High-performance numerical simulation on distributed heterogeneous hardware platforms

Matthias Möller
Delft University of Technology, Department of Applied Mathematics

8th July 2019, SURFsara, Amsterdam

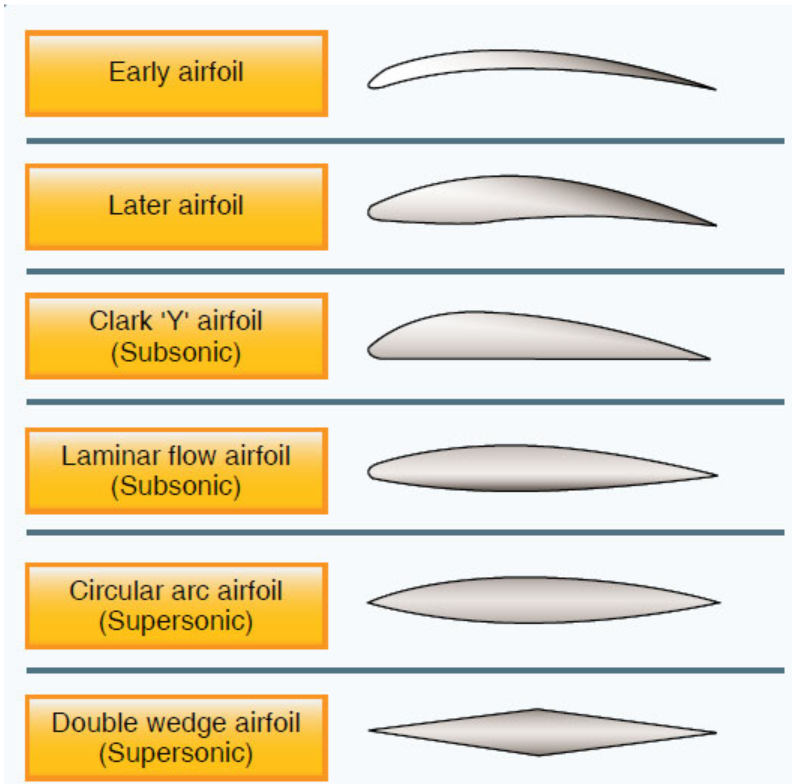
Overview

- Design-Through-Analysis
- Overview of Spline technologies
- DTA for twin-screw compressor
- Implementation aspects
- Recent updates and collaboration plans

Motivation

DESIGN-THROUGH-ANALYSIS

Example: Airfoil design



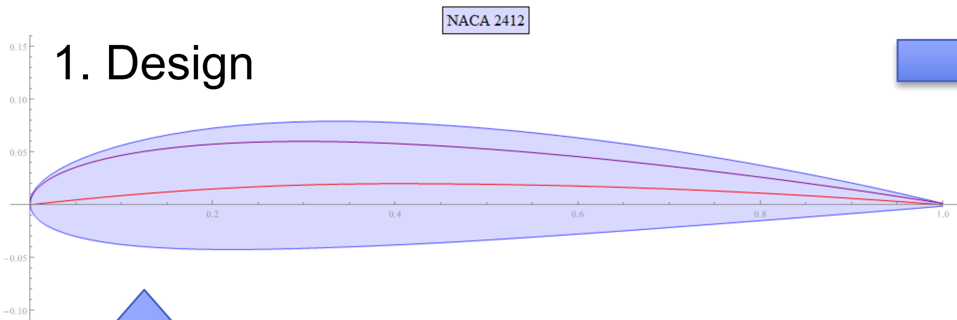
CFInotebook.net

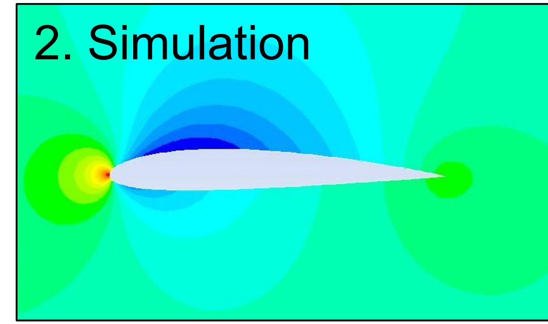
airspacemag.com

Image gallery at nasa.gov

Design-through-analysis cycle

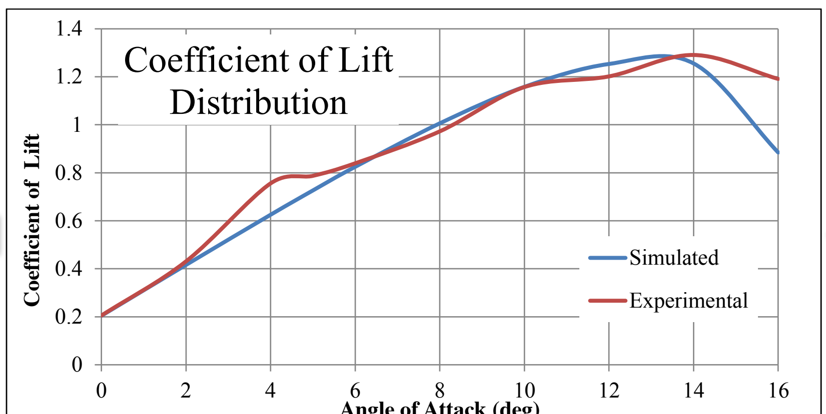
1. Design





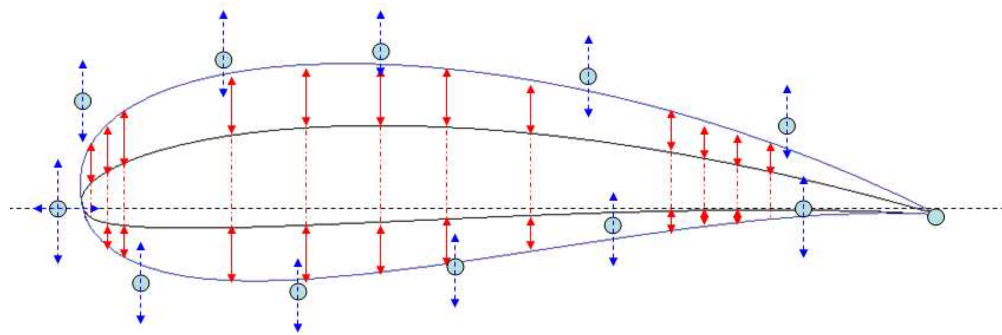
3. Analysis

4. Redesign



Matsson et al. Aerodynamic Performance of the NACA 2412 Airfoil at Low Reynolds Number, 2016 ASEE Annual Conference & Exposition

DTA: 1. Design $D(\boldsymbol{p})$



- Design parameters

$$\boldsymbol{p} = (p_1, \dots, p_{12})$$

- Admissible design space

$$\mathcal{S} = [p_1^{min}, p_1^{max}] \times \cdots \times [p_{12}^{min}, p_{12}^{max}]$$

Mauclère, Automatic 2D Airfoil Generation, Evaluation and Optimisation using MATLAB and XFOIL, Master thesis, 2009

DTA: 2. Simulation

- Mathematical model

Navier-Stokes Equations
3 – dimensional – unsteady

Glenn Research Center

Coordinates: (x,y,z)
Velocity Components: (u,v,w)

Time : t
Density: ρ
Total Energy: Et

Pressure: p
Stress: τ

Heat Flux: q
Reynolds Number: Re
Prandtl Number: Pr

Continuity:
$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0$$

X – Momentum:
$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} + \frac{\partial(\rho uw)}{\partial z} = -\frac{\partial p}{\partial x} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z}\right]$$

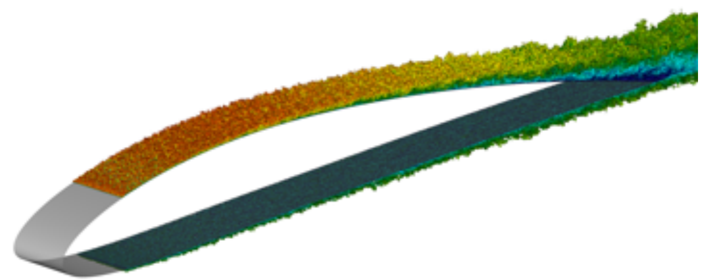
Y – Momentum:
$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho uv)}{\partial x} + \frac{\partial(\rho v^2)}{\partial y} + \frac{\partial(\rho vw)}{\partial z} = -\frac{\partial p}{\partial y} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z}\right]$$

Z – Momentum
$$\frac{\partial(\rho w)}{\partial t} + \frac{\partial(\rho uw)}{\partial x} + \frac{\partial(\rho vw)}{\partial y} + \frac{\partial(\rho w^2)}{\partial z} = -\frac{\partial p}{\partial z} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z}\right]$$

Energy:
$$\frac{\partial(E_T)}{\partial t} + \frac{\partial(uE_T)}{\partial x} + \frac{\partial(vE_T)}{\partial y} + \frac{\partial(wE_T)}{\partial z} = -\frac{\partial(up)}{\partial x} - \frac{\partial(vp)}{\partial y} - \frac{\partial(wp)}{\partial z} - \frac{1}{Re_r Pr_r}\left[\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}\right]$$
$$+ \frac{1}{Re_r}\left[\frac{\partial}{\partial x}(u\tau_{xx} + v\tau_{xy} + w\tau_{xz}) + \frac{\partial}{\partial y}(u\tau_{xy} + v\tau_{yy} + w\tau_{yz}) + \frac{\partial}{\partial z}(u\tau_{xz} + v\tau_{yz} + w\tau_{zz})\right]$$

- Solution for one particular design and one particular angle of attach

$$U = U(D(\boldsymbol{p}); AoA)$$

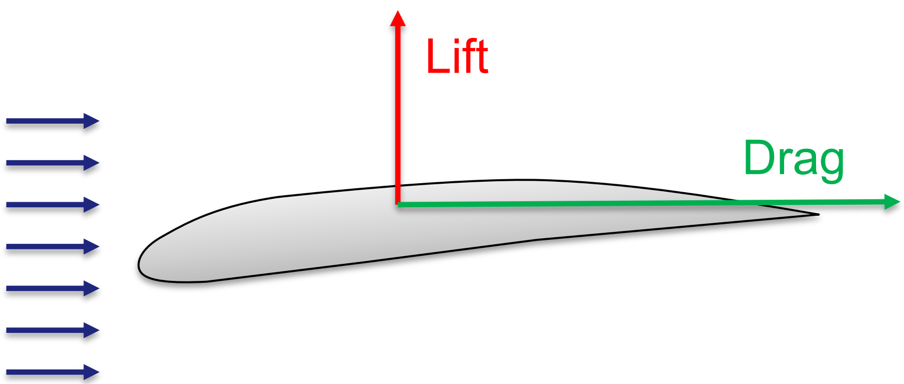


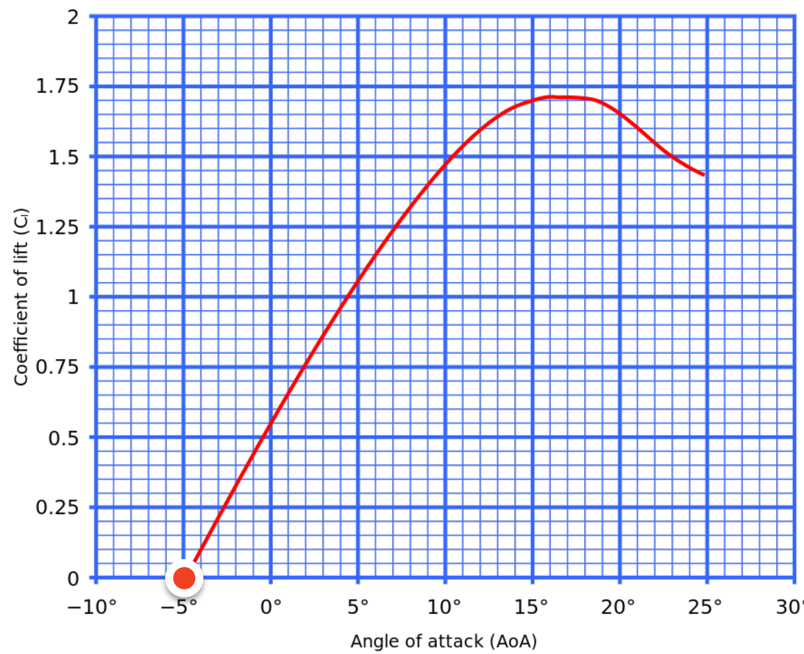
Linné FLOW Centre and SeRC, KTH, Sweden

DTA: 3. Analysis

- Cost functional

$$\mathcal{C}(U; D)$$

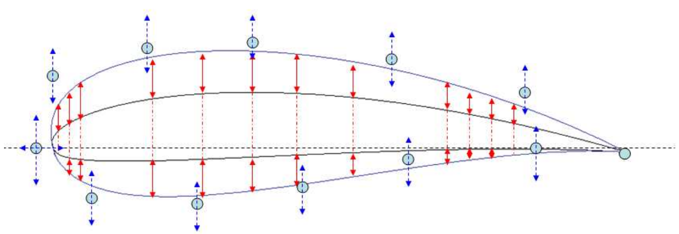


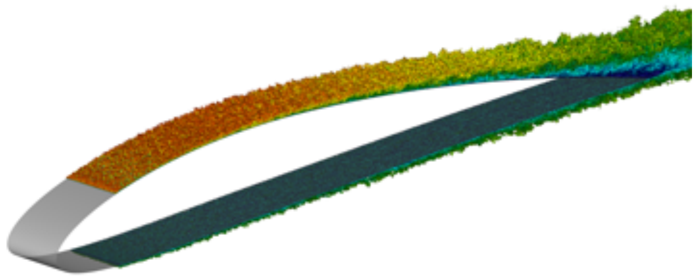


Example: Operation conditions

Design-through-analysis

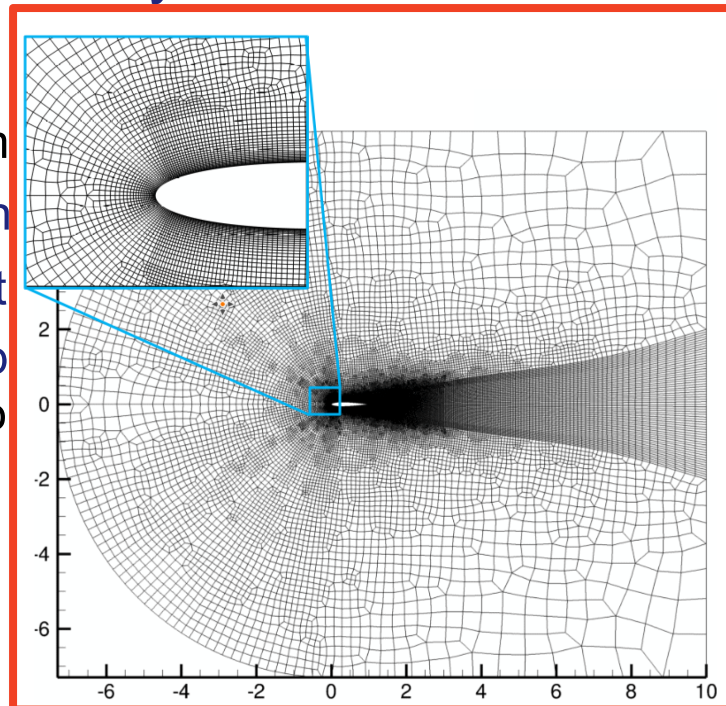
1. Find a set of admissible design parameters $\boldsymbol{p}$ and generate the design $D(\boldsymbol{p})$
2. Compute solutions $U(D(\boldsymbol{p}); AoA)$ to the mathematical model $\mathcal{M}\big(U, D(\boldsymbol{p})\big)$
3. Evaluate the cost functional $\mathcal{C}(U, D(\boldsymbol{p}))$ for all solutions/operating conditions
4. Vary the design parameters $\boldsymbol{p}$ to optimize the cost functional $\mathcal{C}(U, D(\boldsymbol{p}))$ for a wide range of operating conditions and repeat the DTA cycle at step 1

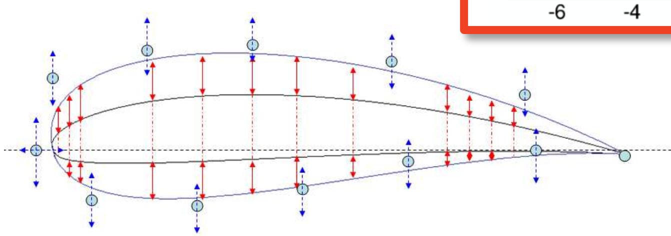


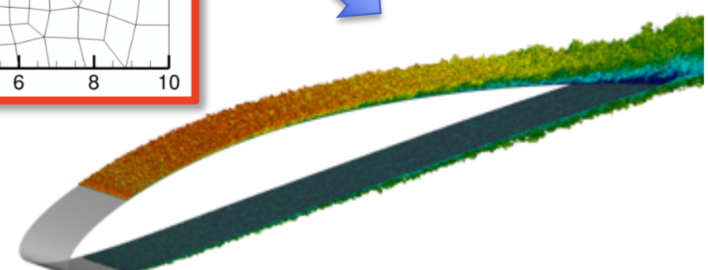


Design-through-analysis

1. Find a set of adm … enerate the design $D(\boldsymbol{p})$
2. Compute solution … al model $\mathcal{M}(U, D(\boldsymbol{p}))$
3. Evaluate the cost … ons/operating conditions
4. Vary the design p … unctional $\mathcal{C}(U, D(\boldsymbol{p}))$ for a wide range of o … DTA cycle at step 1



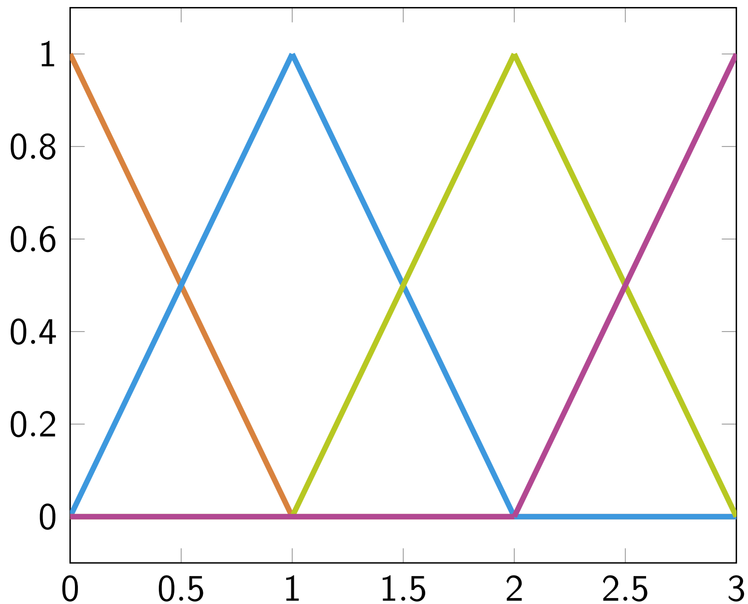




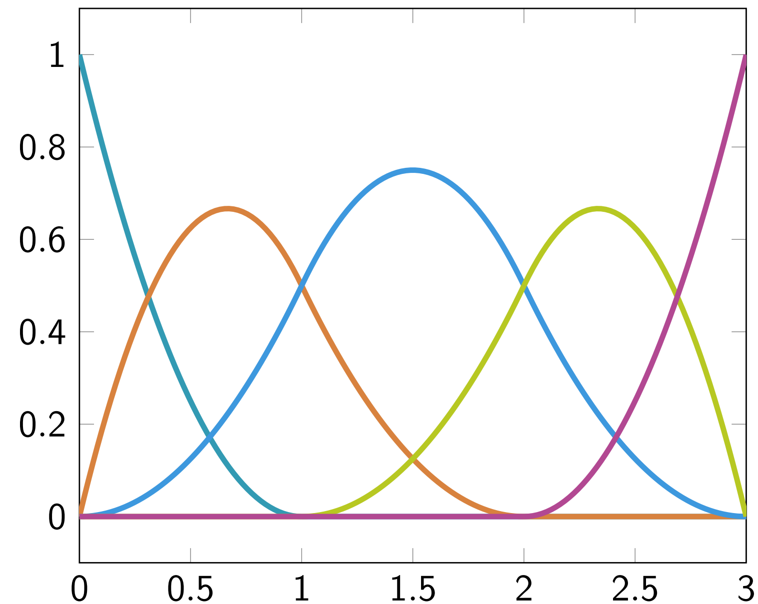
Introduction to

SPLINE TECHNOLOGIES

Basis functions: $\hat{B}_i(\xi) \quad i = 1, \dots, N$

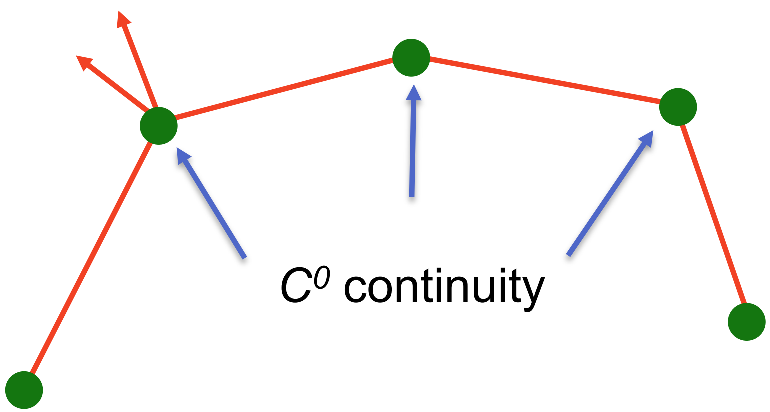


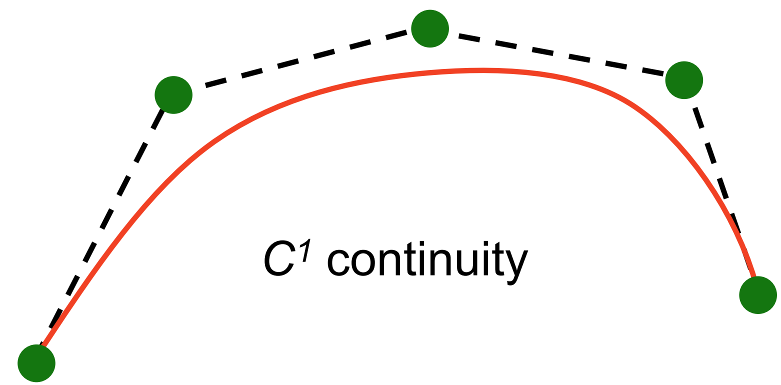
Linear ‘tent’ basis functions



Quadratic B-spline basis functions

Curves: $\boldsymbol{C}(\xi) = \sum_{i=1}^{N} \boldsymbol{c}_i \hat{B}_i(\xi) \quad : \quad [0,1] \to \mathbb{R}^d$

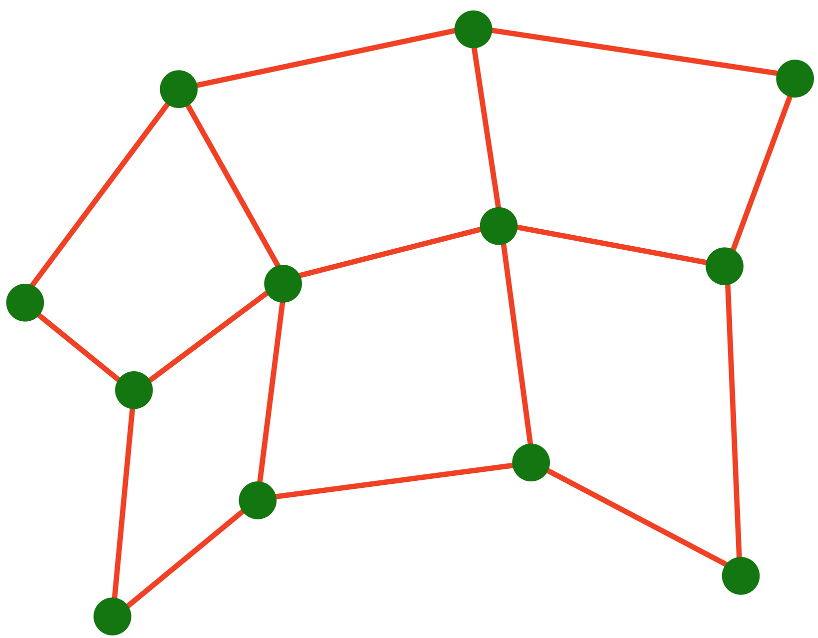




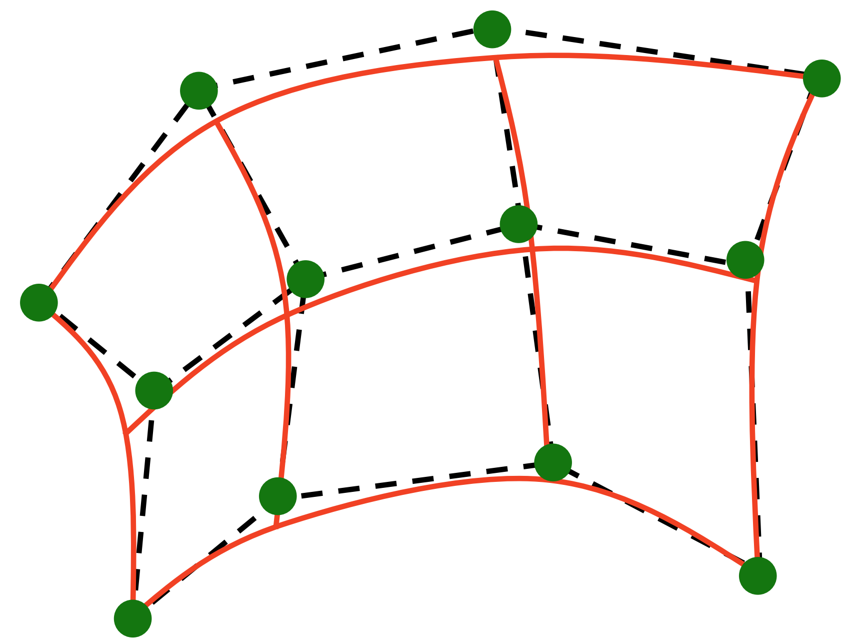
Linear 'tent' basis functions

Quadratic B-spline basis functions

Surfaces: $\boldsymbol{S}(\xi,\eta) = \sum_{i,j=1}^{N,M} \boldsymbol{c}_{i,j}\hat{B}_i(\xi)\hat{B}_j(\eta) \quad : \quad [0,1]\times[0,1] \to \mathbb{R}^d$

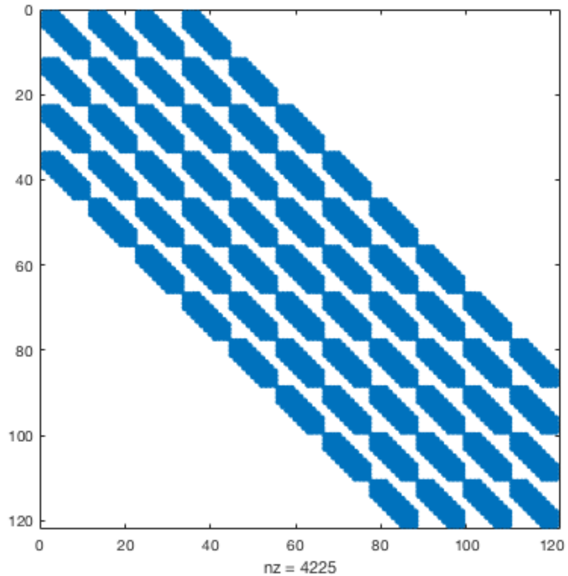


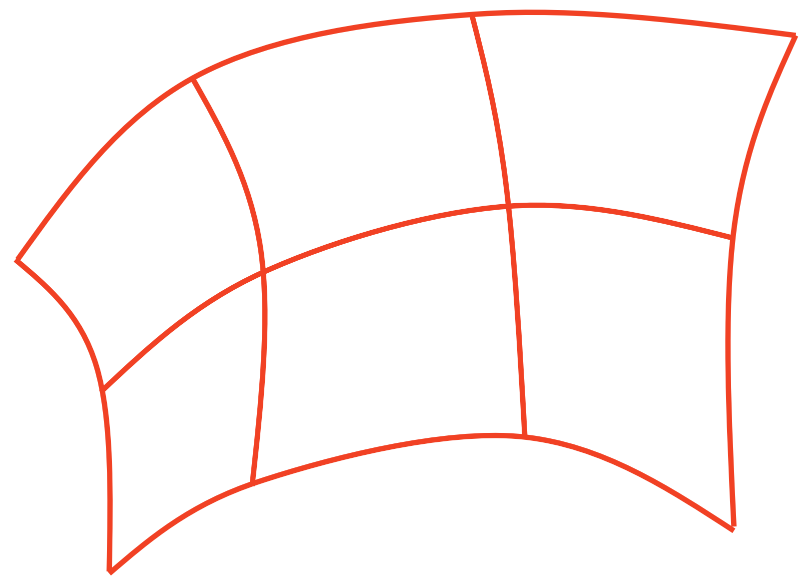
Bilinear basis functions



Biquadratic B-spline basis functions

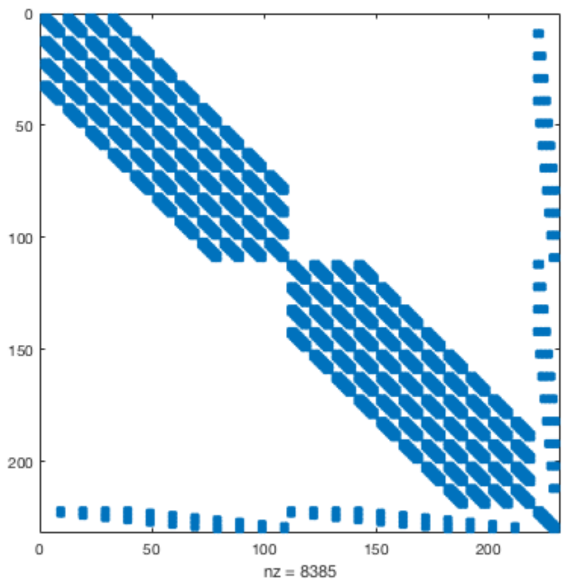
Matrix structure for single-patch domain

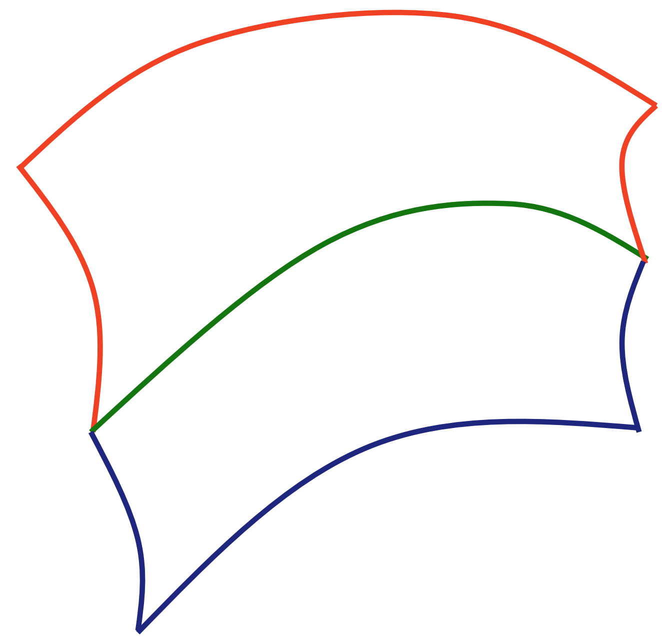




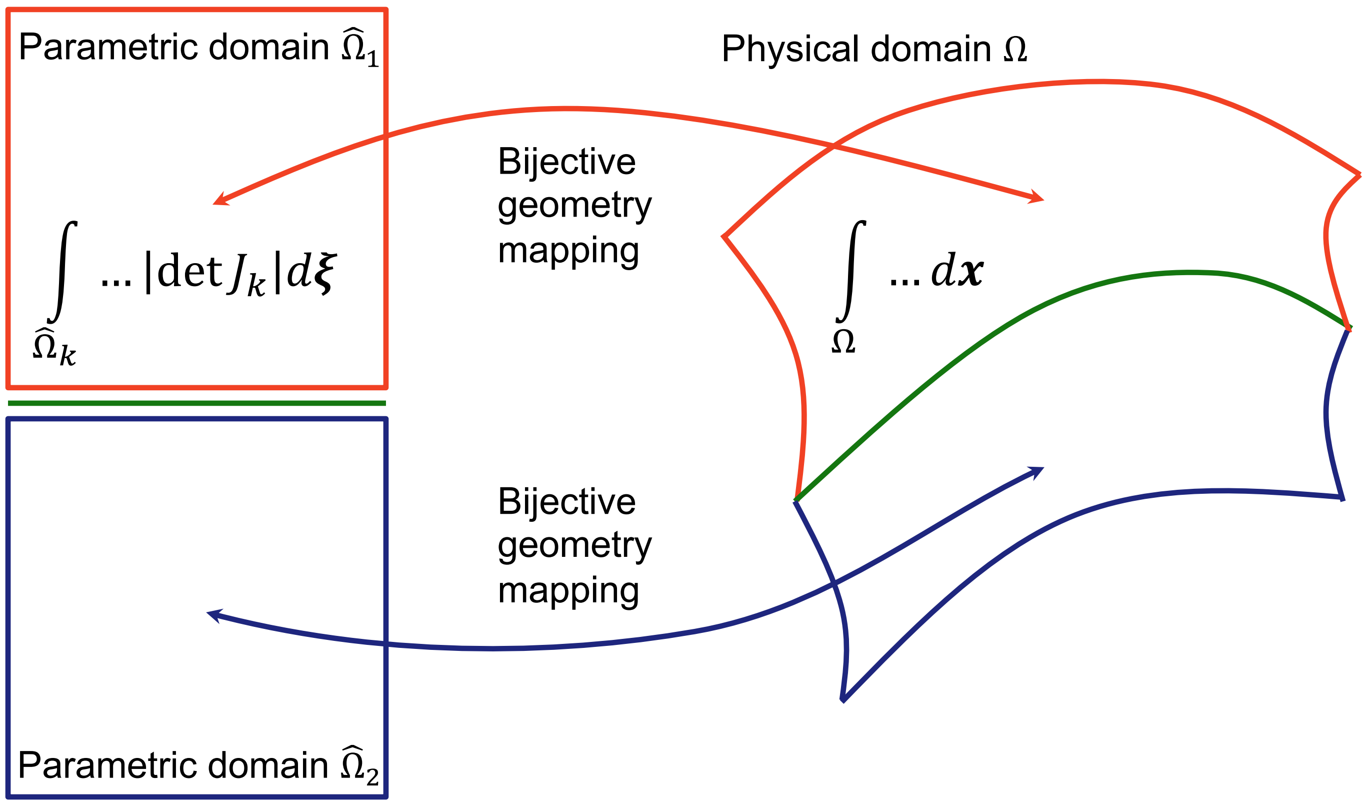
Biquadratic B-spline basis functions

Matrix structure for multi-patch domain

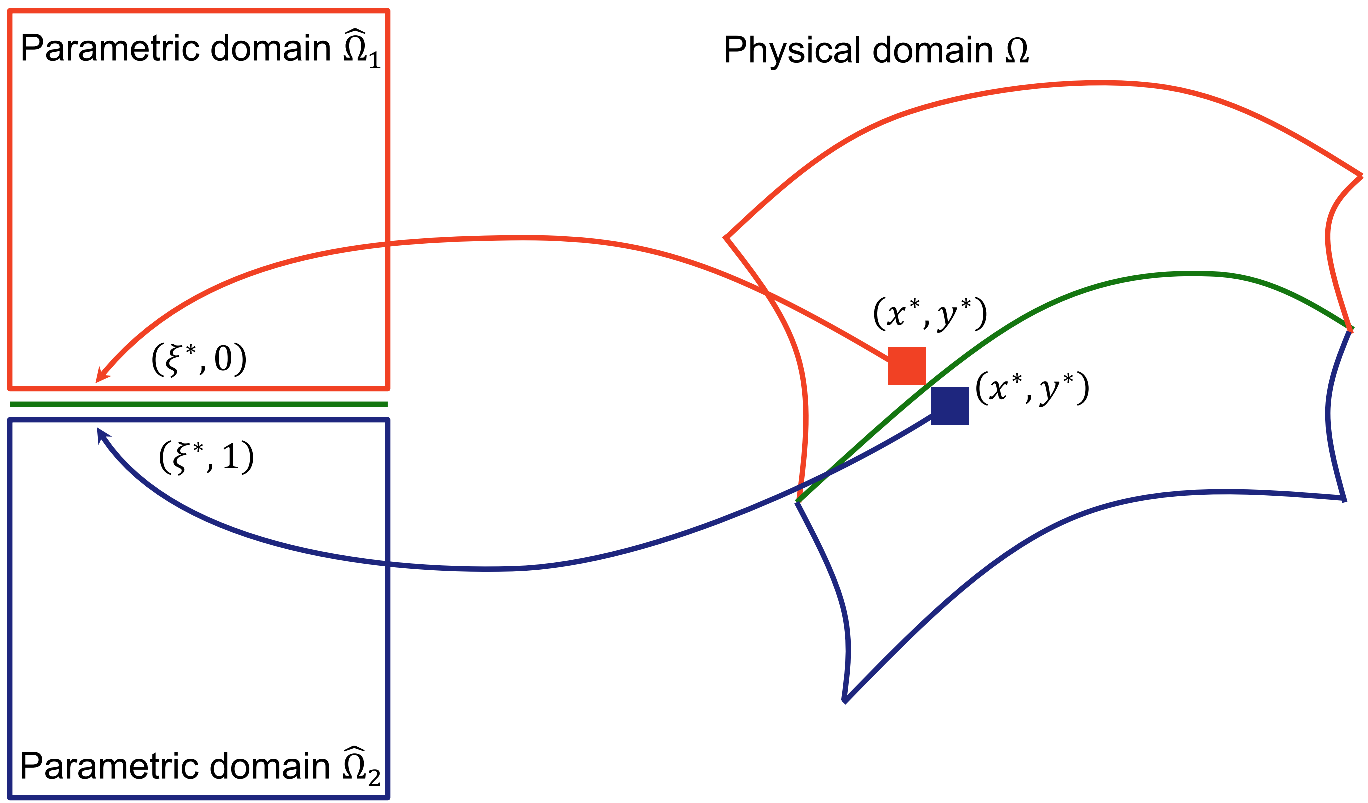




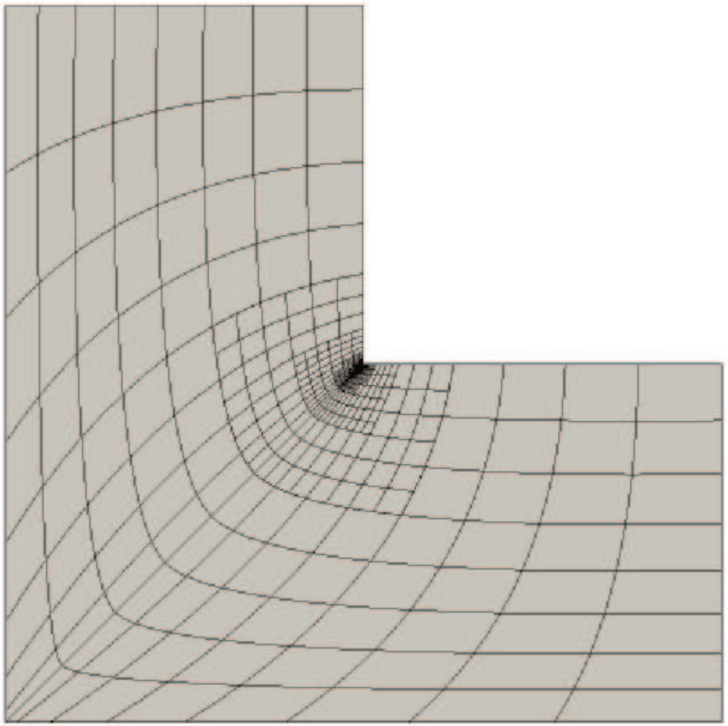
Isogeometric Analysis in a nutshell

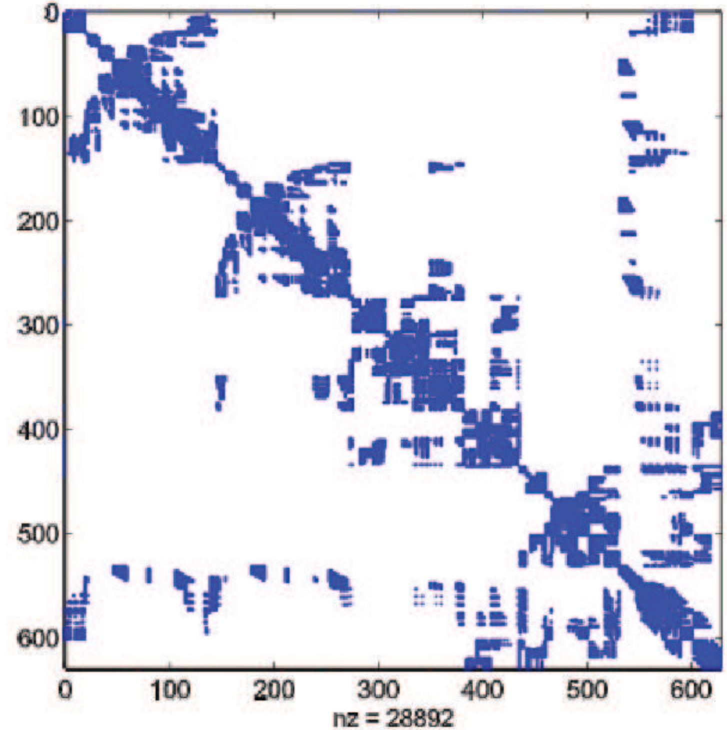


Isogeometric Analysis in a nutshell



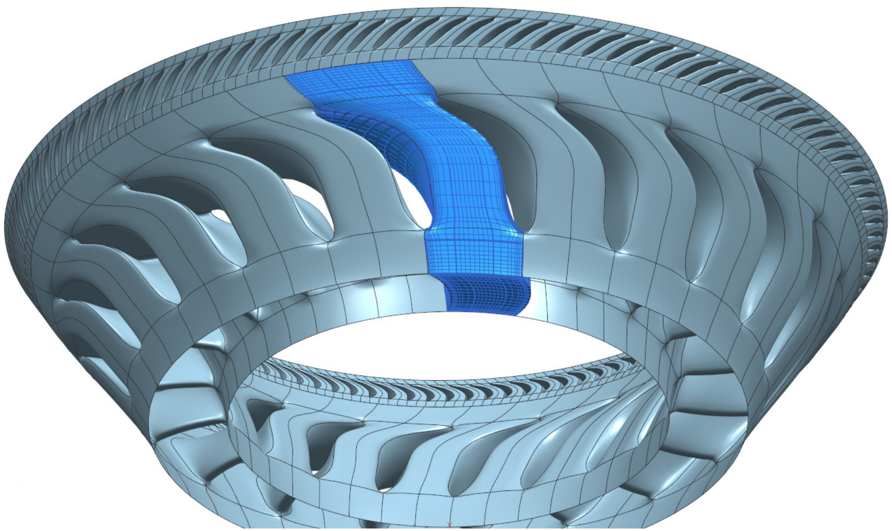
Advanced spline technologies: THB splines



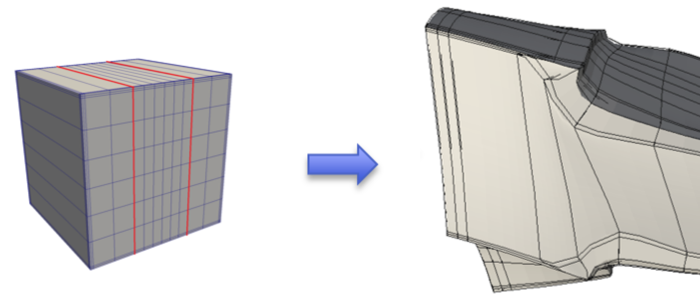


Hennig et al. Adaptive mesh refinement strategies in Isogeometric Analysis – A computational comparison, arXiv:1605.00825

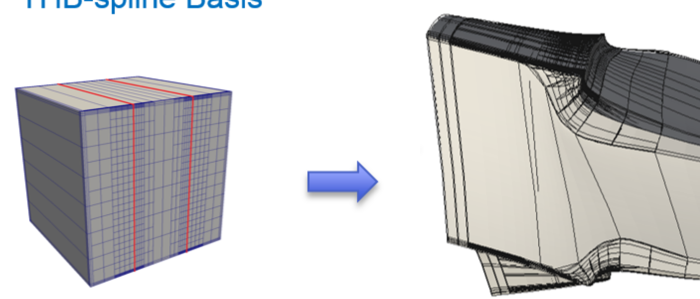
Application of THB splines



B-spline Basis



THB-spline Basis

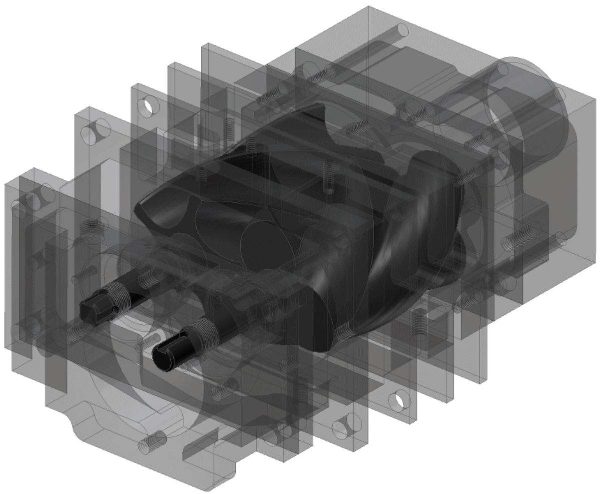


Design-Through-Analysis for

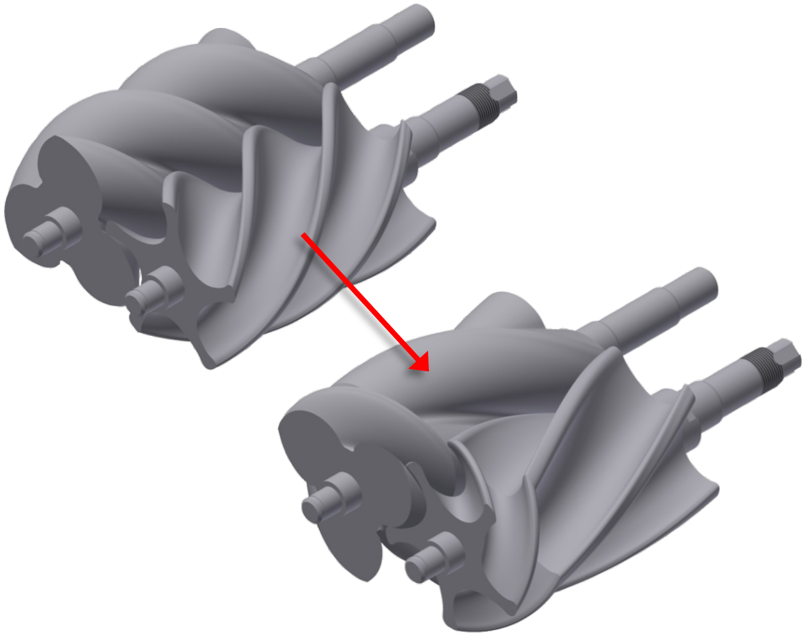
TWIN-SCREW COMPRESSORS

Long-term vision

- Develop a computer code for the efficient geometry modelling, simulation and optimization of rotary twin-screw compressors and expanders



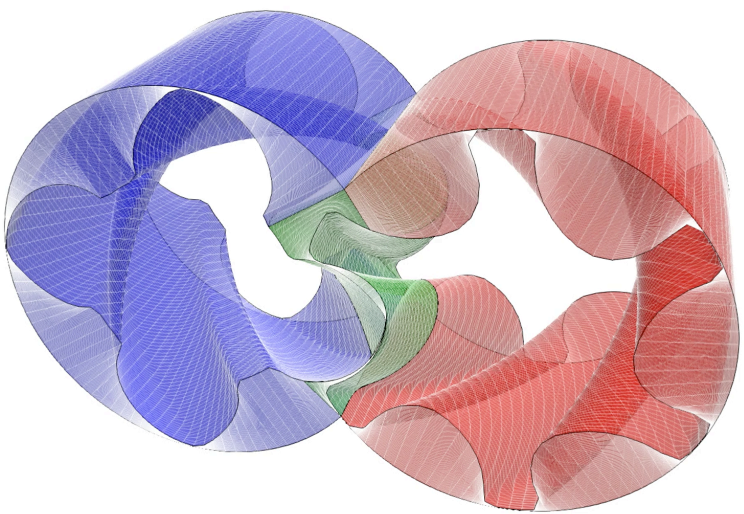
Source: Chair of Fluidics, TU Dortmund University, DE

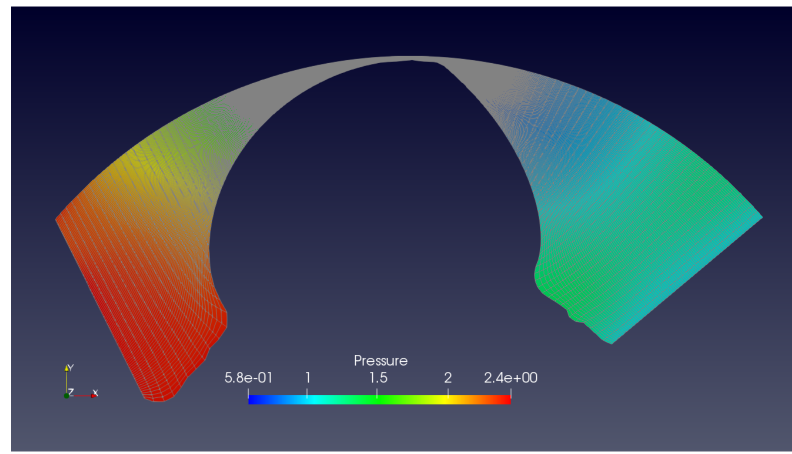


A challenging industrial application

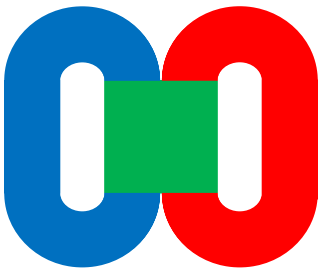
Geometry modelling
- Counter-rotating helical rotors
- Narrow clearances (<0.4mm)
- Complex deforming fluid domain

Multi-physics simulation
- Compressible high-speed flow
- Thermal expansion of solids
- Extension: injection of oil, …

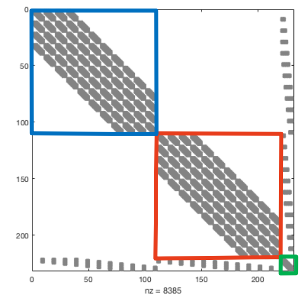




Co-design of geometry model and simulation pipeline

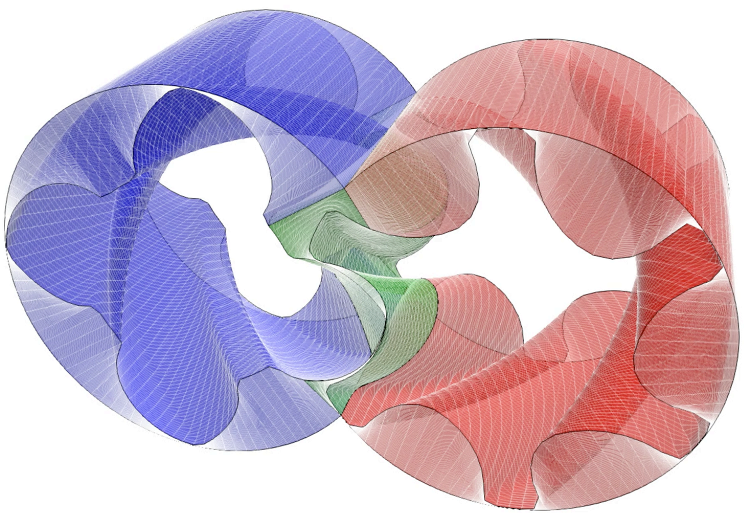


Multi-patch topology



Multi-block matrix

Multi-device computer



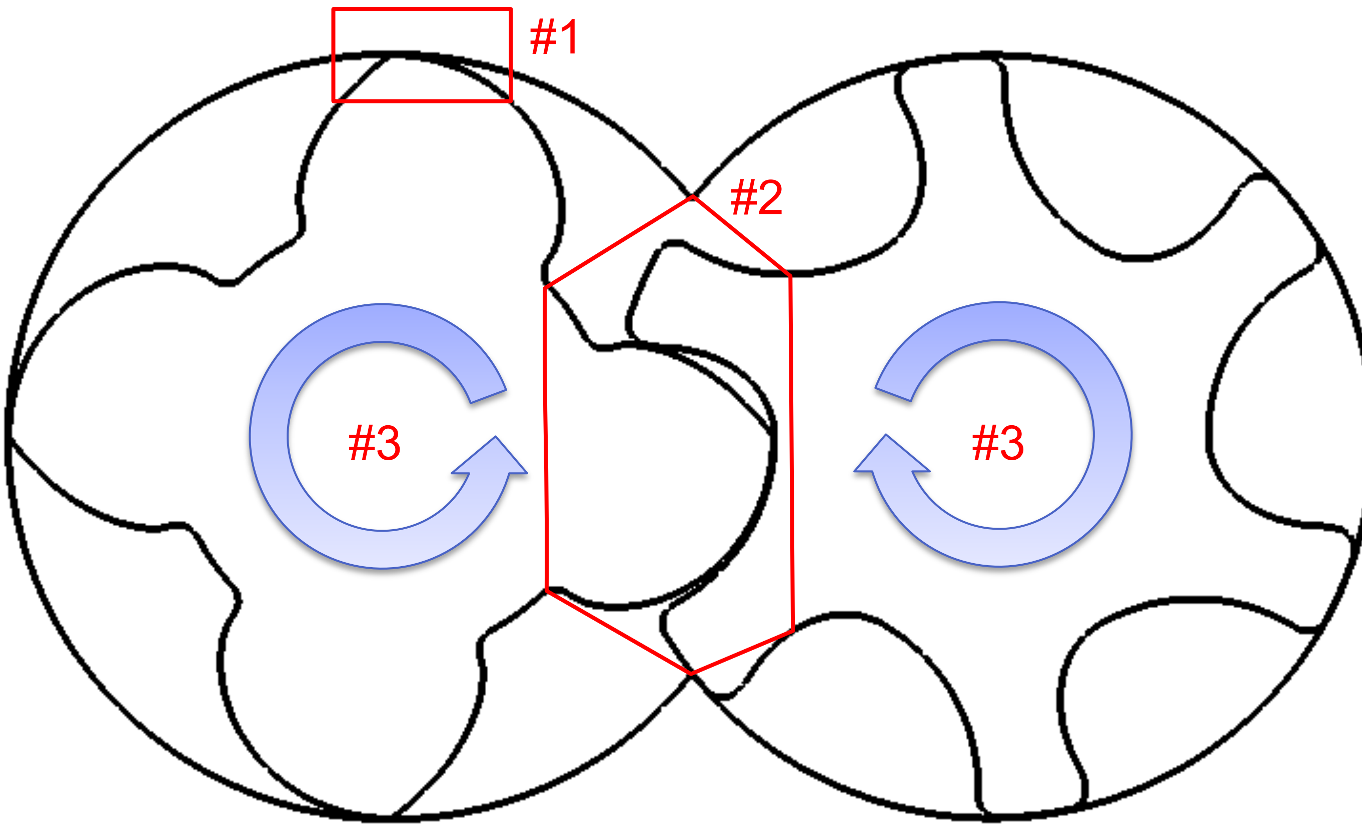
Co-design principles

- No topology changes (casing-to-rotor)
- Exploit block structure of matrices
- Keep design simple and extendible
- Support heterogeneous hardware

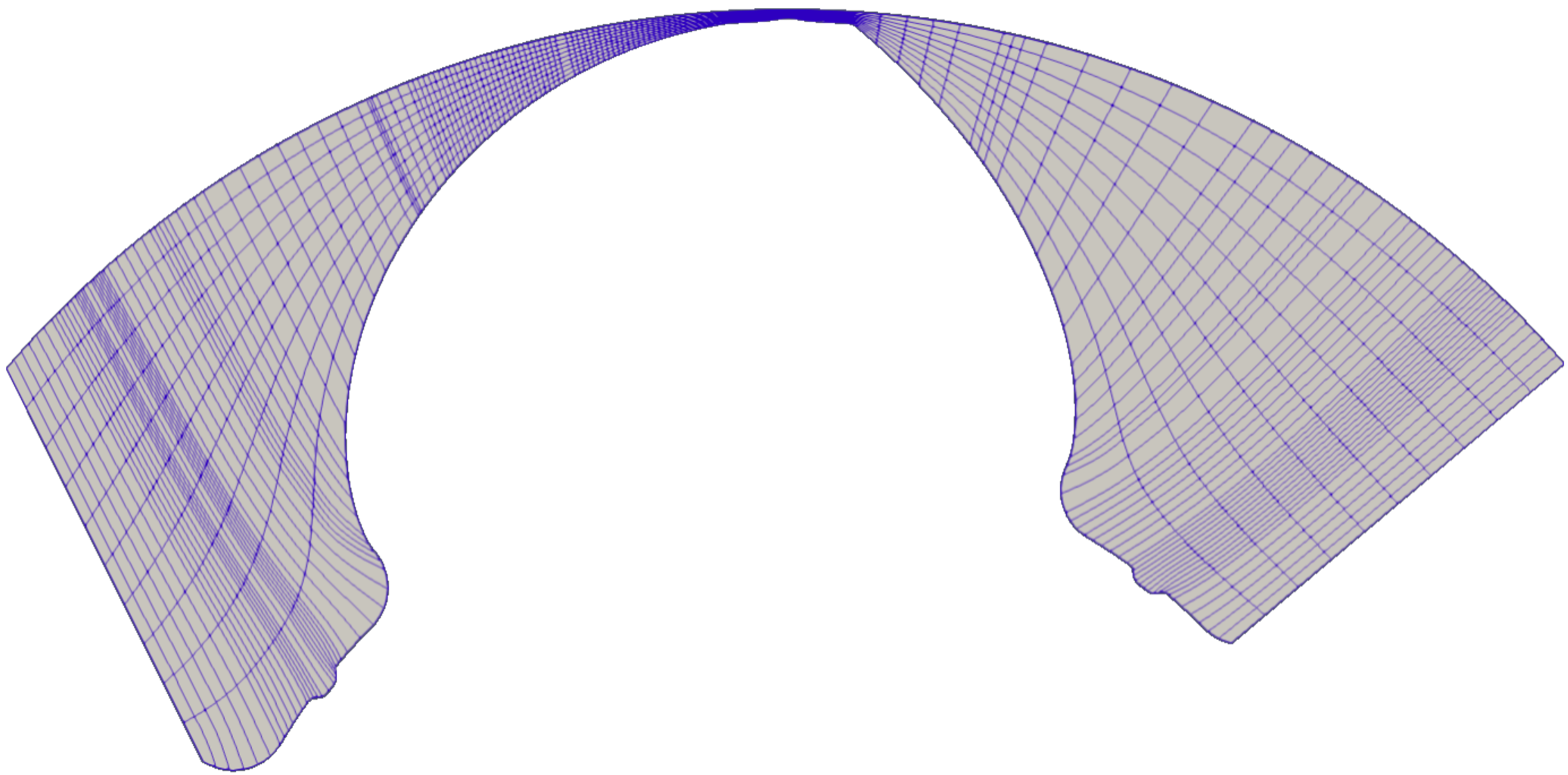
Design-Through-Analysis for twin-screw compressors

GEOMETRY MODELLING

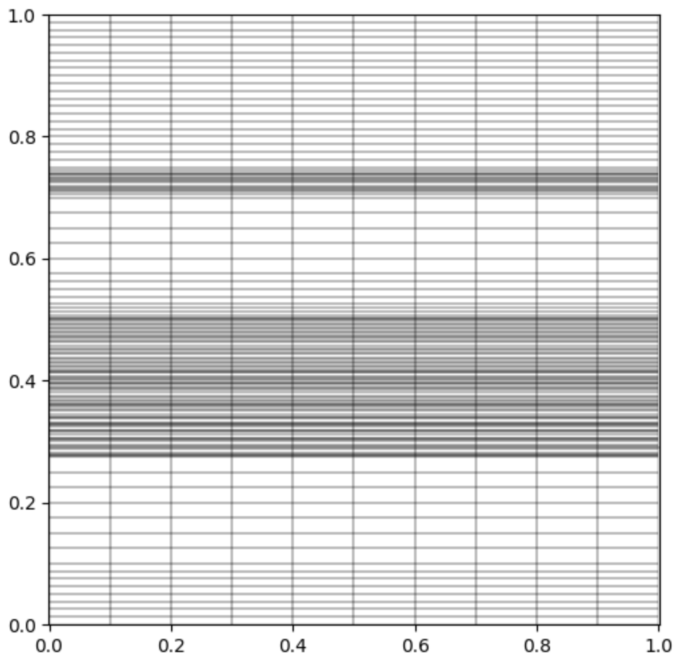
Test cases

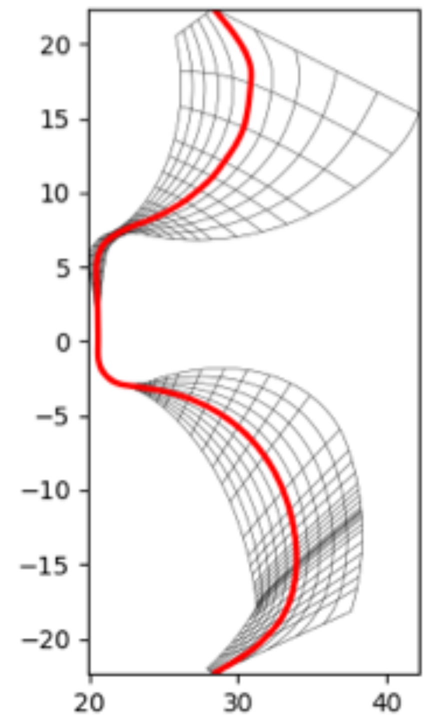


Test case #1: rotor-casing passage

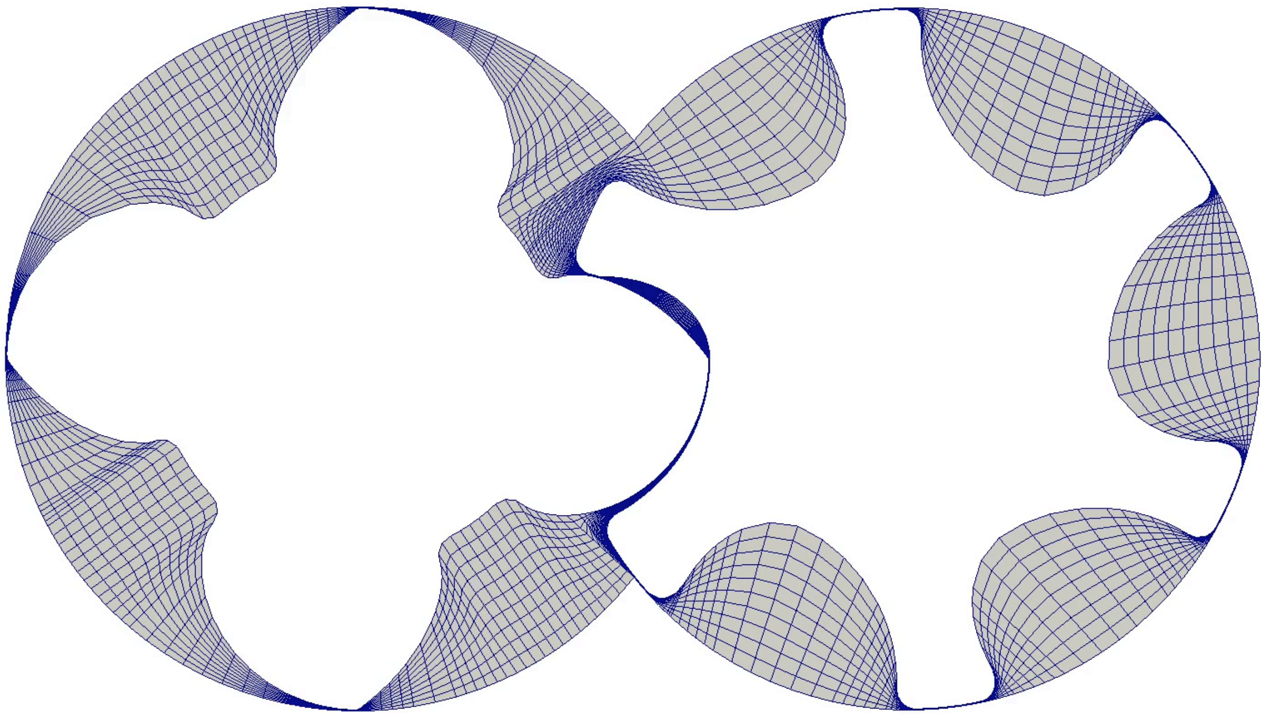


Test case #2: separator with CUSP points

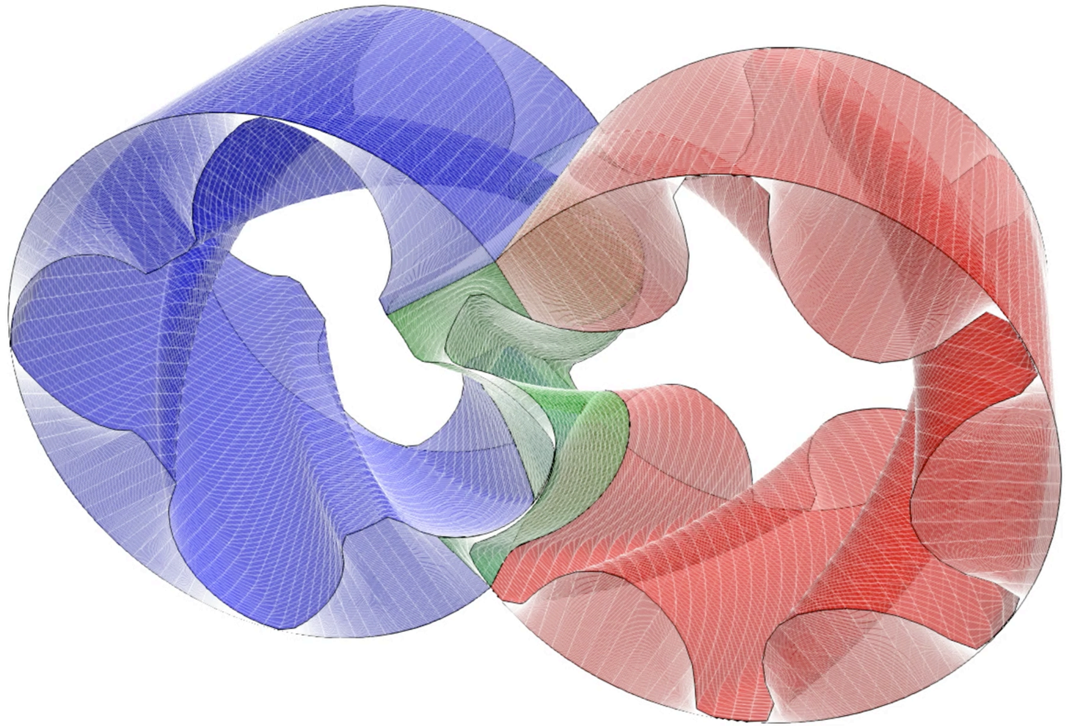




Test case #3: rotating twin rotors



Test case #4: rotating twin screws



Design-Through-Analysis for twin-screw compressors

ISOGEOMETRIC FLOW SOLVER

Compressible Euler equations

$$\frac{\partial}{\partial t}\underbrace{\begin{bmatrix}\rho\\ \rho\boldsymbol{u}\\ \rho E\end{bmatrix}}_{U=\begin{bmatrix}u_1\\ \vdots\\ u_{d+2}\end{bmatrix}} + \nabla\cdot\underbrace{\begin{bmatrix}\rho\boldsymbol{u}\\ \rho\boldsymbol{u}\otimes\boldsymbol{u}+p\mathcal{I}\\ (\rho E+p)\boldsymbol{u}\end{bmatrix}}_{\boldsymbol{F}(U)=\begin{bmatrix}f_1^1 & \cdots & f_1^d\\ \vdots & \ddots & \vdots\\ f_{d+2}^1 & \cdots & f_{d+2}^d\end{bmatrix}} = 0$$

- Equation of state for an ideal gas (isentropic index $\gamma = 1.4$ for dry air)

$$p(\rho, e) = (\gamma - 1)\rho e, \qquad \rho e = (\rho E - 0.5\rho\|\boldsymbol{u}\|^2)$$

- Flux-Jacobian matrix $\boldsymbol{A}(U) = \frac{\partial \boldsymbol{F}(U)}{\partial U}$, homogeneity property $\boldsymbol{F}(\lambda U) = \lambda \boldsymbol{F}(U)$

Variational formulation

- Find solution $U(\cdot, t)$ at fixed time t such that for all test functions W

$$\int_{\Omega} W \partial_t U - \nabla W \cdot \boldsymbol{F}(U) \mathrm{d}\mathbf{x} + \int_{\Gamma} W G\,(U, U^*) \mathrm{ds} = 0$$

- Boundary fluxes

$$G(U, U^*) = \begin{cases} [0, p\boldsymbol{n}, 0]^T & \text{at solid walls} \\ 0.5\big(\boldsymbol{F}(U) + \boldsymbol{F}(U^*)\big) \cdot \boldsymbol{n} & \text{otherwise} \\ -0.5\big|\boldsymbol{A}\big(\mathrm{Roe}(U, U^*)\big)\big| \cdot \boldsymbol{n} & \end{cases}$$

Discretization

- Find solution $U_h(\cdot, t)$ at fixed time t such that for all test functions W_h

$$\int_\Omega W_h \partial_t U_h - \nabla W_h \cdot \boldsymbol{F}_h(U) \mathrm{d}\mathbf{x} + \int_\Gamma W_h G_h(U, U^*) \mathrm{ds} = 0$$

- Fletcher's group approximation (CMAME '83)

$$U_h(\mathbf{x}, t) = \sum_{j=1}^{N} B_j(\mathbf{x}) U_j(t), \qquad B_j = \hat{B}_j \circ \boldsymbol{x}^{-1}$$

$$\boldsymbol{F}_h\big(U(\mathbf{x}, t)\big) = \sum_{j=1}^{N} B_j(\mathbf{x}) \boldsymbol{F}_j(t), \qquad \boldsymbol{F}_j = \boldsymbol{F}(U_j)$$

Semi-discrete formulation

$$\begin{bmatrix} M & & \\ & \ddots & \\ & & M \end{bmatrix} \begin{bmatrix} \dot{u}_1 \\ \vdots \\ \dot{u}_{d+2} \end{bmatrix} - \begin{bmatrix} f_1^1 & \cdots & f_1^d \\ \vdots & \ddots & \vdots \\ f_{d+2}^1 & \cdots & f_{d+2}^d \end{bmatrix} \begin{bmatrix} C^1 \\ \vdots \\ C^d \end{bmatrix} + \begin{bmatrix} g_1^1 & \cdots & g_1^d \\ \vdots & \ddots & \vdots \\ g_{d+2}^1 & \cdots & g_{d+2}^d \end{bmatrix} \begin{bmatrix} S^1 \\ \vdots \\ S^d \end{bmatrix} = 0$$

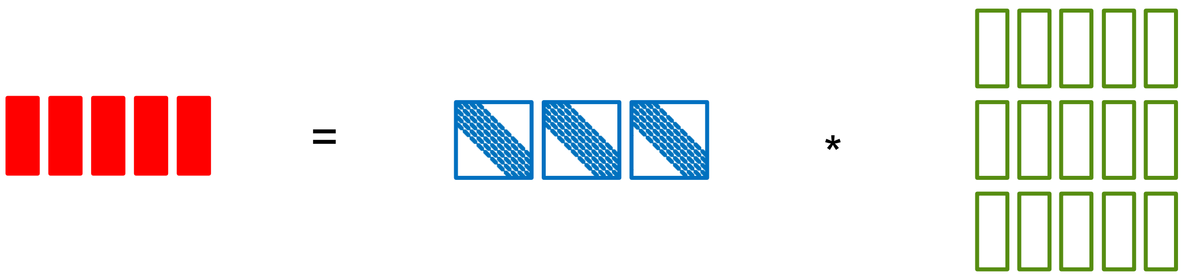
- Constant coefficient matrices

$$M = \left\{ \int_\Omega B_i B_j \mathrm{d}\mathbf{x} \right\}_{i,j=1}^N, C^k = \left\{ \int_\Omega \partial_k \left(B_i \right) B_j \mathrm{d}\mathbf{x} \right\}_{i,j=1}^N, S^k = \left\{ \int_\Omega B_i B_j \boldsymbol{n} \mathrm{ds} \right\}_{i,j=1}^N$$

are pre-assembled using Gauss quadrature and stored to efficiently form the divergence term as SpMV-operation when it is required

From the programmer's perspective

$$M[\dot{u}_1 \;\cdots\; \dot{u}_{d+2}] - [C^1 \;\cdots\; C^d]\begin{bmatrix} f_1^1 & \cdots & f_1^{d+2} \\ \vdots & \ddots & \vdots \\ f_d^1 & \cdots & f_d^{d+2} \end{bmatrix} + [S^1 \;\cdots\; S^d]\begin{bmatrix} g_1^1 & \cdots & g_1^{d+2} \\ \vdots & \ddots & \vdots \\ g_d^1 & \cdots & g_d^{d+2} \end{bmatrix} = 0$$



1 x d+2 block vector of dense vectors

1 x d block vector of sparse matrices

d x d+2 block matrix of function expressions

FDBB: Fluid dynamics building blocks

Low-level API

- Unified function wrappers to the core functionality of established HPC linear algebra libraries, e.g. ArrayFire, Blaze, Eigen, VexCL

- Compile-time block linear algebra backend with full support for
  - Dense vectors
  - Sparse matrices
  - Function expressions

Example

```
vex::vector<double>         x,y;
vex::sparse::matrix<double> Cx,Cy;

BlockMatrix<…,1,2> Mat(Cx,Cy);

auto Expr = make_BlockExpr<2,1>(cos(x)+y,
                                sin(x)-y );

BlockRowVector<…,1> Vec = Mat * Expr;
```

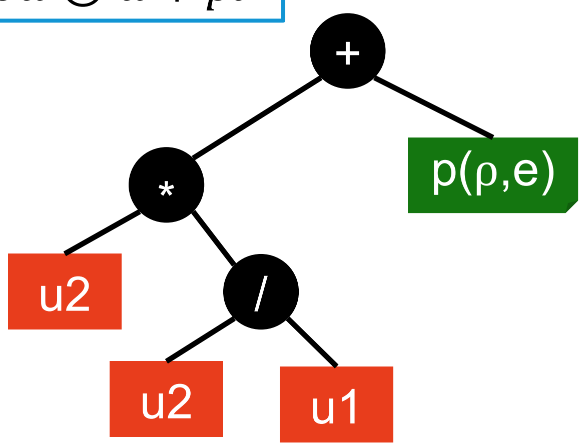
Loops are unrolled at compile time and fused in a single compute kernel

FDBB: Fluid dynamics building blocks

High-level API

- C++ expression templates for
  - Variables & Riemann invariants
  - Fluxes with ’generic’ pressure
  - Equations of state

$$\rho \boldsymbol{u} \otimes \boldsymbol{u} + p\mathcal{I}$$

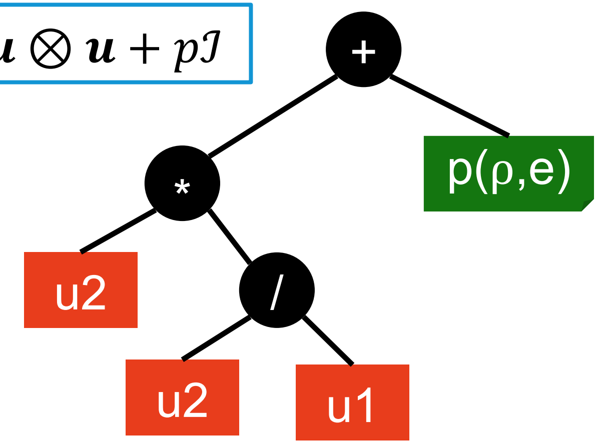


Example

```
using eos = EOS::idealGas<double,
                          ratio<7,2>,
                          ratio<5,2>>;

// 1x4 dim conservative state vector
using var = Variables<eos,2,Conservative>;
auto U = create<vex::vector<double>,4>(N);

// 2x4 dim inviscid flux tensor
using flx = Fluxes<var>;
auto F = flx::inviscid(U);

// Explicit solution update
U += dt * Mat * F
```

Auto-generation of device-optimized CFD kernels

$$\rho \boldsymbol{u} \otimes \boldsymbol{u} + p\mathcal{I}$$



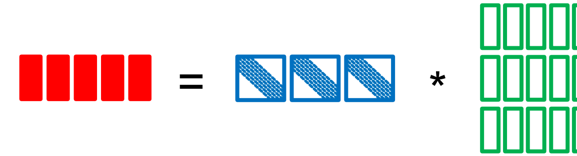

```
double rhs_6_sum = 0;
{
  for(size_t j = 0; j < rhs_6_ell_width; ++j)
  {
    int nnz_idx = idx + j * rhs_6_ell_pitch;
    int c = rhs_6_ell_col[nnz_idx];
    if (c != (int)(-1))
    {
      int idx = c;
      rhs_6_sum += rhs_6_ell_val[nnz_idx] * ( ( ( prm_tag_2_1[idx] * prm_tag_2_1[idx] ) / 
prm_tag_0_1[idx] ) + ( rhs_6_x_4 * ( prm_tag_0_1[idx] * ( ( prm_tag_3_1[idx] / prm_tag_0_1[idx] ) - ( 
( 5.000000000000000e-01 ) * ( ( ( prm_tag_1_1[idx] * prm_tag_1_1[idx] ) + ( prm_tag_2_1[idx] * 
prm_tag_2_1[idx] ) ) / ( prm_tag_0_1[idx] * prm_tag_0_1[idx] ) ) ) ) ) ) );
    } else break;
  }
  if (rhs_6_csr_ptr)
  …
```

Heterogeneous HPC support

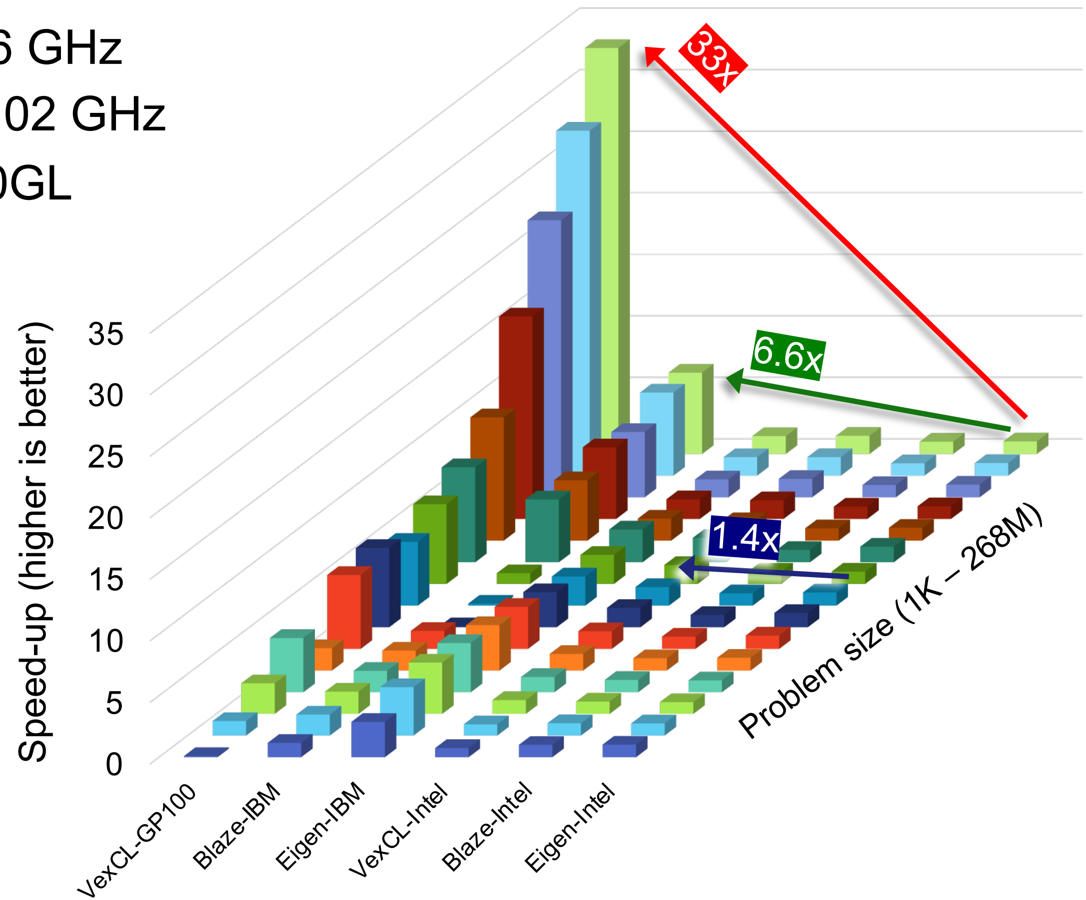
	CPU	CUDA	OpenCL	Intel MIC	FPGA
ArrayFire	X	X	X		
Armadillo	(X)				
Blaze	X				
Eigen	X				
MTL4	(X)				
uBLAS	(X)				
VexCL[1]	X	X	X	(X)	(X[2])
ViennaCL	X	X	X		

[1] source code is generated and JIT-compiled; [2] Maxeler DFEs

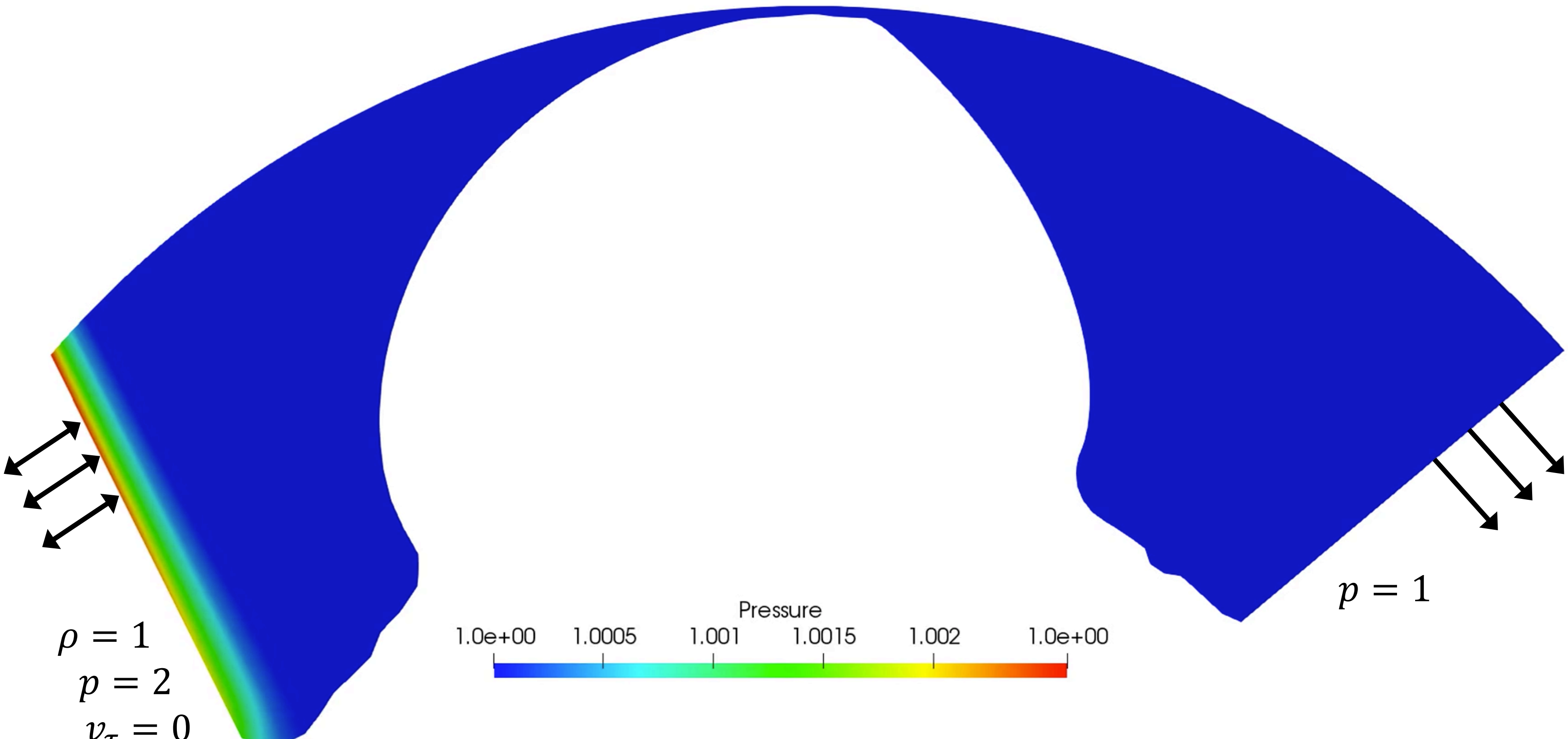
Example: Computational performance



- Intel 2xE5-2670 @2.6 GHz
- IBM Power8NVL @4.02 GHz
- IBM + NVIDIA GP100GL



Test case #1: Inviscid compressible flow, $p_{in}: p_{out} = 2:1$



Conclusions

Hardware-oriented Numerics with IGA: co-design of geometry and simulation

- IGA package G+Smo: *https://github.com/gismo/gismo*
- Open-source FDBB: *https://gitlab.com/mmoelle1/FDBB*

Ongoing and future work

- Extension to turbulent flows and ALE formulation for rotating geometries
- Automatic compute resource scheduling and dynamic load balancing

References

- MM, A. Jaeschke: High-order IGA for compressible flows I+II, arXiv: 1809.10893 and 1809.10896
- MM, A. Jaeschke: FDBB: Fluid Dynamics Building Blocks, arXiv: 1809.09851

Financial support by EC under GA No 672787 is acknowledged

MOTOR
Multi-ObjecTive design Optimization of fluid eneRgy machines

Recent updates

- **MPI-communication and dynamic load-balancing tool framework** : *joint student or master project (UvA, TUD); topic 2*
  - Cluster inventory tool implemented with support for x86, x86_64, ppc64le, ARM, CUDA, OpenCL, SDAccel, …
- **Sparse band matrix implementation** : *student project TUD*
- **DSL-based algorithmic differentiation** : *PhD project by A. Jaeschke*
  - Gradient-based shape optimization in turbomachinery applications
- **PyG+Smo** : *hobby project of mine*
  - Python wrappers of G+Smo library based on PyBind11