Continuous and Discrete adjoint methodologies within ESI CFD solvers

Guillaume Pierrot
ESI Group

Agenda

- Rationale
- Continuous Adjoint Solver within PAM-FLOW
- Discrete Adjoint solver interfaced with CFD-ACE+
- Morphing

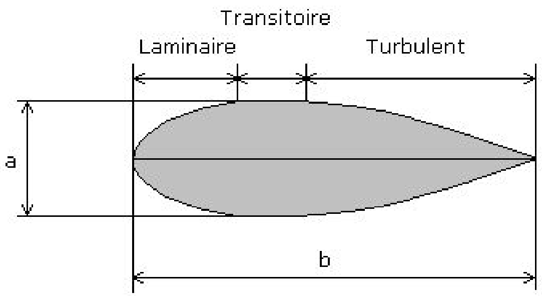
Rationale

- **Different problematics:**

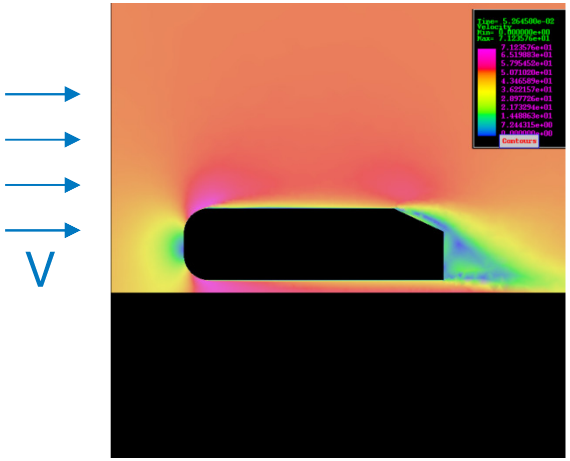
1.05

global optimization

- 2500%

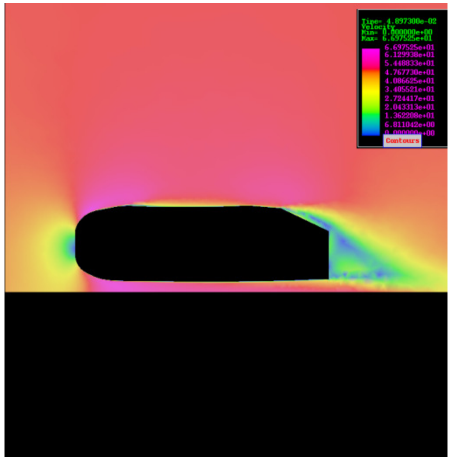


0.04



local improvement

- 17 %



Rationale

- **One would like to improve the performance of some given design wrt some criterion: the "cost function"**
  - may be lift, drag..
  - design parameters may be nodes coordinates, CAD parameters, level set position..

- **Standard numerical simulation provides a way to evaluate a design "a posteriori"**

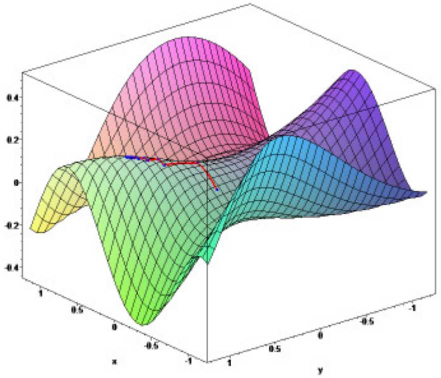
- **Optimal design tools automatically select the best (or at least a better) design wrt some given criterion**

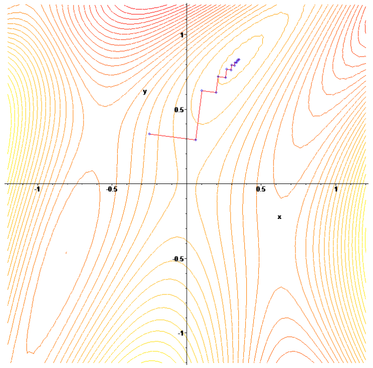
Rationale

- **Different optimization methods:**
  - *non-gradient based* (e.g. genetic algorithms): slow but may succed in finding a global optimum
  - *gradient-based*: quicker but stop as soon as a local optimum is found

locally best descent direction

$$F(x,y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right)\cos(2x + 1 - e^y)$$





Rationale

- **Different approaches for computing the gradient:**
  - *by Finite Differences → PAM-OPT*
  - *by using the adjoint state → PAM-FLOW Adjoint Solver, i-adjoint*

- **Both have their pro and cons**

Rationale

- **By Finite Differences:**
  - Flexible, black box tool: very generic, no knowledge on the underlying application solver is needed
  - But requires a number of runs proportional to the number of design parameters
  - Not sustainable when it tends to be large (e.g. free shape optimization)
  - In pratic, used together with a surrogate model (for CPU savings), which introduces further approximation and complexity

$$\nabla_\beta I \approx \left(\frac{\overset{\text{CFD run 1}}{I_{\beta_1+\delta\beta_1}} - I_{\beta_1}}{\delta\beta_1}, \frac{\overset{\text{CFD run 2}}{I_{\beta_2+\delta\beta_2}} - I_{\beta_2}}{\delta\beta_2}, \ldots, \frac{\overset{\text{CFD run n}}{I_{\beta_n+\delta\beta_n}} - I_{\beta_n}}{\delta\beta_n} \right)$$

Rationale

- **By using the adjoint state:**
  - Less generic: does require some knowledge of the underlying application code
  - More complicated → requires a dedicated tool, the so-called « adjoint solver »
  - But requires only one primal+one adjoint run → cost is independant on the number of design parameters
  - Well suited for shape optimization

Rationale

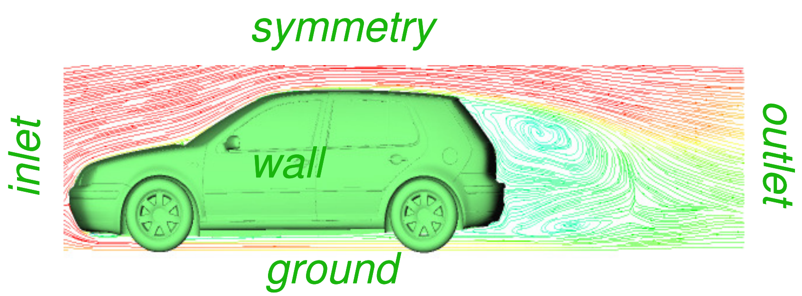
- **2 different approaches for computing the adjoint state:**
  - *Linearize/Dualize then Discretize → Continuous Adjoint Method*
  - *Discretize then Linearize/Dualize → Discrete Adjoint Method*

Rationale

- **ESI adjoint solutions:**
  - *Continuous adjoint solver embedded into PAM-FLOW (vertex-centered FV) (2006)*
  - *Discrete adjoint library interfaced with CFD-ACE+ (cell-centered FV,multiphysics) (2012)*

Continous adjoint solver

- **Incompressible Navier-Stokes:**



$$\begin{cases} \nabla\cdot\left\{\left(v\, v^{\alpha}\right)-\eta\nabla v^{\alpha}\right\}+\partial_{\alpha} p=0 \\ \nabla\cdot v=0 \end{cases}$$

mass & momentum conservation

$$\begin{cases} v=V \quad \text{on } \Gamma_{in} \\ v=0 \quad \text{on } \Gamma_{wall}\cup\Gamma_{ground} \\ p\,n-\eta\,\partial_{n} v=F \quad \text{on } \Gamma_{out} \\ v\cdot n=0\ ;\ \left(\partial_{n} v\right)\cdot t^{\alpha}=0\,,\ \alpha=1,2 \quad \text{on } \Gamma_{sym} \end{cases}$$

boundary conditions

Continous adjoint solver

- **Optimization set-up:**

$$\begin{cases} ! I(\Omega, v^{\alpha}, p, S^{\alpha}_{\beta}, \tau^{\alpha}_{\beta}) = \overbrace{\int_{\Omega} j\left(v^{\alpha}, p, S^{\alpha}_{\beta}\right)}^{\textit{volumic cost function}} + \overbrace{\int_{\partial\Omega} i\left(v^{\alpha}, \tau^{\alpha}_{\beta} n^{\beta}, n^{\alpha}\right)}^{\textit{surfacic cost function}} \\ \text{with } S^{\alpha}_{\beta} = \partial_{\beta} v^{\alpha} \ \text{ and } \ \tau^{\alpha}_{\beta} = p\, \delta_{\alpha\beta} - \eta\, \partial_{\beta} v^{\alpha} \\ s.t.\ \ \text{N - S equations + bcs} \end{cases}$$

Continous adjoint solver

- - - - - *flow eqs*

- - - - - *bcs*

- - - - - constitutive relations

- **KKT theory:**

$$\ell\left(\Omega, v^{\alpha}, p, S^{\alpha}_{\beta}, \tau^{\alpha}_{\beta}, w^{\alpha}, q, b_N, b_D, b^t_N, b^n_D, c^{\alpha}_{\beta}, d^{\alpha}_{\beta}\right) =$$

$$I\left(\Omega, v^{\alpha}, p, S^{\alpha}_{\beta}, \tau^{\alpha}_{\beta}, w^{\alpha}, q\right) + \int_{\Omega} \nabla\cdot\left\{\left(v\,v^{\alpha}\right) + \left(\tau^{\alpha}_{\beta}\right)_{\beta}\right\} w^{\alpha}$$

$$-\int_{\Omega} q\,\nabla\cdot v + \int_{\Gamma_{out}} \left\{F - \left(\tau^{\alpha}\cdot n\right)_{\alpha}\right\} b_N + \int_{\Gamma_{in}\cup\Gamma_{wall}\cup\Gamma_{ground}} \left(v - \delta_{in} V\right) b_D +$$

$$-\int_{\Gamma_{sym}} \left(\tau^{\alpha}\cdot n\right) t^{\alpha}\, b^t_N + \int_{\Gamma_{sym}} \left(v\cdot n\right) b^N_D + \int_{\Omega} \left\{\tau^{\alpha}_{\beta} - \left(p\,\delta_{\alpha\beta} - \eta\,\partial_{\beta} v^{\alpha}\right)\right\} c^{\alpha}_{\beta}$$

$$+\int_{\Omega} \left(S^{\alpha}_{\beta} - \partial_{\beta} v^{\alpha}\right) d^{\alpha}_{\beta}$$

Continous adjoint solver

- **KKT theory:**

$$\frac{\partial \ell\left(\Omega, v^{\alpha}, p, S^{\alpha}_{\beta}, \tau^{\alpha}_{\beta}, q, w^{\alpha}, b_D, b_N, b^{t}_{N}, b^{n}_{D}, c^{\alpha}_{\beta}, d^{\alpha}_{\beta}\right)}{\partial\left(w^{\alpha}, q, b_D, b_N, b^{t}_{N}, b^{n}_{D}, c^{\alpha}_{\beta}, d^{\alpha}_{\beta}\right)} = 0$$

→ *Flow equations + BCs*

$$\frac{\partial \ell\left(\Omega, v^{\alpha}, p, S^{\alpha}_{\beta}, \tau^{\alpha}_{\beta}, q, w^{\alpha}, b_D, b_N, b^{t}_{N}, b^{n}_{D}, c^{\alpha}_{\beta}, d^{\alpha}_{\beta}\right)}{\partial\left(v^{\alpha}, p, S^{\alpha}_{\beta}, \tau^{\alpha}_{\beta}\right)} = 0$$

→ *Adjoint equations + BCs*

$$\frac{\partial \ell\left(\Omega, v^{\alpha}, p, S^{\alpha}_{\beta}, \tau^{\alpha}_{\beta}, q, w^{\alpha}, b_D, b_N, b^{t}_{N}, b^{n}_{D}, c^{\alpha}_{\beta}, d^{\alpha}_{\beta}\right)}{\partial \Omega}$$

→ *Shape derivative (0 at optimum)*

Continous adjoint solver

- **Adjoint equations:**

$$\begin{cases} -v^{\beta}\,\partial_{\alpha}v_{*}^{\beta} - v\cdot\nabla v_{*}^{\alpha} - \nabla\cdot\left(\eta\,\nabla v_{*}^{\alpha}\right) + \nabla p_{*} = \left\{\partial_{\beta}\left(\dfrac{\partial j}{\partial S_{\beta}^{\alpha}}\right)\right\} - \dfrac{\partial j}{\partial v^{\alpha}} \\ \nabla\cdot v_{*} = \dfrac{\partial j}{\partial p} \end{cases}$$

cost function dependant source terms

Continous adjoint solver

- **Adjoint bcs:**

------ *cost function dependant source terms*

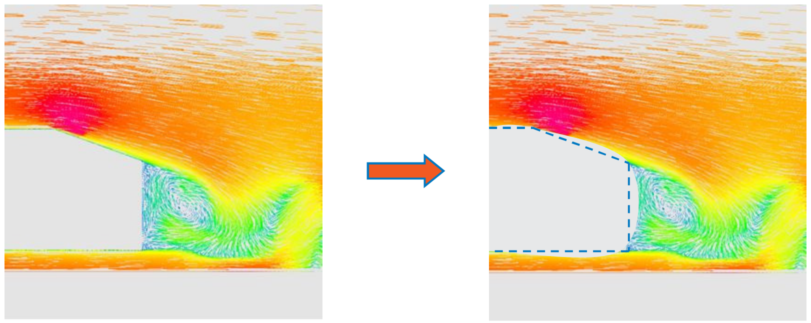
$$
\begin{cases}
v_* = -\dfrac{\partial i}{\partial\left(\tau^\alpha \cdot n\right)} \ \text{on}\ \Gamma_{in} \cup \Gamma_{wall} \cup \Gamma_{ground} \\
p_* n - \eta \partial_n v_* = \left(v \cdot v_*\right) n + \left(v \cdot n\right) v_* + \dfrac{\partial i}{\partial v} + \left(\dfrac{\partial j}{\partial S^\alpha_\beta} n^\beta\right)_\alpha \quad \text{on}\ \ \Gamma_{out} \\
v_* \cdot n = -\dfrac{\partial i}{\partial\left(\tau^\alpha \cdot n\right)} \cdot n \ ; \ \left(\eta\, \partial_n v_*\right) \cdot t^\alpha = -\dfrac{\partial i}{\partial v} \cdot t^\alpha - \left(\dfrac{\partial j}{\partial S^\alpha_\beta} n^\beta\right)_\alpha \cdot t^\alpha, \ \alpha = 1,2 \quad \text{on}\ \ \Gamma_{sym}
\end{cases}
$$

Continous adjoint solver

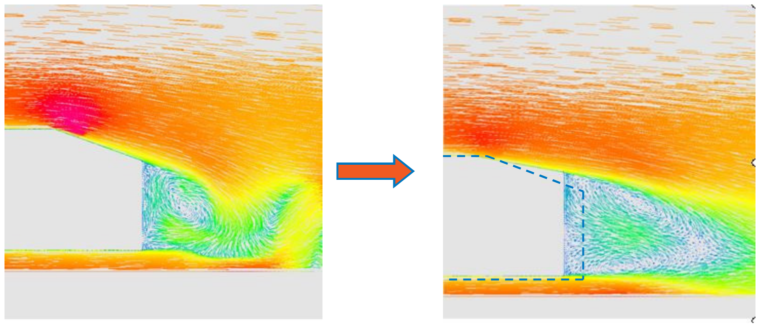
- **Shape derivative:** $\frac{\partial \ell}{\partial \Omega} = \frac{\partial I}{\partial \Omega} - \int_{\Gamma_{wall}} \left\{ \eta \left(\nabla v^{\alpha} \cdot \nabla v_{*}^{\alpha} \right) + S_{\beta}^{\alpha} \frac{\partial j}{\partial S_{\beta}^{\alpha}} \right\} (\varphi \cdot n)$

derivative wrt the profile

derivative wrt the physics



keep the flow / change the shape



keep the shape / change the flow

Continous adjoint solver

- **Application to Aero Force:**

cost function

$$I = \int_{\Gamma_{wall}} \{\, p\,(n\cdot d) - \eta\,\partial_n (v\cdot d)\,\} = \int_{\Gamma_{wall}} (\tau^\alpha \cdot n)_\beta \cdot d$$

$$\begin{cases} -v^\beta\,\partial_\alpha v_*^\beta - v\cdot\nabla v_*^\alpha - \nabla\cdot\left(\eta\,\nabla v_*^\alpha\right) + \nabla p_* = 0 \\ \nabla\cdot v_* = 0 \end{cases}$$

adjoint system

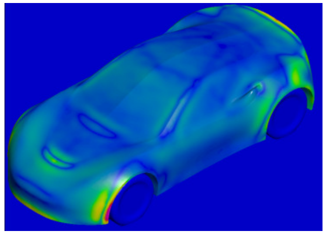
source term

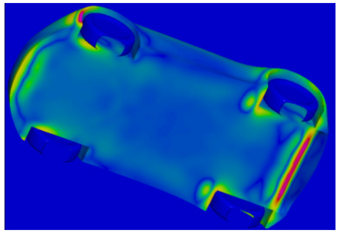
$$\begin{cases} v_* = -d & \text{on } \Gamma_{wall} \\ v_* = 0 & \text{on } \Gamma_{in} \cup \Gamma_{ground} \\ p_*\,n - \eta\,\partial_n v_* = (v\cdot v_*)\,n + (v\cdot n)\,v_* & \text{on } \Gamma_{out} \\ v_*\cdot n = 0\ ;\ (\eta\,\partial_n v_*)\cdot t^\alpha = 0,\ \alpha = 1,2 & \text{on } \Gamma_{sym} \end{cases}$$

adjoint bcs

$$\frac{\partial \ell}{\partial \Omega} = -\int_{\Gamma_{wall}} \eta\,\left(\nabla v^\alpha \cdot \nabla v_*^\alpha\right)(\varphi\cdot n)$$

shape derivative





Continous adjoint solver

cost function

- **Pressure Drop:** $I = +\int_{\Omega} \eta \left(\nabla v^{\alpha}\right)^2 = -\int_{\Gamma_{in}\cup\Gamma_{out}} \left\{ p + v^2/2 \right\} (v\cdot n) - \eta\,\partial_n v^2 = -\int_{\Gamma_{wall}} v^2/2\,(v\cdot n) + \left(\tau^{\alpha}\cdot n\right)_{\beta}\cdot v$

$$\begin{cases} -v^{\beta}\,\partial_{\alpha} v_{*}^{\beta} - v\cdot\nabla v_{*}^{\alpha} - \nabla\cdot\left(\eta\,\nabla v_{*}^{\alpha}\right) + \nabla p_{*} = 0 \\ \nabla\cdot v_{*} = 0 \end{cases}$$

adjoint system

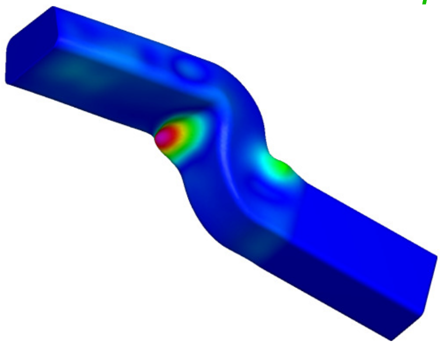
$$\frac{\partial \ell}{\partial \Omega} = -\int_{\Gamma_{wall}} \eta \left(\nabla v^{\alpha}\cdot\nabla v_{*}^{\alpha}\right)(\varphi\cdot n)$$

shape derivative

$$\begin{cases} v_{*} = 0 \quad \text{on } \Gamma_{wall} \\ v_{*} = -V \quad \text{on } \Gamma_{in} \\ p_{*}\,n - \eta\,\partial_n v_{*} = (v\cdot v_{*})\,n + (v\cdot n)\,v_{*} \\ \qquad - \left(v^2/2\right)n - v\,(v\cdot n) \\ \qquad - F \quad \text{on } \Gamma_{out} \end{cases}$$

source terms

adjoint bcs



Continous adjoint solver

Smagorinsky model

$$\begin{cases}\eta = \eta + \eta_t \\ \eta_t = g\left(S^\alpha_\beta\right)\end{cases}$$

- **Some complication: turbulent viscosity**

new variables

- **KKT:**

$$\ell\left(\Omega, v^\alpha, p, S^\alpha_\beta, \tau^\alpha_\beta, w^\alpha, q,\ b_N,\ b_D,\ b^t_N,\ b^n_D,\ c^\alpha_\beta, d^\alpha_\beta, \eta_t, \mu\right) =$$

$$\ell_0\left(\Omega, v^\alpha, p, S^\alpha_\beta, \tau^\alpha_\beta, w^\alpha, q,\ b_N,\ b_D,\ b^t_N,\ b^n_D,\ c^\alpha_\beta, d^\alpha_\beta\right)$$

$$+\int_\Omega \left\{ g\left(S^\alpha_\beta\right) - \eta_t \right\}\mu$$

turbulent lagrangian

standard lagrangian

get it right®

Continous adjoint solver

- **Modified adjoint equations:**

$$\begin{cases} -v^{\beta}\,\partial_{\alpha}v_{*}^{\beta} - v\cdot\nabla v_{*}^{\alpha} - \nabla\cdot\left(\eta\,\nabla v_{*}^{\alpha}\right) + \nabla p_{*} - \partial_{\beta}\left(\dfrac{\partial g}{\partial S_{\beta}^{\alpha}}\,\eta_{*}\right) = \left\{\partial_{\beta}\left(\dfrac{\partial j}{\partial S_{\beta}^{\alpha}}\right)\right\} - \dfrac{\partial j}{\partial v^{\alpha}} \\ \eta_{*} = \nabla v^{\alpha}\nabla v_{*}^{\alpha} + \dfrac{\partial j}{\partial \eta_{t}} \\ \nabla\cdot v_{*} = \dfrac{\partial j}{\partial p} \end{cases}$$

new term

volumetric cost function is allowed to depend on turbulent viscosity

Continous adjoint solver

- **Modified adjoint bcs:**

new terms

$$
\begin{cases}
v_* = -\dfrac{\partial i}{\partial\left(\tau^{\alpha}\cdot n\right)} \quad \text{on } \Gamma_{in} \cup \Gamma_{wall} \\[2ex]
p_* n - \eta\,\partial_n v_* = \left(v\cdot v_*\right) n + \left(v\cdot n\right) v_* + \dfrac{\partial i}{\partial v} + \left(\dfrac{\partial j}{\partial S^{\alpha}_{\beta}} n^{\beta}\right)_{\alpha} + \left(\dfrac{\partial g}{\partial S^{\alpha}_{\beta}} n^{\beta}\right)\eta_* \quad \text{on } \Gamma_{out} \\[2ex]
v_*\cdot n = -\dfrac{\partial i}{\partial\left(\tau^{\alpha}\cdot n\right)}\cdot n \;\; ; \;\; \left(\eta\,\partial_n v_*\right)\cdot t^{\alpha} = -\dfrac{\partial i}{\partial v}\cdot t^{\alpha} - \left(\dfrac{\partial j}{\partial S^{\alpha}_{\beta}} n^{\beta}\right)_{\alpha}\cdot t^{\alpha} + t^{\alpha}\cdot\left(\dfrac{\partial g}{\partial S^{\alpha}_{\beta}} n^{\beta}\right)\eta_*, \;\; \alpha = 1,2 \quad \text{on } \Gamma_{sym}
\end{cases}
$$

Continous adjoint solver

- **Modified shape derivative:**

new term

$$\frac{\partial \ell}{\partial \Omega} = \frac{\partial I}{\partial \Omega} - \int_{\Gamma_{wall}} \left\{ \tilde{\eta}\left(\nabla v^{\alpha} \cdot \nabla v_{*}^{\alpha}\right) + S_{\beta}^{\alpha} \left(\frac{\partial j}{\partial S_{\beta}^{\alpha}} + \frac{\partial j}{\partial \eta_{t}} \frac{\partial g}{\partial S_{\beta}^{\alpha}} \right) \right\} (\varphi \cdot n)$$

modified term

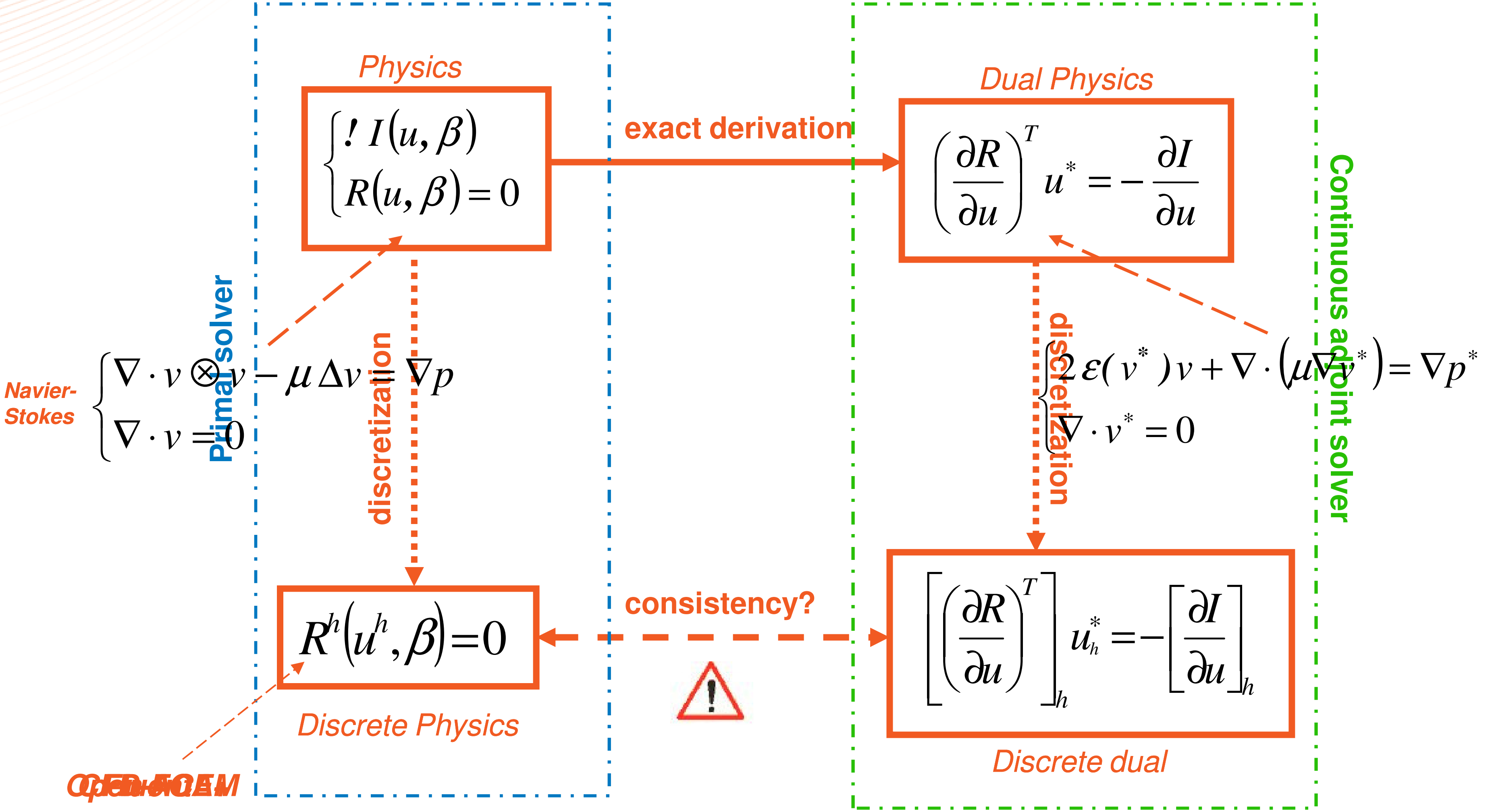
$$\tilde{\eta} = \eta + S_{\beta}^{\alpha} \frac{\partial g}{\partial S_{\beta}^{\alpha}}$$

linearized turbulent viscosity

Continous adjoint solver

- **More complications..**
  - turbulence models (k-epsilon,Spalart-Allmaras..)
  - law of the wall
  - natural convection (+Temperature equation)
  - MHD
  - Chemistry ..

Continous adjoint solver



$$\begin{cases} !\; I(u,\beta) \\ R(u,\beta)=0 \end{cases}$$

$$\left(\frac{\partial R}{\partial u}\right)^T u^* = -\frac{\partial I}{\partial u}$$

$$\begin{cases} \nabla\cdot v\otimes v-\mu\,\Delta v=\nabla p \\ \nabla\cdot v=0 \end{cases}$$

$$\begin{cases} 2\,\varepsilon(v^*)\,v+\nabla\cdot\left(\mu\nabla v^*\right)=\nabla p^* \\ \nabla\cdot v^*=0 \end{cases}$$

$$R^h\left(u^h,\beta\right)=0$$

$$\left[\left(\frac{\partial R}{\partial u}\right)^T\right]_h u_h^* = -\left[\frac{\partial I}{\partial u}\right]_h$$

Continuous adjoint solver

- **Pros:**
  - *Relatively easy-to-implement*
  - *Independant of the numerical scheme of the application code*
  - *CPU and memory efficient*

- **Cons:**
  - *Gradient inconsistency: the computed adjoint state is not the adjoint state of the computed physical field*
  - *Requires by hand differentiation of the underlying physical model → may be tricky (turbulence models..)*
  - *High cost maintenance: any model addendum in the primal solver requires a specific development effort counterpart in the adjoint solver*

Continuous adjoint solver

- **Enriched with converters in 2010 so that it can accomodate results of alternative CFD codes**
  - the CFD may be run using OpenFOAM or Star-CCM+ and the adjoint using PAM-FLOW

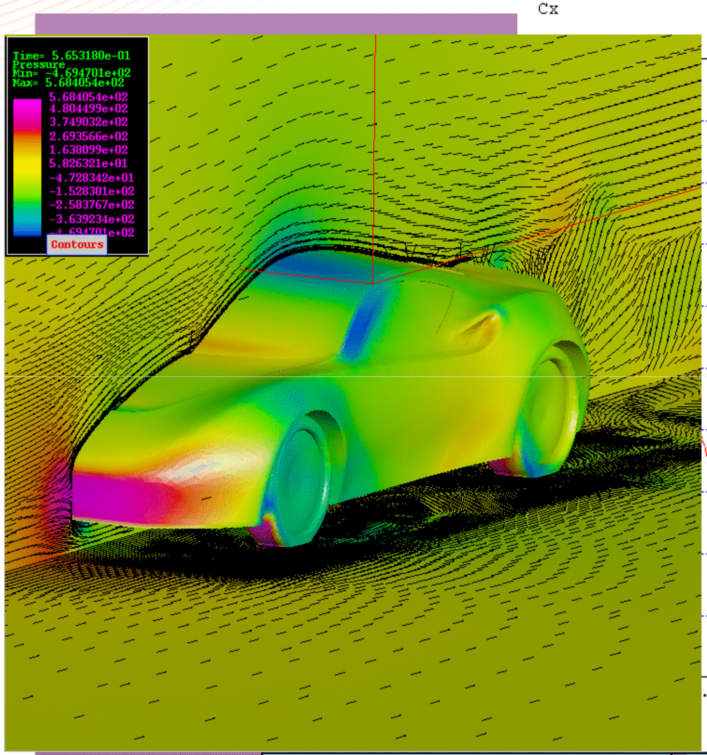
- **Both academic and industrial proof of value**

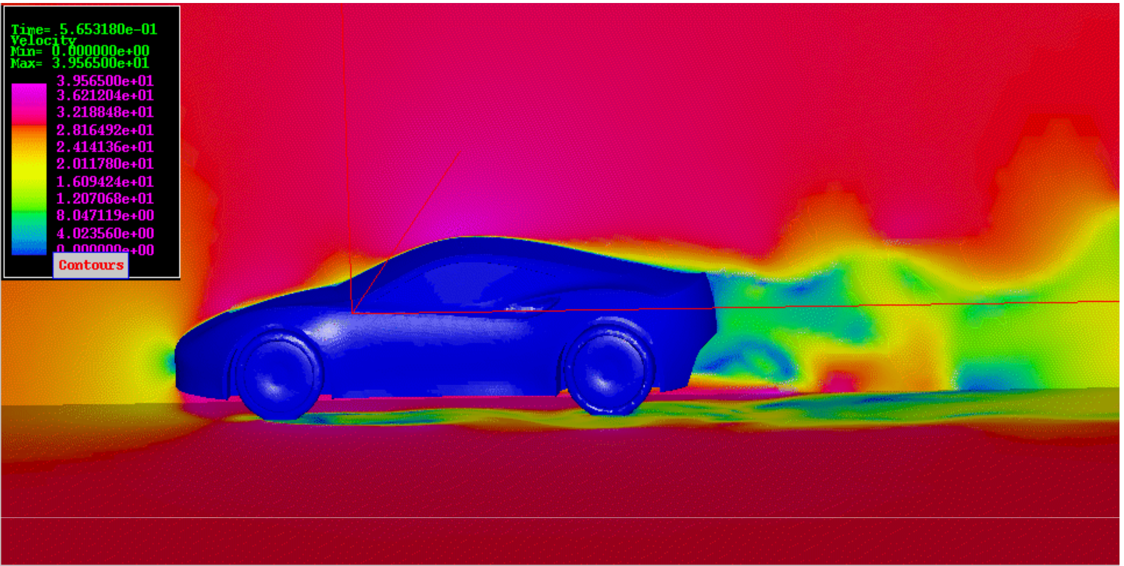
- **Integrated within ESI Visual process environment**

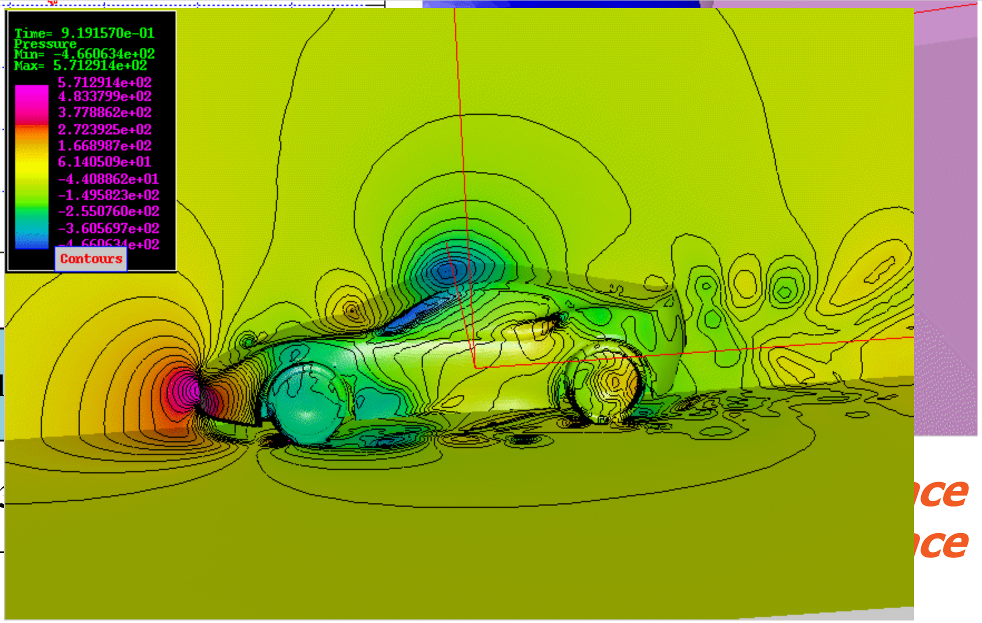
- **Only tet mesh**

Continuous adjoint solver

Drag reduct







Cx

.1

0.2

	Initial
Drag coefficient	0.

Predicted drag coefficient:

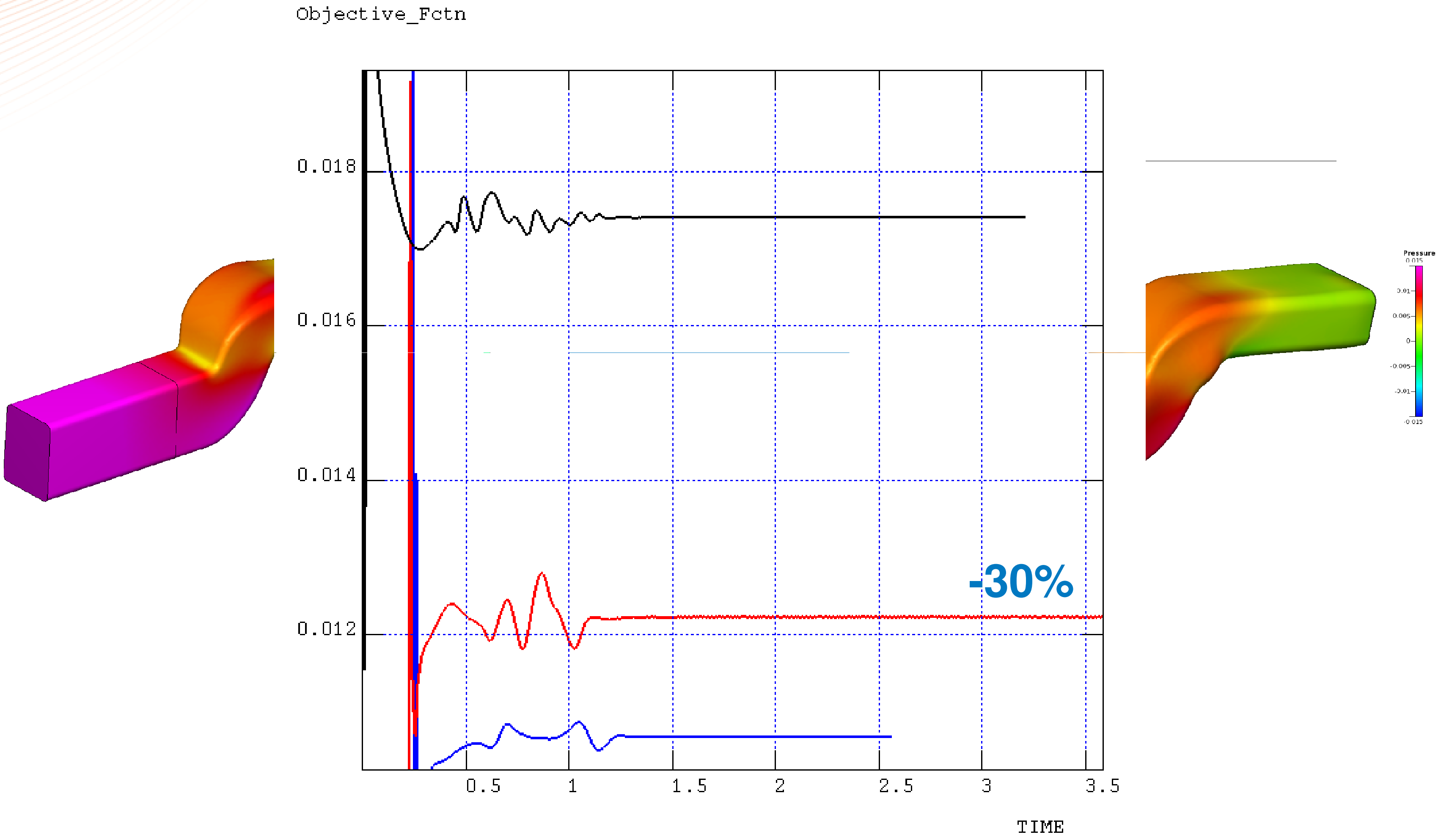
Cd = 0.342

ce

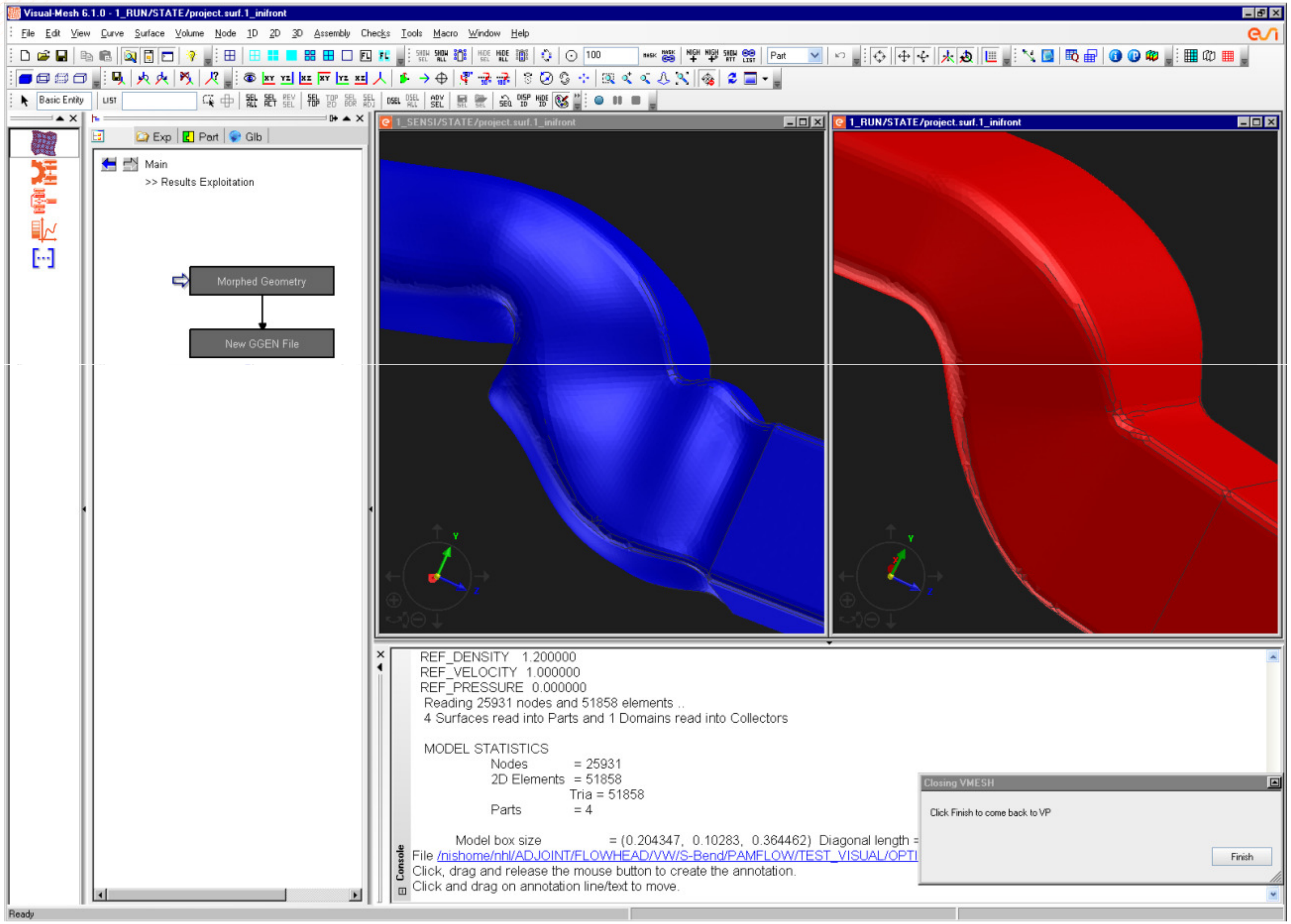
ce

www.esi-group.com

Continuous adjoint solver



Continuous adjoint solver


Visual-Mesh 6.1.0 - 1_RUN/STATE/project.surf.1_inifront
File Edit View Curve Surface Volume Node 1D 2D 3D Assembly Checks Tools Macro Window Help
Basic Entity
LIST
Exp
Part
Glb
Main
>> Results Exploitation
Morphed Geometry
New GGEN File
1_SENSI/STATE/project.surf.1_inifront
1_RUN/STATE/project.surf.1_inifront
Console
Closing VMESH
Finish
Ready

```
REF_DENSITY   1.200000
REF_VELOCITY  1.000000
REF_PRESSURE  0.000000
 Reading 25931 nodes and 51858 elements ..
 4 Surfaces read into Parts and 1 Domains read into Collectors

 MODEL STATISTICS
               Nodes          = 25931
               2D Elements  = 51858
                               Tria = 51858
               Parts           = 4

        Model box size              = (0.204347,  0.10283,  0.364462)  Diagonal length =
File /nishome/nhl/ADJOINT/FLOWHEAD/VW/S-Bend/PAMFLOW/TEST_VISUAL/OPTI
Click, drag and release the mouse button to create the annotation.
Click and drag on annotation line/text to move.
```

Click Finish to come back to VP

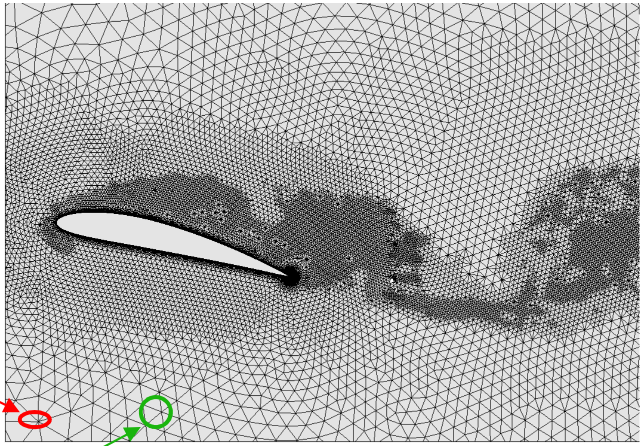
Discrete adjoint solver

- **Optimization set-up:**

$$\begin{cases} ! \; I(x,u) \\ s.t. \quad R(x,u)=0 \end{cases}$$

mesh nodes coordinates

DOF vector: (v1,v2,v3,p)



- **KKT theory:** $\ell(x,u,u^*) = I(\,x,u\,) + (u^*)^T \; R(x,u)$

get it right®

Discrete adjoint solver

- **KKT theory:**

$$\frac{\partial \ell\left(x,u,u^{*}\right)}{\partial u}=\frac{\partial I(\,x,u\,)}{\partial u}+\left(\frac{\partial R\left(x,u\right)}{\partial u}\right)^{T}u^{*}=0 \quad \longrightarrow$$

discrete adjoint equation

$$\frac{\partial \ell\left(x,u,u^{*}\right)}{\partial x}=\frac{\partial I(\,x,u\,)}{\partial x}+\left(\frac{\partial R\left(x,u\right)}{\partial x}\right)^{T}u^{*} \quad \longrightarrow$$

sensitivity vector

Discrete adjoint solver

- **Discrete vs. Continuous shape derivative:**

discrete

$$\frac{\partial \ell}{\partial x} = \left(\frac{\partial I}{\partial x}\right)^T + \left(\frac{\partial R}{\partial x}\right)^T u^*$$

$$\Big\downarrow \ h \to 0$$

BUT:

$$\left(\frac{\partial I}{\partial x}\right)^T \neq \frac{\partial I}{\partial \Omega}$$

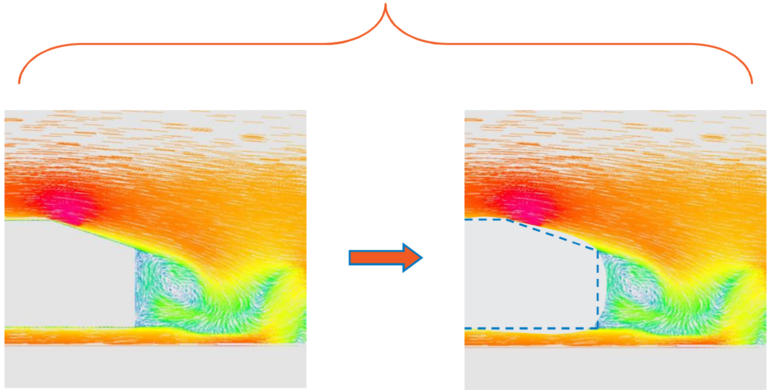
continuous

$$\frac{\partial \ell}{\partial \Omega} = \frac{\partial I}{\partial \Omega} - \int_{\Gamma_{wall}} \left\{ \eta \left(\nabla v^{\alpha} \cdot \nabla v_*^{\alpha} \right) - S_{\beta}^{\alpha} \frac{\partial j}{\partial S_{\beta}^{\alpha}} \right\} (\varphi \cdot n)$$

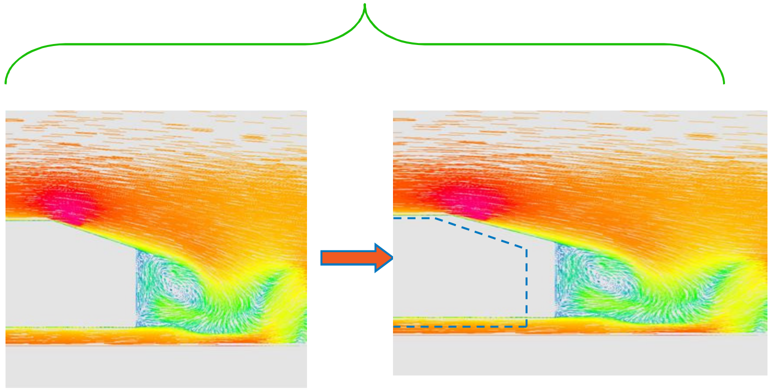
Discrete adjoint solver

- **Indeed:** $\boxed{\dfrac{\partial I}{\partial \Omega}}$ *Lie derivative*

$\boxed{\left(\dfrac{\partial I}{\partial x}\right)^T}$ *Sensitivity to the node positions*



keep the flow / change the shape



advect the flow as shape changes

- **They are connected:**

$$\boxed{\left(\frac{\partial I}{\partial x}\right)^T \rightarrow \frac{\partial I}{\partial \Omega} - \left(\frac{\partial I}{\partial u}\right)(\varphi \cdot \nabla u) - \left(\frac{\partial I}{\partial U_{wall}}\right)(\varphi \cdot \nabla U)}$$

Discrete adjoint solver

- **Similarily:**

$$\boxed{\left(\frac{\partial R}{\partial x}\right)^T u^* \rightarrow \frac{\partial\left(R, u^*\right)}{\partial\Omega} - \left(\left(\frac{\partial R}{\partial u}\right)^T u^*, (\varphi\cdot\nabla u)\right) - \left(\left(\frac{\partial R}{\partial U_{wall}}\right)^T u^*, (\varphi\cdot\nabla U)\right)}$$

- **Hence:**

$$\frac{\partial\ell}{\partial x} \rightarrow \frac{\partial I}{\partial\Omega} - \left(\left(\frac{\partial R}{\partial U_{wall}}\right)^T u^* + \left(\frac{\partial I}{\partial U_{wall}}\right)^T, (\varphi\cdot\nabla U)\right)$$

$$+ \underbrace{\frac{\partial\left(R, u^*\right)}{\partial\Omega}}_{\rightarrow 0} - \left(\underbrace{\left\{\left(\frac{\partial R}{\partial u}\right)^T u^* + \left(\frac{\partial I}{\partial u}\right)^T\right\}}_{0}, (\varphi\cdot\nabla u)\right)$$

- **Finally:**

$$\boxed{\frac{\partial\ell}{\partial x} \approx \frac{\partial I}{\partial\Omega} - \left(\left(\frac{\partial R}{\partial U_{wall}}\right)^T u^* + \left(\frac{\partial I}{\partial U_{wall}}\right)^T, (\varphi\cdot\nabla U)\right)}$$

Discrete adjoint solver

- **3 choices for computing the shape derivative:**

discrete

$$\frac{\partial \ell}{\partial x} = \left(\frac{\partial I}{\partial x}\right)^T + \left(\frac{\partial R}{\partial x}\right)^T u^*$$

→ **exact**

continuous

$$\frac{\partial \ell}{\partial \Omega} = \frac{\partial I}{\partial \Omega} - \int_{\Gamma_{wall}} \left\{ \eta \left(\nabla v^{\alpha} \cdot \nabla v_{*}^{\alpha} \right) - S_{\beta}^{\alpha} \frac{\partial j}{\partial S_{\beta}^{\alpha}} \right\} (\varphi \cdot n)$$

→ **valid at convergence**

hybrid

$$\frac{\partial \ell}{\partial x} \approx \frac{\partial I}{\partial \Omega} - \left(\left(\frac{\partial R}{\partial U_{wall}}\right)^T u^* + \left(\frac{\partial I}{\partial U_{wall}}\right)^T , (\varphi \cdot \nabla U) \right)$$

→ **good (?) compromise if one can not easily move the nodes**

Discrete adjoint solver

- **The adjoint matrix:**

$$\left(\begin{array}{ccc|c}
\frac{\partial R_{v^1}}{\partial v^1} & \frac{\partial R_{v^2}}{\partial v^1} & \frac{\partial R_{v^3}}{\partial v^1} & \frac{\partial R_p}{\partial v^1} \\
\frac{\partial R_{v^1}}{\partial v^2} & \frac{\partial R_{v^2}}{\partial v^2} & \frac{\partial R_{v^3}}{\partial v^2} & \frac{\partial R_p}{\partial v^2} \\
\frac{\partial R_{v^1}}{\partial v^3} & \frac{\partial R_{v^2}}{\partial v^3} & \frac{\partial R_{v^3}}{\partial v^3} & \frac{\partial R_p}{\partial v^3} \\
\hline
\frac{\partial R_{v^1}}{\partial p} & \frac{\partial R_{v^2}}{\partial p} & \frac{\partial R_{v^3}}{\partial p} & \frac{\partial R_p}{\partial p}
\end{array}\right) = \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{B'} & \mathbf{C} \end{array}\right)$$

- **Typical block stencil:**

	2	
2	1	2
1	0	1
2	1	2
	2	

- **Huge, (not so) sparse, unsymmetric, undefinite matrix**

- **Hard to solve (saddle-point system)**

Discrete adjoint solver

- Use an optimal preconditioner (DDM, Multigrid, Subdomain deflation)

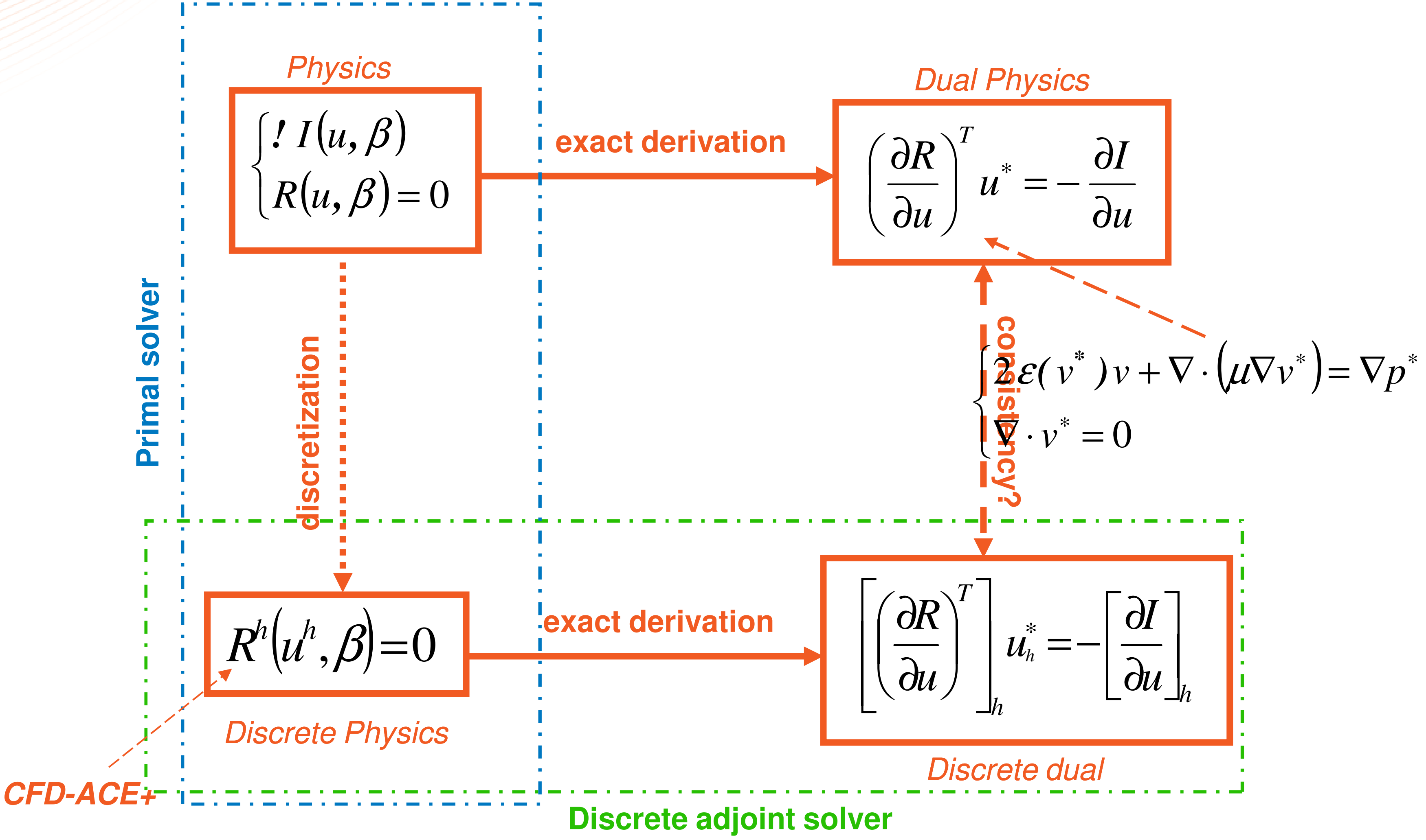
- Or a segregated-like approach: $P\left(x^{n+1}-x^{n}\right)=\left(\frac{\partial I}{\partial u}\right)^{T}-\left(\frac{\partial R}{\partial u}\right)^{T}x^{n}$

$$P^{-1}=\begin{pmatrix} I & -D_A^{-1}B \\ 0 & I \end{pmatrix}\begin{pmatrix} I & 0 \\ 0 & \left(B'D_A^{-1}B-C\right)^{-1} \end{pmatrix}\begin{pmatrix} I & 0 \\ B' & -I \end{pmatrix}\begin{pmatrix} \left(D_A+A\right)^{-1} & 0 \\ 0 & I \end{pmatrix}$$

$$D_A=\alpha\ \mathit{DIAG}\left(A\right)$$

- If matrix has to be assembled, out of coring may be needed

Discrete adjoint solver



$$\begin{cases} ! \, I(u,\beta) \\ R(u,\beta)=0 \end{cases}$$

$$\left(\frac{\partial R}{\partial u}\right)^T u^* = -\frac{\partial I}{\partial u}$$

$$\begin{cases} 2\varepsilon(v^*)v + \nabla\cdot\left(\mu\nabla v^*\right) = \nabla p^* \\ \nabla\cdot v^* = 0 \end{cases}$$

$$R^h\left(u^h,\beta\right)=0$$

$$\left[\left(\frac{\partial R}{\partial u}\right)^T\right]_h u_h^* = -\left[\frac{\partial I}{\partial u}\right]_h$$

Discrete adjoint solver

- **Pros:**
  - *Gradient consistency: the computed adjoint state is the real adjoint of the computed physical field*

- **Cons:**
  - *Depends on the very numerical scheme of the application code*
  - *How to build the discrete adjoint operator?*

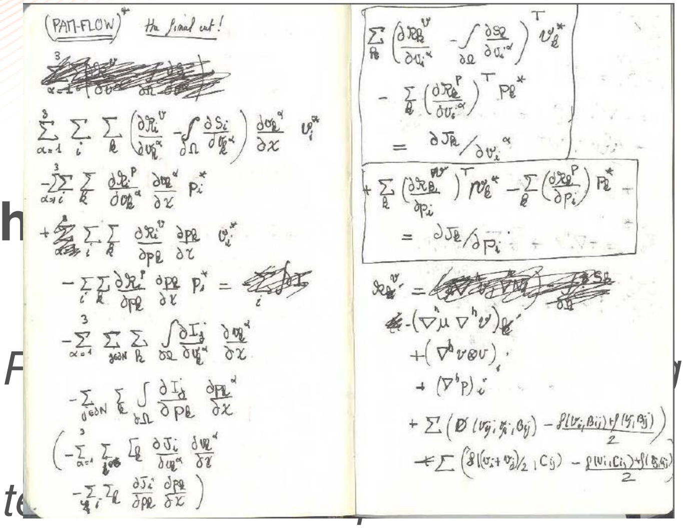
Discrete adjoint solver

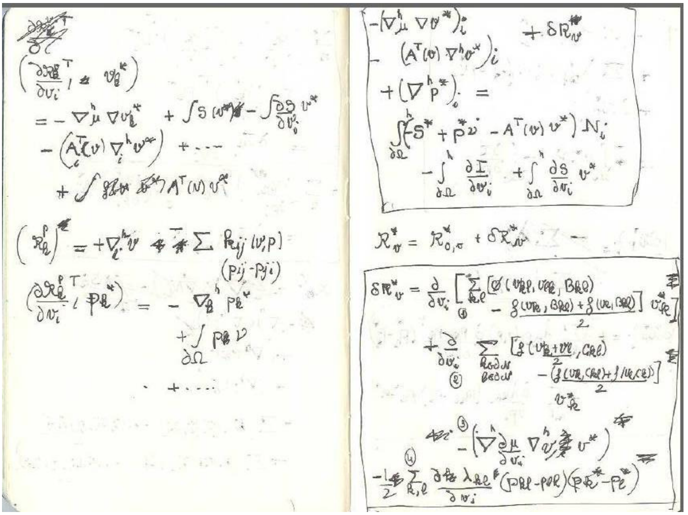
- **Different approaches for building the discrete adjoint operator:**
  - *By hand*
  - *Automatic (Algorithmic) Differentiation*

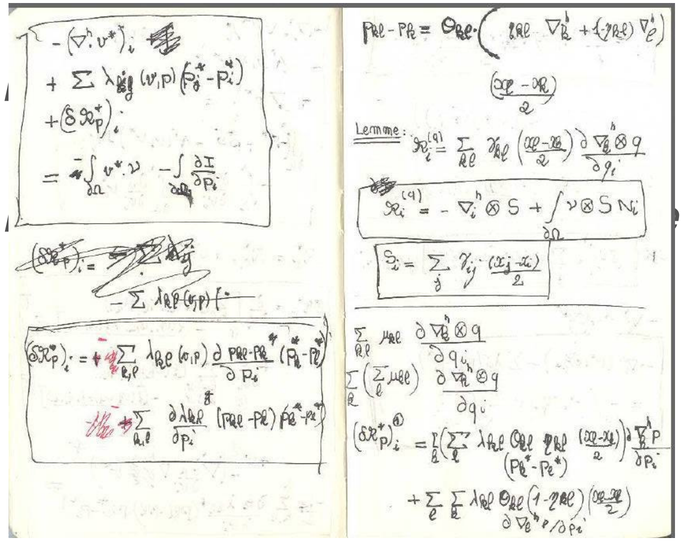
Discrete adjoint solver

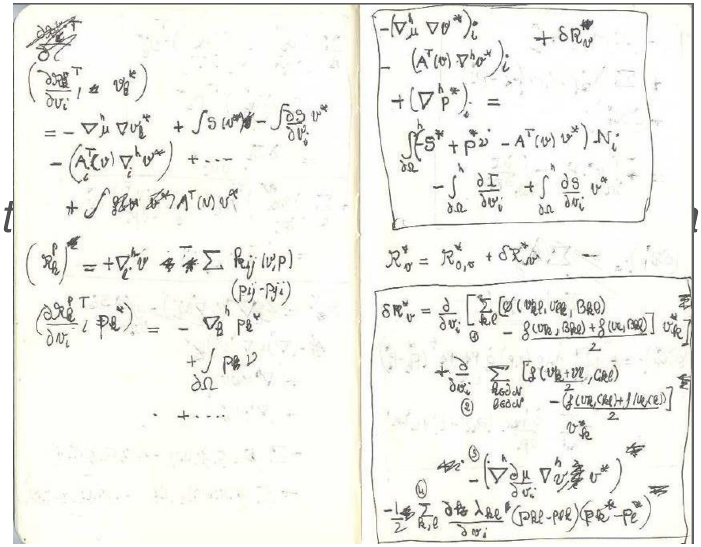
- **By h**

  - *F* … *ge of*
  - *te*
  - …
  - … *erivat* … *ntly*









Discrete adjoint solver

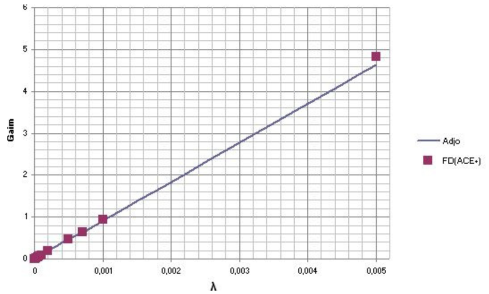
- **By Algorithmic Differentiation:**
  - *The source code istelf is differentiated*
  - *2 modes: direct / reverse*
  - *2 approaches: source transformation / operator overloading*

- **Pros & cons:**
  - *Low cost maintenance: the code is differentiated once and for all (no further effort for accomodating newly developped models in the primal solver)*
  - *But each application solver adjoint derivation has to be adressed mostly independantly*
  - *May turn to be tedious and time consuming depending on the code structure and programming language*
  - *Very invasive: requires full access to the source code*
  - *Severe CPU efficiency and memory consumption challenges*

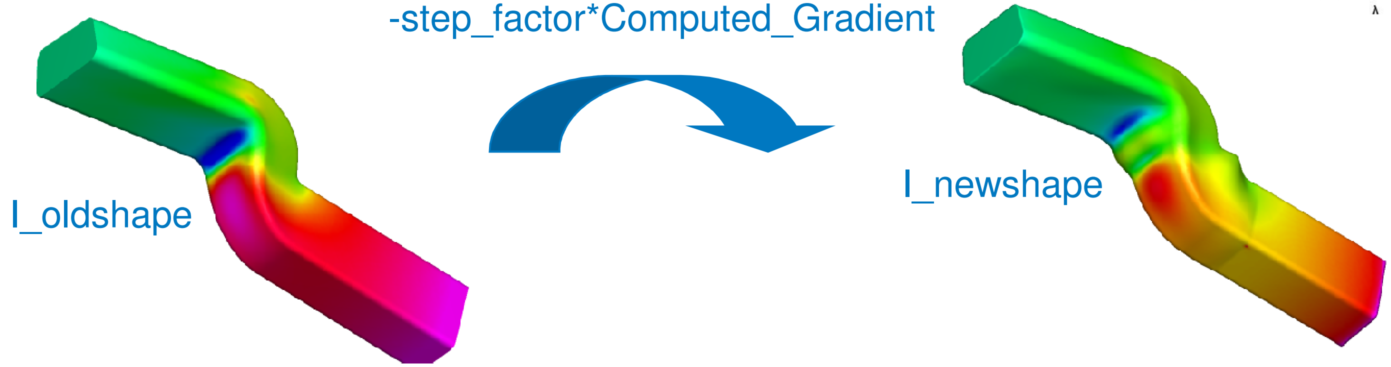
Discrete adjoint solver

- Academic validation against FD
- Dynamic library interfaced with CFD-ACE+
- Any mesh topology
- Enriched with converters so that it can accomodate results of alternative CFD codes
  - the CFD may be run using OpenFOAM or Star-CCM+ and the adjoint using PAM-FLOW
- Limitations:
  - Not yet parallelized

Discrete adjoint solver



- **A word on rigorous validation:**



-step_factor*Computed_Gradient

I_oldshape

I_newshape

Expected Improvement := step_factor*Computed_Gradient_Norm**2

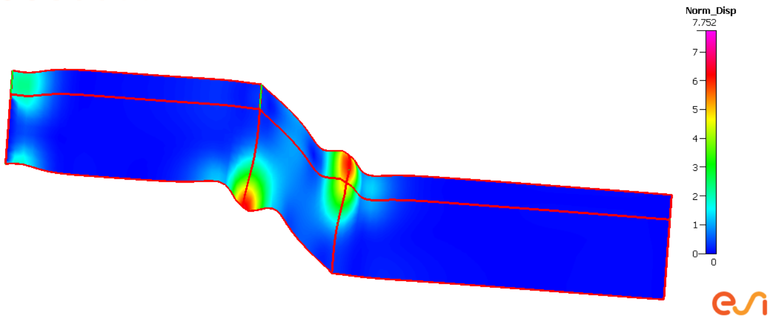
Effective Improvement := I_oldshape-I_newshape

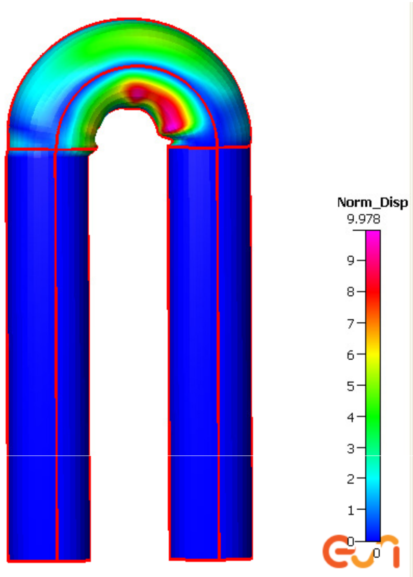
Relative error :=100* ABS(Expected_Improvement-Effective_Improvement)
/Effective_Improvement

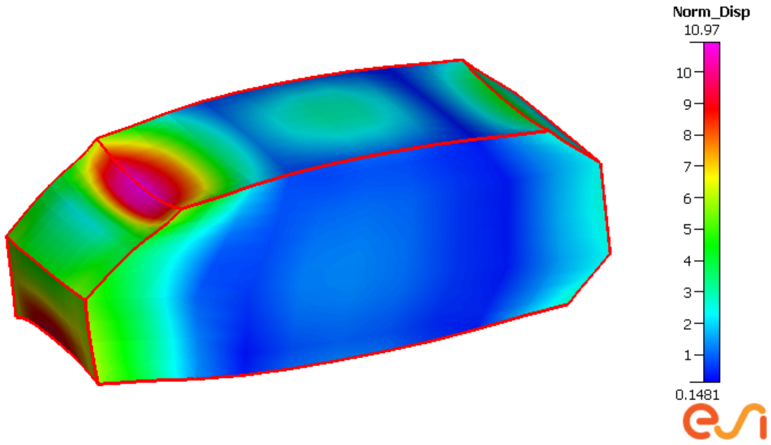
Discrete adjoint solver

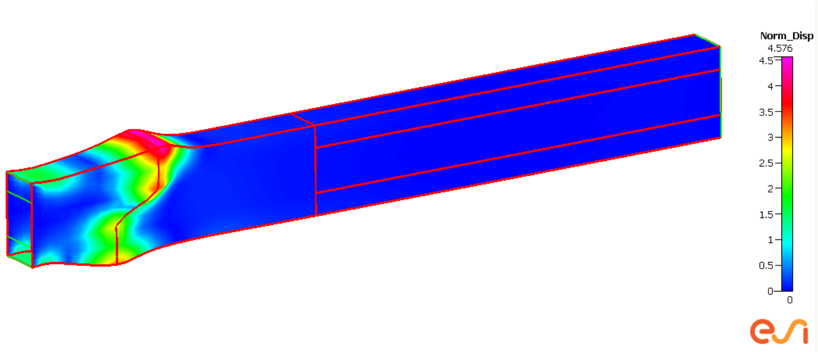
	Airfoil Tet	Airfoil Hex	Aifoil Hex	S-Bend (Viscart)	Ahmed Body (Viscart)
Physics	Laminar	Laminar	Frozen Turbulent	Frozen Turbulent	Frozen Turbulent
Relative error (%)	0,07	0,14	0,37	5,15	0,11

Discrete adjoint solver









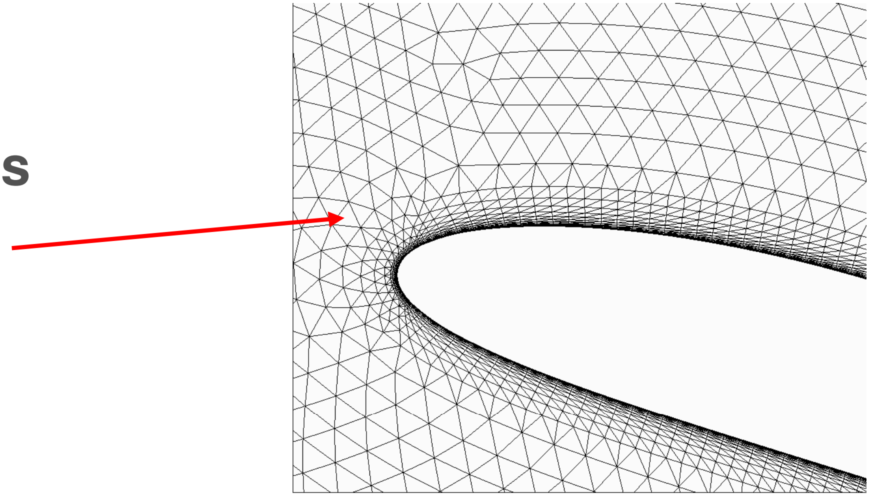
Morphing

- **At first bound interior node displacement to sruface nodes one thanks to harmonic mapping (ALE-like):**
  - **Minimizes mesh isotropic distortion**

$$\begin{cases} -\Delta\,\delta x = 0 \\ \delta x = \delta x_{surf} \quad \text{on } \Gamma_{wall} \\ \delta x = 0 \quad \text{on } \partial\Omega \setminus \Gamma_{wall} \end{cases} \quad \Longrightarrow \quad \delta x_{\boldsymbol{int}} = K\,\delta x_{surf}$$

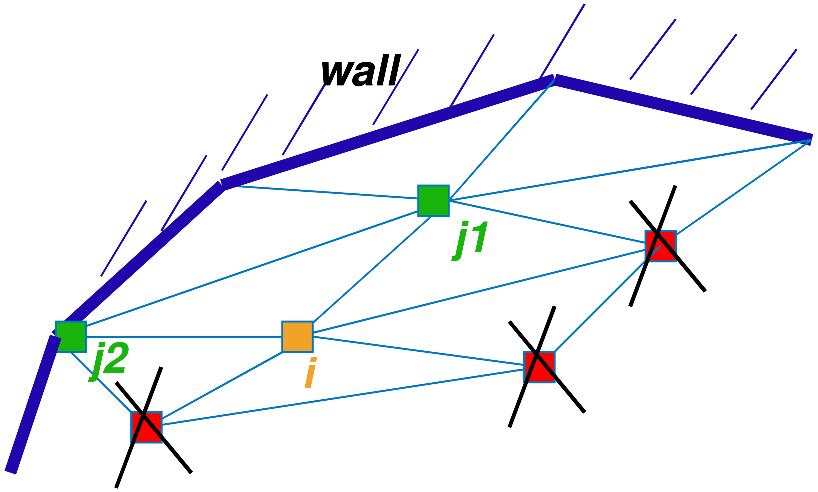
Morphing

- **Not good for boundary layers (anisotropic mesh)**



  - **Use rigidification instead :**

$$\delta x_{i\in bl(n)} = \frac{\sum_{j\in N_i \cap bl(n-1)} \left(1/\left\|x_j - x_i\right\|\right)\delta x_j}{\sum_{j\in N_i \cap bl(n-1)} \left(1/\left\|x_j - x_i\right\|\right)}$$

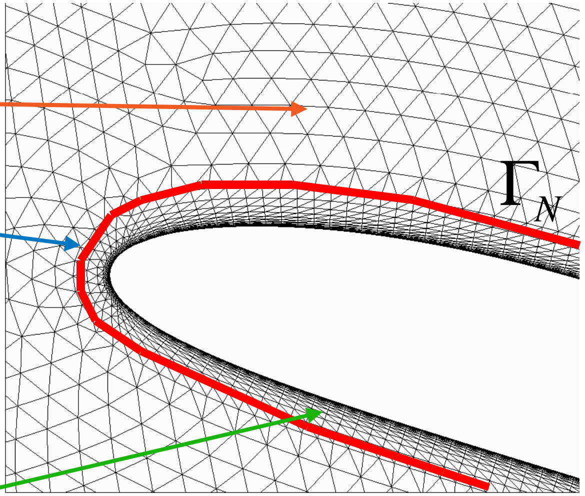


Morphing

- **Then combine Laplace and rigidification:**

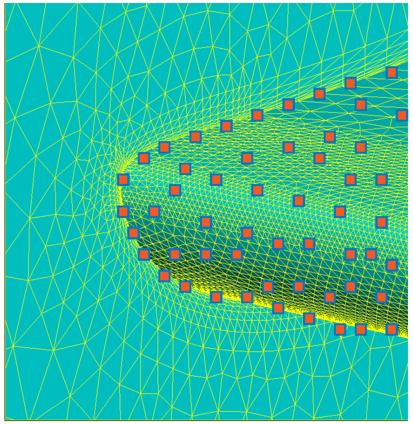
$$\begin{cases} -\Delta\,\delta x = 0 \\ \delta x = \delta x_{bl(N)} \quad \text{on } \Gamma_N \\ \delta x = 0 \quad \text{on } \partial\Omega \setminus \Gamma_{wall} \end{cases}$$

$$\delta x_{i\in bl(n\leq N)} = \frac{\sum_{j\in N_i \cap bl(n-1)} \left(1/\left\|x_j - x_i\right\|\right)\delta x_j}{\sum_{j\in N_i \cap bl(n-1)} \left(1/\left\|x_j - x_i\right\|\right)}$$



Morphing

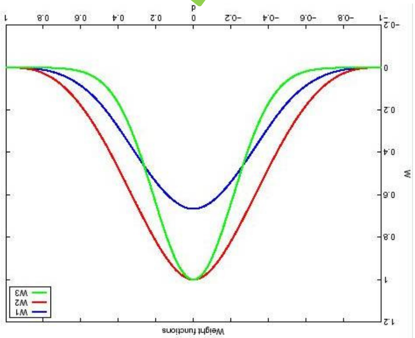
- **For boundary nodes, apply LSQ morphing:**
  - Sample the surface nodes $\left(\delta x^i_{surf}\right) \rightarrow \left(\delta y^j_s\right)$
  - Apply LSQ operator (exact for polynomial up to desired degree)



$$\delta x_{surf}(x) = \sum_{j \in S} \delta y^j_s \, \Phi_j(x)$$

$$\left(\Phi_i(x)\right)_{1 \le i \le n} = \text{argmin}\left(J(\lambda) = \sum_{i \in S} W_\varepsilon(x - x_i)\, \lambda_i^2 \right)$$

$$\text{subject to} \quad \sum_{i\,node} \lambda_i p_k(x_i) = p_k(x) \quad \forall p_k$$



Morphing

- **Finally, adapt the shape derivative accordingly via chain rule:**

  :

$$\boxed{\frac{\partial \ell}{\partial y_s} = \left\{ \left(\frac{\partial \ell}{\partial x_{in}}\right)\left(\frac{\partial x_{in}}{\partial x_{surf}}\right) + \left(\frac{\partial \ell}{\partial x_{surf}}\right) \right\} \left(\frac{\partial x_{surf}}{\partial y_s}\right)}$$

- **Do not forget this step, otherwise the gradient is wrong !**

  :

Morphing

- Morphing option available both within PAM-FLOW Continuous Adjoint Solver and ACE+ Discrete one

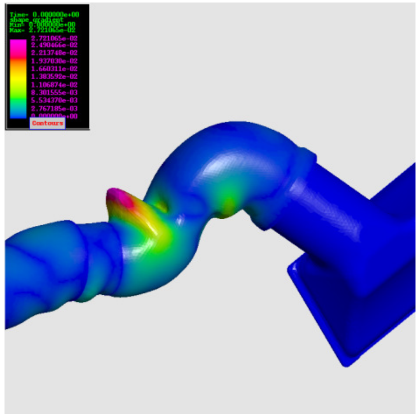
  :

- Limited to small displacements

  :

- Additionaly, the tool provides the value of the maximal step factor so that all volume remain positive

  :



get it right®

www.esi-group.com