# Building a finite element program in MATLAB
# Linear elements in 1d and 2d

D. Peschka

TU Berlin

Supplemental material for the course
"Numerische Mathematik 2 für Ingenieure"
at the Technical University Berlin, WS 2013/2014

# Content

# Plan

# Introduction

Let $V$ a Hilbert space, e.g. $H_0^1(\Omega)$. For a bilinear form $a : V \times V \to \mathbb{R}$ and linear form $f : V \to \mathbb{R}$ solve the following problem:

## Definition (variational problem)

Find $u \in V$ such that

$$a(u, v) = f(v),$$

for all $v \in V$.

## Example (variational problem for $c \geq 0$)

$$a(u, v) = \int_\Omega \nabla u(x) \cdot \nabla v(x) + c\, u(x) v(x)\, \mathrm{d}x,$$
$$f(v) = \int_\Omega f(x) v(x)\, \mathrm{d}x,$$

is the weak form of the elliptic problem $-\Delta u + cu = f$.

# Introduction

By choosing a finite dimensional subspace $V_h \subset V$ we introduce the Galerkin approximation:

## Definition (Galerkin approximation)

Find $u_h \in V_h$ such that

$$a(u_h, v_h) = f(v_h)$$

for all $v_h \in V_h$.

We have a set of functions $w_i$ which form a basis of $V_h$. Then every element of $V_h$ (and in particular the solution) can be written

$$u(x) = \sum_{i=1}^{\dim V_h} \alpha^i w_i(x)$$

Then we may rewrite the Galerkin approximation:

**Definition (Galerkin approximation)**

Find $\alpha^i \in \mathbb{R}$ for $i = 1 \dots \dim V_h$ such that

$$\sum_{i=1}^{\dim V_h} a(w_i, w_j)\alpha^i = f(w_j)$$

for all $j = 1 \dots \dim V_h$. Equivalently we may write

$$A\alpha = f$$

where $A_{ij} = a(w_j, w_i)$, $f_i = f(w_i)$.

# Introduction

**Motivation:** The finite element method (FEM) is a particular method to systematically generate the finite-dimensional subspace $V_h$ of $V$ and generate the Galerkin matrix $A$. Therefore the method

- generates a decomposition of $\Omega = \cup_k \Omega_k$,
- defines basis functions $w_i$ which for each $i$ are only non-zero on a few $\Omega_k$,
- maps these to a reference domain/shape functions.

These basis function are usually piecewise polynomials and globally continuous. The resulting matrix $A$ is sparse. Here we are going to construct a finite-dimensional subspace of the Hilbert space $H^1(\Omega)$ or $H_0^1(\Omega)$. Many concepts shown here are still valid for $H^r(\Omega)$ and $r > 1$.

# Introduction

## Finite elements

Loosely speaking Lagrange finite elements consist of

- geometric objects (triangles, quadrilaterals, tetrahedrons, ...),
- the set of piecewise polynomials $P$ on each geometric object,
- the set of unknowns, i.e. values of basis functions at certain points $\mathbf{p}_i$ (on vertices, edges, elements,...) $i = 1 \dots \dim V_h$.

The construction of FE basis functions $w_i \in P$ is such that the relation between the set of polynomials and the unknowns is

$$w_i(\mathbf{p}_j) = \delta_{ij}$$

and this condition gives unique $w_i$'s.

# Plan

# Decomposition and elements: 1d definition

## Definition (admissible decomposition 1d)

For domain $\Omega \subset \mathbb{R}$ the decomposition

$$\bar{\Omega} = \bigcup_{k=1}^{\texttt{nelement}} \bar{\Omega}_k$$

is called admissible, if $\bar{\Omega}_k \cup \bar{\Omega}_l$ is either

- the full element only for $k = l$,
- a common vertex,
- or otherwise empty.

# Decomposition and elements: 2d definition

## Definition (admissible triangulation/decomposition 2d)

For domain $\Omega \subset \mathbb{R}^2$ with **polygonal boundary** the decomposition

$$\bar{\Omega} = \bigcup_{k=1}^{\texttt{nelement}} \bar{\Omega}_k$$

is called admissible triangulation, if $\bar{\Omega}_k \cup \bar{\Omega}_l$ is either

- the full triangle element only for $k = l$,
- a common vertex,
- **a (full) common edge,**
- or otherwise empty.

# Decomposition and elements: 1d and 2d

**Question:** How do we represent a triangulation in practice?

**Answer 1d:** `x, e2p, npoint, nelement`
where

- `npoint` is the number of points/vertices,
- `nelement` is the number of elements (intervals),
- `x`$\in \mathbb{R}^{\texttt{npoint}}$ is the collection of vertices of triangles,
- `e2p`$\in \mathbb{R}^{\texttt{nelement}\times 2}$ the element-to-point (or vertex) map.

**Answer 2d:** `x, e2p, npoint, nelement`
where

- `npoint` is the number of points/vertices,
- `nelement` is the number of elements (triangles),
- `x,y`$\in \mathbb{R}^{\texttt{npoint}}$ is the collection of vertices of triangles,
- `e2p`$\in \mathbb{R}^{\texttt{nelement}\times 3}$ the element-to-point (or vertex) map.

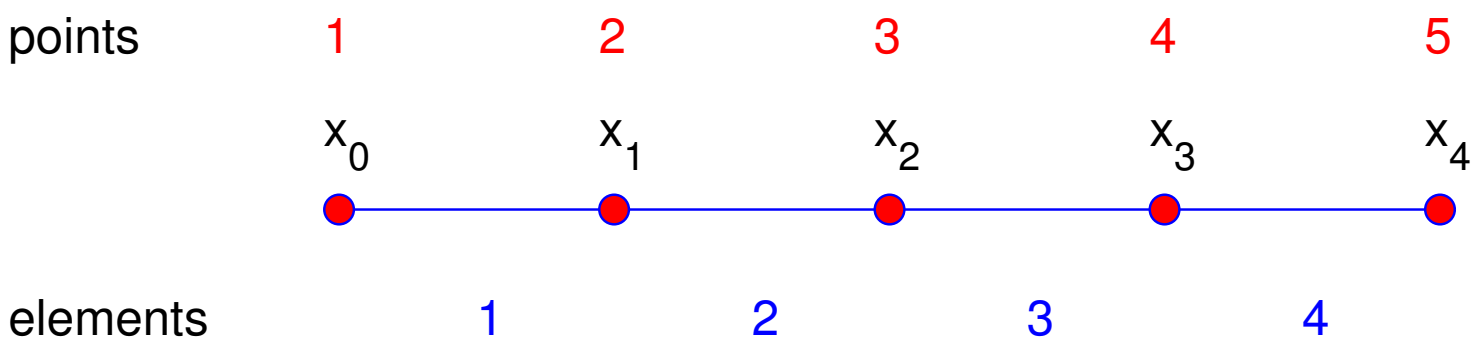
# Decomposition and elements: Remark

Some programs might provide additional information, e.g. in 2d

- number of edges,
- identification of boundary points,
- identification of boundary edges,
- neighboring elements,
- special functions operating on triangulation,

e.g. see `DelaunayTri` class in MATLAB. Despite from being very useful, most of these things can be derived from `x,y,e2p`.

# Decomposition and elements: 1d example

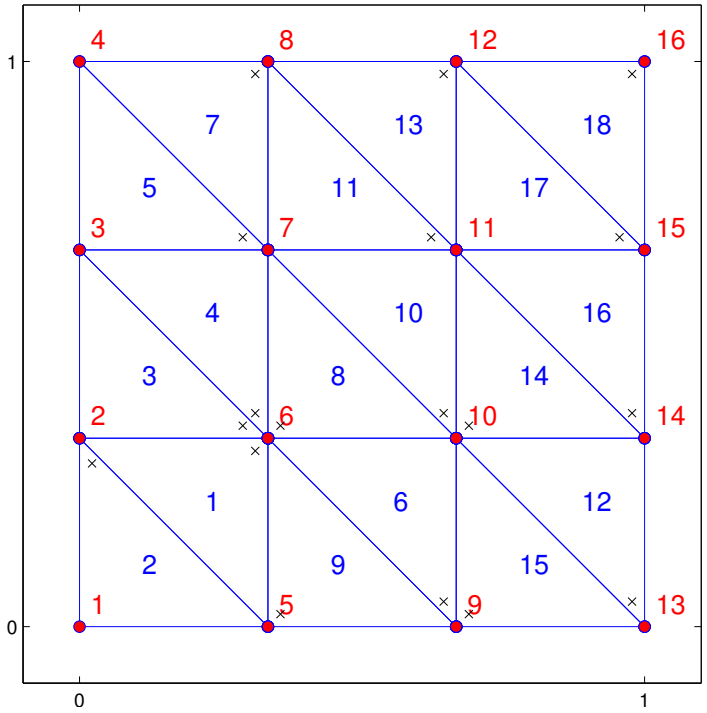



- `npoint=5`,
- `nelement=4`,
- `x=[0 1/4 2/4 3/4 1]`,
- `e2p(1:4,1)=1:4`, `e2p(1:4,2)=2:5`,
- $\bar{\Omega}_k = [x_{k-1}, x_k]$ for $k = 1..$`nelement`.

# Decomposition and elements: 1d MATLAB code

```
1  npoint    = 5;                    % #points in decomposition
2  nelement = npoint - 1;            % #elements/intervals
3
4  x = linspace(0,1,npoint);         % create vertices
5
6  e2p(1:nelement,1) = 1:npoint-1; % create e2p, part 1
7  e2p(1:nelement,2) = 2:npoint;   % create e2p, part 2
8
9  plot(x,0*x,'b-o','MarkerFaceColor','r') % draw decomposition
```

# Decomposition and elements: 2d example




- `npoint=16`,
- `nelement=18`,
- `x=[0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3]/3`,
- `y=[0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3]/3`,
- `e2p`$\in \mathbb{R}^{18\times 3}$.
- $\bar{\Omega}_k = \mathrm{conv}\{\mathbf{x}_{\texttt{e2p(k,1)}}, \mathbf{x}_{\texttt{e2p(k,2)}}, \mathbf{x}_{\texttt{e2p(k,3)}}\}$.

# Decomposition and elements: 2d example

Where the convex-hull of points is defined

$$\mathrm{conv}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) := \left\{ \mathbf{x} \in \mathbb{R}^2 : \mathbf{x} = \sum_{\alpha=1}^{3} \xi_\alpha \mathbf{x}_\alpha; \xi_\alpha \geq 0; \sum_{\alpha=1}^{3} \xi_\alpha = 1 \right\},$$

or alternatively for a point as

$$\begin{pmatrix} 1 \\ x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix},$$

for the barycentric coordinates $\xi_1, \xi_2, \xi_3 \geq 0$.

**Note:** The generalization to 3d gives a tetrahedron, the restriction to 1d gives an interval.

# Decomposition and elements: 2d example

```
e2p =
     6     2     5
     2     1     5
     6     3     2
     6     7     3
     7     4     3
     9    10     6
     8     4     7
     6    10     7
     5     9     6
    10    11     7
    11     8     7
    13    14    10
    12     8    11
    10    14    11
     9    13    10
    14    15    11
    15    12    11
    16    12    15
```

# Decomposition and elements: 2d MATLAB code

```
1  n = 4;                                % parameter
2  [x,y]=meshgrid(linspace(0,1,n));% 2D array of vertices
3  x=x(:); y=y(:);                       % array -> vector
4  e2p = delaunay(x,y);                  % Delaunay triangulation
5  npoint   = size(x,1);                 % # points
6  nelement = size(e2p,1);               % # elements
7
8  % plot the decomposition
9  triplot(e2p,x,y,'o-','Color','b','MarkerFaceColor','r')
```

# Plan

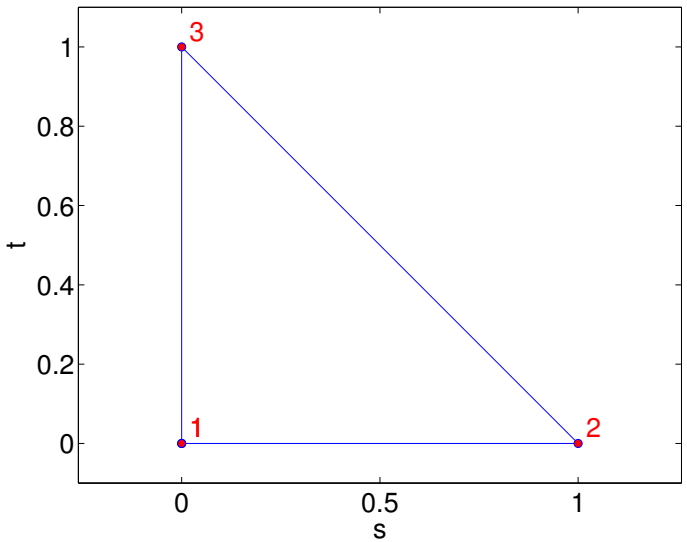
# Reference element: Definition

1d:

$$\Omega_{\text{ref}} = (0,1),$$
$$p_1 = 0, \qquad p_2 = 1$$

2d:

$$\Omega_{\text{ref}} = \{(s,t) \in \mathbb{R}^2 : s,t > 0;\ 0 < s < 1-t\},$$
$$p_1 = (0,0), \qquad p_2 = (1,0), \qquad p_3 = (0,1)$$



# Reference element: Map to element

For each element $F_k$ we have a map

$$F_k : \Omega_{\text{ref}} \to \Omega_k$$

which is affine linear so that the following identity holds

$$F_k(p_m) = \mathbf{x}_i$$

for `i=e2p(k,m)` and $\mathbf{x}_i$ is the $i$th vertex. The map is a linear polynomial, i.e.,

$$F_k(p) = A_k p + B_k$$

where (1d) $A_k, B_k \in \mathbb{R}$ or (2d) $A_k \in \mathbb{R}^{2\times 2}$, $B_k \in \mathbb{R}^2$. We have

$$B_k = \mathbf{x}_{\texttt{e2p(k,1)}}$$

and (1d)

$$A_k = \mathbf{x}_{\texttt{e2p(k,2)}} - \mathbf{x}_{\texttt{e2p(k,1)}} \in \mathbb{R}$$

and (2d)

$$A_k = \left(\mathbf{x}_{\texttt{e2p(k,2)}} - \mathbf{x}_{\texttt{e2p(k,1)}}, \mathbf{x}_{\texttt{e2p(k,3)}} - \mathbf{x}_{\texttt{e2p(k,1)}}\right) \in \mathbb{R}^{2\times 2}$$

# Reference element: Map to element

**Note:** Note that

$$\nabla F_k = A_k$$

MATLAB code fragment to determine $B_k$ for a given $k$

```
1   B(1)=x(e2p(k,1));% first component of B
2   B(2)=y(e2p(k,1));% second component of B
```

MATLAB code fragment to determine $(A_k)_{22}$ for a given $k$

```
1   A(2,2)=y(e2p(k,3))-y(e2p(k,1));% 22 component of A
```

# Reference element

Some extensions are useful:

- transformation $F_k$ using polynomial of higher degree e.g. *isoparametric elements*. [1]
- quadrilateral elements (2d), tetrahedrons (3d)

[1] More complicated since $\nabla F_k$ depends on space and we need to introduce numerical integration techniques.

# Plan

# Basis and shape functions

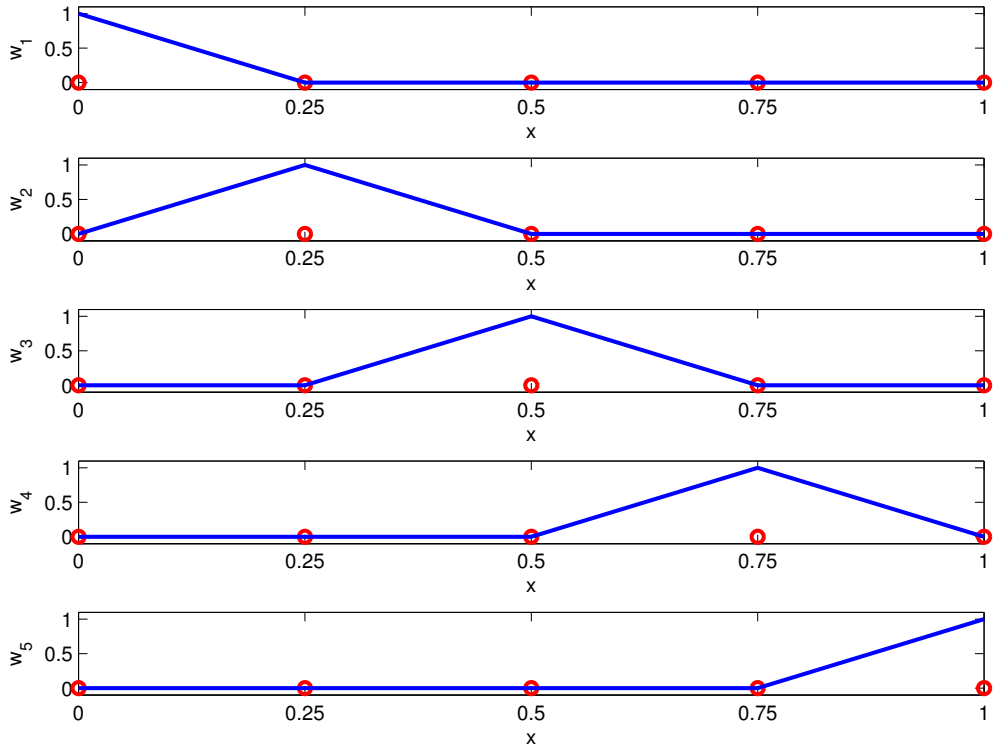
For a given decomposition/triangulation $\Omega_k$ we define basis functions $w_i$ by

- $w_i(\mathbf{x}_j) = \delta_{ij}$,
- $w_i$ is piecewise polynomial (linear) on every $\Omega_k$,
- $w_i$ is continuous,

so that $w_i$ are in the finite element space $w_i \in V_h \subset V$.

# Basis and shape functions: 1d

With `npoint=5` and $V = H^1(0,1)$ we have the following $w_i$:



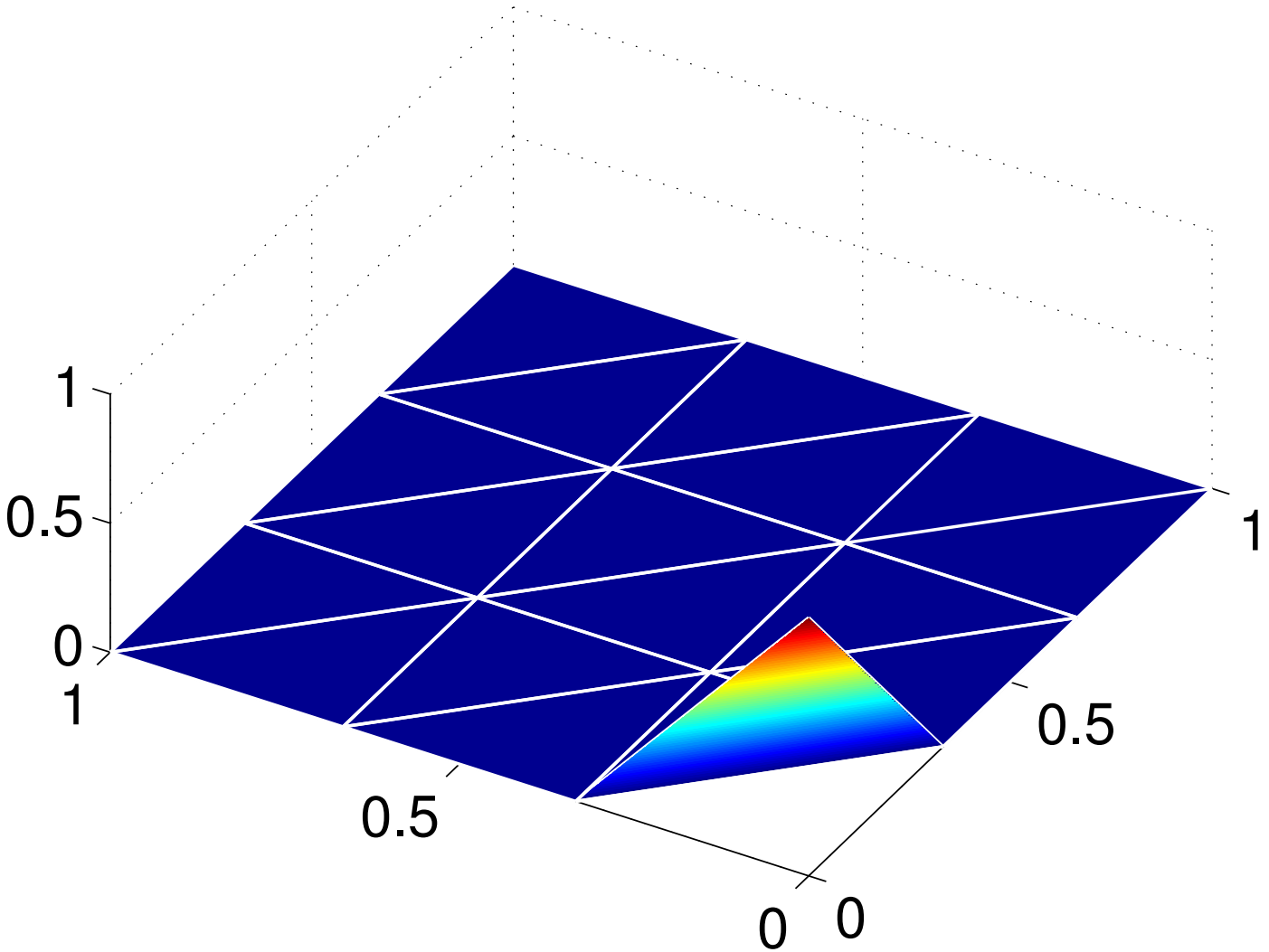

For $V = H_0^1(0,1)$ the (three) basis functions of $V_h$ are

$$\tilde{w}_1(x) = w_2(x), \qquad \tilde{w}_2(x) = w_3(x), \qquad \tilde{w}_3(x) = w_4(x).$$

**Note:** Red circles indicate the position of vertices.

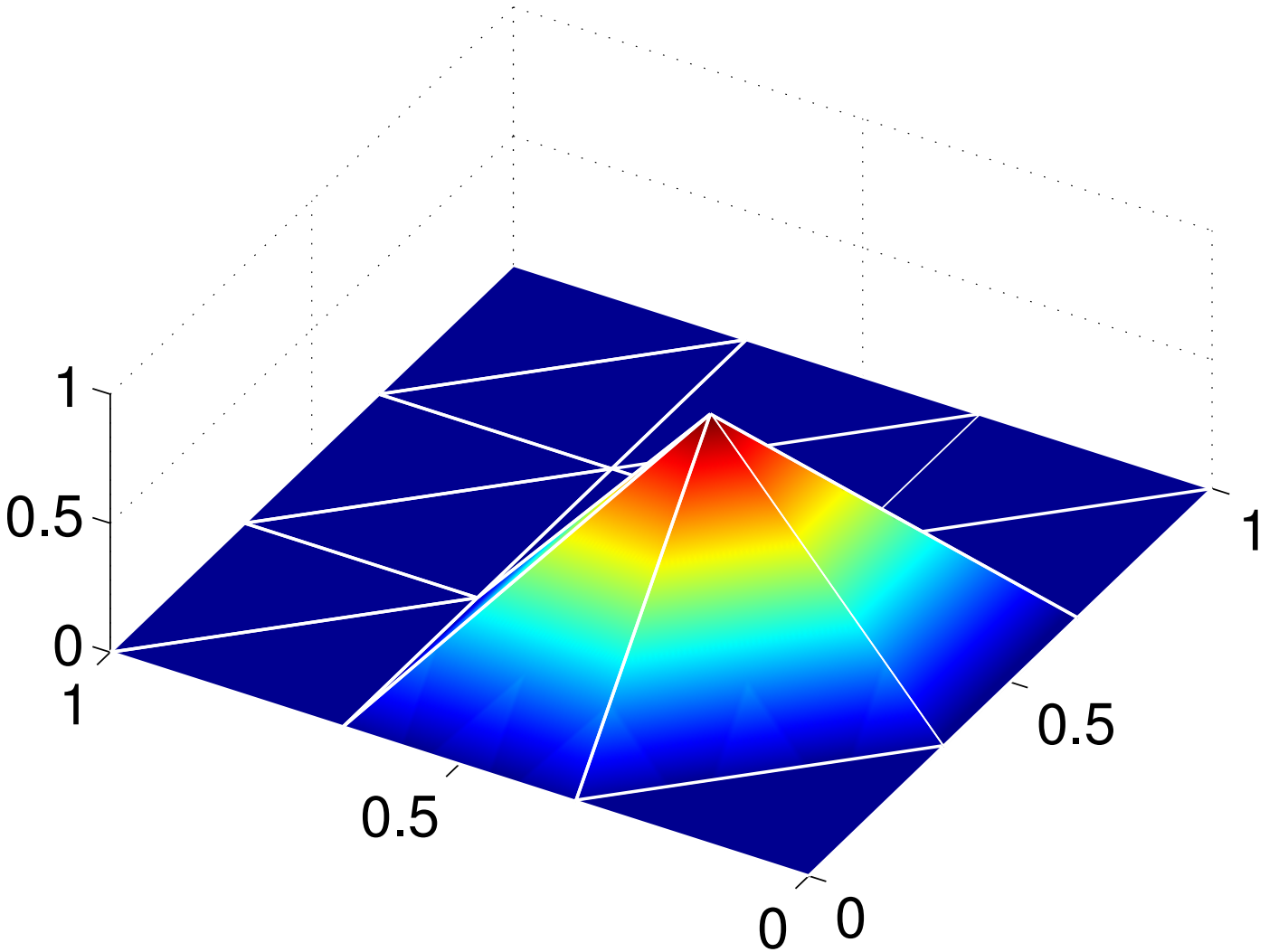
# Basis and shape functions: 2d

With `npoint=16`, $\Omega = (0,1)^2$ and $V = H^1(\Omega)$ we have 16 basis functions $w_i$. For example $i = 1$, $i = 6$, $i = 15$:



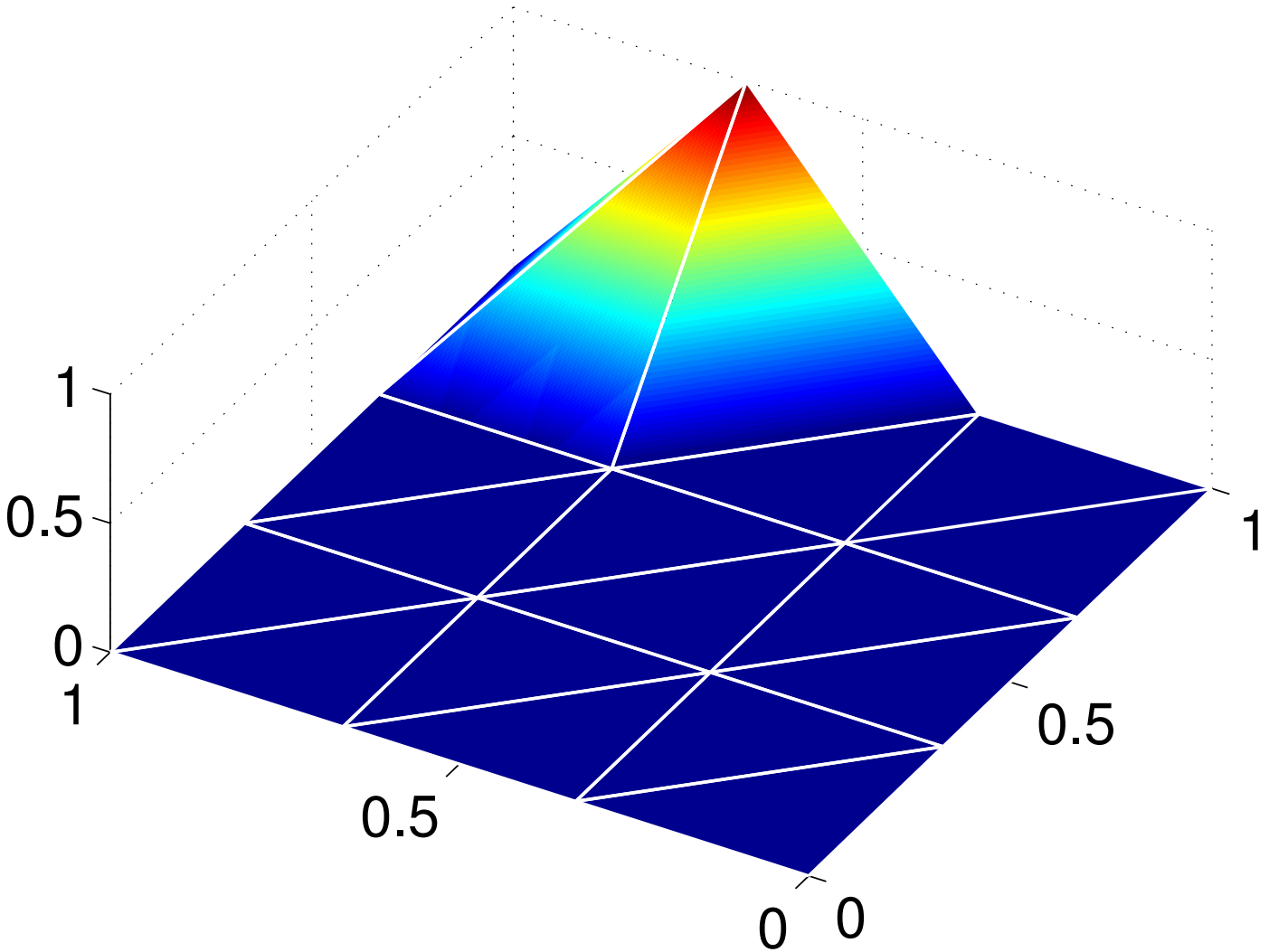
# Basis and shape functions: 2d

With `npoint=16`, $\Omega = (0,1)^2$ and $V = H^1(\Omega)$ we have 16 basis functions $w_i$. For example $i = 1$, $i = 6$, $i = 15$:



# Basis and shape functions: 2d

With `npoint=16`, $\Omega = (0,1)^2$ and $V = H^1(\Omega)$ we have 16 basis functions $w_i$. For example $i = 1$, $i = 6$, $i = 15$:



# Basis and shape functions: basis $\rightarrow$ shape

The basis functions are maps $w_i : \Omega \rightarrow \mathbb{R}$ for $i = 1 \ldots \dim V_h$.
The shape functions are maps $\phi_n : \Omega_{\text{ref}} \rightarrow \mathbb{R}$ for $n = 1 \ldots$ `nphi`.

## Definition (shape functions)

Using the map $F_k$ we have the identity

$$w_i\big(F_k(p)\big) = \phi_n(p)$$

for `i=e2p(k,n)` for every $p \in \Omega_{\text{ref}}$, $k = 1 \ldots$ nelement, $n = 1 \ldots$ `nphi`.

This is the key-property ensuring continuity of $w_i$ and thereby $w_i \in V_h \subset H^1(\Omega)$.
Beyond linear Lagrange elements we need to extend `e2p(1:nelement,1:nphi)`, where each $n = 1 \ldots$ `nphi` is identified with a point $\mathbf{p}_n$ at
1d) i) a vertex, ii) an element
2d) i) a vertex, ii) an edge, iii) an element,
3d) i) a vertex, ii) an edge, iii) a face, iv) an element.

# Basis and shape functions: basis $\rightarrow$ shape

## Example

Linear intervals in 1d: `nphi=2`,
Linear triangles in 2d: `nphi=3`,
Linear tetrahedrons in 3d: `nphi=4`,

Quadratic intervals in 1d: `nphi=3`,
Quadratic triangles in 2d: `nphi=6`,
Quadratic tetrahedrons in 3d: `nphi=10`.

# Basis and shape functions: Shape functions

## Example (1d linear interval, `nphi=2`)

$$\phi_1(s) = 1 - s, \qquad \phi_2(s) = s$$
$$\mathbf{p}_1 = 0, \qquad \mathbf{p}_2 = 1.$$

## Example (2d linear triangle, `nphi=3`)

$$\phi_1(s,t) = 1 - s - t, \quad \phi_2(s,t) = s, \quad \phi_3(s,t) = t,$$
$$\mathbf{p}_1 = (0,0), \quad \mathbf{p}_2 = (1,0), \quad \mathbf{p}_3 = (0,1),$$

## Example (2d quadratic triangle, `nphi=6`)

$$\phi_n(s,t) = a_1^n + a_2^n s + a_3^n t + a_4^n st + a_5^n s^2 + a_6^n t^2$$

where the 36 coefficients are determined by the condition $\phi_n(\mathbf{p}_m) = \delta_{nm}$

$\mathbf{p}_1 = (0,0)$, $\mathbf{p}_2 = (1,0)$, $\mathbf{p}_3 = (0,1)$, $\mathbf{p}_4 = (\frac{1}{2},0)$, $\mathbf{p}_5 = (\frac{1}{2},\frac{1}{2})$, $\mathbf{p}_6 = (0,\frac{1}{2})$.

# Basis and shape functions: Matrices for shape functions

In principle we want to compute integrals such as

$$M = \int_\Omega w_i(x) w_j(x) \,\mathrm{d}\Omega = \sum_{k=1}^{\texttt{nelement}} \int_{\Omega_k} w_i(x) w_j(x) \,\mathrm{d}\Omega$$

Since we have the identity $w_i(F_k(p)) = \phi_n(p)$ we can compute this integral over $\Omega_k = F_k(\Omega_{\mathrm{ref}})$ using integration by substitution

$$\int_{F_k(\Omega_{\mathrm{ref}})} f(x) \,\mathrm{d}x = \int_{\Omega_{\mathrm{ref}}} f\big(F_k(p)\big) \det(\nabla F_k) \,\mathrm{d}p$$

and get

$$\int_{\Omega_k} w_i(x) w_j(x) \,\mathrm{d}\Omega = \int_{\Omega_{\mathrm{ref}}} \phi_m(p) \phi_n(p) \det(A_k) \,\mathrm{d}p$$

with the identity $i = \texttt{e2p(k,m)}, j = \texttt{e2p(k,n)}$.
A similar equation holds for $S = \int_\Omega w_i'(x) w_j'(x) \,\mathrm{d}x$.

# Plan

# The global matrix

How to build the matrix $M$ in 1d:

$$M_{ij} = \sum_{k=1}^{\texttt{nelement}} \int_{\Omega_{\text{ref}}} \phi_m(p)\phi_n(p)\, \det(A_k)\, \mathrm{d}p$$

where $i = \texttt{e2p(k,m)}, j = \texttt{e2p(k,n)}$.

```
%% prepare fem stuff
p     = 7;
nphi  = 2;
x                       = linspace(0,1,2^p+1);
[nelement,npoint,e2p] = generateelements(x);
%% build matrices
ii = zeros(nelement,nphi^2); % sparse i-index
jj = zeros(nelement,nphi^2); % sparse j-index
mm = zeros(nelement,nphi^2); % entry of Galerkin matrix
%% build global from local
for k=1:nelement                % loop over elements
    [edet,dFinv]  = generatetransformation(k,e2p,x); % compute map
    mloc = localmass(edet);                 % element mass matrix
    % compute i,j indices of the global matrix
    ii( k,: ) = [e2p(k,1) e2p(k,2) e2p(k,1) e2p(k,2)]; % local-to-global
    jj( k,: ) = [e2p(k,1) e2p(k,1) e2p(k,2) e2p(k,2)]; % local-to-global
    % compute a(i,j) values of the global matrix
    mm( k,: ) = mloc(:);
end
% create sparse matrices
M=sparse(ii(:),jj(:),mm(:));
```

# The global matrix

MATLAB code generateelements.m

```
1  % Generate infrastructure for FEM.
2  function [nelement,npoint,e2p] = generateelements(x)
3  npoint    = length(x);
4  nelement  = npoint-1;
5  e2p = zeros(nelement,2);
6  e2p(1:nelement,1)=1:npoint-1;
7  e2p(1:nelement,2)=2:npoint;
```

MATLAT code generatetransformation.m

```
1  % Compute the determinant of the Jacobian on every triangle.
2  function [edet,dFinv]=generatetransformation(k,e2p,x)
3  edet = x(e2p(k,2))-x(e2p(k,1));
4  dFinv = 1/edet;
```

MATLAB code localmass.m

```
1  % Local mass matrix M = \int phi_i  phi_j dx
2  function m=localmass(det)
3  m=det*[1/3 1/6;1/6 1/3];
```

MATLAB code localstiff.m

```
1  % Local stiffness matrix S = \int phi'_i * phi'_j dx
2  function S=localstiff(det,dFinv)
3  ndim = 1;
4  nphi = 2;
5
6  gradphiloc(1,1) = -1;
7  gradphiloc(2,1) = +1;
8
9  dphi(1:nphi,1:ndim) = gradphiloc(1:nphi,1:ndim)*dFinv(1:ndim,1:ndim);
10 S=(dphi(:,1)*dphi(:,1)') * det;
```

# The global matrix

When computing $S$ we need partial derivatives of $w_i$, which can be represented for instance by

$$\partial_x w_i(x,y) = \partial_x \phi_m(F_k^{-1}(x,y))$$

where $F_k = A_k\mathbf{p} + B_k = (x,y)^\top$ with $\mathbf{p} = (s,t)^\top$. Then

$$\partial_x w_i = \frac{\partial \phi_m}{\partial s}\frac{\partial s}{\partial x} + \frac{\partial \phi_m}{\partial t}\frac{\partial t}{\partial x}$$
$$\partial_y w_i = \frac{\partial \phi_m}{\partial s}\frac{\partial s}{\partial y} + \frac{\partial \phi_m}{\partial t}\frac{\partial t}{\partial y}$$

so that using $\mathbf{p} = A_k^{-1}((x,y)^\top - B_k)$, we get

$$\nabla w_i = \begin{pmatrix} \partial_x w_i \\ \partial_y w_i \end{pmatrix} = \begin{pmatrix} \partial_x s & \partial_x t \\ \partial_y s & \partial_y t \end{pmatrix} \begin{pmatrix} \partial_s \phi_m \\ \partial_t \phi_m \end{pmatrix} = (A_k^{-1})^\top \nabla \phi_m.$$

so that $\nabla w_i \cdot \nabla w_j = \nabla \phi_m A_k^{-1} A_k^{-\top} \nabla \phi_n$.

End