TMA4220 Introduction: Finite Element methods for Partial Differential Equations

20 August 2018

Outline

Outline

Schedule

12 sessions Week 35–46 + Examination repetition Week 47

Lectures	Monday/Tuesday 10–12	L10
Exercises	Friday 13–15	S22

Notes:

- Introduction on Week 34 on Monday.
- Lectures + corresponding Exercises every Week 35-46.
- A room will be booked for the project at Banachrommet or Nullrommet.
- From Week 35 onwards office hours are offered on Tuesday 13-15
- Please book the office hours latest on Monday.

Evaluation

35%	Projects	2D Poisson Solver (10%)	Week 37, deadline 2018-10-15
		3D Applications (25%)	Week 42, deadline 2018-11-23
65%	Examination	Three problems	2018-12-07

Exercises and Project:

1. Regular Exercises are not compulsory can be handed out for feedback.
2. Project Exercises are compulsory and should be validated for Examination approval.
3. Delivery involves written report **and** source code in a repository.
4. Final handout Week 47 consists of a commented project demo (approx. 10 min).

Examination:

1. Exercises will cover most requirements.
2. Previous examination question studied during the Exercise sessions.
3. Repetition session scheduled at the end of the curriculum.

Course plan

Two main parts:

1. Formulation of PDE problems and the theory of Finite Element methods.
2. Applications to some PDES and numerical algorithms.

- The first part is usually easily understood by Math student but should not scare away other students: the focus is on understanding the concepts.
- The second part involves implementation details and the mathematical requirements are kept at the application level.

Material:

1. Numerical Models for Differential Problems by Alfio Quarteroni (Second Edition, Springer, 2014). A free digital version of the book is available at Springer Link for those with a university IP (e.g., accessed from campus).
2. Additional lecture notes published on the Wiki.
3. Course schedule refreshed the week before for references.

Course plan: Part 1

Mathematical Theory:

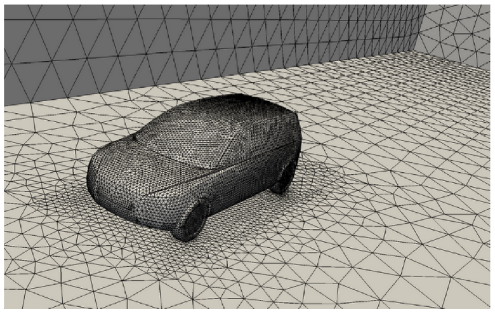
1. 35. Introduction to PDEs, weak solution, variational formulation.
2. 36. Ritz method for the approximation of solutions to elliptic PDEs
3. 37. Galerkin method and well-posedness.
4. 38. Finite Element spaces.
5. 39. Polynomial approximation and error analysis.
6. 40. Time dependent problems.

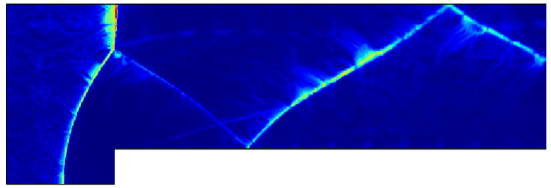
Course plan: Part 2

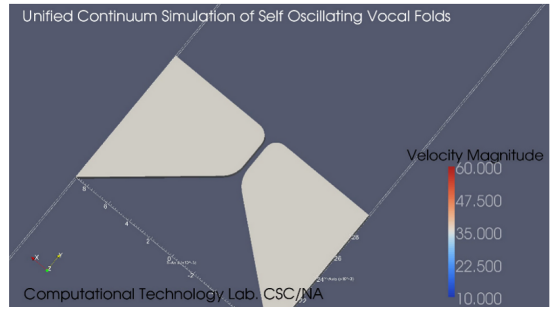
Applications and numerical algorithms:

1. 41. Implementation of the Finite Element method.
2. 42. Linear Elasticity.
3. 43. Mesh generation and adaptive control.
4. 44. Iterative solvers and preconditioners.
5. 45. Stabilized finite element methods.
6. 46. Mixed problems.

Context

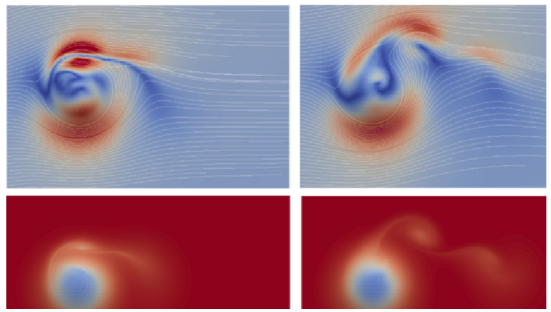


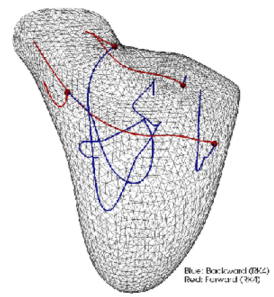




Algorithms and HPC implementations of adaptive finite elements:

- High-Re incompressible turbulent flows
- Compressible flows
- FSI with a Unified Continuum model
- Aeroacoustics
- Variable density flows for Geophysics
- Haemodynamics





Approach

Development of computational methods for scientific research and innovation in engineering and technology.

Covers the entire spectrum of natural sciences, mathematics, informatics:

- Scientific model (Physics, Biology, Medical, . . .)
- Mathematical model
- Numerical model
- Implementation
- Visualization, Post-processing
- Validation

$\rightarrow$ Feedback: virtuous circle

Allows for more realistic problems to be simulated, new theories to be experimented numerically.

Goals:

1. Identify important concepts in developing schemes,
2. Understand how they impact the practices.

Outline

Physical Model

Evolution of a physical quantity $u(\mathbf{x}, t)$ is governed by relations of type:

$$A(\ u(\mathbf{x}, t)\ ) = F(\mathbf{x}) \tag{1}$$

They can be summarized as a balance equation:

$$IN + OUT + PRODUCTION + DESTRUCTION = 0 \tag{2}$$

Example: Fundamental principle of dynamics

$$m\, a = \sum F_i \tag{3}$$

with m mass, a acceleration, F_i forces.

Partial Differential Equations

Physical phenomena: represented in terms of variation of a physical quantity in space and time.

Field: value that can be measured in any point in space and time.

Processes: modelled by differential operators

- Diffusion: Laplace operator, 2nd derivatives in space.
- Transport: Advection operator, 1st derivatives in space and time.
- Wave propagation: D'Alembert operator, 2nd derivatives in space and time.

Intuitive approach with the derivative

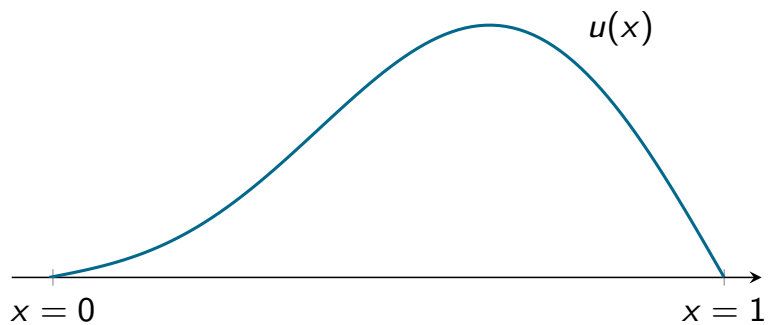
Derivative in one-dimension: $u(x)$

- 1st derivative: variation/speed

$$\frac{du}{dx}(x)$$

- 2st derivation: acceleration/diffusion rate

$$\frac{d^2u}{dx^2}(x)$$



Partial derivative: extension of the previous idea in different dimension: $u(\mathbf{x}, t)$, $(\mathbf{x}, t) \in \Omega \times (0, T)$:

$$\frac{\partial u(\mathbf{x}, t)}{\partial x}$$

Examples of differential operators

With $\mathbf{x}$ point coordinates, t time:

1. Diffusion:
$$-\boldsymbol{\Delta} T(\mathbf{x})$$
2. Advection:
$$\frac{\partial \rho(\mathbf{x}, t)}{\partial t} + \mathbf{u}(\mathbf{x}, t) \nabla \cdot \rho(\mathbf{x}, t)$$

The Poisson problem

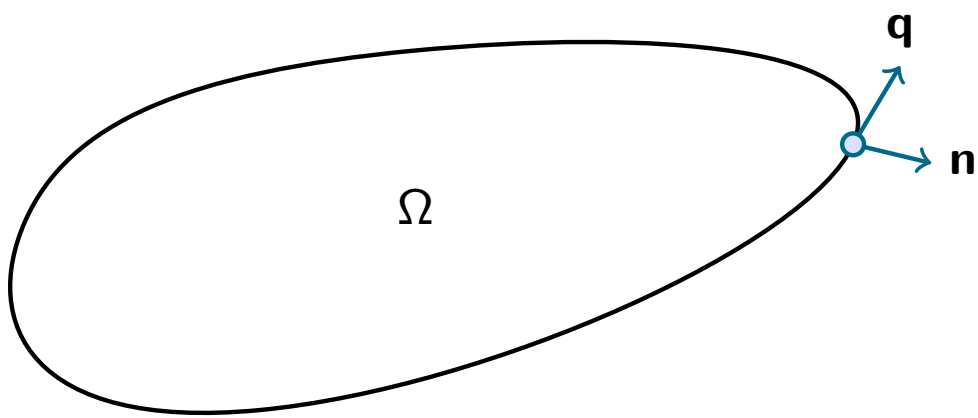
- This problem is important because a number of physical processes are modelled either entirely or in part by the Poisson equation.
- The technical term is a *diffusion process*: u is to be interpreted as the concentration of a physical quantity, and f as the rate at which is introduced ($f > 0$) or removed ($f < 0$) from the domain.
- The physical quantity (typically intensive) may be a concentration, temperature, potentials, ...
- The solution in Ω is uniquely determined by the boundary data and f.

Steady heat transfer

Energy transfered out of an arbitrary domain V can be expressed as

$$\int_{\partial V} \mathbf{q} \cdot \mathbf{n} \mathrm{dS} = \int_{\mathrm{V}} \mathrm{fdV}.$$

where **q** is the heat flux, **n** is the outward surface normal along the boundary ∂V and f represents a volumetric heat source. This basically says that the net energy generation inside the domain must equal the net energy flowing out of the domain.



Steady heat transfer

- Applying the Gauss' divergence theorem, we can write

$$\int_{\partial V} \mathbf{q} \cdot \mathbf{n} \mathrm{dS} = \int_{\mathrm{V}} \nabla \cdot \mathbf{q} \mathrm{dV} = \int_{\mathrm{V}} \mathrm{f} \mathrm{dV},$$

  yielding

$$\nabla \cdot \mathbf{q} = f.$$

- Applying Fourier's law, i.e. $\mathbf{q} = -\kappa \nabla u$, $\kappa > 0$, we get

$$-\nabla \cdot \kappa \nabla u = f \qquad \text{in } \Omega,$$

  to be solved for the temperature u.

- For a constant isotropic heat conductivity κ, we regain the Poisson equation,

$$-\kappa \nabla^2 u = f.$$

A mathematical problem

The mathematical problem is built from the input physical model.

Given data:

- a domain of definition of the problem Ω (physical domain),
- a set of condition at the boundary of Ω: imposed value, flux,
- an initial condition if time-dependent.

Find a function satisfying an equation problem involving:

- differential operators,
- constants.

Mathematical properties

Two fundamental properties should be satisfied:

- **Existence**: there should be at least a solution.
- **Uniqueness**: we likely want likely only one solution otherwise the outcome of the model is tricky to evaluate.

Characterization of the solution in term of regularity:

- **Smoothness**: Is the solution continuous or does it tear/break?
- **Bound**: Can it take values that tend to infinity?

The physical model may be ill-posed: it should be modified to satisfy "good properties".

Norm: a real number given by $\|\mathbf{u}\|$ serves as measure of a function.

Example: Poisson problem

Let us consider the Poisson problem posed in a domain Ω, an open bounded subset of $\mathbb{R}^d$, $d \geq 1$ supplemented with homogeneous Dirichlet boundary conditions:

$$-\boldsymbol{\Delta} u(\mathbf{x}) = f(\mathbf{x}) \tag{4a}$$

$$u(\mathbf{x}) = 0 \tag{4b}$$

with $f \in \mathrm{C}^0(\Omega)$ and the Laplace operator,

$$\boldsymbol{\Delta} = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2} \tag{5}$$

thus involving second order partial derivatives of the unknown u with respect to the space coordinates.

Definition (Classical solution)

A classical solution (or strong solution) of Problem (**??**) is a function $u \in \mathrm{C}^2(\Omega)$ satisfying relations (**??**) and (**??**).

Example: weak formulation of Poisson

The problem can be reformulated to give a new *meaning* to the solution.

Example: Weak formulation of Problem (**??**) reads then:

$$\left| \begin{array}{l} \text{Find } u \in \mathrm{H}_0^1(\Omega), \text{ such that:} \\ \\ \displaystyle\int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \, \mathrm{d}\mathbf{x} = \int_\Omega f v \, \mathrm{d}\mathbf{x} \quad , \; \forall \; v \in \mathrm{H}_0^1(\Omega) \end{array} \right. \tag{6}$$

This is the type of mathematical problem which is the base of Finite Element methods.

Computable solution of partial differential equations

The mathematical problem is posed such that:

- the solution is a function evaluated at any point in space–time,
- and the domain can also be described at any point in space–time.

Functional Analysis: study of mathematical properties of the *continuous problem*.

Infinite dimension: the solution has an infinite number of values corresponding to the infinite number of points where it can be evaluated.

Discretization

- To have a computable algorithm, ony a finite number of points in space-time are selected: discretization.
- The problem should be reformulated to be evaluated on a finite number of points: discrete approximation.

Numerical Analysis: study of the mathematical properties of *discrete problem*.

Finite dimension: the function is represented as a vector of degrees of freedom, higher dimension $\sim$ higher accuracy.

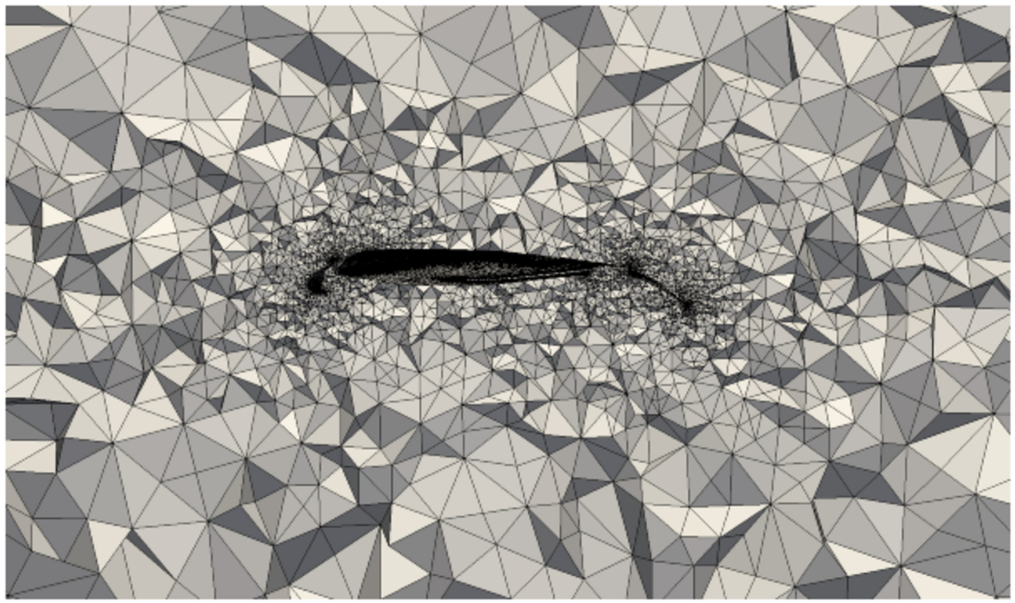
$\Rightarrow$ as the number of unknowns increases the computed discrete solution **converges** (gets closer) to the solution of the continuous problem.

$\Rightarrow$ discretization in time and in space: solve a sequence of steady problems discretized in space, at discrete time-steps.

Discretization of the physical domain

Different possible discretization: cartesian grids, simplicial meshes, Voronoi cells ...

Complex geometries are usually more suited for simplicial meshes:



but other numerical methods like immersed boundaries or fictitious domain approaches maybe used.

Discretization of the physical domain

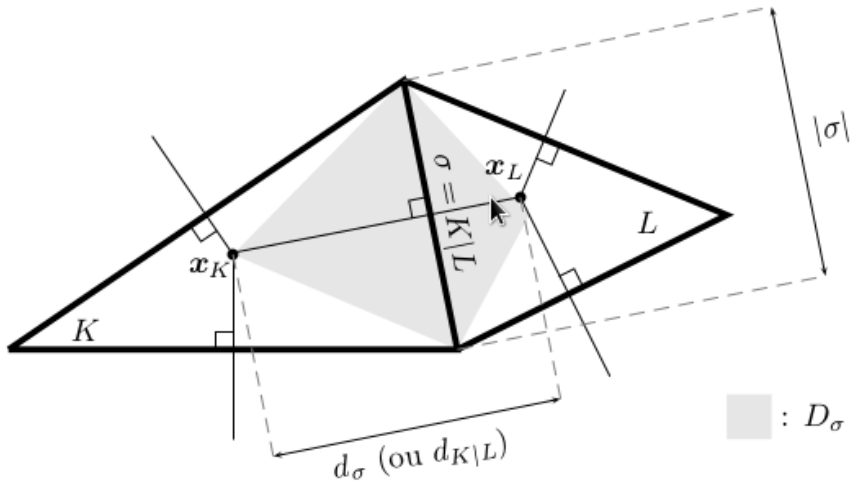


Figure: Geometrical quantities associated to the mesh

Example of simple meshes

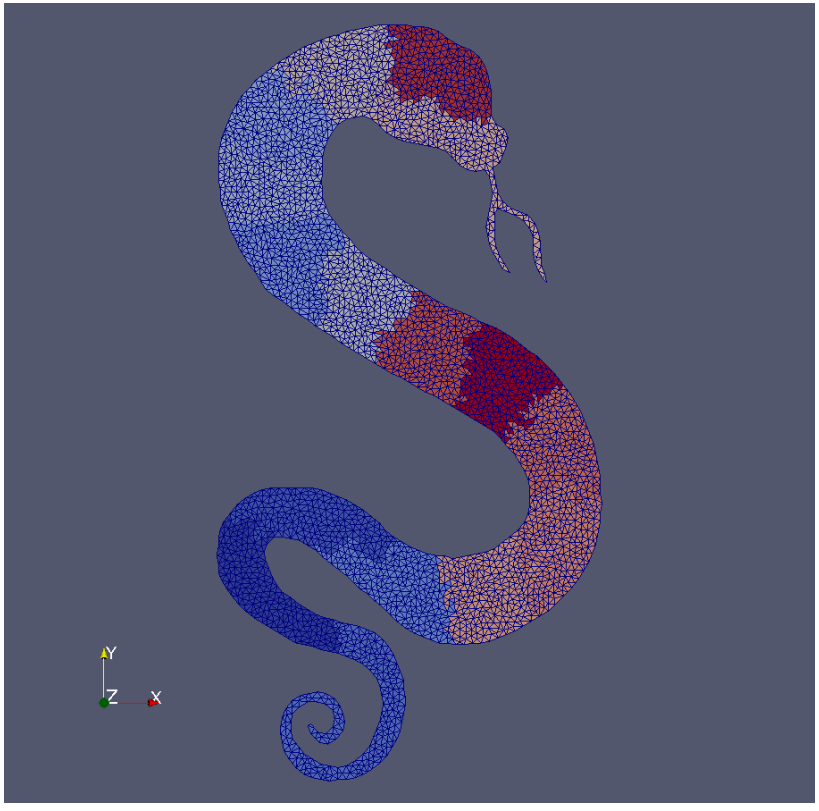


Figure: Snake demo run, 16 processes.

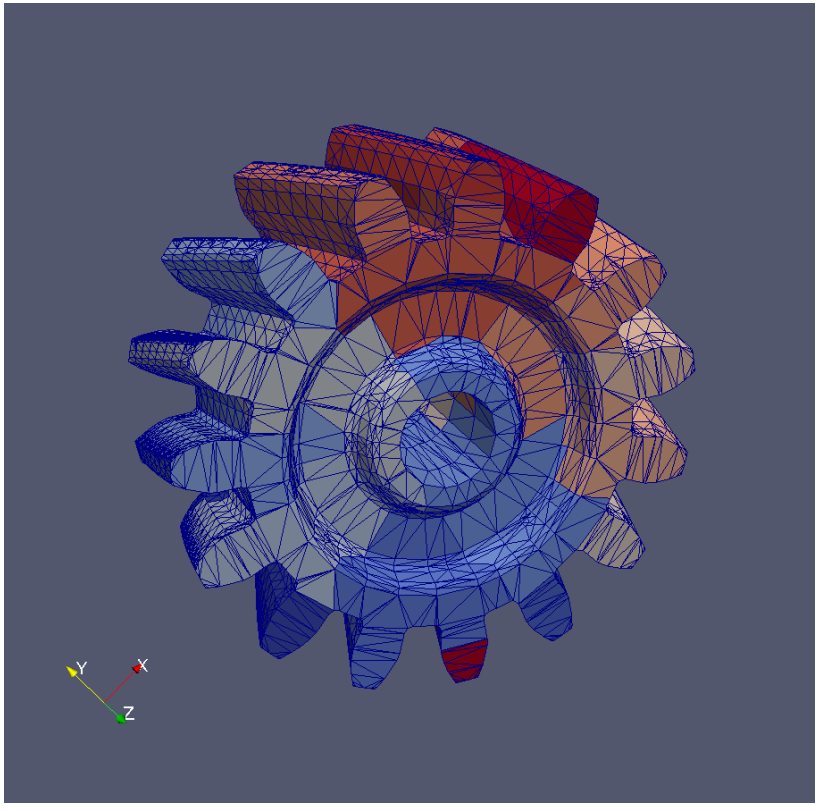


Figure: Gear demo run, 32 processes.

Discrete properties

- Stability: the values of the solution do not go to infinity (*blow up*)

$$\|\mathbf{u}\|_h \leq C$$

- Accuracy: the discrete solution is *close enough* to the continuous solution
- Presence of oscillations?

$$\|\mathbf{u} - \mathbf{u}_h\|_h \leq h^\alpha$$

The properties satisfied by the *continuous problem* are **not** necessarily satisfied by the *discrete approximation*:

- Positivity, Bounds: e.g. energy should be positive.
- Conservation: e.g. mass should not be lost.

Approximations

A continuum of numerical methods to approximate solutions:

- Finite Difference: discretization of derivatives on each axis.
- Finite Volume: discretization of fluxes.
- Discontinuous Galerkin: hybrid FE/FV.
- Finite Element: Galerkin decomposition of the solution.
- Wavelets: hierarchical spaces.
- ...

Discretizations in time:

- Implicit: Backward Differentiation Formulae
- Explicit: Runge–Kutta
- ...

1. They offer different robustness, accuracy, and discrete properties.
2. They have own limitations w.r.t performance and meshes.

Finite Elements

Given a space V_h defined by polynomial functions $\{\phi_j\}$ which are:

- equal to one at $\mathbf{x}_j$,
- zero at $\mathbf{x}_i$, $i \neq j$

so non-zero only on a small domain.

Express Galerkin decomposition:

$$u_h(\mathbf{x}) = \sum_{j=1}^{card(V_h)} u_j \phi_j(\mathbf{x})$$

Assemble contributions of the form:

$$\mathrm{A}_{ij} = \int_\Omega \boldsymbol{\nabla}\phi_j \cdot \boldsymbol{\nabla}\phi_i \,\mathrm{d}\mathbf{x}$$

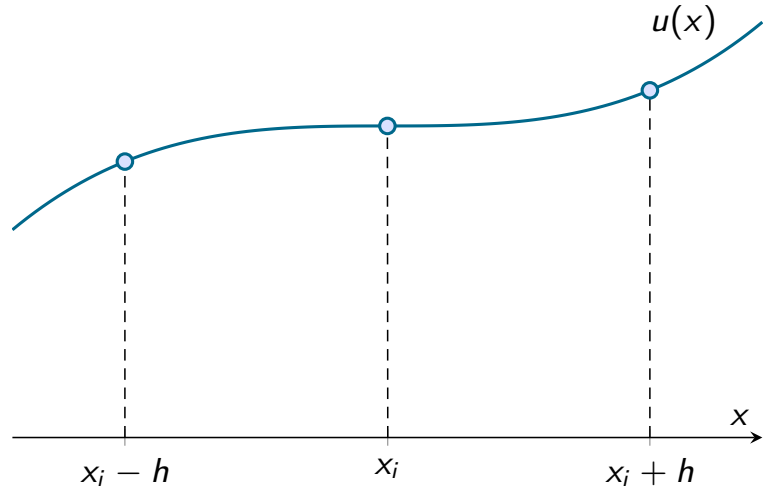
Use quadratures for numerical integration on each cell K:

$$\int_K F(x)\,\mathrm{d}\mathbf{x} \approx \sum_{q=1}^{k} w_q F(x_q)$$

Solve the system: $\mathrm{A}\mathbf{u} = \mathbf{b}$

Finite differences in 1D

- Consider a continuous function 1D function $u(x)$.
- Introduce a grid, $\{x_i\}_{i=0}^{N}$, with $x_i = x_0 + ih$.
- Here h is the grid spacing. For simplicity we consider equidistant grids (constant h).



- Want to approximate derivatives of the function only using data on the grid.

Finite differences in 1D

- First idea: linear approximation of slope

$$u'(x_i) \approx \frac{1}{h}\left(u(x_i + h) - u(x_i)\right).$$

- This is called a one-sided difference (a *forward* difference). Invoking Taylor we find

$$\begin{aligned}\frac{1}{h}\left(u(x_i + h) - u(x_i)\right) &= \frac{1}{h}\left(u(x_i) + hu'(x_i) + \mathcal{O}(h^2) - u(x_i)\right) \\ &= u'(x_i) + \mathcal{O}(h)\end{aligned}$$

  In other words, this is a *first order* approximation to $u'(x_i)$.

Finite differences

- Second idea: a centered difference

$$u'(x_i) \approx \frac{1}{2h}\left(u(x_i + h) - u(x_i - h)\right).$$

- Invoking Taylor we find

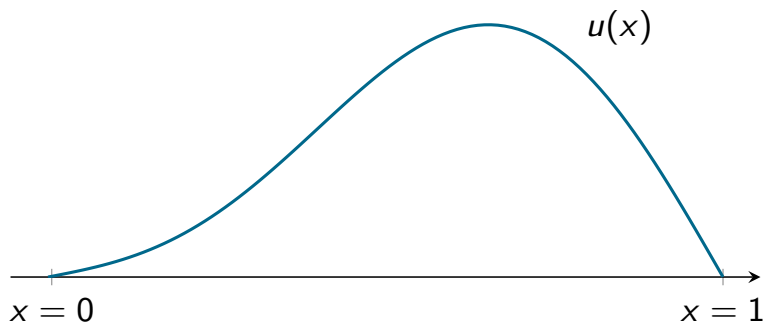
$$\frac{1}{2h}\left(u(x_i + h) - u(x_i - h)\right) = u'(x_i) + \mathcal{O}(h^2).$$

  In other words, this is a *second order* approximation to $u'(x_i)$.

Discretization in 1D

- We now consider the 1D Poisson problem

$$\begin{aligned} -u_{xx} &= f, \qquad \text{in } \Omega = (0,1), \\ u(0) &= u(1) = 0. \end{aligned}$$



- These are called homogenous Dirichlet boundary conditions: the solution is prescribed to be zero on the boundaries of the domain.
- Introduce the grid, $\{x_i\}_{i=0}^{N}$, with $x_i = x_0 + ih$, $h = \frac{1}{N}$.

Discretization in 1D

Local equations can also be expressed as the system

$$\begin{aligned} 2u_1 - u_2 &= h^2 f_2 \\ -u_1 + 2u_2 - u_3 &= h^2 f_3 \\ &\vdots \\ -u_{N-2} + 2u_{N-1} &= h^2 f_{N-1}, \end{aligned}$$

or in matrix form

$$\underbrace{\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{pmatrix}}_{\mathbf{u}} = h^2 \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{pmatrix}}_{\mathbf{f}},$$

Comparison

Physical model

Fluid dynamics
Describe the evolution of physical quantities
May not have good mathematical properties
May have limited application field

Continuous problem

Mathematical Analysis
Provides a well-posed set of equations
Existence and uniqueness
Should enforce physical constraints

Discrete problem

Numerical Analysis
Design a computable approximation
Balance between robustness and accuracy
May not respect physical constraints

Outline

Mathematical model: Implicit LES

Incompressible Navier–Stokes equations:

$$
\left|\begin{array}{rcll}
\multicolumn{4}{c}{\text{Find } \mathbf{W} \equiv (\mathbf{u}, p) \text{ such that:}} \\
\partial_t \mathbf{u} + (\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u} + \boldsymbol{\nabla} p - 2\nu\boldsymbol{\nabla}\cdot\,\epsilon(\mathbf{u}) & = & \mathbf{f} & \text{in } \Omega\times[0,T] \\
\boldsymbol{\nabla}\cdot\mathbf{u} & = & 0 & \text{in } \Omega\times[0,T] \\
\hat{\mathbf{u}}(\cdot,0) & = & \hat{\mathbf{u}}^0 & \text{in } \Omega
\end{array}\right.
$$

Definition (Implicit LES)

"Numerical dissipation acts on small scales similarly to an explicit subgrid model" (Boris, 1989; Sagaut, 1998)

- Focus on fairly High Reynolds number ($10^5 - 10^6$)
- Consider NSE as regularization of the incompressible Euler equations, take $\nu = 0$
- Unresolved scales modelled by numerical dissipation from residual-based stabilization
- No explicit/physical subgrid model needed

Finite Elements: choice of different numerical schemes

Stabilizations for conservation laws, find $u \in V_h$ such that:

- **Galerkin:**

$$(\partial_t u + \nabla \cdot F(u), v) = 0, \quad \forall v \in V_h$$

- **Galerkin–Least Square.**
- **Streamline-diffusion/SUPG:**

$$(\partial_t u + \nabla \cdot F(u), v + \delta(\partial_t v + F'(u)\nabla \cdot v)) + (\varepsilon \nabla u, \nabla v) = 0, \quad \forall v \in V_h$$

- **Residual-based nonlinear viscosity, RV/EV:**

$$(\partial_t u + \nabla \cdot F(u), v) + (\varepsilon \nabla u, \nabla v) = 0, \quad \forall v \in V_h$$

$$\varepsilon = \min(C_1\, h\, |u|, C_2 h^2 |R(u)|)$$

or

$$\varepsilon = \min(C_1\, h\, |u|, C_2 h^2 |D(u)|)$$

with $D(u)$ entropy residual for some $E(u)$ convex.

General Galerkin: Numerical Scheme

- $\mathbf{V}_h$ and X_h linear Lagrange approximations.
- Find $(\mathbf{u}, p) \in \mathbf{V}_h \times \mathrm{X}_h$, such that $\forall (\mathbf{v}, q) \in \mathbf{V}_h \times \mathrm{X}_h$

$$\int_\Omega \delta t^{-1}(\mathbf{u} - \mathbf{u}^*) \cdot \mathbf{v} + \int_\Omega (\bar{\mathbf{u}} \cdot \boldsymbol{\nabla})\bar{\mathbf{u}} \cdot \mathbf{v} + \int_\Omega 2\nu\epsilon(\bar{\mathbf{u}}) : \epsilon(\mathbf{v})$$

$$- \int_\Omega p(\boldsymbol{\nabla} \cdot \mathbf{v}) + \int_\Omega (\boldsymbol{\nabla} \cdot \bar{\mathbf{u}})q + S_h^{sd}(\bar{\mathbf{u}}, p; \mathbf{v}, q) = \int_\Omega \mathbf{f} \cdot \mathbf{v}$$

with $\bar{\mathbf{u}} = 1/2(\mathbf{u} + \mathbf{u}^*)$.

- Streamline-Diffusion $S_h^{sd}(\bar{\mathbf{u}}, p; \mathbf{v}, q) \equiv$

$$\int_\Omega \delta_{\mathcal{M}} \left[(\bar{\mathbf{u}} \cdot \boldsymbol{\nabla})\bar{\mathbf{u}} + \boldsymbol{\nabla} p - \mathbf{f}\right] \cdot \left[(\bar{\mathbf{u}} \cdot \boldsymbol{\nabla})\mathbf{v} + \boldsymbol{\nabla} q\right] + \int_\Omega \varepsilon_{\mathcal{M}} (\boldsymbol{\nabla} \cdot \bar{\mathbf{u}})(\boldsymbol{\nabla} \cdot \mathbf{v})$$

with $\forall K \in \mathcal{M}_\langle$,
$$\begin{aligned} \delta_K &= C_1 \left[\frac{1}{\delta t^2} + \frac{\|\mathbf{u}^*\|_K^2}{h_K^2}\right]^{-1/2} \\ \varepsilon_K &= C_2 \|\mathbf{u}^*\|_K h_K \end{aligned}$$

General Galerkin solver: Boundary layer modelling

- Resolving boundary layers for High-Re is prohibitive.
  $\rightarrow$ Choice of a cheap boundary model [J. Hoffman and N. Jansson, 2010].
- Slip-friction resistance and penetration BC on Γ_w

$$\int_{\Gamma_w} \alpha^{-1} (\mathbf{u} \cdot \mathbf{n})(\mathbf{v} \cdot \mathbf{n}) + \sum_{k=1}^{d-1} \beta (\mathbf{u} \cdot \tau_k)(\mathbf{v} \cdot \tau_k)$$

- Enforce non-penetration strongly and friction weakly:

$$\begin{aligned} \mathbf{u} \cdot \mathbf{n} &= 0 && \text{(node)} \\ \text{wall shear stress contribution} &= \int_{\Gamma_w} \sum_{k=1}^{d-1} \beta (\mathbf{u} \cdot \tau_k)(\mathbf{v} \cdot \tau_k) && \text{(facet)} \end{aligned}$$

- Strong perfect slip BC for $Re \rightarrow \infty$ with $\beta = 0$

General Galerkin solver: Boundary layer modelling

- Enforce non-penetration strongly and friction weakly:

$$\begin{aligned} \mathbf{u}\cdot\mathbf{n} &= 0 && \text{(node)} \\ \text{wall shear stress contribution} &= \int_{\Gamma_w} \sum_{k=1}^{d-1} \beta(\mathbf{u}\cdot\tau_k)(\mathbf{v}\cdot\tau_k) && \text{(facet)} \end{aligned}$$

- Strong perfect slip BC for $Re \to \infty$ with $\beta = 0$

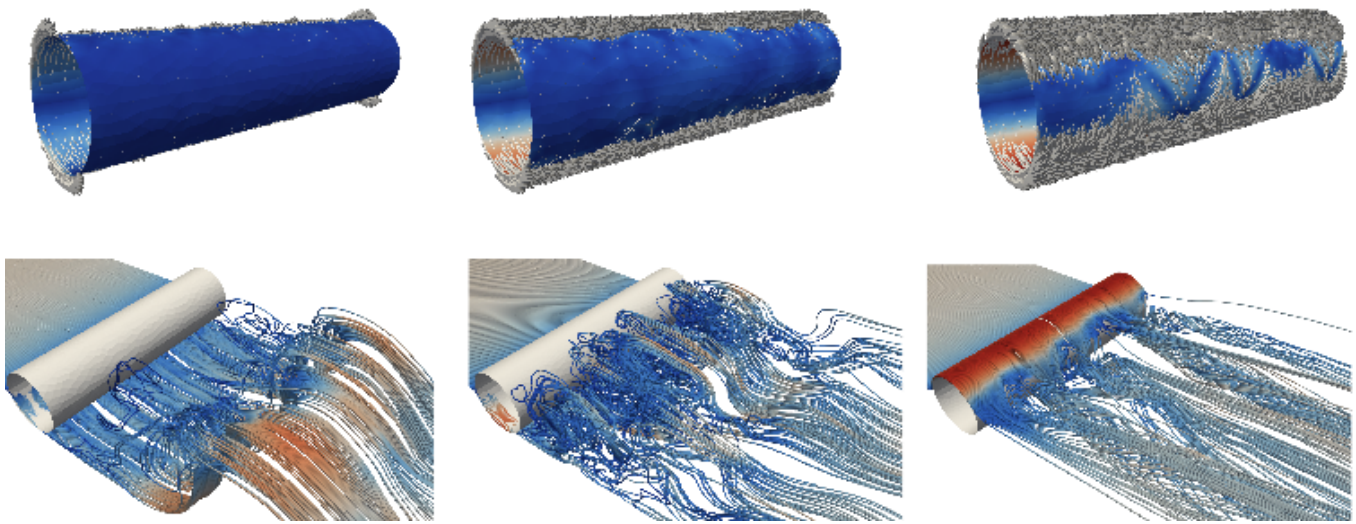


Figure: Parametric study, $\beta = 10^{-1}, 10^{-2}, 10^{-3}$ [J. Hoffman and N. Jansson,2011]

General Galerkin solver: Goal-Oriented Adaptivity

Based on *a posteriori* error estimation of an output functional.

- Solve primal problem forward in time for $\hat{\mathbf{u}}_h = (\mathbf{u}, p)$.
- Define objective functional $M(\hat{\mathbf{u}}) = \int_Q \hat{\mathbf{u}}\psi$ (*e.g.* Drag).
- Solve a linearized dual problem backward in time for $\hat{\varphi}_h = (\mathbf{v}, q)$.
- Error estimate with indicators of the form

$$\mathcal{E}_K \equiv \sum_{n=1}^{N} \Big[\int_{I_n} \sum_i |R_i(\hat{\mathbf{u}}_h)|_K \cdot \omega_i \; dx \; dt + \int_{I_n} |SD_\delta^n(\hat{\mathbf{u}}_h; \hat{\varphi})_K| \; dt \Big]$$

  $\forall K \in \mathcal{M}_h$, with $R_i(\hat{\mathbf{u}}_h)$ residuals of equations, ω_i weight functions depending on $h, \delta t, \hat{\varphi}$
- Refine $5 - 10$ % of the cells, iterate again ...

Progress in numerical methods and software

Progress cannot be achieved only by raw performance:

- improvement of linear solvers,
- model reduction,
- uncertainty quantification,
- development of new programming models,
- ...

Each stage of the development of a new method requires care:

- Implementation: unit testing
- Algorithms: verification
- Conceptual model: validation
- Error propagation, Reliability: uncertainty quantification

Unit testing

These tests are only to **check** the logic:

1. pre- and post-conditions (e.g. bounds)
2. invariants
3. assertions on data structures (initialization, size)

How is it addressed?

- Debug mode
- Using test suites in serial (CHECK)
- Some parallel tests in place

Relevant to: common logic, parallel algorithm (test execution ¡ 1min)

Verification

These tests are only to **evaluate** the mathematical properties of algorithms:

1. convergence analysis (accuracy)
2. test discontinuous solutions (stability)
3. stress the numerical scheme w.r.t free parameters (stability)

How is it addressed?

- Exhibit analytic solutions or construct solutions and compare.
- Parametric study stressing the algorithm w.r.t stability

Relevant to: numerical methods (test execution ¡ 10min)

Validation

These tests are only to compare the value of quantities of interest with benchmarks:

1. Direct Numerical Simulation
2. Experimental data

How is it addressed?

- Setup academic test cases computing e.g drag, lift.

Relevant to: discretization and physical modelling (test execution in hours)

Reliability

These tests are only to **assess** the sensitivity of the algorithms with respect to:

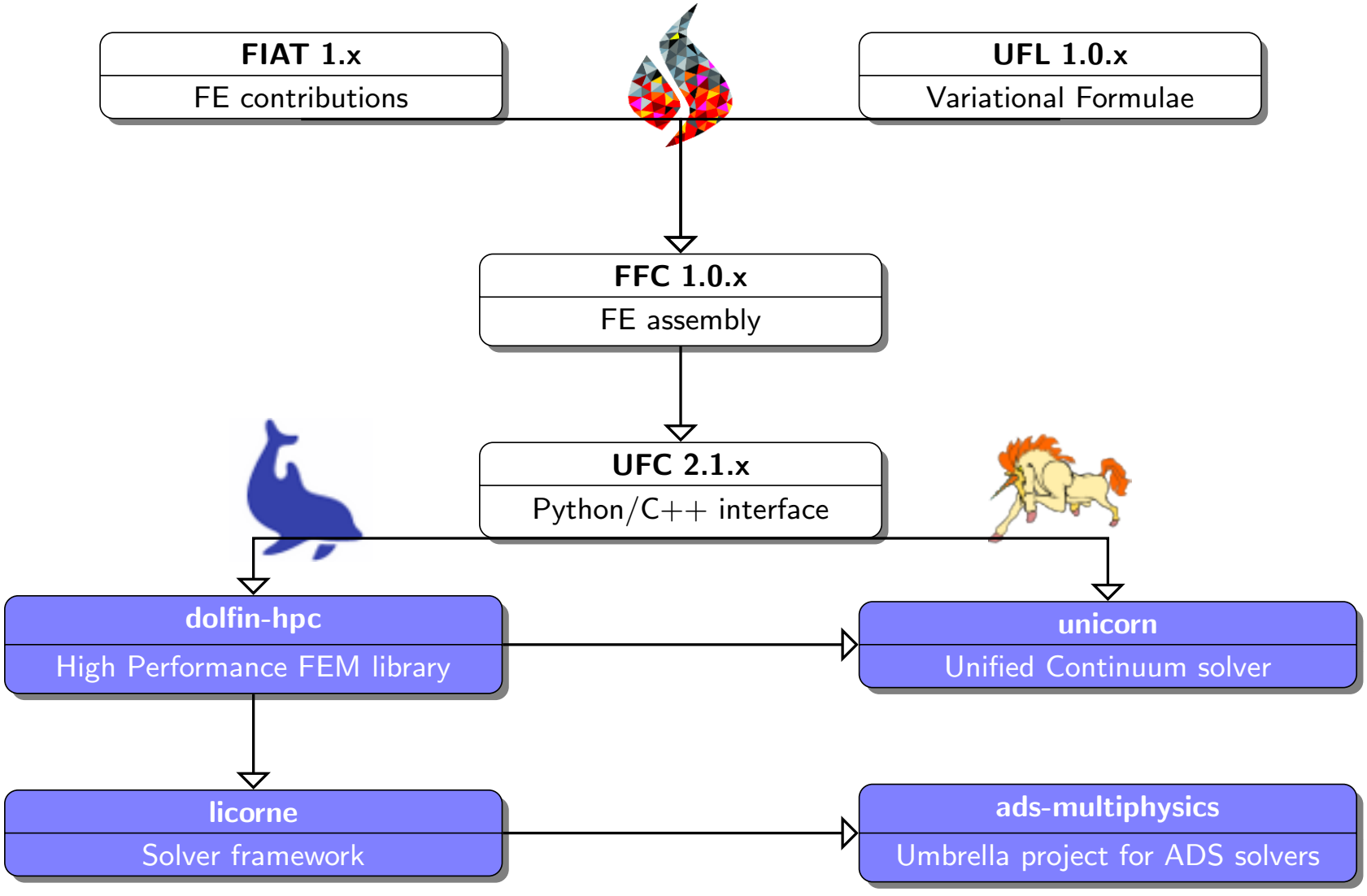
1. model uncertainties (constants, geometry)
2. numerical error propagation
3. experimental data measurement tolerance

How is it addressed?

- Various Uncertainty Quantification methods (MC, sensitivity gradients, ...)
- P-Boxes

Relevant to: all kinds of error/uncertainty propagation (test execution in hours)

FEniCS-based Software for Adaptive Finite Elements



- Supporting libraries
- Solver packages

FEniCS: Writing the Variational Form in Python

Copy/Paste form file

```
element = FiniteElement("Lagrange", triangle, 1)

u = TrialFunction(element)
v = TestFunction(element)
f = Coefficient(element)
g = Coefficient(element)

a = inner(grad(u), grad(v))*dx
L = f*v*dx + g*v*ds
```

Definition of a discrete space

To construct the approximate space V_h, we need two ingredients:

1. An admissible mesh $\mathcal{T}_h$ generated by a triangulation of domain Ω.
2. A reference finite element $(\hat{K}, \hat{\mathcal{P}}, \hat{\Sigma})$ to construct a basis of V_h.

We have the mesh as input data: how about the finite element ?

Reference element

Definition (Finite Element – Ciarlet)

A Finite Element consists of a triple $(K, \mathcal{P}, \Sigma)$, such that

- K is a compact, connected subset of $\mathbb{R}^d$ with non-empty interior and with regular boundary (typically Lipshitz continuous),
- $\mathcal{P}$ is a finite dimensional vector space, $\dim(\mathcal{P}) = N$, of functions $p : K \to \mathbb{R}$, which is the space of shape functions,
- Σ is a set $\{\sigma_j\}$ of linear forms,

$$\begin{array}{rccl} \sigma_j : & \mathcal{P} & \to & \mathbb{R} \\ & p & \mapsto & p_j = \sigma_j(p) \end{array} \qquad , \forall\, j \in [\![1, N]\!]$$

  which is a basis of $\mathcal{L}(\mathcal{P}, \mathbb{R})$, the dual of $\mathcal{P}$.

$\Rightarrow$ In our case, define a cell K, degrees of freedom $\{u_i = u(x_i)\}$ and basis functions $\{\hat{\phi}_i\}$ as Lagrange P1 on triangle.

- UFL constructs an abstraction of the discrete problem.
- FFC generates C++ code complying with UFC interface.
- dolfin-hpc handles mesh algorithms and assembly of the linear system.