# *Introduction to Finite Element Modelling in Geosciences:* **Time For Programming**

D. A. May & M. Frehner,
ETH Zürich

July 7, 2014

## 1 Overview

The purpose of this session is to learn how to program the finite element method (FEM) in one dimension using MATLAB. To illustrate this we use the diffusion equation (discretised in the previous session) which is governed by the following element equations

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

where

$$\mathbf{MM} = \begin{bmatrix} \frac{\Delta x}{3} & \frac{\Delta x}{6} \\ \frac{\Delta x}{6} & \frac{\Delta x}{3} \end{bmatrix}, \tag{2}$$

$$\mathbf{KM} = \kappa \begin{bmatrix} \frac{1}{\Delta x} & \frac{-1}{\Delta x} \\ \frac{-1}{\Delta x} & \frac{1}{\Delta x} \end{bmatrix}, \tag{3}$$

$$\mathbf{F} = s \begin{bmatrix} \frac{\Delta x}{2} \\ \frac{\Delta x}{2} \end{bmatrix} + \begin{bmatrix} N_1 q_A \big|_{x_A} \\ N_2 q_B \big|_{x_B} \end{bmatrix}, \tag{4}$$

and

$$\mathbf{T} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}. \tag{5}$$

The superscripts $n$ and $n+1$ refer to the old (known) and future (unknown) temperatures, respectively. These equations can be written more compactly as

$$\mathbf{KL}\mathbf{T}^{n+1} = \mathbf{KR}\mathbf{T}^{n} + \mathbf{F}, \tag{6}$$

where the element stiffness matrix $\mathbf{KL}$ is

$$\mathbf{KL} = \frac{\mathbf{MM}}{\Delta t} + \mathbf{KM} \tag{7}$$

and the element right hand side matrix $\mathbf{KR}$ is

$$\mathbf{KR} = \frac{\mathbf{MM}}{\Delta t}. \tag{8}$$

Remember that for a 2-node element, these relations form a system of two equations for the two unknown node temperatures $T_1$ and $T_2$. It is important to distinguish these element equations from the global equations formed by summing the element equations node-by-node within a finite element mesh, denoted as

$$\mathbf{KL}_G\mathbf{T}^{n+1} = \mathbf{KR}_G\mathbf{T}^n + \mathbf{F}_G = \mathbf{b}, \tag{9}$$

where the subscript $G$ indicates that the matrices and vectors are global matrices. The size of the matrices $\mathbf{KL}_G$ and $\mathbf{KR}_G$ are $nn \times nn$ (where $nn$ is the number of nodes in the finite element mesh) whereas the matrices **MM** and **KM** have dimensions $2 \times 2$.

# 2 FE Procedure

The basic steps that must be performed in order to solve these equations within a computer program can be summarized as follows:

1. Define all physical parameters (e.g., diffusivity, source term, length of spatial domain) and numerical parameters (e.g., number of elements and nodes).
2. Define the spatial coordinate $x$ and the time domain where you want to obtain the solution.
3. Define the relationship between the element node numbers and the global node numbers.
4. Define boundary node indices and the values of the potential (e.g., temperature) at the boundary nodes.
5. Initialise the global matrices $\mathbf{KL}_G$, $\mathbf{KR}_G$ and $\mathbf{F}_G$ so that their dimensions are defined and that the matrices are filled with zeros.
6. Within an element loop, define the element matrices **MM** and **KM** and the element load vector **F** (see Equations (2), (3) and (4)). Use these to compute **KL** and **KR**. Sum these matrices (and **F**) node-by-node to form the global stiffness matrices $\mathbf{KL}_G$ and $\mathbf{KR}_G$ and the global vector $\mathbf{F}_G$ (see Equation (9)). 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.
7. Within a time loop, perform the operations on the right hand side of Equation (9) (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 $\mathbf{b}$).
8. Apply boundary conditions.
9. Solve equation 9 for the new temperature, and then continue to the next time step.

We will now describe each of these steps in more detail. In the following, we will indicate where MATLAB commands are used via the syntax

```
>alpha = 1.0;
```

**[1] Parameter definition**

This step requires little further explanation. All physical (e.g., $\kappa$, domain length, etc.) and numerical (number of nodes, timestep) parameters must be defined in this section using commands of the form

```
>kappa = 10.0;
```

**[2] Define spatial and time domain**

Similarly this step requires little further explanation. The spatial coordinate vector $x$ can be defined in MATLAB using the command

```
>x = [ 0 : dx : lx ];
```

where lx is the length of the spatial domain and dx is the element length. One can create a similar vector for time.

**[3] Local to global mapping**

In order to sum the element equations node-by-node to form the global matrices one must define the relationship between local node numbers and global node numbers. We achieve this by creating a matrix called $g\_num$ which has the following form for a 5-node (4 element) finite element mesh (comprising four 2-node elements):

$$g\_num = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \end{bmatrix}$$

which in MATLAB is defined via the command,

```
>g_num = [ 1, 2, 3, 4 ; 2, 3, 4, 5 ];
```

The matrix $g\_num$ has the dimensions $nod \times nels$ where $nod$ is the number of nodes per element and $nels$ is the total number of elements in the mesh (both of which must be defined in Section [1] above). Thus, for example, the two local node numbers 1 and 2 in element 3 correspond to the global node numbers 3 and 4 respectively (i.e., $g\_num(1,3) = 3$ and $g\_num(2,3) = 4$).

**[4] Define boundary conditions**

One must define to which global nodes boundary conditions are to be applied (and what the boundary value is). This can be done with the following commands:

```
>bcdof = [ 1, nn ];
```

and

```
>bcval = [ 0.0, 0.0 ];
```

The array *bcdof* specifies which which nodes, namely the first (1) and last ($nn$), in the mesh are to be constrained. The array *bcval* defines the value of the temperature constraint we wish to apply at each boundary node.

**[5] Initialisation**

One must initialise the global matrices $\mathbf{KL}_G$, $\mathbf{KR}_G$ and global vector $\mathbf{F}_G$ using commands of the form

```
>KLG = zeros( nn, nn );
```

and

```
>FG = zeros( nn, 1 );
```

where $nn$ is the number of equations in the finite element mesh (which also equals the total number of nodes in this problem).

**[6] Element matrix assembly**

The element matrices **MM** and **KM** can be formed using commands of the form

```
>MM = [ dx/3.0, dx/6.0 ; dx/6.0, dx/3.0 ];
```

whereas the vector **F** can be formed with the command

```
>F = s*[ dx/2.0 ; dx/2.0 ];
```

Once these are defined, the matrices **KL** and **KR** can be computed according to Equations (7) and (8). The next task is, within a loop over all elements, to sum the element matrices node-by-node to form the global matrices. This step is achieved using the matrix *g_num* just described. For example, in order to form the matrix $\mathbf{KL}_G$ one can use the following commands:

```
>KLG(g_num(:,iel),g_num(:,iel)) = KLG(g_num(:,iel),g_num(:,iel)) + KL;
```

whereas to form the vector $\mathbf{F}_G$ one can use

```
>FG(g_num(:, iel)) = FG(g_num(:, iel)) + F;
```

The variable *iel* is the current element number which varies within the loop from 1 to *nels*.

**[7] Form right-hand-side vector**

Within a time loop one can now form the global right-hand-side vector $b$ with the command

```
>b = KRG * T + FG;
```

where $T$ is the vector (with dimensions $nn \times 1$) of old temperatures and the other vectors and matrices are described above.

**[8] Implement boundary conditions**

The boundary conditions can be imposed using the following sequence of commands:

```
> for i=1:length(bcdof)
>    KLG( bcdof(i), : ) = 0.0;
>    KLG( bcdof(i),bcdof(i)) = 1.0;
>    b( bcdof(i) ) = bcval(i);
> end
```

We make a loop over all nodes that have a (fixed or Dirichlet) boundary condition. First, we put zeros in the row where the boundary conditions should be imposed. These row indices are listed in the array $bcdof$. After that we put a value 1.0 on the diagonal of that row, and put the corresponding value of that boundary condition in the right-hand-side vector.

**[9] Solution**

The final step which must be carried out is to solve the system of equations (i.e., Equation (9)) simultaneously, which can be performed using the Matlab command

```
>T = KLG \ b;
```

This operation will compute the solution vector $T$, which contains the temperature within the domain at the $n+1$ timestep. A new time step can be begun by repeating Steps 7,8,9.

# 3 Exercises

1. Write a code that can solve the 1D diffusion problem with the finite element method. For simplicity we will assume that the boundary integral term is zero. Note that this assumption implies that we are solving the diffusion with zero flux boundary conditions.

2. Modify the code, to take into account source terms, variable grid spacing, and variable thermal diffusivity (assume properties to be constant within one element).

3. A simple (time-dependent) analytical solution for the diffusion equation exists for the case when the forcing term $s = 0$, and the initial temperature distribution is given by

$$T(x, t = 0) = T_{max} \exp\left[-\frac{x^2}{\sigma^2}\right].$$

Here $\sigma$ is the half-width of the distributions and $T_{max}$ is the maximum amplitude of the temperature perturbation at $x = 0$. The solution for this problem is given by

$$T(x, t) = \frac{T_{max}}{\sqrt{1 + 4t\kappa/\sigma^2}} \exp\left[\frac{-x^2}{\sigma^2 + 4t\kappa}\right].$$

This solution is derived assuming that $T \to 0$ at $x = \pm\infty$. Program the analytical solution and compare the analytical solution with the numerical solution with the same initial condition. Use $T_{max} = 100$, $\sigma = 1$ and $-5 \leq x \leq 5$.

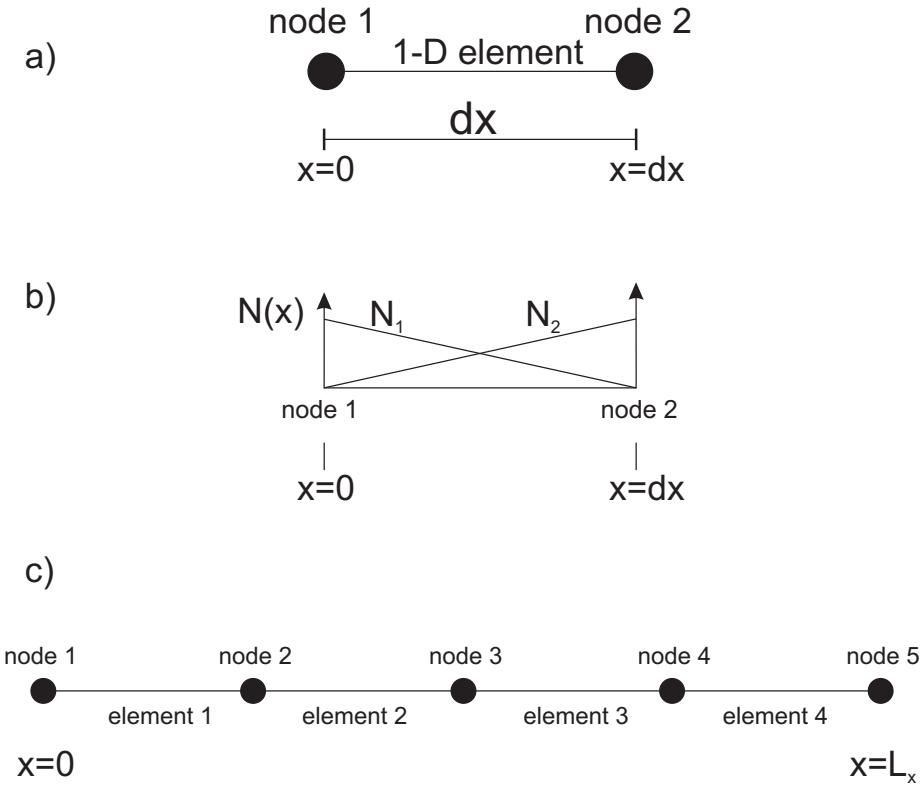


Figure 1: a) a 2-node, one dimensional finite element with b) linear shape functions, and c) a small one dimensional mesh consisting of four elements and a total of five nodes. See text for discussion.