

# Study Guide: Solving differential equations with finite elements

Hans Petter Langtangen<sup>1,2</sup>

Center for Biomedical Computing, Simula Research Laboratory<sup>1</sup>

Department of Informatics, University of Oslo<sup>2</sup>

Nov 11, 2013

Our aim is to extend the ideas for approximating  $f$  by  $u$ , or solving

$$u = f$$

to real differential equations like[[[

$$-u'' + bu = f, \quad u(0) = 1, \quad u'(L) = D$$

Three methods are addressed:

- 1 least squares
- 2 Galerkin/projection
- 3 collocation (interpolation)

Method 2 will be totally dominating!

# Abstract differential equation

$$\mathcal{L}(u) = 0, \quad x \in \Omega \quad (1)$$

Examples (1D problems):

$$\mathcal{L}(u) = \frac{d^2 u}{dx^2} - f(x), \quad (2)$$

$$\mathcal{L}(u) = \frac{d}{dx} \left( \alpha(x) \frac{du}{dx} \right) + f(x), \quad (3)$$

$$\mathcal{L}(u) = \frac{d}{dx} \left( \alpha(u) \frac{du}{dx} \right) - au + f(x), \quad (4)$$

$$\mathcal{L}(u) = \frac{d}{dx} \left( \alpha(u) \frac{du}{dx} \right) + f(u, x) \quad (5)$$

# Abstract boundary conditions

$$\mathcal{B}_0(u) = 0, \quad x = 0, \quad \mathcal{B}_1(u) = 0, \quad x = L \quad (6)$$

Examples:

$$\mathcal{B}_i(u) = u - g, \quad \text{Dirichlet condition} \quad (7)$$

$$\mathcal{B}_i(u) = -\alpha \frac{du}{dx} - g, \quad \text{Neumann condition} \quad (8)$$

$$\mathcal{B}_i(u) = -\alpha \frac{du}{dx} - h(u - g), \quad \text{Robin condition} \quad (9)$$

## Reminder about notation

- $u_e(x)$  is the symbol for the *exact* solution of  $\mathcal{L}(u_e) = 0$
- $u(x)$  denotes an *approximate* solution
- We seek  $u \in V$
- $V = \text{span}\{\psi_0(x), \dots, \psi_N(x)\}$ ,  $V$  has basis  $\{\psi_i\}_{i \in \mathcal{I}_s}$
- $\mathcal{I}_s = \{0, \dots, N\}$  is an index set
- $u(x) = \sum_{j \in \mathcal{I}_s} c_j \psi_j(x)$
- Inner product:  $(u, v) = \int_{\Omega} uv \, dx$
- Norm:  $\|u\| = \sqrt{(u, u)}$

Much is similar to approximating a function (solving  $u = f$ ), but two new topics are needed:

- Variational formulation of the differential equation problem (including integration by parts)
- Handling of boundary conditions

# Residual-minimizing principles

- When solving  $u = f$  we knew the error  $e = f - u$  and could use principles for minimizing the error
- When solving  $\mathcal{L}(u_e) = 0$  we do not know  $u_e$  and cannot work with the error  $e = u_e - u$
- We only have the *error in the equation*: the residual  $R$

Inserting  $u = \sum_j c_j \psi_j$  in  $\mathcal{L} = 0$  gives a residual

$$R = \mathcal{L}(u) = \mathcal{L}\left(\sum_j c_j \psi_j\right) \neq 0 \quad (10)$$

Goal: minimize  $R$  wrt  $\{c_i\}_{i \in \mathcal{I}_s}$  (and hope it makes a small  $e$  too)

$$R = R(c_0, \dots, c_N; x)$$

# The least squares method

Idea: minimize

$$E = \|R\|^2 = (R, R) = \int_{\Omega} R^2 dx \quad (11)$$

Minimization wrt  $\{c_i\}_{i \in \mathcal{I}_s}$  implies

$$\frac{\partial E}{\partial c_i} = \int_{\Omega} 2R \frac{\partial R}{\partial c_i} dx = 0 \quad \Leftrightarrow \quad (R, \frac{\partial R}{\partial c_i}) = 0, \quad i \in \mathcal{I}_s \quad (12)$$

$N + 1$  equations for  $N + 1$  unknowns  $\{c_i\}_{i \in \mathcal{I}_s}$



# The Galerkin method

Idea: make  $R$  orthogonal to  $V$ ,

$$(R, v) = 0, \quad \forall v \in V \quad (13)$$

This implies

$$(R, \psi_i) = 0, \quad i \in \mathcal{I}_s \quad (14)$$

$N + 1$  equations for  $N + 1$  unknowns  $\{c_i\}_{i \in \mathcal{I}_s}$

# The Method of Weighted Residuals

Generalization of the Galerkin method: demand  $R$  orthogonal to some space  $W$ , possibly  $W \neq V$ :

$$(R, v) = 0, \quad \forall v \in W \quad (15)$$

If  $\{w_0, \dots, w_N\}$  is a basis for  $W$ :

$$(R, w_i) = 0, \quad i \in \mathcal{I}_s \quad (16)$$

- $N + 1$  equations for  $N + 1$  unknowns  $\{c_i\}_{i \in \mathcal{I}_s}$
- Weighted residual with  $w_i = \partial R / \partial c_i$  gives least squares

# Terminology: test and trial Functions

- $\psi_j$  used in  $\sum_j c_j \psi_j$  is called *trial function*
- $\psi_i$  or  $w_i$  used as weight in Galerkin's method is called *test function*

# The collocation method

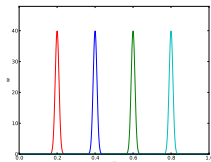
Idea: demand  $R = 0$  at  $N + 1$  points

$$R(x_i; c_0, \dots, c_N) = 0, \quad i \in \mathcal{I}_s \quad (17)$$

Note: The collocation method is a weighted residual method with delta functions as weights

$$0 = \int_{\Omega} R(x; c_0, \dots, c_N) \delta(x - x_i) dx = R(x_i; c_0, \dots, c_N)$$

property of  $\delta(x)$  :  $\int_{\Omega} f(x) \delta(x - x_i) dx = f(x_i), \quad x_i \in \Omega \quad (18)$



# Examples on using the principles

## Goal.

Exemplify the least squares, Galerkin, and collocation methods in a simple 1D problem with global basis functions.

# The first model problem

$$-u''(x) = f(x), \quad x \in \Omega = [0, L], \quad u(0) = 0, \quad u(L) = 0 \quad (19)$$

Basis functions:

$$\psi_i(x) = \sin \left( (i+1) \pi \frac{x}{L} \right), \quad i \in \mathcal{I}_s \quad (20)$$

The residual:

$$\begin{aligned} R(x; c_0, \dots, c_N) &= u''(x) + f(x), \\ &= \frac{d^2}{dx^2} \left( \sum_{j \in \mathcal{I}_s} c_j \psi_j(x) \right) + f(x), \\ &= - \sum_{j \in \mathcal{I}_s} c_j \psi_j''(x) + f(x) \end{aligned} \quad (21)$$

Since  $u(0) = u(L) = 0$  we must ensure that all  $\psi_i(0) = \psi_i(L) = 0$ .  
Then

$$u(0) = \sum_j c_j \psi_j(0) = 0, \quad u(L) = \sum_j c_j \psi_j(L)$$

- $u$  known: Dirichlet boundary condition
- $u'$  known: Neumann boundary condition
- Must have  $\psi_i = 0$  where Dirichlet conditions apply

# The least squares method; principle

$$(R, \frac{\partial R}{\partial c_i}) = 0, \quad i \in \mathcal{I}_s$$

$$\frac{\partial R}{\partial c_i} = \frac{\partial}{\partial c_i} \left( \sum_{j \in \mathcal{I}_s} c_j \psi_j''(x) + f(x) \right) = \psi_i''(x) \quad (22)$$

Because:

$$\frac{\partial}{\partial c_i} (c_0 \psi_0'' + c_1 \psi_1'' + \cdots + c_{i-1} \psi_{i-1}'' + c_i \psi_i'' + c_{i+1} \psi_{i+1}'' + \cdots + c_N \psi_N'') =$$



## The least squares method; equation system

$$\left(\sum_j c_j \psi_j'' + f, \psi_i''\right) = 0, \quad i \in \mathcal{I}_s \quad (23)$$

Rearrangement:

$$\sum_{j \in \mathcal{I}_s} (\psi_i'', \psi_j'') c_j = -(f, \psi_i''), \quad i \in \mathcal{I}_s \quad (24)$$

This is a linear system

$$\sum_{j \in \mathcal{I}_s} A_{i,j} c_j = b_i, \quad i \in \mathcal{I}_s$$

with

$$\begin{aligned} A_{i,j} &= (\psi_i'', \psi_j'') \\ &= \pi^4 (i+1)^2 (j+1)^2 L^{-4} \int_0^L \sin\left((i+1)\pi \frac{x}{L}\right) \sin\left((j+1)\pi \frac{x}{L}\right) dx \\ &= \begin{cases} \frac{1}{2} L^{-3} \pi^4 (i+1)^4 & i = j \\ 0, & i \neq j \end{cases} \end{aligned} \quad (25)$$

# Orthogonality of the basis functions gives diagonal matrix

Useful property:

$$\int_0^L \sin\left((i+1)\pi\frac{x}{L}\right) \sin\left((j+1)\pi\frac{x}{L}\right) dx = \delta_{ij}, \quad \delta_{ij} = \begin{cases} \frac{1}{2}L & i=j \\ 0, & i \neq j \end{cases} \quad (27)$$

$\Rightarrow (\psi_i'', \psi_j'') = \delta_{ij}$ , i.e., diagonal  $A_{i,j}$ , and we can easily solve for  $c_i$ :

$$c_i = \frac{2L}{\pi^2(i+1)^2} \int_0^L f(x) \sin\left((i+1)\pi\frac{x}{L}\right) dx \quad (28)$$

# Least squares method; solution

Let's sympy do the work ( $f(x) = 2$ ):

```
from sympy import *
import sys

i, j = symbols('i j', integer=True)
x, L = symbols('x L')
f = 2
a = 2*L/(pi**2*(i+1)**2)
c_i = a*integrate(f*sin((i+1)*pi*x/L), (x, 0, L))
c_i = simplify(c_i)
print c_i
```

$$c_i = 4 \frac{L^2 \left( (-1)^i + 1 \right)}{\pi^3 (i^3 + 3i^2 + 3i + 1)}, \quad u(x) = \sum_{k=0}^{N/2} \frac{8L^2}{\pi^3 (2k+1)^3} \sin \left( (2k+1)\pi \frac{x}{L} \right) \quad (29)$$

Fast decay:  $c_2 = c_0/27$ ,  $c_4 = c_0/125$  - only one term might be good enough:

$$u(x) \approx \frac{8L^2}{\pi^3} \sin \left( \pi \frac{x}{L} \right) .$$

# The Galerkin method; principle

$$R = u'' + f:$$

$$(u'' + f, v) = 0, \quad \forall v \in V,$$

or

$$(u'', v) = -(f, v), \quad \forall v \in V \quad (30)$$

This is a *variational formulation* of the differential equation problem.

$\forall v \in V$  means for all basis functions:

$$\left( \sum_{j \in \mathcal{I}_s} c_j \psi_j'', \psi_i \right) = -(f, \psi_i), \quad i \in \mathcal{I}_s \quad (31)$$

# The Galerkin method; solution

Since  $\psi_i'' \propto \psi_i$ , Galerkin's method gives the same linear system and the same solution as the least squares method (in this particular example).

# The collocation method

$R = 0$  (i.e., the differential equation) must be satisfied at  $N + 1$  points:

$$-\sum_{j \in \mathcal{I}_s} c_j \psi_j''(x_i) = f(x_i), \quad i \in \mathcal{I}_s \quad (32)$$

This is a linear system  $\sum_j A_{i,j} = b_i$  with entries

$$A_{i,j} = -\psi_j''(x_i) = (j+1)^2 \pi^2 L^{-2} \sin\left((j+1)\pi \frac{x_i}{L}\right), \quad b_i = 2$$

Choose:  $N = 0$ ,  $x_0 = L/2$

$$c_0 = 2L^2/\pi^2$$

# Comparison of the methods

- Exact solution:  $u(x) = x(L - x)$
- Galerkin or least squares ( $N = 0$ ):  $u(x) = 8L^2\pi^{-3} \sin(\pi x/L)$
- Collocation method ( $N = 0$ ):  $u(x) = 2L^2\pi^{-2} \sin(\pi x/L)$ .
- Max error in Galerkin/least sq.:  $-0.008L^2$
- Max error in collocation:  $0.047L^2$

Second-order derivatives will hereafter be integrated by parts

$$\begin{aligned}\int_0^L u''(x)v(x)dx &= - \int_0^L u'(x)v'(x)dx + [vu']_0^L \\ &= - \int_0^L u'(x)v'(x)dx + u'(L)v(L) - u'(0)v(0)\end{aligned}\tag{33}$$

Motivation:

- Lowers the order of derivatives
- Gives more symmetric forms (incl. matrices)
- Enables easy handling of Neumann boundary conditions
- Finite element basis functions  $\varphi_i$  have discontinuous derivatives (at cell boundaries) and are not suited for terms with  $\varphi_i''$

## Boundary function; principles

- What about nonzero Dirichlet conditions? Say  $u(L) = D$
- We always require  $\psi_i(L) = 0$  (i.e.,  $\psi_i = 0$  where Dirichlet



## Boundary function; example (1)

Dirichlet conditions:  $u(0) = C$  and  $u(L) = D$ . Choose for example

$$B(x) = \frac{1}{L}(C(L-x) + Dx) : \quad B(0) = C, \quad B(L) = D$$

$$u(x) = B(x) + \sum_{j \in \mathcal{I}_s} c_j \psi_j(x), \quad (34)$$

$$u(0) = B(0) = C, \quad u(L) = B(L) = D$$

## Boundary function; example (2)

Dirichlet condition:  $u(L) = D$ . Choose for example

$$B(x) = D : \quad B(L) = D$$

$$u(x) = B(x) + \sum_{j \in \mathcal{I}_s} c_j \psi_j(x), \quad (35)$$

$$u(L) = B(L) = D$$

# Impact of the boundary function on the space where we seek the solution

- $\{\psi_i\}_{i \in \mathcal{I}_s}$  is a basis for  $V$
- $\sum_{j \in \mathcal{I}_s} c_j \psi_j(x) \in V$
- But  $u \notin V$ !
- Reason: say  $u(0) = C$  and  $u \in V$  (any  $v \in V$  has  $v(0) = C$ , then  $2u \notin V$  because  $2u(0) = 2C$ )
- When  $u(x) = B(x) + \sum_{j \in \mathcal{I}_s} c_j \psi_j(x)$ ,  $B \neq 0$ ,  $B \notin V$  (in general) and  $u \notin V$ , but  $(u - B) \in V$  since  $\sum_j c_j \psi_j \in V$

# Abstract notation for variational formulations

The finite element literature (and much FEniCS documentation) applies an abstract notation for the variational formulation:

Find  $(u - B) \in V$  such that

$$a(u, v) = L(v) \quad \forall v \in V$$

## Example on abstract notation

$$-u'' = f, \quad u'(0) = C, \quad u(L) = D, \quad u = D + \sum_j c_j \psi_j$$

Variational formulation:

$$\int_{\Omega} u' v' dx = \int_{\Omega} f v dx - v(0)C \quad \text{or} \quad (u', v') = (f, v) - v(0)C \quad \forall v \in V$$

Abstract formulation: find  $(u - B) \in V$  such that

$$a(u, v) = L(v) \quad \forall v \in V$$

We identify

$$a(u, v) = (u', v'), \quad L(v) = (f, v) - v(0)C$$

# Bilinear and linear forms

- $a(u, v)$  is a *bilinear form*
- $L(v)$  is a *linear form*

Linear form means

$$L(\alpha_1 v_1 + \alpha_2 v_2) = \alpha_1 L(v_1) + \alpha_2 L(v_2),$$

Bilinear form means

$$a(\alpha_1 u_1 + \alpha_2 u_2, v) = \alpha_1 a(u_1, v) + \alpha_2 a(u_2, v),$$

$$a(u, \alpha_1 v_1 + \alpha_2 v_2) = \alpha_1 a(u, v_1) + \alpha_2 a(u, v_2)$$

In nonlinear problems: Find  $(u - B) \in V$  such that  
 $F(u; v) = 0 \quad \forall v \in V$

# The linear system associated with abstract form

$$a(u, v) = L(v) \quad \forall v \in V \quad \Leftrightarrow \quad a(u, \psi_i) = L(\psi_i) \quad i \in \mathcal{I}_s$$

We can now derive the corresponding linear system once and for all:

$$a\left(\sum_{j \in \mathcal{I}_s} c_j \psi_j, \psi_i\right) c_j = L(\psi_i) \quad i \in \mathcal{I}_s$$

Because of linearity,

$$\sum_{j \in \mathcal{I}_s} \underbrace{a(\psi_j, \psi_i)}_{A_{i,j}} c_j = \underbrace{L(\psi_i)}_{b_i} \quad i \in \mathcal{I}_s$$

Given  $a(u, v)$  and  $L(v)$  in a problem, we can immediately generate the linear system:

$$A_{i,j} = a(\psi_j, \psi_i), \quad b_i = L(\psi_i)$$

# Equivalence with minimization problem

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

$$a(u, v) = L(v) \quad \forall v \in V,$$

is equivalent to minimizing the functional

$$F(v) = \frac{1}{2}a(v, v) - L(v)$$

over all functions  $v \in V$ . That is,

$$F(u) \leq F(v) \quad \forall v \in V.$$

- Much used in the early days of finite elements
- Still much used in structural analysis and elasticity
- Not as general as Galerkin's method (since  $a(u, v) = a(v, u)$ )



# Examples on variational formulations

## Goal.

Derive variational formulations for many prototype differential equations in 1D that include

- variable coefficients
- mixed Dirichlet and Neumann conditions
- nonlinear coefficients

## Variable coefficient; problem

$$-\frac{d}{dx} \left( \alpha(x) \frac{du}{dx} \right) = f(x), \quad x \in \Omega = [0, L], \quad u(0) = C, \quad u(L) = D \quad (36)$$

- Variable coefficient  $\alpha(x)$
- *Nonzero* Dirichlet conditions at  $x = 0$  and  $x = L$
- Must have  $\psi_i(0) = \psi_i(L) = 0$
- $V = \text{span}\{\psi_0, \dots, \psi_N\}$
- $v \in V$ :  $v(0) = v(L) = 0$

$$u(x) = B(x) + \sum_{j \in \mathcal{I}_s} c_j \psi_j(x)$$

$$B(x) = C + \frac{1}{L}(D - C)x$$

## Variable coefficient; variational formulation (1)

$$R = -\frac{d}{dx} \left( a \frac{du}{dx} \right) - f$$

Galerkin's method:

$$(R, v) = 0, \quad \forall v \in V,$$

or with integrals:

$$\int_{\Omega} \left( \frac{d}{dx} \left( \alpha \frac{du}{dx} \right) - f \right) v \, dx = 0, \quad \forall v \in V.$$

## Variable coefficient; variational formulation (2)

Integration by parts:

$$-\int_{\Omega} \frac{d}{dx} \left( \alpha(x) \frac{du}{dx} \right) v \, dx = \int_{\Omega} \alpha(x) \frac{du}{dx} \frac{dv}{dx} \, dx - \left[ \alpha \frac{du}{dx} v \right]_0^L.$$

Boundary terms vanish since  $v(0) = v(L) = 0$

### Variational formulation.

Find  $(u - B) \in V$  such that

$$\int_{\Omega} \alpha(x) \frac{du}{dx} \frac{dv}{dx} \, dx = \int_{\Omega} f(x) v \, dx, \quad \forall v \in V,$$

Compact notation:

$$\underbrace{(\alpha u', v')}_{a(u,v)} = \underbrace{(f, v)}_{L(v)}, \quad \forall v \in V$$

## Variable coefficient; linear system (the easy way)

With

$$a(u, v) = (\alpha u', v), \quad L(v) = (f, v)$$

we can just use the formula for the linear system:

$$A_{i,j} = a(\psi_j, \psi_i) = (\alpha \psi_j', \psi_i') = \int_{\Omega} \alpha \psi_j' \psi_i' \, dx = \int_{\Omega} \psi_i' \alpha \psi_j' \, dx = a(\psi_i, \psi_j) =$$

$$b_i = (f, \psi_i) = \int_{\Omega} f \psi_i \, dx$$

# Variable coefficient; linear system (full derivation)

$v = \psi_i$  and  $u = B + \sum_j c_j \psi_j$ :

$$(\alpha B' + \alpha \sum_{j \in \mathcal{I}_s} c_j \psi_j', \psi_i') = (f, \psi_i), \quad i \in \mathcal{I}_s.$$

Reorder to form linear system:

$$\sum_{j \in \mathcal{I}_s} (\alpha \psi_j', \psi_i') c_j = (f, \psi_i) + (a(D - C)L^{-1}, \psi_i'), \quad i \in \mathcal{I}_s.$$

This is  $\sum_j A_{i,j} c_j = b_i$  with

$$A_{i,j} = (a \psi_j', \psi_i') = \int_{\Omega} \alpha(x) \psi_j'(x) \psi_i'(x) dx$$

$$b_i = (f, \psi_i) + (a(D - C)L^{-1}, \psi_i') = \int_{\Omega} \left( f(x) \psi_i(x) + \alpha(x) \frac{D - C}{L} \psi_i'(x) \right) dx$$

## First-order derivative in the equation and boundary condition; problem

$$-u''(x) + bu'(x) = f(x), \quad x \in \Omega = [0, L], \quad u(0) = C, \quad u'(L) = E \quad (37)$$

New features:

- first-order derivative  $u'$  in the equation
- boundary condition with  $u'$ :  $u'(L) = E$

Initial steps:

- Must force  $\psi_i(0) = 0$  because of Dirichlet condition at  $x = 0$
- Boundary function:  $B(x) = C(L - x)$  or just  $B(x) = C$
- No requirements on  $\psi_i(L)$  (no Dirichlet condition at  $x = L$ )

## First-order derivative in the equation and boundary condition; details

$$u = C + \sum_{j \in \mathcal{I}_s} c_j \psi_j(x)$$

Galerkin's method: multiply by  $v$ , integrate over  $\Omega$ , integrate by parts.

$$(-u'' + bu' - f, v) = 0, \quad \forall v \in V$$

$$(u', v') + (bu', v) = (f, v) + [u'v]_0^L, \quad \forall v \in V$$

Now,  $[u'v]_0^L = u'(L)v(L) = Ev(L)$  because  $v(0) = 0$  and  $u'(L) = E$ :

$$(u'v') + (bu', v) = (f, v) + Ev(L), \quad \forall v \in V$$



## First-order derivative in the equation and boundary condition; observations

$$(u'v') + (bu', v) = (f, v) + Ev(L), \quad \forall v \in V,$$

Important:

- The boundary term can be used to implement Neumann conditions
- Forgetting the boundary term implies the condition  $u' = 0$  (!)
- Such conditions are called *natural boundary conditions*

# First-order derivative in the equation and boundary condition; abstract notation

Abstract notation:

$$a(u, v) = L(v) \quad \forall v \in V$$

Here:

$$a(u, v) = (u', v') + (bu', v)$$

$$L(v) = (f, v) + Ev(L)$$

# First-order derivative in the equation and boundary condition; linear system

Insert  $u = C + \sum_j c_j \psi_j$  and  $v = \psi_i$ :

$$\sum_{j \in \mathcal{I}_s} \underbrace{((\psi'_j, \psi'_i) + (b\psi'_j, \psi_i))}_{A_{i,j}} c_j = \underbrace{(f, \psi_i) + E\psi_i(L)}_{b_i}$$

Observation:  $A_{i,j}$  is not symmetric because of the term

$$(b\psi'_j, \psi_i) = \int_{\Omega} b\psi'_j \psi_i dx \neq \int_{\Omega} b\psi'_i \psi_j dx = (\psi'_i, b\psi_j)$$

## Terminology: natural and essential boundary conditions

$$(u', v') + (bu', v) = (f, v) + u'(L)v(L) - u'(0)v(0)$$

- Note: forgetting the boundary terms implies  $u'(L) = u'(0) = 0$  (unless prescribe a Dirichlet condition)
- Conditions on  $u'$  are simply inserted in the variational form and called *natural conditions*
- Conditions on  $u$  at  $x = 0$  requires modifying  $V$  (through  $\psi_i(0) = 0$ ) and are known as *essential conditions*

### Lesson learned.

It is easy to forget the boundary term when integrating by parts.  
That mistake may prescribe a condition on  $u'$ !

Problem:

$$-(\alpha(u)u')' = f(u), \quad x \in [0, L], \quad u(0) = 0, \quad u'(L) = E \quad (38)$$

- $V$ : basis  $\{\psi_i\}_{i \in \mathcal{I}_s}$  with  $\psi_i(0) = 0$  because of  $u(0) = 0$
- How does the nonlinear coefficients  $\alpha(u)$  and  $f(u)$  impact the variational formulation?
- (Not much!)

# Nonlinear coefficient; variational formulation

Galerkin: multiply by  $v$ , integrate, integrate by parts

$$\int_0^L \alpha(u) \frac{du}{dx} \frac{dv}{dx} dx = \int_0^L f(u) v dx + [\alpha(u) v u']_0^L \quad \forall v \in V$$

- $\alpha(u(0))v(0)u'(0) = 0$  since  $v(0) = 0$
- $\alpha(u(L))v(L)u'(L) = \alpha(u(L))v(L)E$  since  $u'(L) = E$

$$\int_0^L \alpha(u) \frac{du}{dx} \frac{dv}{dx} dx = \int_0^L f(u) v dx + \alpha(u(L))v(L)E \quad \forall v \in V$$

or

$$(\alpha(u)u', v') = (f(u), v) + \alpha(u(L))v(L)E \quad \forall v \in V$$

# Nonlinear coefficient; where does the nonlinearity cause challenges?

- Abstract notation: no  $a(u, v)$  and  $L(v)$  because  $a$  and  $L$  are nonlinear
- Instead:  $F(u; v) = 0 \quad \forall v \in V$
- What about forming a linear system? We get a *nonlinear* system of algebraic equations
- Must use methods like Picard iteration or Newton's method to solve nonlinear algebraic equations
- But: the variational formulation was not much affected by nonlinearities

# Computing with Dirichlet and Neumann conditions; problem

$$-u''(x) = f(x), \quad x \in \Omega = [0, 1], \quad u'(0) = C, \quad u(1) = D$$

- Use a *global* polynomial basis  $\psi_i \sim x^i$  on  $[0, 1]$
- Because of  $u(1) = D$ :  $\psi_i(1) = 0$
- Basis:  $\psi_i(x) = (1 - x)^{i+1}$ ,  $i \in \mathcal{I}_s$
- $B(x) = Dx$



# Computing with Dirichlet and Neumann conditions; details

$$A_{i,j} = (\psi'_j, \psi'_i) = \int_0^1 \psi'_i(x) \psi'_j(x) dx = \int_0^1 (i+1)(j+1)(1-x)^{i+j} dx,$$

Choose  $f(x) = 2$ :

$$\begin{aligned} b_i &= (2, \psi_i) - (D, \psi'_i) - C\psi_i(0) \\ &= \int_0^1 (2(1-x)^{i+1} - D(i+1)(1-x)^i) dx - C\psi_i(0) \end{aligned}$$

Can easily do the integrals with sympy.  $N = 1$ :

$$\begin{pmatrix} 1 & 1 \\ 1 & 4/3 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} -C + D + 1 \\ 2/3 - C + D \end{pmatrix}$$

$$c_0 = -C + D + 2, \quad c_1 = -1,$$

$$u(x) = 1 - x^2 + D + C(x - 1) \quad (\text{exact solution})$$

## When the numerical method is exact

Assume that apart from boundary conditions,  $u_e$  lies in the same space  $V$  as where we seek  $u$ :

$$u = B + F, \quad F \in V \text{ a}(B + F, v) = L(v) \quad \forall v \in V \quad u_e = B + E, \quad E \in V$$

Subtract:  $a(F - E, v) = 0 \Rightarrow E = F$  and  $u = u_e$

## Tasks:

- Address the model problem  $-u''(x) = 2$ ,  $u(0) = u(L) = 0$
- Uniform finite element mesh with P1 elements
- Show all finite element computations in detail

# Variational formulation, finite element mesh, and basis

$$-u''(x) = 2, \quad x \in (0, L), \quad u(0) = u(L) = 0,$$

Variational formulation:

$$(u', v') = (2, v) \quad \forall v \in V$$

Since  $u(0) = 0$  and  $u(L) = 0$ , we must force

$$v(0) = v(L) = 0, \quad \psi_i(0) = \psi_i(L) = 0$$

Use finite element basis, but exclude  $\varphi_0$  and  $\varphi_{N_n}$  since these are not 0 on the boundary:

$$\psi_i = \varphi_{i+1}, \quad i = 0, \dots, N = N_n - 2$$

Introduce index mapping  $\nu(j)$ :  $\psi_i = \varphi_{\nu(i)}$

$$u = \sum_{j \in \mathcal{I}_s} c_j \varphi_{\nu(j)}, \quad i = 0, \dots, N, \quad \nu(j) = j + 1$$

Irregular numbering: more complicated  $\nu(j)$  table

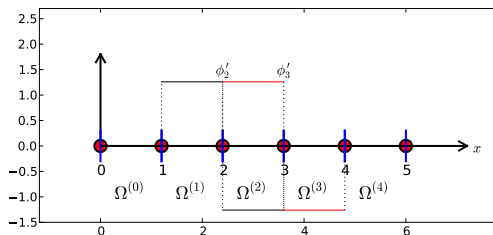
## Computation in the global physical domain; formulas

$$A_{i,j} = \int_0^L \varphi'_{i+1}(x) \varphi'_{j+1}(x) dx, \quad b_i = \int_0^L 2\varphi_{i+1}(x) dx$$

Many will prefer to change indices to obtain a  $\varphi'_i \varphi'_j$  product:  
 $i+1 \rightarrow i, j+1 \rightarrow j$

$$A_{i-1,j-1} = \int_0^L \varphi'_i(x) \varphi'_j(x) dx, \quad b_{i-1} = \int_0^L 2\varphi_i(x) dx$$

# Computation in the global physical domain; details



$$\varphi_i = \pm h^{-1}$$

$$A_{i-1,i-1} = h^{-2}2h = 2h^{-1}, \quad A_{i-1,i-2} = h^{-1}(-h^{-1})h = -h^{-1}, \quad A_{i-1,i} =$$

$$b_{i-1} = 2\left(\frac{1}{2}h + \frac{1}{2}h\right) = 2h$$



# Comparison with a finite difference discretization

- Recall:  $c_i = u(x_{i+1}) \equiv u_{i+1}$
- Write out a general equation at node  $i - 1$ , expressed by  $u_i$

$$-\frac{1}{h}u_{i-1} + \frac{2}{h}u_i - \frac{1}{h}u_{i+1} = 2h \quad (40)$$

The standard finite difference method for  $-u'' = 2$  is

$$-\frac{1}{h^2}u_{i-1} + \frac{2}{h^2}u_i - \frac{1}{h^2}u_{i+1} = 2$$

The finite element method and the finite difference method are identical *in this example*.

(Remains to study the equations involving boundary values)



# Cellwise computations; formulas

- Repeat the previous example, but apply the cellwise algorithm
- Work with one cell at a time
- Transform physical cell to reference cell  $X \in [-1, 1]$

$$A_{i-1,j-1}^{(e)} = \int_{\Omega^{(e)}} \varphi'_i(x) \varphi'_j(x) dx = \int_{-1}^1 \frac{d}{dX} \tilde{\varphi}_r(X) \frac{d}{dX} \tilde{\varphi}_s(X) \frac{h}{2} dX,$$

$$\tilde{\varphi}_0(X) = \frac{1}{2}(1 - X), \quad \tilde{\varphi}_1(X) = \frac{1}{2}(1 + X)$$

$$\frac{d\tilde{\varphi}_0}{dX} = -\frac{1}{2}, \quad \frac{d\tilde{\varphi}_1}{dX} = \frac{1}{2}$$

From the chain rule

$$\frac{d\tilde{\varphi}_r}{dx} = \frac{d\tilde{\varphi}_r}{dX} \frac{dX}{dx} = \frac{2}{h} \frac{d\tilde{\varphi}_r}{dX}$$

## Cellwise computations; details

$$A_{i-1,j-1}^{(e)} = \int_{\Omega^{(e)}} \varphi_i'(x) \varphi_j'(x) dx = \int_{-1}^1 \frac{2}{h} \frac{d\tilde{\varphi}_r}{dX} \frac{2}{h} \frac{d\tilde{\varphi}_s}{dX} \frac{h}{2} dX = \tilde{A}_{r,s}^{(e)}$$

$$b_{i-1}^{(e)} = \int_{\Omega^{(e)}} 2\varphi_i(x) dx = \int_{-1}^1 2\tilde{\varphi}_r(X) \frac{h}{2} dX = \tilde{b}_r^{(e)}, \quad i = q(e, r), \quad r = 0, 1$$

Must run through all  $r, s = 0, 1$  and  $r = 0, 1$  and compute each entry in the element matrix and vector:

$$\tilde{A}^{(e)} = \frac{1}{h} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \quad \tilde{b}^{(e)} = h \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \quad (41)$$

Example:

$$\tilde{A}_{0,1}^{(e)} = \int_{-1}^1 \frac{2}{h} \frac{d\tilde{\varphi}_0}{dX} \frac{2}{h} \frac{d\tilde{\varphi}_1}{dX} \frac{h}{2} dX = \frac{2}{h} \left(-\frac{1}{2}\right) \frac{2}{h} \frac{1}{2} \frac{h}{2} \int_{-1}^1 dX = -\frac{1}{h}$$

## Cellwise computations; details of boundary cells

- The boundary cells involve only one unknown
- $\Omega^{(0)}$ : left node value known, only a contribution from right node
- $\Omega^{(N_e)}$ : right node value known, only a contribution from left node

For  $e = 0$  and  $e = N_e$ :

$$\tilde{A}^{(e)} = \frac{1}{h} \begin{pmatrix} 1 \end{pmatrix}, \quad \tilde{b}^{(e)} = h \begin{pmatrix} 1 \end{pmatrix}$$

Only one degree of freedom ("node") in these cells ( $r = 0$  counts the only dof)

# Cellwise computations; assembly

4 P1 elements:

```
vertices = [0, 0.5, 1, 1.5, 2]
cells = [[0, 1], [1, 2], [2, 3], [3, 4]]
dof_map = [[0], [0, 1], [1, 2], [2]]           # only 1 dof in elm 0, 3
```

Python code for the assembly algorithm:

```
# Ae[e][r,s]: element matrix, be[e][r]: element vector
# A[i,j]: coefficient matrix, b[i]: right-hand side

for e in range(len(Ae)):
    for r in range(Ae[e].shape[0]):
        for s in range(Ae[e].shape[1]):
            A[dof_map[e,r],dof_map[e,s]] += Ae[e][i,j]
            b[dof_map[e,r]] += be[e][i,j]
```

Result: same linear system as arose from computations in the physical domain

# General construction of a boundary function

- Now we address nonzero Dirichlet conditions
- $B(x)$  is not always easy to construct (extend to the interior of  $\Omega$ ), especially not in 2D and 3D
- With finite element  $\varphi_i$ ,  $B(x)$  can be constructed in a completely general way
- $I_b$ : set of indices with nodes where  $u$  is known
- $U_i$ : Dirichlet value of  $u$  at node  $i$ ,  $i \in I_b$

$$B(x) = \sum_{j \in I_b} U_j \varphi_j(x) \quad (42)$$

Suppose we have a Dirichlet condition  $u(x_k) = U_k$ ,  $k \in I_b$ :

$$u(x_k) = \sum_{j \in I_b} U_j \underbrace{\varphi_j(x)}_{\neq 0 \text{ only for } j=k} + \sum_{j \in I_s} c_j \underbrace{\varphi_{\nu(j)}(x_k)}_{=0, k \notin I_s} = U_k$$

## Example with two Dirichlet values; variational formulation

$$-u'' = 2, \quad u(0) = C, \quad u(L) = D$$

$$\int_0^L u' v' \, dx = \int_0^L 2v \, dx \quad \forall v \in V$$

$$(u', v') = (2, v) \quad \forall v \in V$$

## Example with two Dirichlet values; boundary function

$$B(x) = \sum_{j \in I_b} U_j \varphi_j(x) \quad (43)$$

Here  $I_b = \{0, N_n\}$ ,  $U_0 = C$ ,  $U_{N_n} = D$ ,

$$\psi_i = \varphi_{\nu(i)}, \quad \nu(i) = i + 1, \quad i \in \mathcal{I}_s = \{0, \dots, N = N_n - 2\}$$

$$u(x) = C\varphi_0(x) + D\varphi_{N_n}(x) + \sum_{j \in \mathcal{I}_s} c_j \varphi_{\nu(j)} \quad (44)$$

## Example with two Dirichlet values; details

Insert  $u = B + \sum_j c_j \psi_j$  in variational formulation:

$$(u', v') = (2, v) \quad \Rightarrow \quad \left( \sum_j c_j \psi'_j, \psi'_i \right) = (2 - B', \psi_i) \quad \forall v \in V$$

$$\begin{aligned} u(x) &= \underbrace{C \cdot \varphi_0 + D \varphi_{N_n}}_{B(x)} + \sum_{j \in \mathcal{I}_s} c_j \varphi_{j+1} \\ &= C \cdot \varphi_0 + D \varphi_{N_n} + c_0 \varphi_1 + c_1 \varphi_2 + \cdots + c_N \varphi_{N_n-1} \end{aligned}$$

$$A_{i-1,j-1} = \int_0^L \varphi'_i(x) \varphi'_j(x) dx, \quad b_{i-1} = \int_0^L (f(x) - C \varphi'_0(x) - D \varphi'_{N_n}(x)) \varphi_i(x) dx$$

for  $i, j = 1, \dots, N+1 = N_n-1$ .

New boundary terms from  $-\int B' \varphi_i dx$ :  $C/2$  for  $i=1$  and  $-D/2$  for  $i=N_n-1$



## Example with two Dirichlet values; cellwise computations

- Element matrices as in the previous example (with  $u = 0$  on the boundary)
- New element vector in the first and last cell

From the last cell:

$$\tilde{b}_0^{(N_e)} = \int_{-1}^1 \left( f - D \frac{2}{h} \frac{d\tilde{\varphi}_1}{dX} \right) \tilde{\varphi}_0 \frac{h}{2} dX = \left( \frac{h}{2} (2 - D \frac{2}{h} \frac{1}{2}) \right) \int_{-1}^1 \tilde{\varphi}_0 dX = h - D/2$$

From the first cell:

$$\tilde{b}_0^{(0)} = \int_{-1}^1 \left( f - C \frac{2}{h} \frac{d\tilde{\varphi}_0}{dX} \right) \tilde{\varphi}_1 \frac{h}{2} dX = \left( \frac{h}{2} (2 + C \frac{2}{h} \frac{1}{2}) \right) \int_{-1}^1 \tilde{\varphi}_1 dX = h + C/2$$

# Modification of the linear system; ideas

- Method 1: incorporate Dirichlet values through a  $B(x)$  function and demand  $\psi_i = 0$  where Dirichlet values apply
- Method 2: drop  $B(x)$ , drop demands to  $\psi_i$ , just assemble as if there were no Dirichlet conditions, and modify the linear system instead

Method 2: always  $\psi_i = \varphi_i$  and

$$u(x) = \sum_{j \in \mathcal{I}_s} c_j \varphi_j(x), \quad \mathcal{I}_s = \{0, \dots, N = N_n\} \quad (45)$$

Attractive way of incorporating Dirichlet conditions.

$u$  is treated as unknown at all boundaries when computing entires in the linear system

## Modification of the linear system; original system

$$-u'' = 2, \quad u(0) = 0, \quad u(L) = D$$

Assemble as if there were no Dirichlet conditions:

$$\frac{1}{h} \begin{pmatrix} 1 & -1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ -1 & 2 & -1 & \ddots & & & & & \vdots \\ 0 & -1 & 2 & -1 & \ddots & & & & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & 0 & -1 & 2 & -1 & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & & \ddots & \ddots & \ddots & -1 \\ 0 & \dots & \dots & \dots & \dots & \dots & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} h \\ 2h \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 2h \\ h \end{pmatrix} \quad (46)$$

## Modification of the linear system; row replacement

- Dirichlet condition  $u(x_k) = U_k$  means  $c_k = U_k$  (since  $c_k = u(x_k)$ )
- Replace first row by  $c_0 = 0$
- Replace last row by  $c_N = D$

$$\frac{1}{h} \begin{pmatrix} h & 0 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ -1 & 2 & -1 & \ddots & & & & & \vdots \\ 0 & -1 & 2 & -1 & \ddots & & & & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & 0 & -1 & 2 & -1 & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & & \ddots & \ddots & \ddots & -1 \\ 0 & \dots & \dots & \dots & \dots & \dots & 0 & 0 & h \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} 0 \\ 2h \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 2h \\ D \end{pmatrix} \quad (47)$$

## Modification of the linear system; element matrix/vector

In cell 0 we know  $u$  for local node (degree of freedom)  $r = 0$ .  
Replace the first cell equation by  $\tilde{c}_0 = 0$ :

$$\tilde{A}^{(0)} = A = \frac{1}{h} \begin{pmatrix} h & 0 \\ -1 & 1 \end{pmatrix}, \quad \tilde{b}^{(0)} = \begin{pmatrix} 0 \\ h \end{pmatrix} \quad (48)$$

In cell  $N_e$  we know  $u$  for local node  $r = 1$ . Replace the last equation in the cell system by  $\tilde{c}_1 = D$ :

$$\tilde{A}^{(N_e)} = A = \frac{1}{h} \begin{pmatrix} 1 & -1 \\ 0 & h \end{pmatrix}, \quad \tilde{b}^{(N_e)} = \begin{pmatrix} h \\ D \end{pmatrix} \quad (49)$$

# Symmetric modification of the linear system; algorithm

- The modification above destroys symmetry of the matrix:  
e.g.,  $A_{0,1} \neq A_{1,0}$
- Symmetry is often important in 2D and 3D (faster computations)
- A more complex modification can preserve symmetry!

Algorithm for incorporating  $c_i = U_i$  in a symmetric way:

- 1 Subtract column  $i$  times  $U_i$  from the right-hand side
- 2 Zero out column and row no  $i$
- 3 Place 1 on the diagonal
- 4 Set  $b_i = U_i$



Symmetric modification applied to  $\tilde{A}^{(N_e)}$ :

$$\tilde{A}^{(N_e)} = A = \frac{1}{h} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \tilde{b}^{(N-1)} = \begin{pmatrix} h + D/h \\ D \end{pmatrix} \quad (51)$$



### Neumann conditions.

How can we incorporate  $u'(0) = C$  with finite elements?

$$-u'' = f, \quad u'(0) = C, \quad u(L) = D$$

- $\psi_i(L) = 0$  because of Dirichlet condition  $u(L) = D$
- No demand to  $\psi_i(0)$

# The variational formulation

Galerkin's method:

$$\int_0^L (u''(x) + f(x))\psi_i(x)dx = 0, \quad i \in \mathcal{I}_s$$

Integration of  $u''\psi_i$  by parts:

$$\int_0^L u'(x)\psi_i'(x) dx - (u'(L)\psi_i(L) - u'(0)\psi_i(0)) - \int_0^L f(x)\psi_i(x) dx = 0, \quad i \in \mathcal{I}_s$$

- $u'(L)\psi_i(L) = 0$  since  $\psi_i(L) = 0$
- $u'(0)\psi_i(0) = C\psi_i(0)$  since  $u'(0) = C$

## Method 1: Boundary function and exclusion of Dirichlet degrees of freedom

- $\psi_i = \varphi_i, i \in \mathcal{I}_s = \{0, \dots, N = N_n - 1\}$
- $B(x) = D\varphi_{N_n}(x), u = B + \sum_{j=0}^N c_j \varphi_j$

$$\int_0^L u'(x) \varphi_i'(x) dx = \int_0^L f(x) \varphi_i(x) dx - C \varphi_i(0), \quad i \in \mathcal{I}_s$$

$$\sum_{j=0}^{N=N_n-1} \left( \int_0^L \varphi_i'(x) \varphi_j'(x) dx \right) c_j = \int_0^L (f(x) \varphi_i(x) - D \varphi_N'(x) \varphi_i(x)) dx - C$$

(52)

for  $i = 0, \dots, N = N_n - 1$ .

## Method 2: Use all $\varphi_i$ and insert the Dirichlet condition in the linear system

- Now  $\psi_i = \varphi_i$ ,  $i = 0, \dots, N = N_n$
- $\varphi_N(L) \neq 0$ , so  $u'(L)\varphi_N(L) \neq 0$
- However, the term  $u'(L)\varphi_N(L)$  in  $b_N$  will be erased when we insert the Dirichlet value in  $b_N = D$

We can forget about the term  $u'(L)\varphi_i(L)$ !

### Result.

Boundary terms  $u'\varphi_i$  at points  $x_i$  where Dirichlet values apply can always be forgotten.

$$u(x) = \sum_{j=0}^{N=N_n} c_j \varphi_j(x)$$

$$\sum_{j=0}^{N=N_n} \left( \int_0^L \varphi_i'(x) \varphi_j'(x) dx \right) c_j = \int_0^L f(x) \varphi_i(x) \varphi_i(x) dx - C \varphi_i(0) \quad (53)$$

# How the Neumann condition impacts the element matrix and vector

The extra term  $C\varphi_0(0)$  affects only the element vector from the first cells since  $\varphi_0 = 0$  on all other cells.

$$\tilde{A}^{(0)} = A = \frac{1}{h} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad \tilde{b}^{(0)} = \begin{pmatrix} h - C \\ h \end{pmatrix} \quad (54)$$

# The finite element algorithm

The differential equation problem defines the integrals in the variational formulation.

Request these functions from the user:

```
integrand_lhs(phi, r, s, x)
boundary_lhs(phi, r, s, x)
integrand_rhs(phi, r, x)
boundary_rhs(phi, r, x)
```

Must also have a mesh with vertices, cells, and dof\_map

# Python pseudo code; the element matrix and vector

```
<Declare global matrix, global rhs: A, b>

# Loop over all cells
for e in range(len(cells)):

    # Compute element matrix and vector
    n = len(dof_map[e]) # no of dofs in this element
    h = vertices[cells[e][1]] - vertices[cells[e][0]]
    <Declare element matrix, element vector: A_e, b_e>

    # Integrate over the reference cell
    points, weights = <numerical integration rule>
    for X, w in zip(points, weights):
        phi = <basis functions + derivatives at X>
        detJ = h/2
        x = <affine mapping from X>
        for r in range(n):
            for s in range(n):
                A_e[r,s] += integrand_lhs(phi, r, s, x)*detJ*w
                b_e[r] += integrand_rhs(phi, r, x)*detJ*w

    # Add boundary terms
    for r in range(n):
        for s in range(n):
            A_e[r,s] += boundary_lhs(phi, r, s, x)*detJ*w
            b_e[r] += boundary_rhs(phi, r, x)*detJ*w
```

# Python pseudo code; boundary conditions and assembly

```
for e in range(len(cells)):
    ...

    # Incorporate essential boundary conditions
    for r in range(n):
        global_dof = dof_map[e][r]
        if global_dof in essbc_dofs:
            # dof r is subject to an essential condition
            value = essbc_docs[global_dof]
            # Symmetric modification
            b_e -= value*A_e[:,r]
            A_e[r,:] = 0
            A_e[:,r] = 0
            A_e[r,r] = 1
            b_e[r] = value

    # Assemble
    for r in range(n):
        for s in range(n):
            A[dof_map[e][r], dof_map[e][r]] += A_e[r,s]
            b[dof_map[e][r]] += b_e[r]

<solve linear system>
```



# Variational formulations in 2D and 3D

How to do integration by parts is the major difference when moving to 2D and 3D.

Rule for multi-dimensional integration by parts.

$$-\int_{\Omega} \nabla \cdot (a(\mathbf{x}) \nabla u) v \, d\mathbf{x} = \int_{\Omega} a(\mathbf{x}) \nabla u \cdot \nabla v \, d\mathbf{x} - \int_{\partial\Omega} a \frac{\partial u}{\partial n} v \, ds \quad (55)$$

- $\int_{\Omega} () \, d\mathbf{x}$ : area (2D) or volume (3D) integral
- $\int_{\partial\Omega} () \, ds$ : line(2D) or surface (3D) integral
- $\partial\Omega_N$ : Neumann conditions  $-a \frac{\partial u}{\partial n} = g$
- $\partial\Omega_D$ : Dirichlet conditions  $u = u_0$
- $v \in V$  must vanish on  $\partial\Omega_D$  (in method 1)

## Example on integration by parts; problem

$$\mathbf{v} \cdot \nabla u + \alpha u = \nabla \cdot (a \nabla u) + f, \quad \mathbf{x} \in \Omega \quad (56)$$

$$u = u_0, \quad \mathbf{x} \in \partial\Omega_D \quad (57)$$

$$-a \frac{\partial u}{\partial n} = g, \quad \mathbf{x} \in \partial\Omega_N \quad (58)$$

- Known:  $a$ ,  $\alpha$ ,  $f$ ,  $u_0$ , and  $g$ .
- Second-order PDE: must have *exactly one boundary condition at each point of the boundary*

Method 1 with boundary function and  $\psi_i = 0$  on  $\partial\Omega_D$ :

$$u(\mathbf{x}) = B(\mathbf{x}) + \sum_{j \in \mathcal{I}_s} c_j \psi_j(\mathbf{x}), \quad B(\mathbf{x}) = u_0(\mathbf{x})$$

## Example on integration by parts; details (1)

Galerkin's method: multiply by  $v \in V$  and integrate over  $\Omega$ ,

$$\int_{\Omega} (\mathbf{v} \cdot \nabla u + \alpha u) v \, dx = \int_{\Omega} \nabla \cdot (a \nabla u) \, dx + \int_{\Omega} f v \, dx$$

Integrate second-order term by parts:

$$\int_{\Omega} \nabla \cdot (a \nabla u) v \, dx = - \int_{\Omega} a \nabla u \cdot \nabla v \, dx + \int_{\partial\Omega} a \frac{\partial u}{\partial n} v \, ds,$$

Resulting variational form:

$$\int_{\Omega} (\mathbf{v} \cdot \nabla u + \alpha u) v \, dx = - \int_{\Omega} a \nabla u \cdot \nabla v \, dx + \int_{\partial\Omega} a \frac{\partial u}{\partial n} v \, ds + \int_{\Omega} f v \, dx$$

## Example on integration by parts; details (2)

Note:  $v \neq 0$  only on  $\partial\Omega_N$ :

$$\int_{\partial\Omega} a \frac{\partial u}{\partial n} v \, ds = \int_{\partial\Omega_N} \underbrace{a \frac{\partial u}{\partial n}}_{-g} v \, ds = - \int_{\partial\Omega_N} g v \, ds$$

The final variational form:

$$\int_{\Omega} (\mathbf{v} \cdot \nabla u + \alpha u) v \, dx = - \int_{\Omega} a \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega_N} g v \, ds + \int_{\Omega} f v \, dx$$

Or with inner product notation:

$$(\mathbf{v} \cdot \nabla u, v) + (\alpha u, v) = -(a \nabla u, \nabla v) - (g, v)_N + (f, v)$$

$(g, v)_N$ : line or surface integral over  $\partial\Omega_N$ .

## Example on integration by parts; linear system

$$u = B + \sum_{j \in \mathcal{I}_s} c_j \psi_j, \quad B = u_0$$

$$A_{i,j} = (\mathbf{v} \cdot \nabla \psi_j, \psi_i) + (\alpha \psi_j, \psi_i) + (a \nabla \psi_j, \nabla \psi_i)$$

$$b_i = (g, \psi_i)_N + (f, \psi_i) - (\mathbf{v} \cdot \nabla u_0, \psi_i) + (\alpha u_0, \psi_i) + (a \nabla u_0, \nabla \psi_i)$$

# Transformation to a reference cell in 2D/3D (1)

We want to compute an integral in the physical domain by integrating over the reference cell.

$$\int_{\Omega^{(e)}} a(\mathbf{x}) \nabla \varphi_i \cdot \nabla \varphi_j \, d\mathbf{x} \quad (59)$$

Mapping from reference to physical coordinates:

$$\mathbf{x}(\mathbf{X})$$

with Jacobian  $J$ ,

$$J_{i,j} = \frac{\partial x_j}{\partial X_i}$$

- $d\mathbf{x} \rightarrow \det J \, d\mathbf{X}$ .
- Must express  $\nabla \varphi_i$  by an expression with  $\tilde{\varphi}_r$ ,  $i = q(e, r)$ :  
 $\nabla \tilde{\varphi}_r(\mathbf{X})$
- We want  $\nabla_{\mathbf{x}} \tilde{\varphi}_r(\mathbf{X})$  (derivatives wrt  $\mathbf{x}$ )
- What we readily have is  $\nabla_{\mathbf{X}} \tilde{\varphi}_r(\mathbf{X})$  (derivative wrt  $\mathbf{X}$ )

## Transformation to a reference cell in 2D/3D (2)

Can derive

$$\begin{aligned}\nabla_{\mathbf{X}} \tilde{\varphi}_r &= J \cdot \nabla_{\mathbf{x}} \varphi_i \\ \nabla_{\mathbf{x}} \varphi_i &= \nabla_{\mathbf{x}} \tilde{\varphi}_r(\mathbf{X}) = J^{-1} \cdot \nabla_{\mathbf{X}} \tilde{\varphi}_r(\mathbf{X})\end{aligned}$$

Integral transformation from physical to reference coordinates:

$$\int_{\Omega^{(e)}} a(\mathbf{x}) \nabla_{\mathbf{x}} \varphi_i \cdot \nabla_{\mathbf{x}} \varphi_j \, d\mathbf{x} = \int_{\tilde{\Omega}_r} a(\mathbf{x}(\mathbf{X})) (J^{-1} \cdot \nabla_{\mathbf{X}} \tilde{\varphi}_r) \cdot (J^{-1} \cdot \nabla_{\mathbf{X}} \tilde{\varphi}_s) \det J \, d\mathbf{X} \quad (60)$$



# Numerical integration

Numerical integration over reference cell triangles and tetrahedra:

$$\int_{\tilde{\Omega}^r} g \, dX = \sum_{j=0}^{n-1} w_j g(\bar{\mathbf{X}}_j)$$

Module `numint.py` contains different rules:

```
>>> import numint
>>> x, w = numint.quadrature_for_triangles(num_points=3)
>>> x
[(0.16666666666666666, 0.16666666666666666),
 (0.6666666666666666, 0.16666666666666666),
 (0.16666666666666666, 0.6666666666666666)]
>>> w
[0.16666666666666666, 0.16666666666666666, 0.16666666666666666]
```

- Triangle: rules with  $n = 1, 3, 4, 7$  integrate exactly polynomials of degree 1, 2, 3, 4, resp.
- Tetrahedron: rules with  $n = 1, 4, 5, 11$  integrate exactly polynomials of degree 1, 2, 3, 4, resp.

# Time-dependent problems

- So far: used the finite element framework for discretizing in space
- What about  $u_t = u_{xx} + f$ ?
  - 1 Use finite differences in time to obtain a set of recursive spatial problems
  - 2 Solve the spatial problems by the finite element method

## Example: diffusion problem

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega, t \in (0, T] \quad (61)$$

$$u(\mathbf{x}, 0) = l(\mathbf{x}), \quad \mathbf{x} \in \Omega \quad (62)$$

$$\frac{\partial u}{\partial n} = 0, \quad \mathbf{x} \in \partial\Omega, t \in (0, T] \quad (63)$$

## A Forward Euler scheme; ideas

$$[D_t^+ u = \alpha \nabla^2 u + f(\mathbf{x}, t)]^n, \quad n = 1, 2, \dots, N_t - 1 \quad (64)$$

$$u^{n+1} = u^n + \Delta t (\alpha \nabla^2 u^n + f(\mathbf{x}, t_n)) \quad (65)$$

- $u^n = \sum_j c_j^n \psi_j$
- Compute  $u^0$  from  $I$
- Compute  $u^{n+1}$  from  $u^n$

## A Forward Euler scheme; stages in the discretization

- $u_e(\mathbf{x}, t)$ : exact solution of the space-and time-continuous problem
- $u_e^n(\mathbf{x})$ : exact solution of time-discrete problem (after applying a finite difference scheme in time)
- $u^n = \sum_{j \in \mathcal{I}_s} c_j^n \psi_j \approx u_e^n(\mathbf{x})$ : solution at the time- and space-discrete problem (after applying a Galerkin method in space)

$$\frac{\partial u_e}{\partial t} = \alpha \nabla^2 u_e + f(\mathbf{x}, t) \quad (66)$$

$$u_e^{n+1} = u_e^n + \Delta t (\alpha \nabla^2 u_e^n + f(\mathbf{x}, t_n)) \quad (67)$$

$$u_e^n \approx u^n = \sum_{j=0}^{N_s} c_j^n \psi_j(\mathbf{x}), \quad u_e^{n+1} \approx u^{n+1} = \sum_{j=0}^{N_s} c_j^{n+1} \psi_j(\mathbf{x})$$

$$R = u^{n+1} - u^n - \Delta t (\alpha \nabla^2 u^n + f(\mathbf{x}, t_n))$$

# A Forward Euler scheme; weighted residual (or Galerkin) principle

$$R = u^{n+1} - u^n - \Delta t (\alpha \nabla^2 u^n + f(\mathbf{x}, t_n))$$

The weighted residual principle:

$$\int_{\Omega} R w_i \, dx = 0, \quad i = 0, \dots, N_s,$$

results in

$$\int_{\Omega} [u^{n+1} - u^n - \Delta t (\alpha \nabla^2 u^n + f(\mathbf{x}, t_n))] w_i \, dx = 0, \quad i = 0, \dots, N$$

Galerkin:  $w_i = \psi_i$

## A Forward Euler scheme; integration by parts

Isolating the unknown  $u^{n+1}$  on the left-hand side:

$$\int_{\Omega} u^{n+1} \psi_i \, dx = \int_{\Omega} [u^n - \Delta t (\alpha \nabla^2 u^n + f(\mathbf{x}, t_n))] \psi_i \, dx$$

Integration by parts:

$$\int_{\Omega} \alpha \nabla^2 u^n \psi_i \, dx = - \int_{\Omega} \alpha \nabla u^n \cdot \nabla \psi_i \, dx + \underbrace{\int_{\partial\Omega} \alpha \frac{\partial u^n}{\partial n} \psi \, dx}_{=0 \text{ because } \partial u^n / \partial n = 0}$$

$$\int_{\Omega} u^{n+1} \psi_i \, dx = \int_{\Omega} u^n \psi_i \, dx - \Delta t \int_{\Omega} \alpha \nabla u^n \cdot \nabla \psi_i \, dx + \Delta t \int_{\Omega} f^n \psi_i \, dx \quad (68)$$

# New notation for the solution at the most recent time levels

- $u$  and  $u$ : the spatial unknown function to be computed
- $u_1$  and  $u_1$ : the spatial function at the previous time level  $t - \Delta t$
- $u_2$  and  $u_2$ : the spatial function at  $t - 2\Delta t$
- This new notation gives close correspondance between code and math

$$\int_{\Omega} u \psi_i \, dx = \int_{\Omega} u_1 \psi_i \, dx - \Delta t \int_{\Omega} \alpha \nabla u_1 \cdot \nabla \psi_i \, dx + \Delta t \int_{\Omega} f^n \psi_i \, dx \quad (69)$$

or

$$(u, \psi_i) = (u_1, \psi_i) - \Delta t (\alpha \nabla u_1, \nabla \psi_i) + (f^n, \psi_i) \quad (70)$$



# Deriving the linear systems

$$u = \sum_{j=0}^N c_j \psi_j(\mathbf{x}), \quad u_1 = \sum_{j=0}^N c_{1,j} \psi_j(\mathbf{x})$$

Insert these in

$$(u, \psi_i) = (u_1, \psi_i) - \Delta t (\alpha \nabla u_1, \nabla \psi_i) + (f^n, \psi_i)$$

and order terms as matrix-vector products:

$$\sum_{j=0}^N \underbrace{(\psi_i, \psi_j)}_{M_{i,j}} c_j = \sum_{j=0}^N \underbrace{(\psi_i, \psi_j)}_{M_{i,j}} c_{1,j} - \Delta t \sum_{j=0}^N \underbrace{(\nabla \psi_i, \alpha \nabla \psi_j)}_{K_{i,j}} c_{1,j} + (f^n, \psi_i), \quad i =$$

(71)

$$Mc = Mc_1 - \Delta t K c_1 + f \quad (72)$$

$$M = \{M_{i,j}\}, \quad M_{i,j} = (\psi_i, \psi_j), \quad i, j \in \mathcal{I}_s$$

$$K = \{K_{i,j}\}, \quad K_{i,j} = (\nabla \psi_i, \alpha \nabla \psi_j), \quad i, j \in \mathcal{I}_s$$

$$f = \{(f(\mathbf{x}, t_n), \psi_i)\}_{i \in \mathcal{I}_s}$$

$$c = \{c_i\}_{i \in \mathcal{I}_s}$$

$$c_1 = \{c_{1,i}\}_{i \in \mathcal{I}_s}$$

# Computational algorithm

- ➊ Compute  $M$  and  $K$ .
- ➋ Initialize  $u^0$  by either interpolation or projection
- ➌ For  $n = 1, 2, \dots, N_t$ :
  - ➊ compute  $b = Mc_1 - \Delta t Kc_1 + f$
  - ➋ solve  $Mc = b$
  - ➌ set  $c_1 = c$

Initial condition:

- Interpolation:  $c_{1,j} = I(\mathbf{x}_j)$  (finite elements)
- Projection: solve  $\sum_j M_{i,j} c_{1,j} = (I, \psi_i)$ ,  $i \in \mathcal{I}_s$ .

# Comparing P1 elements with the finite difference method; ideas

- P1 elements in 1D
- Uniform mesh on  $[0, L]$  with cell length  $h$
- No Dirichlet conditions:  $\psi_i = \varphi_i$ ,  $i = 0, \dots, N = N_n$
- Have found formulas for  $M$  and  $K$  at the element level
- Have assembled the global matrices
- Have developed corresponding finite difference operator formulas

# Comparing P1 elements with the finite difference method; results

Diffusion equation with finite elements is equivalent to

$$[D_t^+(u - \frac{1}{6}h^2 D_x D_x u) = \alpha D_x D_x u + f]_i^n \quad (73)$$

Can lump the mass matrix by Trapezoidal integration and get the standard finite difference scheme

$$[D_t^+ u = \alpha D_x D_x u + f]_i^n \quad (74)$$

# Discretization in time by a Backward Euler scheme

Backward Euler scheme in time:

$$[D_t^- u = \alpha \nabla^2 u + f(\mathbf{x}, t)]^n.$$

$$u_e^n - \Delta t (\alpha \nabla^2 u_e^n + f(\mathbf{x}, t_n)) = u_e^{n-1} \quad (75)$$

$$u_e^n \approx u^n = \sum_{j=0}^N c_j^n \psi_j(\mathbf{x}), \quad u_e^{n+1} \approx u^{n+1} = \sum_{j=0}^N c_j^{n+1} \psi_j(\mathbf{x})$$

# The variational form of the time-discrete problem

$$\int_{\Omega} (u^n \psi_i + \Delta t \alpha \nabla u^n \cdot \nabla \psi_i) \, dx = \int_{\Omega} u^{n-1} \psi_i \, dx - \Delta t \int_{\Omega} f^n \psi_i \, dx \quad (76)$$

or

$$(u, \psi_i) + \Delta t \alpha (\nabla u, \nabla \psi_i) = (u_1 \psi_i) + \Delta t (f^n, \psi_i) \quad (77)$$

The linear system: insert  $u = \sum_j c_j \psi_j$  and  $u_1 = \sum_j c_{1,j} \psi_j$ ,

$$(M + \Delta t \alpha K)c = Mc_1 + f \quad (78)$$

Can interpret the resulting equation system as

$$[D_t^-(u - \frac{1}{6}h^2 D_x D_x u) = \alpha D_x D_x u + f]_i^n \quad (79)$$

Lumped mass matrix (by Trapezoidal integration):

$$[D_t^- u = \alpha D_x D_x u + f]_i^n \quad (80)$$

(Standard finite difference method)



# Dirichlet boundary conditions

Dirichlet condition at  $x = 0$  and Neumann condition at  $x = L$ :

$$u(\mathbf{x}, t) = u_0(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega_D, \quad (81)$$

$$-\alpha \frac{\partial}{\partial n} u(\mathbf{x}, t) = g(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega_N \quad (82)$$

Forward Euler in time:

$$\int_{\Omega} u^{n+1} v \, dx = \int_{\Omega} (u^n - \Delta t \alpha \nabla u^n \cdot \nabla v) \, dx - \Delta t \int_{\partial\Omega_N} g v \, ds, \quad \forall v \in V \quad (83)$$

# Boundary function

$$u^n(\mathbf{x}) = u_0(\mathbf{x}, t_n) + \sum_{j \in \mathcal{I}_s} c_j^n \psi_j(\mathbf{x})$$

$$\begin{aligned} \sum_{j \in \mathcal{I}_s} \left( \int_{\Omega} \psi_i \psi_j \, d\mathbf{x} \right) c_j^{n+1} = & \sum_{j \in \mathcal{I}_s} \left( \int_{\Omega} (\psi_i \psi_j - \Delta t \alpha \nabla \psi_i \cdot \nabla \psi_j) \, d\mathbf{x} \right) c_j^n - \\ & \int_{\Omega} (u_0(\mathbf{x}, t_{n+1}) - u_0(\mathbf{x}, t_n) + \Delta t \alpha \nabla u_0(\mathbf{x}, t_n) \cdot \nabla \psi_i) \, d\mathbf{x} \\ & + \Delta t \int_{\Omega} f \psi_i \, d\mathbf{x} - \Delta t \int_{\partial\Omega_N} g \psi_i \, d\mathbf{s}, \quad i \in \mathcal{I}_s \end{aligned}$$

!split ===== Finite element basis functions =====

\*  $B(\mathbf{x}, t_n) = \sum_{j \in I_b} U_j^n \varphi_j$  \*  $\psi_i = \varphi_{\nu(j)}$ ,  $j \in \mathcal{I}_s$  \*  $\nu(j)$ ,  $j \in \mathcal{I}_s$ , are the node numbers corresponding to all nodes without a Dirichlet condition

$$u^n = \sum_{j \in I_b} U_j^n \varphi_j + \sum_{j \in \mathcal{I}_s} c_{1,j} \varphi_{\nu(j)},$$

$$u^{n+1} = \sum_{j \in I_b} U_j^{n+1} \varphi_j + \sum_{j \in \mathcal{I}_s} c_{1,j} \varphi_{\nu(j)}$$

# Modification of the linear system (1)

- Drop boundary function
- Compute as if there are not Dirichlet conditions
- Modify the linear system to incorporate Dirichlet conditions
- $\mathcal{I}_s$  holds the indices of all nodes  $\{0, 1, \dots, N = N_n\}$

$$\sum_{j \in \mathcal{I}_s} \left( \underbrace{\int_{\Omega} \varphi_i \varphi_j \, dx}_{M_{i,j}} \right) c_j = \sum_{j \in \mathcal{I}_s} \left( \underbrace{\int_{\Omega} \varphi_i \varphi_j \, dx}_{M_{i,j}} - \Delta t \underbrace{\int_{\Omega} \alpha \nabla \varphi_i \cdot \nabla \varphi_j \, dx}_{K_{i,j}} \right) c_{1,j} \\ \underbrace{- \Delta t \int_{\Omega} f \varphi_i \, dx - \Delta t \int_{\partial \Omega_N} g \varphi_i \, ds}_{f_i}$$

## Modification of the linear system (2)

$$Mc = n, \quad b = Mc_1 - \Delta t K c_1 + \Delta t f. \quad (84)$$

For each  $k$  where a Dirichlet condition applies:  $u(x_k, t_{n+1}) = U_k^{n+1}$

- Set row  $k$  in  $M$  to zero and 1 on the diagonal:  $M_{k,j} = 0$ ,  
 $j \in \mathcal{I}_s$ ,  $M_{k,k} = 1$
- $b_k = U_k^{n+1}$

## Modification of the linear system (3)

Backward Euler discretization in time:

$$Ac = b, \quad A = M + \Delta t K, \quad b = Mc_1 + \Delta t f. \quad (85)$$

- Set row  $k$  to zero and 1 on the diagonal:  $M_{k,j} = 0, j \in \mathcal{I}_s$ ,  
 $M_{k,k} = 1$
- Set row  $k$  to zero:  $K_{k,j} = 0, j \in \mathcal{I}_s$
- $b_k = U_k^{n+1}$

Observe:  $A_{k,k} = M_{k,k} + \Delta t K_{k,k} = 1 + 0$ , so  $c_k = U_k^{n+1}$

# Analysis of the discrete equations

The diffusion equation  $u_t = \alpha u_{xx}$  allows a (Fourier) wave component

$$u = A_e^n e^{ikx}, \quad A_e = e^{-\alpha k^2 \Delta t} \quad (86)$$

Numerical schemes often allow the similar solution

$$u_q^n = A^n e^{ikx} \quad (87)$$

$A$ : amplification factor to be computed

# Amplification factor for the Forward Euler method; results

$$p = k\Delta x/2 \text{ and } C = \alpha\Delta t/\Delta x^2:$$

$$A = 1 - 4C \frac{\sin^2 p}{1 + \underbrace{\frac{2}{3} \sin^2 p}_{\text{from } M}}$$

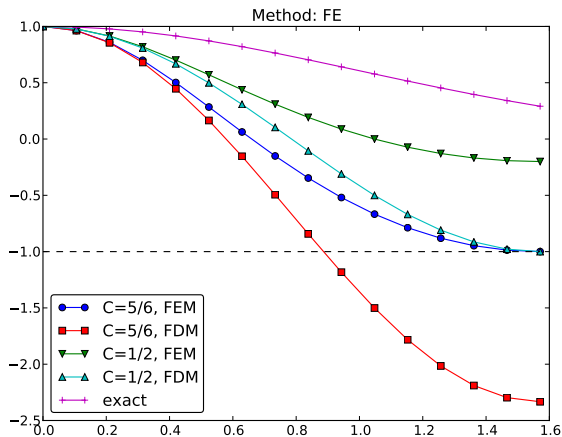
(See notes for details)

Stability:  $-1 \geq A$ :

$$C \leq \frac{5}{6} \quad \Rightarrow \quad \Delta t \leq \frac{5\Delta x^2}{6\alpha} \quad (88)$$

Finite differences:  $C \leq \frac{1}{2}$

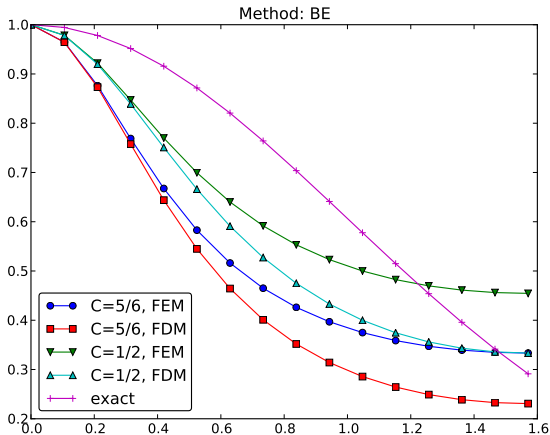
# Amplification factor for the Forward Euler method; plot



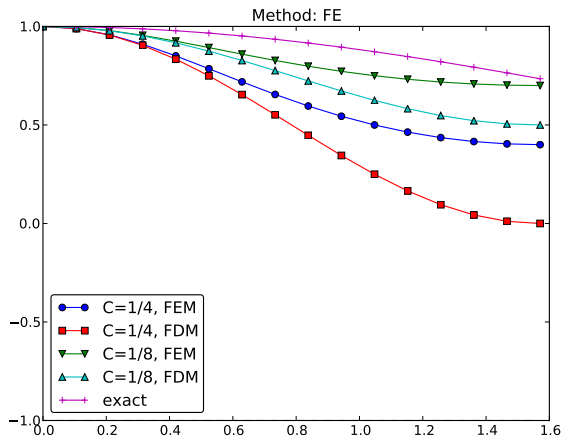


# Amplification factor for the Backward Euler method; results

$$A = \left( 1 + 4C \frac{\sin^2 p}{1 + \frac{2}{3} \sin^2 p} \right)^{-1} \quad (\text{unconditionally stable})$$



# Amplification factors for smaller time steps; Forward Euler



# Amplification factors for smaller time steps; Backward Euler

