Master of Science on Computational Science

Institute of Computational Science

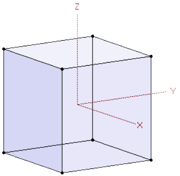
prof. Dr. Rolf Krause & Dr. Drosos Kourounis

17 Sep 2015, LAB1

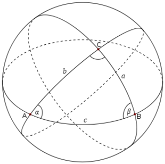
Solution of Partial differential equations (PDEs)

Analytical methods can be used only in specific 2D or 3D geometries

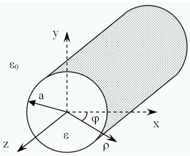
Cartecian



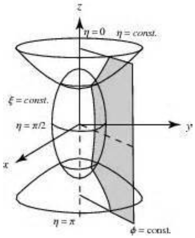
Spherical



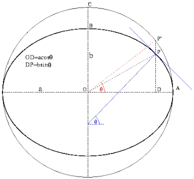
Cylindrical



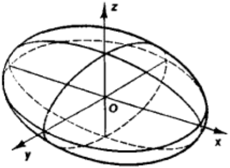
Spheroidal (prolate)



Spheroidal (oblate)



Ellipsoidal



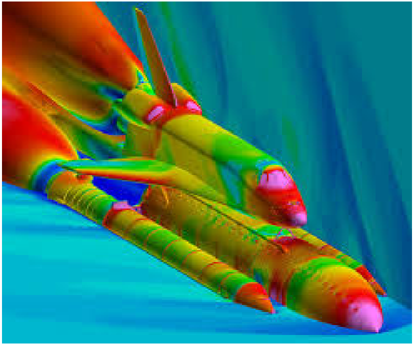
Why Finite Elements?

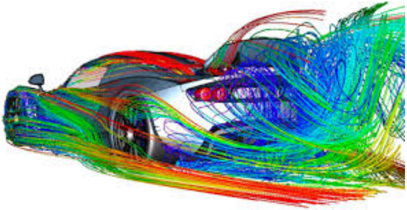
Finite elements are flexible enough to handle

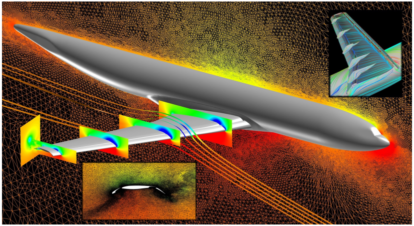
- Any geometry
- Inhomogeneous coefficients
- Anisotropic tensors fields
- Arbitrary boundary conditions

Lets see some examples ...

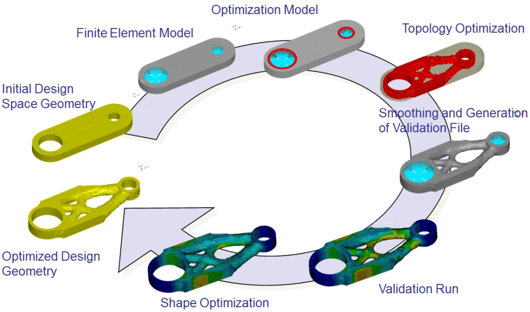
Aerodynamics

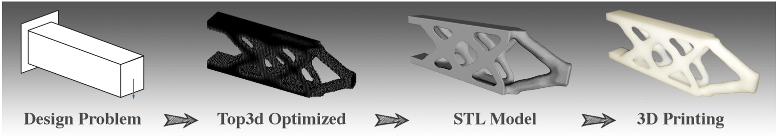




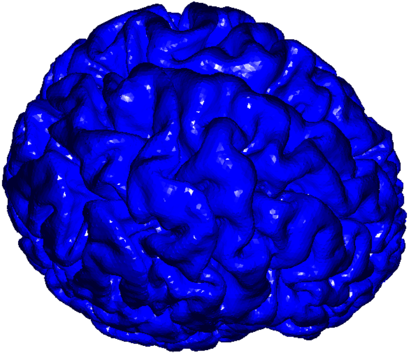


Structural Optimization

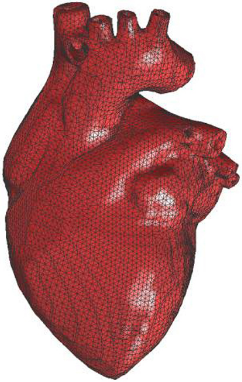




Brain, heart: biomedical applications

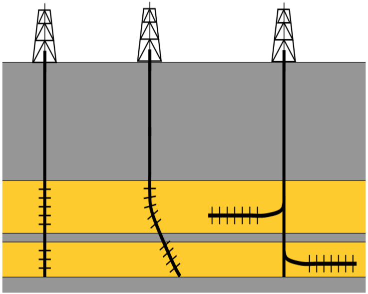


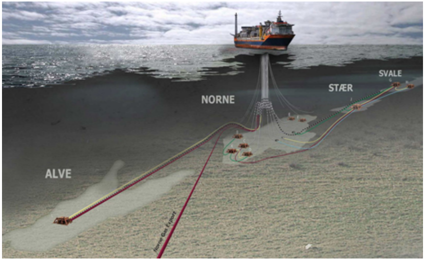
Source localization



Heart simulation

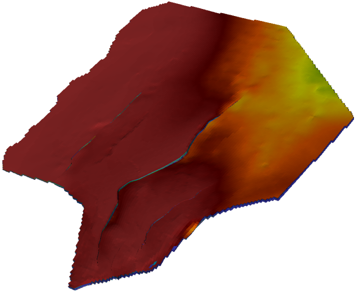
Reservoir simulation



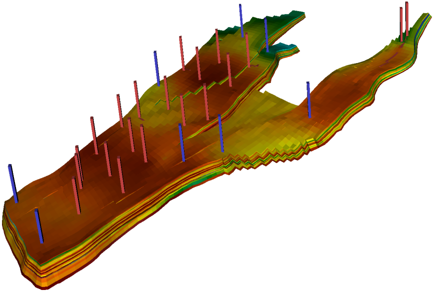


Reservoir simulation

Johansen formation



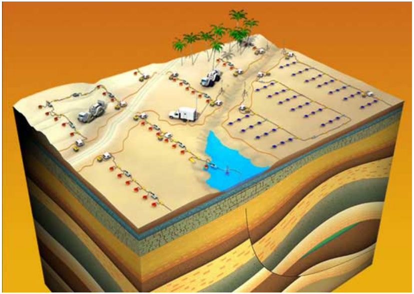
Norne field

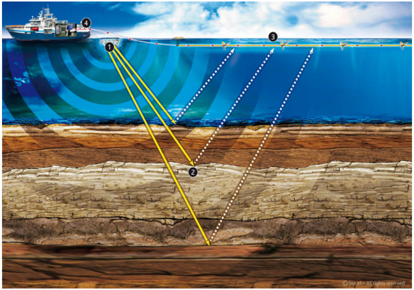


Inhomogeneous and anisotropic tensor fields (permeability)

Seismic Inversion

A framework for multi-scale seismic modelling and inversion (ETH Zurich)





Identifiability in FWI

$$\frac{\partial^2 \mathbf{u}}{\partial t^2} - \nabla \cdot \boldsymbol{\sigma} = \mathbf{f}$$

- Acoustic, elastic, poroelastic waves
- Algorithms aware of limited data
- Optimize seismic experiments

The linear system

After discretization in space or space and time we end up with

$$A\,x = b, \quad \text{or in the time dependent case,}$$
$$A_n\,x_n = b_n$$

Solution techniques

- Direct sparse methods
- Iterative methods

Direct sparse solvers

Complexity in 2D

- $O(N^{1.5})$ factorization
- $O(N \log N)$ solution

Complexity in 3D

- $O(N^2)$ factorization
- $O(N^{1.5})$ solution

Available Software

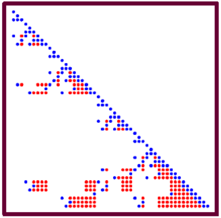
- PARDISO
- UMFPACK, CHOLMOD
- SUPERLU
- MUMPS

Efficient linear algebra

$\boxed{Ax = b \quad A \in R^{n\times n}}$

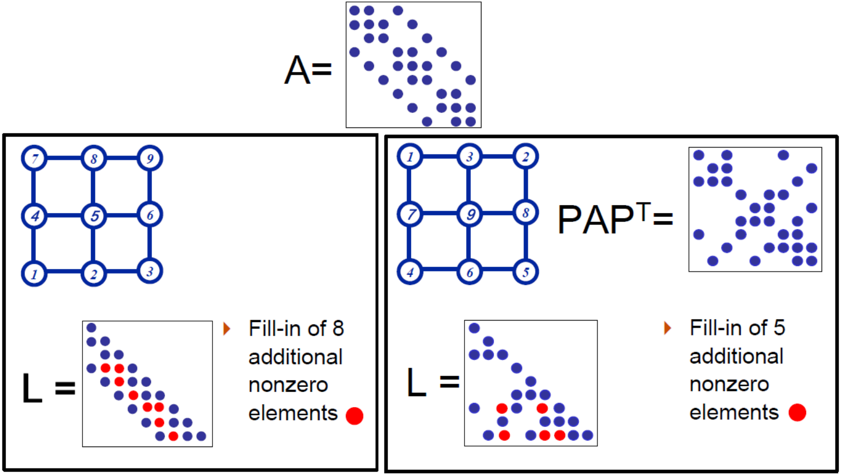
$\boxed{PQAP^T Px = PQb \quad A \in R^{n\times n}}$

$$\begin{aligned} A &= LU \\ LUx &= b \\ Ly &= b \\ Ux &= y \end{aligned}$$

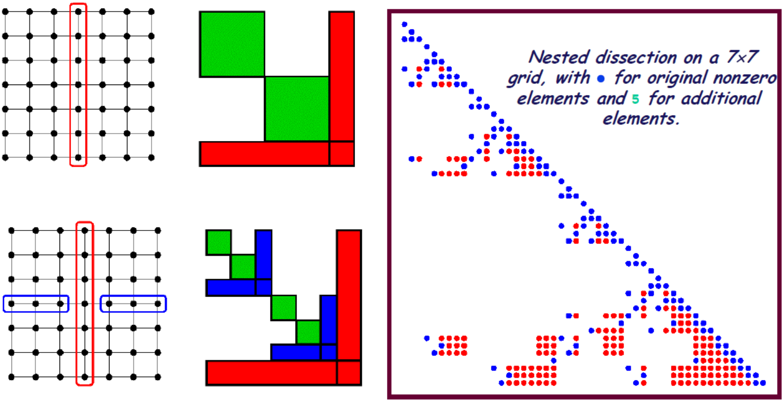


$$\begin{aligned} PQAP^T &= LU \\ LUPx &= PQb \\ Ly &= PQb \\ UPx &= y \end{aligned}$$

- Permutations P and Q chosen to preserve **sparsity** and **maintain stability** in $PAQ = LU$.
- L = Lower triangular, U = Upper triangular (**sparse**)


A=
7
8
9
4
5
6
1
2
3
L =
Fill-in of 8 additional nonzero elements
1
3
2
7
9
8
4
6
5
PAP^T=
L =
Fill-in of 5 additional nonzero elements

Nested dissection ordering on a 7×7 grid



Direct Sparse Solvers

PARDISO performance in 2D

mesh	Elapsed time in seconds				memory
nodes	init	fact	back-sub	total	MB
65175	0.704	0.265	0.019	0.988	26.0
263838	3.199	1.539	0.111	4.849	117.5
1068112	14.766	9.518	0.535	24.819	527.7

PARDISO performance in 3D

mesh	Elapsed time in seconds				memory
nodes	init	fact	back-sub	total	MB
32002	0.636	2.783	0.069	3.488	80.4
256011	6.925	185.469	1.428	193.8	1438.3
2000396	76.625	11762.1	21.46	11860	24403.0

Iterative Krylov subspace methods

They work with matrix-vector products Ay for given vectors y

Symmetric systems

- **PCG**
- **MINRES**
- **SYMMLQ**
- **LSMR**
- **SQMR**

Nonsymmetric systems

- **GMRES**
- **CGS**
- **BICGSTAB**
- **QMR**
- **LSQR**

Preconditioners

$$A\,x = b$$

$$\text{left:} \quad P^{-1}A\,x = P^{-1}b$$

$$\text{right:} \quad AP^{-1}\,y = b, \quad P\,x = y$$

Iterative Krylov subspace methods

At each timestep of the simulation whether we solve linear or nonlinear PDEs we need to solve until convergence one or several linear systems:

At the nth timestep

$$A_n\, x_n = b_n$$

Left preconditioning

$$P_n^{-1}(A_n\, x_n) = P_n^{-1}\, b_n$$

High condition number

Lower condition number

Iterative Krylov subspace methods require only matrix-vector products Ay for given vectors y and depending on the type of the matrix we have

Symmetric matrices

- **PCG**
- **MINRES**

Nonsymmetric matrices

- **GMRES**
- **BICGSTAB**

Generalized minimum residual methods

GMRES(A,M,b,tol)

$x_0 = M^{-1}b,\ r_0 = b - Ax_0,$

$\beta = \|r_0\|_2\ u_1 = \dfrac{r_0}{\beta},\ k = 0$

while $\|r_k\|_2 > \beta\ tol$

$\quad k = k + 1\ z_k = M^{-1}v_k,\ w = Az_k$

$\quad$ for $i = 1, 2, \ldots, k$ do

$\qquad h_{i,k} = u_i^T w,\ w = w - h_{i,k}u_i$

$\quad$ end for

$\quad h_{k+1,k} = \|w\|_2,\ u_{k+1} = \dfrac{w}{h_{k+1,k}}$

$\quad V_k = [v_1, \ldots, v_k]$

$\quad H_k = \{h_{i,j}\},\ 1 \le i \le j + 1,\ 1 \le j \le k$

$\quad y_k = \text{argmin}_y \|\beta e_1 - H_k\, y\|_2$

$\quad x_k = x_0 + M^{-1}V_k\, y_k,\ r_k = b - A\, x_k$

end while

FGMRES(A,M,b,tol)

$x_0 = M_0^{-1}b,\ r_0 = b - Ax_0,$

$\beta = \|r_0\|_2\ u_1 = \dfrac{r_0}{\beta},\ k = 0$

while $\|r_k\|_2 > \beta\ tol$

$\quad k = k + 1, z_k = M_k^{-1}v_k,\ w = Az_k$

$\quad$ for $i = 1, 2, \ldots, k$ do

$\qquad h_{i,k} = u_i^T w,\ w = w - h_{i,k}u_i$

$\quad$ end for

$\quad h_{k+1,k} = \|w\|_2,\ u_{k+1} = \dfrac{w}{h_{k+1,k}}$

$\quad V_k = [v_1, \ldots, v_k],\ Z_k = [z_1, \ldots, z_k]$

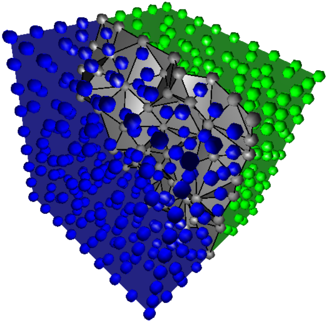
$\quad H_k = \{h_{i,j}\},\ 1 \le i \le j + 1,\ 1 \le j \le k$

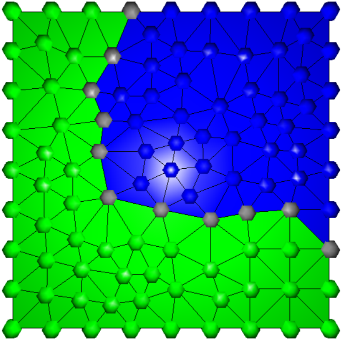
$\quad y_k = \text{argmin}_y \|\beta e_1 - H_k\, y\|_2$

$\quad x_k = x_0 + Z_k\, y_k,\ r_k = b - A\, x_k$

end while

Sophisticated iterative methods: domain decomposition





$$\begin{pmatrix} A_{11} & 0 & A_{1B} \\ 0 & A_{22} & A_{2B} \\ A_{1B}^T & A_{2B}^T & A_{BB} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_B \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_B \end{pmatrix}$$

Our journey to the Finite Element realm

will step through

1. *the* **Mesh generation** (IO)[a]
2. *the* **Discretization** (Hexahedra/Tetrahedra)
3. *the* **Linear Algebra** (Direct/Iterative solvers)
4. *the* **Sparse Linear Algebra**
5. *the* **Implementation** in MATLAB[b]
6. *the* **Assignments** in LaTeX[c]
7. *the* **Additional Software** in FEnICS[d]

---

[a] `https://wci.llnl.gov/simulation/computer-codes/visit/`
[b] `http://www.mathworks.ch/moler/exm/book.pdf`
[c] `https://www.macports.org` $sudo port install texlive +full
[d] `http://fenicsproject.org`