NATIONAL TECHNICAL UNIVERSITY OF ATHENS
Parallel CFD & Optimization Unit
Laboratory of Thermal Turbomachines

The Continuous Adjoint Method on Graphics Processing Units (GPUs) For Compressible Flows

OPTi 2014

Mehdi Ghavami Nejad
K. Tsiakas
V. Asouti
X. Trompoukis

Goals

- Using continuous adjoint method with the aim of computing the objective function gradients with respect to the design variables for compressible flow and unstructured grid(2D&3D).

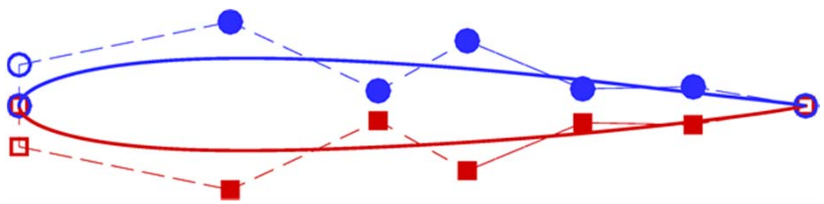
- Implementing the code on Graphics Processing Units (GPUs) for taking advantage of the inherent massive parallel processors.

- Objective function as the projected aerodynamic force at the direction $\boldsymbol{r_k}$.

Objective Function $\boldsymbol{F} = \int_{S_w} \boldsymbol{P}\, \boldsymbol{n_k r_k dS} - \int_{S_w} \tau_{km}\, \boldsymbol{n_m r_k dS}$

Method : Continuous Adjoint

- **Parameterization of the aerodynamic shape by Bezier control points as the design variables**



- **Defining the augmented objective function as :**

$$\boldsymbol{F_{aug}} = \boldsymbol{F} + \int_{\Omega} \Psi_{\boldsymbol{n}} \boldsymbol{R_n} \boldsymbol{d}\Omega$$

$$\boldsymbol{R_n} = \boldsymbol{0} \quad \Longrightarrow \quad \frac{\delta \boldsymbol{F_{aug}}}{\delta \boldsymbol{b_i}} = \frac{\delta \boldsymbol{F}}{\delta \boldsymbol{b_i}} + \int_{\Omega} \Psi_{\boldsymbol{n}} \frac{\partial \boldsymbol{R_n}}{\partial \boldsymbol{b_i}} \boldsymbol{d}\Omega$$

Method : Continuous Adjoint

$$\int_{\Omega} \Psi_n \frac{\partial R_n}{\partial b_i} d\Omega = \int_S \Psi_n \frac{\partial f_{nk}^{inv}}{\partial b_i} n_k dS - \int_{\Omega} A_{nmk} \frac{\partial \Psi_n}{\partial x_k} \frac{\partial U_m}{\partial b_i} d\Omega - \int_S \Psi_n \frac{\partial f_{nk}^{vis}}{\partial b_i} n_k dS + \int_{\Omega} \frac{\partial \Psi_n}{\partial x_k} \frac{\partial f_{nk}^{vis}}{\partial b_i} d\Omega$$

Field adjoint equation : $\frac{\partial \Psi_m}{\partial t} - A_{nmk} \frac{\partial \Psi_n}{\partial x_k} - K_k \frac{\partial V_k}{\partial U_m} = 0$

A : Jacobian flux
U : Conservative flow variables
V : Primitive flow variables

$$K_n = \begin{bmatrix} -\frac{T}{\rho} \frac{\partial}{\partial x_k}\left(k \frac{\partial \Psi_5}{\partial x_k}\right) \\ \frac{\partial \tau_{(n-1)m}^{adj}}{\partial x_m} - \tau_{(n-1)m} \frac{\partial \Psi_5}{\partial x_m} \\ \frac{T}{P} \frac{\partial}{\partial x_k}\left(k \frac{\partial \Psi_5}{\partial x_k}\right) \end{bmatrix}$$

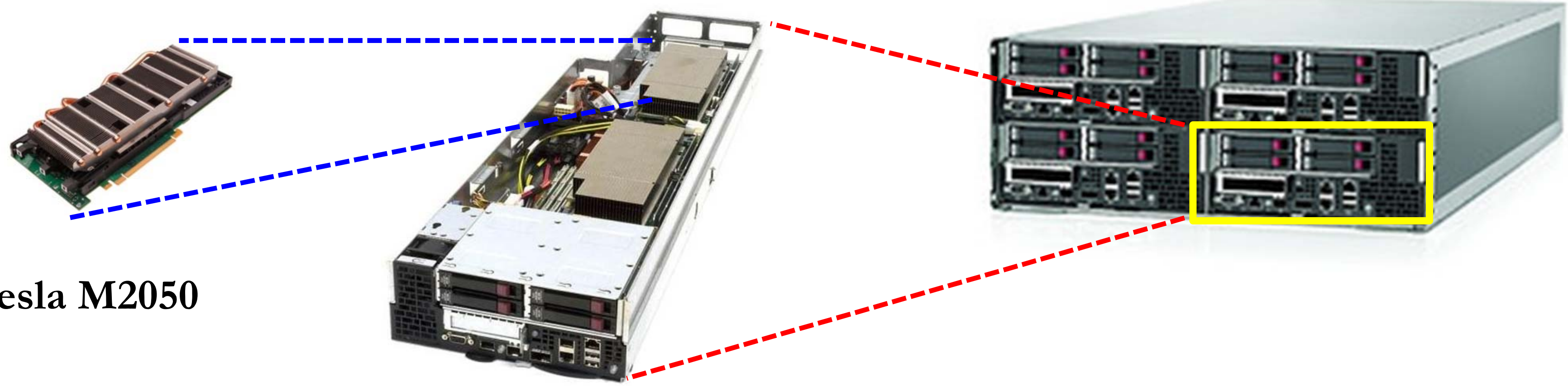
Sensitivity Derivatives

$$\frac{\delta \boldsymbol{F_{aug}}}{\delta \boldsymbol{b_i}} = \int_{S_w} \boldsymbol{P}\frac{\delta}{\delta \boldsymbol{b_i}}(\boldsymbol{n_k r_k dS}) - \int_{S_w} \tau_{km}\frac{\delta}{\delta \boldsymbol{b_i}}(\boldsymbol{n_m r_k dS}) + \int_{S_w} \Psi_5 \boldsymbol{q_k}\frac{\delta(\boldsymbol{n_k dS})}{\delta \boldsymbol{b_i}}$$

$$- \int_{S_w} \Psi_n \frac{\partial \boldsymbol{f_{nk}^{inv}}}{\partial \boldsymbol{x_l}}\frac{\delta \boldsymbol{x_l}}{\delta \boldsymbol{b_i}}\boldsymbol{n_k dS} + \int_{S_w} (\Psi_{k+1}\boldsymbol{P} - \Psi_n \boldsymbol{f_{nk}^{inv}})\frac{\delta(\boldsymbol{n_k dS})}{\delta \boldsymbol{b_i}}$$

$$+ \int_{S_w} \left[\left(-\tau_{\boldsymbol{km}}^{\boldsymbol{adj}} + \Psi_5 \tau_{km}\right)\frac{\partial \upsilon_m}{\partial x_l} + \Psi_5 \frac{\partial q_k}{\partial x_l} + \Psi_{m+1}\frac{\partial \tau_{km}}{\partial x_l}\right]\frac{\delta \boldsymbol{x_l}}{\delta \boldsymbol{b_i}}\boldsymbol{n_k dS}$$

GPUs on the same board **GPUs Cluster**



Tesla M2050

- **CUDA 5**
- **Adjoint Methods on GPUs**
- **Programming and running on multiple GPUs**

- **Four-Node HP system**
- **3 Tesla M2050 per node**

Advantages & Disadvantages

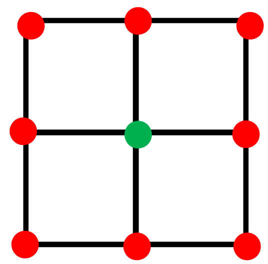
Pros :

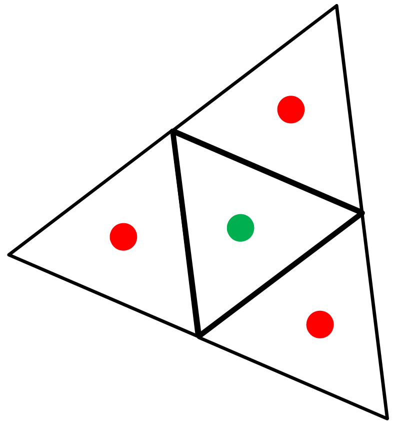
- **Massively parallel processors.**
- **High FLOPS (floating-point operations per second).**
- **High memory bandwidth.**

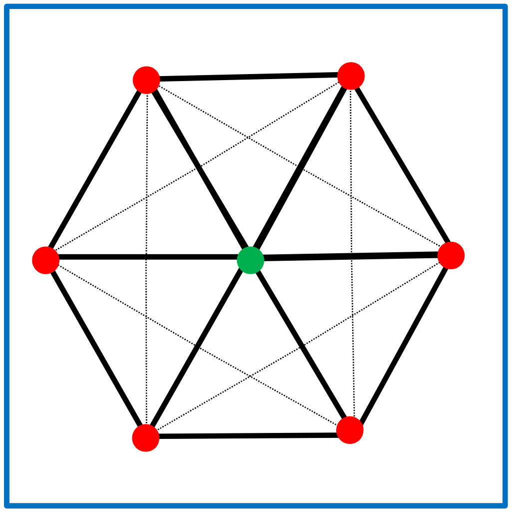
Cons :

- **Limited amount of memory.**
- **Limited amount of cash memory.**

Vertex-centered finite volume method on unstructured grids, on GPUs is the most difficult case (compared to either structured grids or cell-centered schemes on unstructured grids).



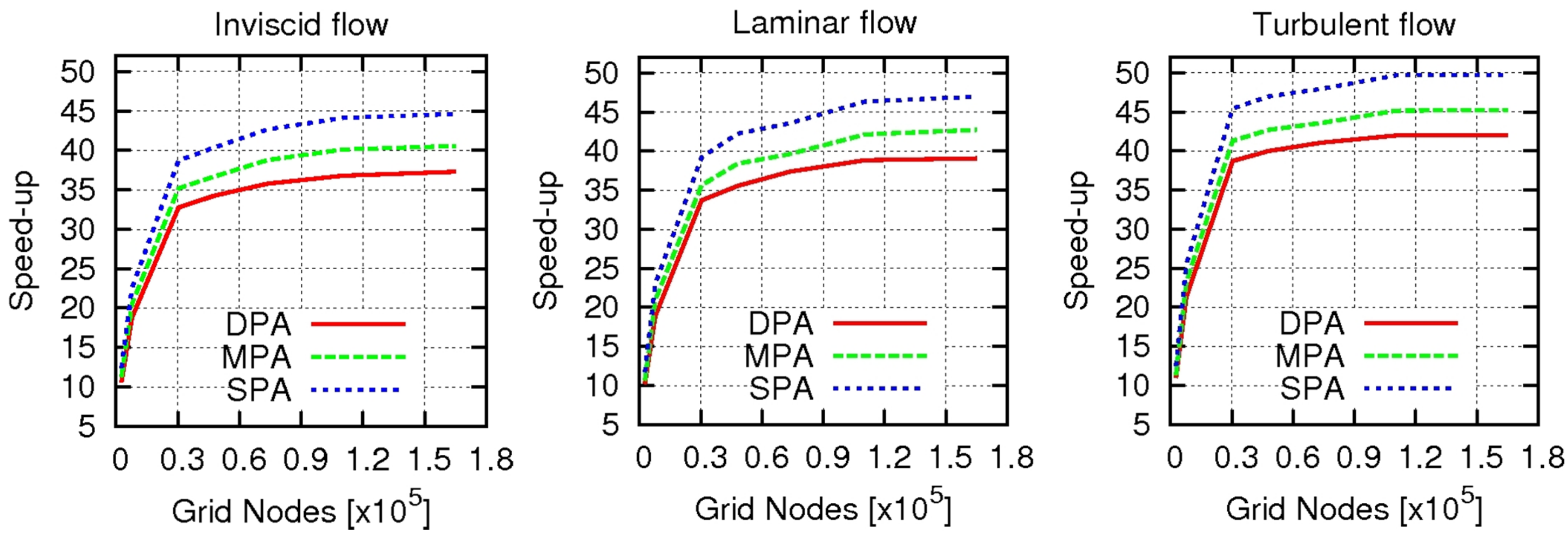




Mixed Precision Arithmetics

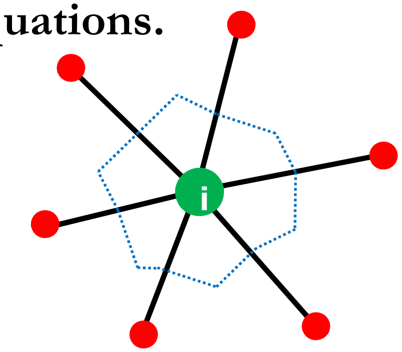
- **Mixed precision arithmetics results in over 30% reduction in GPU memory usage and, consequently, higher speed-up due to the increased number of variables in the cash memory.**

$$LHS(U^k) \cdot \Delta U = RHS(U^k)$$
$$U^{k+1} = U^k + \Delta U$$



One-Kernel, Two-kernel Scheme

- One-kernel scheme for the primal and the left side of the adjoint equations, two-kernel scheme for the right side of the adjoint equations.



- Global memory which has the largest capacity is reserved for large data such as the Jacobian matrix.

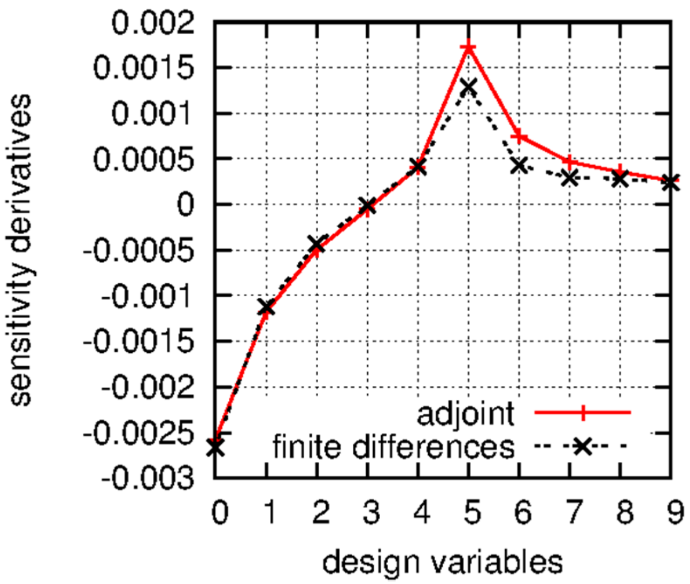
- Data accessed frequently and randomly like the primal or adjoint variable gradients are stored in texture memory.

- Constant quantities like the gas constants are stored in the (fast) constant memory.

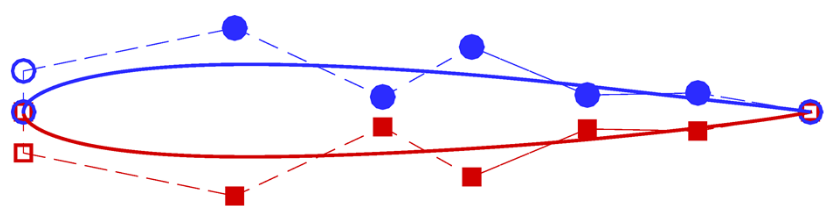
- Threads of the same block exchange data through the shared memory.

Shape optimization of an airfoil

Objective function F = -$W_L C_L$ + $W_D C_D$
W_L **= 0.1,** W_D **= 1.0,** M_∞ **= 0.3,** $\alpha_\infty = 2^\circ$ **and Re = 1000.**

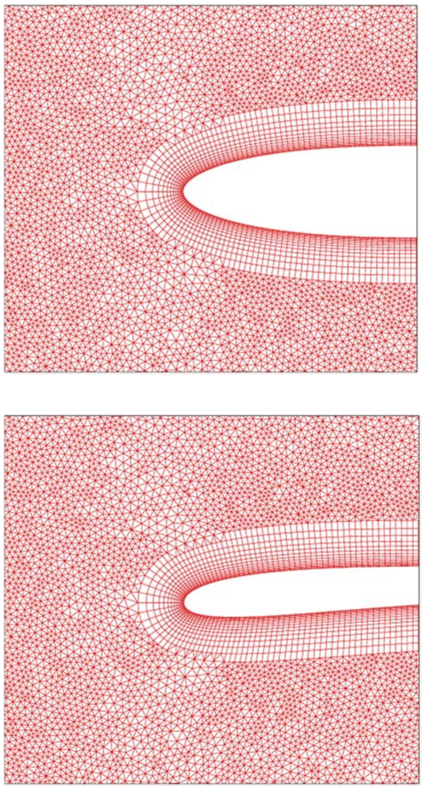


Sensitivity derivatives using finite differences and adjoint

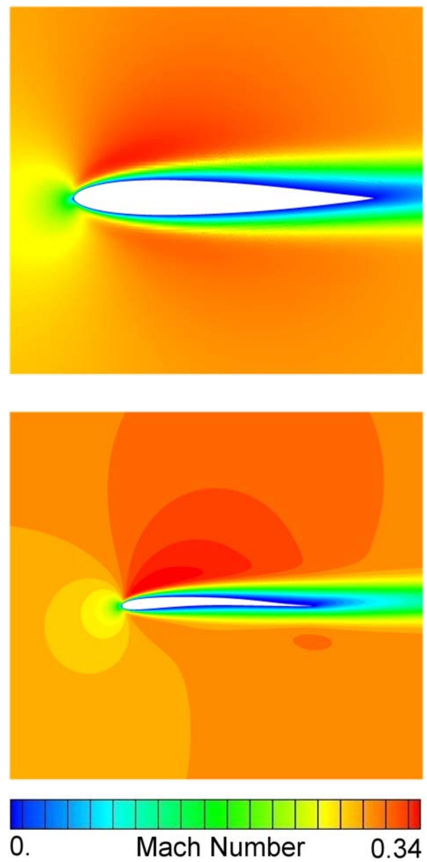


Initial geometry of NACA0012 and Bezier control points, 56270 nodes

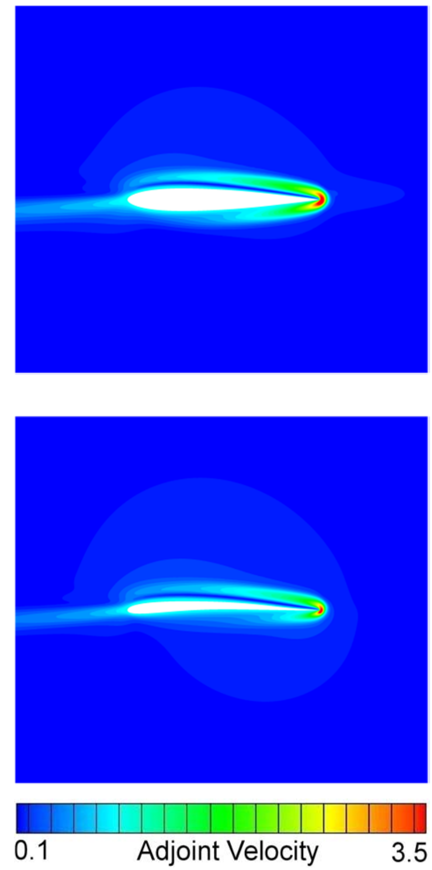
Shape optimization of an airfoil



Mesh at leading edge of initial (top) and optimized (bottom) shape



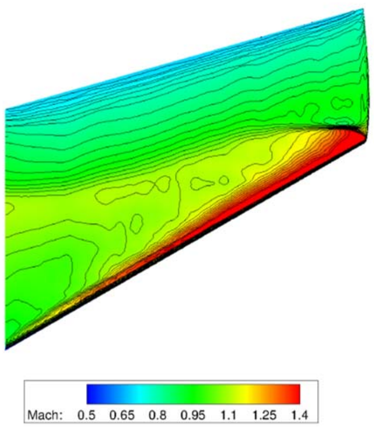
Mach number of initial (top) optimized (bottom) shape

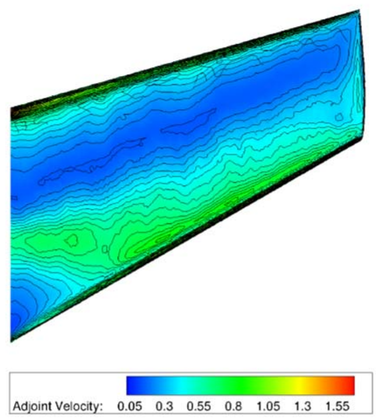


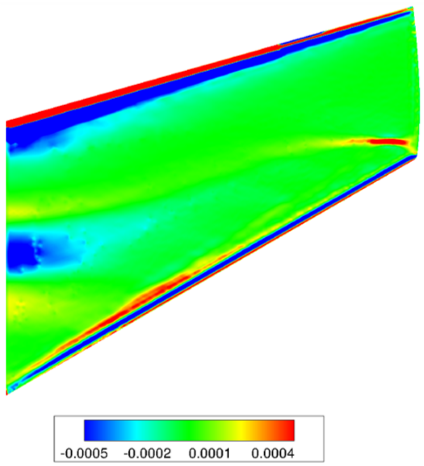
Adjoint velocity of initial (top) optimized (bottom) shape

Sensitivity map of a transonic wing

Drag minimization of ONERA M6 for inviscid flow at $M_{\infty} = 0.84$ and $\alpha_{\infty} = 4^{\circ}$







Mach Iso lines **Adjoint velocity Iso lines** **Sensitivity map**

Comments

- More than 100% increase in C_L from 0.120 (initial) to 0.264 (optimized).
- 13.5% reduction in C_D from 0.118 (initial) to 0.102 (optimized).
- Primal and adjoint solution takes 20 minutes to complete for 30 Optimization cycles on a single Nvidia Tesla M2050.
- The same procedure using a dual quad core Intel Xeon CPU E5620 (2.40 GHz) takes 15 hours!
- Primal and adjoint equations are solved 47times and 52times faster respectively compared to running the case on the corresponding CPU.