Basic Concepts in 1-D - Outline

Basics

- Formulation
- Basis functions
- Stiffness matrix

Poisson's equation

- Regular grid
- Boundary conditions
- Irregular grid

Numerical Examples

Let us start with a simple linear system of equations

$$\boxed{\mathbf{Ax} = \mathbf{b}} \qquad \qquad | * \mathbf{y}$$

and observe that we can generally multiply both sides of this equation with **y** without changing its solution. Note that **x**,**y** and **b** are vectors and **A** is a matrix.

$$\rightarrow \mathbf{yAx} = \mathbf{yb} \qquad \mathbf{y} \in \Re^{\mathbf{n}}$$

We first look at Poisson's equation

$$\boxed{-\Delta u(x) = f(x)}$$

where u is a scalar field, f is a source term and in 1-D

$$\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2}$$

Formulation – Poisson's equation

We now multiply this equation with an arbitrary function v(x), (dropping the explicit space dependence)

$$-\Delta u v = f v$$

... and integrate this equation over the whole domain. For reasons of simplicity we define our physical domain *D* in the interval [0, 1].

$$-\int_D \Delta u v = \int_D f v$$

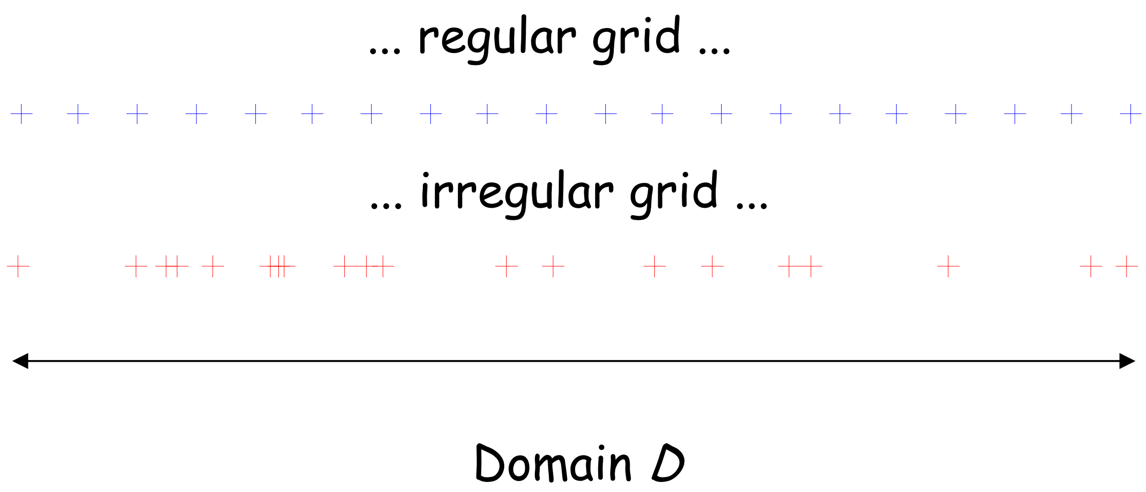
$$-\int_0^1 \Delta u v dx = \int_0^1 f v dx$$

Das Reh springt hoch,
das Reh springt weit,
warum auch nicht,
es hat ja Zeit.

... why are we doing this? ... be patient ...

Discretization

As we are aiming to find a numerical solution to our problem it is clear we have to discretize the problem somehow. In FE problems – similar to FD – the functional values are known at a discrete set of points.



The key idea in FE analysis is to approximate all functions in terms of basis functions φ, so that

$$\boxed{u \approx \tilde{u} = \sum_{i=1}^{N} c_i \varphi_i}$$

Basis function

$$u \approx \tilde{u} = \sum_{i=1}^{N} c_i \varphi_i$$

where N is the number nodes in our physical domain and c_i are real constants.

With an appropriate choice of basis functions φ_i, the coefficients c_i are equivalent to the actual function values at node point i. This – of course – means, that φ_i=1 at node i and 0 at all other nodes ...

Doesn't that ring a bell?

Before we look at the basis functions, let us ...

Partial Integration

... partially integrate the left-hand-side of our equation ...

$$-\int_0^1 \Delta u v dx = \int_0^1 f v dx$$

$$-\int_0^1 (\nabla \bullet \nabla u) v dx = \boxed{\left[\nabla u v\right]_0^1} + \int_0^1 \nabla v \nabla u dx$$

we assume for now that the derivatives of u at the boundaries vanish so that for our particular problem

$$-\int_0^1 (\nabla \bullet \nabla u) v dx = \int_0^1 \nabla v \nabla u dx$$

... so that we arrive at ...

$$\int_0^1 \nabla u \nabla v dx = \int_0^1 f v dx$$

... with u being the unknown. This is also true for our approximate numerical system

$$\int_0^1 \nabla \tilde{u} \nabla v dx = \int_0^1 f v dx$$

... where ...

$$\tilde{u} = \sum_{i=1}^{N} c_i \varphi_i$$

was our choice of approximating u using basis functions.

Partial Integration

$$\int_0^1 \nabla \tilde{u} \nabla v dx = \int_0^1 f v dx$$

... remember that v was an arbitrary real function ...
if this is true for an arbitrary function it is also true if

$$v = \varphi_j$$

... so any of the basis functions previously defined ...

... now let's put everything together ...

The discrete system

The ingredients:

$$v = \varphi_k$$

$$\tilde{u} = \sum_{i=1}^{N} c_i \varphi_i$$

$$\boxed{\int_0^1 \nabla \tilde{u} \nabla v dx = \int_0^1 f v dx}$$

$$\boxed{\int_0^1 \nabla \left(\sum_{i=1}^{n} c_i \varphi_i \right) \nabla \varphi_k dx = \int_0^1 f \varphi_k dx}$$

... leading to ...

... the coefficients c_k are constants so that for one particular function φ_k this system looks like ...

$$\sum_{i=1}^{n} c_i \int_0^1 \nabla \varphi_i \nabla \varphi_k dx = \int_0^1 f \varphi_k dx$$

... probably not to your surprise this can be written in matrix form

$$b_i A_{ik} = g_k$$

$$A_{ik}^T b_i = g_k$$

The solution

... with the even less surprising solution

$$\boxed{b_i = \left(A_{ik}^T\right)^{-1} g_k}$$

remember that while the b_i's are really the coefficients of the basis functions these are the actual function values at node points i as well because of our particular choice of basis functions.

This become clear further on ...

The basis functions

we are looking for functions φ_i with the following property

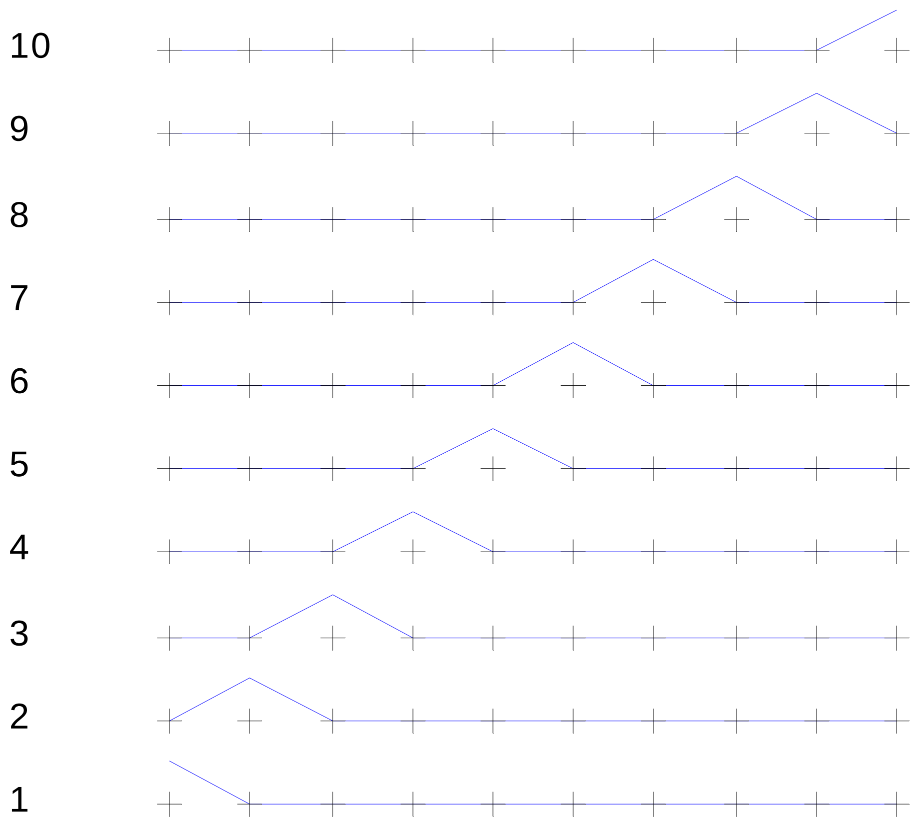
$$\varphi_i(x) = \begin{cases} 1 & for \quad x = x_i \\ 0 & for \quad x = x_j,\ j \neq i \end{cases}$$

... otherwise we are free to choose any function ...

The simplest choice are of course linear functions:

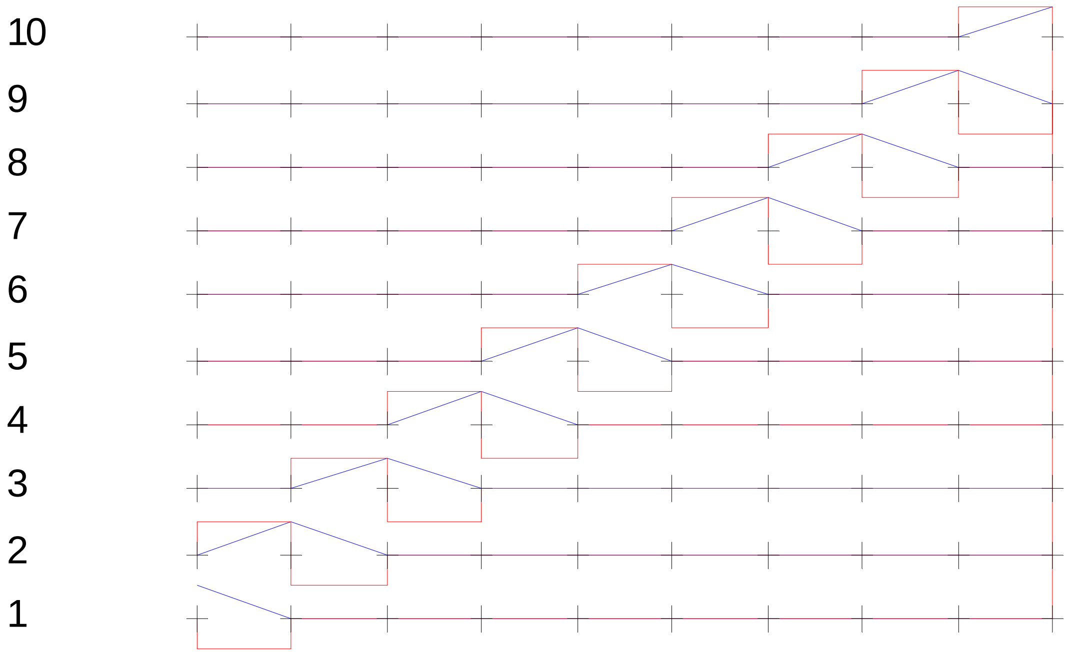
+ grid nodes

blue lines – basis functions φ_i



The basis functions - gradient

To assemble the stiffness matrix we need the gradient (red) of the basis functions (blue)



The stiffness matrix

Knowing the particular form of the basis functions we can now calculate the elements of matrix A_{ij} and vector g_i

$$\boxed{\sum_{i=1}^{n} c_i \int_0^1 \nabla\varphi_i \nabla\varphi_k dx = \int_0^1 f\varphi_k dx} \rightarrow \boxed{b_i A_{ik} = g_k}$$

$$A_{ik} = \int_0^1 \nabla\varphi_i \nabla\varphi_k dx \qquad g_k = \int_0^1 f\varphi_k dx$$

Note that φ_i are continuous functions defined in the interval [0,1], e.g.

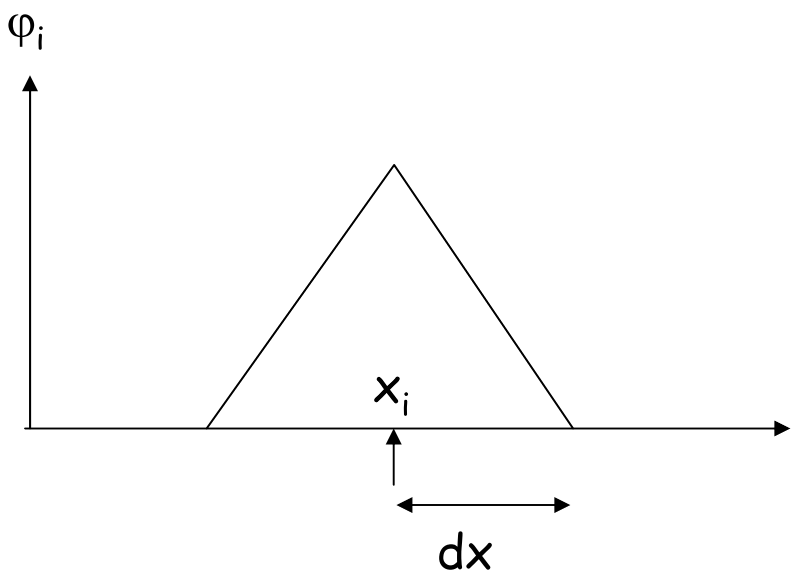
$$\varphi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & for \quad x_{i-1} < x \le x_i \\ \dfrac{x_{i+1} - x}{x_{i+1} - x_i} & for \quad x_i < x < x_{i+1} \\ 0 & elsewhere \end{cases}$$

Let us - for now - assume a regular grid ... then

The stiffness matrix - regular grid

$$\varphi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & \text{for} \quad x_{i-1} < x \le x_i \\ \dfrac{x_{i+1} - x}{x_{i+1} - x_i} & \text{for} \quad x_i < x < x_{i+1} \\ 0 & \text{elsewhere} \end{cases} \quad \Rightarrow \quad \varphi_i(\tilde{x}) = \begin{cases} \dfrac{\tilde{x}}{dx} + 1 & \text{for} \quad -dx < \tilde{x} \le 0 \\ 1 - \dfrac{\tilde{x}}{dx} & \text{for} \quad 0 < \tilde{x} < dx \\ 0 & \text{elsewhere} \end{cases}$$

... where we have used ...

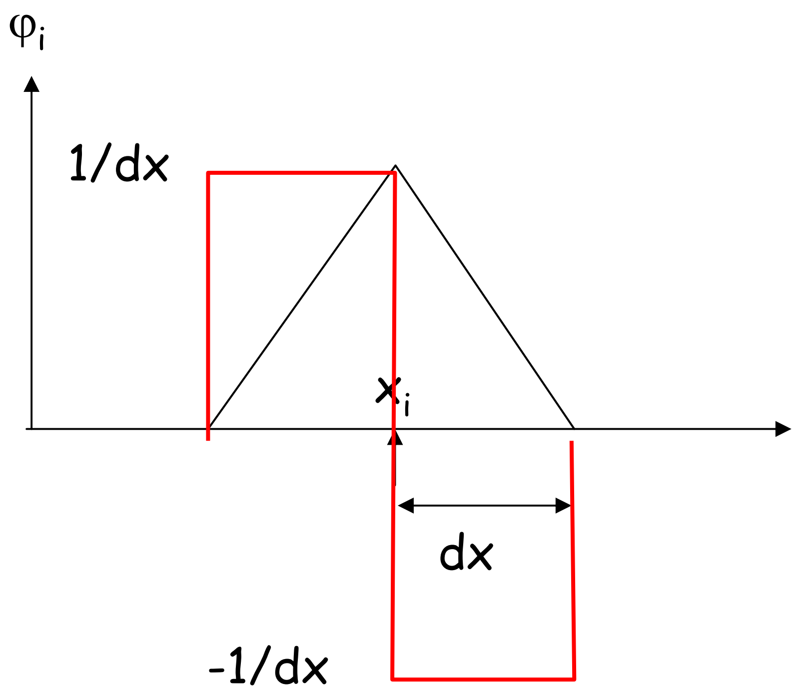


$$\boxed{\begin{aligned} \tilde{x} &= x - x_i \\ dx &= x_i - x_{i-1} \end{aligned}}$$

Regular grid - Gradient

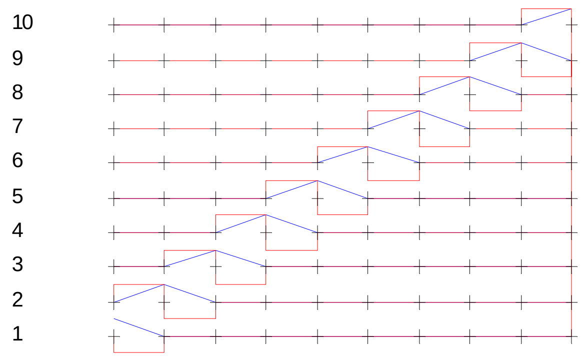
$$\nabla\varphi_i(\tilde{x}) = \begin{cases} 1/dx & for \quad -dx < \tilde{x} \le 0 \\ -1/dx & for \quad 0 < \tilde{x} < dx \\ 0 & elsewhere \end{cases}$$

$$\tilde{x} = x - x_i$$
$$dx = x_i - x_{i-1}$$



Stiffness matix - elements

$$A_{ik} = \int_0^1 \nabla \varphi_i \nabla \varphi_k dx$$



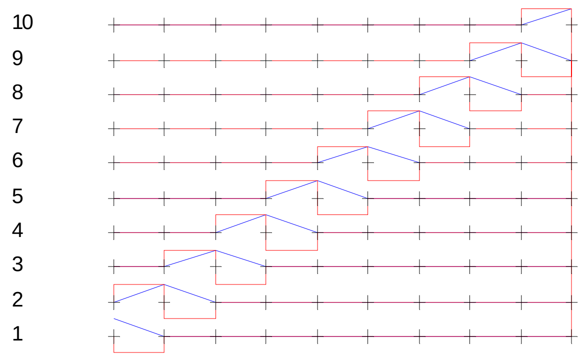
... we have to distinguish various cases ... e.g. ...

$$A_{11} = \int_0^1 \nabla \varphi_1 \nabla \varphi_1 dx = \int_{x_1}^{x_1+dx} \nabla \varphi_1 \nabla \varphi_1 dx = \int_{x_1}^{x_1+dx} \frac{-1}{dx}\frac{-1}{dx} dx = \frac{1}{dx^2} \int_0^{dx} dx = \boxed{\frac{1}{dx}}$$

$$A_{22} = \int_0^1 \nabla \varphi_2 \nabla \varphi_2 dx = \int_{x_2-dx}^{x_2} \nabla \varphi_2 \nabla \varphi_2 dx + \int_{x_2}^{x_2+dx} \nabla \varphi_2 \nabla \varphi_2 dx$$

$$= \frac{1}{dx^2} \int_{-dx}^{0} dx + \frac{1}{dx^2} \int_0^{dx} dx = \boxed{\frac{2}{dx}}$$

$$\boxed{A_{ik} = \int_0^1 \nabla \varphi_i \nabla \varphi_k dx}$$



... and ...

$$A_{12} = \int_0^1 \nabla \varphi_1 \nabla \varphi_2 dx = \int_{x_1}^{x_1+dx} \nabla \varphi_1 \nabla \varphi_2 dx = \int_{x_1}^{x_1+dx} \frac{-1}{dx}\frac{1}{dx} dx$$

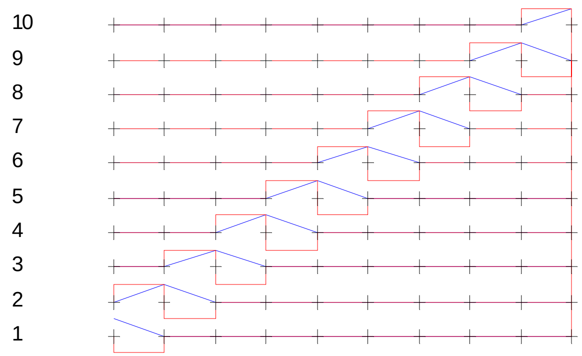
$$= \frac{-1}{dx^2}\int_0^{dx} dx = \boxed{\frac{-1}{dx}}$$

$$A_{21} = A_{12}$$

... so that finally the stiffness matrix looks like ...

Stiffness matix - elements

$$A_{ik} = \int_0^1 \nabla\varphi_i \nabla\varphi_k \, dx$$



$$A_{ij} = \frac{1}{dx}\begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}$$

... so far we have ignored sources and boundary conditions ...

Boundary conditions - sources

... let us start restating the problem ...

$$\boxed{-\Delta u(x) = f(x)}$$

... which we turned into the following formulation ...

$$\sum_{i=1}^{n} c_i \int_0^1 \nabla \varphi_i \nabla \varphi_k dx = \int_0^1 f \varphi_k dx$$

... assuming ...

$$\tilde{u} = \sum_{i=1}^{N} c_i \varphi_i$$ with b.c. $$\tilde{u} = \sum_{i=2}^{N-1} c_i \varphi_i + u(0)\varphi_1 + u(1)\varphi_N$$

where u(0) and u(1) are the values at the boundaries of the domain [0,1]. How is this incorporated into the algorithm?

Boundary conditions - sources

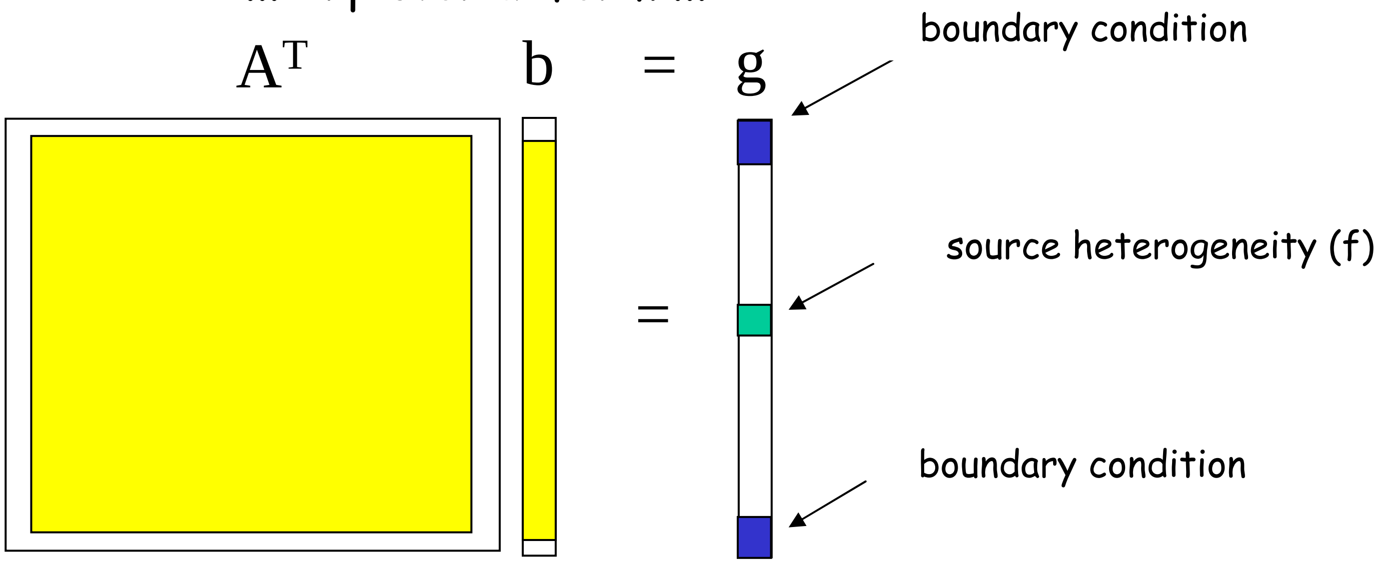
$$\sum_{i=1}^{n} c_i \int_0^1 \nabla\varphi_i \nabla\varphi_k dx = \int_0^1 f\varphi_k dx$$

$$-\Delta u(x) = f(x)$$

... which we turned into the following formulation ...

$$\sum_{i=2}^{n-1} c_i \int_0^1 \nabla\varphi_i \nabla\varphi_k dx = \int_0^1 f\varphi_k dx + u(0)\int_0^1 \nabla\varphi_1 \nabla\varphi_k dx + u(1)\int_0^1 \nabla\varphi_n \nabla\varphi_k dx$$

... in pictorial form ...



... the system *feels* the boundary conditions through the (modified) source term

Numerical example - regular grid

$$-\Delta u(x) = f(x)$$

Domain: [0,1]; nx=100;
dx=1/(nx-1);f(x)=δ(1/2)
Boundary conditions:
u(0)=u(1)=0

Matlab FD code

```
f(nx/2)=1/dx;


for it = 1:nit,

uold=u;

du=(csh(u,1)+csh(u,-1));

u=.5*( f*dx^2 + du );

u(1)=0;

u(nx)=0;


end
```

Matlab FEM code

```
% source term
s=(1:nx)*0;s(nx/2)=1.;
% boundary left  u_1  int{ nabla phi_1  nabla phij }
u1=0;   s(1) =0;
% boundary right  u_nx int{ nabla phi_nx nabla phij }
unx=0; s(nx)=0;

% assemble matrix Aij

A=zeros(nx);

for i=2:nx-1,
   for j=2:nx-1,
      if i==j,
         A(i,j)=2/dx;
      elseif j==i+1
         A(i,j)=-1/dx;
      elseif j==i-1
         A(i,j)=-1/dx;
      else
         A(i,j)=0;
      end
   end
end
fem(2:nx-1)=inv(A(2:nx-1,2:nx-1))*s(2:nx-1)';
fem(1)=u1;
fem(nx)=unx;
```

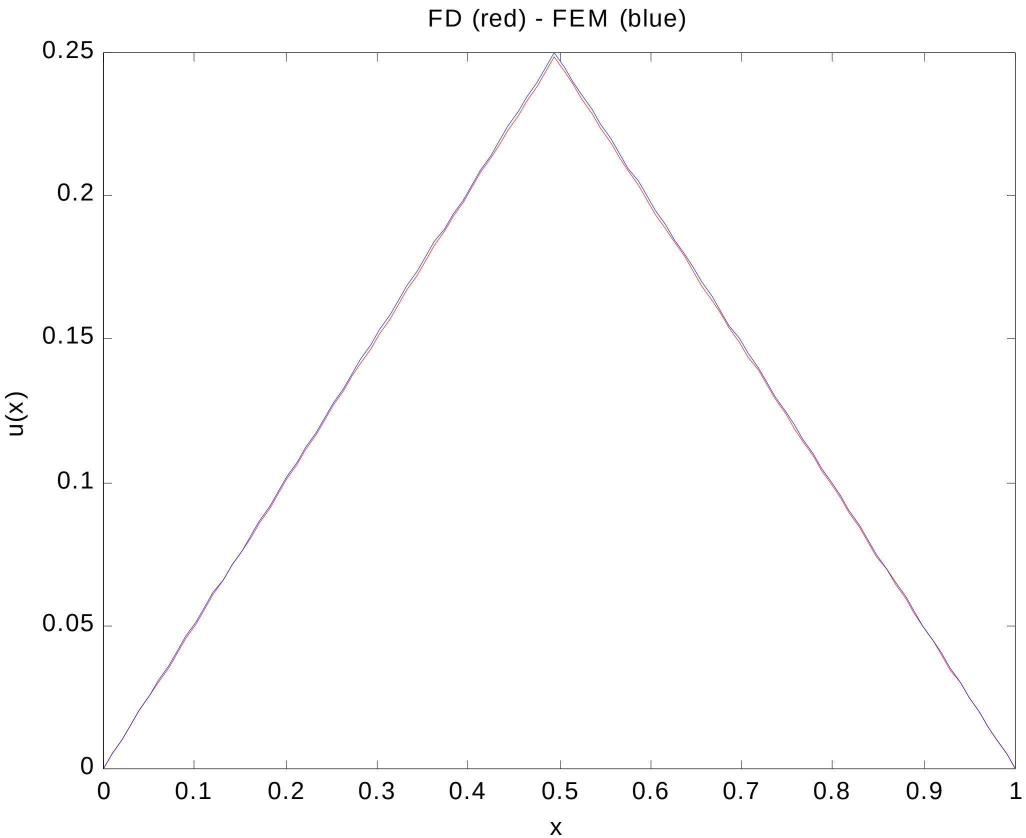
Numerical example - regular grid

$$-\Delta u(x) = f(x)$$

Domain: [0,1]; nx=100;
dx=1/(nx-1);f(x)=δ(1/2)
Boundary conditions:
u(0)=u(1)=0

Matlab FD code (red)

Matlab FEM code (blue)



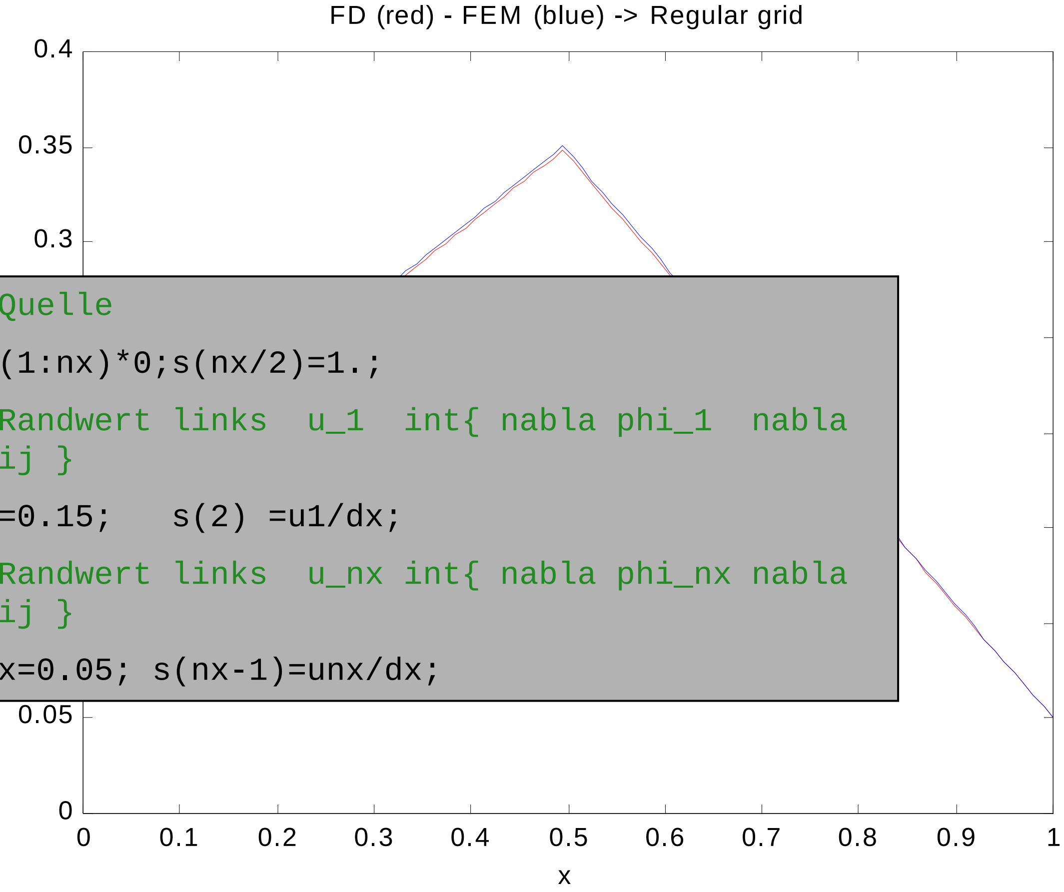
Regular grid - non-zero b.c.

$$-\Delta u(x) = f(x)$$

Domain: [0,1]; nx=100;
dx=1/(nx-1);f(x)=δ(1/2)
Boundary conditions:
u(0)=0.15
u(1)=0.05

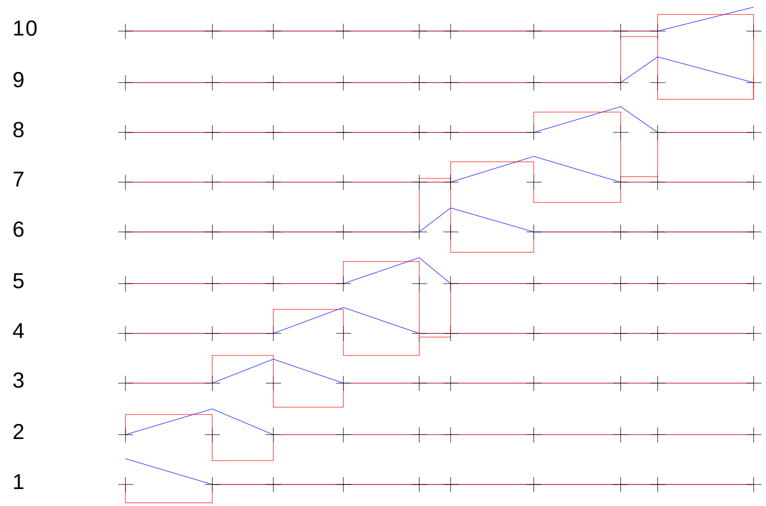
Matlab FD code (red)

Matlab FEM code (blue)




```
% Quelle
s=(1:nx)*0;s(nx/2)=1.;
% Randwert links  u_1  int{ nabla phi_1  nabla
phij }
u1=0.15;   s(2) =u1/dx;
% Randwert links  u_nx int{ nabla phi_nx nabla
phij }
unx=0.05; s(nx-1)=unx/dx;
```

$$A_{ik} = \int_0^1 \nabla \varphi_i \nabla \varphi_k dx$$



$$A_{12} = \int_0^1 \nabla \varphi_1 \nabla \varphi_2 dx = \int_{x_1}^{x_1+h_1} \nabla \varphi_1 \nabla \varphi_2 dx = \int_{x_1}^{x_1+h_1} \frac{-1}{h_1}\frac{1}{h_1} dx$$

$$= \frac{-1}{h_1^2}\int_0^{h_1} dx = \boxed{\frac{-1}{h_1}} = A_{21}$$

$$A_{ii} = \boxed{\frac{1}{h_{i-1}} + \frac{1}{h_i}}$$

i=1	2	3	4	5	6	7
+	+	+	+	+	+	+
h_1	h_2	h_3	h_4	h_5	h_6	

Numerical example - irregular grid

$$-\Delta u(x) = f(x)$$

Domain: [0,1]; nx=100;
dx=1/(nx-1);f(x)=δ(1/2)
Boundary conditions:
u(0)=u0; u(1)=u1

i=1 2 3 4 5 6 7

\+ + + + + + +

h_1 h_2 h_3 h_4 h_5 h_6

Stiffness matrix A

```
for i=2:nx-1,
   for j=2:nx-1,
      if i==j,
         A(i,j)=1/h(i-1)+1/h(i);
      elseif i==j+1
         A(i,j)=-1/h(i-1);
      elseif i+1==j
         A(i,j)=-1/h(i);
      else
         A(i,j)=0;
      end
   end
end
```

Irregular grid - non-zero b.c.

$$-\Delta u(x) = f(x)$$

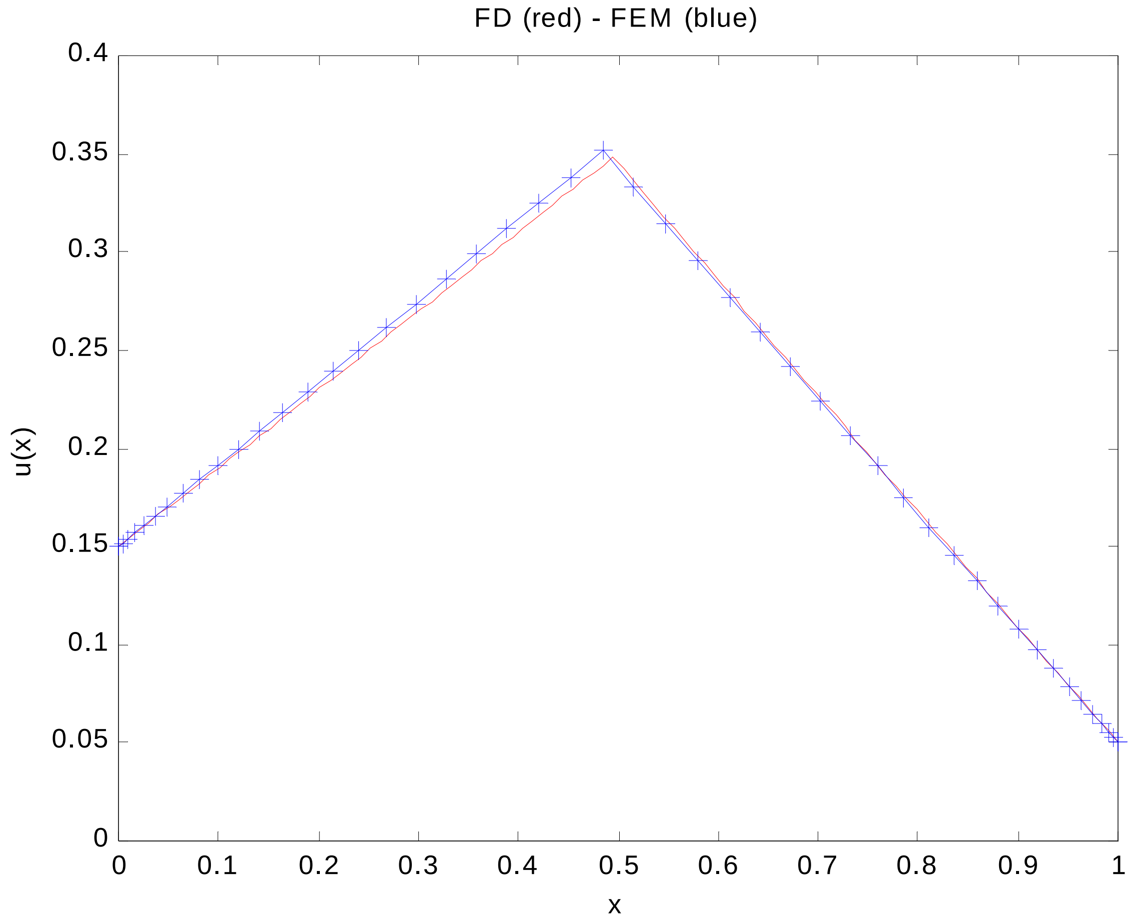
Domain: [0,1]; nx=100;
dx=1/(nx-1);f(x)=δ(1/2)
Boundary conditions:
u(0)=0.15
u(1)=0.05

Matlab FD code (red)

Matlab FEM code (blue)

+ FEM grid points

FEM on Chebyshev grid



Finite elements – summary of the basics

In finite element analysis we approximate a function defined in a Domain D with a set of orthogonal basis functions with coefficients corresponding to the functional values at some node points.

The solution for the values at the nodes for some partial differential equations can be obtained by solving a linear system of equations involving the inversion of (sometimes sparse) matrices.

Boundary conditions are inherently satisfied with this formulation which is one of the advantages compared to finite differences.