igatools: a TERRIFIC package for Numerical Assembly of Mathematical Operators.

September 1, 2011-August 30, 2014
www.terrific-project.eu

European Community's Seventh Framework Programme
Grant Agreement 284981
Call FP7-2011-NMP-ICT-FoF

Nicola Cavallini, Pablo Antolin, Massimiliano Martinelli, Sebastian Pauletti, Oliver Wëger, Carlo Lovadina, Giancarlo Sangalli

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

The TERRIFIC toolkit

“Write programs that do one thing and do it well.
Write programs to work together.”

igatools
General Purpose Assembly of Operators

TUKL
Application oriented Solvers using B−Splines and NURBS.

JKU
Feature detection and volume segmentation

INRIA
-geometric modeler
- tools for visualization, interaction and computation.

SINTEF
-geometry and topology
- model exchange
-isogeometric modelling

TERRIFIC

OO is the programming paradigm that realize interaction.

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Why OO Software?

"Write programs that do one thing and do it well.
Write programs to work together."

igatools
General Purpose Assembly of Operators

TUKL
Application oriented Solvers using B−Splines and NURBS.

JKU
Feature detection and volume segmentation

INRIA
-geometric modeler
- tools for visualization, interaction and computation.

SINTEF
-geometry and topology
- model exchange
-isogeometric modelling

TERRIFIC

>>> Object obj

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Why OO Software?

"Write programs that do one thing and do it well.
Write programs to work together."

igatools

General Purpose Assembly of Operators

TUKL

Application oriented Solvers using B−Splines and NURBS.

JKU

Feature detection and volume segmentation

INRIA

-geometric modeler
- tools for visualization, interaction and computation.

SINTEF

-geometry and topology
- model exchange
-isogeometric modelling

TERRIFIC

```
>>> Object obj
>>> smth = obj.do_something()
```

Why OO Software?

"Write programs that do one thing and do it well.
Write programs to work together."

igatools
General Purpose Assembly of Operators

TUKL
Application oriented Solvers using B−Splines and NURBS.

JKU
Feature detection and volume segmentation

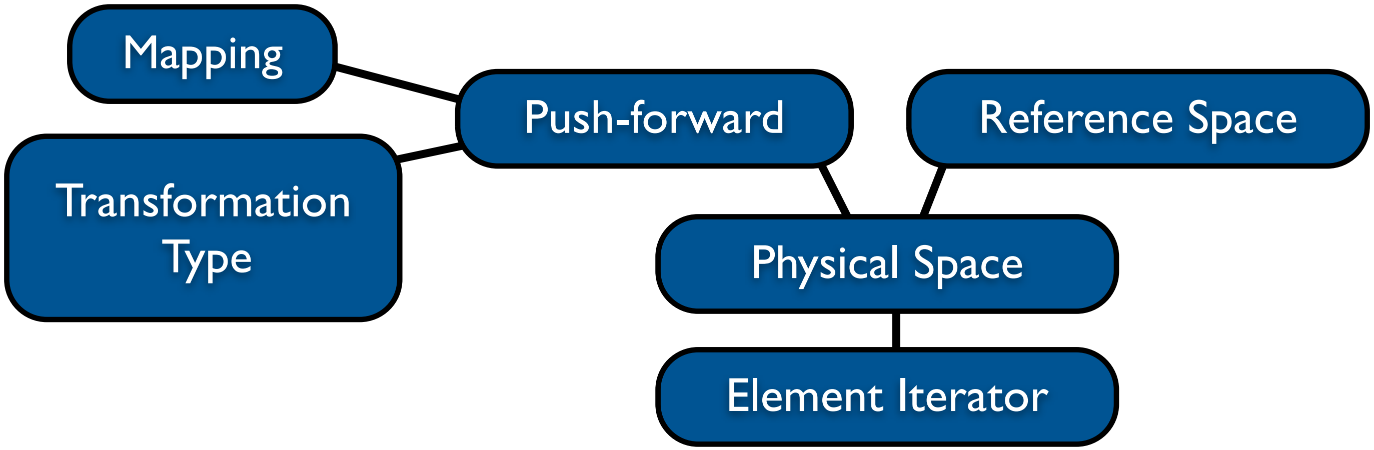
INRIA
-geometric modeler
- tools for visualization, interaction and computation.

SINTEF
-geometry and topology
- model exchange
-isogeometric modelling

TERRIFIC

```
>>> Object obj
>>> smth = obj.do_something()
>>> Another anth
>>> cool = anth.do_something_cool(obj.do_something())
```

igatools



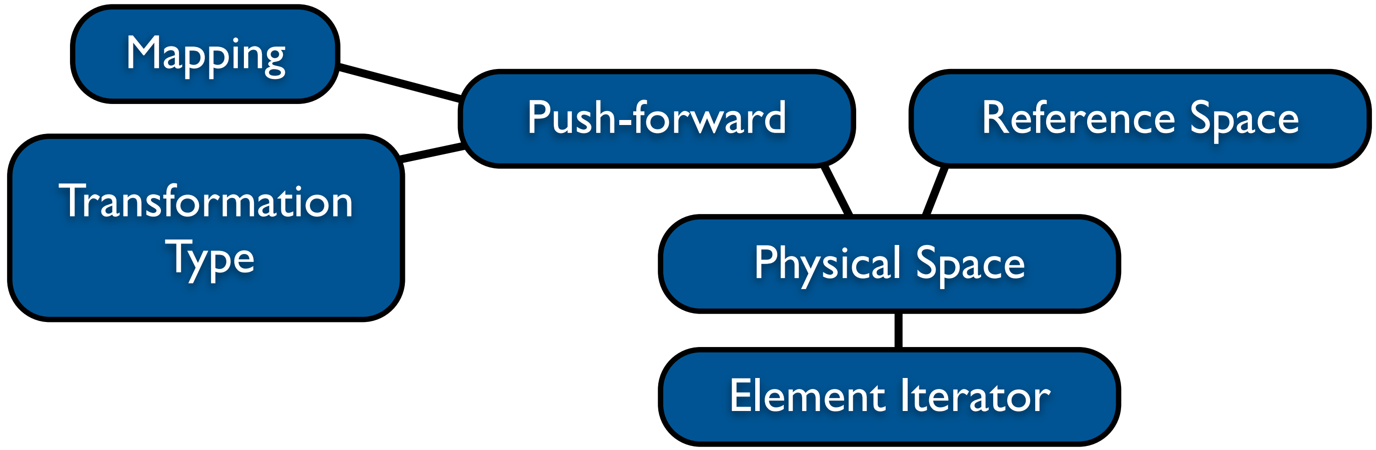
$$\int_\Omega \nabla u \cdot \nabla v = \sum_K \int_K \nabla u_j \cdot \nabla v_i = A_{ij}$$

```
element = phys_space->begin();
endc = phys_space->end();

for(; element != endc; ++element) {
    local_dofs = element->get_local_to_global();
    for (q = 0; q < n_qpoints; q++) {
      for (i = 0; i < n_basis; i ++) {
        for (j = 0; j <n_basis; j ++) {
          local_matrix(i,j) += scalar_product(
                element->get_gradient(i,q) ,  element->get_gradient(j,q) )
                * element->get_w_mes(q);  }  }  }
    matrix.add(local_dofs,local_dofs,local_matrix);}
```

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

igatools




```
element = phys_space->begin();
endc = phys_space->end();
```

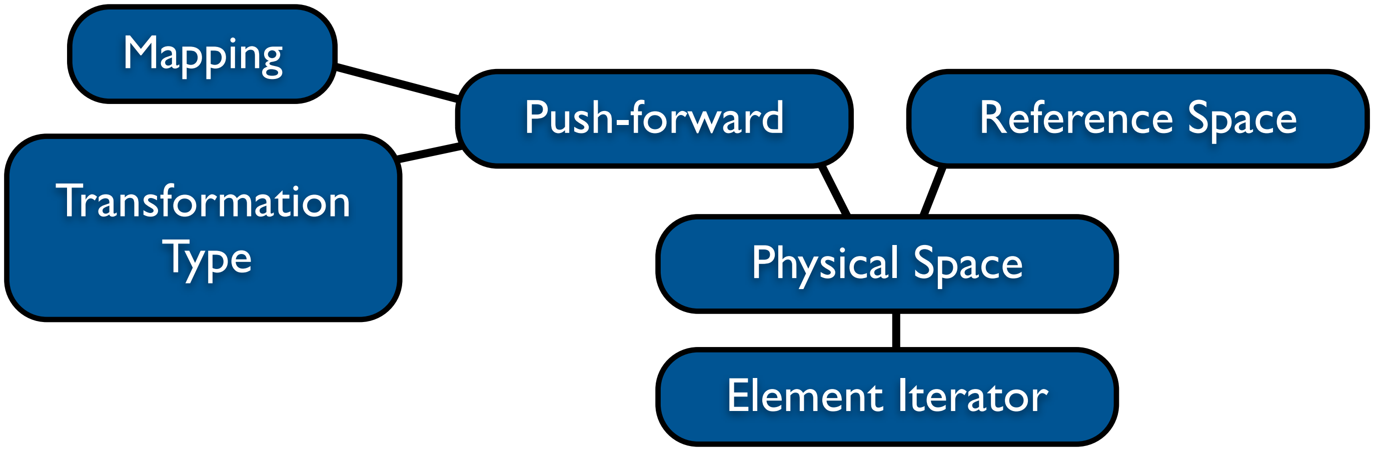
$$\int_{\Omega} \nabla u \cdot \nabla v = \sum_{K} \int_{K} \nabla u_j \cdot \nabla v_i = A_{ij}$$

```
for(; element != endc; ++element) {
    local_dofs = element->get_local_to_global();
    for (q = 0; q < n_qpoints; q++) {
      for (i = 0; i < n_basis; i ++) {
        for (j = 0; j <n_basis; j ++) {
          local_matrix(i,j) += scalar_product(
                      element->get_gradient(i,q) ,  element->get_gradient(j,q) )
                      * element->get_w_mes(q);  }  }  }
    matrix.add(local_dofs,local_dofs,local_matrix);}
```

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

igatools



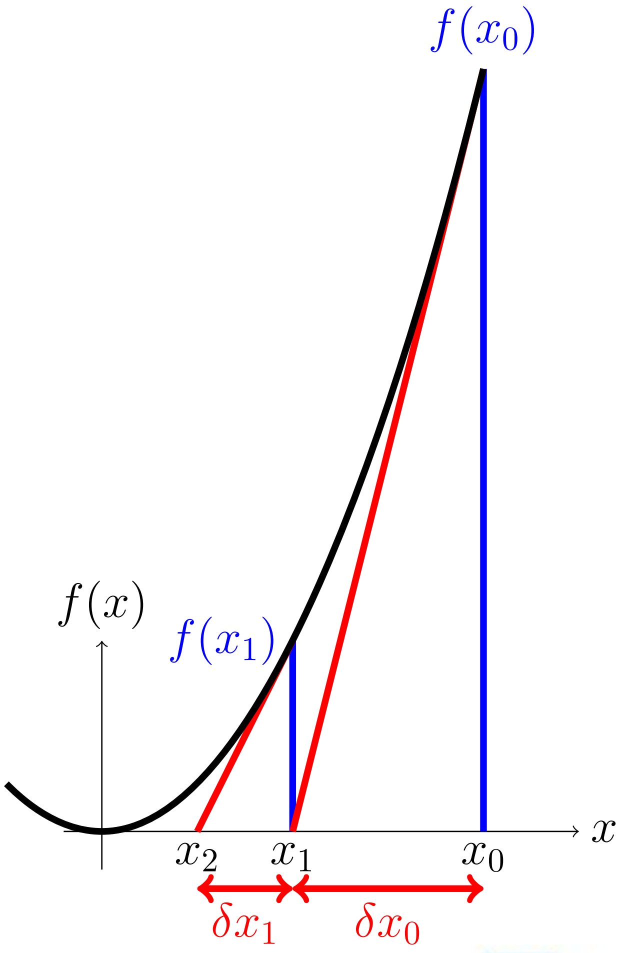
$$\int_\Omega \nabla u \cdot \nabla v = \sum_K \int_K \nabla u_j \cdot \nabla v_i = A_{ij}$$

```
element = phys_space->begin();
endc = phys_space->end();

for(; element != endc; ++element) {
    local_dofs = element->get_local_to_global();
    for (q = 0; q < n_qpoints; q++) {
      for (i = 0; i < n_basis; i ++) {
        for (j = 0; j <n_basis; j ++) {
          local_matrix(i,j) += scalar_product(
                element->get_gradient(i,q), element->get_gradient(j,q))
                * element->get_w_mes(q);  }  }  }
    matrix.add(local_dofs,local_dofs,local_matrix);}
```

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Computational Mechanics



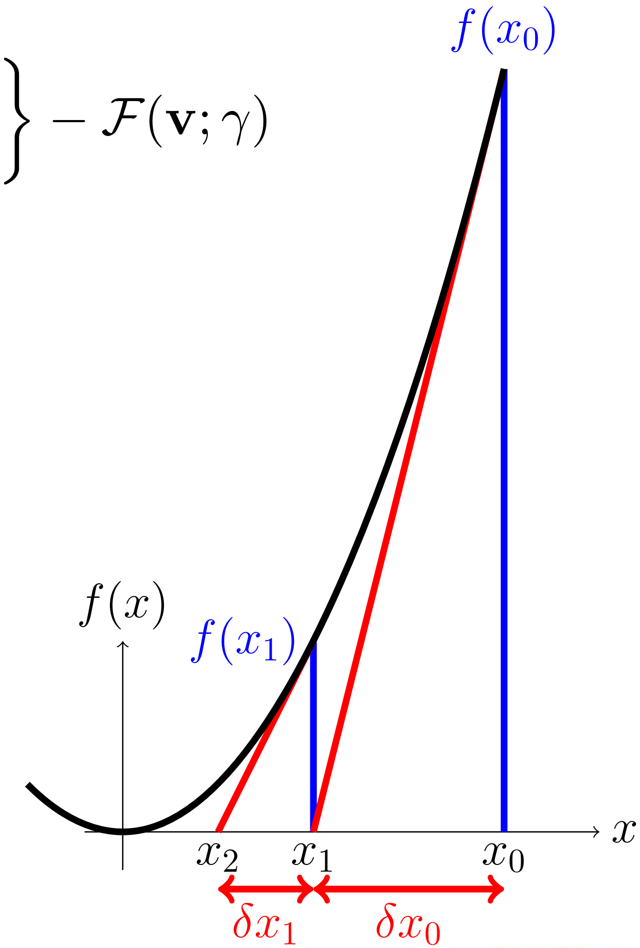
TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Computational Mechanics

- The Model Equations.

$$\Pi_m^{inc}(\hat{\mathbf{u}}, \hat{p}) = \int_\Omega \left\{ \frac{1}{2}\mu \left[\mathbf{I} : \hat{\mathbf{C}} - 2 \right] - \mu \ln \hat{J} + \hat{p}\ \Theta(\hat{J}) \right\} - \mathcal{F}(\mathbf{v}; \gamma)$$



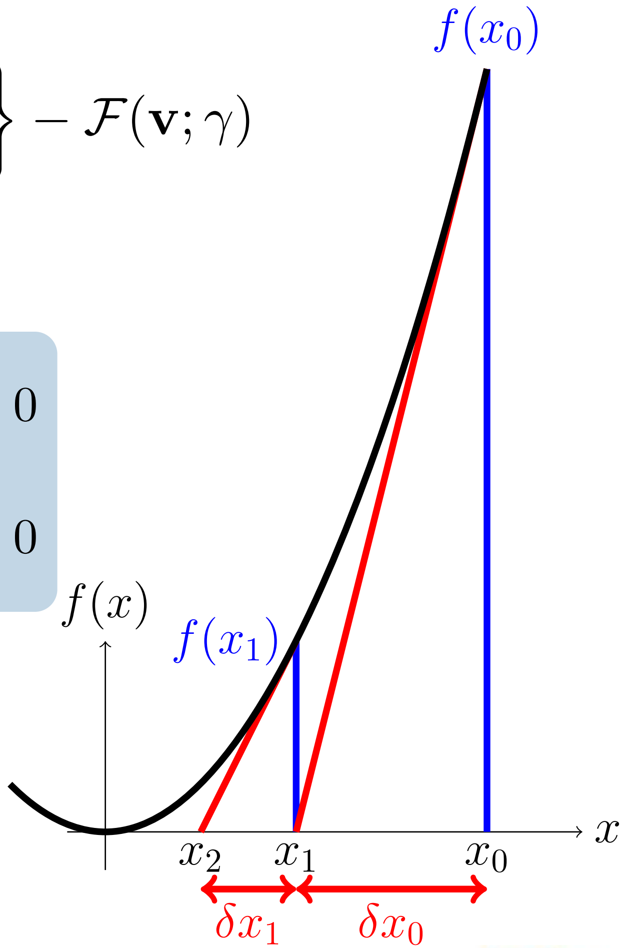
Computational Mechanics

- The Model Equations.

$$\Pi_m^{inc}(\hat{\mathbf{u}},\hat{p}) = \int_\Omega \left\{ \frac{1}{2}\mu \left[\mathbf{I} : \hat{\mathbf{C}} - 2 \right] - \mu \ln \hat{J} + \hat{p}\ \Theta(\hat{J}) \right\} - \mathcal{F}(\mathbf{v};\gamma)$$

$$\mathrm{d}\Pi_m^{inc}(\hat{\mathbf{u}},\hat{p})[\mathbf{v},q] = 0$$

$$\begin{cases} \mu \displaystyle\int_\Omega [\hat{\mathbf{F}} - \hat{\mathbf{F}}^{-T}] : \nabla\mathbf{v} + \int_\Omega \hat{p}\ \pi(\hat{J})\hat{\mathbf{F}}^{-T} : \nabla\mathbf{v} - \mathcal{F}(\mathbf{v};\gamma) = 0 \\ \displaystyle\int_\Omega \Theta(\hat{J}) q = 0 \end{cases}$$



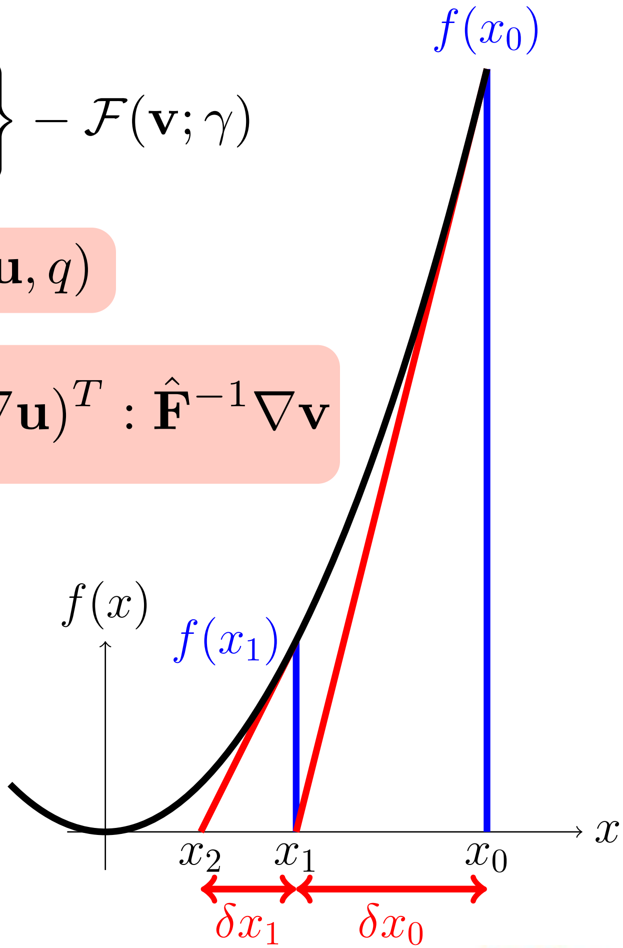
Computational Mechanics

- The Model Equations.

$$\Pi_m^{inc}(\hat{\mathbf{u}},\hat{p}) = \int_\Omega \left\{ \frac{1}{2}\mu \left[\mathbf{I} : \hat{\mathbf{C}} - 2 \right] - \mu \ln \hat{J} + \hat{p}\ \Theta(\hat{J}) \right\} - \mathcal{F}(\mathbf{v};\gamma)$$

$$\mathrm{d}^2\Pi_m^{inc}(\hat{\mathbf{u}},\hat{p})[(\mathbf{u},p),(\mathbf{v},q)] = a_\gamma(\mathbf{u},\mathbf{v}) + b_\gamma(\mathbf{v},p) + b_\gamma(\mathbf{u},q)$$

$$\begin{cases} a_\gamma(\mathbf{u},\mathbf{v}) := \mu \int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} + \int_\Omega \left[\mu - \hat{p}\ \pi(\hat{J}) \right] (\hat{\mathbf{F}}^{-1}\nabla\mathbf{u})^T : \hat{\mathbf{F}}^{-1}\nabla\mathbf{v} \\ \qquad + \int_\Omega \hat{p}\ \kappa(\hat{J})(\hat{\mathbf{F}}^{-T} : \nabla\mathbf{u})(\hat{\mathbf{F}}^{-T} : \nabla\mathbf{v}) \\ b_\gamma(\mathbf{v},q) := \int_\Omega q\ \pi(\hat{J})\hat{\mathbf{F}}^{-T} : \nabla\mathbf{v} \end{cases}$$



Computational Mechanics

$$\int_\Omega \hat{p}\ \pi(\hat{J})\hat{\mathbf{F}}^{-T} : \nabla \mathbf{v}$$

- Assembly of the mixed term:

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Computational Mechanics

$$\int_\Omega \hat{p}\ \pi(\hat{J})\hat{\mathbf{F}}^{-T} : \nabla \mathbf{v}$$

- Assembly of the mixed term:

```
T< 1, 1, tensor::cont, Tdouble > prex_q;

T< dim_ref, 1, tensor::cov,
T< dim_phys, 1, tensor::cont, Tdouble > > defgrad_q, defo_grad_q;

for (; defo_elem != defo_end ; ++defo_elem, ++prex_elem){
   for (Index q = 0; q < n_qp; ++q){

      for(Index i = 0; i < defo_loc_ndofs; ++i)
         defo_grad_q += defo_vec(dof) * grad_phi_q[i];

      for(Index i = 0; i < prex_loc_ndofs; ++i)
         prex_q += prex_vec(dof) * prex_phi_q[i];

      defgrad_q = unit_defgrad + defo_grad_q;

      inverse<dim_ref, dim_phys> (defgrad_q,defgrad_inv_q);
      defgrad_invT_q = co_tensor(transpose(defgrad_inv_q));

      for (Index i = 0; i < defo_loc_ndofs; ++i){

         defo_loc_res(i) += mat_mu * prex_q[0] *
            scalar_product(defgrad_invT_q,
                           grad_phi_q[i]) *w[q] ;}}}
```

Listing 1: Assemble of the mixed term in the momentum equation.

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Computational Mechanics

$$\int_\Omega \hat{p}\ \pi(\hat{J})\hat{\mathbf{F}}^{-T} : \nabla \mathbf{v}$$

- Assembly of the mixed term:

```
T< 1, 1, tensor::cont, Tdouble > prex_q;

T< dim_ref, 1, tensor::cov,
T< dim_phys, 1, tensor::cont, Tdouble > > defgrad_q, defo_grad_q;

for (; defo_elem != defo_end ; ++defo_elem, ++prex_elem){
   for (Index q = 0; q < n_qp; ++q){

      for(Index i = 0; i < defo_loc_ndofs; ++i)
         defo_grad_q += defo_vec(dof) * grad_phi_q[i];

      for(Index i = 0; i < prex_loc_ndofs; ++i)
         prex_q += prex_vec(dof) * prex_phi_q[i];

      defgrad_q = unit_defgrad + defo_grad_q;

      inverse<dim_ref, dim_phys> (defgrad_q,defgrad_inv_q);
      defgrad_invT_q = co_tensor(transpose(defgrad_inv_q));

      for (Index i = 0; i < defo_loc_ndofs; ++i){

         defo_loc_res(i) += mat_mu * prex_q[0] *
            scalar_product(defgrad_invT_q,
                           grad_phi_q[i]) *w[q] ;}}}
```

Listing 1: Assemble of the mixed term in the momentum equation.

Computational Mechanics

$$\int_{\Omega} \hat{p}\ \pi(\hat{J})\hat{\mathbf{F}}^{-T} : \nabla \mathbf{v}$$

- Assembly of the mixed term:

```
T< 1, 1, tensor::cont, Tdouble > prex_q;

T< dim_ref, 1, tensor::cov,
T< dim_phys, 1, tensor::cont, Tdouble > > defgrad_q, defo_grad_q;

for (; defo_elem != defo_end ; ++defo_elem, ++prex_elem){
   for (Index q = 0; q < n_qp; ++q){

      for(Index i = 0; i < defo_loc_ndofs; ++i)
         defo_grad_q += defo_vec(dof) * grad_phi_q[i];

      for(Index i = 0; i < prex_loc_ndofs; ++i)
         prex_q += prex_vec(dof) * prex_phi_q[i];

      defgrad_q = unit_defgrad + defo_grad_q;

      inverse<dim_ref, dim_phys> (defgrad_q,defgrad_inv_q);
      defgrad_invT_q = co_tensor(transpose(defgrad_inv_q));

      for (Index i = 0; i < defo_loc_ndofs; ++i){

         defo_loc_res(i) += mat_mu * prex_q[0] *
            scalar_product(defgrad_invT_q,
                           grad_phi_q[i]) *w[q] ;}}}
```

Listing 1: Assemble of the mixed term in the momentum equation.

TERRIFIC Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Computational Mechanics

$$\int_\Omega \hat{p}\ \pi(\hat{J})\hat{\mathbf{F}}^{-T} : \nabla \mathbf{v}$$

- Assembly of the mixed term:

```
T< 1, 1, tensor::cont, Tdouble > prex_q;

T< dim_ref, 1, tensor::cov,
T< dim_phys, 1, tensor::cont, Tdouble > > defgrad_q, defo_grad_q;

for                                                        lem){
    f



                                                           ];






                                                           inv_q);
                                                           nv_q));




                scalar_product(defgrad_invT_q,
                               grad_phi_q[i]) *w[q] ;}}}
```

GOAL:
Respect the mathematical formulation of operators, so that any SciCom user can easily assemble sophisticated operators.

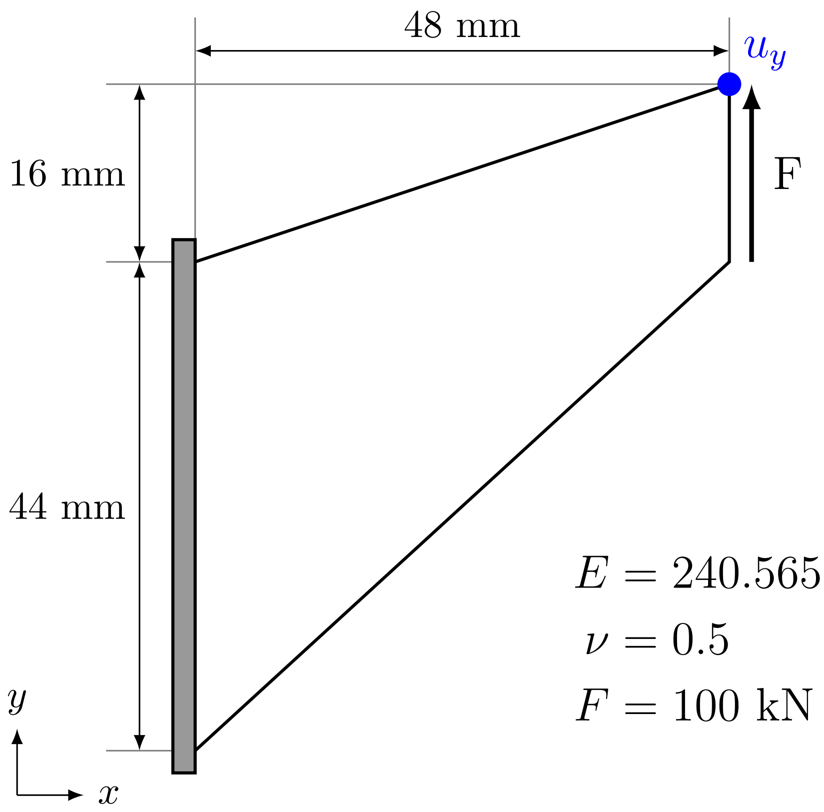
Listing 1: Assemble of the mixed term in the momentum equation.

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Computational Mechanics

- Benchmark Cook's membrane.



$E = 240.565$ MPa

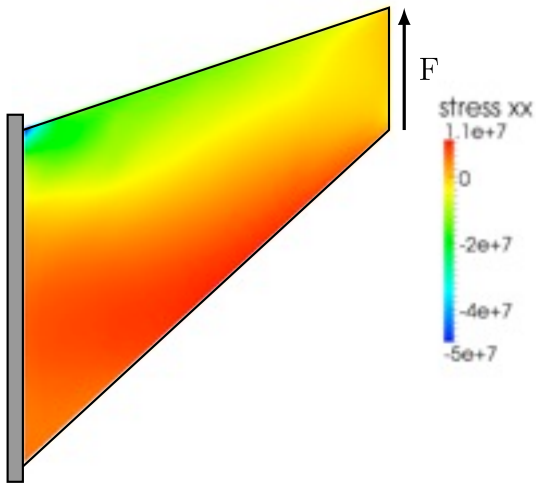
$\nu = 0.5$

$F = 100$ kN

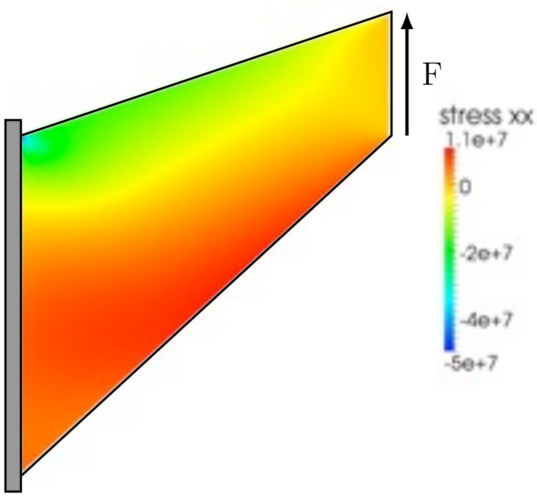
Computational Mechanics

- Benchmark Cook's membrane.

shape functions $= S_3^1/S_2^1$
elements $= 16 \times 16$



shape functions $= S_3^1/S_2^1$
elements $= 4 \times 4$



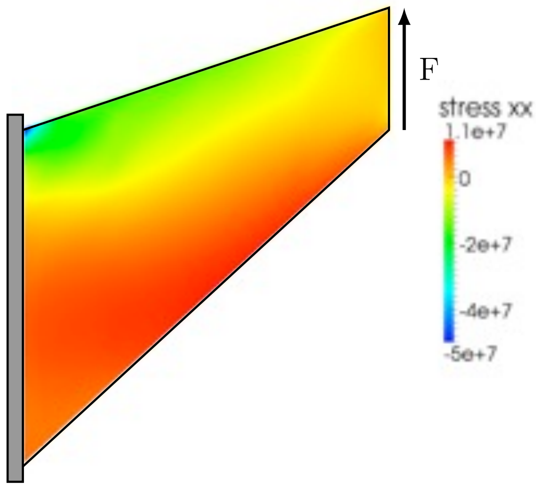
TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

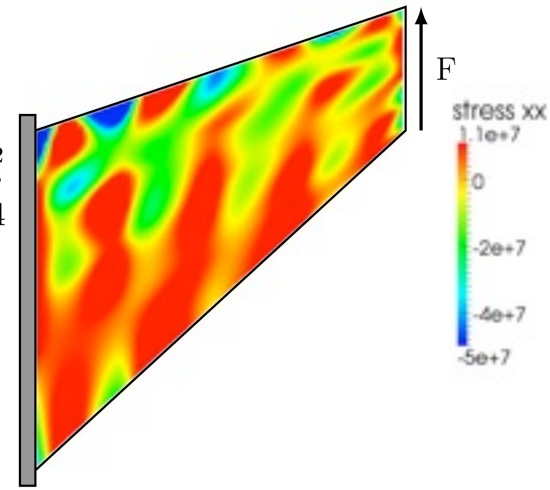
Computational Mechanics

- Benchmark Cook's membrane.

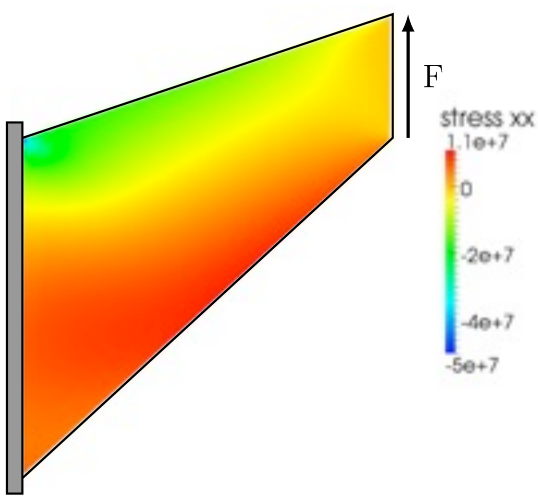
shape functions $= S_3^1/S_2^1$
elements $= 16 \times 16$



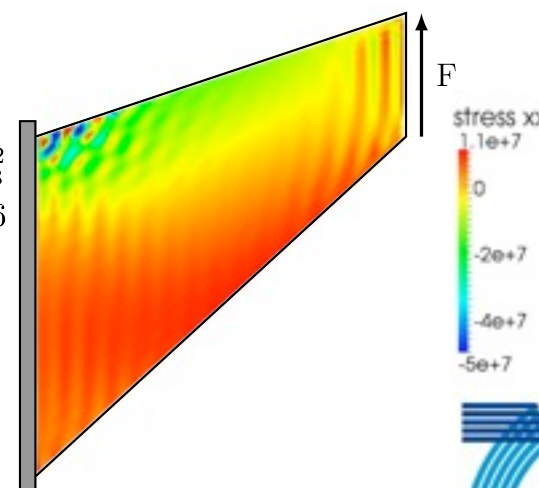
shape functions $= S_3^2$
elements $= 4 \times 4$



shape functions $= S_3^1/S_2^1$
elements $= 4 \times 4$

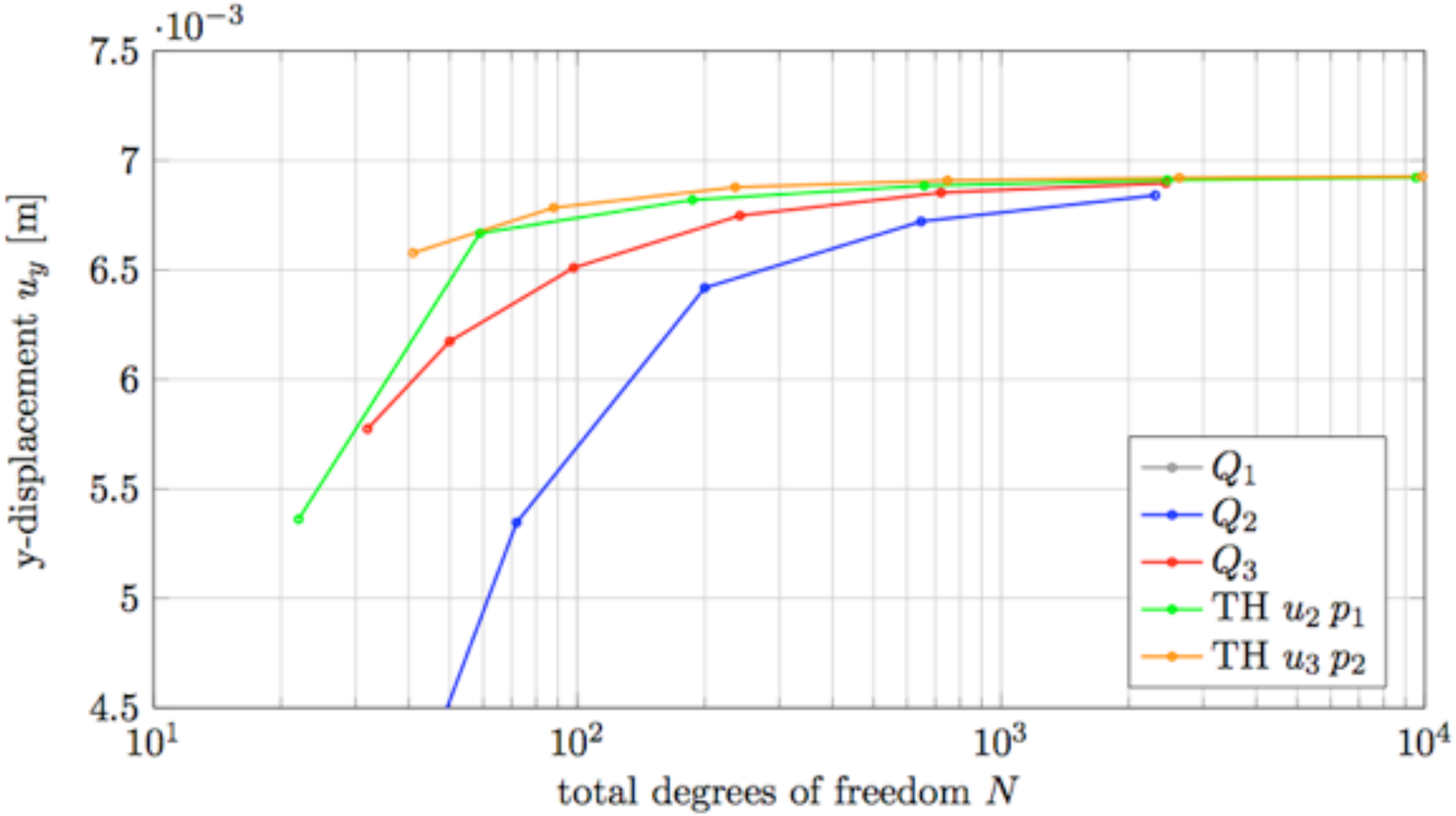


shape functions $= S_3^2$
elements $= 16 \times 16$



Computational Mechanics

- Benchmark Cook's membrane.

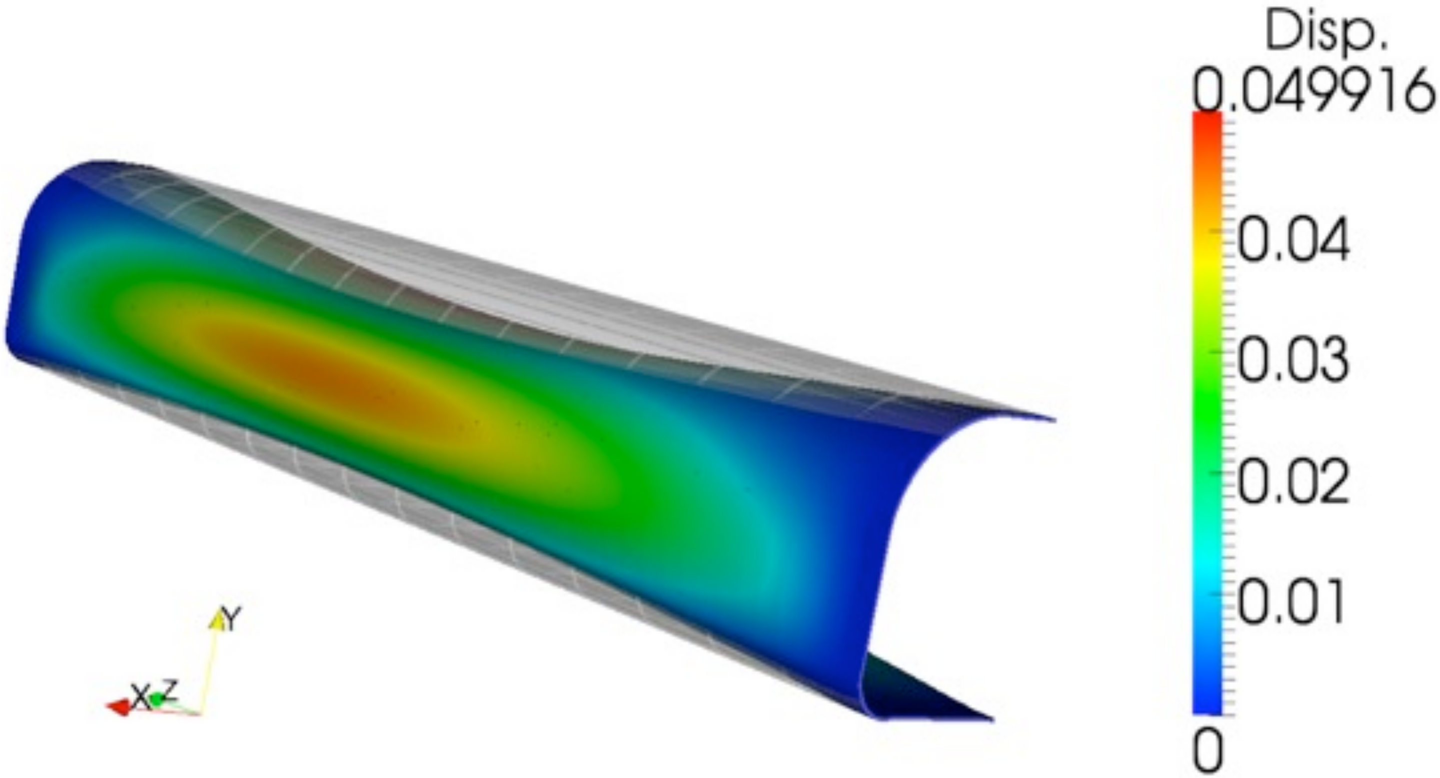


TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

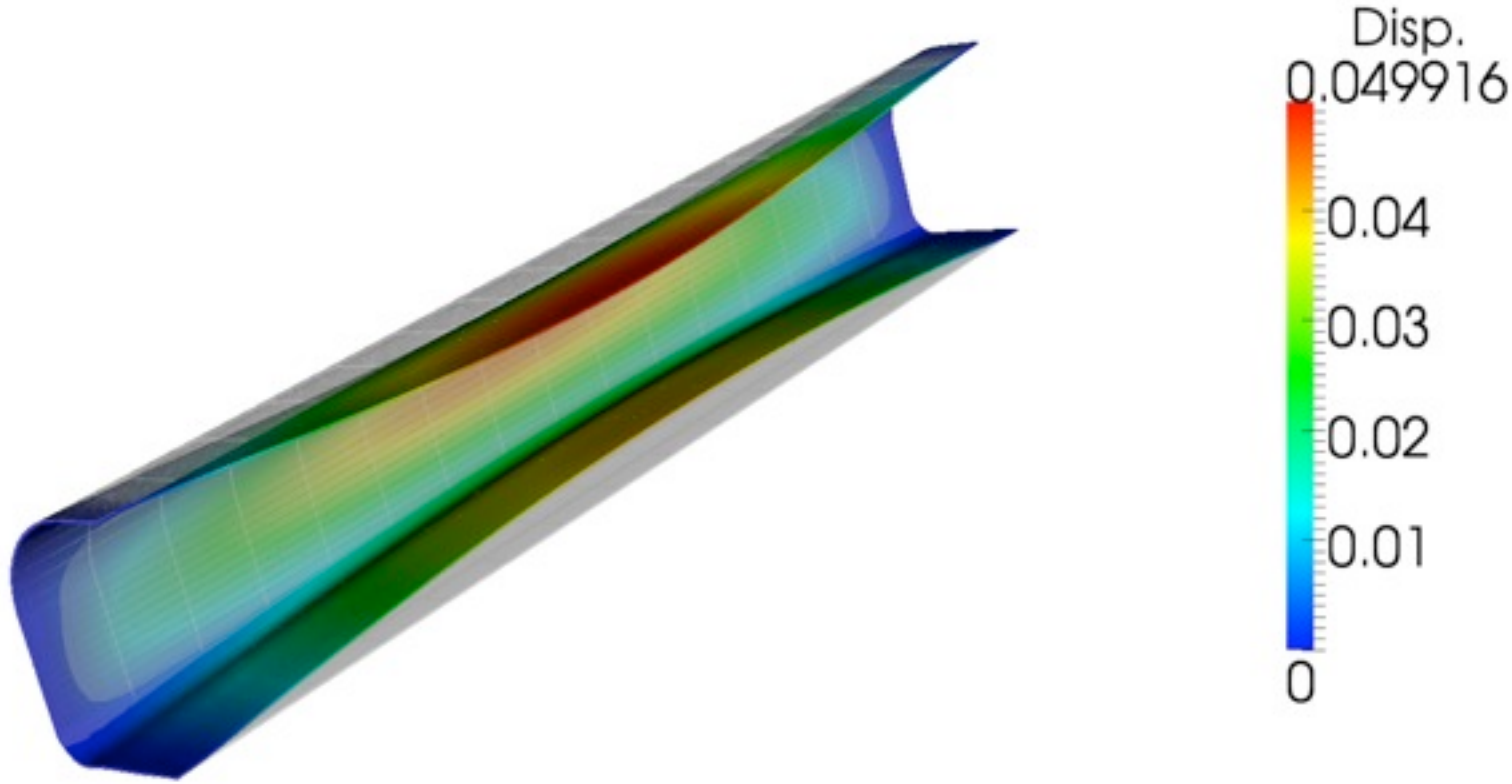
Computational Mechanics

- Industrial Test Case



Computational Mechanics

- Industrial Test Case



TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

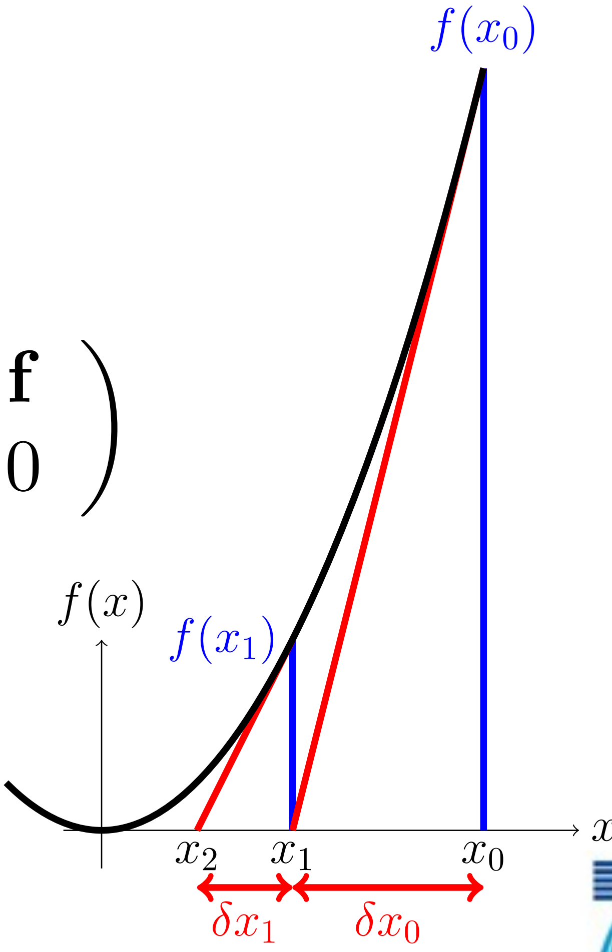
Fluid Dynamics

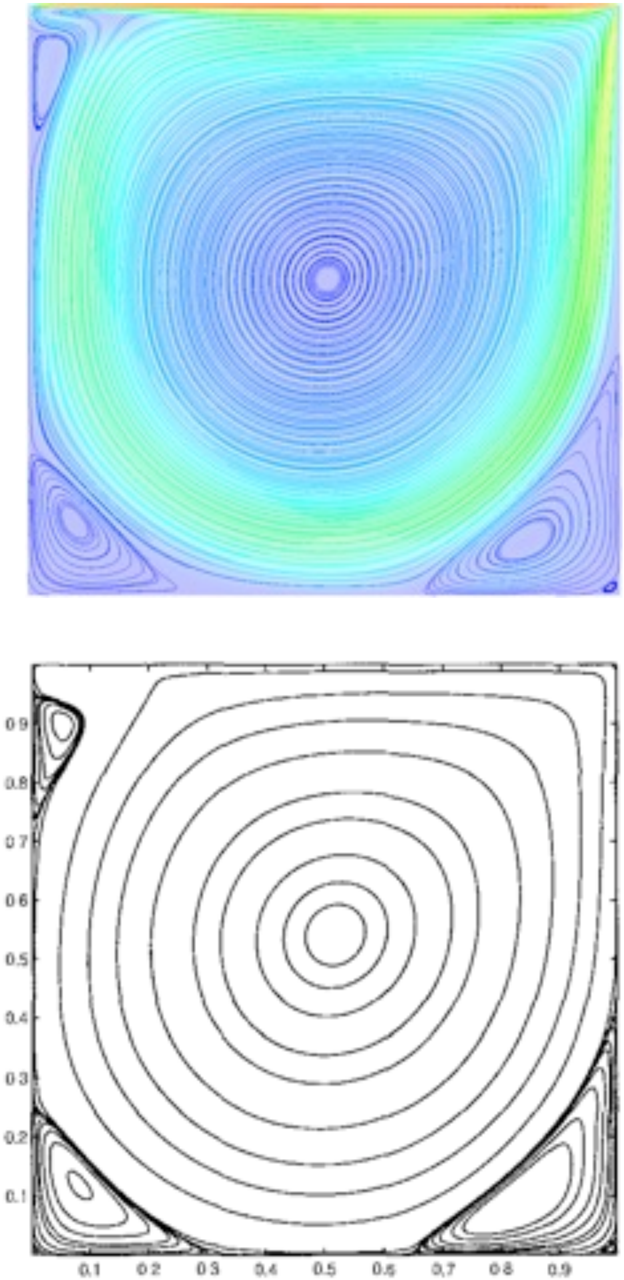
- The Model Equations.

$$\rho\,(\nabla\mathbf{u})\,\mathbf{u} - \mu\nabla^2\mathbf{u} + \nabla p = \mathbf{f}$$
$$\nabla\cdot\mathbf{u} = 0$$

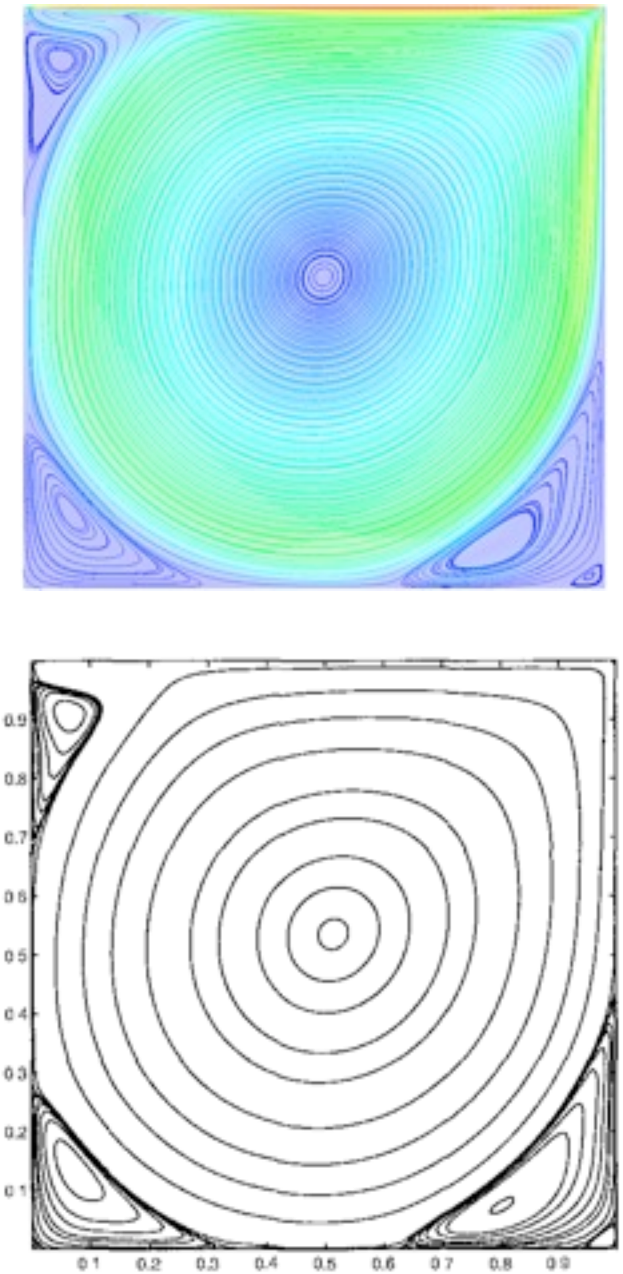
In Matrix Form:

$$\begin{pmatrix} A + N(\mathbf{u}) & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}$$





Re = 3200



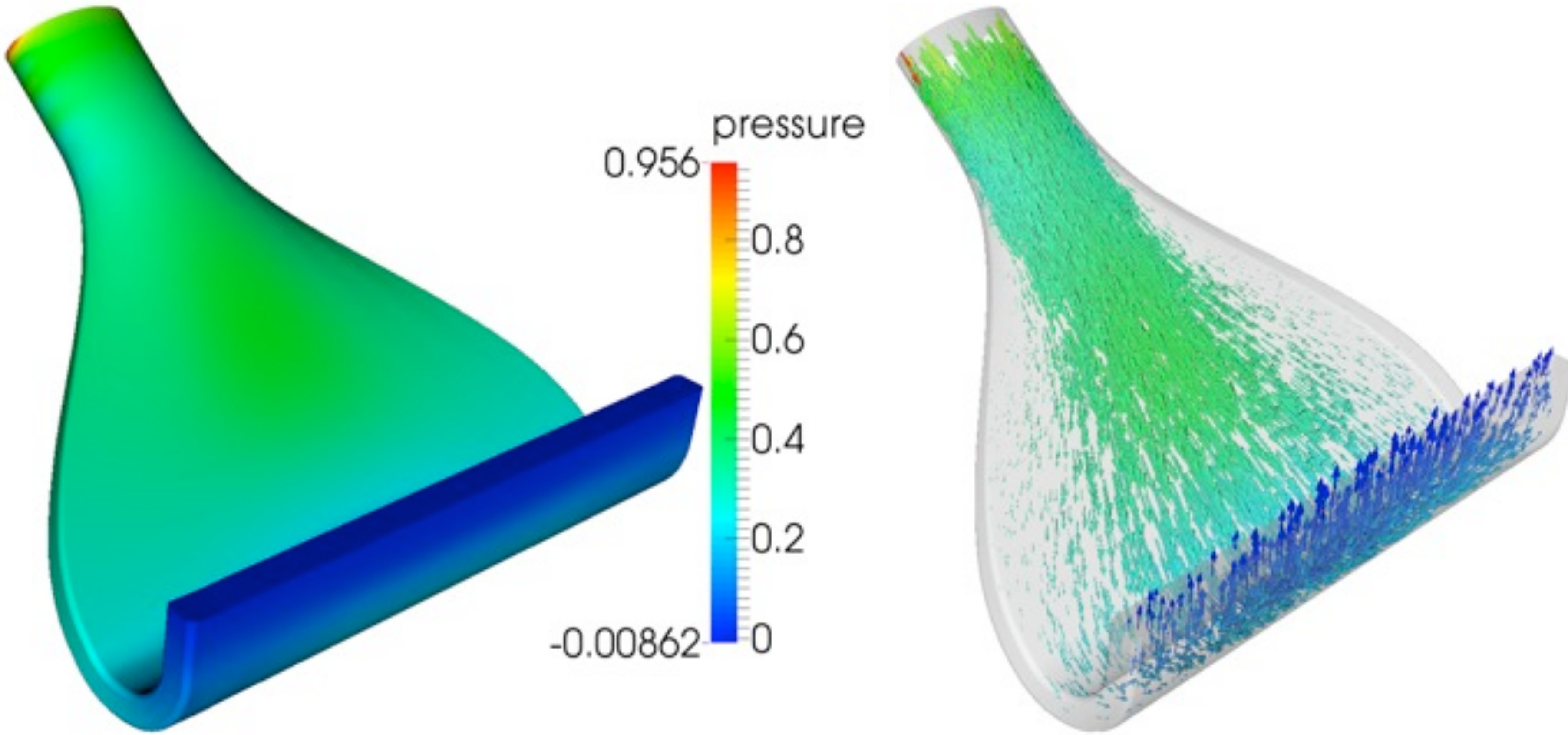
Re = 5000

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Fluid Dynamics

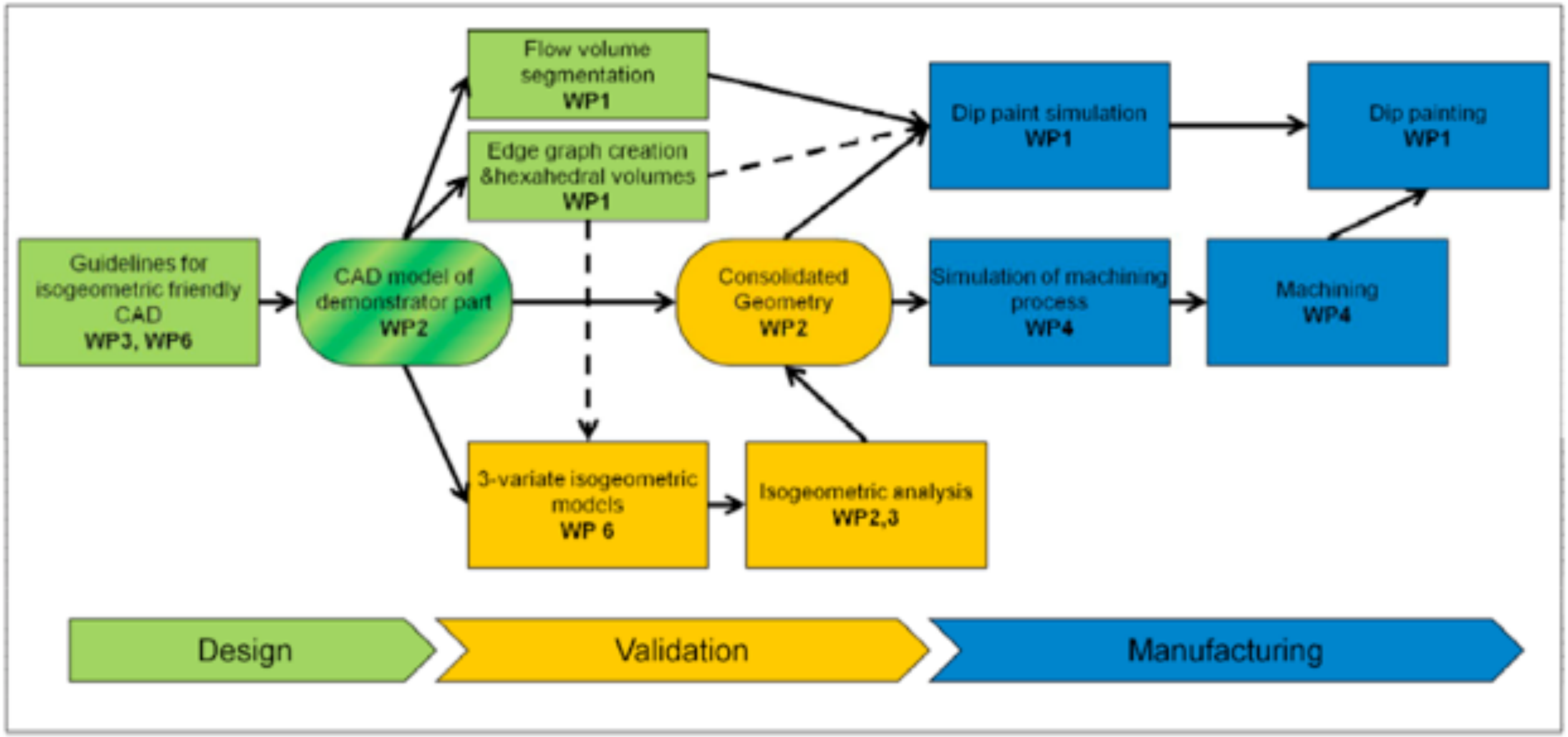
- Industrial Test Case



TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

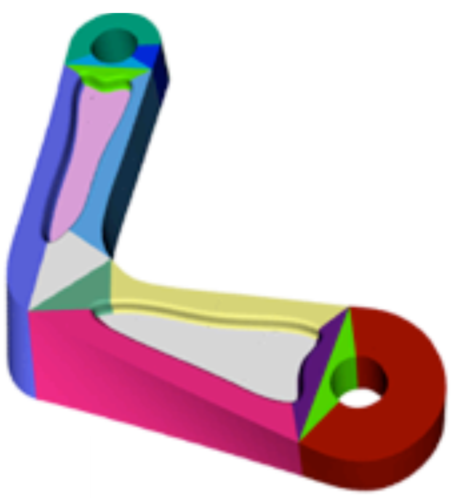
Interplay on The Demonstrator



TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

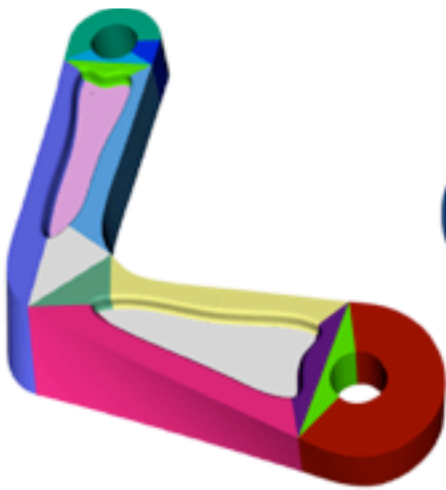
Interplay on The Demonstrator



TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Interplay on The Demonstrator



JOHANNES KEPLER UNIVERSITÄT LINZ | JKU

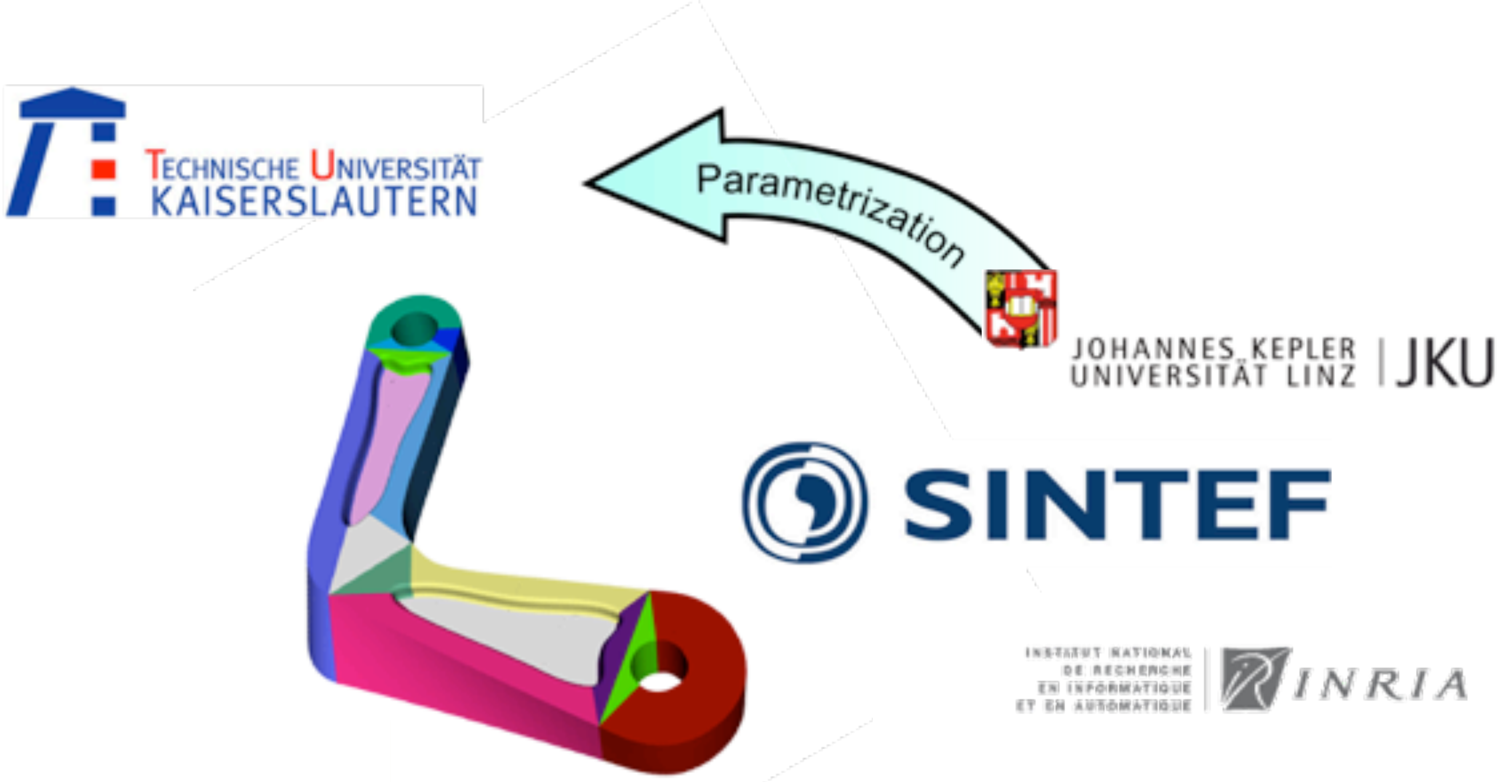
SINTEF

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE | INRIA

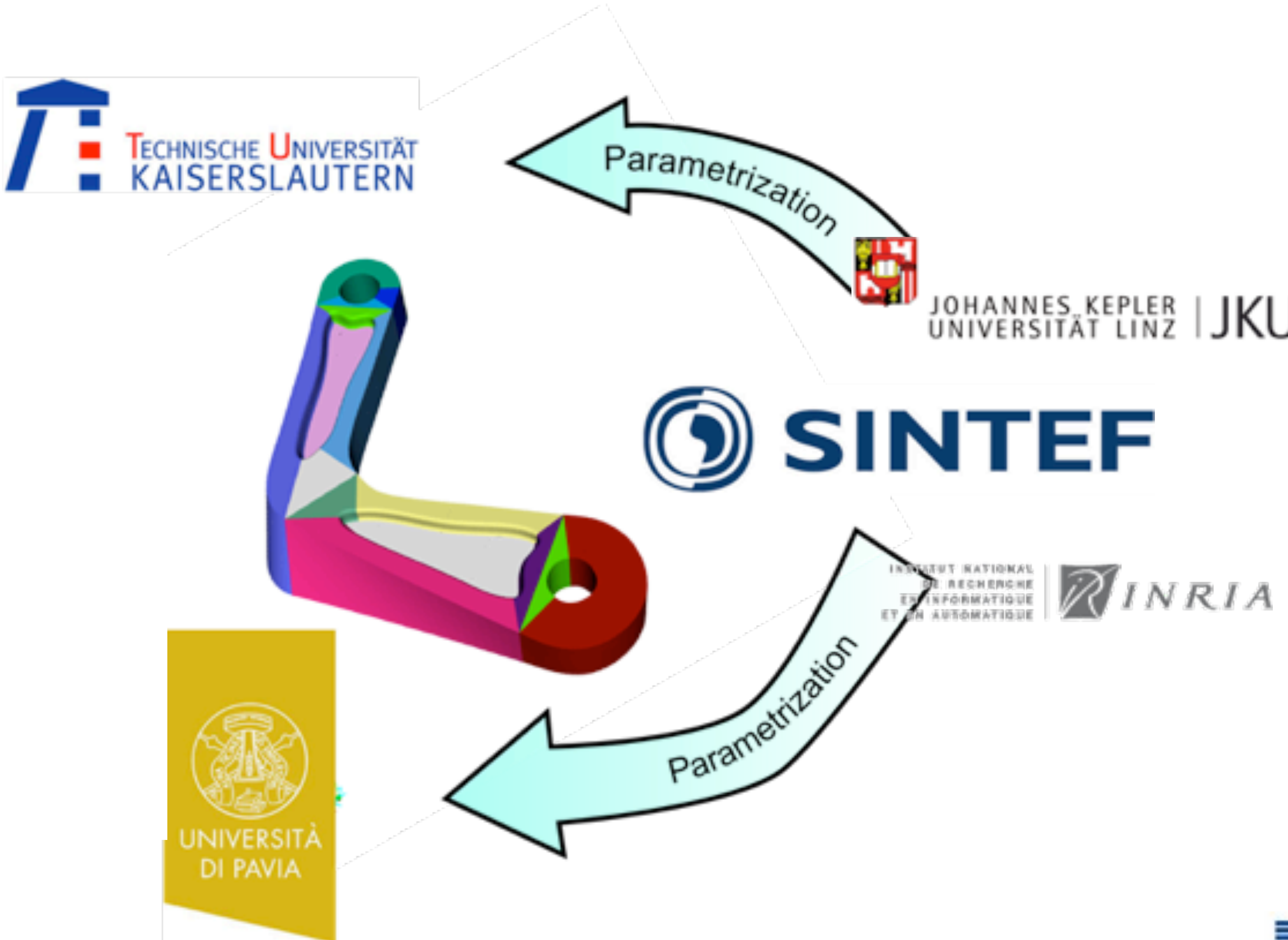
TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

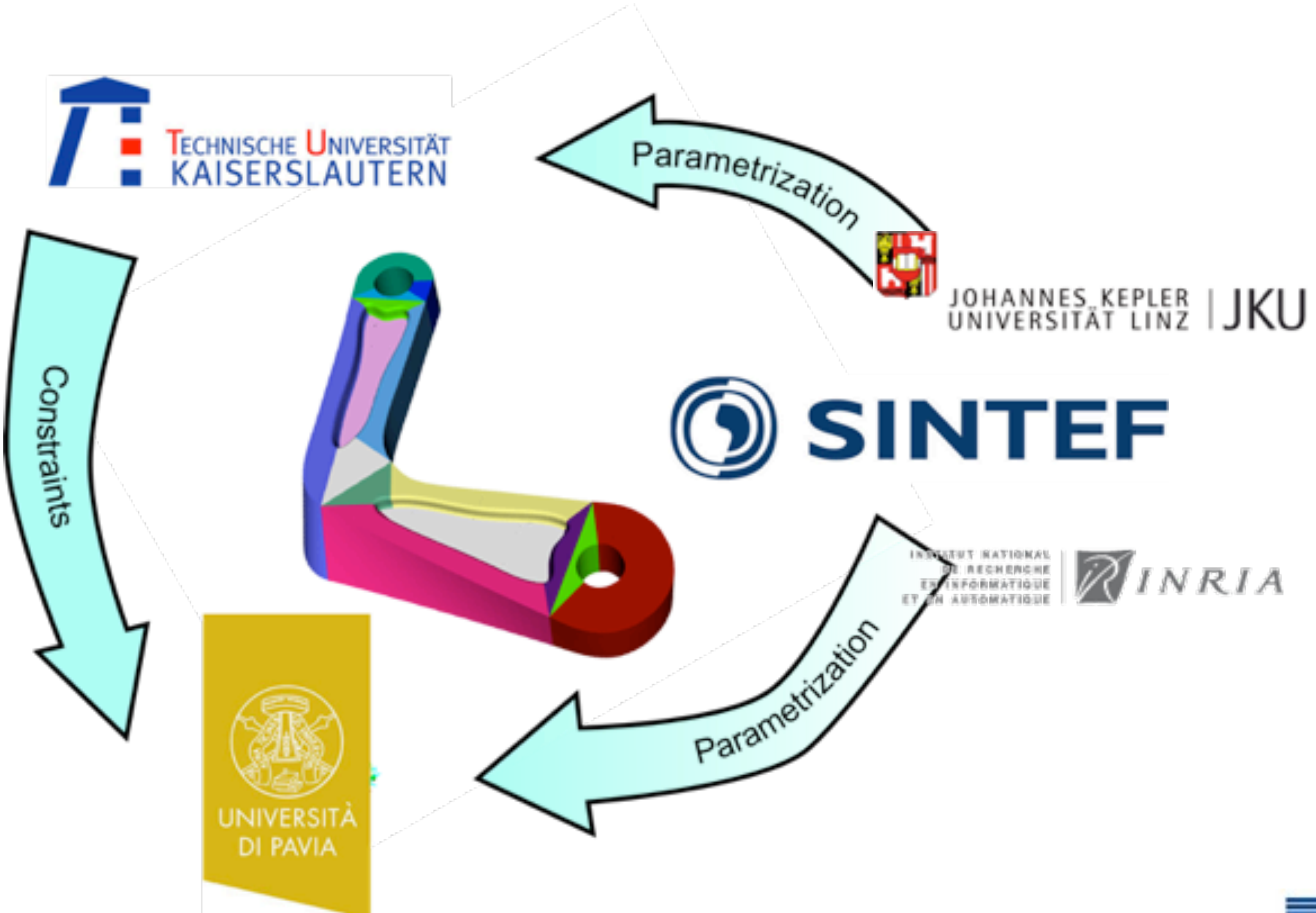
Interplay on The Demonstrator



Interplay on The Demonstrator



Interplay on The Demonstrator



Constraints define the matching of the solution in between patches

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

Conclusions

- We developed a modular software that is capable of interfacing with other TERRIFIC modules.
- On the PDE side, this software makes easy the assembly of sophisticated operators.

Fluid Dynamics

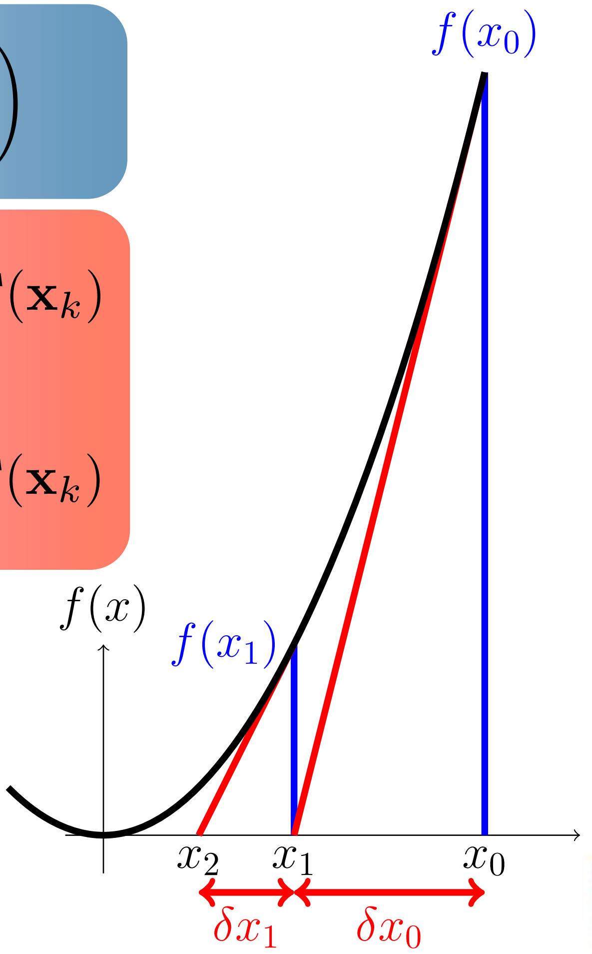
$$\begin{pmatrix} A+N(\mathbf{u}_0) & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}_k = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}$$

$$\mathbf{F}(\mathbf{x}_k) = \begin{pmatrix} A+N(\mathbf{u}_k) & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}_k - \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} A+N(\mathbf{u}_k) & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta\mathbf{u} \\ \delta p \end{pmatrix}_k = -\mathbf{F}(\mathbf{x}_k)$$

$$\begin{pmatrix} A+N(\mathbf{u}_k)+D(\mathbf{u}_k) & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta\mathbf{u} \\ \delta p \end{pmatrix}_k = -\mathbf{F}(\mathbf{x}_k)$$

$$\begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}_{k+1} = \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix}_k + \begin{pmatrix} \delta\mathbf{u} \\ \delta p \end{pmatrix}_k$$



$$N_{ij} = \int_\Omega (\nabla \mathbf{u}_j)\,\mathbf{u} \cdot \mathbf{v}_i =$$

$$\int_\Omega \begin{pmatrix} \partial_x u_x & \partial_y u_x \\ \partial_x u_y & \partial_y u_y \end{pmatrix}_j \begin{pmatrix} u_x \\ u_y \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}_i$$

```
for (dof_index q = 0; q < quad.get_num_points(); q++)
{
   Tensor<dim_phys, rank, contravariant, double> vel_q;

   for (dof_index i = 0; i <local_ndofs; i ++){
      auto phi = element->get_value(i,q);
      vel_q = vel_q+vel[local_dofs[i]]*phi;}

   for (dof_index i = 0; i <local_ndofs; i ++){
      for (dof_index j = 0; j <local_ndofs; j ++){
         adv_ij = scalar_product(
                  action(element->get_gradient(j,q),vel_q),
                            element->get_value(i,q) )*
                            element->get_w_measures()[q;]}}
}
```

Listing 1: Advection assemble.

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME

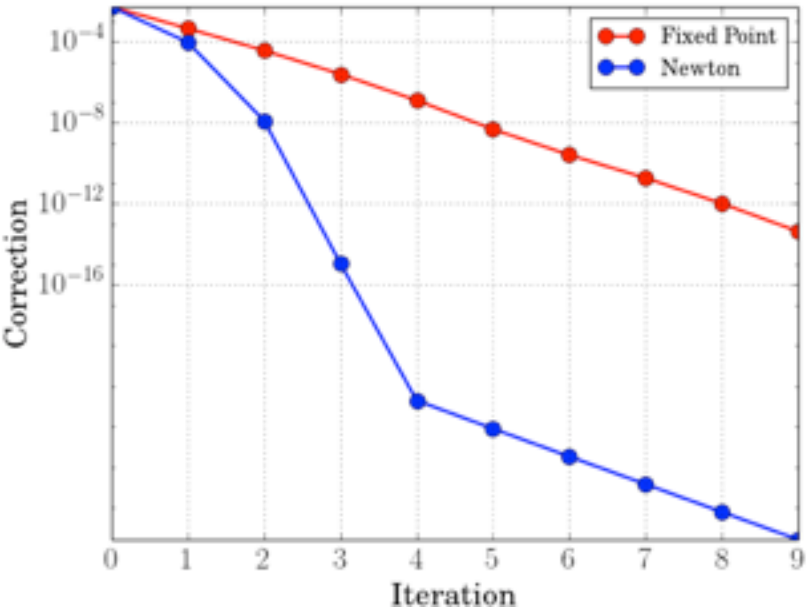
$$D_{ij} = \int_\Omega (\nabla \mathbf{u})\, \mathbf{u}_j \cdot \mathbf{v}_i =$$

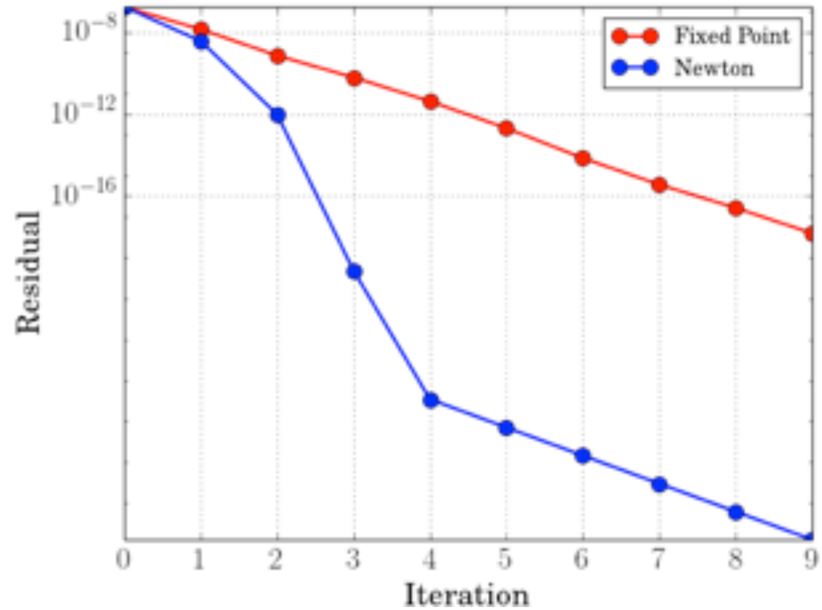
$$\int_\Omega \begin{pmatrix} \partial_x u_x & \partial_y u_x \\ \partial_x u_y & \partial_y u_y \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix}_j \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}_i$$

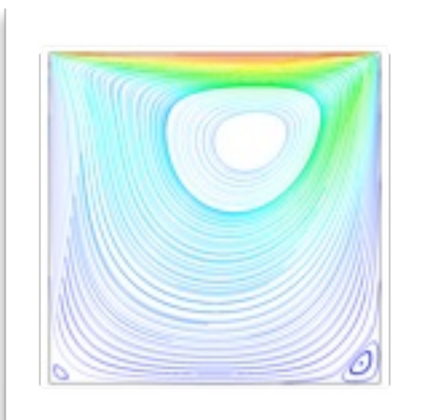
```
for (dof_index q = 0; q < quad.get_num_points(); q++)
{
   Tensor<dim_phys, rank, covariant,
   Tensor<dim_phys, rank, contravariant, double>> grad_vel_q;

   for (dof_index i = 0; i <local_ndofs; i++){
      auto grad_phi = element->get_gradient(i,q);
      grad_vel_q = grad_vel_q + vel[local_dofs[i]] * grad_phi;}

   for (dof_index i = 0; i <local_ndofs; i++){
      for (dof_index j = 0; j <local_ndofs; j++){
         jac_ij = scalar_product(action(
                                    grad_vel_q,
                                           element->get_value(j,q)),
                                    element->get_value(i,q) )*
                                    element->get_w_measures()[q];}}
}
```

Listing 1: Jacobian assemble.

Fluid Dynamics





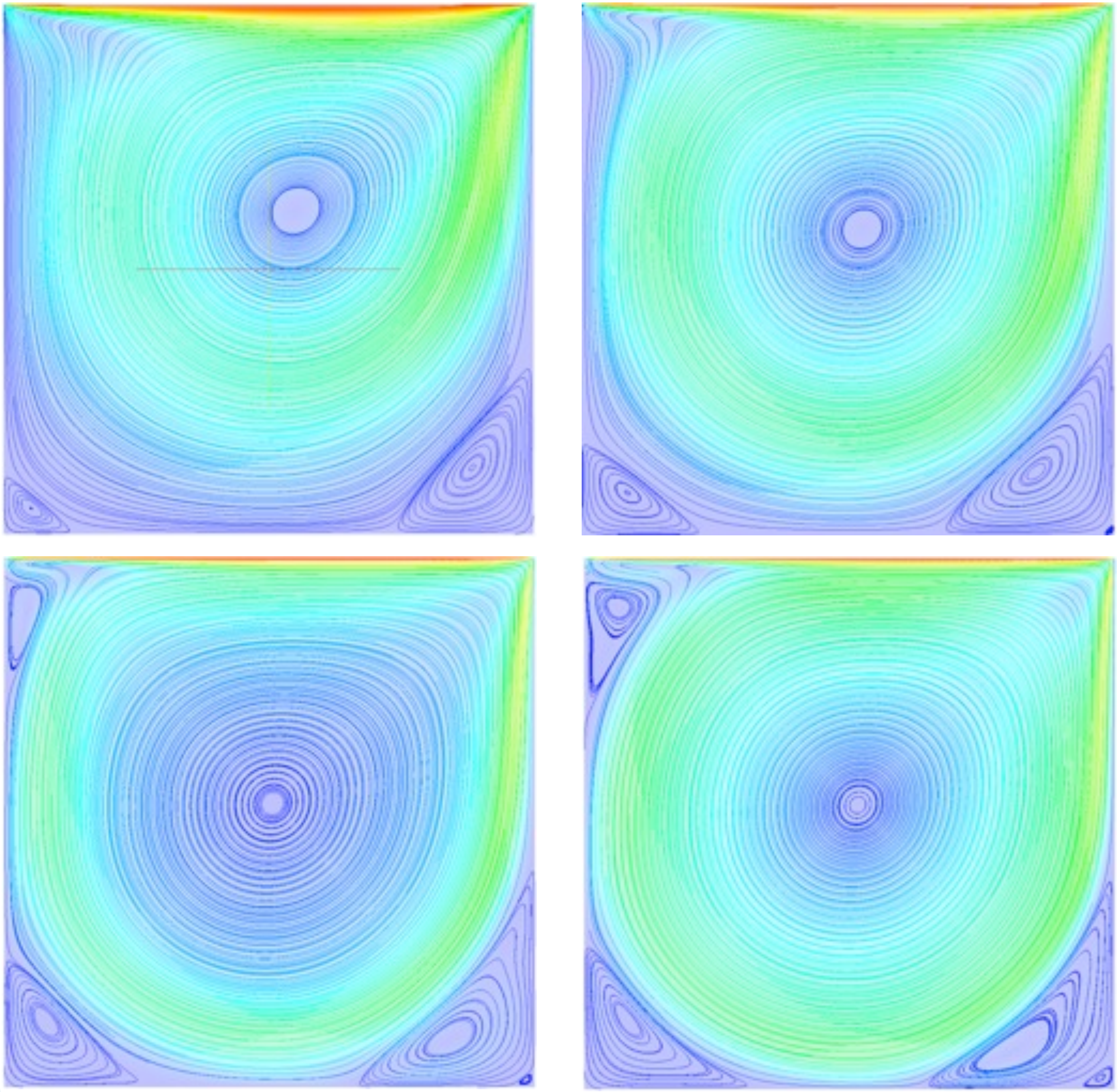


$$\begin{pmatrix} A + N(\mathbf{u}_k) & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta \mathbf{u} \\ \delta p \end{pmatrix}_k = -\mathbf{F}(\mathbf{x}_k)$$

$$\begin{pmatrix} A + N(\mathbf{u}_k) + D(\mathbf{u}_k) & B^t \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta \mathbf{u} \\ \delta p \end{pmatrix}_k = -\mathbf{F}(\mathbf{x}_k)$$

TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK PROGRAMME



TERRIFIC
Enhancing Interoperability

SEVENTH FRAMEWORK
PROGRAMME