Parallel in Time simulation of the Navier-Stokes equations using the Finite Element Method

Dipl.-Ing. Konstantinos Ioakimidis

ioakimidis@ihs.uni-stuttgart.de
konstantinos.ioakimidis@gmail.com
Skype: konstantin.ioakimidis

Universität Stuttgart
www.ihs.uni-stuttgart.de

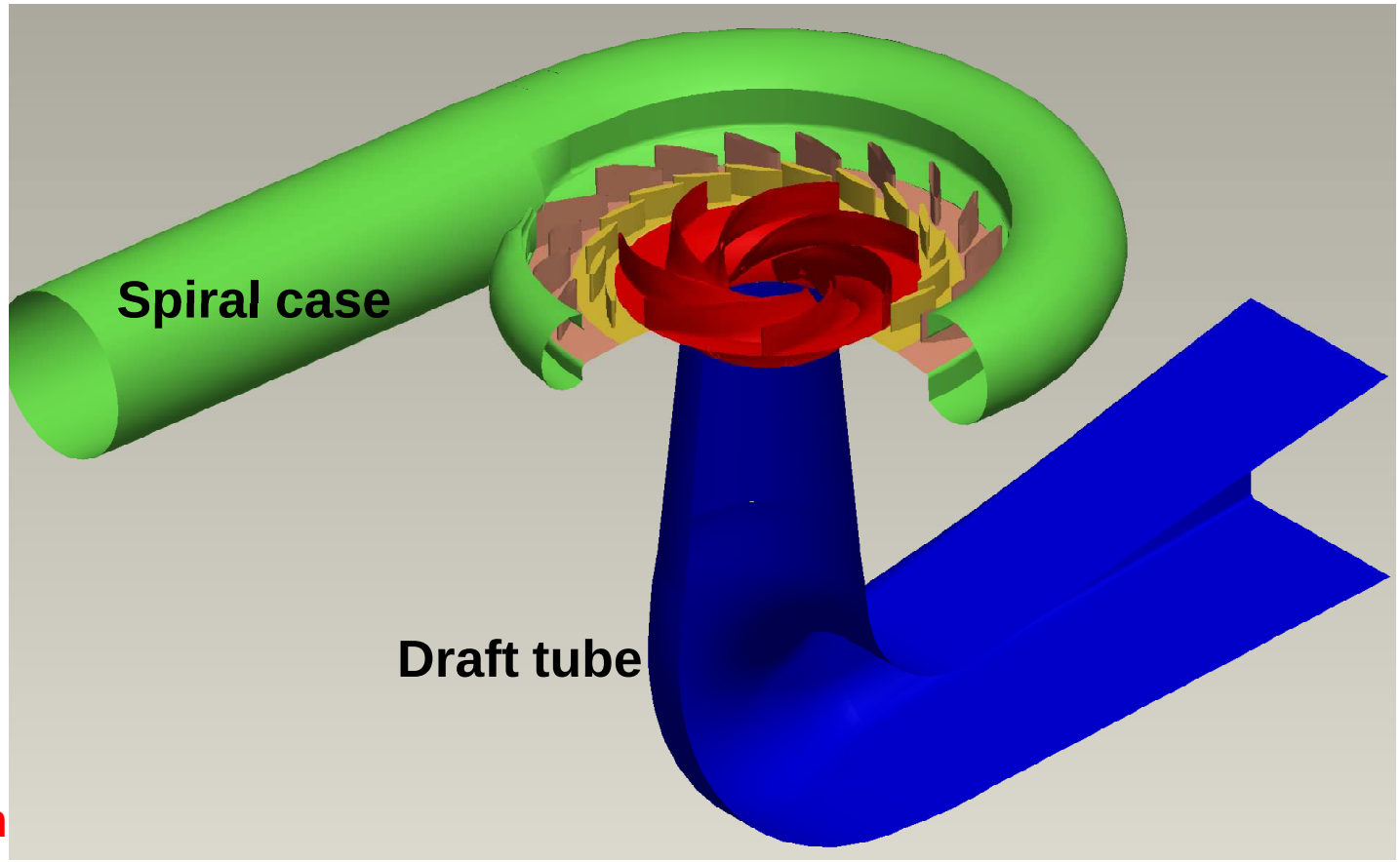
Outline

- Geometry
  problem cases
- Unsteadiness in hydraulic machinery (HM)
  phenomena
  FENFLOSS and some results
- The actual problem
- FENFLOSS
  General information
  Code structure
- New code & algorithm
- Parallel in Time (not working yet)
- Future Work

Geometry

- Problem size
  more than100 million nodes
  up to 600 million of unknowns for single phase flow

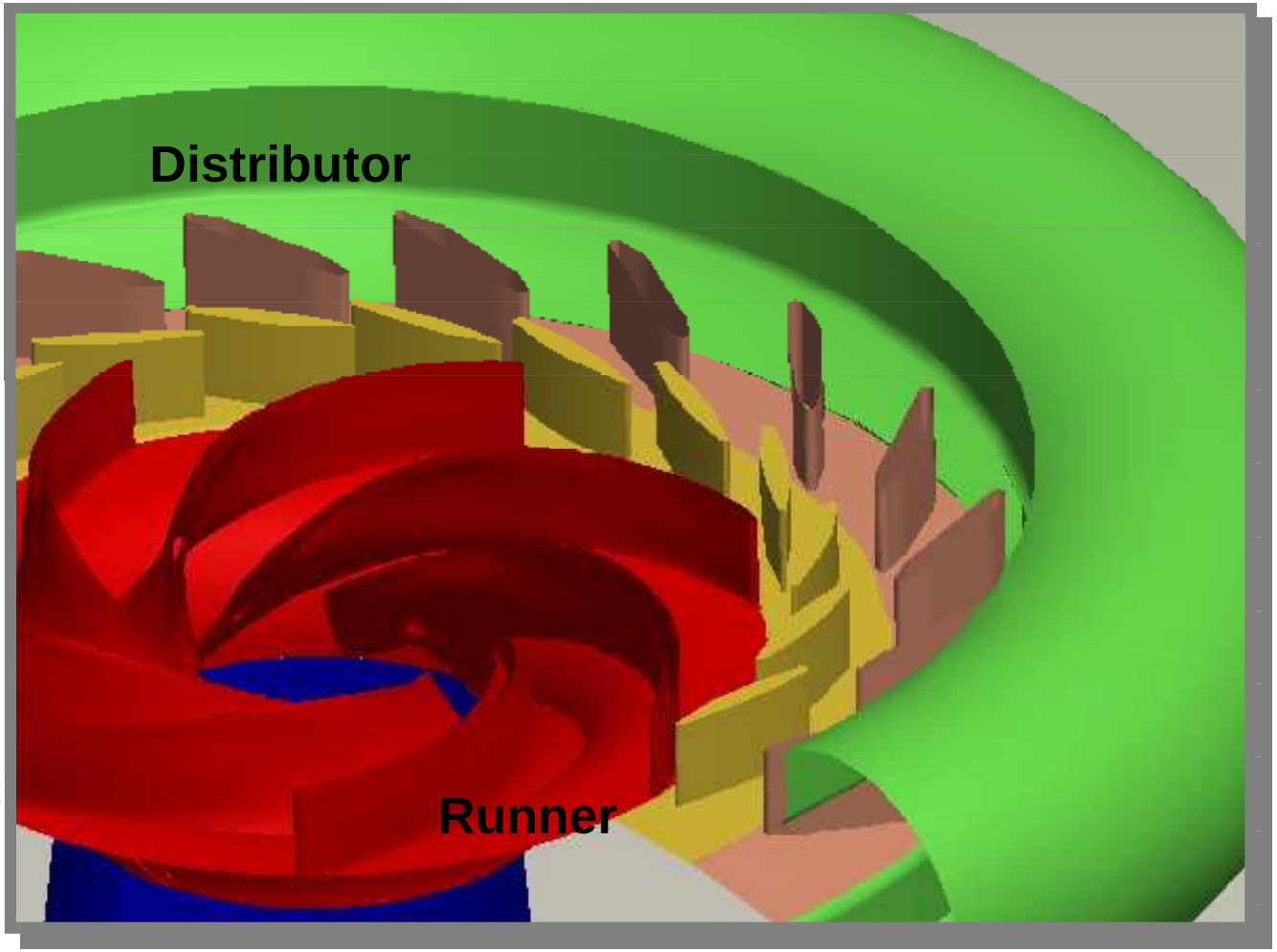
- **pipe,**
- **diffusor**



Wall friction
Swirling flow
Flow separation

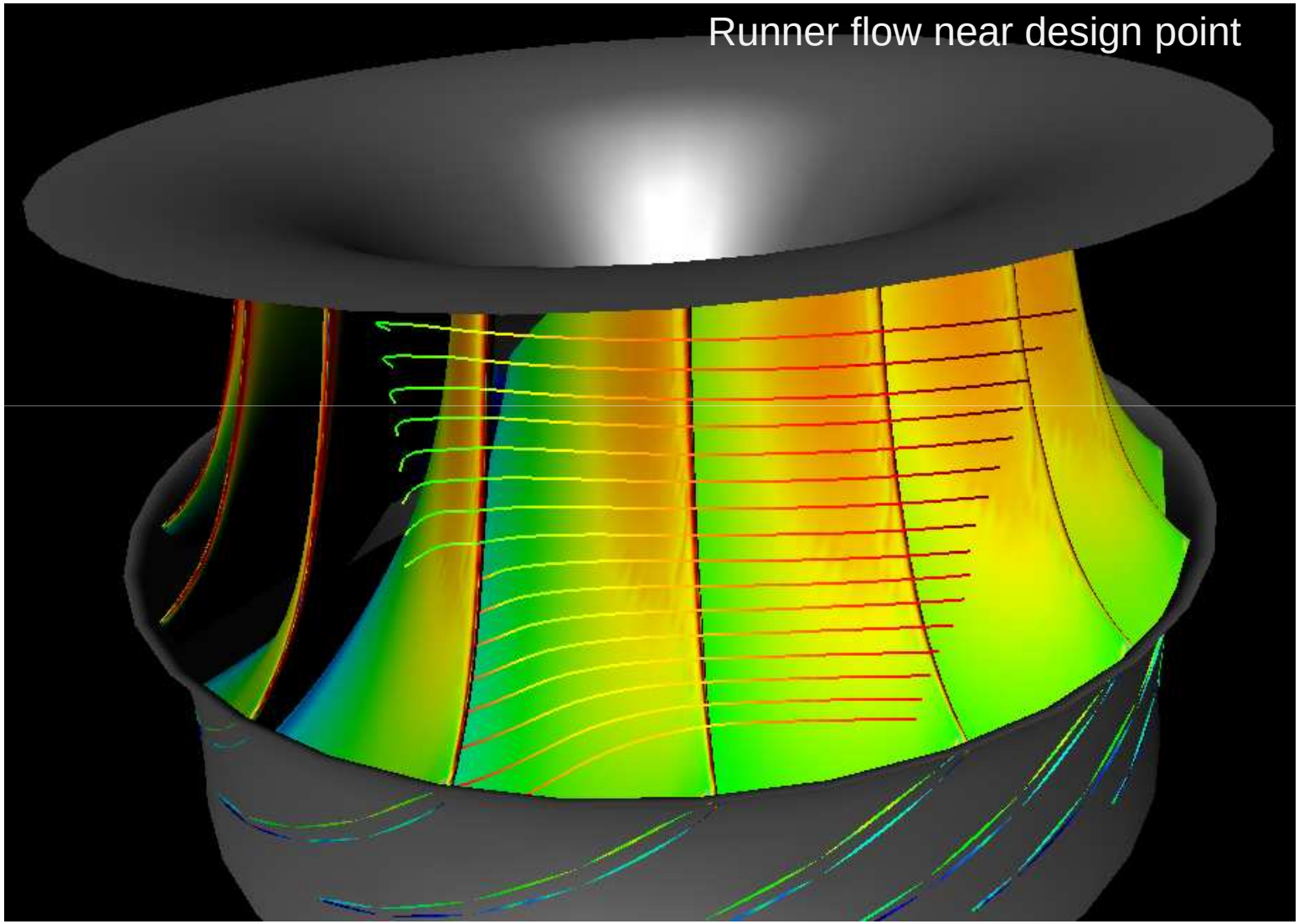
Geometry zoom in

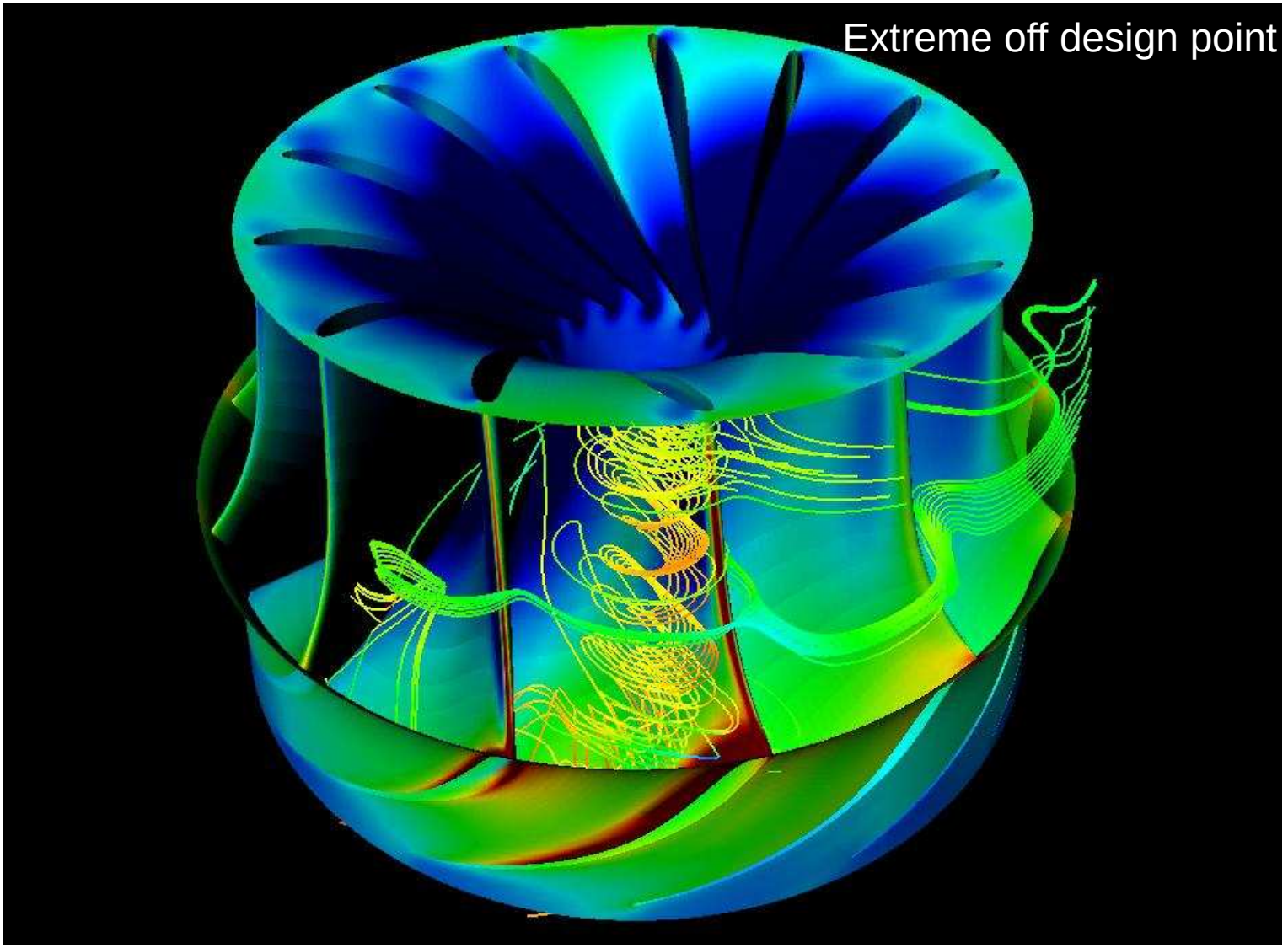
- **Hydrofoil**
- **Rotating Runner**
- **Gap Flow**



Flow Curvature
Flow Separation
Turbulence Transport
Wake Flow
Vortex Flow

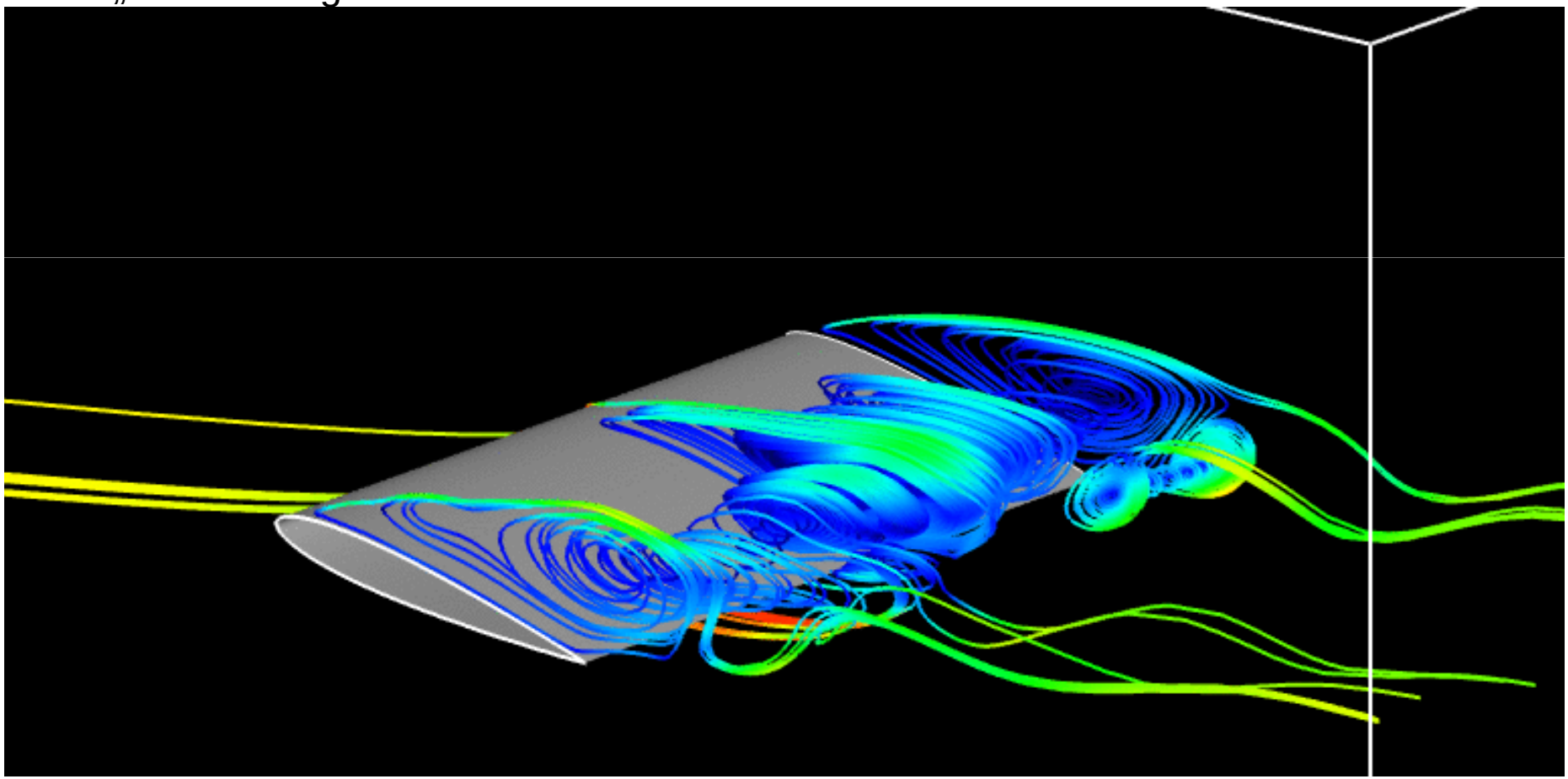
- Unsteady phenomena in hydraulic machinery
  - **Externally forced unsteadiness**
    „forced“ changing geometry
    e. g. Rotor-Stator Interaction, valve closure
  - **Free unsteadiness**
    unsteady vortex movement, flow instabilities
    e. g. draft tube vortex rope, vortex shedding
    e. g. Karman vortex street, flow induced vibrations
    changing geometry caused by unsteady flow.


Runner flow near design point

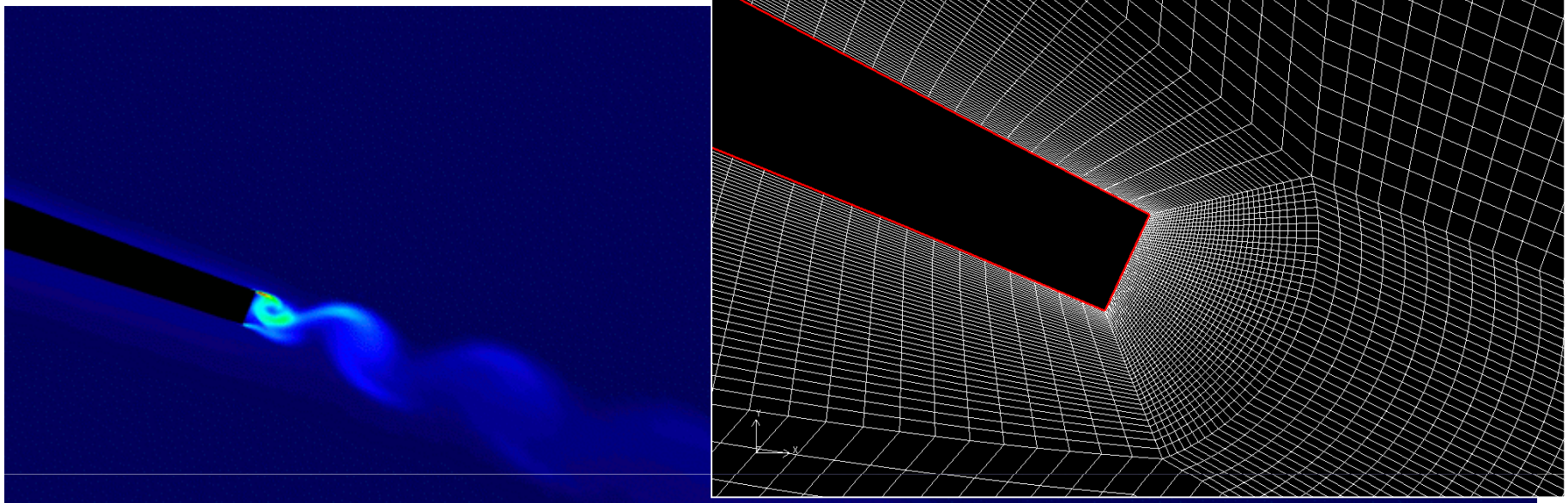

Extreme off design point

Free unsteadiness

Too high turbulence viscosity will result in steady state recirculation
No „bad“ averaged results



Unsteadiness in hydraulic machinery

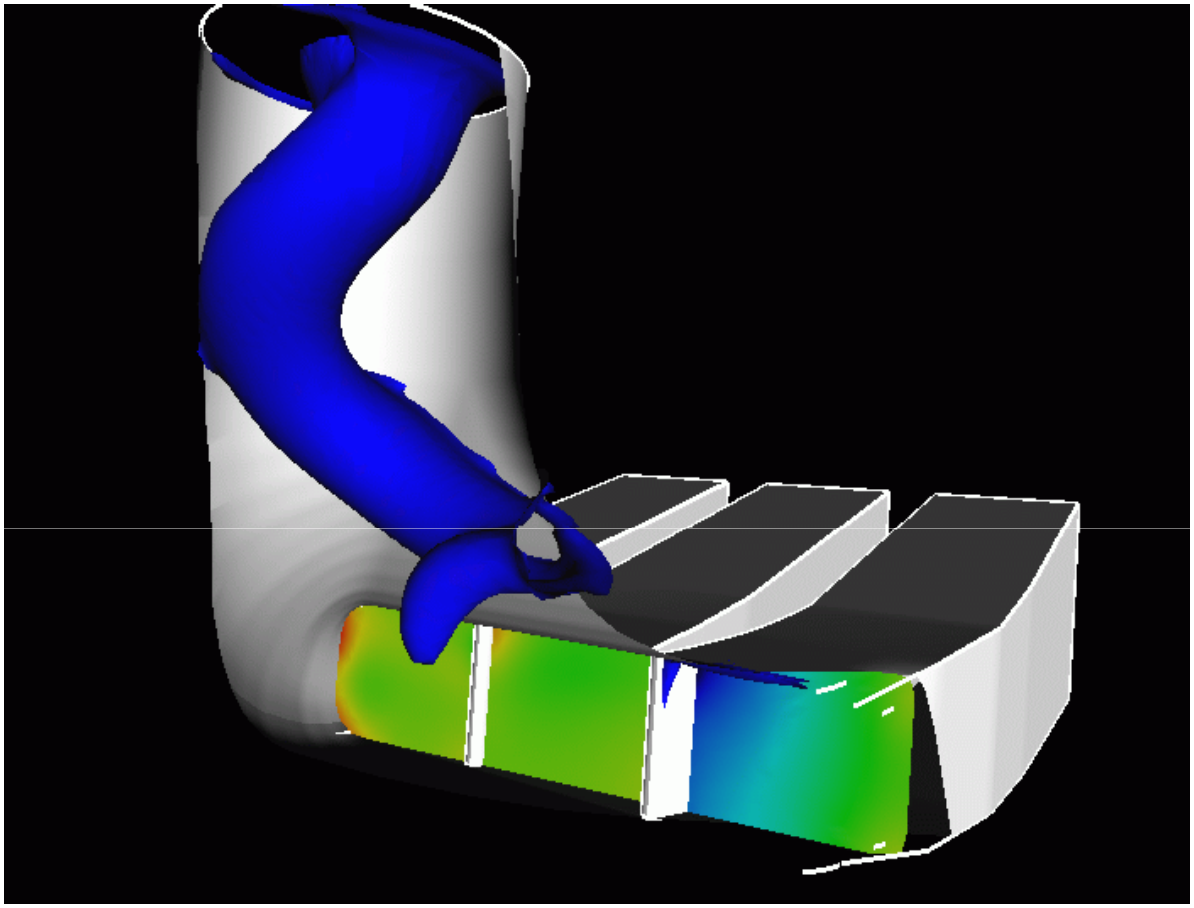


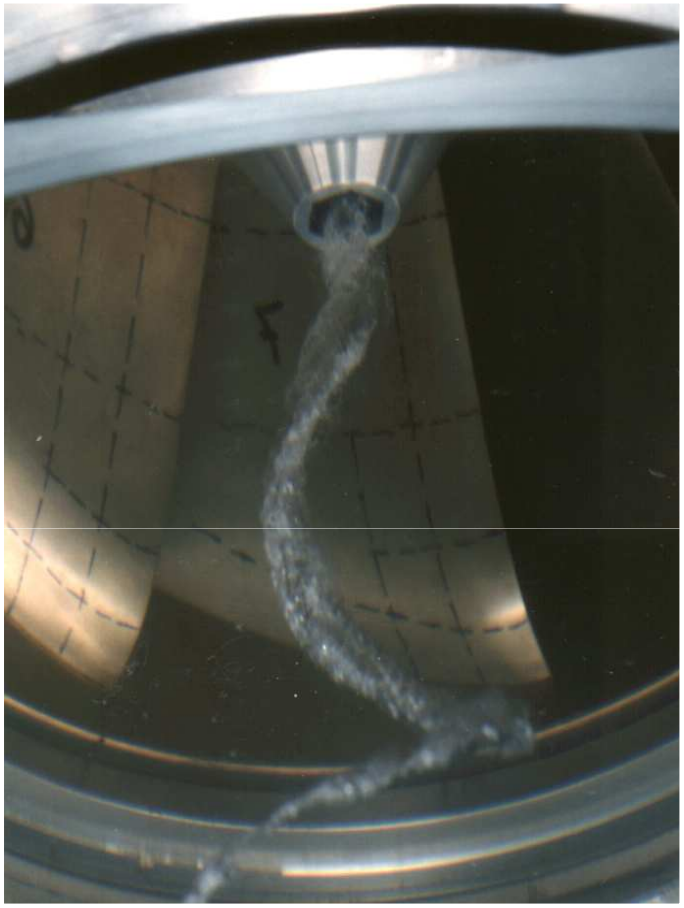
Vortex shedding frequency corresponds with estimation

However very fine grids required
(~ 8 mio nodes)

Too high turbulent viscosity will suppress all unsteadiness => completely wrong results

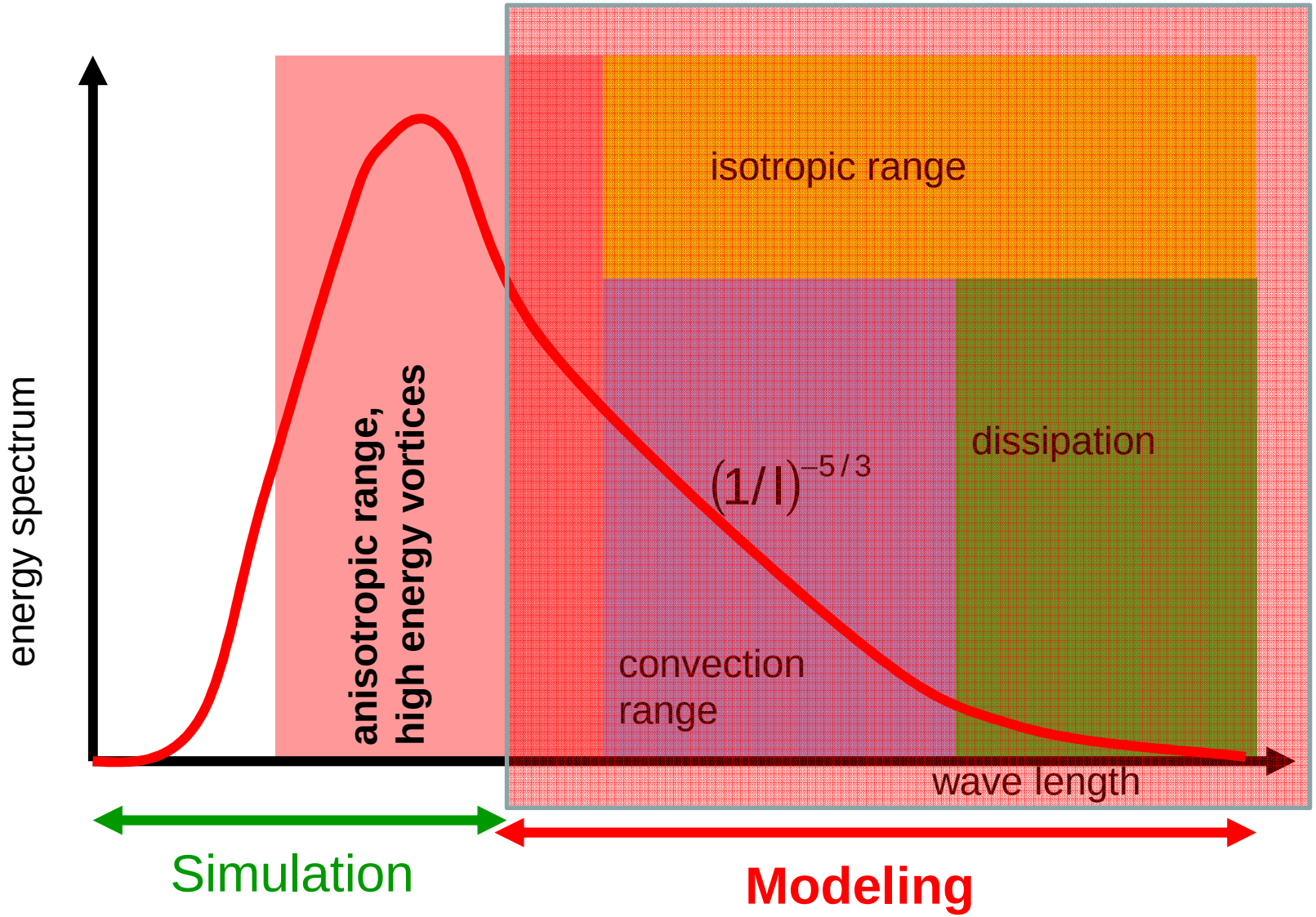
Unsteadiness in hydraulic machinery



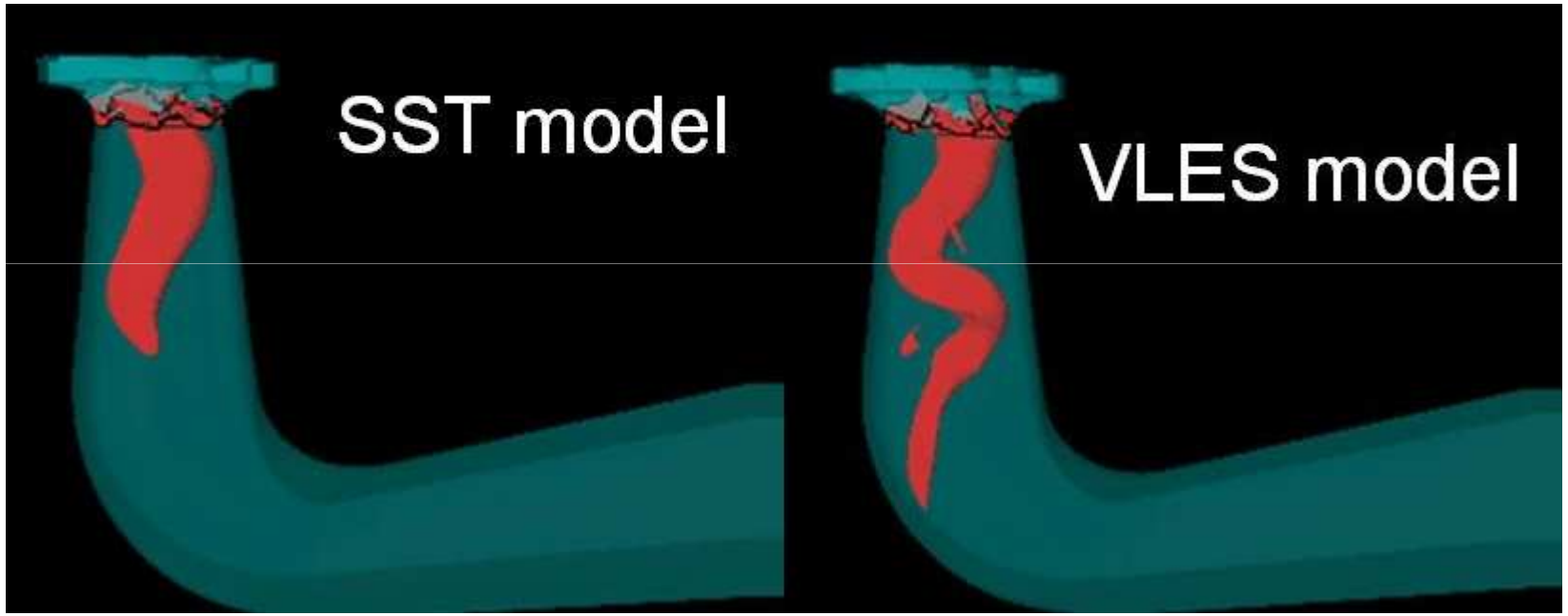


Unstable vortex movement, vortex instability, breakdown
strong swirling flow, strongly anisotropic turbulence behavior
Sophisticated turbulence modeling

Very Large Eddy Simulation



Comparison Turbulence Models



Picture: Huiming Wang, Institute of Fluid Mechanics and Hydraulic Machinery (IHS), Universität Stuttgart

- Developed at IHS since early 80s
  Originaly designed by Dr. A. Ruprecht still active at IHS
  Parallelised and Vectorised by Dr. M. Maihofer
- Incompressible Navier Stokes (RANS) Equations
  Steady and Unsteady Flows
  Laminar as well as Turbulent Flows
  Turbulence Models already implemented
  Simulates any kind of Incompressible flow, especially in Hydraulic Machinery
- Finite Element Method on Unstructured Meshes
  a Q1P0 Petrov-Galerkin formulation on hexahedral elements is applied
  pressure correction scheme
- Can handle Static and Moving Meshes
  Efficient algorithm implemented for coupling Static & Moving Parts
  e.g. Rotor-Stator coupling
- Can be coupled with structural mechanics code to Simulate Fluid Structure Interaction (FSI)
  FSI as well as API interface implemented by Dr. F. Lippold

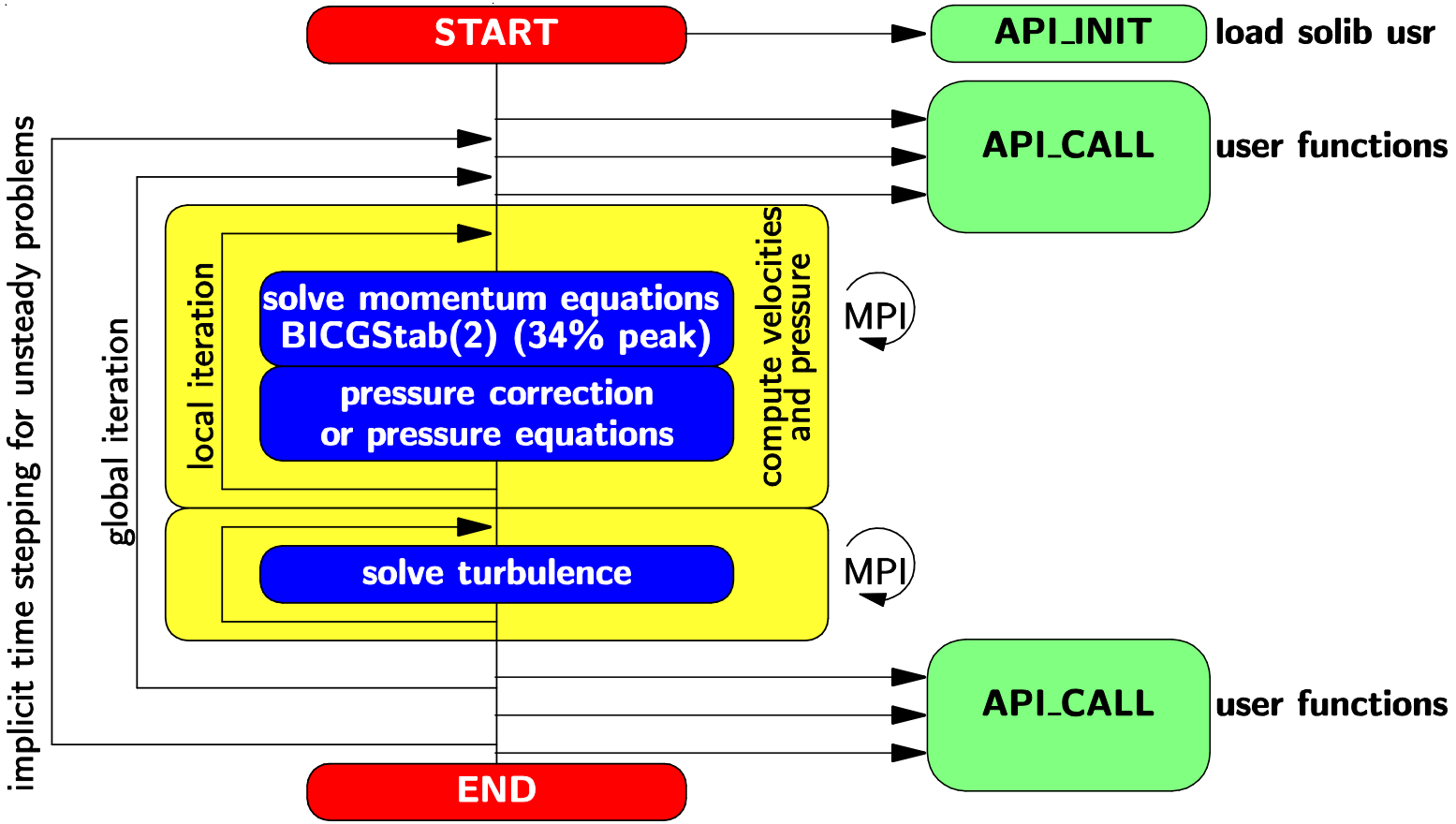
- Dynamically Load Libraries – API Interface
  Enables communication with visualisation tools e.g. COVISE (HLRS)
  Exchanging data during the simulation run, in order to automatic visualisation of the time-step results.
  Plenty of Turbomachinery designs made in IHS.
- The Code is not Publicly available

Code Structure

FENFLOSS basic building blocks are:

- Read Input (already distributed) files
  Distributed Memory Approach (DMP) based on MPI
  Each process reads only its own data from those files
  Domain Decomposition based on the METIS-library (guarantee good balancing)
  Advantage -> independence of solution from the number of partitions (speed-up)
- Initialise Data
- Timestep Loop for Unsteady Flows
- Equations System Solving Iterations (Linearisation)
  Navier – Stokes Equations are non-linear
  A Richardson Iteration is used to solve the the global non-linear problem
- Assembly of the Global FEM Matrix
  Time expensive step
- Solving Linear Equation System
  The linear system is solved by the solver based on Krylov Method
  (e.g. Van der Vorst‘s BiCGStab(2))
  The parallelised & vectorised linear solver handles computations with some millions nodes.

FENFLOSS General Flowdiagram



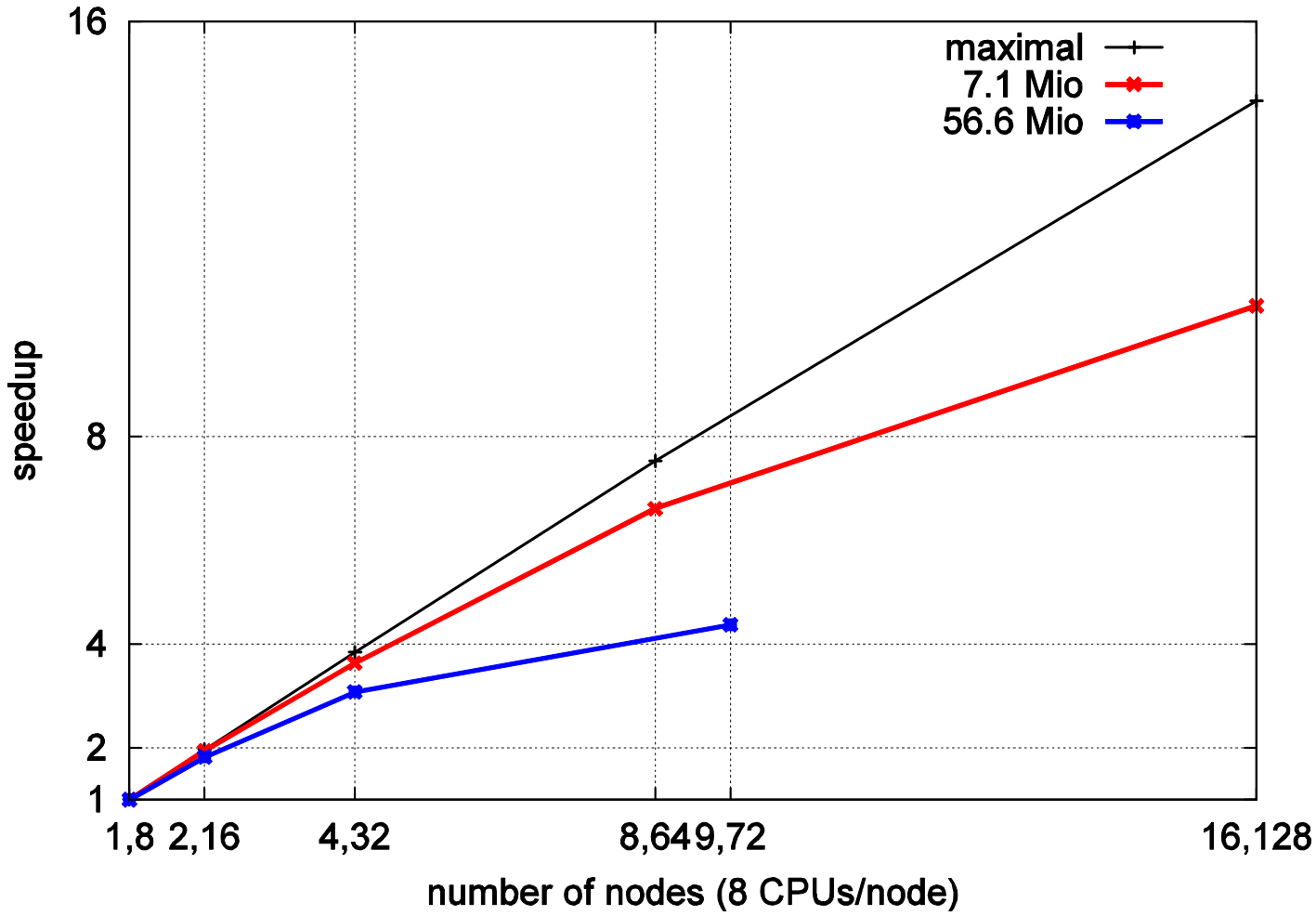
Speedup I

- Performance on the NEC SX-8
  special storage scheme (JAD)
  optimised loops
  -> very efficient on high performance vector computers
- Cooperation with NEC-HLRS TERAFLOP Project
  vector performance of almost 35% of the peak performance
  i.e. 16 GFlop/s per vector CPU

Large Nr. of Processors -> higher communication effort
decreasing vector length

…lower performance

Speedup II


16
8
4
2
1
speedup
maximal
7.1 Mio
56.6 Mio
1,8 2,16
4,32
8,64 9,72
16,128
number of nodes (8 CPUs/node)

New code & algorithm

We need

- To deal with spatial oscillations produced by the discretization of the convection transport terms
- To simplify momentum equation (no pressure gradients)
  - advection-convection problem
- Pressure stability (to use arbitrary interpolation functions) to satisfy the well-known Babuska-Brezzi condition
- Satisfy the continuity
- Implementation simplicity
  useful for testing algorithms, turbulence models ….
- „Easy“ parallelizable in Space and Time

that is the Characteristic Galerkin Method of Characteristic Based Split

(Greek: ΄Υδραμα) => rich in water

call preprocessing

DO timestepping

call calculate_timestep/local_timestep

call momentum_for_velocity

call poison_for_pressure

call momentum_for_velocity_correction

END DO

call postprocessing

(Greek: ΄Υδραμα) => rich in water

- Continuity

$$\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} = 0$$

- x Momentum

$$\frac{\partial u_1}{\partial t} + u_1 \frac{\partial u_1}{\partial x_1} + u_2 \frac{\partial u_1}{\partial x_2} = -\frac{1}{\rho}\frac{\partial p}{\partial x_1} + \nu\left(\frac{\partial^2 u_1}{\partial x_1^2} + \frac{\partial^2 u_1}{\partial x_2^2}\right)$$

- y Momentum

$$\frac{\partial u_2}{\partial t} + u_1 \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_2}{\partial x_2} = -\frac{1}{\rho}\frac{\partial p}{\partial x_2} + \nu\left(\frac{\partial^2 u_2}{\partial x_1^2} + \frac{\partial^2 u_2}{\partial x_2^2}\right)$$

equations of the previous section. If the CG procedure is applied to the above equations, a semi-discrete from of the equations is obtained, namely,

intermediate x_1 momentum equation

$$\frac{\tilde{u}_1 - {u_1}^n}{\Delta t} = -u_1 \frac{\partial {u_1}^n}{\partial x_1} - u_2 \frac{\partial {u_1}^n}{\partial x_2} + \nu \left(\frac{\partial^2 u_1}{\partial x_1^2} + \frac{\partial^2 u_1}{\partial x_2^2} \right)^n$$

$$+ u_1 \frac{\Delta t}{2} \frac{\partial}{\partial x_1} \left[u_1 \frac{\partial {u_1}^n}{\partial x_1} + u_2 \frac{\partial {u_1}^n}{\partial x_2} \right]$$

$$+ u_2 \frac{\Delta t}{2} \frac{\partial}{\partial x_2} \left[u_1 \frac{\partial {u_1}^n}{\partial x_1} + u_2 \frac{\partial {u_1}^n}{\partial x_2} \right] \qquad (7.129)$$

Roland W. Lewis, Perumal Nithiarasu, Kankanhalli N. Seetharamu, *Fundamentals of the Finite Element Method for Heat and Fluid Flow,* John Wiley & Sons Ltd, 2004

that is, (neglecting third-order terms)

$$\frac{\partial u_1^{n+1}}{\partial x_1} + \frac{\partial u_2^{n+1}}{\partial x_2} - \frac{\partial \tilde{u}_1}{\partial x_1} - \frac{\partial \tilde{u}_2}{\partial x_2} = -\frac{\Delta t}{\rho}\left(\frac{\partial^2 p}{\partial x_1^2} + \frac{\partial^2 p}{\partial x_2^2}\right)^n \tag{7.134}$$

Note that from the continuity equation

$$\frac{\partial u_1^{n+1}}{\partial x_1} + \frac{\partial u_2^{n+1}}{\partial x_2} = 0 \tag{7.135}$$

On substituting the above equation into Equation 7.134, we obtain the pressure equation as follows:

$$\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x_1^2} + \frac{\partial^2 p}{\partial x_2^2}\right)^n = \frac{1}{\Delta t}\left(\frac{\partial \tilde{u}_1}{\partial x_1} + \frac{\partial \tilde{u}_2}{\partial x_2}\right) \tag{7.136}$$

Roland W. Lewis, Perumal Nithiarasu, Kankanhalli N. Seetharamu, *Fundamentals of the Finite Element Method for Heat and Fluid Flow,* John Wiley & Sons Ltd, 2004

Step 1:

$$[\mathbf{M}]\frac{\Delta\{\tilde{u}_1\}}{\Delta t} = -[\mathbf{C}]\{u_1\}^n - [\mathbf{K}_m]\{u_1\}^n - [\mathbf{K}_s]\{u_1\}^n + \{f_1\}$$

$$[\mathbf{M}]\frac{\Delta\{\tilde{u}_2\}}{\Delta t} = -[\mathbf{C}]\{u_2\}^n - [\mathbf{K}_m]\{u_2\}^n - [\mathbf{K}_s]\{u_2\}^n + \{f_2\}$$

Step 2:

$$[\mathbf{K}]\{p\}^n = -\frac{1}{\Delta t}\left[[\mathbf{G}_1]\{\tilde{u}_1\} + [\mathbf{G}_2]\{\tilde{u}_2\}\right] + \{f_3\}$$

Step 3:

$$[\mathbf{M}]\{u_1\}^{n+1} = [\mathbf{M}]\{\tilde{u}_1\}^n - \Delta t[\mathbf{G}_1]\{p\}^n$$

$$[\mathbf{M}]\{u_2\}^{n+1} = [\mathbf{M}]\{\tilde{u}_2\}^n - \Delta t[\mathbf{G}_2]\{p\}^n$$

Semi-explicit Parareal method based on convergence acceleration technique

Loïc MICHEL

February 18, 2014

Abstract

The Parareal algorithm is used to solve time-dependent problems considering multiple solvers that may work in parallel. The key feature is a initial rough approximation of the solution that is iteratively refined by the parallel solvers. We report a derivation of the Parareal method that uses a convergence acceleration technique to improve the accuracy of the solution. Our approach uses firstly an explicit ODE solver to perform the parallel computations with different time-steps and then, a decomposition of the solution into specific convergent series, based on an extrapolation method, allows to refine the precision of the solution. Our proposed method exploits basic explicit integration methods, such as for example the explicit Euler scheme, in order to preserve the simplicity of the global parallel algorithm. The first part of the paper outlines the proposed method applied to the simple explicit Euler scheme and then the derivation of the classical Parareal algorithm is discussed and illustrated with numerical examples.

Conclusions & Future Work

- Complex phenomena

- We are not Parallel in Time yet!
  as fast as possible
- pfasst and/or parareal
- Performance & Scalability on Cray XE6 (HLRS)
- Fast & Better Results

…Thank you for your attention ! ! !