$$
- There are N linearly independent eigenvectors.

Note that unlike in the original problems, there is no question about having more than N eigenvalues, since we are dealing with an $N \times N$ matrix. Counting eigenvalues as many times as their multiplicity we have N of them that we can arrange in increasing order

$$0 < \lambda_{h,1} \leq \lambda_{h,2} \leq \ldots \leq \lambda_{h,N}.$$

Choosing linearly independent eigenvectors in case we have multiplicity higher than one, we can choose vectors $\boldsymbol{\phi}_n$ such that

$$\mathbf{W}\boldsymbol{\phi}_n = \lambda_{h,n}\mathbf{M}\boldsymbol{\phi}_n$$

and

$$\boldsymbol{\phi}_n \cdot (\mathbf{M}\boldsymbol{\phi}_m) = \delta_{nm}.$$

The vectors $\boldsymbol{\phi}_n$ give a basis of $\mathbb{R}^N$. The corresponding finite element functions

$$\phi_{h,n} = \sum_{j=1}^{N} \phi_{n,j}\varphi_j, \qquad \boldsymbol{\phi}_n = (\phi_{n1}, \ldots, \phi_{nN})^\top$$

form a basis for V_h^0. Note that the $\mathbf{M}-$orthogonality of the eigenvectors is just the matrix form of the orthogonality condition

$$\int_\Omega u_{h,n}\, u_{h,m} = \delta_{nm}.$$

4.3 Convergence

So far we have two different problems. The continuous problem (the Dirichlet eigenvalues of the Laplace operator) has infinitely many solutions. The discrete problem (approximation by finite elements of the weak formulation of the eigenvalue problem) has a finite number of solutions. We will not deal in full detail with convergence of the discrete solutions to the exact solutions. We will however mention here two important properties. The first one is, let's say so, peculiar:

> *with the increasing order of continuous and discrete eigenvalues that takes into account their multiplicity, discrete eigenvalues always overestimate continuous eigenvalues*
> $$\lambda_n \le \lambda_{h,n}, \qquad n = 1, \ldots, N.$$

The second one is what we would expect from a discretization method:

> *discrete eigenvalues converge to continuous eigenvalues; for fixed (but arbitrary n)*
> $$\lambda_{h,n} \overset{h\to 0}{\longmapsto} \lambda_n,$$
> *if the triangulations become finer.*

You can think of you matrix eigenvalue problem as having infinetily many solutions: the N eigenvalues and then $\lambda_{h,N+1} = \lambda_{h,N+2} = \ldots = +\infty$. These non–finite eigenvalues obviously overestimate the corresponding continuous ones. When you increase the dimension of the space (you refine the mesh) you bring some newer eigenvalues from infinity. They begin to approximate the corresponding higher eigenvalues of the exact problems, which are larger as the dimension of the discrete space increases. Note that

$$\frac{\lambda_N}{\lambda_1} \approx \frac{\lambda_N}{\lambda_{h,1}} \le \frac{\lambda_{h,N}}{\lambda_{h,1}}.$$

This means that the ratio between the largest and smallest generalized eigenvalue of $\mathbf{W}$ diverges. Because the mass matrix is in principle well–conditioned, we can prove with this that the stiffness matrix is ill–conditioned. How bad the conditioning is depends on how fast the Dirichlet eigenvalues diverge. Anyway, you have to expect bad behavior of the stiffness matrix in anything that depends on conditioning.

5 Exercises

E4.1. Bad quadrilaterals

Figure 4.7 will help you to solve both questions in this exercise.

1. Take four points that define a convex quadrilateral, given in rotating order: $\mathbf{p}_1^K$, $\mathbf{p}_2^K$, $\mathbf{p}_3^K$ and $\mathbf{p}_4^K$. (They could be, for instance, the vertices of the reference square). If we give the third and the fourth vertices in the wrong order to the transformation, what is the transformed figure we obtain?

2. Take now the four vertices of a non–convex quadrilateral given in rotating order and consider the usual bilinear transformation from the reference square. Using the fact that vertical and horizontal lines in the reference square are mapped to straight lines in the physical element, what kind of figure are we mapping?

E4.2. Computation of the $\mathbb{P}_1$ lumped mass matrix

We will go step by step. Using the three vertex formula compute exactly the following integral

$$\int_K N_\alpha^K.$$

Adding the previous results, prove that

$$\int_\Omega \varphi_i = \tfrac{1}{3}\,\mathrm{area}\,\big(\mathrm{supp}\,\varphi_i\big).$$

Prove now that the sum of all nodal basis functions is the unit function

$$\sum_j \varphi_j \equiv 1.$$

(To do this, compare nodal values of both functions and note that constant functions belong to V_h.) Finally use the following trick based on the preceding identity

$$\int_\Omega \varphi_i = \sum_j \int_\Omega \varphi_j \varphi_i$$

to prove that

$$\tfrac{1}{3}\,\mathrm{area}\,\big(\mathrm{supp}\,\varphi_i\big) = \sum_j m_{ij}$$

and therefore the $i-$th diagonal element of the lumped mass matrix can be computed by adding all the elements of the $i-$th row of the mass matrix.

E4.3. Generalized eigenvalues from FE approximation

Assume that we are given a discrete problem

$$\left[\begin{array}{l} \text{find } \lambda_h \text{ such that there exists } 0 \neq u_h \in V_h^0 \text{ satisfying} \\ \displaystyle\int_\Omega \nabla u_h \cdot \nabla v_h = \lambda_h \int_\Omega u_h\, v_h \qquad \forall v_h \in V_h^0, \end{array}\right.$$

where V_h^0 is any subspace of $H_0^1(\Omega)$ for which we have a basis $\{\varphi_i \,|\, i = 1, \ldots, N\}$. Following the ideas of Lesson 1 (when we converted the Galerkin equations to a system of linear equations), prove that this problem is equivalent to the generalized eigenvalue problem

$$\left[\begin{array}{l} \text{find } \lambda_h \text{ such that there exists } \mathbf{0} \neq \mathbf{u} \in \mathbb{R}^N \text{ satisfying} \\ \mathbf{W}\mathbf{u} = \lambda_h \mathbf{M}\mathbf{u}. \end{array}\right.$$

for the matrices

$$w_{ij} = \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i, \qquad m_{ij} = \int_\Omega \varphi_j\, \varphi_i.$$

Lesson 5

Evolution problems

There are many different approaches in the application of finite element techniques to evolution problems. In fact, there are also many different types of evolution problems. In this lesson we are going to concentrate on two evolution equations:

- the heat equation, a good example of parabolic behavior (transient diffusion)
- the wave equation, the simplest model of hyperbolic equation of the second order.

We can group the FEM–based approaches for these evolution equations:

- methods that discretize space and time simultaneously, and
- methods that discretize one of these variables and then the other.

We are going to do as follows. We'll first take the heat equation and do time discretization with finite differences and then space discretization with finite elements. Afterwards we will see how discretization only of the space variable with FEM leads to a system of ordinary differential equations, for which you can use a great variety of methods. If we use a finite element type method for the time variable we end up with something very similar to applying a FEM discretization at the same time to space–and–time. Finally, we'll go for the wave equation and show some basic ideas.

1 FEM with forward Euler for the heat equation

First of all, let us state the problem. We are given a polygon Ω in the $(x, y)-$variable space. We are going to consider only Dirichlet boundary conditions, since they are usually more complicated. The extension to mixed boundary conditions is not very difficult, provided that the Dirichlet and Neumann boundaries remain fixed in time. The origin of times will be $t_0 = 0$. We will state the problem for all positive time values, from 0 to ∞, although we will be mainly thinking of solving the problem in a finite time interval $(0, T)$. The problem is the following: find $u(\mathbf{x}, t) = u(x, y, t)$ such that

$$\left[\begin{array}{ll} u_t = \Delta_{\mathbf{x}} u + f, & \text{in } \Omega \times (0, \infty), \\ u(\,\cdot\,, 0) = u_0, & \text{in } \Omega, \\ u(\,\cdot\,, t) = g, & \text{on } \Gamma \text{ for all } t > 0. \end{array}\right.$$

Many new things again, so let's go step by step:

- u_t is the partial derivative with respect to t and $\Delta_{\mathbf{x}} u$ is the Laplacian in the $(x, y)-$variables.
- $f : \Omega \times (0, \infty) \to \mathbb{R}$ is a given function of time and space.
- $g : \Gamma \times (0, \infty) \to \mathbb{R}$ is a given function of time and of the space variable on the boundary. It gives the enforced Dirichlet boundary condition for all times.
- When both f and g are independent of time, there is still evolution, but we will see that it is just a transient state converging to a steady–state solution. We'll talk about this on the section about stability.
- $u_0 : \Omega \to \mathbb{R}$ is a function of the space variable and represents the initial condition.

In principle, we are going to make ourselves our life simple by assuming that u_0 is a continuous function (and we can evaluate it without difficulty) and that f and g_0 are continuous in the time variable.

1.1 Time semidiscretization with forward Euler

Let us first take a partition in time, which can be non–uniform (variable time–step)

$$0 = t_0 < t_1 < t_2 < \ldots < t_n < \ldots$$

If our time interval is finite (as it is for any practical problem), the partition finishes in a certain point t_M. This is not important right now. The time–step is

$$\delta_n = t_{n+1} - t_n.$$

For us, doing a time step will be moving from an already computed approximation if time t_n to time t_{n+1}. The time–steps δ_n are given as if they were known from the beginning. Unless you are taking it to be uniform (which is not really a very good idea in most practical situations), time–steps are computed with information about the behavior of the numerical solution solution and about the performance of the method as we proceed in discrete time. For the sake of exposition, we do as if we already knew all time–steps in advance.

We freeze the source term and boundary data at each time t_n by simple evaluation

$$f_n = f(\,\cdot\,, t_n) : \Omega \to \mathbb{R}, \qquad g_n = g(\,\cdot\,, t_n) : \Gamma \to \mathbb{R}.$$

When the data functions are not continuous we should be willing to average in a time interval around t_n instead. Time semidiscretization strives to obtain approximations

$$u(\,\cdot\,, t_n) \approx u_n : \Omega \to \mathbb{R}.$$

The first attempt we will do is the forward (or explicit) Euler method. It consists of looking at the equation in discrete time n and approximating a time derivative in this time by the forward quotient

$$\phi'(t_n) \approx \frac{\phi(t_{n+1}) - \phi(t_n)}{t_{n+1} - t_n} = \frac{\phi(t_{n+1}) - \phi(t_n)}{\delta_n}.$$

If we take the heat equation and use this forward Euler approximation, we obtain the recurrence

$$\frac{u_{n+1} - u_n}{\delta_n} = \Delta u_n + f_n$$

or in explicit form

$$u_{n+1} = u_n + \delta_n \Delta u_n + \delta_n f_n.$$

This recurrence is started at $n = 0$ with the initial data function u_0. Bullet points again:

- Note that all functions are functions of the space variable, so the Laplace operator in space variables is just the Laplace operator. Time is now discrete time and appears as the n subindex everywhere.
- In principle this formula gives you u_{n+1} from u_n and the source function. Where is g? We've lost it in the way! There seems to be no way of imposing the boundary condition without entering in open conflict with the recurrence.
- There's more. If you begin with u_0, you take two derivatives to compute u_1. Then another two to compute u_2 and so on and so on. You had better have many space derivatives available! How could we possibly think of approximating u_n by a finite element function, which only has the first derivatives?

The answer to the last two questions comes from the use of a weak formulation for the recurrence. The price will be losing this explicit recurrence character that made the forward Euler approximation really explicit.

Consider Green's Theorem applied to u_n. Yes, I know we want to compute u_{n+1} (we already know u_n). Follow me anyway. We have

$$\int_\Omega (\Delta u_n)\, v + \int_\Omega \nabla u_n \cdot \nabla v = \int_\Gamma (\partial_n u_n)\, v.$$

Multiply this by δ_n and for lack of knowledge of the Neumann boundary data function, impose v to be zero on the boundary. We have therefore

$$\delta_n \int_\Omega (\Delta u_n)\, v + \delta_n \int_\Omega \nabla u_n \cdot \nabla v = 0, \qquad \text{for all } v \text{ such that } v = 0 \text{ on } \Gamma.$$

Substitute now the Laplacian of u_n, that is

$$\delta_n \Delta u_n = u_{n+1} - u_n - \delta_n f_n$$

and move what you know (data and functions at time n) to the right hand side to obtain

$$\int_\Omega u_{n+1} v = \int_\Omega u_n\, v - \delta_n \int_\Omega \nabla u_n \cdot \nabla v + \delta_n \int_\Omega f_n\, v, \qquad v = 0 \text{ on } \Gamma.$$

Now there seems to be room for imposing the missing Dirichlet boundary condition, implicitly at time $n+1$, since the test is satisfying the homogeneous Dirichlet boundary condition. The sequence of problems would be consequently: begin with u_0 and then for each n,

$$\left[\begin{array}{l} \text{find } u_{n+1} \in H^1(\Omega) \text{ such that} \\ u_{n+1} = g_{n+1}, \qquad \text{on } \Gamma, \\ \displaystyle\int_\Omega u_{n+1} v = \int_\Omega u_n\, v - \delta_n \int_\Omega \nabla u_n \cdot \nabla v + \delta_n \int_\Omega f_n\, v, \qquad \forall v \in H^1_0(\Omega). \end{array}\right.$$

Recall (last section of the previous Lesson) that

$$H^1_0(\Omega) = \{ v \in H^1(\Omega) \,|\, v = 0, \quad \text{on } \Gamma\}.$$

The problem looks more like what we have been solving so far[1]. Only there is no stiffness–term for the unknown, which is a problem (there is no way we will obtain ellipticity of the bilinear form in energy norm), and at the same time there is a stiffness term in the right–hand side, which is more complicated than usual. Don't worry, we are getting near something reasonable.

1.2 Full discretization

We are there. Take a finite element method. Any of the methods exposed in Lessons 1, 2 and 3 will do the job. We have the space

$$V_h \subset H^1(\Omega)$$

associated to a triangulation of the domain, a nodal basis, the concept of Dirichlet nodes (all nodes on the boundary) and the subspace

$$V_h^0 = V_h \cap H^1_0(\Omega) = \{ v_h \in V_h \,|\, v_h = 0, \quad \text{on } \Gamma\}.$$

Nodes are numbered as usual and we take two lists: Dir, the one of indices of Dirichlet nodes, and Ind, the remaining nodes. The Dirichlet nodes are then

$$\mathbf{p}_i, \qquad i \in \text{Dir}.$$

The main point now is to substitute all the infinite–dimensional elements of the problem

$$\left[\begin{array}{l} \text{find } u_{n+1} \in H^1(\Omega) \text{ such that} \\ u_{n+1} = g_{n+1}, \qquad \text{on } \Gamma, \\ \displaystyle\int_\Omega u_{n+1} v = \int_\Omega u_n\, v - \delta_n \int_\Omega \nabla u_n \cdot \nabla v + \delta_n \int_\Omega f_n\, v, \qquad \forall v \in H^1_0(\Omega), \end{array}\right.$$

[1]For knowledgeable mathematicians, I know, this sequence of problems is giving you the creeps. It is so ill–posed! You will have to wait to the fully discretized problem to get some satisfaction.

by the their discrete counterparts, which is easy: for each n we have to

$$\left[\begin{array}{l} \text{find } u^h_{n+1} \in V_h \text{ such that} \\ u^h_{n+1}(\mathbf{p}_i) = g_{n+1}(\mathbf{p}_i), \qquad \forall i \in \text{Dir}, \\ \displaystyle\int_\Omega u^h_{n+1} v_h = \int_\Omega u^h_n\, v_h - \delta_n \int_\Omega \nabla u^h_n \cdot \nabla v_h + \delta_n \int_\Omega f_n\, v_h, \qquad \forall v_h \in V^0_h. \end{array}\right.$$

This looks more like something we can do. Before going for matrices, we have to give a starting point for this recurrence: $u^h_0 \in V_h$ can be computed by interpolating in the nodes of the grid the initial data function u_0. This is not the best option, but it is definitely the simplest one.

We need to reintroduce matrices and vectors to give a simpler idea of what we are doing here in each time step. The nodal values of u_n are given in the vector $\mathbf{u}^n$. They are divided into values on free/interior nodes $\mathbf{u}^n_{\text{Ind}}$ and values on the Dirichlet nodes $\mathbf{u}^n_{\text{Dir}}$. Actually, the Dirichlet condition states that

$$\mathbf{u}^{n+1}_{\text{Dir}} = \mathbf{g}_{n+1},$$

where $\mathbf{g}_{n+1}$ is the vector of values of $g_{n+1} = g(\,\cdot\,, t_{n+1})$ on Dirichlet nodes.

We are going to pick up two pieces of the mass matrix

$$\mathbf{M}_{\text{Ind}} = \Big[\int_\Omega \varphi_j \varphi_i \Big]_{i,j\in\text{Ind}}, \qquad \mathbf{M}_{\text{Dir}} = \Big[\int_\Omega \varphi_j \varphi_i \Big]_{i\in\text{Dir}, j\in\text{Ind}}.$$

The matrix $\mathbf{M}_{\text{Ind}}$ is square shaped, with as many rows as there are interior nodes. On the other hand $\mathbf{M}_{\text{Dir}}$ is rectangular, with as many rows as there are interior nodes and one column per Dirichlet node. We will glue them together in the rectangular matrix

$$\mathbf{M}_{\text{all}} = \left[\ \mathbf{M}_{\text{Ind}} \,\middle|\, \mathbf{M}_{\text{Dir}}\ \right].$$

This division is made so that we can write products

$$\mathbf{M}_{\text{all}}\mathbf{u}^{n+1} = \mathbf{M}_{\text{Ind}}\mathbf{u}^{n+1}_{\text{Ind}} + \mathbf{M}_{\text{Dir}}\mathbf{u}^{n+1}_{\text{Dir}}.$$

Its rectangular shape reflects the fact that testing with nodal basis function is ignored. We similarly construct the matrices $\mathbf{W}_{\text{Dir}}$, $\mathbf{W}_{\text{Ind}}$ and $\mathbf{W}_{\text{all}}$.

At this stage of the course we have seen this kind of arguments enough times so that you will easily recognize that the step in variational form

$$\left[\begin{array}{l} \text{find } u^h_{n+1} \in V_h \text{ such that} \\ u^h_{n+1}(\mathbf{p}_i) = g_{n+1}(\mathbf{p}_i), \qquad \forall i \in \text{Dir}, \\ \displaystyle\int_\Omega u^h_{n+1} v_h = \int_\Omega u^h_n\, v_h - \delta_n \int_\Omega \nabla u^h_n \cdot \nabla v_h + \delta_n \int_\Omega f_n\, v_h, \qquad \forall v_h \in V^0_h, \end{array}\right.$$

is the same as the system

$$\left[\begin{array}{l} \mathbf{u}^{n+1}_{\text{Dir}} = \mathbf{g}_{n+1}, \\ \mathbf{M}_{\text{all}}\mathbf{u}^{n+1} = \mathbf{M}_{\text{all}}\mathbf{u}^n - \delta_n \mathbf{W}_{\text{all}}\mathbf{u}^n - \mathbf{f}_n, \end{array}\right.$$

where $\mathbf{f}_n$ is the vector with elements

$$\int_\Omega f_n \, \varphi_i, \qquad i \in \mathrm{Ind}.$$

We can also write each step as the solution of the system

$$\mathbf{M}_{\mathrm{Ind}} \mathbf{u}_{\mathrm{Ind}}^{n+1} = \mathbf{M}_{\mathrm{all}} \mathbf{u}^n - \delta_n \mathbf{W}_{\mathrm{all}} \mathbf{u}^n - \mathbf{f}_n - \mathbf{M}_{\mathrm{Dir}} \mathbf{g}_{n+1}$$

to compute only values on free nodes. Values on Dirichlet nodes are incorporated to this formulation but we have also to remind ourselves to keep them in the full vector $\mathbf{u}^{n+1}$, that will be used in the next time–step.

1.3 Some features of the method

Making the method really explicit. It may come to you as a surprise to see that working the explicit equations of the forward Euler method with Finite Elements you end up with a system to be solved in each time–step, so the explicit method is not so explicit after all. Note however that:

- The matrix is always the same and it is always the mass matrix, so you have good conditioning of the system together with symmetry and positive definiteness.
- Therefore if you do some preprocess (a factorization of the matrix) or you build a good preconditioner, it's going to be useful for all time steps. Moreover, for each time step you have a linear system that you can try to solve with an iterative method (Conjugate Gradient looks like the best option), but you have a guess of the starting point for the iterations: why not begin with the value in the previous time?
- If you are not happy yet with the implicit character of these equations, you can substitute the mass matrix (at least the one that appears on the left hand side) by the **lumped mass matrix**, which is diagonal. A diagonal system is immediate to solve.

Diffusion or propagation of heat? There are some good reasons to make the method completely explicit: you compute the time steps faster, since you don't have to solve any linear system, no matter how well conditioned this is. There are reasons not to make it fully explicit. In fact the argument I'm going to give to you here is somewhat tricky and you'll have to take it with a grain of salt. The real reason for going implicit is given in the stability analysis.

Let us consider just the first time step in the case where $f \equiv 0$ and $g \equiv 0$. We only have to compute the free nodes in all steps, because the boundary condition is homogeneous. Let us consider the $\mathbb{P}_1$ method and let us take a free node that is completely surrounded by free nodes. As initial condition we take an impulse in that node, that is, if the node is given the index i, we are starting with

$$u_0^h = \varphi_i.$$

In matrix form we are beginning with the vector $\mathbf{e}_i$, that has all components zero but the $i-$th that is one. This is the system we solve:

$$\mathbf{M}_{\mathrm{Ind}}\mathbf{u}^1_{\mathrm{Ind}} = \mathbf{M}_{\mathrm{Ind}}\mathbf{e}_i - \delta_0 \mathbf{W}_{\mathrm{Ind}}\mathbf{e}_i.$$

Note that the $i-$th row of $\mathbf{M}_{\mathrm{Ind}}$ and $\mathbf{W}_{\mathrm{Ind}}$ is the only one used in the right–hand side. It contains non–zero elements only on the positions of adjacent (neighboring) nodes. The vector $\mathbf{M}_{\mathrm{Ind}}\mathbf{e}_i - \delta_0 \mathbf{W}_{\mathrm{Ind}}\mathbf{e}_i$ propagates the unit value on the $i-$th node to its neighboring nodes. All other elements of this vector are still zero.

If you do mass lumping, that's all that is going to be non–zero in $\mathbf{u}^1$. In the next step, we will reach the following set of neighbors (neighbors of neighbors of the $i-$th node). What we are doing here is propagating heat at finite speed: the physics are all wrong! Heat diffusion is done at infinite speed. A unit impulse in time zero heats all the domain at any positive time. In truth, the values far from the heating source are very, very small at small times, but they are non zero. If we keep the mass matrix without lumping, at least it looks like we can reach all nodes in the first time step. The reason is the fact that $\mathbf{M}^{-1}_{\mathrm{Ind}}$ has most (if not all) elements non–zero. The process is much more similar to diffusion, although what we call diffusion, that's done by $\mathbf{W}^{-1}_{\mathrm{Ind}}$. But I cannot explain why right now.

Changing spaces with time. In some cases, with highly varying source terms and boundary conditions it could be wiser to change the finite element space from time to time, maybe even at all time–steps[2]. Think in the step $n \mapsto (n+1)$. Assume that u^n_h we computed with a $\mathbb{P}_1$ finite element on a given triangulation. The space is denoted $V_{h,n}$ and $V^0_{h,n}$ is the subspace obtained by eliminating the Dirichlet nodes. For whichever the reason, we are going to change grid and compute u^h_{n+1} in a new space $V_{h,n+1}$. In principle these are the discrete variational equations:

$$\left[\begin{array}{l} \text{find } u^h_{n+1} \in V_{h,n+1} \text{ such that} \\ u^h_{n+1}(\mathbf{p}_i) = g_{n+1}(\mathbf{p}_i), \qquad \forall i \in \mathrm{Dir}(n+1), \\ \displaystyle\int_\Omega u^h_{n+1} v_h = \int_\Omega u^h_n\, v_h - \delta_n \int_\Omega \nabla u^h_n \cdot \nabla v_h + \delta_n \int_\Omega f_n\, v_h, \qquad \forall v_h \in V^0_{h,n+1}. \end{array}\right.$$

It looks the same but it isn't exactly the same. If you add a superindex with the discrete time to the nodal bases, you will see that in the left–hand side, you have a usual mass matrix for the current space

$$\int_\Omega \varphi^{n+1}_j \varphi^{n+1}_i.$$

However, since u^h_n was computed on the old grid, the two matrices that appear on the right–hand side are

$$\int_\Omega \varphi^n_j \varphi^{n+1}_i \qquad \text{and} \qquad \int_\Omega \nabla\varphi^n_j \cdot \nabla\varphi^{n+1}_i.$$

[2]This change of space with time is the daily bread in finite element methods for waves, but since we are taking the heat equation as the first model problem, it's okay if we have a look at this here.

These matrices do not need even to be square. But there's more. The very nice idea of assembly is much more complicated if the triangulations are not related and what we did in Lesson 2 is definitely not valid here anymore. With this naïve approach things really get messy.

What can be done in practice is taking a very different approach, consisting of pre-processing the solution in time n to move it to the grid of time $n+1$. In essence it is like interpolating u_n^h to the new space $V_{h,n+1}$. This can be a somewhat complicated process but has the advantage that the u_n^h we input in the right–hand side is now in the same space as the u_{n+1}^h we want to compute and the assembly process can be used again.

1.4 Stability analysis

Let's simplify again the problem to have source term and boundary conditions that do not depend on time. The problem is therefore

$$\left[\begin{array}{ll} u_t = \Delta_{\mathbf{x}} u + f, & \text{in } \Omega \times (0,\infty), \\ u(\,\cdot\,,0) = u_0, & \text{in } \Omega, \\ u(\,\cdot\,,t) = g, & \text{on } \Gamma \text{ for all } t>0, \end{array}\right.$$

with f and g independent of time. If we ignore the initial condition we can look for the only steady–state solution to the problem

$$\left[\begin{array}{ll} -\Delta u_{\lim} = f, & \text{in } \Omega, \\ u_{\lim} = g, & \text{on } \Gamma. \end{array}\right.$$

Assume now that we know all Dirichlet eigenvalues and eigenfunctions of the Laplace operator in Ω:

$$\left[\begin{array}{ll} -\Delta \phi_k = \lambda_k \phi_k, & \text{in } \Omega, \\ \phi_k = 0. & \text{on } \Gamma. \end{array}\right.$$

The solution to the heat diffusion problem is

$$u(\mathbf{x},t) = u_{\lim}(\mathbf{x}) + \sum_{k=1}^{\infty} c_k\, e^{-\lambda_k t} \phi_k(\mathbf{x}), \qquad c_k = \int_\Omega (u_0 - u_{\lim})\, \phi_k.$$

This formula[3] shows that the solution goes exponentially fast to the steady–state solution. The occurrence of negative exponentials at increasing velocities (λ_k diverges as k goes to infinity) makes the initial times very hard to compute with precision.

In case we are dealing with zero data

$$f \equiv 0, \qquad g \equiv 0,$$

[3]You might (should) recognize it from your course(s) on differential equations. It is the solution obtained by separation of variables

the formula for the solution is really simple: it's just diffusion of the initial condition towards the zero solution

$$u(\mathbf{x},t)=\sum_{k=1}^{\infty} c_k\, e^{-\lambda_k\, t}\phi_k(\mathbf{x}), \qquad c_k=\int_\Omega u_0\,\phi_k.$$

Let us see what the numerical method does. Since boundary conditions vanish we don't have to take into account Dirichlet nodes. In the $n-$th time–step we solve

$$\mathbf{M}_{\mathrm{Ind}}\mathbf{u}^{n+1}_{\mathrm{Ind}}=\mathbf{M}_{\mathrm{Ind}}\mathbf{u}^{n}_{\mathrm{Ind}}-\delta_n\mathbf{W}_{\mathrm{Ind}}\mathbf{u}^{n}_{\mathrm{Ind}}.$$

Let us drop the Ind subindex and keep in mind that we are only computing in the interior nodes. Also for simplicity assume that $\delta_n=\delta$ for all n, that is, we are using a fixed time step. This is the very simple $n-$th time step:

$$\mathbf{M}\mathbf{u}^{n+1}=\mathbf{M}\mathbf{u}^{n}-\delta\mathbf{W}\mathbf{u}^{n}.$$

There is only a finite number of linearly independent eigenvectors (that are nodal values of the discrete eigenvectors):

$$\mathbf{W}\boldsymbol{\phi}_k=\lambda_{h,k}\mathbf{M}\boldsymbol{\phi}_k.$$

Maybe you should go back to Section 4 of Lesson 4 to review this. Recall that $\lambda_{h,k}\ge\lambda_k$ approximates this $k-$th exact eigenvalue for h sufficiently small. Take $\mathbf{u}^0=\boldsymbol{\phi}_k$ as initial condition in the recurrence that determines the discrete time steps. Then the equation for the first time–step is

$$\mathbf{M}\mathbf{u}^{1}=\mathbf{M}\boldsymbol{\phi}_k-\delta\mathbf{W}\boldsymbol{\phi}_k=(1-\lambda_{h,k}\delta)\,\mathbf{M}\boldsymbol{\phi}_k.$$

Therefore, using the fact that $\mathbf{M}$ is invertible, we have $\mathbf{u}^1=(1-\delta\lambda_{h,k})\boldsymbol{\phi}_k$. The following time steps are similar and we obtain the following formula for all the time–steps

$$\mathbf{u}^n=(1-\lambda_{h,k}\delta)^n\boldsymbol{\phi}_k.$$

Note that $\lambda_{h,k}$ is trying to approximate λ_k and $\boldsymbol{\phi}_k$ is trying to approximate the nodal values of ϕ_k. The formula for the recurrence is trying to approximate the diffusive solution

$$e^{-\lambda_k\delta n}\phi_k=e^{-\lambda_k t_n}\phi_k.$$

Is it doing a good job? Independently of whether this approximation is good or not, let us just look at the asymptotic behavior. The exact solution goes to zero as n goes to infinity. What about the discrete solution? Well, not always. It will do the right thing if

$$|1-\lambda_{h,k}\delta|<1,$$

which is equivalent (note that δ and $\lambda_{h,k}$ are positive) to

$$\lambda_{h,k}\delta<2.$$

This should be satisfied for all discrete eigenvalues. Since we have ordered them from smallest to largest, it has to be satisfied by the largest of them

$$\lambda_{h,N(h)}\delta < 2.$$

Why do I say that it has? The fact is that any initial condition can be decomposed as

$$\mathbf{u}^0 = \sum_{k=1}^{N(h)} c_k \boldsymbol{\phi}_k$$

and the corresponding discrete evolution is therefore

$$\mathbf{u}^n = \sum_{k=1}^{N(h)} c_k (1 - \lambda_{h,k}\delta)^n \boldsymbol{\phi}_k.$$

The orthogonality condition of the discrete eigenvector proves that $\mathbf{u}^n$ goes to zero as $n \to \infty$ (that is, it has the correct asymptotic value) if and only if all conditions $\lambda_{h,k}\delta < 2$ hold.

Let's discuss the condition

$$\lambda_{h,N(h)}\delta < 2.$$

If we take the fixed time–step to begin with, the condition is of the form

$$\lambda_{h,N(h)} < 2/\delta.$$

Note that $\lambda_{h,N(h)} \geq \lambda_{N(h)}$. If we take a very fine grid (a very precise finite element method) it is very likely that you are getting to capture a very large eigenvalue and the stability condition does not hold any longer. This **conditional stability** says that given the time–step you can only try to do *this good* with finite elements, but if you try to be too precise you lose stability. This may be a shock to you. One would think that each part of the discretization process can be done as precisely as possible without taking care of the others. The conditional stability denies that.

If you fix the finite element grid, the inequality can be read as

$$\delta < 2/\lambda_{h,N(h)}$$

which says that you have to take time–steps that are short enough in order not to lose stability[4]. It is difficult to make oneself an idea of how the largest discrete eigenvalue grows with finer grids. For the one dimensional problem the precise formula is known. Given the variety of polygonal domains you can think of, the question is less clear in two dimensions.

[4]People in the ODE discretization community call this problem stiff and say that explicit methods are not linearly stable and should not be applied (or applied with great care) to stiff problems. More on this in Section 3.

Remark. In fact, the term $(1-\lambda_{h,k}\delta)^n$ can be oscillating even when going to zero, so we even might like it to be positive in addition to convergent to zero. The condition is then $\lambda_{h,k}\delta < 1$. $\Box$

What else? Convergence of course. Well, let's not do this here. The issue becomes really difficult. Note only that: (a) use of forward Euler in time means you should expect no more that error proportional to time step (order one); (b) the effort made in the space discretization should agree with the low order in time; (c) imposition of non–homogeneous Dirichlet conditions becomes openly critical here. Doing the simplest thing here makes you lose convergence order. You have to look at the theory (and we are so not going to to that now) to understand why. Anyway, never use high order in space with low order in time. You are wasting your efforts. Second, be careful with stability. You don't have it for free! If you have fixed your time–step you cannot try to be too precise in space.

2 FEM with backward Euler for the heat equation

It has taken time, but have had a very close look at the very simplest discretization method for the heat equation. If you have your FEM code for the steady state problem, it is easy to create a FEM code for the forward Euler and FEM discretization of the heat equation. We move now to improve our method.

2.1 Time semidiscretization with backward Euler

First thing we have to improve is conditional stability. That condition is definitely not the best thing to have, in particular since you really don't know precisely whether it holds or not unless you compute the largest generalized eigenvalue of $\mathbf{W}$.

We begin from scratch. Almost. The backward Euler discretization uses the same quotient as the forward Euler method but to approximate the value of the derivative in discrete time $(n+1)$

$$\phi'(t_{n+1}) \approx \frac{\phi(t_{n+1})-\phi(t_n)}{t_{n+1}-t_n} = \frac{\phi(t_{n+1})-\phi(t_n)}{\delta_n}.$$

Correspondingly, we look at the heat equation in time t_{n+1} and impose the backward Euler approximation

$$\frac{u_{n+1}-u_n}{\delta_n} = \Delta u_{n+1} + f_{n+1},$$

or equivalently

$$-\delta_n \Delta u_{n+1} + u_{n+1} = u_n + \delta_n f_n.$$

Let's not forget the boundary condition, which now enters the game naturally

$$u_{n+1} = g_{n+1}, \qquad \text{on } \Gamma.$$

Equation and boundary condition constitute a boundary value problem like those we have been studying all along this course. Note that the diffusion parameter is the time step (it is very small) but that this parameter is also multiplying the source term. If you formally take it to zero, what you obtain is a constant solution, which is what happens with evolution when you stop the clock counting times.

The boundary value problem to obtain u_{n+1} has nothing special. Its weak formulation is done in the usual way, as if there was no time in the equation

$$\left[\begin{array}{l} \text{find } u_{n+1} \in H^1(\Omega) \text{ such that} \\ u_{n+1} = g_{n+1}, \qquad \text{on } \Gamma, \\ \delta_n \displaystyle\int_\Omega \nabla u_{n+1} \cdot \nabla v + \int_\Omega u_{n+1} v = \int_\Omega u_n\, v + \delta_n \int_\Omega f_n\, v, \qquad \forall v \in H^1_0(\Omega). \end{array}\right.$$

2.2 Full discretization

Taking the finite element space instead of the exact Sobolev space, we obtain a sequence of problems

$$\left[\begin{array}{l} \text{find } u^h_{n+1} \in V_h \text{ such that} \\ u^h_{n+1}(\mathbf{p}_i) = g_{n+1}(\mathbf{p}_i), \qquad \forall i \in \text{Dir}, \\ \delta_n \displaystyle\int_\Omega \nabla u^h_{n+1} \cdot \nabla v_h + \int_\Omega u^h_{n+1} v_h = \int_\Omega u^h_n\, v_h + \delta_n \int_\Omega f_n\, v_h, \qquad \forall v_h \in V^0_h. \end{array}\right.$$

The recurrence (the time–steps) has to be started with an initial condition of u^h_0 given, as we had in the explicit method. You can go back to the previous section and you will notice that the only serious change is the stiffness term changing sides. It is implicit now.

Using the same notations for the vectors of unknowns and for the pieces of the matrices, we have a fully implicit method now

$$\left[\begin{array}{l} \mathbf{u}^{n+1}_{\text{Dir}} = \mathbf{g}_{n+1}, \\ \Big(\delta_n \mathbf{W}_{\text{all}} + \mathbf{M}_{\text{all}}\Big)\mathbf{u}^{n+1} = \mathbf{M}_{\text{all}}\mathbf{u}^n - \mathbf{f}_n, \end{array}\right.$$

Note again that the stiffness matrix has changed sides in the system. The system to be solved in each time step is actually

$$\Big(\delta_n \mathbf{W}_{\text{Ind}} + \mathbf{M}_{\text{Ind}}\Big)\mathbf{u}^{n+1}_{\text{Ind}} = \mathbf{M}_{\text{all}}\mathbf{u}^n + \Big(\delta_n \mathbf{W}_{\text{Dir}} + \mathbf{f}_n - \mathbf{M}_{\text{Dir}}\Big)\mathbf{g}_{n+1}.$$

You can take from here a first idea: the cost of programming the forward and the backward Euler is exactly the same. The main difference is that in the implicit method you have to solve a linear system in each time step and there is not diagonal approximation for the corresponding matrix. The matrix itself varies with time–step, but if you have to look for a preconditioner, you just have to take care of the stiffness matrix, which is the bad guy here (mass=good, stiffness=bad) in terms of conditioning. For fixed time stepping, the matrix is always the same, by the way.

If you put a point source in time zero, it diffuses instantaneously to the whole domain thanks to the inverse of the matrix of the system.

2.3 Stability analysis

With vanishing boundary conditions and zero sources as well as with fixed time–step we solve the recurrence

$$(\delta\mathbf{W}+\mathbf{M})\mathbf{u}^{n+1}=\mathbf{M}\mathbf{u}^{n}$$

to follow the free evolution of the system with initial condition $\mathbf{u}^0$. If $\mathbf{u}^0=\boldsymbol{\phi}_k$ (we use the same notation as in the corresponding subsection for the forward method), then the eigenvectors satisfy

$$(\delta\mathbf{W}+\mathbf{M})\boldsymbol{\phi}_k=(1+\lambda_{h,k}\delta)\mathbf{M}\boldsymbol{\phi}_k.$$

Therefore, it is simple to check that

$$\mathbf{u}^n=(1+\lambda_{h,k}\delta)^{-n}\boldsymbol{\phi}_k.$$

This discrete evolution is always correct, since $0<(1+\lambda_{h,k}\delta)^{-1}<1$. The method is therefore **unconditionally stable**. Expected convergence is similar to the one of the forward Euler approximation, since both time discretizations have the same order. What changes here is stability.

3 Doing first space and then time

In of the exercises I've proposed to have a look at the scheme developed by John Crank and Phyllis Nicolson using the same quotient to approximate the average of the derivatives in both points. It leads to a sort of average of the forward and backward Euler methods[5]. This is an easy way of increasing order of convergence in time: formally it goes up to order two. Doing better with finite differences in time requires using more time points for each steps. We could also forget about finite differences in time and do Galerkin (finite elements) also in that variable.

Instead we are going to try something else. The following approach is the origin of many ideas but definitely requires that your space triangulation remains fixed, so forget about it if things are changing really fast and you want to remesh from time to time.

We are back to the heat diffusion problem. Here it is again

$$\left[\begin{array}{ll} u_t=\Delta_{\mathbf{x}}u+f, & \text{in } \Omega\times(0,\infty),\\ u(\,\cdot\,,0)=u_0, & \text{in } \Omega,\\ u(\,\cdot\,,t)=g, & \text{on } \Gamma \text{ for all } t>0.\end{array}\right.$$

For the moment, let us think of time as an additional parameter, forget the initial condition and deal with this as an elliptic problem. For each t, the space function $u=u(\,\cdot\,,t)$ (mathematicians, forgive me for not changing the name) satisfies:

$$\left[\begin{array}{ll} -\Delta u+u_t=f, & \text{in } \Omega,\\ u=g, & \text{on } \Gamma.\end{array}\right.$$

[5]Note that properly speaking the Crank–Nicolson scheme uses also finite differences for the space variables.

Using Green's Theorem we obtain a weak formulation

$$\left[\begin{array}{ll} u = g, & \text{on } \Gamma, \\ \displaystyle\int_\Omega \nabla u \cdot \nabla v + \int_\Omega u_t\, v = \int_\Omega f\, v, & \forall v \in H^1_0(\Omega). \end{array}\right.$$

Hey, teacher! Your forgot to write the space for u! No, I didn't. We can try to think of u as a function that for each t, gives an element of $H^1(\Omega)$, but I really prefer not to write the correct spaces. First of all, because they are complicated. Second,... because they are complicated, if we want to have the right spaces where we are certain to have a solution and not some safe spaces where everything looks nice but we will never be able to show that there is a solution.

Instead, let us go to discretization. The idea is the same: for each time we associate a function in the finite element space (it will be the same space for all times). So, fix V_h and V_h^0 as usual. A time–dependent element of V_h is something of the form

$$u_h(t,\mathbf{x}) = \sum_{j=1}^N u_j(t)\, \varphi_j(\mathbf{x}).$$

The coefficients vary with time, but the global basis is always the same since the triangulation is fixed. In fact, when we are dealing with the nodal basis functions $u_j(t) = u_h(t,\mathbf{p}_j)$, so we are following the nodal values of the discrete function. The partial derivative of this function with respect to time is

$$\sum_{j=1}^N \dot u_j\, \varphi_j.$$

Then, the semidiscrete in space problem looks for u_h such that for all t

$$\left[\begin{array}{ll} u_h(\,\cdot\,,t) \in V_h, & \\ u_h(\mathbf{p},t) = g(\mathbf{p},t), & \text{for all } \mathbf{p} \text{ Dirichlet node,} \\ \displaystyle\int_\Omega \nabla_{\mathbf{x}} u_h \cdot \nabla v_h + \int_\Omega u_{h,t}\, v_h = \int_\Omega f\, v_h, & \forall v \in V_h^0. \end{array}\right.$$

We also need an initial condition

$$u_h(\,\cdot\,,0) = \sum_j u_j(0)\varphi_h = u_h^0,$$

where $u_h^0 \in V_h$ approximates the initial condition u_0. If we decide ourselves for interpolating data, this means that we are giving an initial condition to the coefficients

$$u_j(0) = u_0(\mathbf{p}_j), \qquad \forall j.$$

The problem can be easily written using these coefficients

$$\left[\begin{array}{ll} u_i(t) = g(\mathbf{p}_i,t), & \forall i \in \text{Dir}, \\ \displaystyle\sum_{j=1}^N \Big(\int_\Omega \nabla\varphi_i \cdot \nabla\varphi_i\Big) u_j(t) + \sum_{j=1}^N \Big(\int_\Omega \varphi_j\, \varphi_i\Big) \dot u_j(t) = \int_\Omega f\, \varphi_i, & \forall i \in \text{Ind}. \end{array}\right.$$

This system holds for all t. This is a somewhat non–standard but simple differential system. We can get rid of the algebraic (non–standard) part by simply substituting the Dirichlet conditions inside the formulation to obtain

$$\sum_{j\in\mathrm{Ind}} w_{ij}u_j(t) + \sum_{j\in\mathrm{Ind}} m_{ij}\dot u_j(t) = \int_\Omega f\varphi_i - \sum_{j\in\mathrm{Dir}}\Big(w_{ij}g(t,\mathbf{p}_j) - m_{ij}g_t(t,\mathbf{p}_j)\Big), \qquad \forall i\in\mathrm{Ind}.$$

This looks much more like a system of linear differential equations. Let us simplify expression by adding notations. We consider the following functions of time:

$$f_i(t) = \int_\Omega f(\,\cdot\,,t)\varphi_i, \qquad i\in\mathrm{Ind},$$
$$g_j(t) = g(t,\mathbf{p}_j), \qquad j\in\mathrm{Dir}.$$

The system is therefore

$$\sum_{j\in\mathrm{Ind}} w_{ij}u_j(t) + \sum_{j\in\mathrm{Ind}} m_{ij}\dot u_j(t) = f_i(t) - \sum_{j\in\mathrm{Dir}}\Big(w_{ij}g_j(t) - m_{ij}\dot g_j(t)\Big), \qquad \forall i\in\mathrm{Ind}.$$

You will have noticed that this way of discretizing the problem, imposes the need to compute the time derivative of the Dirichlet data. It's because they are essential (Neumann data would appear like source terms, happily placed inside integral signs). If you want to avoid this derivative of data, you have to deal with the algebraic–differential system as was first obtained.

Using the matrix notation introduced in the first section of this lesson, we can write

$$\mathbf{W}_{\mathrm{Ind}}\mathbf{u}_{\mathrm{Ind}} + \mathbf{M}_{\mathrm{Ind}}\dot{\mathbf{u}}_{\mathrm{Ind}} = \mathbf{f} - \mathbf{W}_{\mathrm{Dir}}\mathbf{g} - \mathbf{M}_{\mathrm{Dir}}\dot{\mathbf{g}}.$$

Now, we write everything together, more in the style of how we write differential systems:

$$\left[\begin{array}{l} \mathbf{u}_{\mathrm{Ind}}(0) = \mathbf{u}_0, \\ \mathbf{M}_{\mathrm{Ind}}\dot{\mathbf{u}}_{\mathrm{Ind}} = -\mathbf{W}_{\mathrm{Ind}}\mathbf{u}_{\mathrm{Ind}} + \mathbf{f} - \mathbf{W}_{\mathrm{Dir}}\mathbf{g} - \mathbf{M}_{\mathrm{Dir}}\dot{\mathbf{g}}. \end{array}\right.$$

This is a linear system of differential equations (with initial values) given in implicit form. To make it explicit you would have to premultiply by $\mathbf{M}_{\mathrm{Ind}}^{-1}$. In principle you don't have to compute the inverse of the mass matrix to know how to multiply by it. The reason is the fact that

the vector $\mathbf{M}_{\mathrm{Ind}}^{-1}\mathbf{v}$ *is the solution to the system* $\mathbf{M}_{\mathrm{Ind}}\mathbf{x} = \mathbf{v}$.

Therefore, you just need to know how to solve linear systems with $\mathbf{M}_{\mathrm{Ind}}$ as matrix. You don't even need that much. Most packages that solve numerically systems of differential equations (with Runge–Kutta methods for instance) already consider the implicit situation, where the derivative is premultiplied by an invertible matrix.

This approach allows you to use high order in space and high order in time very easily, because the processes are separated. In fact, many people in the numerical ODE community use the heat equation after space discretization as a benchmark for their methods, since the resulting system is stiff. Remember all those fastly decaying exponentials in the separation of variable solutions? In the differential system they become large negative eigenvalues, which are difficult to handle. For stiff problems, the safe bet is the use implicit methods. Anything explicit will be understandably conditionally convergent, requiring short time steps or a very rigid step control strategy.

Remark. If you apply the forward or backward Euler method to this differential system you obtain the methods you had in Sections 1 and 2 if:

- g is independent of time
- g depends on time but you substitute the occurrence of $\dot{\mathbf{g}}$ in the $n-$th time step by the quotient $(\mathbf{g}_{n+1} - \mathbf{g}_n)/\delta_n$.

This coincidence of lower order methods in the simplest cases is something you find over and over in numerical analysis. □

4 Some ideas about the wave equation

There is a long stretch since the beginning of this course, ninety–something pages ago. We need to put an end to it, but it would be wrong (for me) to end a lesson of evolution problems with nothing on the wave equation[6]. You'll see how this is very simple to introduce. To make it simpler we will use homogeneous Dirichlet conditions in the entire boundary of the domain.

The wave propagation problem is then

$$\left[\begin{array}{ll} u_t = \Delta_{\mathbf{x}} u + f, & \text{in } \Omega \times (0,\infty), \\ u(\,\cdot\,,0) = u_0, & \text{in } \Omega, \\ u_t(\,\cdot\,,0) = v_0, & \text{in } \Omega, \\ u(\,\cdot\,,t) = 0, & \text{on } \Gamma \text{ for all } t > 0. \end{array}\right.$$

If we try the finite difference in time approach, the simplest thing to do is to apply the central difference approximation (some people call this Newmark's method[7]) to the second derivative. If we take a fixed time step, this means approximating

$$\phi''(t_n) \approx \frac{\phi(t_{n+1}) - 2\phi(t_n) + \phi(t_{n-1})}{\delta^2}.$$

When applied to the time–variable in the wave equation we obtain the explicit time–step

$$\frac{u_{n+1} - 2u_n + u_{n-1}}{\delta^2} = \Delta u_n + f_n.$$

After doing the weak formulation and introducing finite element spaces and bases, we end up with

$$\mathbf{M}\mathbf{u}_{n+1} = 2\mathbf{M}\mathbf{u}_n - \mathbf{M}\mathbf{u}_{n-1} - \delta^2 \mathbf{W}\mathbf{u}_n + \delta^2 \mathbf{f}_n.$$

[6]You can easily claim that I'm not dealing with conservation laws either. True. You are right. That's not my turf.

[7]As far as I know about this, the method proposes by Nathan Newmark is something more general destined to approximate second order equations. There is however a developed habit of calling this central difference approximation for the time derivative in the wave equation, Newmark's method.

(Only free nodes appear in all the expressions, since we have taken homogeneous Dirichlet boundary conditions). The initial value for $\mathbf{u}_0$ is easy. You have data. You still need $\mathbf{u}_1$ (the nodal values of u_1^h. For that, you can do very easy (and not very well) by taking a Taylor approximation

$$u_1 = u_0 + \delta v_0,$$

or take a false discrete time -1 and use the equation

$$\frac{u_1 - 2u_0 + u_{-1}}{\delta^2} = \Delta u_0 + f_0$$

together with the central difference approximation

$$\frac{u_1 - u_{-1}}{2\delta} = v_0$$

to obtain the equation

$$u_1 = \tfrac{1}{2}\delta^2 \Delta u_0 + u_0 + \delta v_0 + \tfrac{1}{2}\delta^2 f_0.$$

Then you need to give a weak formulation of this too. And do all the finite element stuff. Nothing you don't know how to do. Some really fast last strokes:

- Space discretization has made the equations implicit but it's only with the mass matrix. To obtain the good physics (finite velocity of propagation), the use of the lumped mass matrix is highly recommended. Wait for a couple of points to know more about this.

- The method is explicit so it is going to be **conditionally stable**. The stability condition is a bit harder to derive in this situation. It reads like

$$\delta^2 \lambda_{h,N} < 4$$

  and it is called a Courant–Friedrichs–Lewy condition[8] and always refered by the initials CFL condition.

- Things with the wave equation happen quite fast so most people are willing to accept the short time–step imposed by a CFL condition, since they want to observe the propagation anyway.

- Implicit methods have the advantage of unconditional stability but get the physics wrong. When you are trying to follow the propagation of wave–fronts you sort of dislike the diffusion that would be caused by the presence of the inverse of the stiffness matrix.

- Wave propagation is however a delicate matter. If you take the explicit method, made fully explicit by the use of mass lumping, you move (in the $\mathbb{P}_1$ method) from node to node in each time step. That is, the speed of numerical propagation is

[8]We have already met Richard Courant, moral father of the $\mathbb{P}_1$ element. Now, meet Kurt Friedrichs and Hans Lewy. All three of them were German (Lewy's birthplace counts as Poland nowadays) and moved to America.

controlled by the time step. If you take a very, very short time–step to be sure that you are satisfying the CFL condition, you may be going too fast, so you have to play it safe but not too safe. This balance between stability and correct speed of propagation makes the discretization of wave phenomena a difficult but extremely interesting problem.

5 Exercises

E5.1. Crank–Nicolson and FEM for the heat equation

The Crank–Nicolson scheme consists of using the quotient to approximate the average of the derivative in t_n and t_{n+1}:

$$\frac{1}{2}\phi'(t_{n+1}) + \frac{1}{2}\phi'(t_n) \approx \frac{\phi(t_{n+1}) - \phi(t_n)}{t_{n+1} - t_n} = \frac{\phi(t_{n+1}) - \phi(t_n)}{\delta_n}.$$

We can apply this to the heat equation and propose this problem as $n-$th time step:

$$\left[\begin{array}{ll} \dfrac{u_{n+1} - u_n}{\delta_n} = \dfrac{1}{2}\Big(\Delta u_n + \Delta u_{n+1}\Big) + \dfrac{1}{2}(f_n + f_{n+1}), & \text{in } \Omega \\ u_{n+1} = g_{n+1}, \qquad \text{on } \Gamma. & \end{array}\right.$$

- Write the preceding time–step as a reaction–diffusion problem to compute u_{n+1}.
- Write a weak formulation taking care of not having the Laplacian of u_n in the right–hand side but a stiffness term (you will have to use Green's formula twice, once in t_n and once in t_{n+1}).
- Write the discrete equations obtained from the FEM discretization of the weak formulation.
- Show that the method is unconditionally stable (use the same particular case: fixed time–step, $f \equiv 0$ and $g \equiv 0$).

E5.2. Full discretization of the wave equation

We have already said that from the three terms recurrence

$$\frac{u_{n+1} - 2u_n + u_{n-1}}{\delta^2} = \Delta u_n + f_n,$$

a finite element method gives you this other full discrete three–term recurrence

$$\mathbf{M}\mathbf{u}_{n+1} = 2\mathbf{M}\mathbf{u}_n - \mathbf{M}\mathbf{u}_{n-1} - \delta^2\mathbf{W}\mathbf{u}_n + \delta^2\mathbf{f}_n.$$

Prove it. (You just have to follow step by step what we did for the heat equation and the forward Euler discretization. Note again the we have dropped the subscript Ind everywhere).

E5.3. Space semidiscretization of the wave equation

We begin again

$$\left[\begin{array}{ll} u_t = \Delta_{\mathbf{x}} u + f, & \text{in } \Omega \times (0,\infty), \\ u(\,\cdot\,,0) = u_0, & \text{in } \Omega, \\ u_t(\,\cdot\,,0) = v_0, & \text{in } \Omega, \\ u(\,\cdot\,,t) = 0, & \text{on } \Gamma \text{ for all } t > 0. \end{array}\right.$$

(Note that we have homogeneous Dirichlet boundary conditions). Taking the approach of space–first, prove that we arrive at a system of differential equations of the second order:

$$\mathbf{M}_{\mathrm{Ind}} \ddot{\mathbf{u}}_{\mathrm{Ind}} + \mathbf{W}_{\mathrm{Ind}} \mathbf{u}_{\mathrm{Ind}} = \mathbf{f}.$$

You just have to follow carefully the same process for the heat equation, with the additional simplification of having zero boundary conditions. To finish, note that we have two initial conditions that we can incorporate to the differential system.

Appendices: Additional matters

1 Getting organized with the $\mathbb{P}_1$ method

A important aspect of the implementation of the Finite Element Method (and of any other non–trivial numerical method) is the choice of good data structures to store the necessary information. We are going to detail here one usual option for these structures. Keep in mind that we are going to see much more complicated methods in the sequel, so part of what is done here is preparing the ground for more complicated situations.

We assume that the boundary is divided in sides with a numbering of boundary subdomains. It is necessary to know what the boundary condition is on each sumdomain and how to evaluate the corresponding function for the boundary condition. The following

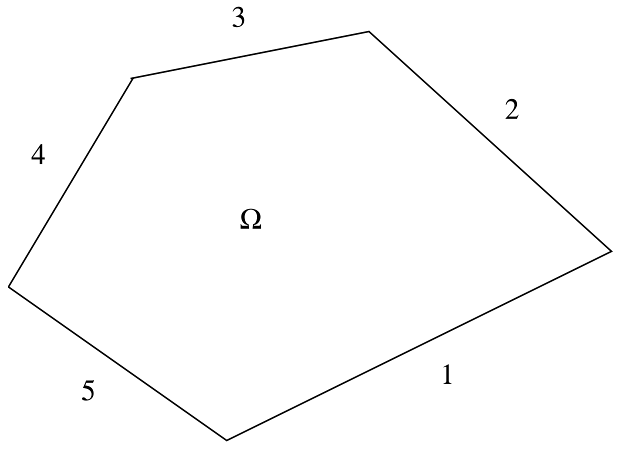


Figure 5.1: Numbering of the sides of the boundary

data are needed for the implementation of the $\mathbb{P}_1$ finite element method:

- The global number of nodes nNod.
- A numbering of the nodes and their coordinates. This can simply be done by giving a double vector with the coordinates of all nodes. Numbering is done by component:

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \end{bmatrix}$$

- The number of triangles.
- A relation of nodes elementwise:

$$\begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ \vdots & \vdots & \vdots \end{bmatrix}$$

  (for instance, the 4th row of this matrix are the global indices for the 1st, 2nd and 3rd vertices of the fourth triangle).
- A list of Dirichlet nodes (Dir), mentioning on what boundary subdomain they are so that we know which function to evaluate.
- The number of Neumann edges (edges that lie on the Neumann boundary)
- A list of the Neumann edges, indicating what their vertices are and on which boundary subdomain they lie.

Usually grid generators give Dirichlet edges instead of nodes, i.e.,

- A list of Dirichlet edges (edges on the Dirichlet boundary), indicating what their vertices are and on which boundary subdomain they lie.

From this list, the construction of the list Dir and the complementary list Ind is a simple preprocess that has to be performed before the assembly process is begun.

In the example we have been following along this Lesson and the previous one, we have the following data:

- 18 nodes (of which 6 are Dirichlet nodes)
- 23 triangles
- 6 Neumann edges
- Relation between local and global numbering of vertices for triangles:

$$\text{23 rows}\left\{\begin{bmatrix} 1 & 3 & 2 \\ 1 & 4 & 3 \\ 4 & 8 & 3 \\ 3 & 6 & 2 \\ 3 & 7 & 6 \\ 3 & 8 & 7 \\ 7 & 12 & 6 \\ 8 & 12 & 7 \\ 4 & 9 & 8 \\ \vdots & \vdots & \vdots \end{bmatrix}\right.$$

- A list of Dirichlet nodes, indicating the boundary subdomain (the side of Γ) where they are
$$\begin{bmatrix} 9 & 1 \\ 13 & 1 \\ 17 & 1 \\ 18 & 2 \\ 15 & 2 \\ 14 & 2 \end{bmatrix}$$
(in this list it is not relevant that the order is increasing). Node number 18 could be placed on the 2nd or the 1st side. Since the Dirichlet condition cannot be discontinuous in this formulation, it is immaterial which choice is taken.

- The list Ind is obtained from the list $\{1, 2, \ldots, 18\}$ by erasing everything that appears on the firs column of Dir
$$\mathrm{Ind} = (1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 16)$$

- List of Neumann edges (the third column is the number of the side of Γ where they are:
$$\begin{bmatrix} 10 & 14 & 3 \\ 5 & 10 & 3 \\ 2 & 5 & 4 \\ 1 & 2 & 4 \\ 1 & 4 & 5 \\ 4 & 9 & 5 \end{bmatrix}$$

- Instead of the list of Dirichlet nodes with their associated boundary side, we could be given a list of Dirichlet edges with the boundary side (third column)
$$\begin{bmatrix} 9 & 13 & 1 \\ 13 & 17 & 1 \\ 17 & 18 & 1 \\ 18 & 15 & 2 \\ 15 & 14 & 2 \end{bmatrix}$$
and build therewith both Dir and Ind.

2 The one dimensional problem

3 Bibliography