# *Introduction to Finite Element Modelling in Geosciences:* Basic Principles

D. A. May & M. Frehner
ETH Zürich

July 6, 2014

## 1 Literature & general information

The following script is largely based on lecture notes from Guy Simpson (now at University of Geneva) and Boris Kaus (now at Johannes Gutenberg University Mainz), who taught a similar course at ETH in the past. The emphasis in the lecture is to write Finite Element codes from scratch. All codes are written in MATLAB, and we expect you to be familiar with MATLAB. It is also advantageous if you already followed classes on programming the finite difference method (e.g. 651-4241-00L, Numerical Modelling I & II: Theory & applications by Gerya and Tackley). We will mostly use lecture notes. In addition, we recommend the following textbooks:

- The Finite Element Method using MATLAB. Kwon and Bang (1997).
- The Finite Element Method, Hughes (2000).
- The Finite Element Method, vol. 1, Zienkiewicz and Taylor (2000).
- Programming the Finite Element Method, Smith and Griths (1998).
- Computational Techniques for Fluid Dynamics, vol. 1, Fletcher (2000).
- Finite Elements and Fast Iterative Solvers, Elman, Silvester & Wathan (2005).

For a brief and more general background on finite elements and computational geodynamics, you can have a look at

- Numerical Geodynamics: An introduction to computational methods with focus on solid Earth applications of continuum mechanics. Lecture notes, University of Southern California, Becker and Kaus (2010). Available on www.gfd.ethz.ch/~kaus/Teaching.html
- Computational Geodyamics. Ismail-Zadeh and Tackley (2010).

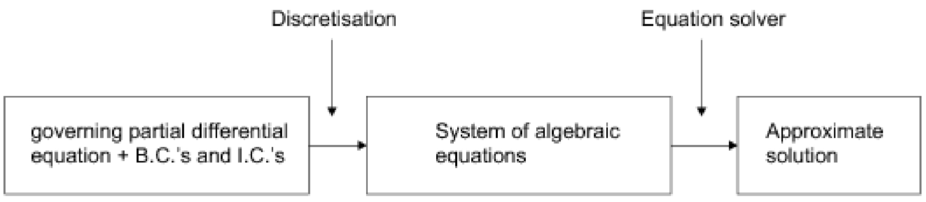
Discretisation
Equation solver
governing partial differential equation + B.C.'s and I.C.'s
System of algebraic equations
Approximate solution

Figure 1: Overview of the computational solution technique.

## 2 Introduction

The purpose of this course is to learn how to solve differential equations with the Finite Element Method (FEM). For the most part we assume that the equations governing the processes of interest are known and given. We will focus on the practical problem of how one goes from the equation to its solution. Although there are a variety of numerical techniques that one can use to obtain numerical solutions we will focus here solely on FEM.

The process of obtaining a computational solution consists of two stages shown schematically in Figure 1. The first stage converts the continuous partial differential equation and auxiliary (boundary and initial) conditions into a discrete system of algebraic equations. This first stage is called discretisation. The second stage involves solving the system of algebraic equations to obtain an approximate solution to the original differential equation. The solution is approximate since errors are introduced by the replacement of continuous differential terms in the governing partial differential equation by algebraic expressions connecting nodal values on a finite grid. Errors are also introduced during the solution stage but these tend to be small in comparison to discretisation errors unless the method is unstable.

To convert the governing partial differential equation to a system of algebraic equations (or ordinary differential equations) a number of choices are available. The most common are the finite difference, finite element, finite volume and spectral methods. In principle, the solution does not depend on the method chosen. Each method has its own advantages and disadvantages. The best approach is to choose the method which best suits the problem being investigated. This course will only deal with the finite element method, which is widely used in practice and is extremely powerful. The technique is slightly harder to learn than, for example, the finite difference technique. However, as you will see, the effort invested in initially learning FEM pays of later in the wide range of problems that the method is capable of solving. The FEM is especially well suited (though not restricted) to solving mechanical problems and problems which involve complex shapes. Another advantage of programming in the finite element method is that the main structure of the code remains the same even for very different physical problems. Thus, once you learn this basic structure, you can modify it to solve a variety of problems with minimal effort.

## 3 The Finite Element Method

The finite element method is a technique for solving partial differential equations. Usually, only the spatial derivatives are discretized with the finite element method whereas finite differences are used to discretize time derivatives. Spatial discretization is carried out locally over small regions of simple but arbitrary shaped elements (the finite elements). This discretization process results in matrix equations relating the loads (input) at specified points in the element (called nodes) to the displacements (output) at these same points. In order to solve equations over larger regions, one sums node-by-node the matrix equations for the smaller sub-regions (elements) resulting in a global matrix equation. This system of equations can then be solved simultaneously by standard linear algebra techniques to yield nodal displacements. This last step completes the numerical solution of the differential equation.

## 4 The Finite Element Method in 1D

The various steps involved in performing the finite element method are best illustrated with a simple example. Consider the partial differential equation

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial T}{\partial x}\right) + s(x), \tag{1}$$

which governs transient heat conduction in one dimension with a source term $s(x)$. The dependent variable in this equation is the temperature $T(x,t)$, the independent variables are time $t$ and distance $x$, and $\kappa(x)$ is the thermal diffusivity. We are interested in computing the temperature function $T(x,t)$ which satisfies Equation (1) (i.e., the solution) over the domain $\Omega = [x_A, x_B]$ subject to either i) Dirichlet boundary conditions of the form

$$\begin{aligned} T(x_A,t) &= T_A \\ T(x_B,t) &= T_B, \end{aligned} \tag{2}$$

where $T_A, T_B$ are prescribed temperatures, or ii) Neumann boundary conditions of the form

$$\begin{aligned} \kappa\frac{\partial T}{\partial x}\bigg|_{x=x_A,t} &= q_A \\ \kappa\frac{\partial T}{\partial x}\bigg|_{x=x_B,t} &= q_B, \end{aligned} \tag{3}$$

where $q_A, q_B$ are prescribed fluxes or iii) some mixture of Dirichlet and Neumann conditions. We also require an initial condition

$$T(x, t=0) = T_0(x). \tag{4}$$

The first step of the finite element method involves choosing an element-type which defines where and how the discretization is carried out. The simplest element for one dimensional problems is a 2-node element (Figure 2a). As we will see, one can use more nodes per element which will have the effect of increasing accuracy but also increasing the amount of equations and thus the cost of numerical solution.

The second step of the finite element method involves approximating the continuous variable $T$ in terms of nodal variables $T_i$ using simple functions $N_i(x)$ called shape functions. If one focuses on one element (which contains 2 nodes), and one assumes that temperature varies linearly between two nodes, one can write

$$T(x) \approx N_1(x)T_1 + N_2(x)T_2, \tag{5}$$

or, using matrix notation

$$T(x) \approx \begin{bmatrix} N_1(x) & N_2(x) \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \mathbf{NT}. \tag{6}$$

In these equations, $T$ is the continuous variable which we are approximating within any given element in terms of the temperatures at the two nodes $T_1$ and $T_2$. Since we made the choice that temperature varies linearly between two nodes, we have to use the following shape functions,

$$N_1(x) = 1 - \frac{x}{L}, \qquad N_2(x) = \frac{x}{L}, \tag{7}$$

where $L$ is the length of the element and $x$ is the spatial variable which varies from 0 at node 1 to $L$ at node 2 (Figure 2b). Note the following important properties of the shape functions

- $N_1 = 1$ at node 1 while $N_1 = 0$ at node 2.
- $N_2 = 0$ at node 1 while $N_2 = 1$ at node 2.
- $N_1(x) + N_2(x) = 1$ (over the entire element).
- The functions are only local (i.e., they only connect adjacent nodes).

Note that the shape functions are simply interpolating functions (i.e., they are used to interpolate the solution over a finite element). Also, as will become clear in the following lectures, the choice of the shape functions is directly related to the choice of an element type. For example in one-dimension, variation in a 2-node element cannot be uniquely described by a function with an order greater than linear (two parameter model), variation in a 3-node element cannot be uniquely described by a function with an order greater than quadratic (three parameter model), etc.

The next step of the finite element method is to substitute our approximation for the continuous variable into the governing differential equation. Thus, substituting Equation (6) into Equation (1) leads to

$$\frac{\partial}{\partial t}\left(\begin{bmatrix} N_1(x) & N_2(x) \end{bmatrix}\begin{bmatrix} T_1 \\ T_2 \end{bmatrix}\right) - \frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial}{\partial x}\left(\begin{bmatrix} N_1(x) & N_2(x) \end{bmatrix}\begin{bmatrix} T_1 \\ T_2 \end{bmatrix}\right)\right) - s(x) = R, \tag{8}$$

where $R$ (the residual) is a measure of the error introduced during discretization. Note that the original partial differential equation has now been replaced by an equation in the discretized (nodal) variables $T_1$ and $T_2$. Thus, we now have one equation for two unknowns, which obviously cannot be solved. The problem now reduces to finding values for $T_1$ and $T_2$ such that the residual is minimized (ideally $R$ is zero as in the original equation). However to do so, we have to generate a system of equations where the number of equations equals the number of unknowns.

In the finite element method, the unknown coefficients $T_i$ are determined by requiring that the integral of the weighted residual is zero on an element

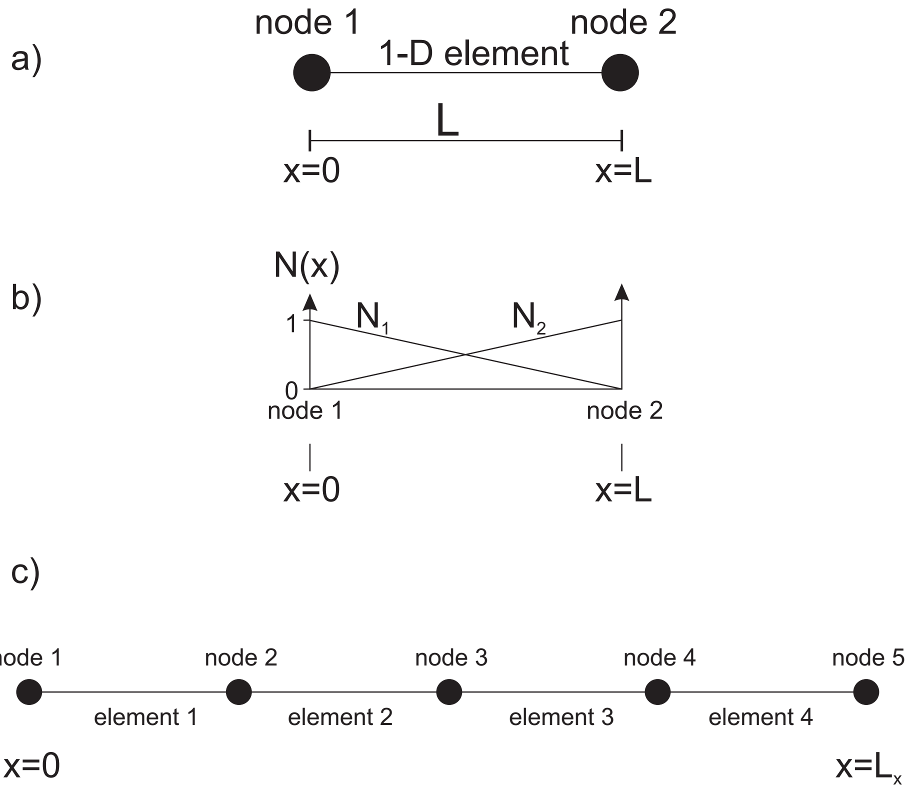

Figure 2: a) a 2-node, one dimensional finite element with b) linear shape functions, and c) a small one dimensional mesh consisting of four elements and a total of five nodes. See text for discussion.

basis. To achieve this step practically, one must multiply (or "weight") the residual in Equation (8) by a set of weighting functions (each in turn), integrate over the element volume and equate to zero. Many methods (e.g., collocation, subdomain, least squares and Galerkin) can be used to achieve this process, the difference between which depends on the choice of the weighting functions. In this course we will only consider the Galerkin method. In the Galerkin method, the weighting functions are chosen to be identical to the shape functions $N$. By carrying out the steps just described one obtains

$$\begin{aligned}&\int_0^L \begin{bmatrix}N_1\\N_2\end{bmatrix}\frac{\partial}{\partial t}\left(\begin{bmatrix}N_1 & N_2\end{bmatrix}\begin{bmatrix}T_1\\T_2\end{bmatrix}\right)dx\\ -&\int_0^L \begin{bmatrix}N_1\\N_2\end{bmatrix}\frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial}{\partial x}\left(\begin{bmatrix}N_1 & N_2\end{bmatrix}\begin{bmatrix}T_1\\T_2\end{bmatrix}\right)\right)dx - \int_0^L \begin{bmatrix}N_1\\N_2\end{bmatrix} s(x)\,dx = \begin{bmatrix}0\\0\end{bmatrix}.\end{aligned} \tag{9}$$

Note that in this example where the shape functions are linear, double differentiation of these functions would cause them to vanish (obviously not very desirable). This difficulty is resolved by applying Green's theorem (integration by parts). In one dimension, applied over the volume $\Omega = [x_A, x_B]$ this yields

$$\int_\Omega N_i \frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial N_j}{\partial x}\right)dx = -\int_\Omega \kappa(x)\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\,dx + \kappa(x) N_i \frac{\partial N_j}{\partial x}\bigg|_{x_A}^{x_B}. \tag{10}$$

We note that the final term in the above expression which is evaluated at the end points of the domain corresponds to a discrete flux. By invoking Green's theorem, we have implicitly introduced the Neumann boundary condition (see Equation (3)) into our discretisation. We will assume that $\kappa(x)$ and $s(x)$ are constant over each element volume $[x_A, x_B]$ and the constant values are denoted by $\bar{\kappa}$ and $\bar{s}$ respectively. Note that this choices does not preclude using a spatially dependent value for $\kappa$ or $s$ over the entire model domain, it only places a restriction on how these coefficients may vary *within* an element. Under these assumptions and using Equation (10), we can write Equation (9) as

$$\begin{aligned}&\begin{bmatrix}\int_0^L N_1N_1\,dx & \int_0^L N_1N_2\,dx\\ \int_0^L N_2N_1\,dx & \int_0^L N_2N_2\,dx\end{bmatrix}\frac{\partial}{\partial t}\begin{bmatrix}T_1\\T_2\end{bmatrix}\\ &\qquad + \bar{\kappa}\begin{bmatrix}\int_0^L \frac{\partial N_1}{\partial x}\frac{\partial N_1}{\partial x}\,dx & \int_0^L \frac{\partial N_1}{\partial x}\frac{\partial N_2}{\partial x}\,dx\\ \int_0^L \frac{\partial N_2}{\partial x}\frac{\partial N_1}{\partial x}\,dx & \int_0^L \frac{\partial N_2}{\partial x}\frac{\partial N_2}{\partial x}\,dx\end{bmatrix}\begin{bmatrix}T_1\\T_2\end{bmatrix}\\ &\qquad\qquad - \bar{s}\begin{bmatrix}\int_0^L N_1\,dx\\ \int_0^L N_2\,dx\end{bmatrix} - \begin{bmatrix}N_1 q_A\big|_{x_A}\\ N_2 q_B\big|_{x_B}\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}.\end{aligned} \tag{11}$$

Note that one now has two equations for the two unknowns $T_1$ and $T_2$ as desired. On evaluating the integrals (using $\mathbf{N}$ defined in Equation (7)), Equation (11) becomes

$$\begin{bmatrix} \frac{L}{3} & \frac{L}{6} \\ \frac{L}{6} & \frac{L}{3} \end{bmatrix} \frac{\partial}{\partial t} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} + \bar{\kappa} \begin{bmatrix} \frac{1}{L} & -\frac{1}{L} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} - \bar{s} \begin{bmatrix} \frac{L}{2} \\ \frac{L}{2} \end{bmatrix} - \begin{bmatrix} N_1 q_A \big|_{x_A} \\ N_2 q_B \big|_{x_B} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{12}$$

which can be simplified using matrix notation to

$$\mathbf{MM}\left(\frac{\partial \mathbf{T}}{\partial t}\right) + \mathbf{KMT} = \mathbf{F}, \tag{13}$$

where

$$\mathbf{MM} = \begin{bmatrix} \dfrac{L}{3} & \dfrac{L}{6} \\ \dfrac{L}{6} & \dfrac{L}{3} \end{bmatrix}, \tag{14}$$

$$\mathbf{KM} = \bar{\kappa} \begin{bmatrix} \dfrac{1}{L} & -\dfrac{1}{L} \\ -\dfrac{1}{L} & \dfrac{1}{L} \end{bmatrix}, \tag{15}$$

$$\mathbf{F} = \bar{s} \begin{bmatrix} \dfrac{L}{2} \\ \dfrac{L}{2} \end{bmatrix} + \begin{bmatrix} N_1 q_A \Big|_{x_A} \\ N_2 q_B \Big|_{x_B} \end{bmatrix}, \tag{16}$$

and

$$\mathbf{T} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}. \tag{17}$$

The next step we perform is the discretization of the time derivative, which is usually achieved using a finite difference approximation. Assuming an implicit time discretization, one can rewrite Equation (13) as

$$\mathbf{MM}\left(\frac{\mathbf{T}^{n+1} - \mathbf{T}^{n}}{\Delta t}\right) + \mathbf{KMT}^{n+1} = \mathbf{F}, \tag{18}$$

where $\mathbf{T}^{n+1}$ is the future temperature at the nodes (i.e., the unknowns) and $\mathbf{T}^{n}$ is the vector of old (i.e., known) temperatures. Rearranging, one can write this as

$$\left(\frac{1}{\Delta t}\mathbf{MM} + \mathbf{KM}\right)\mathbf{T}^{n+1} = \frac{1}{\Delta t}\mathbf{MMT}^{n} + \mathbf{F}, \tag{19}$$

or more compactly as

$$\mathbf{KLT}^{n+1} = \mathbf{KRT}^{n} + \mathbf{F}, \tag{20}$$

where

$$\mathbf{KL} = \begin{bmatrix} \dfrac{L}{3\Delta t} + \dfrac{\bar{\kappa}}{L} & \dfrac{L}{6\Delta t} - \dfrac{\bar{\kappa}}{L} \\ \dfrac{L}{6\Delta t} - \dfrac{\bar{\kappa}}{L} & \dfrac{L}{3\Delta t} + \dfrac{\bar{\kappa}}{L} \end{bmatrix} \tag{21}$$

and

$$\mathbf{KR} = \begin{bmatrix} \dfrac{L}{3\Delta t} & \dfrac{L}{6\Delta t} \\ \dfrac{L}{6\Delta t} & \dfrac{L}{3\Delta t} \end{bmatrix} \tag{22}$$

and the **F** vector is

$$\mathbf{F} = \bar{s}L \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}.$$

In Equation (20), everything appearing on the right hand side is known (and it combines to form a vector). The matrix **KL** is referred to as the element stiffness matrix whereas **T** is the unknown element vector (and the subscript $n+1$ has been dropped for clarity). For the purposes of following discussions we introduce the following notation:

$$\mathbf{KL} = \begin{bmatrix} KL_{11} & KL_{12} \\ KL_{21} & KL_{22} \end{bmatrix}.$$

Thus, for example, the term $KL_{11}$ has the value $\frac{L}{3\Delta t} + \frac{\bar{\kappa}}{L}$. Similar notation is assumed for **KR**.

Remember that so far we have only carried out discretization for a single element, whereas we generally want to divide the solution domain into many elements so as to obtain an accurate solution. Accordingly, let us consider a small one-dimensional mesh, consisting of four elements (once you get the idea you can easily consider more elements). This situation is depicted in Figure 2c. Now, instead of having just two unknowns, we have five, related to the five nodes in the mesh. One now generates a global matrix equation by summing node-by-node the matrix equation derived for a single element (i.e., equation 17). Thus, for example, note that whereas node 1 contains a contribution only from element 1, node 2 has contributions from both elements 1 and 2 (Figure 2c). Performing this process (using the notation introduced above and assuming that each element matrix is the same) leads to

$$\begin{bmatrix} KL_{11} & KL_{12} & 0 & 0 & 0 \\ KL_{21} & KL_{22}+KL_{11} & KL_{12} & 0 & 0 \\ 0 & KL_{21} & KL_{22}+KL_{11} & KL_{12} & 0 \\ 0 & 0 & KL_{21} & KL_{22}+KL_{11} & KL_{12} \\ 0 & 0 & 0 & KL_{21} & KL_{22} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}^{n+1}$$
$$= \begin{bmatrix} KR_{11} & KR_{12} & 0 & 0 & 0 \\ KR_{21} & KR_{22}+KR_{11} & KR_{12} & 0 & 0 \\ 0 & KR_{21} & KR_{22}+KR_{11} & KR_{12} & 0 \\ 0 & 0 & KR_{21} & KR_{22}+KR_{11} & KR_{12} \\ 0 & 0 & 0 & KR_{21} & KR_{22} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix}^{n}$$
$$+ sL \begin{bmatrix} \frac{1}{2} \\ 1 \\ 1 \\ 1 \\ \frac{1}{2} \end{bmatrix}, \quad (23)$$

which, using matrix notation, becomes

$$\mathbf{KL}_G \mathbf{T}^{n+1} = \mathbf{KR}_G \mathbf{T}^n + \mathbf{F}_G, \quad (24)$$

where the subscript $G$ indicates that the matrices and vectors refer to the entire "global" or assembled problem and not simply to a single element. Note that the matrices $\mathbf{KL}_G$ and $\mathbf{KR}_G$ are symmetric which is an important (though not a necessary) property when it comes to solving the system of equations. Finally, note that most non-zero terms are clustered near the main diagonal (called diagonal dominance) which also helps when the equations are solved. Equation (24) can also be written in the form

$$\mathbf{KL}_G \mathbf{T}^{n+1} = \mathbf{b}, \tag{25}$$

which is the classic form for a system of linear equations. In this last expression, $\mathbf{KL}_G$ is referred to as the stiffness (or coefficient) matrix (and is known), $\mathbf{b}$ is referred to as the "right hand side vector" or "load vector" and $\mathbf{T}^{n+1}$ is the unknown "solution vector" or "reaction vector".

The final step before one can solve the linear system of equations represented by Equation (25) is to impose boundary conditions. There are several possible choices; fixed temperature, fixed temperature gradient, or some combination of the two. Fixed temperature boundaries may be implemented by performing the following steps: (1) zeroing the entries in the relevant equation (2) placing a 1 on the diagonal entry of the stiffness matrix and (3) setting the value at the correct position of the right hand side vector equal to the desired value. Alternatively, if one wishes to implement zero-flux boundary conditions one does not have to do anything explicitly (i.e., it is the default boundary condition created when one ignored the boundary terms in Equation (10)). Non-zero-flux boundary conditions are slightly more complex and will be considered in a future session.

You are now ready to compute the solution to Equation (25). The basic steps that need to be incorporated into a computer program can be summarized as follows:

1. Define all physical (e.g., diffusivity, source term, length of spatial domain) and numerical (e.g., number of elements and nodes) parameters.

2. Define the spatial coordinate $x$ and the time domain where you want to obtain the solution

3. Within an element loop, define the element matrices $\mathbf{MM}$ and $\mathbf{KM}$ and the element load vector $\mathbf{F}$ (see Equations (14) to (16)). Use these to compute $\mathbf{KL}$ and $\mathbf{KR}$ (see Equations (21) & (22)). Sum these matrices node-by-node to form the global matrices $\mathbf{KL}_G$ and $\mathbf{KR}_G$ and the global vector $\mathbf{F}_G$ (see Equations (23)). If the element properties do not depend on time these global matrices only need to be calculated once and can be saved for later use.

4. Within a time loop, perform the operations on the right hand side of Equation (25) (i.e., firstly multiply $\mathbf{KR}_G$ with the old temperature vector $\mathbf{T}^n$ and then add the resulting vector to $\mathbf{F}_G$) to form the right-hand-side vector $\mathbf{b}$.

5. Apply boundary conditions.

6. Solve Equation (25) for the new temperature, and then continue to the next time step.

## 5 Exercises

1. Read the text above and try to understand it.
2. We solved the system for second order derivatives. Can we employ linear shape functions if we want to solve, for example, the bending equation for a thin plate, which has fourth order derivatives? If no, which shape functions should be employed?