Aalto University
Department of Radio
Science and Engineering

Introduction to Finite Element Method

Pasi Ylä-Oijala

April 24 and 28, 2014

INTRODUCTION

General Issues and Content

- Short introduction to finite element method (FEM).
- Based on course Numerical Methods in Electromagnetics, given at Department of Radio Science and Engineering, Aalto University.
- This course consists of lectures and (Matlab) exercises.
- Contact and questions: pasi.yla-oijala(at)aalto.fi.

Content:

1. Introduction
2. General recipe of FEM
3. 1D scalar FEM
4. 2D (3D) scalar FEM
5. Example – Capacitance computation in electrostatics

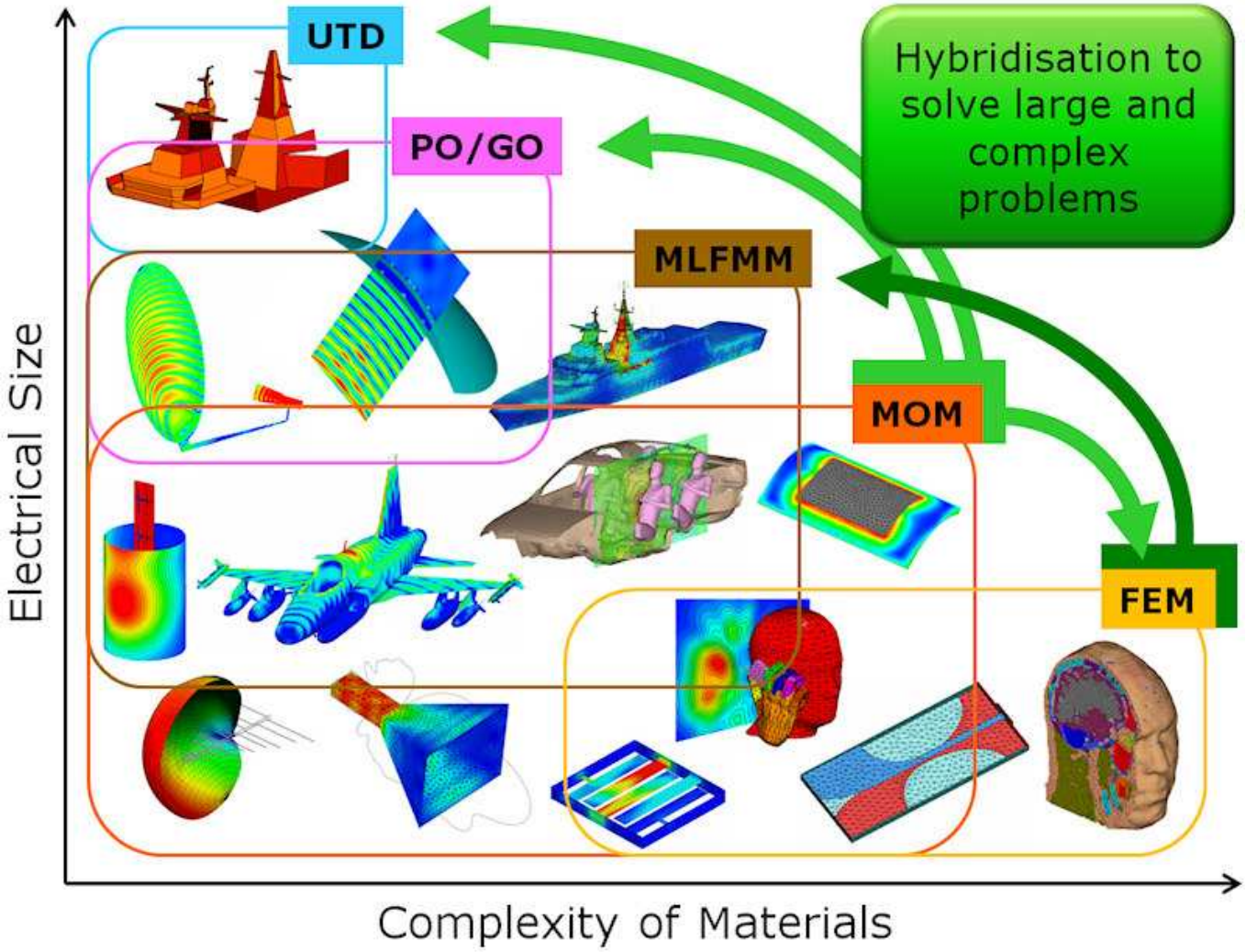
INTRODUCTION

CSE & FEM

- During the last few decades the importance of numerical simulations has significantly increased. Reasons for this are the ever increasing capacity of computers and the development of more and more sophisticated numerical methods and algorithms.
- The outcome of this is the recent arising of the Computational Science and Engineering, not only as a secondary “cost saving field”, but as an independent scientific domain.
- Finite element method (FEM) is one of the most versatile and widely used numerical techniques for finding approximate solutions of boundary value problems arising from partial differential equation-based mathematical modeling of physical phenomena.
- FEM is applied e.g., in structural analysis, fluid dynamics, solid mechanics, acoustics, and electromagnetics.

INTRODUCTION

Applications and Numerical Methods in Electromagnetics



The idea in FEM is to convert an infinite dimensional continuous linear operator equation into a finite dimensional discrete matrix equation

$$\boxed{\mathcal{D}[u] = v \qquad \Longrightarrow \qquad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}.} \tag{1}$$

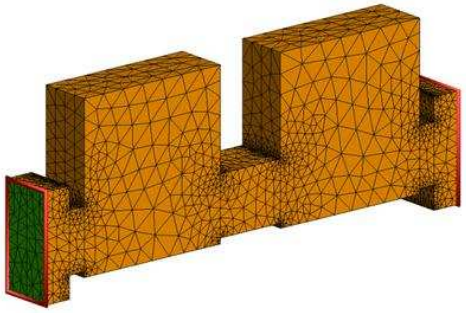
Generally, FEM can be understood to consist of the following steps:

1. Geometrical modeling (solid modeling, mesh generation).
2. Physical and mathematical modeling (PDE, boundary conditions, material parameters, weak formulation, function spaces).
3. Numerical modeling (discrete FE spaces).
4. Implementation (computer programming).
5. Computations and simulations (matrix assembly, solution of linear system).
6. Post-processing (visualization, parameter computation).

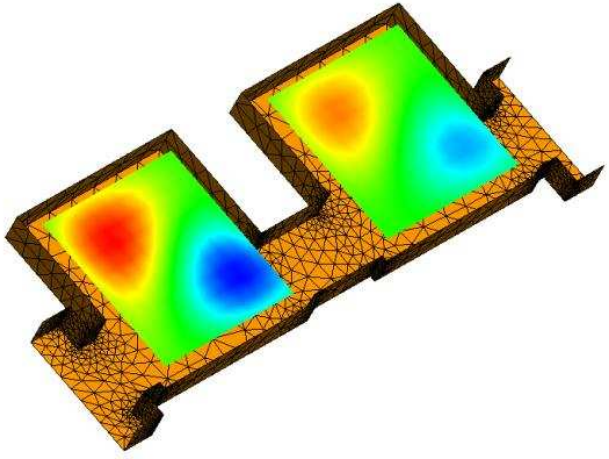
INTRODUCTION

From Design Through Mathematical Formulation and Programming to Simulations

Design:



Solution:



Mathematical Formulation:

For a given $\boldsymbol{J} \in \mathcal{H}_{Div}(\Omega)$ find $\boldsymbol{E} \in \mathcal{H}_{Curl}(\Omega)$ satisfying

$$\left\langle \nabla \times \boldsymbol{w}\,,\ \frac{1}{\mu_r}\nabla \times \boldsymbol{E} \right\rangle - k_0^2 \left\langle \boldsymbol{w}\,,\ \varepsilon_r \boldsymbol{E} \right\rangle = i\omega\mu_0 \left\langle \boldsymbol{w}\,,\ \boldsymbol{J} \right\rangle, \quad \text{in } \Omega,$$

for all $\boldsymbol{w} \in \mathcal{H}_{Curl}(\Omega)$ and $\gamma_t \boldsymbol{E} = 0$ on Γ.

C++ Code:

```
const UINT N = A.get_M();
const UINT nele = mesh->get_nele();
sparse_mat<bool> E(nele,N);
for (UINT ele = 0; ele != nele; ++ele)
E.register_element(ele, 3);
E.register_complete();
for (UINT ele = 0; ele != nele; ++ele)
{ const auto &row = rwgs(ele);
for (unsigned int j = 0; j != row.get_M();
++j)
E.push_in(ele, abs(row(j)) - 1); }
E.finalize();
const sparse_mat<bool> *const ET2 =
E.transpose();
const sparse_mat<bool> &ET = *ET2;
dynamic_vec<bool> flags(N);
```

Aalto University
Department of Radio
Science and Engineering

INTRODUCTION

Notations

- Points in $\mathbf{R}^n$, $n = 1, 2, 3$, are denoted by $\boldsymbol{r} = x$, $\boldsymbol{r} = (x, y)$, and $\boldsymbol{r} = (x, y, z)$.
- Unit vectors in rectangular coordinate system are $\boldsymbol{e}_x$, $\boldsymbol{e}_y$ and $\boldsymbol{e}_z$.
- Vectors are denoted by boldface and vector fields by capitals

$$\boldsymbol{F}(\boldsymbol{r}) = F_x(\boldsymbol{r})\boldsymbol{e}_x + F_y(\boldsymbol{r})\boldsymbol{e}_y + F_z(\boldsymbol{r})\boldsymbol{e}_z. \tag{2}$$

- Gradient of a scalar function is

$$\nabla f(\boldsymbol{r}) = \frac{\partial f(\boldsymbol{r})}{\partial x}\boldsymbol{e}_x + \frac{\partial f(\boldsymbol{r})}{\partial y}\boldsymbol{e}_y + \frac{\partial f(\boldsymbol{r})}{\partial z}\boldsymbol{e}_z. \tag{3}$$

- Divergence of a vector function is

$$\nabla \cdot \boldsymbol{F}(\boldsymbol{r}) = \frac{\partial F_x(\boldsymbol{r})}{\partial x} + \frac{\partial F_y(\boldsymbol{r})}{\partial y} + \frac{\partial F_z(\boldsymbol{r})}{\partial z}. \tag{4}$$

- Curl of a vector function is

$$\nabla \times \boldsymbol{F} = \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\boldsymbol{e}_x + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\boldsymbol{e}_y + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\boldsymbol{e}_z. \tag{5}$$

- Laplacian of a scalar function is

$$\Delta f(\boldsymbol{r}) = \nabla \cdot \nabla f(\boldsymbol{r}) = \frac{\partial^2 f(\boldsymbol{r})}{\partial_{x^2}} + \frac{\partial^2 f(\boldsymbol{r})}{\partial_{y^2}} + \frac{\partial^2 f(\boldsymbol{r})}{\partial_{z^2}} \; (= \nabla^2 f(\boldsymbol{r})). \tag{6}$$

INTRODUCTION

Gauss Formulas and Inner Product

- Let Ω be a simply connect closed domain in $\mathbf{R}^n$ with sufficient smooth boundary Γ, and unit normal vector $\boldsymbol{n}$. Then

$$\int_\Omega \nabla u \, d\Omega = \int_\Gamma \boldsymbol{n} \, u \, d\Gamma \tag{7}$$

$$\int_\Omega \nabla \cdot \boldsymbol{F} \, d\Omega = \int_\Gamma \boldsymbol{n} \cdot \boldsymbol{F} \, d\Gamma \tag{8}$$

$$\int_\Omega \nabla \times \boldsymbol{F} \, d\Omega = \int_\Gamma \boldsymbol{n} \times \boldsymbol{F} \, d\Gamma, \tag{9}$$

- $L^2(\Omega)$ symmetric (inner) product (without complex conjugate!)

$$\langle u \, , \, v \rangle = \int_\Omega u(\boldsymbol{r}) \, v(\boldsymbol{r}) \, d\Omega \quad \text{or} \quad \langle \boldsymbol{u} \, , \, \boldsymbol{v} \rangle = \int_\Omega \boldsymbol{u}(\boldsymbol{r}) \cdot \boldsymbol{v}(\boldsymbol{r}) \, d\Omega. \tag{10}$$

- In addition, denote

$$\langle u \, , \, v \rangle_\Gamma = \int_\Gamma u(\boldsymbol{r}) \, v(\boldsymbol{r}) \, d\Gamma \quad \text{or} \quad \langle \boldsymbol{u} \, , \, \boldsymbol{v} \rangle_\Gamma = \int_\Gamma \boldsymbol{u}(\boldsymbol{r}) \cdot \boldsymbol{v}(\boldsymbol{r}) \, d\Gamma. \tag{11}$$

GENERAL RECIPE

Boundary Value Problem

- Let Ω be an open bounded domain in $\mathbf{R}^n$ with sufficiently smooth boundary Γ.
- Consider the following partial differential equation

$$-\nabla \cdot \big(\alpha \nabla u(\boldsymbol{r})\big) + \beta u(\boldsymbol{r}) = f(\boldsymbol{r}), \qquad \boldsymbol{r} \in \Omega. \tag{12}$$

- Divide Γ into two parts $\Gamma = \Gamma_D \cup \Gamma_N$ so that $\Gamma_D \cap \Gamma_N = \emptyset$.
- Consider two types of boundary conditions:

$$u|_{\Gamma_D} = g^D, \qquad \text{Dirichlet (“essential”)} \tag{13}$$

$$\alpha \frac{\partial u}{\partial n}\bigg|_{\Gamma_N} = g^N, \qquad \text{Neumann (“natural”)} \tag{14}$$

  Here u is a unknown function, α and β are given coefficients, f, g^D and g^N are known functions.
- Solutions of this boundary value problem (BVP) are called strong solutions – *“equality holds at every point”*.
- In FEM we, however, consider: weak solutions – *“equality holds in weighted average sense”*.

GENERAL RECIPE

Weak Formulation

- Define Sobolev spaces

$$H^0(\Omega) := \left\{u \in L^2(\Omega)\right\}, \tag{15}$$

$$H^1(\Omega) := \left\{u \in L^2(\Omega) \text{ and } \nabla u \in (L^2(\Omega))^3\right\}, \tag{16}$$

the Dirichlet and Neumann trace spaces

$$\gamma_D u := u|_{\Gamma_D} \quad : \quad H^1(\Omega) \mapsto H^{1/2}(\Gamma_D), \tag{17}$$

$$\gamma_N u := \partial u/\partial n|_{\Gamma_N} \quad : \quad H^1(\Omega) \mapsto H^{-1/2}(\Gamma_N), \tag{18}$$

and the dual space of $H^1(\Omega)$, $(H^1(\Omega))' = H^{-1}(\Omega)$.

- Weak formulation of BVP (12) - (14) reads:
  For given $f \in (H^1(\Omega))', g^D \in H^{1/2}(\Gamma)$ and $g^N \in H^{-1/2}(\Gamma)$, find such $u \in H^1(\Omega)$, $\gamma_D u = g^D$, that

$$\langle \nabla w\,,\, \alpha \nabla u\rangle + \langle w\,,\, \beta u\rangle = -\left\langle w\,,\, g^N\right\rangle_{\Gamma_N} + \langle w\,,\, f\rangle\,, \tag{19}$$

holds for all $w \in H^1(\Omega)$, $\gamma_D w = 0$.

GENERAL RECIPE

Finite Element Spaces

- The next step in FEM is to find a suitable set of *finite elements* (FE).
- Generally, FE is a triple

$$(T, P_T, \Sigma_T), \tag{20}$$

  where
  - T is a geometric domain ("an element")
  - P_T is a space of functions (polynomials) on T ("an approximation")
  - Σ_T is a set of linear functionals on P_T (degrees of freedom, dof).
- An union of all (T, P_T, Σ_T) is called a (global) FE space.
- Important properties of a FE (space):
  - FE (T, P_T, Σ_T) is said to be *unisolvent* if specifying a value for each dof in Σ_T uniquely determines a function in P_T.
  - FE (T, P_T, Σ_T) is said to be *H conforming* if the corresponding FE space is a subspace of a function space H.
- In the following we shall use the following important result: "The FE space of piecewise continuous polynomials is H^1 conforming and unisolvent".

GENERAL RECIPE

Discrete Problem – Basis Functions

- Assume that we have a discrete FE space U^h that is H^1 conforming and unisolvent.
- Discrete problem can now be formulated as: Find such $u^h \in U^h$, $u^h|_{\gamma_D} = g^D$, that

$$\left\langle \nabla w^h \, , \, \alpha \nabla u^h \right\rangle + \left\langle w^h \, , \, \beta u^h \right\rangle = - \left\langle w^h \, , \, g^N \right\rangle_{\Gamma_N} + \left\langle w^h \, , \, f \right\rangle , \qquad (21)$$

  holds for all $w^h \in U^h$, $w^h|_{\gamma_D} = 0$.
- In practice, u is approximated with a linear combination of basis functions $u_1, \ldots, u_N \in U^h$

$$u(\boldsymbol{r}) \approx \sum_{n=1}^{N} c_n \, u_n(\boldsymbol{r}) = u^h(\boldsymbol{r}). \qquad (22)$$

- Then choose a set of testing functions $w_1, \ldots, w_M \in U^h$. Usually we have $w_m = u_m$ for all m (excluding the testing functions on Γ_D) and $M = N$.

GENERAL RECIPE

Discrete Problem – Matrix Equation

- This gives a set of linear equations, i.e., a matrix equation

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}, \tag{23}$$

where $\boldsymbol{A}$ is $N \times N$ system matrix, $\boldsymbol{x} = [c_1, \ldots, c_N]^T$ is the coefficient vector to be found, and $\boldsymbol{b}$ is $N \times 1$ source vector.

- Elements of $\boldsymbol{A}$ and $\boldsymbol{b}$ are given by

$$\begin{aligned} A_{mn} &= \langle \nabla u_m \,,\, \alpha \nabla u_n \rangle + \langle u_m \,,\, \beta u_n \rangle \,, \\ &= \int_{\Omega_{mn}} \alpha(\boldsymbol{r}) \nabla u_m(\boldsymbol{r}) \cdot \nabla u_n(\boldsymbol{r})\, d\Omega + \int_{\Omega_{mn}} \beta(\boldsymbol{r}) u_m(\boldsymbol{r}) \nabla u_n(\boldsymbol{r})\, d\Omega, \qquad (24) \\ b_m &= \langle u_m \,,\, f \rangle - < u_m \,,\, g^N >_{\Gamma_N} \\ &= \int_{\mathrm{spt}(u_m)} u_m(\boldsymbol{r}) f(\boldsymbol{r})\, d\Omega - \int_{\mathrm{spt}(u_m) \cap \Gamma_N} u_m(\boldsymbol{r}) g^N(\boldsymbol{r})\, d\Gamma, \qquad (25) \end{aligned}$$

for all $n, m = 1, \ldots, N$. Here $\Omega_{mn} = \mathrm{spt}(u_m) \cap \mathrm{spt}(u_n)$,

- Matrix $\boldsymbol{A}$ is sparse (most elements are 0) and symmetric.

- Next we consider more details of FEM implementations in 1D.
- Consider the following second order differential equation in 1D

$$-\frac{d}{dx}\left(\alpha\frac{du(x)}{dx}\right)+\beta u(x)=f(x),\quad x\in[a,b], \tag{26}$$

where u is the unknown function to be found, α and β are known coefficients and f is a given function.

- Boundary conditions at $x=a$ and $x=b$ are either

$$u(x)=g^D(x),\qquad \text{Dirichlet}, \tag{27}$$

or

$$\alpha\frac{du}{dx}(x)=g^N(x),\qquad \text{Neumann}. \tag{28}$$

- With Dirichlet boundary condition the weak formulation reads

$$\left\langle \frac{d}{dx}w\,,\ \alpha\frac{du}{dx}\right\rangle + \langle w\,,\ \beta u\rangle = \langle w\,,\ f\rangle\,, \quad u(a) = g^D(a),\ u(b) = g^D(b). \tag{29}$$

  Here the testing function w vanishes at the end points of the interval $[a, b]$, i.e., $w(a) = 0$ and $w(b) = 0$.

- With Neumann boundary condition the weak formulation becomes

$$\left\langle \frac{d}{dx}w\,,\ \alpha\frac{du}{dx}\right\rangle + \langle w\,,\ \beta u\rangle = \langle w\,,\ f\rangle - w(b)\,g^N(b) + w(a)\,g^N(a). \tag{30}$$

- In the following, the weak formulation is first discretized using testing functions that do not vanish at the end points, and the boundary conditions are later enforced to the discretized matrix equation.

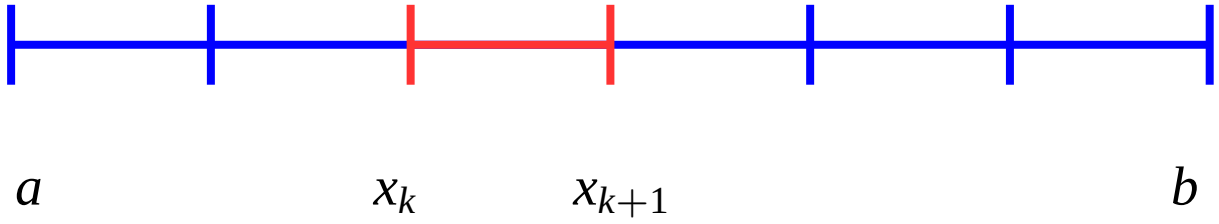


Figure: 1D mesh.

- Divide interval $[a, b]$ into small line segments, called elements, $e_k = [x_k, x_{k+1}]$, $k = 1, 2, \ldots, K$.
- Approximate unknown function u with piece-wise continuous first order polynomials.
- FE (T, P_T, Σ_T) is then given by

$$T = e_k, \quad P_T = P^{(1)}(x), \quad \Sigma_T = [u^h(x_k), u^h(x_{k+1})] \tag{31}$$

  - Geometric element is interval e_k.
  - Approximating functions $P^{(1)}(x)$ are first order polynomials of x on e_k.
  - dof are the values of the approximation u^h at the end points of e_k.

- Function u is approximated as a linear combination of piece-wise linear continuous basis functions u_n

$$u(\boldsymbol{r}) \approx u^h(\boldsymbol{r}) = \sum_{n=1}^{N_N} c_n u_n(\boldsymbol{r}). \tag{32}$$

Here N_N is the number of the nodes of the mesh (points x_k).

- Piece-wise linear functions are defined as

$$u_n(\boldsymbol{r}) = \begin{cases} 1 & \text{if } \boldsymbol{r} = x_n, \\ 0 & \text{if } \boldsymbol{r} = x_m,\ m \neq n, \\ \text{linear} & \text{otherwise.} \end{cases} \tag{33}$$

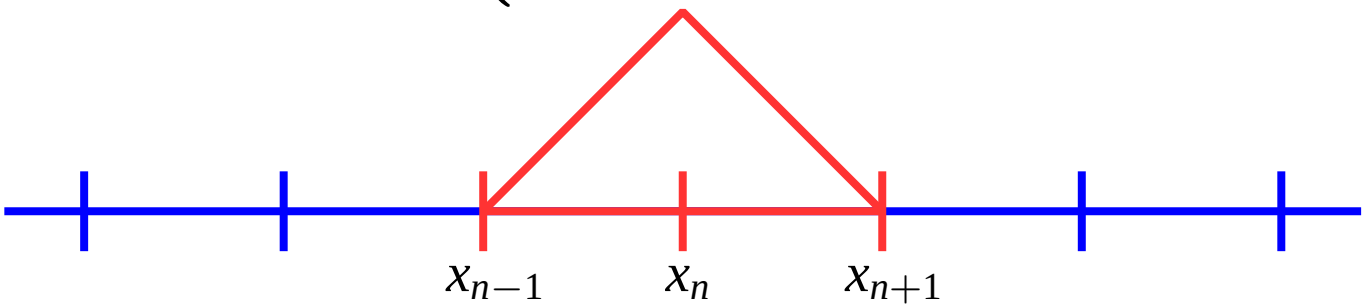


Figure: A linear basis function in 1D.

1D FINITE ELEMENT METHOD

Basis Functions

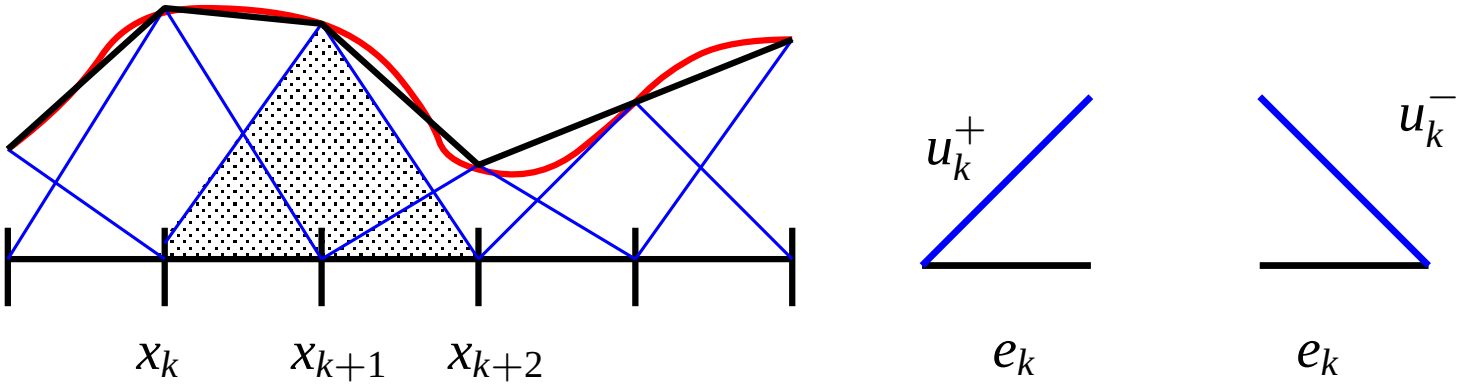


Figure: Linear approximation of a 1D function and two linear shape functions.

- This approximation is unisolvent, the value of u^h at each node x_n uniquely defines the value of the approximation u^h.
- It is conforming in $H^1([a,b])$.
- It gives linear interpolation on each element e_k.
- The total number of dof is the number of nodes (division points).
- Basis functions are defined on two adjacent elements, excluding the basis functions associated to the end points $x = a$ and $x = b$.

- Define two linear functions on an interval e_k

$$u_k^+(x) = \frac{x - x_{k-1}}{L_{k-1}}, \quad u_k^-(x) = \frac{x_{k+1} - x}{L_k}, \tag{34}$$

  and $L_k = x_{k+1} - x_k$ is the length of the interval e_k.
- These functions are restrictions of the the linear basis functions on e_k, i.e., $u_n|_{e_k}$, and are sometimes called linear shape functions.
- Piece-wise linear functions for $n = 2, \ldots, N-1$ are now given by

$$u_n(x) = \begin{cases} u_n^+(x), & \text{if } x \in e_{n-1} = [x_{n-1}, x_n], \\ u_n^-(x), & \text{if } x \in e_n = [x_n, x_{n+1}], \\ 0 & \text{otherwise}, \end{cases} \tag{35}$$

  and for $n = 1$ and $n = N = K + 1$ they are

$$\begin{aligned} u_1(x) &= u_1^-(x), \\ u_N(x) &= u_N^+(x), \end{aligned}$$

- Using the same piece-wise linear functions as the testing functions (Galerkin's method), i.e., $w_m = u_m$, for all m, gives a matrix equation

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}. \tag{36}$$

Elements of $\boldsymbol{A}$ and $\boldsymbol{b}$ are:

1. Dirichlet boundary condition:

$$A_{mn} = \left\langle \frac{du_m}{dx}\,,\, \alpha \frac{du_n}{dx} \right\rangle + \langle u_m\,,\, \beta u_n \rangle\,, \tag{37}$$

$$b_m = \langle u_m\,,\, f \rangle\,. \tag{38}$$

2. Neumann boundary condition:

$$A_{mn} = \left\langle \frac{du_m}{dx}\,,\, \alpha \frac{du_n}{dx} \right\rangle + \langle u_m\,,\, \beta u_n \rangle\,, \tag{39}$$

$$b_m = \langle u_m\,,\, f \rangle - u_m(a)\, g^N(a) + u_m(b)\, g^N(b). \tag{40}$$

- Dirichlet: Testing functions should vanish at the end points and boundary data has to be enforced separately.
- Neumann: Boundary data appears in the weak formulation.

- Elements of matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ (without the boundary conditions) could be evaluated with the following simple looking algorithm:

Zero matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$.
for $m = 1, \ldots, N$ **do**

$$\boldsymbol{b}(m) = \int_{\mathrm{spt}(u_m)} u_m(x)\, f(x)\, dx$$

for $n = 1, \ldots, N$ **do**

$$\boldsymbol{A}(m,n) = \int_{\mathrm{spt}(u_m)\cap\mathrm{spt}(u_n)} \left(\alpha(x) \frac{du_m(x)}{dx} \frac{du_n(x)}{dx} + \beta(x) u_m(x) u_n(x) \right) dx$$

end for
end for

- This algorithm is very inefficient because by looping over the basis and testing functions the integrals will be computed several times.
- Can we do this more efficiently?

- Define two 2×2 "local matrices" and a 2×1 "local vector" of element e_k

$$\texttt{alok1}(i,j) = \int_{e_k} N_i^k(x)\, N_j^k(x)\, dx, \tag{41}$$

$$\texttt{alok2}(i,j) = \int_{e_k} \frac{dN_i^k(x)}{dx}\, \frac{dN_j^k(x)}{dx}\, dx, \tag{42}$$

$$\texttt{blok1}(i) = \int_{e_k} N_i^k(x)\, f(x)\, dx, \tag{43}$$

  $i,j = 1,2$, and $N_i^k = u_n|_{e_k}$, $i = 1,2$, are linear shape functions of element e_k, i.e., the "u^+" and "u^-" functions defined before.
- Assume that coefficients α and β have constant values α_k and β_k in element e_k, and that $\boldsymbol{A}$ and $\boldsymbol{b}$ are initialized with zeros. Then $\boldsymbol{A}$ and vector $\boldsymbol{b}$ can be assembled using the following algorithms:

1D FINITE ELEMENT METHOD

System Matrix Assembly

for $k = 1, \ldots, K$ **do**
  % Compute local matrices `alok1` and `alok2` for element K
  for $i = 1, \ldots, 2$ **do**
    for $j = 1, \ldots, 2$ **do**
      $\texttt{alok1}(i,j) \leftarrow \int_{e_k} N_i^k(x)\, N_j^k(x)\, dx$
      $\texttt{alok2}(i,j) \leftarrow \int_{e_k} \frac{dN_i^k(x)}{dx} \frac{dN_j^k(x)}{dx}\, dx$
    end for
  end for
  % Add local matrices to the global one
  for $i = 1, \ldots, 2$ **do**
    for $j = 1, \ldots, 2$ **do**
      $\boldsymbol{A}(n_i^k, n_j^k) \leftarrow \boldsymbol{A}(n_i^k, n_j^k) + \alpha_k \texttt{alok2}(i,j) + \beta_k \texttt{alok1}(i,j)$
    end for
  end for
end for

- n_i^k, n_i^k and i, j are “global” and “local” indeces of the nodes of e_k.

1D FINITE ELEMENT METHOD

Source Vector Assembly

for $k = 1, \ldots, K$ **do**
  Compute local vector `blok1`
  for $i = 1, \ldots, 2$ **do**
    $\texttt{blok1}(i) \leftarrow \int_{e_k} N_i^k(x) f(x)\, dx$
  end for
  Add local vector to the global one
  for $i = 1, \ldots, 2$ **do**
    $\boldsymbol{b}(n_i^k) \leftarrow \boldsymbol{b}(n_i^k) + \texttt{blok1}(i)$
  end for
end for

- The benefit of these algorithms compared to the previous one is that by looping over the elements of the mesh (once), an integral over each element is computed only once.
- The drawback is that we would need additional bookkeeping of the global and local indeces. This, however, is rather trivial, as will be seen later.

- Neumann boundary data is given by

$$< u_m \, , \, g^N >= u_m(b)g^N(b) - u_m(a)g^N(a). \tag{44}$$

- Because testing function get value one at points $x = a$ and $x = b$

$$< u_m \, , \, g^N >= g^N(b) - g^N(a). \tag{45}$$

- Neumann boundary data (g^N) is added to vector $\boldsymbol{b}$ as

$$\boldsymbol{b}(bn) = \boldsymbol{b}(bn) \pm g^N(x_{bn}), \tag{46}$$

where bn is an index of a boundary node x_{bn}.

- Dirichlet boundary data is given by

$$u^h(a) = g^D(a) \quad \text{and} \quad u^h(b) = g^D(b). \tag{47}$$

- To set the Dirichlet data we need to remove the testing functions associated to the boundary nodes bn (if we have used testing functions defined at the boundary nodes). This agrees to setting rows bn of matrix $\boldsymbol{A}$ and elements bn of $\boldsymbol{b}$ to zero.
- Next value one is set to the diagonal of matrix $\boldsymbol{A}$ and wanted boundary data is set to vector $\boldsymbol{b}$

$$bn \begin{bmatrix} 0 & \dots & 0 & \overset{bn}{1} & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \\ c_{bn} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ g^D(x_{bn}) \\ \\ \end{bmatrix} \tag{48}$$

- The problem is that this leads to a non-symmetric matrix.

- Consider next numerical evaluation of the matrix and vector elements

$$\texttt{alok1}(i,j) = \int_{e_k} N_i^k(x)\, N_j^k(x)\, dx, \tag{49}$$

$$\texttt{alok2}(i,j) = \int_{e_k} \frac{dN_i^k(x)}{dx}\, \frac{dN_j^k(x)}{dx}\, dx, \tag{50}$$

$$\texttt{blok1}(i) = \int_{e_k} N_i^k(x)\, f(x)\, dx. \tag{51}$$

- 1D these elements can be in most cases computed analytically.
- Next we, however, introduce their numerical evaluation.

- Define a reference element $\hat{e} = [0, 1]$ and a linear mapping from $\hat{e}$ to $e_k = [a_k, b_k]$

$$\begin{aligned} x &= \mathcal{F}_k(\xi) = \sum_{i=1}^{2} \hat{N}_i(\xi) p_i^k = a_k \hat{N}_1(\xi) + b_k \hat{N}_2(\xi) \\ &= a_k + (b_k - a_k)\xi = a_k + L_k \xi, \end{aligned} \tag{52}$$

where p_i^k are the end points of e_k ($p_1^k = a_k, p_2^k = b_k$) and $\hat{N}_i^k$ are linear shape functions on $\hat{e}$ defined as

$$\hat{N}_1(\xi) = 1 - \xi, \quad \text{(the "minus" function)}, \tag{53}$$

$$\hat{N}_2(\xi) = \xi, \quad \text{(the "plus" function)}. \tag{54}$$

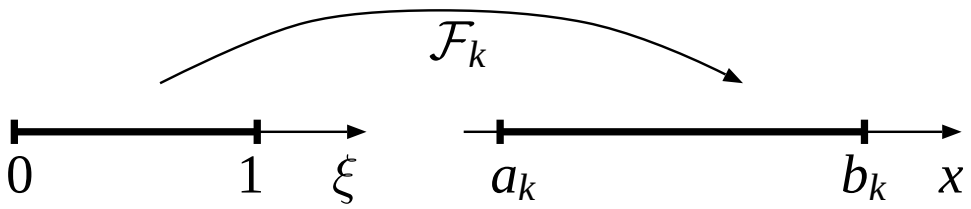


Figure: Mapping $\mathcal{F}_k$ from the reference element $\hat{e}$ to an element e_k.

- Assume that we have a numerical quadrature rule on the reference element

$$\int_{\hat{e}} g(\xi)\, d\xi = \int_0^1 g(\xi)\, d\xi \approx \sum_{p=1}^{P} \omega_p g(\xi_p) \tag{55}$$

  where ξ_p and ω_p are the integration points and weights on $\hat{e}$.
- Now an integral on e_k can be computed numerically using an integral quadrature defined on the reference element $\hat{e}$

$$\int_{e_k} f(x)\, dx = \int_{\hat{e}} f(\mathcal{F}_k(\xi))\, |\det(J_{\mathcal{F}_k})|\, d\xi \approx |\det(J_{\mathcal{F}_k})| \sum_{p=1}^{P} \omega_p f(\mathcal{F}_k(\xi_p)) \tag{56}$$

  where $J_{\mathcal{F}_k}$ is the Jacobian of $\mathcal{F}_k$

$$J_{\mathcal{F}_k} = \frac{\partial \mathcal{F}_k}{\partial_\xi} = b_k - a_k = L_k. \tag{57}$$

- In other words, for a linear mapping $|\det(J_{\mathcal{F}_k})|$ is the length of e_k.

- Matrix elements including products of linear shape functions read

$$\int_{e_k} N_i^k(x)\, N_j^k\, dx = \int_{\hat{e}} N_i^k(\mathcal{F}_k(\xi))\, N_j^k(\mathcal{F}_k(\xi))\, |\det(J_{\mathcal{F}_k})|\, d\xi. \tag{58}$$

- Define

$$N_i^k(x) := \hat{N}_i(\mathcal{F}_k^{-1}(x)) = \hat{N}_i(\xi). \tag{59}$$

- Then integral (58) can be evaluated using integration points and weights, and the shape functions defined on the reference element $\hat{e}$

$$\int_{\hat{e}} N_i^k(\mathcal{F}_k(\xi))\, N_j^k(\mathcal{F}_k(\xi))\, |\det(J_{\mathcal{F}_k})|\, d\xi = |\det(J_{\mathcal{F}_k})| \int_0^1 \hat{N}_i(\xi)\, \hat{N}_j(\xi)\, d\xi$$

$$\approx L_k \sum_{p=1}^{P} \omega_p \hat{N}_i(\xi_p)\, \hat{N}_j(\xi_p)). \tag{60}$$

- Using the chain rule, derivative of a nodal shape function is

$$\frac{d\hat{N}_i(\xi)}{d\xi} = \frac{dN_i^k(x)}{dx}\frac{d\mathcal{F}_k}{d\xi} \quad \text{i.e.,} \quad \frac{dN_i^k(x)}{dx} = \left(\frac{d\mathcal{F}_k}{d\xi}\right)^{-1}\frac{d\hat{N}_i(\xi)}{d\xi}. \tag{61}$$

- Since

$$\frac{d\mathcal{F}_k}{d\xi} = b_k - a_k = L_k, \quad \left(\frac{d\mathcal{F}_k}{d\xi}\right)^{-1} = \frac{1}{L_k}, \tag{62}$$

we have

$$\frac{d\hat{N}_i}{d\xi} = L_k\frac{dN_i^k}{dx} \quad \text{and} \quad \frac{dN_i^k}{dx} = \frac{1}{L_k}\frac{d\hat{N}_i}{d\xi} \tag{63}$$

- Further, since $|\det(J_{\mathcal{F}_k})| = L_k$, $d\hat{N}_1/d\xi = -1$ and $d\hat{N}_2/d\xi = 1$, we get

$$\int_{e_k}\frac{dN_i^k(x)}{dx}\frac{dN_j^k(x)}{dx}dx = |\det(J_{\mathcal{F}_k})|\int_{\hat{e}}\left(\frac{d\mathcal{F}_k}{d_\xi}\right)^{-1}\frac{d\hat{N}_i(\xi)}{d\xi}\left(\frac{d\mathcal{F}_k}{d_\xi}\right)^{-1}\frac{d\hat{N}_j(\xi)}{d\xi}d\xi$$

$$= L_k\int_{\hat{e}}\left(\frac{1}{L_k}\frac{d\hat{N}_i(\xi)}{d\xi}\right)\left(\frac{1}{L_k}\frac{d\hat{N}_j(\xi)}{d\xi}\right)d\xi = \begin{cases} \frac{1}{L_k} & \text{if } i = j, \\ \frac{-1}{L_k} & \text{if } i \neq j. \end{cases} \tag{64}$$

- To summarize, the elements of the local matrices and vector are

$$\texttt{alok1}(i,j) \approx L_k \sum_{p=1}^{P} \omega_p \hat{N}_i(\xi_p)\, \hat{N}_j(\xi_p)), \tag{65}$$

$$\texttt{alok2}(i,j) = \begin{cases} \dfrac{1}{L_k} & \text{if } i = j, \\ \dfrac{-1}{L_k} & \text{if } i \neq j, \end{cases} \tag{66}$$

$$\texttt{blok1}(i) \approx L_k \sum_{p=1}^{P} \omega_p \hat{N}_i(\xi_p)\, f(\mathcal{F}_k(\xi_p)). \tag{67}$$

- The reason for reducing integrals to the reference element is that we need to generate the integration points and weights only once.
- Note that the formula for `alok2` is valid only for linear functions.

- Define two mesh data structures, coordinates of the nodes of the elements

$$\texttt{coord} = [x_1, x_2, \ldots, x_{K+1}], \tag{68}$$

and the element topology, the indeces of the nodes of the elements,

$$\texttt{etopol} = \begin{bmatrix} n_{1,1}, n_{1,2}, \ldots, n_{1,K} \\ n_{2,1}, n_{2,2}, \ldots, n_{2,K} \end{bmatrix}. \tag{69}$$

- Coordinates of the nodes of element k:

$$\texttt{coord}(\texttt{etopol}(1,k)) \quad \text{and} \quad \texttt{coord}(\texttt{etopol}(2,k)). \tag{70}$$

- Global node indeces n_i^k and n_j^k:

$$n_i^k = \texttt{etopol}(i,k) \quad \text{and} \quad n_j^k = \texttt{etopol}(j,k). \tag{71}$$

- "Mesh" of interval $[a, b]$ including K elements and $K + 1$ nodes:

$$\texttt{x} = \texttt{linspace}(\texttt{a}, \texttt{b}, \texttt{K} + 1); \tag{72}$$

- `coord` and `etopol`:

$$\texttt{coord} = \texttt{x}; \quad \texttt{etopol} = [\texttt{1} : \texttt{K}, \texttt{2} : \texttt{K} + \texttt{1}]; \tag{73}$$

- Integration points and weight on the reference element $[0, 1]$ (P is the number of points):

$$[\texttt{xi}, \texttt{w}] = \texttt{gausslegendre}(\texttt{P}); \tag{74}$$

- Linear shape functions at the integration points ξ:

$$\texttt{N1} = \texttt{1} - \texttt{xi}; \quad \texttt{N2} = \texttt{xi}; \tag{75}$$

- Integration points on an element $e_k = [a_k, b_k]$:

$$\texttt{xk} = \texttt{ak} + \texttt{xi} * (\texttt{bk} - \texttt{ak}); \tag{76}$$

- Integral of a function f times a shape function N over e_k:

$$\int_{e_k} f(x)\, N(x)\, dx = \texttt{detJk} * (\texttt{fun}(\texttt{xk}) .* \texttt{N}) * \texttt{w}; \tag{77}$$

  Here $N = N(\xi)$ (values of a shape function at points ξ on the reference element) and `fun(xk) = @(xk) f(xk)` (values of function f at points xk on the element e_k) should be row vectors and `w` (weights on the reference element) should be a column vector, and `detJk` is the Jacobian determinant of mapping $\mathcal{F}_k$.
- A function handle can be used to compute $f(x) = x^2$ at n points x on interval $[a, b]$ e.g., as

```
fun = @(x)x.^2;
x = linspace(a,b,n);
f = fun(x);
```

- Assume that the global indeces of the end points of an element division of interval $[a, b]$ are 1 and $K + 1$.
- Assume also that matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ have been assembled using testing functions that do not vanish at the boundary nodes.
- Neumann boundary data gNa$= g^N(a)$, gNb$= g^N(b)$ is added to source vector $\boldsymbol{b}$ as

$$\mathtt{b(1)} = \mathtt{b(1)} + \mathtt{gNa}; \qquad \mathtt{b(K+1)} = \mathtt{b(K+1)} + \mathtt{gNb}; \tag{78}$$

- To set the Dirichlet data we need to remove the testing functions associated to the boundary nodes

$$\mathtt{A(1,:)} = \mathtt{0}; \quad \mathtt{A(K+1,:)} = \mathtt{0}; \quad \mathtt{b(1)} = \mathtt{0}; \quad \mathtt{b(K+1)} = \mathtt{0}; \tag{79}$$

add value 1 to the diagonal of $\boldsymbol{A}$

$$\mathtt{A(1,1)} = \mathtt{1}; \qquad \mathtt{A(K+1,K+1)} = \mathtt{1}; \tag{80}$$

and the Dirichlet boundary data gDa$= g^D(a)$, gDb$= g^D(b)$ to vector $\boldsymbol{b}$

$$\mathtt{b(1)} = \mathtt{gDa}; \qquad \mathtt{b(K+1)} = \mathtt{gDb}; \tag{81}$$

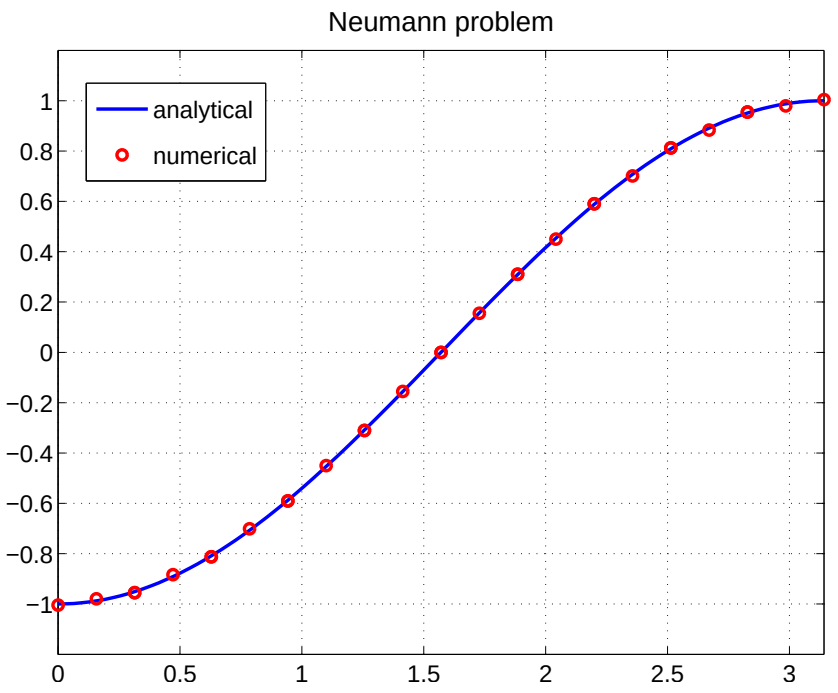
1D FINITE ELEMENT METHOD

On Matlab Programming – Summary of Steps

1. generate mesh on interval [a,b] and define coord and etopol
2. generate integration points on the reference element
3. define linear shape functions and their derivatives
4. initialization of $\boldsymbol{A}$ and $\boldsymbol{b}$, i.e., set elements of $\boldsymbol{A}$ and $\boldsymbol{b}$ to zero
5. assemble $\boldsymbol{A}$ and $\boldsymbol{b}$ by looping over the elements
   - 5.1 find global indeces of the nodes of element e_k
   - 5.2 find coordinates of the end points of element e_k
   - 5.3 define integration points on element e_k
   - 5.4 compute Jacobian of mapping F and its inverse on element k
   - 5.5 compute local matrices and local vector on element e_k
   - 5.6 add local matrices and local vector to the global ones ($\boldsymbol{A}$ and $\boldsymbol{b}$)
6. enforce boundary conditions to the global matrix and vector
7. solve the matrix equation

1D FINITE ELEMENT METHOD

Matlab Exercises



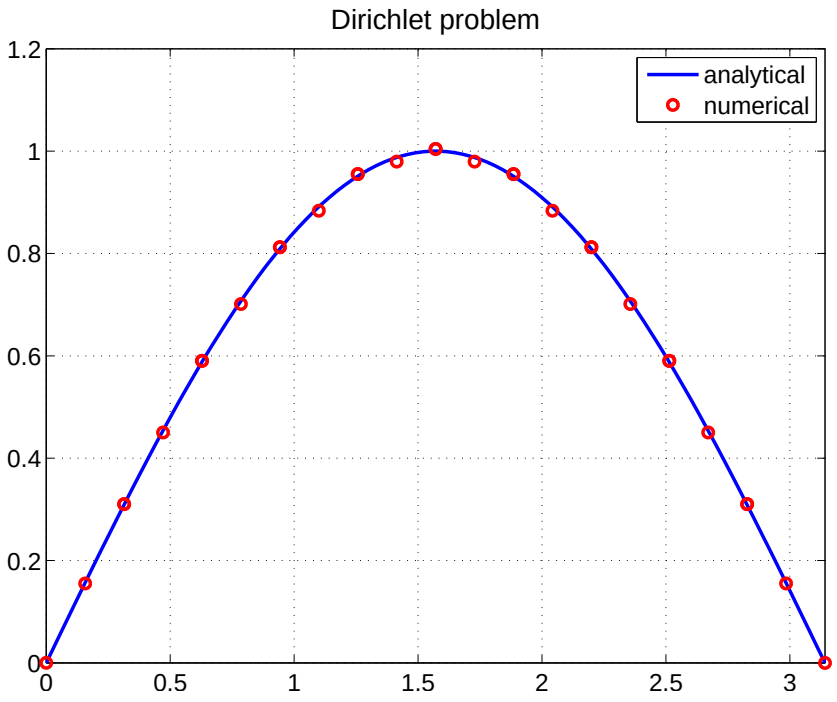


Figure: Solutions for Neumann and Dirichlet problems (exercises 2. and 3.).