Matematisk modellering og numeriske metoder

Lektion 14

Morten Grud Rasmussen

13. november 2016

1 Numerical methods for solving differential equations

1.1 Conservation laws

In the following p denotes a point on the line, on the plane or in 3d-space. As the following considerations are valid in all three cases, we will not specify whether $p = x$, $p = (x, y)$, or $p = (x, y, z)$. Regarding terminology, we will use the spatial notions, i.e. "cube" will be used also for "rectangles" in the plane, and "interval" on the line. Likewise, we will write $\mathrm{d}V$ (as in [V]olume) when we integrate, regardless of which dimension we are dealing with, and $\mathrm{d}A$ when we integrate the "surface" (which in the case of two dimensions is a line integral, and in one dimension reduces to a sum over the end points).

As you recall from the derivation of the heat equation, we used that the thermal energy was *conserved*, so that a temporal change of the temperature in a cube $-\frac{\partial}{\partial t}\int_T u(p,t)\,\mathrm{d}V$ could be attributed to what passed through the surface area of the cube, $\int_A f(u)\cdot n\,\mathrm{d}A$ (where A is the surface of the cube T, f is the "flux" and n is an outer noral vector of unit length):

$$\frac{\partial}{\partial t}\int_T u(p,t)\,\mathrm{d}V + \int_A f(u)\cdot n\,\mathrm{d}A = 0.$$

As mentioned, this was under the assumption that the total thermal energy was *conserved*, and the equation above is therefore an example of a *conservation law*. Notice that in the above formulation, u does not have to be a temperature but can be any kind of quantity from continuum mechanics. Now what happens if we actually want to model something where thermal energy is added or subtracted (e.g. if a endo- or exothermal reaction takes place in space)? Now let S denote the added energy (S could stand for "source") in a given point in space, where S might also depend on time t and temperature u. Then the conservation law above should be modified as follows:

$$\frac{\partial}{\partial t}\int_T u(p,t)\,\mathrm{d}V + \int_A f(u)\cdot n\,\mathrm{d}A - \int_T S(p,u(p),t) = 0,$$

which again is called a *conservation law*, as it again builds on conversation principles. Notice again that the equation above could be a conservation law for any kind of contiuum mechanics quantity. As in the case with the heat equation, we can now apply Gauss' divergence theorem and get

$$\int_T \Big(\frac{\partial u}{\partial t} + \operatorname{div} f(u) - S \Big) \, \mathrm{d}V = 0.$$

With this integral we can then argue that since the cube is arbitrarily chosen, the integrant must be 0, at least if it is continuous:

$$\frac{\partial u}{\partial t} + \operatorname{div} f(u) - S = 0$$

(try to consider what each of the terms might correspond to physically – and remember that $f(u)$ is the flux of u, not just u). This is called the *strong* version of the conversation law. Notice that this is under the assumption that the integrant is continuous, which means that it doesn't cover the so-called *generalized solutions*. Fear not! This can be avoided. Note that the integral form of the conservation law doesn't require contiuity, and hence it is a weaker form (hence the name *the strong version*), but we can do away with even less! First, note that

$$\int_T \Big(\frac{\partial u}{\partial t} + \operatorname{div} f(u) - S \Big) \, \mathrm{d}V = \int_{\mathbb{R}^d} \Big(\frac{\partial u}{\partial t} + \operatorname{div} f(u) - S \Big) 1_T \, \mathrm{d}V = 0,$$

where d is the dimension and 1_T is the so-called *indicator function* of T, given by

$$1_T(p) = \begin{cases} 1 & \text{if} \quad (p) \in T \\ 0 & \text{otherwise} \end{cases}.$$

The integral form of the conservation law, which should hold for any cubes T, can thus be interpreted as the average of the expression $\frac{\partial u}{\partial t} + \operatorname{div} f(u) - S$ being tested over all possible boxes. We have of course chosen cubes out of convenience, but we might as well have chosen balls, say. Now here comes a new idea; we could also use a family of *smooth test functions*, where "smooth" means that all partial derivatives of all orders exist, instead of functions of the type 1_T! In that case, we get the expression

$$\int_{\mathbb{R}^d} \Big(\frac{\partial u}{\partial t} + \operatorname{div} f(u) - S \Big) w \, \mathrm{d}V = 0,$$

where w is a function of p. Here w stands for "weight" (the function is also called a weight function). By applying the spatial version of partial integration[1], we get

$$\int_{\mathbb{R}^d} \left(\Big(\frac{\partial u}{\partial t} - S \Big) w - f(u) \cdot \nabla w \right) \mathrm{d}V + \int_A f \cdot w \, \mathrm{d}A = 0. \tag{1}$$

Notice that we have "moved" a spatial derivative from $f(u)$ to w (which is smooth, and this last expression hence needs less assumptions on f. This means that (1) might hold for all w in a class of smooth test functions even if the other formulations of the conservation laws are not satisfied and is therefore called the *weak* version of the conservation law.

[1]The spatial version of partial integration is related to Gauss' divergence theorem!

1.2 Numerical considerations – pointwise representation

Many PDE's cannot be solved exactly, and we are forced to do numerical approximations. To perform numerical approximations we need to represent the continuous solution with only finitely many numbers.

Assume first that we have a continuous function u of one variable x. As u is continuous, $u(x)$ is close to $u(x_0)$ whenever x is close to x_0. Using this kind of reasoning, we conclude that we should get a reasonable representation of u if we pick values x_j, $j = 0, \ldots, J$ densely along the domain of definition of u, perhaps evenly spaced (such that $x_i - x_{i-1}$ is fixed) and approximate u from the value $u(x_i)$ of u in these fixed points. We choose u_j for $j = 0, \ldots, J$ so that $u_j \approx u(x_j)$, and try to describe u using these x_j's and u_j's. We call the x_i's for a *grid* or a *mesh*.

1.3 The Finite Difference Method

The simplest (and oldest) numerical method (where "numerical method" means a method of finding a numerical approximation of the "real" (analytic) solution) is the finite difference method. It's building blocks consists of *difference approximations* of derivatives. The basic idea is that derivatives, ordinary as well as partial and of arbitary order, can be approximated by different kinds of *difference quotients*. In a moment, we will go through some of the most basic ones. To ease the understanding, we will illustrate the concepts with a concrete example. To this end, we consider our one-dimensional heat equation with $c^2 = 1$, $x \in [0, 1]$ (i.e. $L = 1$), $t \in [0, T]$, Dirichlet boundary conditions and initial condition f:

$$\begin{aligned} u_t &= u_{xx} && \text{(the heat equation)} \\ u(0,t) &= u(1,t) = 0 && \text{(the boundary condition)} \\ u(x,0) &= f(x) && \text{(the initial condition)} \end{aligned}$$

First, we must choose a grid in both space and time, and these we will pick *uniformly*, i.e. $x_{j+1} - x_j = h$ for all $j = 0, \ldots, J-1$ and $t_{n+1} - t_n = k$ for all $n = 0, \ldots, N-1$. We choose $t_0 = 0$, $t_N = T$, $x_0 = 0$, and $x_J = 1$, so that the endpoints of our spatial and temporal domain is included in the mask. The solution u will then be approximated by $u_j^n \approx u(x_j, t_n)$. Notice that the n in u_j^n *doesn't* mean that u_j is raised to a power; it's just an index denoting the current time step!

Different first order difference quotients

As you know, the heat equation contains a first order time derivative. We will approximate the time derivative using the u_j^n's (remember that j indicates the spatial position, while the n indicates the temporal position). The three simplest ways of doing this are the following expressions:

$$\begin{aligned} u_t(x_j, t_n) &\approx \frac{u_j^{n+1} - u_j^n}{k} && \text{(\textit{forward difference})} \\ u_t(x_j, t_n) &\approx \frac{u_j^n - u_j^{n-1}}{k} && \text{(\textit{backward difference})} \\ u_t(x_j, t_n) &\approx \frac{u_j^{n+1} - u_j^{n-1}}{2k} && \text{(\textit{central difference})} \end{aligned}$$

each of which have their pros and cons, both when it comes to implementation and precision. The idea behind the all are, however, the same: the fractions are the slopes between the involved values of u_j^m, $m \in \{n-1, n, n+1\}$.

The central second order difference quotient

We will now find an approximation of the second order spatial derivative, u_{xx}, which also appears in the heat equation. This means that we need to find a difference quotient between two difference quotients, e.g.

$$u_{xx}(x_j, t_n) \approx \frac{\frac{u^n_{j+1}-u^n_j}{h} - \frac{u^n_j-u^n_{j-1}}{h}}{h} = \frac{u^n_{j+1} - 2u^n_j + u^n_{j-1}}{h^2},$$

which is called the *central second order difference quotient*.

An explicit method

We will now describe a so-called *explicit* finite difference method of numerically solving the heat equation; the term explicit comes from the fact that – knowing the initial conditions – for every time step one has an explicit expression for every point in the next time step. The method is known as the FTCS method, which is short for *forward in time, central in space*, which should make sense if one takes a closer look at the following. The idea is as follows: as

$$u_t = u_{xx}, \qquad \text{while} \qquad u_t(x_j, t_n) \approx \frac{u^{n+1}_j - u^n_j}{k} \qquad \text{and} \qquad u_{xx}(x_j, t_n) \approx \frac{u^n_{j+1} - 2u^n_j + u^n_{j-1}}{h^2},$$

then

$$\frac{u^{n+1}_j - u^n_j}{k} \approx \frac{u^n_{j+1} - 2u^n_j + u^n_{j-1}}{h^2}, \tag{2}$$

where the approximation is best for small values of h and very small values of k. If we now remember that u^n_j are just *approximations* of $u(x_j, t_n)$, then we can simply *define* our u^n_j as the solution of the equations above for all combinations of the following choices of j and n:

$$j \in \{1, \ldots, J-1\} \qquad \text{and} \qquad n \in \{1, \ldots, N\}.$$

The remaining cases are determined by the initial and boundary conditions. As for the initial conditions, an obvious choice is to do the following:

$$u^{t_0}_j = u^0_j = f(x_j) \qquad \text{for} \qquad j \in \{0, \ldots, J\},$$

while the boundary conditions gives us

$$u^n_0 = u^n_J = 0 \qquad \text{for} \qquad n \in \{0, \ldots, N\},$$

where of course we need to assume that $f(0) = f(1) = 0$ in order for the initial and boundary conditions to be consistent, meaning that u^0_0 and u^0_J are both 0 independent of which of the expressions we use to define them.

That the method is *explicit* means that we do not have to solve any systems of equations to get to the next time step, we only need to isolate u^{n+1}_j in (2):

$$u^{n+1}_j = (1 - 2\frac{k}{h^2})u^n_j + \frac{k}{h^2}(u^n_{j-1} + u^n_{j+1}) = (1-2r)u^n_j + r(u^n_{j-1} + u^n_{j+1}), \qquad \text{where} \qquad r = \frac{k}{h^2},$$

giving us u^{n+1}_j as a function of u^n_{j-1}, u^n_j, and u^n_{j+1}, all of which belong to an earlier time step.

We will now write it all in matrix form:

$$\begin{bmatrix} u_0^0 \\ u_1^0 \\ u_2^0 \\ \vdots \\ u_j^0 \\ \vdots \\ u_{J-2}^0 \\ u_{J-1}^0 \\ u_J^0 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_j) \\ \vdots \\ f(x_{J-2}) \\ f(x_{J-1}) \\ f(x_J) \end{bmatrix}$$

and

$$\begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_j^{n+1} \\ \vdots \\ u_{J-2}^{n+1} \\ u_{J-1}^{n+1} \\ u_J^{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & & & \cdots & & & & 0 \\ r & 1-2r & r & 0 & & \cdots & & & 0 \\ 0 & r & 1-2r & r & 0 & & \cdots & & 0 \\ & & & & & & & & \\ \vdots & & \ddots & \ddots & \ddots & & & & \vdots \\ & & & & & & & & \\ 0 & & \cdots & & 0 & r & 1-2r & r & 0 \\ 0 & & & \cdots & & 0 & r & 1-2r & r \\ 0 & & & & \cdots & & & 0 & 1 \end{bmatrix} \begin{bmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ u_j^n \\ \vdots \\ u_{J-2}^n \\ u_{J-1}^n \\ u_J^n \end{bmatrix} \qquad \text{for} \qquad n \geq 0,$$

or briefly $u^0 = f$ and $u^n = Au^{n-1} = A^n u^0$ for $n \geq 1$, for a suitable definition of u^n, A, and f. Clearly, no equations need to be solved, one should just compute matrix products, which is why it's called an *explicit* method. One can show that this method is what is called *numerically stable* and *convergent* whenever $r = \frac{k}{h^2} \leq \frac{1}{2}$. The numerical errors are at most proportional to the size of k and the square of h, which one can also write as $E = O(k) + O(h^2)$.

An implicit method

We will now describe another method, a so-called *implicit method*, where one has to solve some equations after writing down the model in order to get the result. Where the other method was called the FTCS-method, this method is called the BTCS-method: "backwards in space, central in space."

As this name indicates, the main difference between the two methods lies in the fact that one uses the *backward difference* approximation

$$u_t(x_j, t_n) \approx \frac{u_j^n - u_j^{n-1}}{k}$$

instead of the *forward difference* approximation of the time derivative. Using the same arguments as before, we get the system of linear equations

$$\frac{u_j^{n+1} - u_j^n}{k} = \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2}.$$

Notice that here we cannot find u_j^{n+1} just by using information from earlier time steps; the equation contains u_{j+1}^{n+1} and u_{j-1}^{n+1} which are also unknown, and instead we end up with the *system of linear equations*:

$$(1+2r)u_j^{n+1} - ru_{j-1}^{n+1} - ru_{j+1}^{n+1} = u_j^n \qquad \text{where} \qquad r = \frac{k}{h^2},$$

or, when we also consider the initial and boundary conditions and write it all up in matrix form:

$$\begin{bmatrix} u_0^0 \\ u_1^0 \\ u_2^0 \\ \vdots \\ u_j^0 \\ \vdots \\ u_{J-2}^0 \\ u_{J-1}^0 \\ u_J^0 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_j) \\ \vdots \\ f(x_{J-2}) \\ f(x_{J-1}) \\ f(x_J) \end{bmatrix}$$

and

$$\begin{bmatrix} 1 & 0 & & & \cdots & & & 0 \\ -r & 1+2r & -r & 0 & \cdots & & & 0 \\ 0 & -r & 1+2r & -r & 0 & \cdots & & 0 \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & & \cdots & 0 & -r & 1+2r & -r & 0 \\ 0 & & & \cdots & 0 & -r & 1+2r & -r \\ 0 & & & & \cdots & & 0 & 1 \end{bmatrix} \begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_j^{n+1} \\ \vdots \\ u_{J-2}^{n+1} \\ u_{J-1}^{n+1} \\ u_J^{n+1} \end{bmatrix} = \begin{bmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ u_j^n \\ \vdots \\ u_{J-2}^n \\ u_{J-1}^n \\ u_J^n \end{bmatrix} \qquad \text{for} \qquad n \geq 0,$$

or more compactly $u^0 = f$ and $u^n = Au^{n+1}$ for a suitable definition of u^n, f, and A. Notice that as opposed to before, where we had $u^n = Au^{n-1} = A^n u^0$, we cannot write u^{n+1} directly in an iterated form; at each time step we have to solve the *system of linear equations* $u^n = Au^{n+1}$. This makes this method computationally more involved, but the reward is that this method *always* is numerically stable (independent of r), and even though the error still behaves as in the explicit method, $E = O(k) + O(h^2)$, the implicit method gives better results, in particular for large time steps.

1.4 Numerical considerations – element-wise representation

Let us assume that we still want to represent a function using finitely many numbers, but instead of finding approximations of the function in a given set of points, we subdivide the function in *elements*[2], where we assume that each element of the function can be approximated by a function of a simpler type, e.g. a first order polynomial, in a satisfying way.

[2]The literature disagrees about whether it's the domain or the function which is subdivided in "elements." We remain neutral.

Assume first that we have a continuous function u of one variable. As u is continuous, we also want the approximation to be continuous. We now subdivide the domain of u into a series of subintervals ("elements"), which can be written as $[x_{i-1}, x_i]$ for a suitable choice of x_i, $i = 0, \ldots, N$. On each subinterval we approximate u with a first order polynomial (we could also have chosen higher order polynomials or some completely different class of functions), but in such a way that the approximation is still *continuous*, i.e. the polynomial on $[x_{i-1}, x_i]$ should agree with the polynomial on $[x_i, x_{i+1}]$ in the common point x_i.

As opposed to earlier, where u was approximated by values in a series of points, we now approximate u using a *function* from a certain *class* V of functions (V is the set of functions which are continuous on u's domain and equals a first order polynomial on the subintervals $[x_i, x_{i+1}]$), which are not differentiable everywhere (they are usually not differentiable in x_i), and whose second order derivative is 0 where it exists (the second derivative of a first order polynomial is 0).

The question now is how one in a sensible way chooses the function $v \in V$ such that v is a good approximation of u. For many reasons, some of which only appear in higher dimensions than the one dimension we have in the present example, the finite difference method is not optimal for this, which should also be intuitively clear since this method makes no use of the fact that v is an actual function with the same domain as u, and not just a finite collection of values. If one combines the ideas from the section about conservation laws with the fact that v is not differentiable everywhere, one should be able to guess that a good solution is to reformulate the differential equation in a *weak* form.

1.5 The Finite Element Method

We will now introduce the finite element method by illustrating it with two examples. One example will be an ODE example which can actually be solved numerically in a more elegant way, but which can be described in very concretely and which serves the purpose of illustrating the ideas quite clearly, while the other example is a PDE example, which we will go through more superficially, and which serves the purpose of illustrating how to use the method on more complex problems. The two problems are chosen so that they resemble each other and can be described briefly as Poisson problems with Dirichlet boundary conditions in one and two dimensions, respectively.

Example 1

The first example is the one-dimensional Poisson problem with homogeneous Dirichlet boundaries on $[0, 1]$:

$$u''(x) = f(x) \qquad \text{for} \quad x \in (0, 1), \quad u(0) = u(1) = 0,$$

where $f\colon (0, 1) \to \mathbb{R}$ is a given function.

Example 2

The second example is the two-dimensional Poisson problem with the homogeneous Dirichlet boundary conditions on the "sufficiently nice" set Ω whose boundary is denoted by $\partial\Omega$:

$$\begin{aligned} u_{xx}(x, y) + u_{yy}(x, y) &= f(x, y) && \text{for} \quad (x, y) \in \Omega, \\ u(x, y) &= 0 && \text{for} \quad (x, y) \in \partial\Omega. \end{aligned}$$

for a given function $f\colon \Omega \to \mathbb{R}$.

The weak formulation of the problem

As already hinted at, the first step is to reformulate the problem in the weak form. As discussed in the first section, the weak formulation of a differential equation is constructed by multiplying both sides with a test function and integrating. In the case of Example 1 we get:

$$\int_0^1 u''(x)v(x)\,\mathrm{d}x = \int_0^1 f(x)v(x)\,\mathrm{d}x,$$

where v is a test function. Because of the boundary conditions $u(0) = u(1) = 0$, we limit our test functions to those satisfying:

$$v(0) = v(1) = 0. \tag{3}$$

Combining this with partial integrations gives

$$\begin{aligned}\int_0^1 f(x)v(x)\,\mathrm{d}x &= [u'(x)v(x)]_{x=0}^1 - \int_0^1 u'(x)v'(x)\,\mathrm{d}x \\ &= u'(1)\cdot 0 - u'(0)\cdot 0 - \int_0^1 u'(x)v'(x)\,\mathrm{d}x \\ &= -\int_0^1 u'(x)v'(x)\,\mathrm{d}x. \end{aligned} \tag{4}$$

In *exactly the same way* we get

$$\int_\Omega fv\,\mathrm{d}s = -\int_\Omega \nabla u\cdot\nabla v\,\mathrm{d}s, \tag{5}$$

in the case of Example 2, where $\nabla w = \frac{\partial w}{\partial x} + \frac{\partial w}{\partial y}$. For a suitable choice of test functions (4) and (5) are the weak formulations of the problems. The next topic is what kind of test functions we should use.

Choice of test functions

One can show that a good choice of test functions is using the same type of functions as the numerical solution we are looking for. This means that in Example 1, the test functions should be continuous on $[0, 1]$ and equal a first order polynomial on the subintervals $[x_i, x_{i+1}]$, where $x_0 = 0$, $x_N = 1$, and $x_i < x_{i+1}$ is a subdivision of $[0, 1]$. We notice that we already have an additional requirement on the test functions, namely that they should take the value 0 at the endpoints 0 and 1, which is also dictated for the numerical solution by the Dirichlet boundary conditions! We will denote the set of test functions by V_0, i.e.

$$V_0 = \{v\colon [0,1]\to\mathbb{R} \mid v \text{ continouos}, v(0) = v(1) = 0, \text{ and } v_{|[x_i,x_{i+1}]}(x) = p_i(x), i = 0,\ldots,N-1\},$$

where $v_{|[x_i,x_{i+1}]}$ denotes the restriction of v to $[x_i, x_{i+1}]$ and p_i denotes a first order polynomial.Hence, V_0 denotes both the set of test functions and the set of functions in which we search for our numerical solution.

How about Example 2? First we need to generalize what we mean by an *element* in Example 2. Where in Example 1, it is only possible to divide the interval in one way, namely subintervals, the two-dimensional domain of Example 2 can be divided in many more ways. It turns out that

there many good ways of doing this. One of the most common ways if doing it is to subdivide the domain into triangular areas (note that this emphatically does not agree with a finite difference approach). What about the requirement that the numerical approximations should be first order polynomials on each triangle? Well, polynomials also exist in higher dimensional version, and a first order polynomial of to variables is a function of the form

$$p(x, y) = a_1 x + a_2 y + b$$

(while a second order polynomial of two variables is of the form $a_{11}x^2+a_{12}xy+a_{22}y^2+b_1x+b_2y+c$).

Choice of basis for the test functions

Again we begin with Example 1. It is not difficult to see that V_0 is a *finite-dimensional vector space*, which means that there exist a finite number of functions ($N-1$ to be precise) such that all elements of V_0 can be written as a linear combination of these finitely many functions. Such a set is called a *basis*. Since $v(0) = v(1) = 0$ for all V_0 and v must be continuous, the same must hold for the basis functions. As v equals a first order polynomial on each subinterval of the type $[x_i, x_{i+1}]$, we know v on $[x_i, x_{i+1}]$ if we know $v(x_i)$ and $v(x_{i+1})$. Using this, one can prove that

$$v_k(x) = \begin{cases} \frac{x-x_{k-1}}{x_k-x_{k-1}} & \text{for} \quad x \in [x_{k-1}, x_k] \\ \frac{x_{k+1}-x}{x_{k+1}-x_k} & \text{for} \quad x \in [x_k, x_{k+1}] \\ 0 & \text{ellers,} \end{cases}$$

$k = 1, \ldots, N-1$, constitutes a basis for V_0. Hence, a basis function v_k takes the value 1 in x_k and 0 in every other x_i, is continuous and equals a first order polynomial on each subinterval $[x_i, x_{i+1}]$.

This description of the v_k's is the key to the description of a basis in Example 2, where we chose triangular elements. Each triangle can be given by the coordinates of the three corners, and the set of corners (all triangles included), we can denote by $\{x_k : k \in K\}$, for a suitable index set K.Let K_0 denote those k for which $x_k \notin \partial\Omega$. The basis functions are then those v_k, $k \in K_0$, satisfying that $v_k(x_k) = 1$, $v_k(x_j) = 0$ for $k \neq j \in K$, and v_k restricted to each triangular element equals some first order polynomial.

Taking advantage of the small support of the basis vectors

The *support* of a function f is the smallest closed set S such that $f(x) = 0$ for $x \notin S$. For the basis vectors v_k in Example 1, the support is thus $[x_{k-1}, x_{k+1}]$. If N is big, these intervals are typically small, and v_k is said to have small support. The support of a function contains the support of the derivative of the function, and this means that the following two integrals will be 0 for most choices of k and j (again assuming that N is big):

$$\int_0^1 v_j(x)v_k(x)\,\mathrm{d}x \qquad \text{and} \qquad \int_0^1 v_j'(x)v_k'(x)\,\mathrm{d}x$$

(concretely, they will be 0 for $|k-j| > 1$).

Likewise,

$$\int_\Omega v_j v_k\,\mathrm{d}s \qquad \text{and} \qquad \int_\Omega \nabla v_j \cdot \nabla v_k\,\mathrm{d}s$$

will be 0, whenever x_j and x_k are not "neighboring corners."

Matrix formulation of the problem

We now want to show how one finds a numerical solution in the case of Example 1. As we have a basis for V_0, finding a numerical solution to our problem corresponds to finding coefficients to the basis vectors. Finding coefficients for a basis sounds like a linear algebra problem, and it also turns out one can formulate it as a matrix problem. We begin by noting that we are looking for a numerical solution v of the form

$$v(x) = \sum_{i=1}^{N-1} u_i v_i(x),$$

where the u_i are unknown coefficients and the v_i are the basis vectors. The weak form of the problem (4) thus takes the form

$$\int_0^1 f(x)v_j(x)\,\mathrm{d}x = -\int_0^1 \Big(\sum_{k=1}^{N-1} u_k v_k'(x)\Big)v_j'(x)\,\mathrm{d}x = -\sum_{k=1}^{N-1} u_k \int_0^1 v_k'(x)v_j'(x)\,\mathrm{d}x,$$

which must hold for all $j = 1, \ldots, N-1$. Here the left-hand side can be evaluated numerically, and the right-hand side is the sum of some unknown coefficients multiplied by some integrals that can also be evaluated numerically, and which are 0 most of the time anyway (in fact it would be enough to just sum over $k \in \{j-1, j, j+1\}$). Written in matrix form, we get

$$-\begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{N-1,1} \\ A_{2,1} & A_{2,2} & \cdots & A_{N-1,2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1,N-1} & A_{2,N-1} & \cdots & A_{N-1,N-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N-1} \end{bmatrix},$$

or $-Au = b$ for a suitable definition of A, u, and b, where

$$A_{j,k} = \int_0^1 v_j'(x)v_k'(x)\,\mathrm{d}x$$

and

$$b_j = \int_0^1 f(x)v_j(x)\,\mathrm{d}x.$$

Here we note that $A_{j,k}$ is 0 except for $|j-k| \leq 1$, or, put in a different way, the A matrix has 0 entries except on, right above, and right below the diagonal, cf. the discussion about the small support, which means that there are efficient ways of solving the matrix equation, even for large values of N. In the case of Example 2, we get something completely analogous, just with $A = (A_{j,k})_{j,k\in K_0}$, $u = (u_k)_{k\in K_0}$, and $b = (b_j)_{j\in K_0}$ suitably redefined. Again, A is a so-called *sparse* matrix and hence it is also possible to solve this matrix equation numerically, even for a very large number of elements (corresponding to a large set K_0).

Final comment concerning the finite element method

It should be stressed that the free choice of "elements" in the finite element method can be used in very efficient ways. If some area is of particular interest, and precision in this area is important, one can just increase the resolution *locally*. Or if a model describes something which consists of

several different kinds of material, and the transision between the different materials is of special importance, one can just increase the resolution in these areas – again locally and hence without increasing the computational costs too much. A last example is that if a solution behaves in a fairly simple way in some parts of the domain, while other areas have a more involved behavior, one can of course again just increase the resolution in these areas. In fact, it is possible to estimate the local "quality" of the output as a biproduct of the computations, and one can thus automate such a refinement of the element division.

1.6 Numerical considerations – conservation laws and volumes

So far, we haven't really taken advantage of the fact that we are dealing with *conversation laws*. In many cases, however, this property can mean a lot for the model and be of great help in the computations. Assume for example that we have some form of chemical solution in a liquid in some more or less complicated system, and that we are fully satisfied with knowing the average value of the concentration in so-called *control volumes*, but interested in that the flow is as correct as possible, and that possible errors cancel out, such that we for example don't suddenly have a larger amount of dissolved material than was originally added to the system. In such cases, one can use the conservation laws, and it is this idea the *finite volume method* builds on. In the finite volume method, the domain of the model of the problem is subdivided into so-called control volumes, e.g. consisting of tetrahedrons ("three-dimensional triangles"), and the numerical solution will be the average value in these control volumes.

1.7 The Finite Volume Method

As mentioned above, the conservation laws play a great role in the finite volume method. First, the space is subdivided into some control volumes. The idea is, as already hinted at above, to use that the integral over divergence terms can be converted into surface integrals via Gauss' divergence theorem.These surface integrals on each control volume are then evaluated as flux values on the surface of each control volume. The method is locally conservative, because the idea is to take advantage of the fact that a flux out of the side of one control volume necessarily must equal the flux into the side of the neighboring control volume. We will now briefly illustrate the idea use Example 1 from the finite element section.

Example 1 retold in conservative form

The one-dimensional Poisson problem with homogeneous Dirichlet boundary conditions on $[0, 1]$ is given by:

$$\begin{aligned} u''(x) &= f(x) \qquad \text{for} \quad x \in (0,1) \\ u(0) &= u(1) = 0, \end{aligned}$$

where $f\colon (0,1) \to \mathbb{R}$ is a given function.

As we are in one dimension, the divergence is just the x-derivative, and we can thus reformulate the problem in the following way:

$$\operatorname{div} u'(x) = f(x) \qquad \text{for} \quad x \in (0,1). \tag{6}$$

Subdividing the domain

As we need to be able to refer to both midpoints and edges of the control volumes (which are just intervals, thanks to the one dimension), we need to put a bit more care into our subdivision than earlier. More precisely we need

$$0 = x_0 = x_{\frac{1}{2}} < x_1 < x_{\frac{3}{2}} < \cdots < x_{i-\frac{1}{2}} < x_i < x_{i+\frac{1}{2}} < \cdots < x_N < x_{N+\frac{1}{2}} = x_{N+1} = 1,$$

and our control volumes are then $K_i = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}})$, $i = 1, \ldots, N$. We write $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ and $\Delta x_{i+\frac{1}{2}} = x_{i+1} - x_i$ for $i = 1, \ldots N$. The unknowns are now u_i, $i = 1, \ldots, N$ and should be interpreted as approximations of the average values of u over K_i.

Matrix formulation of the problem

We integrate (6) over K_i and apply Gauss' divergens theorem, which in one dimension reduces to the fundamental theorem of calculus:

$$u'(x_{i+\frac{1}{2}}) - u'(x_{i-\frac{1}{2}}) = \int_{K_i} f(x)\,\mathrm{d}x.$$

We can now approximate u' with a difference quotient (*central difference*):

$$\frac{u(x_{i+1}) - u(x_i)}{\Delta x_{i+\frac{1}{2}}} - \frac{u(x_i) - u(x_{i-1})}{\Delta x_{i-\frac{1}{2}}} \approx \int_{K_i} f(x)\,\mathrm{d}x.$$

As u_i should just *approximate* u's average value over K_i, we choose to define the u_i's by the following relation:

$$\frac{u_{i+1} - u_i}{\Delta x_{i+\frac{1}{2}}} - \frac{u_i - u_{i-1}}{\Delta x_{i-\frac{1}{2}}} = \int_{K_i} f(x)\,\mathrm{d}x,$$

which is valid for all $i = 1, \ldots, N$, if one puts $u_0 = u_{N+1} = 0$. This can also be written in matrix form:

$$Au = f,$$

where $u = \begin{bmatrix} u_0 & u_1 & \cdots & u_N & u_{N+1} \end{bmatrix}^T$, $f = \begin{bmatrix} 0 & \int_{K_1} f(x)\,\mathrm{d}x & \cdots & \int_{K_i} f(x)\,\mathrm{d}x & \cdots \int_{K_N} f(x)\,\mathrm{d}x & 0 \end{bmatrix}^T$ and $A = (A_{ij})_{i,j\in\{0,1,\ldots,N,N+1\}}$ is given by $(Au)_0 = u_0$, $(Au)_{N+1} = u_{N+1}$, and $(Au)_i = \frac{u_{i+1}-u_i}{\Delta x_{i+\frac{1}{2}}} - \frac{u_i - u_{i-1}}{\Delta x_{i-\frac{1}{2}}}$ for $i = 1, \ldots, N$.

Final comment concerning the finite volume method

As for the finite element method, it should be stressed that the free choice of control volumes in the finite volume method can be used efficiently for approximately the same reasons as before and with approximately the same arguments. In addition, the locally conservative method means that errors tend to cancel: what one cell loses, the next one gains, so that the overall flow stays close to being correct.