Spatio-Temporal Modeling in Biology

Dagmar Iber, Roberto Croce

Computational Biology (CoBI), D-BSSE, ETHZ

Outline

- Key Technology: Numerical Simulation
- The Finite Difference Method (FDM)
- The Finite Element Method (FEM)
- Some software packages

Problems with experiments of prototypes

- Very expensive experiments:
  - Construction of airplanes
  - Construction of skyskrapers
- Very dangerous experiments:
  - The combustion processes in power stations
  - Explosive and radioactive materials
  - Viruses and poisons
- Experments not realizable, because of scale:
  - Galaxy collisions
  - Nano materials
  - Genetic engineering

Numerical Simulation as Key Technology

- Expensive experiments → Simulations are less expensive
- Dangerous experiments → Simulations are safe
- Experiments not possible, because of scale → Problems of scale do not exist for numerical simulations

- Scientific Computing combines the modeling-techniques of engineers with the numerical methods of mathematicians and the high performance computing of computer scientists together in an applied interdisciplinary research field.

Why solving PDEs numerically?

- Only very few PDEs can be solved directly.
- Therefore, we solve the PDE approximately on a machine.
- Finite Difference Methods (FDM) and Finite Element Methods (FEM) are two of the main choices for this kind of problems.
- FDM is easy to implement, but hard to use on complex geometries.
- FEM is strong and fits complicated geometries very well, but more involved to implement.

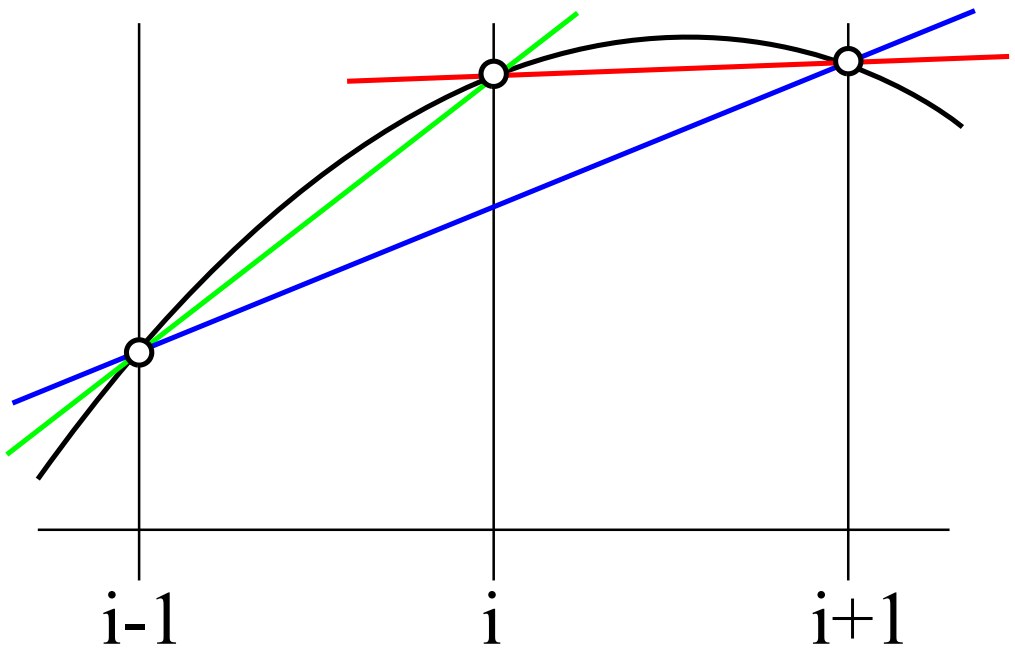
Numerical Methods

Hence, let's start with the approach of solutions via

Finite Difference Methods (FDM)

Finite Difference Methods

- one-sided approximation
- centered approximation



Finite Difference Methods

- one-sided:

$$D_+u(x_i) = \frac{u(x_i + h) - u(x_i)}{h} \tag{1}$$

$$D_-u(x_i) = \frac{u(x_i) - u(x_i - h)}{h} \tag{2}$$

- centered approximation:

$$D_0u(x_i) = \frac{u(x_i + h) - u(x_i - h)}{2h} = \frac{1}{2}\left(D_+u(x_i) - D_-u(x_i)\right) \tag{3}$$

- third order approximation:

$$D_3u(x_i) = \frac{1}{6h}\left(2u(x_i + h) + 3u(x_i) - 6u(x_i - h) + u(x_i - 2h)\right) \tag{4}$$

2D Elliptic Equations

- $a_1 u_{xx} + a_2 u_{xy} + a_3 u_{yy} + a_4 u_x + a_5 u_y + a_6 u = f$
- $a_2^2 - 4 a_1 a_3 < 0$
- example: Poisson equation (diffusion)

$$u_{xx} + u_{yy} = f \tag{5}$$

- IC: $u(x, y, 0) = u_0(x, y, 0)$, BC: $u(x, y, t)\ \{x, y\} \in \partial\Omega$
- centerd differences:

$$\frac{1}{h_x^2}\left[u_{i-1,j} - 2u_{ij} + u_{i+1,j}\right] + \frac{1}{h_y^2}\left[u_{i,j-1} - 2u_{ij} + u_{i,j+1}\right] = f_{ij} \tag{6}$$

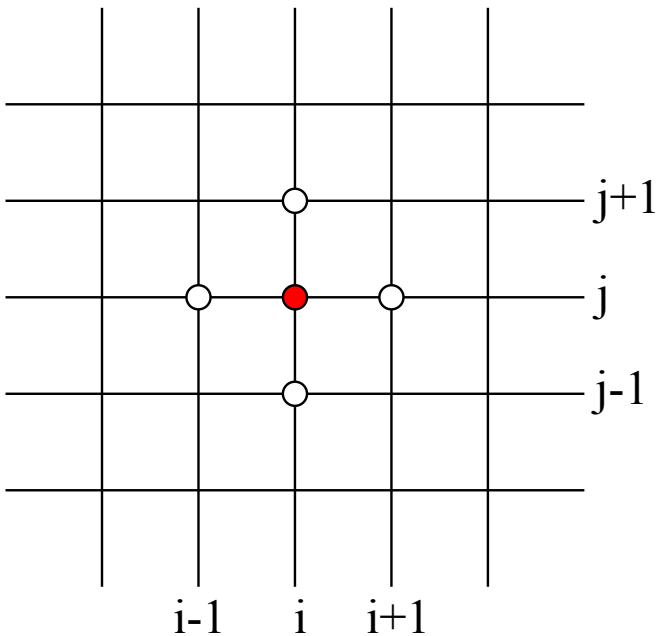
$$\frac{1}{h^2}\left[u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}\right] = f_{ij} \tag{7}$$

- lexicographic ordering: $u = [u_{11} \ldots u_{m1}, u_{12} \ldots u_{m2}, \ldots, u_{1m} \ldots u_{mm}]$

2D Elliptic Equations

- centered differences:

$$\nabla_h^2 u(x) = \frac{1}{h^2}\left[u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + j_{i,j+1} - 4u_{ij}\right] \tag{8}$$



2D Elliptic Equations

- the system now looks like:

$$A = \begin{bmatrix} T & I & & & \\ I & T & I & & \\ & I & T & I & \\ & & & \cdots & \cdots \end{bmatrix} \tag{9}$$

- with

$$T = \begin{bmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & 1 & -4 & 1 & \\ & & & \cdots & \cdots \end{bmatrix} \tag{10}$$

Initial Value Problem for ODE

- example:

$$u'(t) = f(u(t), t) \qquad \text{IC: } u(t_0) = \eta \tag{11}$$

- very simple: forward Euler

$$\frac{u^{n+1} - u^n}{\Delta t} = f(u^n) \qquad O(\Delta t) \tag{12}$$

- less simple: backward Euler

$$\frac{u^{n+1} - u^n}{\Delta t} = f(u^{n+1}) \qquad O(\Delta t) \tag{13}$$

- less simple: trapezoidal method

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{1}{2}\left[f(u^n) + f(u^{n+1})\right] \qquad O(\Delta t^2) \tag{14}$$

Diffusion Equation (parabolic)

$$u_t = u_{xx} \tag{15}$$

- IC: $u(x,0) = \eta(x)$, BC: $u(0,t) = g_0(t)$, $u(1,t) = g_1(t)$
- discretization: $x_i = i \cdot \Delta x$, $t_n = n \cdot \Delta t$
- very simple: centered difference in space, forware Euler in time.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{h^2}\left[u_{i-1}^n - 2u_i^n + u_{i+1}^n\right] \qquad \frac{\Delta t}{h^2} < \frac{1}{2} \tag{16}$$

- Crank-Nicholson:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2}\left[D^2 u_i^n + D^2 u_i^{n+1}\right] \tag{17}$$

$$\cdots = \frac{1}{2h^2}\left[u_{i-1}^n - 2u_i^n + u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}\right] \tag{18}$$

Crank-Nicholson

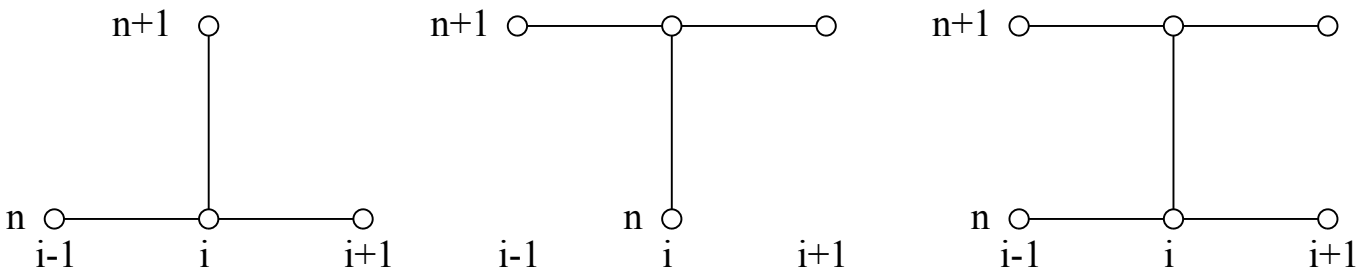
$$-ru_{i-1}^{n+1} + (1+2r)u_i^{n+1} - ru_{i+1}^{n+1} = ru_{i-1}^n + (1-2r)u_i^n + ru_{i+1}^n \qquad r = \frac{\Delta t}{2h^2} \tag{19}$$

$$\begin{bmatrix} (1+2r) & -r & & \\ -r & (1+2r) & r & \\ & -r & (1+2r) & -r \\ & & & \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \cdots \end{bmatrix} = \begin{bmatrix} r(g_0^n + g_0^{n+1}) + (1+2r)u_1^n + ru_2^n \\ ru_1^n + (1+2r)u_2^n + ru_3^n \\ \cdots \\ \end{bmatrix} \tag{20}$$

- solve this system in $O(m)$

Stencils for discretization

stencil of forward Euler, backward Euler, Crank-Nicholson:



Introduction FEM

FEM is a numerical method to solve boundary value problems.

Idea:

- Find the weak formulation of the problem
- Divide the domain into subdomains (meshing the domain)
- Choose basic functions for the subdomains
- Formulate a system of linear equations and solve it

Solving an ODE in 1D with FEM

Let's consider the following ODE:

$$\frac{d^2u}{dx^2} + 1 = 0 \quad , \quad x \in \Omega \tag{21}$$

$$u(x) = 0 \quad , \quad x \in \partial\Omega \tag{22}$$

where $\Omega = [0, 1] \subset \mathbb{R}$.

Exact Solution

The ODE can be solved analytically by using the ansatz $u(x) = ax^2 + bx + c$.

$$\begin{aligned} \frac{d^2u}{dx^2} = 2a = -1 \quad &\Rightarrow \quad a = -\frac{1}{2}. \\ u(0) = 0 \quad &\Rightarrow \quad c = 0. \\ u(1) = 0 \quad &\Rightarrow \quad a + b = 0 \Rightarrow b = \frac{1}{2}. \end{aligned}$$

FEM: Weak formulation

Multiplying the differential equation by a test function v and integrating over the domain gives the weak formulation of the problem.

$$\int_\Omega v \left(\frac{d^2 u}{dx^2} + 1 \right) dx = 0$$

FEM: Weak formulation

Because v also has to fulfill the BC's integration by parts yields:

$$\int_\Omega v \frac{d^2u}{dx^2} dx = \left[\frac{du}{dx} v\right]_0^1 - \int_\Omega \frac{du}{dx}\frac{dv}{dx} dx = -\int_\Omega \frac{du}{dx}\frac{dv}{dx} dx.$$

Strong formulation

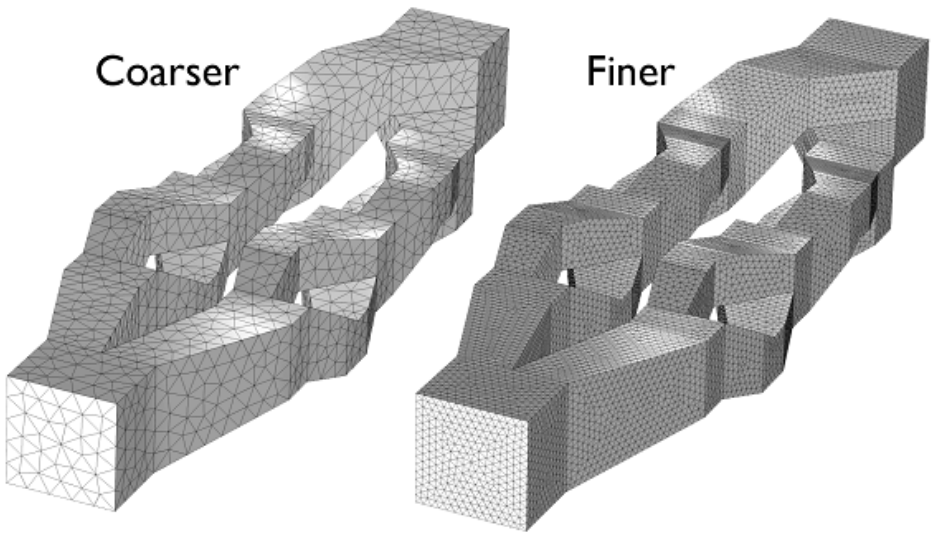
$$\frac{d^2u}{dx^2} + 1 = 0$$

Weak formulation

$$-\int_\Omega \frac{du}{dx}\frac{dv}{dx} dx + \int_\Omega v dx = 0$$

FEM: Discretization

Discretization of the domain. This is obvious in 1D, but much more demanding in 2D or 3D.



In our case we divide the domain into N equal subdomains.

Basis Functions

- We choose a basis $\{\phi_p^{(h)}\}_{p \in N_h}$ of V_h. Here, N_h with $|N_h| < \infty$ is the index set representing representing the discretization points in the domain Ω
- Then, with the assumption $u_h = \sum_{p \in N_h} u_p \phi_p^{(h)}$, we get:

$$
\begin{aligned}
& a(u_h, v) = (f, v) && \text{for all } v \in V_h \\
\Leftrightarrow \quad & a(u_h, \phi_p^{(h)}) = (f, \phi_p^{(h)}) && \text{for all } p \in N_h \\
\Leftrightarrow \quad & \textstyle\sum_{q \in N_h} u_q a(\phi_q^{(h)}, \phi_p^{(h)}) = (f, \phi_p^{(h)}) && \text{for all } p \in N_h
\end{aligned}
$$

- We define $\mathbf{A} = (a_{pq})_{p,q \in N_h}$, $\mathbf{f} = (f_q)_{q \in N_h}$ with the entries $f_q = (f, \phi_q^{(h)}) = \int_\Omega f \phi_q^{(h)} dx$ and $a_{p,q} = a(\phi_q^{(h)}, \phi_p^{(h)}) = (\nabla \phi_q^{(h)}, \nabla \phi_p^{(h)})$ and the solution vector $\mathbf{u} = (u_q)_{q \in N_h}$
- We call $\mathbf{A}$ the **Stiffness Matrix** and $\mathbf{f}$ the **Load Vector**.

Basis Functions

Because in the weak formulation the functions only have to be differentiable once, we can use piecewise linear basis functions.

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{h} & x \in [x_{i-1}, x_i] \\ \frac{x_{i+1} - x}{h} & x \in [x_i, x_{i+1}] \\ 0 & \text{else} \end{cases}$$

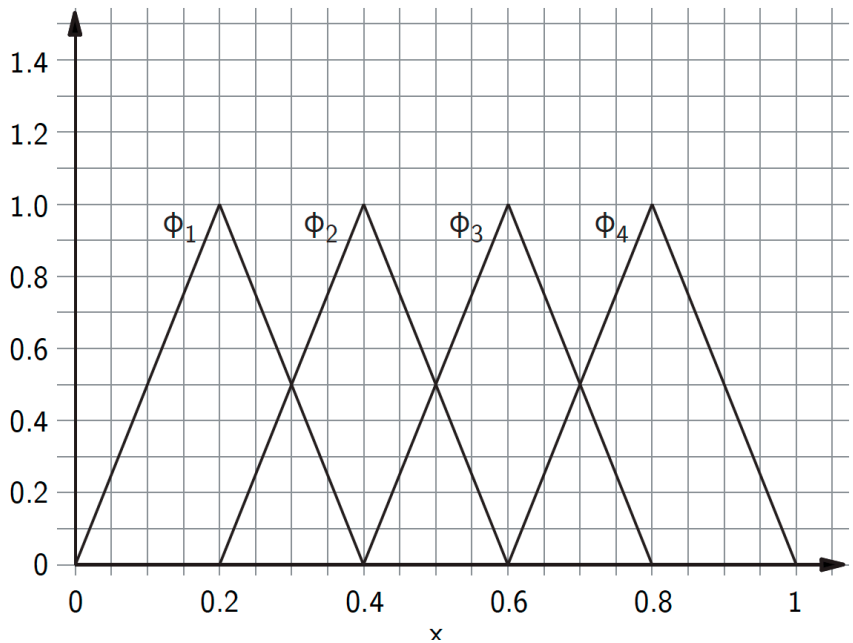


Figure: Linear basis functions ϕ_i for $i \in N_h$.

FEM: System of linear equations

Plugging the discretized functions into the weak formulation yields:

$$\sum_{i,j} u_j \int_0^1 \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}dx = \sum_i f_i \int_0^1 \phi_i dx$$

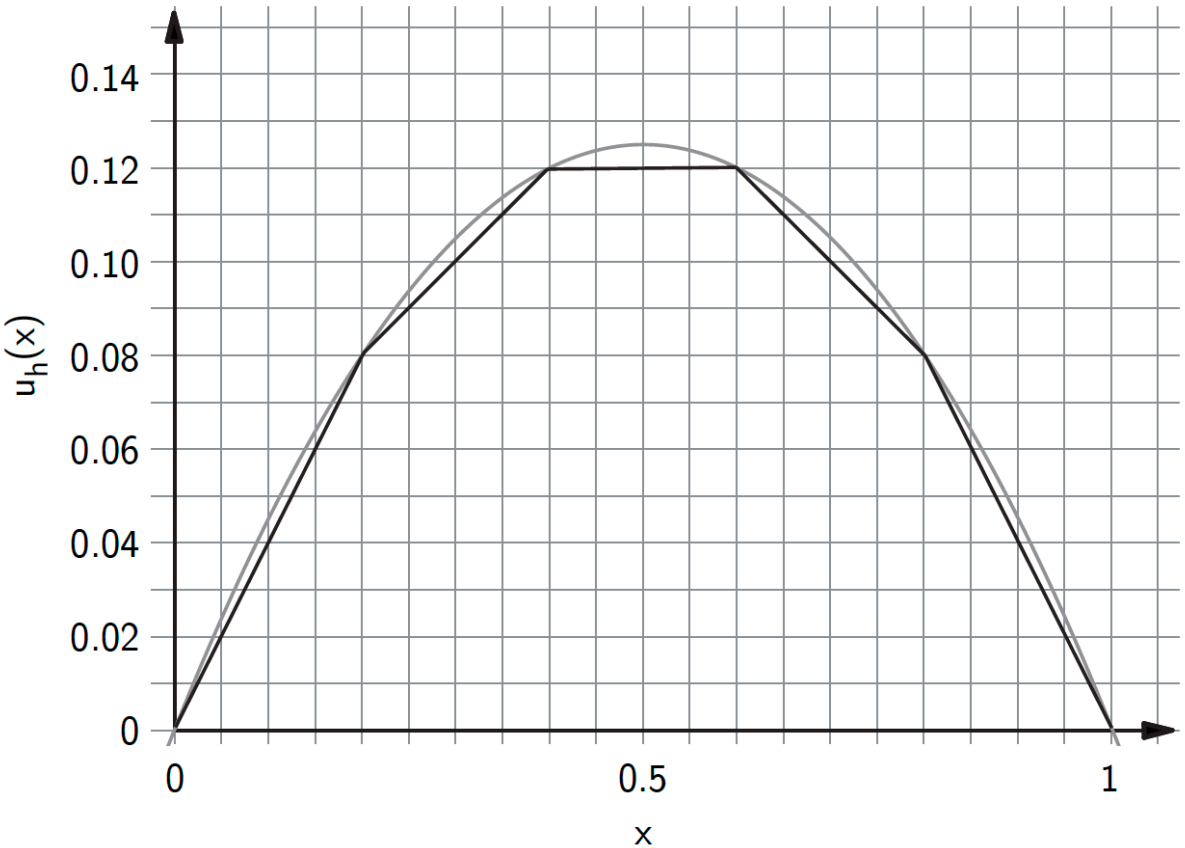
And the according linear system then yields:

$$\frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = h\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

FEM: System of linear equations

Solving the system of linear equations results in:

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} 2h^2 \\ 3h^2 \\ 3h^2 \\ 2h^2 \end{pmatrix}$$



Poisson Problem (2D)

E.g. heat conduction, electrostatics, diffusion of substances.

$$-\nabla^2 u(\boldsymbol{x}) = f(\boldsymbol{x}) \qquad \boldsymbol{x} \text{ in } \Omega \tag{23a}$$

$$u(\boldsymbol{x}) = 0 \qquad \boldsymbol{x} \text{ on } \partial\Omega \tag{23b}$$

where u is the trial function.

Let's derive the weak formulation. First, multiply from the left by a test function φ and integrate over Ω:

$$-\int_\Omega \varphi \nabla^2 u = \int_\Omega \varphi f \tag{24}$$

Poisson Problem (2D): Weak Formulation

Then, we integrate by parts:

$$\int_\Omega \nabla\varphi \cdot \nabla u - \int_{\partial\Omega} \varphi \boldsymbol{n} \cdot \nabla u = \int_\Omega \varphi f \tag{25}$$

On $\partial\Omega$, $\varphi = 0$ is required per definition and hence:

$$\int_\Omega \nabla\varphi \cdot \nabla u = \int_\Omega \varphi f \tag{26}$$

which can be written as:

$$a\left(u, \varphi\right) = f\left(\varphi\right) \tag{27}$$

- **bilinear form:** $a\left(u, \varphi\right) = \int_\Omega \nabla\varphi \cdot \nabla u$ is the
- **linear form:** $f\left(\varphi\right) = \int_\Omega \varphi f$

FEM: Discretization

Galerkin method:

- We approximate the infinite dimensional funcion space $V = H_0^1(\Omega)$ with a subspace $V_h \subset V$ with $dim(V_h) < \infty$.
- For our purposes the trial- and test-functions belong to the same subspace V_h.

The discrete problem reads then as

$$\text{Find } u_h \in V_h \subset V \text{ such that}$$
$$a(u_h, \varphi) = f(\varphi) \text{ for all } \varphi \in V_h$$

where V_h is a suitable function space, $a(u_h, \varphi)$ is a *bilinear form* and f *linear form*.

- h stands for discretization parameter; the notation suggests that the approximate solution will converge to the true solution of the given (continuous) problem as $h \to 0$.

Discretization

The discretized solution reads:

$$u_h = \sum_j U_j \phi_j (\boldsymbol{x}) \tag{28}$$

with U_j the unknown DoF, and $\phi_j (\boldsymbol{x})$ the FEM shape functions.
Inserting into $(\nabla \phi_i, u_h) = (\phi_i, f)$, we get a linear system:

$$AU = F \tag{29}$$

with $A_{ij} = (\nabla \phi_i, \nabla \phi_j) = a((\phi_i, \phi_j))$ as **Stiffness Matrix** and $F_i = (\phi_i, f) = f(\phi_i)$ as **Load Vector**.

How to compute A_{ij} and F_i?

$$A_{ij} = (\nabla\phi_i, \nabla\phi_j) = \sum_{K\in\mathcal{T}} \int_K \nabla\phi_i \cdot \nabla\phi_j \tag{30a}$$

$$F_i = (\phi_i, f) = \sum_{K\in\mathcal{T}} \int_K \phi_i f \tag{30b}$$

where K is the cell index, and $\mathcal{T}$ the mesh.

Quadrature yiels:

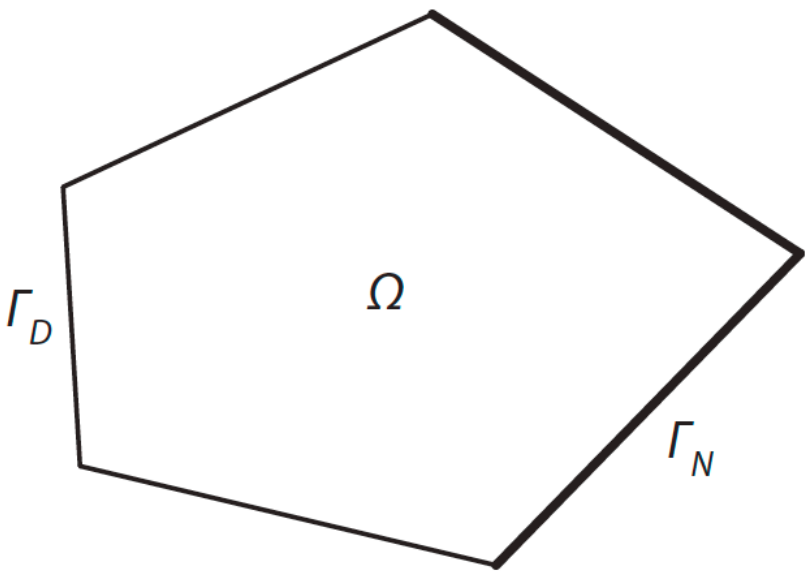
$$A_{ij}^K = \int_K \nabla\phi_i \cdot \nabla\phi_j \approx \sum_q \nabla\phi_i\left(\boldsymbol{x}_q^K\right) \cdot \nabla\phi_j\left(\boldsymbol{x}_q^K\right) w_q^K \tag{31a}$$

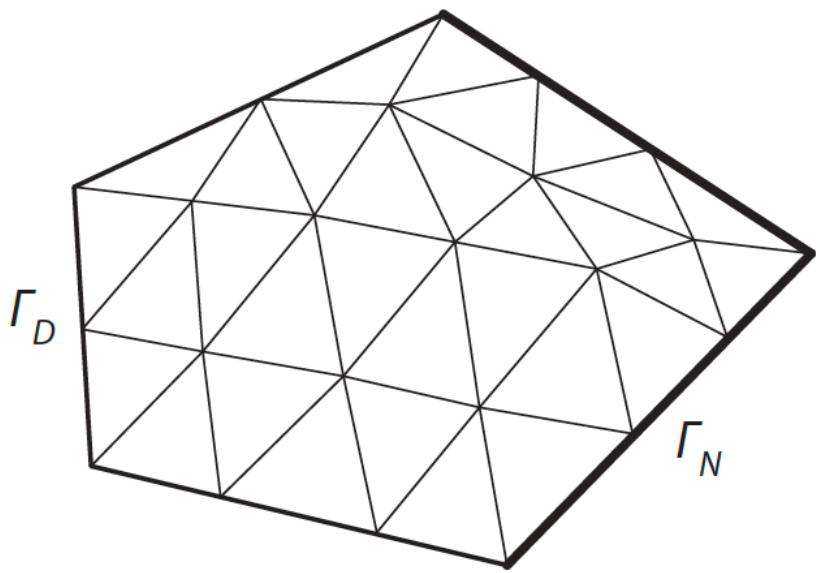
$$F_i^K = \int_K \phi_i f \approx \sum_q \phi_i\left(\boldsymbol{x}_q^K\right) f\left(\boldsymbol{x}_q^K\right) w_q^K \tag{31b}$$

with $\boldsymbol{x}_q^K$ being the qth qudarature point of cell K, and w_q^K the quadrature weight.

Triangulation of the domain

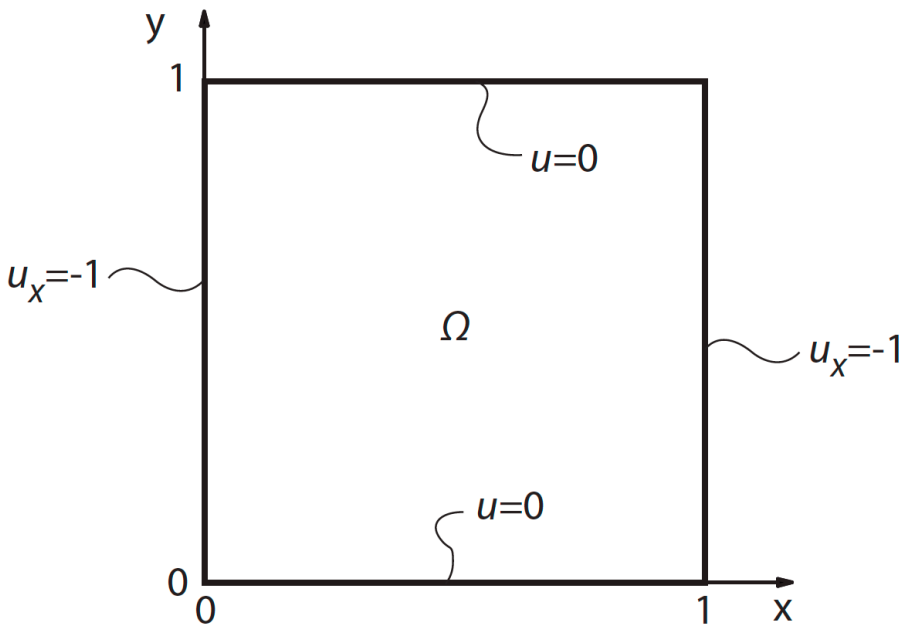
Triangles should cover the whole domain, have common edges with their neighbors and each edge has to belong to only one BC.

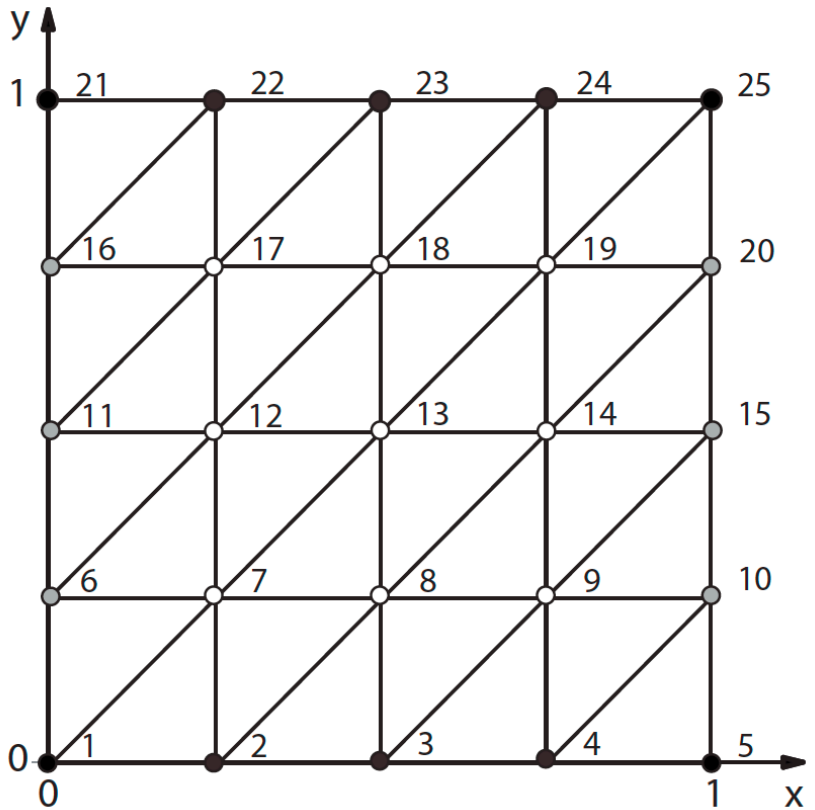




Triangulation of the domain

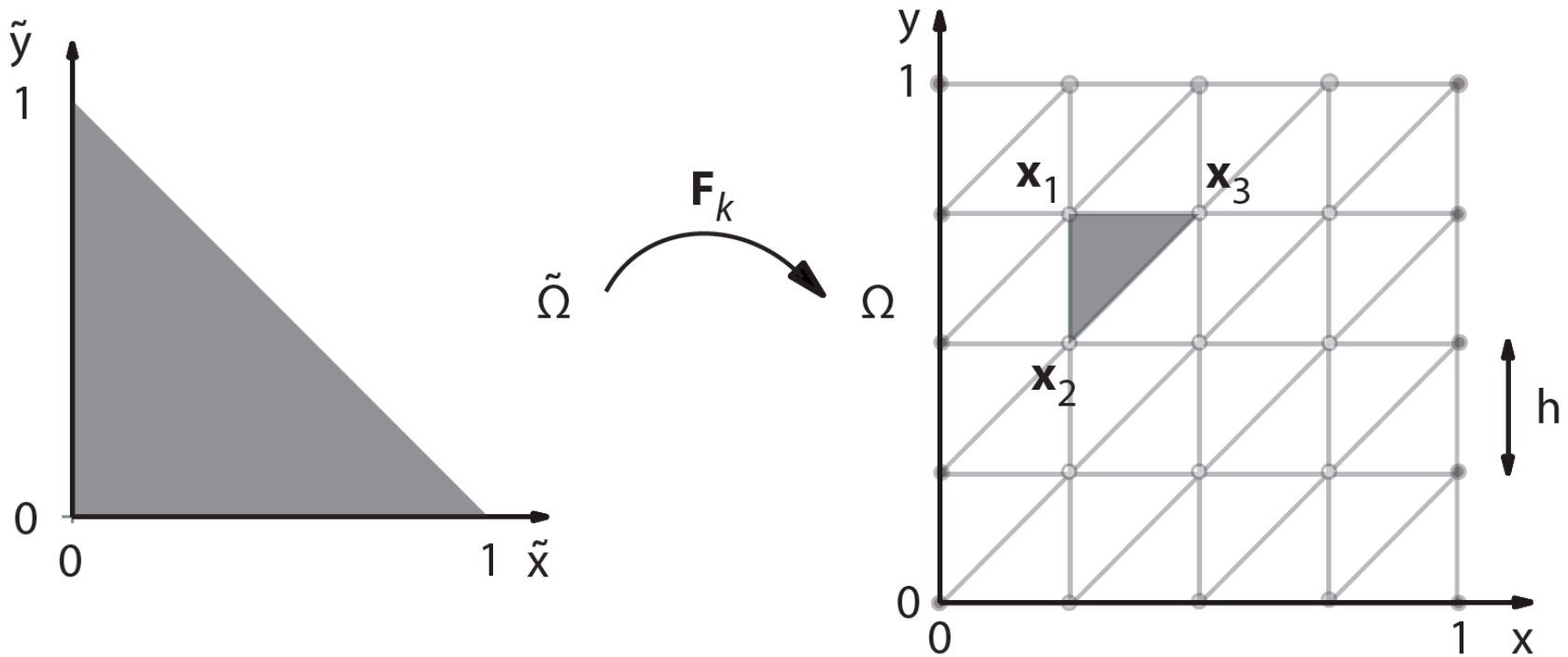
Let's consider the Poisson equation on the following domain:





Mapping triangles

Because the triangles can be very different it is helpful to define all basis functions on a reference triangle and then map it to the real domain.



Basis functions in 2D

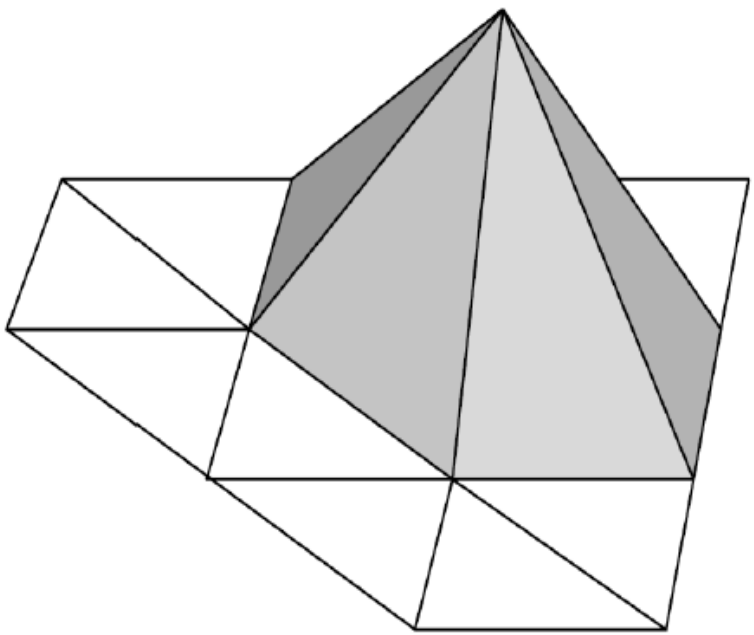
On the reference domain we can for example define three different basis functions denoted by N_k.

$$\begin{aligned} N_1 &= 1 - \tilde{x} - \tilde{y} \\ N_2 &= \tilde{x} \\ N_3 &= \tilde{y} \end{aligned}$$

Basis functions in 2D

Arranging the basis functions properly will result in the following 2D hat function:

$$\phi_i(x) = \begin{cases} N_3(\mathbf{x}) & \mathbf{x} \in T_1 \bigcup T_4 \\ N_2(\mathbf{x}) & \mathbf{x} \in T_2 \bigcup T_5 \\ N_1(\mathbf{x}) & \mathbf{x} \in T_3 \bigcup T_6 \\ 0 & \text{else} \end{cases}$$



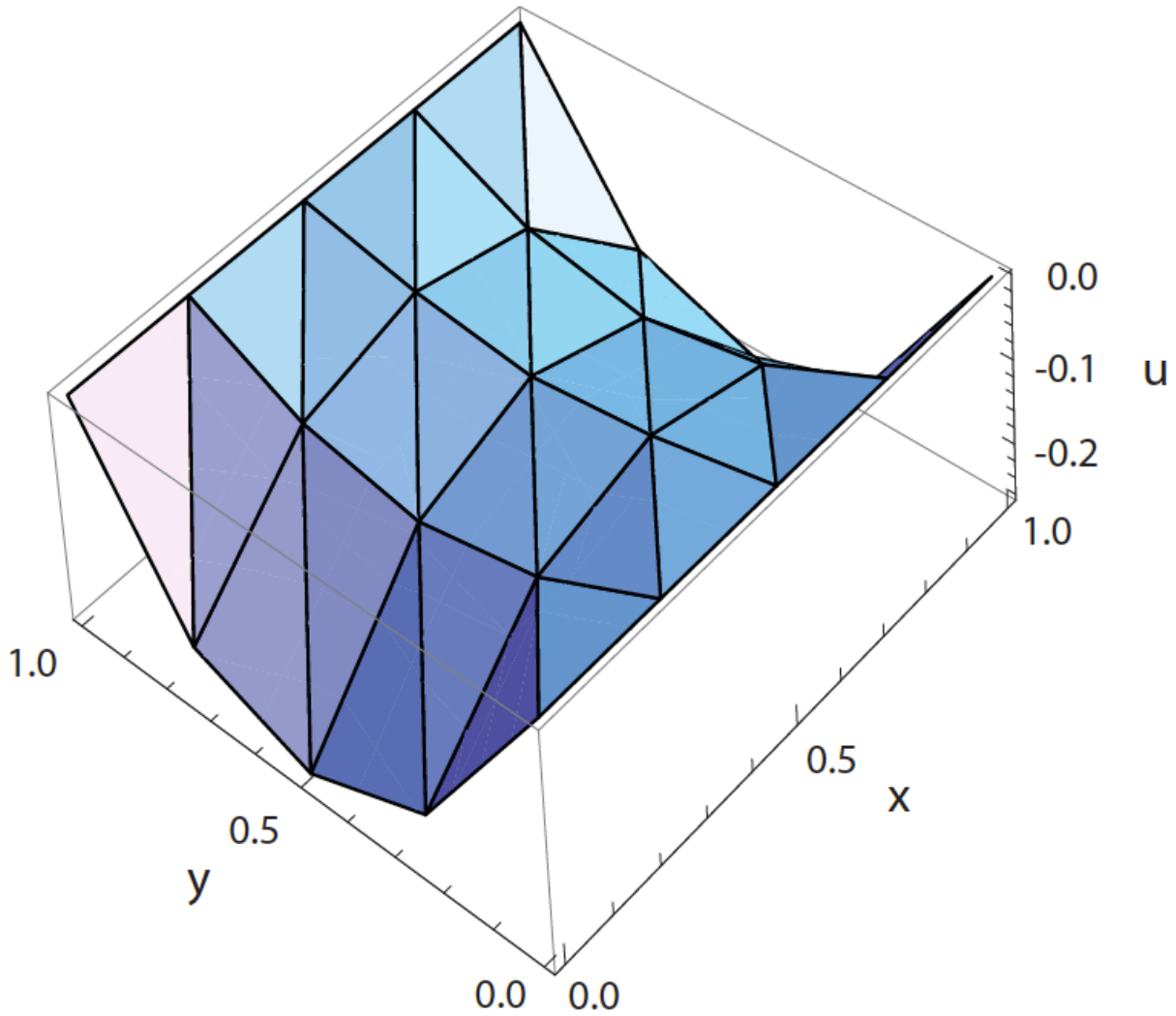
System of linear equations

The system of linear equations can now be assembled with A as stiffness matrix and b as load vector:

$$\begin{aligned} (A)_{i,j} &= \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, dV \\ (b)_i &= \int_\Omega \phi_i f \, dV \end{aligned}$$

We then just have to solve the linear system $\mathbf{Au} = \mathbf{b}$ to get the solution vector u. Efficient solvers for linear systems are for example **Multigrid methods**.

Solution example



Time Dependent PDE's

Up until now we only considered time-independent problems, as they are easier to handle. Nevertheless what we are really in terested in is solving time-dependent PDE's such as the diffusion equation.

Diffusion equation

$$\frac{du(\mathbf{x},t)}{dt} = D\Delta u(\mathbf{x},t)$$

Temporal discretization

For temporal discretization the same methods already used in FDM can be applied:

- Forward Euler
- Backward Euler
- Crank-Nicholson

where the forward Euler method is the easiest but most unstable method.

Temporal discretization

Let's look at the diffusion equation in a forward euler scheme:

$$\begin{aligned} \frac{u^k - u^{k-1}}{\Delta t} &= \nabla^2 u^{k-1} \\ u^k &= \Delta t \nabla^2 u^{k-1} + u^{k-1} \end{aligned}$$

where k denotes the discrete time steps that are Δt apart.
∇^2 is the Laplace operator.

Time-Dependent PDE's: Weak formulation

We can again easily find a weak formulation for the problem.

$$\int_\Omega \varphi u^k \, dx = -\Delta t \int_\Omega \nabla\varphi \cdot \nabla u^{k-1} \, dx + \varphi u^{k-1} \, dx \tag{32}$$

which can be again written in terms of the basis functions ϕ_i and ϕ_j. In order to have a shorter notation, we will define $M = \int_\Omega \phi_i \phi_j \, dx$ (**mass matrix**) and $A = \int_\Omega \nabla\phi_i \nabla\phi_j \, dx$ (stiffness matrix). This results in:

$$MU^k = (M - \Delta t \, A) \, U^{k-1} \tag{33}$$

This is solved iteratively until all u^k are known.

Summary

- Transforming the problem into a weak formulation, setting up appropriate basis functions on a discretized domain and solving the resulting relatively sparse system of linear equations are the key components of the finite element method.
- The power of the FEM is the flexibility regarding the geometry.
- FEM requires less regularity for its functions, because of weak formulation.
- FEM needs substantially more CPU-memory compared to FDM.
- Implementation of FEM is more complex compared to FDM.

Finite Element Softwares

COMSOL Multiphysics

- easy to use
- GUI, Matlab interface
- commercial, closed-source
- tutorial next week

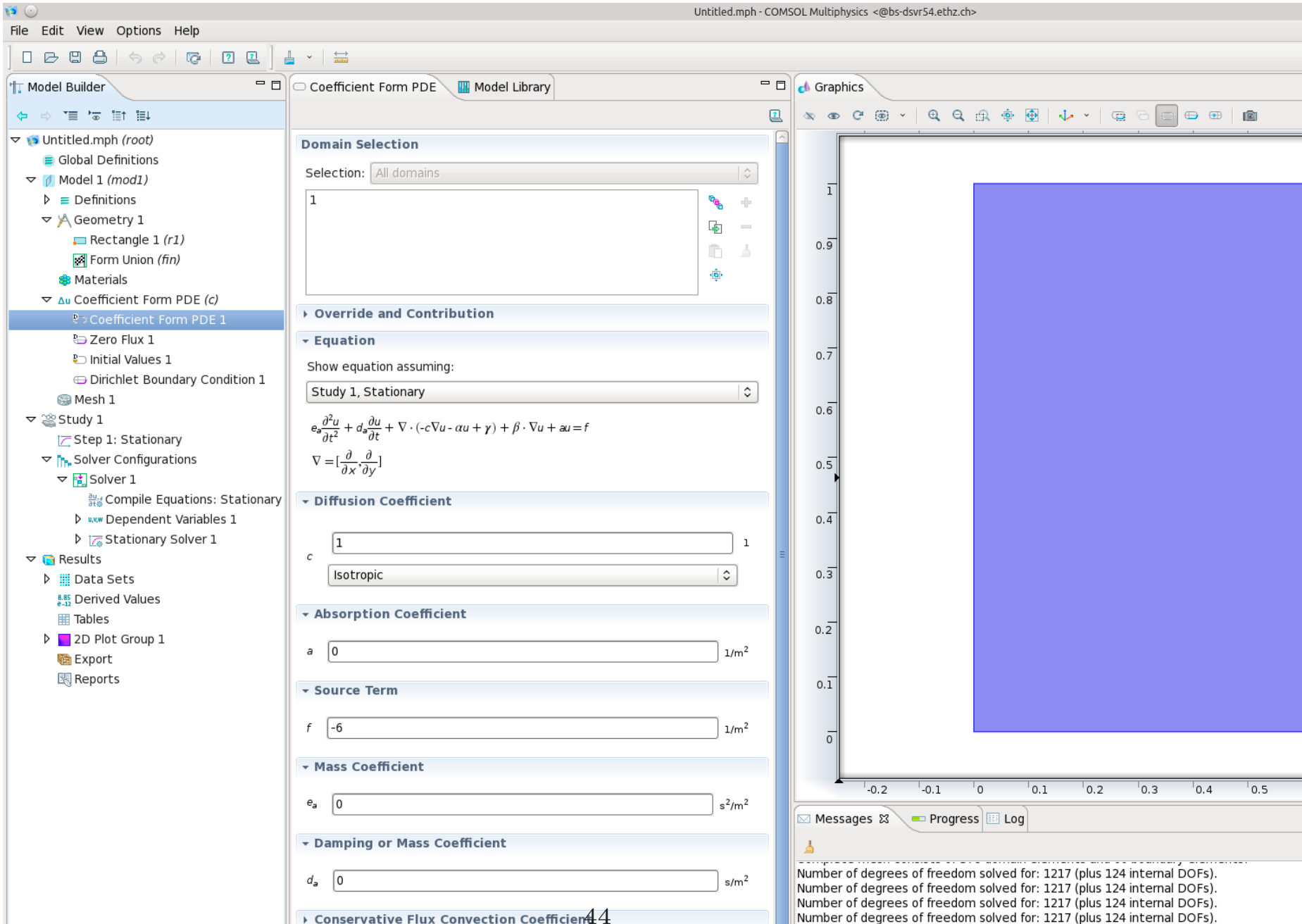
FEniCS ▸ http://fenicsproject.org/

- Python interface
- FOSS
- intermediate level

deal.ii ▸ http://www.dealii.org/

- C++
- FOSS
- difficult

COMSOL


Untitled.mph - COMSOL Multiphysics <@bs-dsvr54.ethz.ch>
File Edit View Options Help
Model Builder
Untitled.mph (root)
Global Definitions
Model 1 (mod1)
Definitions
Geometry 1
Rectangle 1 (r1)
Form Union (fin)
Materials
Coefficient Form PDE (c)
Coefficient Form PDE 1
Zero Flux 1
Initial Values 1
Dirichlet Boundary Condition 1
Mesh 1
Study 1
Step 1: Stationary
Solver Configurations
Solver 1
Compile Equations: Stationary
Dependent Variables 1
Stationary Solver 1
Results
Data Sets
Derived Values
Tables
2D Plot Group 1
Export
Reports
Coefficient Form PDE
Model Library
Domain Selection
Selection: All domains
1
Override and Contribution
Equation
Show equation assuming:
Study 1, Stationary
$e_a\frac{\partial^2 u}{\partial t^2}+d_a\frac{\partial u}{\partial t}+\nabla\cdot(-c\nabla u-\alpha u+\gamma)+\beta\cdot\nabla u+au=f$
$\nabla=[\frac{\partial}{\partial x},\frac{\partial}{\partial y}]$
Diffusion Coefficient
c
1
1
Isotropic
Absorption Coefficient
a
0
1/m²
Source Term
f
-6
1/m²
Mass Coefficient
e_a
0
s²/m²
Damping or Mass Coefficient
d_a
0
s/m²
Conservative Flux Convection Coefficient
Graphics
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
Messages
Progress
Log
Number of degrees of freedom solved for: 1217 (plus 124 internal DOFs).
Number of degrees of freedom solved for: 1217 (plus 124 internal DOFs).
Number of degrees of freedom solved for: 1217 (plus 124 internal DOFs).
Number of degrees of freedom solved for: 1217 (plus 124 internal DOFs).

FEnICS: example

```
1  from dolfin import *
2
3  # Create mesh and define function space
4  mesh = UnitSquare(6, 4)
5  V = FunctionSpace(mesh, 'Lagrange', 1)
6
7  # Define boundary conditions
8  u0 = Expression('1 + x[0]*x[0] + 2*x[1]*x[1]')
9
10 def u0_boundary(x, on_boundary):
11     return on_boundary
12 bc = DirichletBC(V, u0, u0_boundary)
13
14 # Define variational problem
15 u = TrialFunction(V)
16 v = TestFunction(V)
17 f = Constant(-6.0)
18 a = inner(nabla_grad(u), nabla_grad(v))*dx
19 L = f*v*dx
20
21 # Compute solution
22 u = Function(V)
23 solve(a == L, u, bc)
24
25 # Plot solution and mesh
26 plot(u)
27 plot(mesh)
28
29 # Dump solution to file in VTK format
30 file = File('poisson.pvd')
31 file << u
```

deal.ii: assembling the system (i)

```
void Step3::assemble_system ()
{
  QGauss<2>  quadrature_formula(2);
  FEValues<2> fe_values (fe, quadrature_formula,
                         update_values | update_gradients |
                             update_JxW_values);
  const unsigned int    dofs_per_cell = fe.dofs_per_cell;
  const unsigned int    n_q_points    = quadrature_formula.size();
  FullMatrix<double>    cell_matrix (dofs_per_cell, dofs_per_cell);
  Vector<double>        cell_rhs (dofs_per_cell);
  std::vector<types::global_dof_index> local_dof_indices (
       dofs_per_cell);
  DoFHandler<2>::active_cell_iterator
  cell = dof_handler.begin_active(),
  endc = dof_handler.end();
  for (; cell!=endc; ++cell)
    {
      fe_values.reinit (cell);
      cell_matrix = 0;
      cell_rhs = 0;
      for (unsigned int i=0; i<dofs_per_cell; ++i)
        for (unsigned int j=0; j<dofs_per_cell; ++j)
          for (unsigned int q_point=0; q_point<n_q_points; ++q_point)
            cell_matrix(i,j) += (fe_values.shape_grad (i, q_point) *
                                 fe_values.shape_grad (j, q_point) *
                                 fe_values.JxW (q_point));
      for (unsigned int i=0; i<dofs_per_cell; ++i)
        for (unsigned int q_point=0; q_point<n_q_points; ++q_point)
          cell_rhs(i) += (fe_values.shape_value (i, q_point) *
                          1 *
                          fe_values.JxW (q_point));
```

deal.ii: assembling the system (ii)

```
        cell->get_dof_indices (local_dof_indices);
        for (unsigned int i=0; i<dofs_per_cell; ++i)
          for (unsigned int j=0; j<dofs_per_cell; ++j)
            system_matrix.add (local_dof_indices[i],
                               local_dof_indices[j],
                               cell_matrix(i,j));
        for (unsigned int i=0; i<dofs_per_cell; ++i)
          system_rhs(local_dof_indices[i]) += cell_rhs(i);
      }
    std::map<types::global_dof_index,double> boundary_values;
    VectorTools::interpolate_boundary_values (dof_handler,
                                              0,
                                              ZeroFunction<2>(),
                                              boundary_values);
    MatrixTools::apply_boundary_values (boundary_values,
                                        system_matrix,
                                        solution,
                                        system_rhs);
}
```

The shown example and excellent tutorials can be found on

- http://www.dealii.org/8.0.0/doxygen/tutorial/index.html

Literature

- Thomas Jenni, *Einfuehrung in die Finite-Elemente-Methode*, ETH Zurich, 2011
- Erik G. Thompson, *Introduction to the Finite Element Method*, Wiley, 2005
- Ralf Hiptmayr, *Numerical Methods for Partial Differential Equations*, lecture notes, 2010
- Press, Teukolsky Vetterling, Flannery, *Numerical Recipes*, Cambridge University Press, 2007
- LeVeque, *Finite Differece Methods for Ordinary and Partial Differential Equations*
- Gerald W. Recktenwald, *Finite-Difference Approximations to the Heat Equation*, 2011

Thanks!!

Thanks for your attention!

Slides for this talk will be available at:
`http://www.bsse.ethz.ch/cobi/education`

Spatio-Temporal Modeling in Biology

Dagmar Iber, Roberto Croce

Computational Biology (CoBI), D-BSSE, ETHZ