Part II

Discretization in space

Chapter 9

Elliptic partial differential equations and steady-state problems

9.1 The one-dimensional heat equation

9.1.1 General form

LeVeque, section 2.1.

9.1.2 Boundary conditions

LeVeque, section 2.2

9.1.3 The steady-state equation

LeVeque, section 2.3

9.2 The two-dimensional Poisson equation

LeVeque, section 3.1

9.3 Classes of spatial discretisation methods

Add lecture overview of classes of discretisation methods.

When discretising in time, everything is natural: you have a solution at one moment in time, which you want to push forward to some next moment in time. Then, there are some ways to do this, but they all amount to "time-stepping".

For the space discretisation of elliptic partial differential equations, one needs to find the solution at each point in space *simultaneously.* The unknown solution is a function that needs to be found as a whole! We thus need a finite-dimensional representation of that solution. In the literature, there are four mainstream ideas to achieve this goal.

9.3.1 Finite difference methods

One represents an unknown function $u(x)$ by its value on a fixed grid, consisting of a discrete number of grid point $\{x\}_{i=1}^{I}$. One then needs to replace the spatial derivatives that one encounters with finite difference approximation on the grid.

Methods that do this are called (not surprisingly) *finite difference methods.* They are the most easy methods to understand, but can only be implemented easily if the domain of the PDE is regular, if you don't have too irregular grids, etc.

In this course, we will mainly study finite differences methods. They are not the most common methods in practice, but the phenomena that arise in the numerical solutions also arise with different methods, and they can easily be understood in this particular setting.

9.3.2 Finite element methods

An alternative is to represent the solution in a finite-dimensional subspace of the full (infinite-dimensional) space in which the solution of the partial differential equation lives. One way to construct the finite-dimensional space is by choosing *local basis functions* (functions that are zero on most of the domain, and only non-zero locally).

This method (the *finite element method*) is much more commonly used in practice. This approach is very flexible, lends itself well to convergence analysis, and for relatively straightforward implementation on irregular domains. The drawback is that the theory is more involved. On regular meshes, their behaviour is very similar to that of finite difference method.

We will discuss the finite element method in Chapter 11

9.3.3 Spectral methods

Spectral methods follow the same principle as the finite element method, i.e., represent the solution in a finite dimensional space. Only here, the functions are chosen as eigenfunctions of the differential operator. The resulting methods converge incredibly fast, but their application domain is limited to situations where these eigenfunctions are known analytically.

9.3.4 Finite volume methods

Just like finite differences methods, finite volume methods discretise the solution on a finite grid. Instead of interpreting the solution as the pointwise value of the exact solution, the finite volume method interprets the values on the grid as *cell averages*. This is crucial if the problem has some sort of conserved quantities that can only be transported, not created or destroyed.

With a finite volume method, one can explicitly write that the total quantity of some substance in a cell can only change via in- or outflow through the boundaries. This structure has a physical nature for conservation laws, as they arise, for instance in fluid flow. Hence, finite volume methods are popular in this type of application.

Also many numerical artefacts of finite volume methods can be understood from studying finite difference schemes.

Chapter 10

Finite difference methods

10.1 Two-point boundary value problems

10.1.1 Derivation of method

LeVeque: section 2.4

10.1.2 Boundary conditions

LeVeque: section 2.12

10.1.3 Solution of the linear systems

Point to sparsity.

Explain Thomas algorithm here, instead of in the parabolic part.

10.2 Order of a spatial discretization

LeVeque: section 2.5 and section 2.8.

Local error. Consistency

10.3 Stability and convergence

10.3.1 General observations

LeVeque: section 2.6, section 2.7 and section 2.9

Issue arises because we have vectors or increasing size as $\Delta x \to 0$.

Consistency + stability is convergence. Stability is slightly different than in the ODE case, but the principle is the same.

10.3.2 Stability of finite differences in the L2-norm

LeVeque: section 2.10

10.4 More general equations

LeVeque: section 2.15

With a first derivative appearing.

10.5 Higher-order methods

LeVeque: Section 2.20.1.

10.6 Two-dimensional problems

10.6.1 5-point method for two-dimensional Poisson equation

LeVeque: section 3.2 and section 3.4

10.6.2 Curved boundary discretization

Difficult with finite differences.

10.7 Conclusions and summary

10.7.1 Key concepts

Forward difference, backward difference, central difference.

Local (truncation) error and consistency, order of a method.

Global error, numerical stability, convergence.

10.7.2 Key insights

Finite differences are a convenient way to discretise in space: represent the solution on a grid (a finite set of point values) and replace every derivative in sight with an appropriate finite difference.

The local truncation error can be obtained just like in the ODE case: by filling in the exact solution in the numerical scheme and bounding the remainder term as a function of the mesh size. A finite difference method is *consistent* if the local truncation error tends to zero with vanishing mesh size.

Finite difference methods are convergent if they are consistent and stable. The concept of stability requires some care, since it involves inversion of a matrix of growing size as the mesh size tends to zero.

When studying convergence for finite difference methods, we compare grid functions on a sequence of growing grids. In that situation, there is no norm equivalence. Both the technique to prove convergence and the statements on convergence themselves will depend on the chosen norm.

Chapter 11

The finite element method

In this chapter, we discuss the basics of the finite element method, which is the most commonly used method in publicly available simulation software, and therefore also in scientific and engineering applications. We start by detailing the construction of the finite element method in section 11.1 in the very simple setting of a two-point boundary value problem. Given the importance of the finite element method in practice and its elegant formulation that we are about to discover, one might wonder why so much attention is given to finite differences in this course. As one might expect, part of the answer lies in the observation that finite difference methods allow for a much more direct approach, allowing a quantitative understanding of all important numerical behaviour without excessive technicality in the derivations. (In principle, we have only used Taylor expansions up to this point in the course.) Moreover, many of the numerical properties of finite element methods are similar to those of finite difference methods. We give a more detailed comparison of finite difference and finite element methods in section 11.2, before turning to a more general formulation of the finite element method in higher dimensions in section 11.3. In that section, we also comment on some of the most important considerations to keep in mind when implementing or using a finite element method. We comment on higher order methods in section 11.4 and conclude with some remarks on time-dependent problems in section 11.5.

11.1 Two-point boundary value problems

Consider, for simplicity, the following linear two-point boundary value problem in one space variable:

$$-\frac{d}{dx}\left(a(x)\frac{d}{dx}u(x)\right)+b(x)u(x)=f(x), \qquad 0\le x\le 1, \tag{11.1}$$

in which the function $u(x)$ is the unknown and the (known) functions a, b and f satisfy: $a(x) > 0$ (and differentiable) and $b(x) \geq 0$. Equation (11.1) is supplemented with Dirichlet boundary conditions:

$$u(0) = u_0, \qquad u(1) = u_1, \tag{11.2}$$

with $u_{0,1}$ some specified constants. Such two-point boundary value problems arise in many applications. Let us here simply point out that the solution $u(x)$ of equation (11.1) can be seen to be the steady state solution of the reaction-diffusion PDE:

$$\frac{\partial}{\partial t}u(x,t) = \frac{\partial}{\partial x}\left(a(x)\frac{\partial}{\partial x}u(x,t)\right) - b(x)u(x,t) + f(x), \qquad 0 \leq x \leq 1.$$

In this section, we will, however, not pin ourselves to a particular origin of the problem (11.1), instead viewing it merely as a prototypical equation for which it is relatively straightforward to develop a finite element method.

11.1.1 Approximation in a finite-dimensional subspace

In finite difference methods, the basic discretization principle is that we represent the solution $u(x)$ by its values on a grid (a discrete set of points) and subsequently replace every derivative in sight by a finite difference approximation on this grid. The finite element method starts from a different point of view: *first approximate the solution in a finite-dimensional space.* To this end, we need to introduce a basis that defines the finite-dimensional space. We choose to define a function φ_0 that satisfies the boundary conditions (11.2), and a set of M linearly independent functions $\{\varphi_m\}_{m=1}^M$ that satisfy the homogeneous Dirichlet boundary conditions $\varphi_m(0) = \varphi_m(1) = 0$, $1 \leq m \leq M$. For the time being, we do not specify any further the possible choices for these basis functions. We will come back to this point later (in section 11.1.4).

Given the functions φ_0 and $\{\varphi_m\}_{m=1}^M$, we may write the numerical solution

$$u_M(x) = \varphi_0(x) + \sum_{m=1}^{M} c_m\varphi_m(x), \qquad 0 \leq x \leq 1, \tag{11.3}$$

and the problem reduces to finding the coefficients $\{c_m\}_{m=1}^M$ such that the function $u_M(x)$ approximates, in some sense, the solution of the original problem (11.1).

The function φ_0 is special. Since it is only there to ensure that u_M satisfies the boundary conditions (11.2), there is no unknown coefficient c_0 attached to it. As a consequence, we will define the linear space as

$$\overset{\circ}{\mathbb{H}}_M := \mathrm{Sp}\left\{\varphi_1, \varphi_2 \ldots, \varphi_M\right\},$$

the span of the basis $\{\varphi_m\}_{m=1}^M$(i.e., the set of all linear combinations of these functions). We then search the function

$$u_M - \varphi_0 \in \overset{\circ}{\mathbb{H}}_M,$$

such that u_M is an approximation to the solution of (11.1) in some sense.

11.1.2 Approximation criterion

The next step now is to give a precise meaning to the word 'approximation'. How do we choose the coefficients $\{c_m\}_{m=1}^M$ in equation (11.3) adequately?

11.1.2.1 The collocation method

One idea is to use the collocation method that we have also encountered during the construction of implicit Runge–Kutta methods. One then forces the differential equation (11.1) to be satisfied exactly by the numerical solution u_M in a set of M distinct *collocation points* $\{x_m\}_{m=1}^M$. This results in the following linear system of equations:

$$\begin{aligned} &-\frac{d}{dx}\left(a(x_m)\left\{\frac{d}{dx}\varphi_0(x_m) + \sum_{m'=1}^M c_{m'}\frac{d}{dx}\varphi_{m'}(x_m)\right\}\right) \\ &\quad + b(x_m)\left\{\varphi_0(x_m) + \sum_{m'=1}^M c_{m'}\varphi_{m'}(x_m)\right\} = f(x_m), \qquad 1 \le m \le M, \end{aligned} \tag{11.4}$$

Given that the basis functions φ_m (and hence their derivatives) are known explicitly, the only unknowns in (11.4) are the coefficients $\{c_m\}_{m=1}^M$, which can be readily solved for.

11.1.2.2 Galerkin projection

An alternative approach is the Galerkin projection, which tries to use information on the whole interval $(0,1)$, instead of only a discrete set of points. To this end, we introduce the *residual*

$$r_M(x) := -\frac{d}{dx}\left(a(x)\frac{d}{dx}u_M(x)\right) + b(x)u_M(x) - f(x). \tag{11.5}$$

If u_M, by a incredible stroke of luck, would happen to be the exact solution to the differential equation (11.1), we know that $r_M \equiv 0$. Hence, in general, we can guess that the numerical solution u_M will approximate the exact solution

u well when r_M is close to zero. We will therefore propose to make r_M as small as possible in an appropriate norm.

One way to ensure that the residual is as small as possible is to require that it is *orthogonal* to the space $\overset{\circ}{\mathbb{H}}_M$. To this end, we introduce a scalar product $\langle \cdot, \cdot \rangle$ on the space $\overset{\circ}{\mathbb{H}}_M$ and search for the set of coefficients $\{c_m\}_{m=1}^M$ that ensure that the following orthogonality conditions are satisfied:

$$\langle r_M, \varphi_m \rangle = 0, \qquad 1 \leq m \leq M. \tag{11.6}$$

The equations (11.6) enforce the residual r_M to be orthogonal to each of the basis functions φ_m and, consequently, to the complete space $\overset{\circ}{\mathbb{H}}_M$; they are called the *Galerkin conditions.* A typical choice for the scalar product $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product (the L_2 inner product):

$$\langle \varphi, \phi \rangle = \int_0^1 \varphi(x)\phi(x)dx. \tag{11.7}$$

We will use (11.7), as is often done in practice, but it should be clear that other choices are very well possible and lead to different finite element approximations.

11.1.2.3 Relation to best approximation

In approximation theory in Hilbert spaces, the *best approximation* u_M of a known function u in a finite-dimensional subspace such as $\overset{\circ}{\mathbb{H}}_M$ is obtained by requiring the *error* $e_M = u_M - u$ to be orthogonal to the subspace $\overset{\circ}{\mathbb{H}}_M$, i.e., by requiring the equations

$$\langle e_M, \varphi_m \rangle_H = 0, \qquad 1 \leq m \leq M, \tag{11.8}$$

to be satisfied for some suitably defined scalar product $\langle \cdot, \cdot \rangle_H$ (with associated norm $\| \cdot \|_H$). For the solution of the differential equation (11.1), however, the function u to approximate itself is unknown, so that we cannot solve the equations (11.8)! The Galerkin conditions are only a *surrogate* for the uncomputable 'best approximation'!

Remark 11.1 (On scalar products)**.** Note that we have introduced a second scalar product here, which can (and usually will) be different from the scalar product $\langle \cdot, \cdot \rangle$ that was used in the Galerkin projection (11.6)! We will discuss the relation between these two scalar products below.

There is some theory required to deduce that the Galerkin conditions provide a *good* surrogate for the best approximation. The main theorem on this matter, of which we only mention the name as an element of general culture, is the *Lax-Milgram theorem.* It requires a number of ingredients from functional analysis that we will not be able to introduce in these notes – hence, a thorough theoretical convergence analysis of the finite element method is out of scope here[1]. The missing ingredients mostly concern the introduction of suitable function spaces (so-called *Sobolev spaces*) and the associated scalar products (and norms).

For the two-point boundary value problem (11.1), the Lax-Milgram theorem shows that we really have

$$\left\| u_M - u \right\|_H = \min_{v_M \in \varphi_0 + \overset{\circ}{\mathbb{H}}_M} \left\| v_M - u \right\|_H ,$$

for a suitable choice of the norm $\| \cdot \|_H$, which is called the "energy norm". We will be able to specify the energy norm more precisely in section 11.1.3. Thus, the Galerkin projection indeed gives the best approximation in the finite-dimensional subspace when measuring the error in the energy norm.

However, be careful: while the above statement is true for the two-point boundary value problem (11.1), it is not true in general! For more general two-point boundary value problems (with some suitable assumptions that we – again! – not detail at this point), the Galerkin projection usually *does not* yield the best approximation in any norm. Nevertheless, it gives some sort of 'near-best' solution, characterised by the inequality:

$$\| u_M - u \| \leq C \min_{v_M \in \varphi_0 + \overset{\circ}{\mathbb{H}}_M} \| v_M - u \| , \tag{11.9}$$

with the norm $\| \cdot \|$ the appropriate Sobolev norm[2]. Thus, the Lax-Milgram theorem can rigorously bound the 'non-optimality' of the approximation. Also, equation (11.9) ensures convergence when enlarging the approximation space $\overset{\circ}{\mathbb{H}}_M$ (i.e., letting the number of basis functions M tend to infinity)[3].

11.1.3 Weak solutions

When filling in the finite-dimensional approximation (11.3) in the definition of the residual (11.5), we get

$$r_M = - \left[(a\varphi_0')' + \sum_{m=1}^{M} c_m \left(a\varphi_m' \right)' \right] + b \left[\varphi_0 + \sum_{m=1}^{M} c_m \varphi_m \right] - f,$$

[1]Whether this is fortunate or unfortunate probably depends on your point of view. For engineering applications, it is probably sufficient to realize that such an analysis exists. For the development of new finite element methods for particular problems, it might be worthwhile to delve a bit deeper into this theory in a follow-up course.

[2]The energy norm only exists for problems such as (11.1), which are *self-adjoint* – another term that we employ here without further explanation.

[3]This is the final point at which we are lacking some functional analysis background. To make this statement rigorous, we need to ensure that the linear space $\overset{\circ}{\mathbb{H}}_M$ is *complete* when M tends to infinity, such that every possible solution of (11.1) can be represented by an element in it. But since we have not even described in which function space solutions of the boundary value problem (11.1) lie, it would clearly be multiple bridges too far to go into these issues here.

in which we have used the notation $d/dx\,(\cdot) \equiv (\cdot)'$ for conciseness. Then, we can write the orthogonality conditions (11.6) as

$$\sum_{m'=1}^{M} c_{m'} \left[\left\langle -\left(a\varphi'_{m'}\right)', \varphi_m \right\rangle + \langle b\varphi_{m'}, \varphi_m \rangle \right] \\ = \langle f, \varphi_m \rangle - \left[\left\langle -\left(a\varphi'_0\right)', \varphi_m \right\rangle + \langle b\varphi_0, \varphi_m \rangle \right], \qquad 1 \le m \le M. \quad (11.10)$$

A next step in the derivation of the finite element method is to use partial integration to get rid of the second order derivatives of the basis functions $\{\varphi_m\}_{m=0}^{M}$. There are a number of reasons to want to do this:

1. Generally speaking, when fewer derivatives of the basis functions φ_m appear in the equations to solve, more options remain open to choose from.
2. In particular, it will turn out that some of the most commonly used and most practical choices indeed have reasonably poor smoothness properties. Most common choices are only piecewise differentiable in $[0, 1]$.
3. To be very precise, since the value of an integral is independent of the values of the integrand on a finite set of points, even piecewise linear basis functions are suitable, as soon as only first derivatives appear in the integrand.

Remark 11.2 (Weak solutions)**.** Note that, by allowing numerical solutions (11.3) that are only piecewise differentiable and not twice differentiable at all, the computed numerical solution cannot be a solution to the two-point boundary value problem 11.1 in the strict sense. We call solutions in the strict sense *strong solutions*, and – by contrast – we call the solutions that we compute using Galerkin projection *weak solutions*. We will come back to this point in section 11.1.5.

We thus want to get rid of the second derivatives in (11.10), and we will achieve this (as announced) via partial integration. To perform this partial integration, we will need to be specific about the scalar product we use. Up to now, we had only suggested the L_2 inner product (11.7) as one of the options. At this point, we will go further: we *will* from now on choose (11.7) as our inner product! With this choice, equation (11.10) becomes

$$\sum_{m'=1}^{M} c_{m'} \left[-\int_0^1 \left(a(x)\varphi'_{m'}(x)\right)' \varphi_m(x)dx + \int_0^1 b(x)\varphi_{m'}(x)\varphi_m(x)dx \right] \\ = \int_0^1 f(x)\varphi_m(x)dx - \left[-\int_0^1 \left(a(x)\varphi'_0(x)\right)' \varphi_m(x)dx + \int_0^1 b(x)\varphi_0(x)\varphi_m(x)dx \right], \\ 1 \le m \le M. \quad (11.11)$$

Now we perform partial integration, using the fact (by construction) that the functions φ_m vanish at $x = 0$ and $x = 1$:

$$\begin{aligned} -\int_0^1 \left(a(x)\varphi'_{m'}(x)\right)' \varphi_m(x)dx &= -\, a(x)\varphi'_{m'}(x)\varphi_m(x)\big|_0^1 + \int_0^1 a(x)\varphi'_{m'}(x)\varphi'_m(x)dx \\ &= \int_0^1 a(x)\varphi'_{m'}(x)\varphi'_m(x)dx. \end{aligned}$$

When introducing the notation

$$a_{m,m'} = \int_0^1 a(x)\varphi'_{m'}(x)\varphi'_m(x) + b(x)\varphi_{m'}(x)\varphi_m(x)dx, \tag{11.12}$$

and

$$f_m = \int_0^1 f(x)\varphi_m(x)dx - a_{m,0} \tag{11.13}$$

it is easy to see that (11.11) reduces to

$$\sum_{m'=1}^{M} a_{m,m'}c_{m'} = f_m, \qquad 1 \le m \le M, \tag{11.14}$$

which we write even shorter in matrix notation as

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{f}, \tag{11.15}$$

in which we have introduced

$$\boldsymbol{A} = \left(a_{m,m'}\right)_{m,m'=1}^{M}, \qquad \boldsymbol{c} = \left(c_m\right)_{m=1}^{M}, \qquad \boldsymbol{f} = \left(f_m\right)_{m=1}^{M}.$$

Remark that equation (11.12) introduces a specific inner product

$$\langle u, v\rangle_H = \int_0^1 a(x)u'(x)v'(x) + b(x)u(x)v(x)dx, \tag{11.16}$$

from which the energy norm $\|u\|_H = \sqrt{\langle u, u\rangle_H}$ can be derived.

Since we have now minimised the differentiability requirements on $\{\varphi_m\}_{m=1}^{M}$, and (as expected) all quantities $a_{m,m'}$ can be computed once the basis functions are chosen, equation (11.14) is the form of the orthogonality conditions that will find its way into to finite element software.

11.1.4 Local basis functions

11.1.4.1 The principle of local basis functions

To compute the finite element approximation (11.3), any finite element software needs to perform two steps:

- *construct* the linear system (11.15), i.e., compute the matrix elements $a_{m,m'}$, $1 \leq m, m' \leq M$ and the right hand sides f_m, $1 \leq m \leq M$;
- *solve* the resulting linear system.

It should be clear that there is an interest in choosing the basis functions such that many matrix elements $a_{m,m'}$ become zero. One natural thought after a course on numerical approximation theory is then to choose the $\{\varphi_m\}_{m=1}^M$ such that they form an 'orthogonal basis' for the space $\overset{\circ}{\mathbb{H}}_M$ with respect to the scalar product (11.16). However, this is usually not done in finite element software, since it is impractical: requiring orthogonality with respect to (11.16) would require the computation of the orthogonal basis *for every equation that one wants to solve*, since this inner product depends on $a(x)$ and $b(x)$[4].

Instead, we will limit the number of non-zero matrix elements $a_{m,m'}$ by choosing the basis functions $\{\varphi_m\}_{m=1^M}$ to have only local support (choose them to be only non-zero locally), such that only for a limited number of values m and m' the product $\varphi_m\varphi_{m'}$ are non-zero. In this discussion, we will only consider *piecewise linear* basis functions on an equidistant mesh. We start by defining the mesh size $h = 1/(M+1)$ and the mesh points $\{x_m\}_{m=1}^M$, with $x_m = mh$. We attach a basis function to each mesh point x_m as follows:

$$\varphi_m(x) = \begin{cases} \left(x - (x_m - h)\right)/h, & x \in [x_{m-1}, x_m] \\ \left(-x + (x_m + h)\right)/h, & x \in [x_m, x_{m+1}] \\ 0, & x \notin [x_{m-1}, x_{m+1}] \end{cases} \tag{11.17}$$

(Note that, because φ_m is continuous on $[0,1]$, it does not matter that we have two function prescriptions at the mesh points $x_{m\pm1}$.) Figure 11.1 shows these basis functions for the case with $M = 4$, which corresponds to $h = 0.2$. We see that these mesh points divide the interval $[0,1]$ into $M+1$ *finite elements* (indicated by the dotted lines). We can make a number of observations:

- In each of the internal elements, only two basis functions are non-zero; due to the boundary conditions only one basis function is non-zero in the first and last element.
- In each element, the solution can be any linear function.
- The solution is continuous, but not differentiable, across elements.
- The support of the basis function φ_m only overlaps with the support of φ_{m-1} and φ_{m+1}.

As a consequence of the last observation, we see that the matrix $\boldsymbol{A}$ in equation (11.15) becomes tridiagonal for this choice of basis functions. Thus, only

[4]Additionally, for more general problems, the partial integration that leads to (11.12) does not necessarily imply the definition of a scalar product. This is related to the fact that the energy norm only exists when the problem is self-adjoint, see a previous footnote. When (11.12) does not induce a scalar product, constructing an orthogonal basis is not only impractical, it is even impossible.

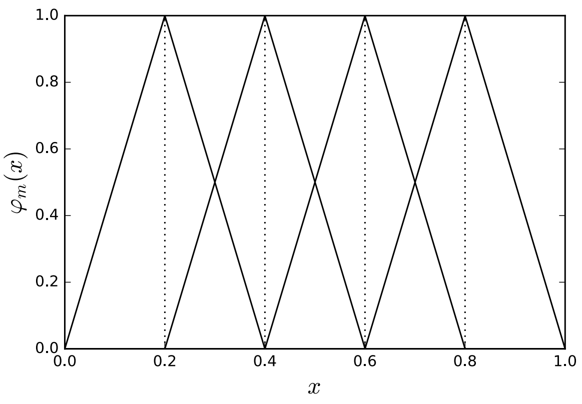

Figure 11.1: Basis functions $\varphi_m(x)$, $m = 1, \ldots, 4$ (solid lines), as well as an indication of the element boundaries (dotted lines).

$O(M)$ elements $a_{m,m'}$ need to be computed (instead of $O(M^2)$) and the linear system can readily be solved by the Thomas algorithm that we have encountered in an earlier chapter.

11.1.4.2 Explicit computation of matrix elements

Let us now explicitly compute the system matrix $\boldsymbol{A}$ for the two-point boundary value problem (11.1). For simplicity of the computations below, we choose $b(x) \equiv b$, independent of x, and homogeneous Dirichlet boundary conditions $u(0) = u(1) = 0$. We start with the computation of $a_{m,m-1}$. From (11.12), we see that

$$\begin{aligned} a_{m,m-1} &= \int_0^1 a(x)\varphi'_{m-1}(x)\varphi'_m(x) + b\varphi_{m-1}(x)\varphi_m(x)dx \\ &= \int_{x_{m-1}}^{x_m} a(x)\varphi'_{m-1}(x)\varphi'_m(x) + b\varphi_{m-1}(x)\varphi_m(x)dx, \end{aligned}$$

in which we can restrict the domain of integration to the interval $[x_{m-1}, x_m]$ because this is the only element in which both $\varphi_{m-1}(x)$ and $\varphi_m(x)$ are non-zero.

We can compute these integrals much more conveniently by introducing the *local coordinate* ξ, which is chosen such that $x = x_{m-1} + h\xi$. Using this coordinate, we obtain (locally in the element $[x_{m-1}, x_m]$)

$$\varphi_{m-1}(x) = \varphi_{m-1}(x_{m-1} + h\xi) = 1 - \xi, \qquad \varphi_m(x) = \varphi_m(x_{m-1} + h\xi) = \xi. \tag{11.18}$$

and

$$\varphi'_{m-1}(x) = \frac{d}{d\xi}\varphi_{m-1}(x_{m-1}+h\xi)\cdot\frac{d\xi}{dx} = -\frac{1}{h}, \qquad \varphi'_m(x) = \frac{1}{h} \tag{11.19}$$

We then have, using the above formulas and $dx = hd\xi$,

$$\begin{aligned} a_{m,m-1} &= -\frac{1}{h^2}\int_0^1 a(x_{m-1}+h\xi)hd\xi + \int_0^1 b(1-\xi)\xi hd\xi \\ &= -\frac{1}{h}A_{m-1} + \frac{1}{6}bh, \end{aligned}$$

in which we have introduced

$$A_{m-1} = \int_0^1 a(x_{m-1}+h\xi)d\xi.$$

which still needs to be computed. For a general function $a(x)$, this can typically not be done analytically. In practice, one therefore often uses a quadrature rule, for instance the trapezoidal rule:

$$A_{m-1} = \frac{1}{2}\left(a(x_{m-1}) + a(x_m)\right).$$

Note that, given the piecewise linear basis functions, a higher order quadrature rule at this point does not really make much sense.

With a very similar reasoning, we obtain

$$a_{m,m+1} = -\frac{1}{h}A_m + \frac{1}{6}bh.$$

The computation of $a_{m,m}$ follows the same principle, albeit that the integrand is now non-zero on *two* elements: $[x_{m-1}, x_m]$ and $[x_m, x_{m+1}]$. We thus obtain:

$$\begin{aligned} a_{m,m} &= \int_0^1 a(x)\varphi'_m(x)\varphi'_m(x) + b\varphi_m(x)\varphi_m(x)dx \\ &= \int_{x_{m-1}}^{x_m} a(x)\varphi'_m(x)\varphi'_m(x) + b\varphi_m(x)\varphi_m(x)dx \\ &\quad + \int_{x_m}^{x_{m+1}} a(x)\varphi'_m(x)\varphi'_m(x) + b\varphi_m(x)\varphi_m(x)dx, \end{aligned}$$

in which we have split the integral into the contributions stemming from the individual elements. We now perform the local coordinate transform in each element individually to obtain (using (11.18) and (11.19)):

$$\begin{aligned} a_{m,m} &= \frac{1}{h^2}\int_0^1 a(x_{m-1}+h\xi)hd\xi + \int_0^1 b\xi^2 hd\xi \\ &\quad + \frac{1}{h^2}\int_0^1 a(x_m+h\xi)hd\xi + b\int_0^1 (1-\xi)^2 hd\xi, \\ &= \frac{1}{h}\left(A_{m-1}+A_m\right) + 2\cdot\frac{1}{3}bh \end{aligned}$$

Finally, we are left with the computation of the elements f_m, defined in (11.13). We thus start from

$$f_m = \int_0^1 f(x)\varphi_m(x)dx - a_{m,0},$$

and notice that f_m depends on the basis function φ_0 through the quantity $a_{m,0}$. This is therefore the place where we need to worry about the boundary conditions, because these will affect the choice for φ_0. However, given that we have imposed homogeneous Dirichlet boundary conditions, we can safely choose $\varphi_0 \equiv 0$, such that $a_{m,0} = 0$. We thus continue as follows (again using (11.18) and (11.19)):

$$\begin{aligned} f_m &= \int_0^1 f(x)\varphi_m(x)dx \\ &= \int_{x_{m-1}}^{x_m} f(x)\varphi_m(x)dx + \int_{x_m}^{x_{m+1}} f(x)\varphi_m(x)dx \\ &= \int_0^1 f(x_{m-1}+h\xi)\xi h d\xi + \int_0^1 f(x_m+h\xi)(1-\xi)hd\xi. \end{aligned}$$

We again observe that the integrals cannot be computed analytically. If we also use the trapezoidal rule here, we get:

$$f_m = \frac{h}{2}\left(f(x_{m-1})\cdot 0 + f(x_m)\cdot 1\right) + \frac{h}{2}\left(f(x_m)\cdot 1 + f(x_m)\cdot 0\right) = hf(x_m).$$

Thus, after dividing away a factor h that is common to all the terms, equation (11.14) reduces to:

$$\begin{aligned} \left(-\frac{1}{h^2}A_{m-1} + \frac{1}{6}b\right)c_{m-1} + \left(\frac{1}{h^2}\left(A_{m-1}+A_m\right) + \frac{2}{3}b\right)c_m \\ + \left(-\frac{1}{h^2}A_m + \frac{1}{6}b\right)c_{m+1} = f(x_m). \end{aligned} \tag{11.20}$$

Let us stand still for a moment on the two crucial tricks that made the above computation relatively simple to perform:

- We have split the integral over the domain into integrals over individual elements.
- We introduced a local coordinate ξ inside each element to perform the individual integrations.

Any finite element code, regardless of the complexity of the problem, will use these tricks to efficiently construct the matrices $\boldsymbol{A}$ and $\boldsymbol{f}$. The fact that this is always possible is one of the powers of the finite element method. It, for instance, allows to readily introduce adaptive grids. (The only change then is that the value of h changes from element to element, but the above procedure carries out without problems.)

Remark 11.3 (Interpretation of coefficients)**.** Due to the choice of the basis functions, we see that c_m is an approximation to the solution $u(x_m)$, which allows us to visualise the solution without the need to evaluate (11.3) in the grid points.
Remark 11.4 (Relation to finite difference methods)**.** The resulting discretization (11.20) is very similar to a finite difference discretization of the same problem. One can recognize the diffusive part and the right hand side. A peculiarity is the fact that the local term $bu(x)$ is spread out over three terms. Note, however, that for the one-dimensional Poisson equation $u'' = f$, equation (11.20) reduces to

$$-\frac{1}{h^2}c_{m'-1} + \frac{2}{h^2}c_{m'} - \frac{1}{h^2}c_{m'+1} = f(x_m),$$

which is *identical* to the finite difference approximation.
Remark 11.5 (Contrast with spectral methods)**.** Since the choice of basis functions is up to the user, one might consider other criteria than locality to be important. One reasoning would be to consider global basis functions that are so representative of the solution of the boundary value problem that only a limited number of them are needed to adequately represent the solution. One then needs to compare the computational cost of solving a relatively small (but full) linear system with that of a sparse (but potentially very large) linear system. Given that finite element methods with piecewise linear basis functions result in a space discretization error of $O(h^2)$, this might be a potentially interesting avenue to follow. (We have not shown – or even discussed – the order of finite element methods here, but a quick look at Remark 11.4 might convince you of the $O(h^2)$ convergence.) That avenue leads to so-called *spectral methods*, and we will not discuss such methods in these notes.

11.1.5 Variational formulation

In the previous sections, we have developed the finite element method for a very simple two-point boundary value problem. Before we will generalize the method to more complex situations and really discuss its virtues over the finite difference method, we will make a detour that derives the Galerkin equations (11.6) in an alternative way.

Let us consider a *variational problem*, in which we are given a functional $\mathcal{J}(u) : \mathbb{H} \to \mathbb{R}$, in which $\mathbb{H}$ is some function space, that we – again! – do not specify further here, except to note that $\mathbb{H}$ only contains functions that satisfy the boundary conditions (11.2). We are now interested in finding a function $u \in \mathbb{H}$ that and minimizes $\mathcal{J}$:

$$u = \arg\min_{v\in\mathbb{H}} \mathcal{J}(v).$$

In particular, we will consider here the functional

$$\mathcal{J}(v) := \int_0^1 \Big[a(x)\,(v'(x))^2 + b(x)\,(v(x))^2 - 2f(x)v(x)\Big]\,dx,$$

with the functions $a(x)$, $b(x)$ and $f(x)$ (not coincidentally) satisfying the same conditions as those in the boundary value problem (11.1). It is possible to prove that the minimizer u exists, such that the variational problem always has a solution. As can be expected by now, we will not do this here.

We also introduce the space $\overset{\circ}{\mathbb{H}}$ of all functions v that obey homogeneous Dirichlet boundary conditions. Then, given that $u \in \mathbb{H}$ minimizes $\mathcal{J}$, we know that we have

$$\mathcal{J}(u) \le \mathcal{J}(u+v), \qquad v \in \overset{\circ}{\mathbb{H}}. \tag{11.21}$$

Let us choose $v \neq 0$ and $\epsilon \neq 0$. We can then write

$$\begin{aligned}
\mathcal{J}(u+\epsilon v) &= \int_0^1 \Big[a\,(u'+\epsilon v')^2 + b\,(u+\epsilon v)^2 - 2f(u+\epsilon v)\Big]\,dx \\
&= \int_0^1 \Big[a\Big((u')^2 + 2\epsilon u'v' + \epsilon^2\,(v')^2\Big) + b\,\big(u^2 + 2\epsilon uv + \epsilon^2 v^2\big) - 2f\,(u+\epsilon v)\Big]\,dx \\
&= \int_0^1 \Big[a\,(u')^2 + bu^2 - 2fu\Big]\,dx + 2\epsilon\int_0^1 \left[au'v' + buv - fv\right]dx \\
&\qquad\qquad + \epsilon^2\int_0^1 \Big[a\,(v')^2 + bv^2\Big]\,dx \\
&= \mathcal{J}(u) + 2\epsilon\int_0^1 \left[au'v' + buv - fv\right]dx + \epsilon^2\int_0^1 \Big[a\,(v')^2 + bv^2\Big]\,dx.
\end{aligned}$$

Using (11.21), we see that

$$2\epsilon\int_0^1 \left[au'v' + buv - fv\right]dx + \epsilon^2\int_0^1 \Big[a\,(v')^2 + bv^2\Big]\,dx \ge 0,$$

for all non-zero values of ϵ. As $|\epsilon|$ can be made arbitrarily small, we can safely neglect the second order term. Moreover, this condition is valid for *any* ϵ, and we are allowed to replace ϵ by $-\epsilon$ without violating the condition. Thus, we deduce that, necessarily,

$$\int_0^1 \left[a(x)u'(x)v'(x) + b(x)u(x)v(x) - f(x)v(x)\right]dx = 0, \forall v \in \overset{\circ}{\mathbb{H}}. \tag{11.22}$$

While we have only shown that (11.22) is necessary, it can also be proved to be sufficient. (We will, of course, not do this here...)

The above formulation can be seen to be equivalent to the *weak form* of the two-point boundary value problem (11.1), which is obtained by requiring the residual of the solution u of (11.1) to be orthogonal to the whole space $\overset{\circ}{\mathbb{H}}$:

$$\int_0^1 \Big[-\left(a(x)u'(x)\right)' + b(x)u(x) - f(x) \Big] v(x)dx.$$

In principle, this is the same as requiring that the residual is identically zero, since it needs to be orthogonal to the whole space. However, there is a point in doing this, as we can perform partial integration (as we have done in section 11.1.3) to reduce the smoothness requirements on u. This can be seen to result in (11.22).

One can therefore also recover the Galerkin conditions by searching for the minimizer u_M in the finite dimensional subspace $\mathbb{H}_M$ (see equation (11.3)):

$$u = \arg \min_{v \in \mathbb{H}_M} \mathcal{J}(v).$$

Since v is now a finite-dimensional object, we can proceed in the classical way to formulate optimality conditions. We write

$$\mathcal{J}_M(\boldsymbol{c}) := \mathcal{J}\left(\varphi_0 + \sum_{m=1}^{M} c_m \varphi_m \right), \qquad \boldsymbol{c} \in \mathbb{R}^M,$$

and compute the coefficients $\boldsymbol{c}$ by requiring $\nabla_{\boldsymbol{c}} \mathcal{J}_M(\boldsymbol{c}) = 0$. Writing down these equations, one recovers exactly the Galerkin conditions (11.14). Indeed, we have

$$\begin{aligned}\frac{1}{2}\frac{\partial \mathcal{J}_M}{\partial c_m} = \sum_{m'=1}^{M} c_{m'} \int_0^1 \left(a\varphi'_{m'}\varphi'_m + b\varphi_{m'}\varphi_m \right) dx \\ + \int_0^1 \left(a\varphi'_0\varphi'_m + b\varphi_0\varphi_m \right) dx - \int_0^1 f\varphi_m dx, \qquad 1 \leq m \leq M,\end{aligned}$$

which results in exactly the Galerkin conditions (11.14).

11.2 Comparison with finite difference methods

11.2.1 Why mainly finite differences in this course?

Now that the finite difference and the finite element method have both been introduced, it is clear that the finite difference method has a number of clear advantages over the finite element method, mainly in terms of simplicity:

- Discretizing all derivatives in sight on a grid is relatively straightforward to do, even without a thorough mathematical background;

- Also performing a numerical analysis of the properties (consistency, stability) of the resulting schemes is straightforward, and a Fourier analysis of the resulting schemes does not require advanced mathematical tools.

Clearly, the mathematical setup of the finite element method is much more involved. For this reason, most of the theory on numerical methods for PDEs in this course concentrated on finite difference schemes. For the finite element method, we restricted ourselves to a detailed description of its construction, merely hinting at some of the subtleties that should be accounted for in a proper convergence analysis. Nevertheless, the finite element method is much more commonly used in practice. For this to make sense, there must be an advantage that comes with this increased complexity!

11.2.2 Practical advantages of finite element methods

There are a number of practical advantages that follow immediately from the element-wise formulation of the finite element method. We announce those advantages here:

- *Flexibility.* The finite element method can easily handle complex geometries and boundary conditions, whereas the finite difference method is usually restricted to (a variation of) a rectangular domain. The finite element method will turn out to discretize the domain into triangles, which allow more flexibility than the rectangles that are typically required in finite difference schemes. Usually, a separate software program to triangularize the computational domain can be used to generate a suitable mesh for a given simulation.

- *Adaptivity.* Additionally, the flexibility of the element shapes and sizes allows for a straightforward *local mesh refinement* in regions of the spatial domain in which the error is large (or can be expected to be large *a priori*. Also the introduction of higher-order methods is relatively straightforward by simply considering local polynomial approximations of higher order.

- *Generality.* The finite element method is very general, and allows one to easily simulate (in one simulation) coupled multiphysics problems (for instance, fluid-structure interaction). Here, different PDEs represent the physics in different parts of the spatial domain – the coupling is done with appropriate boundary conditions. The relative ease with which boundary conditions are constructed is a significant factor in solving such complex problems.

In section 11.3, we will formulate the finite element method in a more general (multi-dimensional) situation. There, we will illustrate these advantages in more detail.

11.2.3 Theoretical advantages of finite element methods

It would lead us too far to discuss at full length the theory that exists for finite element method. In view of the very brief discussion above, we only state the following theoretical advantages of the finite element method, that all stem from the principle of approximation in a finite-dimensional space:

- There exists a large literature on a priori error estimation and convergence analysis, which can be used during discretization and mesh generation.
- The numerical approximation is guaranteed to be reasonable in every point of the spatial domain (also in between the mesh points).
- There is also a large literature on *a posteriori* error estimation, i.e., obtaining upper bounds on the error once the solution is computed. These results can be used to adaptively refine the grid where needed.

11.3 Finite element methods in higher dimensions

Now that we have discussed the formulation and principles of the finite element method for the specific case of a one-dimensional two-point boundary value problem, we are ready to generalize. Let us, in these notes, restrict ourselves to two space dimensions, such that we search for the solution $u(x,y)$ of the following PDE:

$$\mathcal{L}u = f, \qquad (x,y) \in \Omega, \tag{11.23}$$

with some suitable boundary conditions that we will specify later. The direct two-dimensional generalization of the two-point boundary value problem (11.1) is obtained by choosing

$$\mathcal{L} := -\nabla\left(a(x,y)\nabla\right) + b(x,y),$$

in which $a(x,y) > 0$ and differentiable, and $b(x,y) > 0$.

11.3.1 Finite-dimensional space and local basis functions

In two space dimensions, we look for a piecewise linear approximation on a *set of two-dimensional subdomains* that cover the computational domain. In each subdomain, the solution is then represented by a plane in the three-dimensional space (x,y,u). Since a plane is completely specified by three points, the subdomains will need to be triangles (as we need to specify the solution in exactly three points per subdomain to have a unique representation). (Hence, we will not use rectangles for now, as is typically done with finite difference schemes.

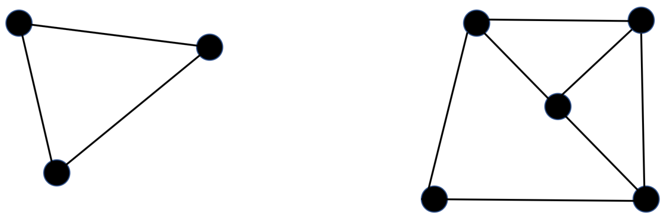

Figure 11.2: A triangle and a non-conforming triangle.

Some remarks on the use of rectangles are given at the end of this section.) To ensure the continuity of the solution in the whole domain, solutions need to be continuous on triangle edges. As a consequence, all vertices need to be shared among triangles; no vertex of a triangle can lie on an edge of a different triangle, see Figure 11.2.

When partitioning the domain Ω into a set of J triangles $\{\Omega_j\}_{j=1}^{J}$, this immediately implies that the boundary of the domain $\partial\Omega$ will consist of a finite number of straight segments. Thus, the equation will be solved on *an approximation of the domain* and with boundary conditions that are imposed at *an approximation of the boundary* $\partial\Omega$.

For one-dimensional problems, we defined the basis functions as the piecewise linear functions that vanish at all mesh points except for one. In two dimensions, this corresponds to the so-called *chapeau functions*[5]. A chapeau function is one in exactly one vertex, zero in all other vertices and has support only in the elements surrounding the vertex in which it takes the value one.

As a consequence (and unfortunately), making explicit use of these chapeau functions is very impractical, because many different configurations are possible, and the number of elements in each support (and hence the shape of the basis function) may change from vertex to vertex, see Figure 11.3.

At the same time, we know from the derivations in section (11.1.4) that it will be convenient to compute the integrals that appear in the matrix $\boldsymbol{A}$ (see equation (11.15)) on an element-by-element basis. Therefore, all we need is a representation of the solution *inside each element* Ω_j. Let us denote by $\left\{(x_j^\ell, y_j^\ell)\right\}_{\ell=1}^{3}$ the three vertices of triangle Ω_j. Since, inside Ω_j, the approximate solution is linear, we can write it as

$$u_j(x,y) = \alpha_j + \beta_j x + \gamma_j y, \qquad (x,y) \in \Omega_j,$$

[5]By using the French word “chapeau” instead of the English word “hat”, the intention is to indicate that the chapeau functions are constructed using the same principle as the hat functions in the one-dimensional case, but are nonetheless slightly more complicated – or sophisticated, just like the French language apparently is.

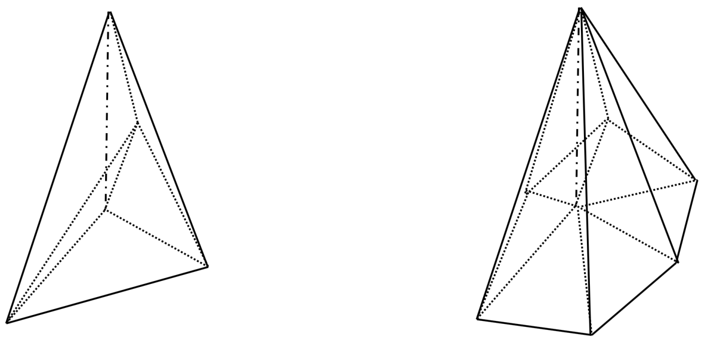
Figure 11.3: Different configurations of chapeau functions in two space dimensions.

in which the values of α_j, β_j and γ_j can be found by solving the linear system:

$$\alpha_j + \beta_j x_j^\ell + \gamma_j y_j^\ell = u_j^\ell, \qquad \ell = 1, 2, 3 \tag{11.24}$$

with $\{u_j^\ell\}_{\ell=1}^3$ the solution values in the vertices. Both the values α_j, β_j and γ_j and the three solution values $\{u_j^\ell\}_{\ell=1}^3$ specify completely the approximate solution in the elements. In particular, the values $\{u_j^\ell\}_{\ell=1}^3$ are not known yet: they are exactly what we are trying to compute! The linear system (11.24) shows the equivalence between the solution values in the vertices $\{u_j^\ell\}_{\ell=1}^3$ and the parameters α_j, β_j and γ_j.

Remark 11.6 (Quadrilateral elements)**.** Sometimes, one prefers quadrilateral elements (rectangles aligned with the axes). One can then interpolate using functions of the form:

$$u_j(x,y) = u_j^1(x)u_j^2(y), \qquad \text{with} \qquad u_j^1(x) = \alpha_j + \beta_j x, \quad u_j^2(y) = \gamma_j + \delta_j y.$$

Clearly, we have now four parameters, which is perfectly suited to interpolation at the four corners of a rectangle. Along both horizontal edges, $u_j^2(y)$ is constant. Since $u_j^1(x)$ is completely specified by the values at the corners, the function $u_j(x,y)$ is independent of the interpolated values elsewhere on the mesh. A similar argument holds for the vertical edges. This ensures global continuity of the resulting approximation on the whole domain.

11.3.2 Weak formulation in higher dimensions

Now that we know how we will represent the numerical solution inside each element, we need to write down the equations that need to be solved to solve the PDE (11.23). As in the one-dimensional case, we will perform a Galerkin projection (requiring orthogonality of the residual to the approximation space, see equation (11.6)) and perform "partial integration" to get rid of the second order derivative.

In the two-dimensional case, we will need to use various multivariate counterparts of integration by parts. To recall a few examples of these, we denote by $\boldsymbol{x} = (x, y)$ a point in the domain Ω, by $\boldsymbol{s}$ a point on the boundary $\partial\Omega$ and by $\partial v(\boldsymbol{s})/\partial n$ the directional derivative of v on the boundary $\partial\Omega$ in the direction of the outward normal to the boundary. One particular case is the *divergence theorem*:

$$\int_\Omega \nabla\cdot[a(\boldsymbol{x})\nabla v(\boldsymbol{x})]\, w(\boldsymbol{x})d\boldsymbol{x} = \int_{\partial\Omega} a(\boldsymbol{s})w(\boldsymbol{s})\frac{\partial v(\boldsymbol{s})}{\partial n}d\boldsymbol{s} - \int_\Omega a(\boldsymbol{x})\left[\nabla v(\boldsymbol{x})\right]\left[\nabla w(\boldsymbol{x})\right] d\boldsymbol{x}.$$

Another example is *Green's formula*:

$$\int_\Omega \left[\nabla^2 v(\boldsymbol{x})\right] w(\boldsymbol{x})d\boldsymbol{x} + \int_\Omega \left[\nabla v(\boldsymbol{x})\right]\left[\nabla w(\boldsymbol{x})\right] d\boldsymbol{x} = \int_{\partial\Omega} \frac{\partial v(\boldsymbol{s})}{\partial n} w(\boldsymbol{s})d\boldsymbol{s}. \tag{11.25}$$

Both formulas are special cases of *Stokes's theorem*, which we will not cover in detail in these notes.

Let us make use of the above formulas to derive the weak formulation of the two-dimensional *Poisson equation*:

$$-\nabla^2 u = f \qquad \text{or} \qquad -\left(\frac{\partial^2}{\partial x^2}u(x,y) + \frac{\partial^2}{\partial y^2}u(x,y)\right) = f(x,y),$$

for $(x, y) \in \Omega$ with homogeneous Dirichlet boundary conditions. Writing this equation in weak form and using (11.25), we get

$$\begin{aligned} -\int_\Omega \nabla^2 u\; v d\boldsymbol{x} &= \int_\Omega \nabla u \cdot \nabla v d\boldsymbol{x} - \int_{\partial\Omega} v(\boldsymbol{s})\frac{\partial u(\boldsymbol{s})}{\partial n}d\boldsymbol{s} \\ &= \int_\Omega \nabla u \cdot \nabla v d\boldsymbol{x} = \int_\Omega f v d\boldsymbol{x}, \end{aligned} \tag{11.26}$$

since we again only allow test functions v that vanish on the boundary $\partial\Omega$.

The finite element formulation then amounts to a finite-dimensional version of (11.26), i.e., we formally write

$$u_M(\boldsymbol{x}) = \varphi_0(\boldsymbol{x}) + \sum_{m=1}^{M} c_m\varphi_m(\boldsymbol{x}), \tag{11.27}$$

in which the functions φ_m are the chapeau functions mentioned earlier.
Remark 11.7 (On the chapeau functions)**.** As announced in section 11.3.1 will never make direct use of these basis functions. Instead, we will – as in the one-dimensional case – always make use of local representations inside each element. That is the reason we say we "formally write". How the direct use of the chapeau functions is avoided, will become clear in section 11.3.3.

With the formal representation (11.27), we obtain the following equations,

$$\int_\Omega \nabla u_M \cdot \nabla\varphi_m d\boldsymbol{x} = \int_\Omega f\varphi_m d\boldsymbol{x}, \qquad 1 \le m \le M,$$

which reduce to the linear system

$$\sum_{m'=1}^{M} c_{m'} \int_\Omega \nabla\varphi_{m'} \cdot \nabla\varphi_m d\boldsymbol{x} = \int_\Omega f\varphi_m d\boldsymbol{x} - \int_\Omega \nabla\varphi_0 \cdot \nabla\varphi_m d\boldsymbol{x}, \qquad 1 \le m \le M,$$

which, again, results in the linear system

$$\sum_{m'=1}^{M} a_{m,m'} c_{m'} = f_m, \qquad 1 \le m \le M,$$

with

$$a_{m,m'} = \int_\Omega \nabla\varphi_{m'} \cdot \nabla\varphi_m d\boldsymbol{x}, \qquad f_m = \int_\Omega f\varphi_m d\boldsymbol{x} - \int_\Omega \nabla\varphi_0 \cdot \nabla\varphi_m d\boldsymbol{x}. \tag{11.28}$$

Compare the above equations with (11.14) for the one-dimensional case.

11.3.3 Assembling the stiffness matrix

While the above equations are written in terms of the chapeau functions φ_m, we have already indicated in section 11.3.1 that these functions are not ideal for the computation of the matrix elements $a_{m,m'}$ because their shape depends significantly on the number of triangles in which a given vertex participates. To circumvent this problem, we again employ the two tricks that were also used in the one-dimensional case in section 11.1.4:

- We split the integral into the contributions per element;
- We introduce a local coordinate system in each element.

We will limit ourselves to an outline of the general principles.

Let us first elaborate the computation of the coefficients

$$\begin{aligned} a_{m,m'} &= \int_\Omega \nabla\varphi_{m'} \cdot \nabla\varphi_m d\boldsymbol{x} = \sum_{j=1}^{J} \int_{\Omega_j} \nabla\varphi_{m'} \cdot \nabla\varphi_m d\boldsymbol{x} \\ &= \sum_{j \in \mathcal{J}_{m,m'}} \int_{\Omega_j} \nabla\varphi_{m'} \cdot \nabla\varphi_m d\boldsymbol{x}, \end{aligned}$$

in which the set $\mathcal{J}_{m,m'}$ consists of the indices of all elements Ω_j in which both the basis functions $\varphi_{m'}$ and φ_m are non-zero. We have now split the integral into the element-wise contributions and restricted ourselves to the elements in which these contributions are non-zero.

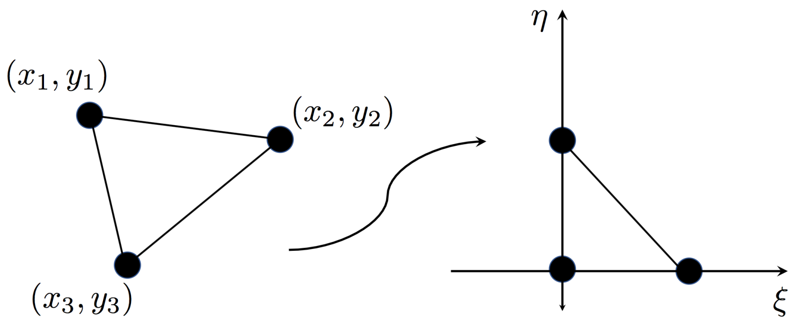

Figure 11.4: Local coordinate transform in 2D.

The second step is the introduction of a local coordinate system $\boldsymbol{\xi} = (\xi, \eta)$. This is done by mapping each element (with vertices $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$) onto a reference triangle with vertices at $\boldsymbol{x}_1 = (0, 0)$, $\boldsymbol{x}_2 = (1, 0)$ and $\boldsymbol{x}_3 = (0, 1)$, see also Figure 11.4. When introducing the notation $|\Omega_j|$ for the area of the element Ω_j, the following transformations are straightforward to check:

$$\begin{aligned}
\xi &= \frac{1}{2\,|\Omega_j|}\left((y_3 - y_1)(x - x_1) - (x_3 - x_1)(y - y_1)\right), \\
\eta &= \frac{1}{2\,|\Omega_j|}\left(-(y_2 - y_1)(x - x_1) + (x_2 - x_1)(y - y_1)\right).
\end{aligned}$$

and the three basis functions in this local coordinate system can be seen to be:

$$\psi_1(\xi, \eta) = 1 - \xi - \eta, \qquad \psi_2(\xi, \eta) = \xi, \qquad \psi_3(\xi, \eta) = \eta.$$

Then, as in the one-dimensional case, the required integrals can be computed by taking the appropriate products of two of these basis functions inside each element. We will not work out the details here.

Remark 11.8 (Computation of the right hand side)**.** As in the one-dimensional case (see equation (11.13)), the computation of f_m cannot be done analytically, but requires some numerical quadrature formula. Also in this case, it is convenient to use the element-wise formulation and local coordinates.

11.4 Higher order methods

11.4.1 Higher order polynomials in each element

One natural way to improve the order of accuracy of the finite element method is to consider higher order polynomials to approximate the solution inside each

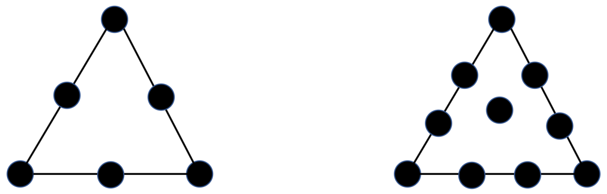
Figure 11.5: Schematic picture of higher order finite elements (edges and vertices). Left: second order element; right: third order element. .

element. As before, we want a representation of the solution that satisfies the following properties:

- Global continuity of the numerical approximation, i.e., continuity on the element edges in particular;
- A representation that allows an element-by-element computation of all necessary matrix entries.

Both requirements can be fulfilled by choosing appropriate vertices and representing the numerical solution inside an element in terms of local basis functions that are each a polynomial of the desired degree that is zero in all of these vertices except one.

Consider a general quadratic function in $\mathbb{R}^2$ of the form

$$u_j(x,y) = \alpha_j + \beta_j x + \gamma_j y + \delta_j x^2 + \eta_j xy + \zeta_j y^2,$$

which represents the numerical solution inside the element Ω_j. We see that we need to determine six parameters; we will therefore need six local basis functions and six vertices in a triangle. We also know that the solution is uniquely determined on each element edge as soon as three point on this edge are fixed. (This is because the solution is a quadratic function on every straight line, and therefore also along element edges.) As a result, a second order element can be graphically depicted as the left triangle in Figure 11.5. Similarly, a cubic function in two space dimensions has ten parameters (check this!) and we therefore need ten basis functions inside one element. Since we need to specify function values at four points along each edge to ensure global continuity of the solution, we have one degree of freedom left that cannot be placed along an element edge. We will place it at the center of mass of the triangle. This leads to the element that is schematically depicted as the right triangle in Figure 11.5.

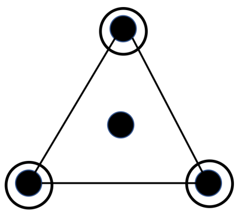

Figure 11.6: Schematic picture of a finite element that results in basis functions that have continuous derivatives across element edges. The interpolation is based on the function value and (when an extra circle is present around the vertex) both spatial derivatives.

11.4.2 Higher-order continuity along element edges

Piecewise continuity of the numerical solution is sufficient in most common situations. However, for some equations (for instance, including the biharmonic operator ∇^4), one also requires the first derivative of the basis functions be be continuous across element boundaries. This can be ensured by introducing an element of the form in figure 11.6, in which one interpolates the solution value and both spatial derivatives in each of the three vertices (which gives nine conditions), together with the solution value in the center of mass of the triangle.

11.4.3 Periodic table of finite element methods

We have only covered a small portion of finite element theory and practice. As mentioned before, a lot of effort in finite element analysis goes out to error analysis, error control and adaptivity. Another branch of research deals with the choice of the basis functions and representations. It is not necessary to represent the solution in terms of the solution values on the vertices. This is a choice – a common choice in many situations, but other choices are possible nevertheless.

Depending on the form of the equation, one can have a special interest to preserve qualitative features of the solution, dependencies between different variables in a specific model, etc. To illustrate the richness of finite element practice would be out of scope in these notes. Here, we only refer to the following website that aims at collecting and systematizing many finite element methods "out there": `http://femtable.org`. In these notes, we have only covered the P-type finite elements in one and two space dimensions.

11.5 Time-dependent problems

Move to the space-time part of the book

11.5.1 General derivation

The use of the finite element method can be extended to time-dependent problems of the form:

$$\partial_t u(x,t) + \mathcal{L}u(x,t) = f(x,t). \tag{11.29}$$

(Compare this equation with (11.23).)

The key idea is to *first* discretize the equation in space, leaving time continuous, and to only discretize time afterwards. In the finite element method, this can be achieved by introducing a numerical approximation in the finite-dimensional subspace *at each moment in time*, i.e., we write

$$u_M(x,t) = \varphi_0(x) + \sum_{m=1}^{M} c_m(t)\varphi_m(x),$$

in which the only change is that we have made the coefficients $\{c_m(t)\}_{m=1}^M$ dependent on time. Since the number of coefficients M is finite, the spatial discretization using finite elements transforms the PDE (11.29) into a (potentially very large) system of ordinary differential equations for the coefficients $\{c_m(t)\}_{m=1}^M$.

Let us now define the residual as

$$r_M(x,t) = \partial_t u_M(x,t) + \mathcal{L}u_M(x,t) - f(x,t),$$

and again perform the Galerkin projection onto the basis functions:

$$\langle r_M, \varphi_m \rangle = 0, \qquad 1 \le m \le M,$$

which we can expand to

$$\begin{aligned}\langle r_M, \varphi_m \rangle &= \langle \partial_t u_M, \varphi_m \rangle + \langle \mathcal{L}u_M, \varphi_m \rangle - \langle f, \varphi_m \rangle \\ &= \sum_{m'=1}^{M} \frac{d}{dt} c_{m'}(t) \langle \varphi_{m'}, \varphi_m \rangle + \sum_{m'=1}^{M} c_{m'}(t) \langle \mathcal{L}\varphi_{m'}, \varphi_m \rangle + \langle \mathcal{L}\varphi_0, \varphi_m \rangle - \langle f, \varphi_m \rangle\end{aligned}$$

to obtain the following system of ordinary differential equations for $c_m(t)$:

$$\sum_{m'=1}^{M} \frac{d}{dt} c_{m'}(t) \langle \varphi_{m'}, \varphi_m \rangle + \sum_{m'=1}^{M} c_{m'}(t) \langle \mathcal{L}\varphi_{m'}, \varphi_m \rangle = \langle f, \varphi_m \rangle - \langle \mathcal{L}\varphi_0, \varphi_m \rangle .$$

As in (11.12) (for the one-dimensional setting) and equation (11.28) (for the two-dimensional case), we again introduce the notation

$$a_{m,m'} = \langle \mathcal{L}\varphi_{m'}, \varphi_m \rangle \,, \qquad f_m = \langle f, \varphi_m \rangle - \langle \mathcal{L}\varphi_0, \varphi_m \rangle$$

(Remember that the coefficients $a_{m,m'}$ are typically computed using partial integration or a multidimensional variant.) The additional quantities to compute now are the coefficients

$$b_{m,m'} = \langle \varphi_{m'}, \varphi_m \rangle \,,$$

resulting in the shorthand notation

$$\sum_{m'=1}^{M} \frac{d}{dt} c_{m'}(t) b_{m,m'} + \sum_{m'=1}^{M} c_{m'}(t) a_{m,m'} = f_m. \tag{11.30}$$

Compare equation (11.30) with equation (11.14) for the time-independent case. As we did in the one-dimensional case in equation (11.15), we can again write these equations in matrix form as:

$$\boldsymbol{B} \frac{d}{dt} \boldsymbol{c}(t) + \boldsymbol{A}\boldsymbol{c}(t) = \boldsymbol{f}. \tag{11.31}$$

in which the matrix $\boldsymbol{B} = (b_{m,m'})_{m,m'=1}^{M}$ is called the *mass matrix.* For the one-dimensional two-point boundary value problem (11.1) and the piecewise linear basis functions (11.17), it can easily be checked (by following the computations in section 11.1.4.2), that the matrix B is tridiagonal, with elements

$$b_{m,m-1} = 1/6, \qquad b_{m,m} = 4/6, \qquad b_{m,m+1} = 1/6.$$

Remark 11.9 (Mass lumping)**.** Sometimes, the appearance of the mass matrix $\boldsymbol{B}$ is considered a nuisance, and one would have preferred the system (11.31) to read

$$\frac{d}{dt} \boldsymbol{c}(t) + \boldsymbol{A}\boldsymbol{c}(t) = \boldsymbol{f}, \tag{11.32}$$

i.e., the matrix $\boldsymbol{B}$ is replaced by the identity matrix. This is called *mass lumping* and it has advantages if one wants to use a software for time discretization that does not allow adding a mass matrix. When one discretizes (11.32) with forward Euler in time, one can show that this corresponds to only applying the finite element discretization in space and combining it with a finite difference approximation in time.

11.5.2 Artificial viscosity methods for advection-dominated problems

The theory on finite difference methods allows to easily understand the behaviour of space-time discretizations of various kinds of PDEs (using only a von

Neumann stability analysis and the principle of modified equations). One (very important) example is the introduction of *upwinding*. With finite elements, upwinding is very hard to encode in the choice of basis functions. However, one can make use of the observation that upwinding results in an additional diffusion term with a diffusion coefficient that depends on the mesh width. In finite element methods for advection dominated problems, one then simply modifies the equations to explicitly contain such an artificial diffusion term. (The name "artificial viscosity" arises because, in fluid flow, the quantity that is modeled is usually momentum, and diffusion of momentum is called viscosity.)