Identification of vortices in flow using Graphics Processing Unit (GPU)

Jakub Šístek
joint work with
Václav Kolář and Pavel Moses

Institute of Mathematics of the AS CR, Praha
Institute of Hydrodynamics of the AS CR, Praha
Czech Technical University in Prague

Seminar *Current Problems in Numerical Analysis*
Praha, November 26, 2010

Outline

Outline

Overview of GPU computing

Identification of vortices in flow fields

Numerical results

Conclusion

Raise of GPU computing

- Graphics Processing Units (GPU) has evolved under pressure of computer games into very powerful though highly specialized chips with many cores used for *fast rendering of images* – pixels independent, single precision, specific datatypes
- early General-purpose computing on Graphics Processing Units (GPGPU) – difficult and marginal – but attract some attention on its performance – people realized, that they have (a certain type of) multicore chips already in their computers at the same time when CPU vendors anounced transition to two or four cores
- **June 2007** – milestone – NVIDIA released *Compute Unified Device Architecture* (CUDA) version 1.0 – a 'human-friendly' interface that presents an extension to C++

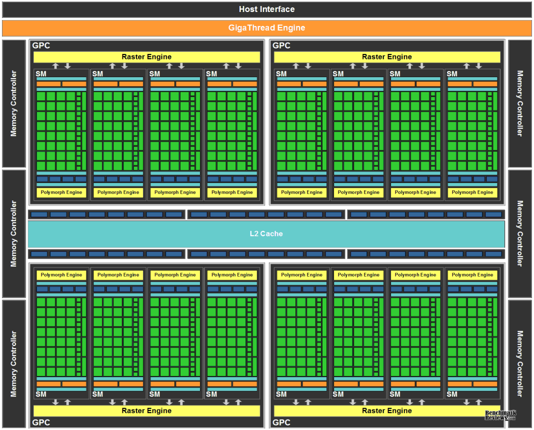
Raise of GPU computing

- December 2008 – *OpenCL* 1.0 released – standardization of GPU-type computing (Apple, NVIDIA, AMD, IBM, Intel, and others)
- March 2010 – release of Fermi type chips by NVIDIA
- November 2010 – in Top 500 list of most powerful computers (www.top500.org), Chinese computers *Tianhe-1A* and *Nebulae* ranked 1st and 3rd, respectively – both based on NVIDIA Tesla 2050/2070 cards for acceleration

CUDA

- Compute Unified Device Architecture
- NVIDIA's attempt to enter high performance computing (HPC)
- currently version 2.1
- extension to C++
- collection of driver, compiler, debugger, visual profiler and sample codes to provide basic tools for quick development of new applications

Fermi GPU chip



- 512 stream processors 700 MHz
- 1,536 MB RAM (GeForce 480), 15 streming multiprocessors
- GeForce (general use), Tesla (HPC), Quadro (CAD)
- 1,345 Gflops (CPU AMD Athlon X4 around 0.5 Gflops)

Memory hierarchy on GPU

NVIDIA GeForce GTX 480:

- device memory (1,536 MB) – bandwidth 177 GB/s
- shared memory (64 kB per multiprocessor) – bandwidth 1.344 TB/s
- texture memory
- registers

Efficient usage of memory levels is the key to exploit the GPU power.

Program execution on a GPU

- basic program unit is a *thread*
- 32 threads make a *warp* that is assigned to a multiprocessor – can synchronize, use shared memory to communicate data
- when executed, several warps are assigned to each multiprocessor and served by a sophisticated *runtime system*
- in CUDA code, threads are grouped into two dimensional *blocks* – for optimal performance, size of block is a multiple of size of the multiprocessor
- data divided into *blocks* which are executed by multiprocessors – single thread corresponds to single data unit (e.g. array element)
- *blocks* are organised into two dimensional *grid*

Program execution on a GPU

1. serial code running at CPU allocates space in memory of GPU and copies data to it (device memory)
2. parallel execution of *kernel function*
3. after completion, data are stored in memory of GPU and copied back to RAM, control is given back to CPU

Kernel function

- written for a single thread
- preferably accessing private memory location
- executed by runtime system

Sample kernel function

Function that copies integer 2D array x to y

```
__global__ void kernel_copy(int ldim, int *x, int *y)
{
 // where am I, from 0
 unsigned int i = blockIdx.x*blockDim.x+threadIdx.x;
 unsigned int j = blockIdx.y*blockDim.y+threadIdx.y;

 // copy my array
 y[i+ldim*j] = x[i+ldim*j];
}
```

The kernel is invoked by calling:

```
kernel_copy <<<dimGrid,dimBlock>>>(1, d_a, d_b );
```

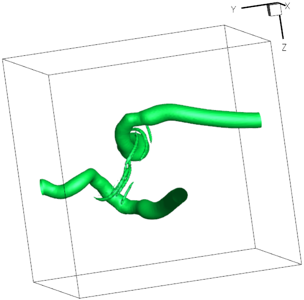
Here d_a and d_b are pointers to device memory.

Problems suitable for GPU computing

- **arithmetic intensity** – memory transfers slow, data should reside in device memory as long as possible with large operations/transfers ratio – **It is usually not straightforward to use GPU by a set of library functions!**
- low memory requirements for each thread
- low or no amount of communication among threads – access to non-local memory can slow down the computation
- no need for global synchronization of threads during kernel execution
- first GPU libraries are emerging – e.g. MAGMA project – linear algebra for GPU (group of Prof. Dongarra) – collaboration of CPU and GPU

Outline

Motivation



Interaction of Burgers vortices, DNS data by Prof. Rist (IAG Stuttgart)

- *visualization* of vortical structures in a flow is important in numerous areas of fluid mechanics, mainly in modelling of turbulence
- a generally accepted definition of *a vortex* and its identification in flow field is still missing
- existing methods for vortex identification are not general enough to cover all types of vortical flows

Basics of vortex identification

- current vortex identification methods are mostly based on double decomposition of velocity gradient matrix

$$\nabla \mathbf{u} = \mathbf{S} + \mathbf{\Omega},$$

where

  - $\mathbf{u} = (u, v, w)$ ... flow velocity
  - $\mathbf{S}$ ... *strain rate tensor*, $\mathbf{S} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$
  - $\mathbf{\Omega}$ ... *vorticity tensor*, $\mathbf{\Omega} = \frac{1}{2}(\nabla \mathbf{u} - (\nabla \mathbf{u})^T)$

This can be also written as

$$\mathbf{\Omega} = \frac{1}{2} \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix},$$

where

$$\omega_x = \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \quad \omega_y = \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}, \quad \omega_z = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

are components of *vorticity vector* $\omega = \nabla \times \mathbf{u}$.

Brief overview of existing methods

- popular methods are based on a definition of a scalar function that would locally discriminate vortex and non-vortex regions

chosen component of vorticity vector

- the simplest method
- plot ω_x, ω_y or ω_z according to known flow properties
- applicable only for known flow fields with vortical structures aligned with an axis

magnitude of vorticity vector

- defined as $|\omega|$
- invariant under rotation
- able to quantify the swirling motion
- influenced by shear

Brief overview of existing methods

Q-criterion [Hunt et al. (1988)]

- positive second invariant of ∇u
- region where vorticity magnitude dominates over strain-rate magnitude

$$Q = \frac{1}{2}\left(\|\boldsymbol{\Omega}\|^2 - \|\mathbf{S}\|^2\right) > 0$$

Here we use the norm $\|\mathbf{A}\|^2 = \mathrm{tr}(\mathbf{A}\mathbf{A}^T)$.

Brief overview of existing methods

Δ-criterion [Dallmann (1983), Vollmers et al. (1983), Chong et al. (1990)]

- define vortex as the region where ∇u has a pair of complex eigenvalues – this corresponds to spiral or closed streamlines
- for incompressible flows, this correspond to

$$\Delta = \left(\frac{Q}{3}\right)^3 + \left(\frac{R}{2}\right)^2 > 0,$$

where Q (see above) and $R = \mathrm{Det}(\nabla u)$ are invariants of ∇u.

Brief overview of existing methods

λ_2-criterion [Jeong and Hussain (1995)]

- dynamic considerations – search for *a pressure minimum across the vortex* – assumes that it corresponds with vortices
- define the vortex as a connected region, where the pressure Hessian, approximated by $\mathbf{S}^2 + \mathbf{\Omega}^2$, has two negative eigenvalues
- $\mathbf{S}^2 + \mathbf{\Omega}^2$ is symmetric – it has real eigenvalues $\lambda_1 \leq \lambda_2 \leq \lambda_3$
- the final criterion is thus defined as region where

$$\lambda_2 < 0$$

- valid for incompressible fluids only
- but the pressure minimum is neither sufficient nor necessary condition for existence of a vortex
- paper *On The Identification of a Vortex* has been cited about a thousand times (as for November 2010)

Triple Decomposition Method

- conventional double decomposition of motion near a point into symmetric and skew-symmetric parts corresponding to *straining motion* and *rigid rotation* has long history (Cauchy-Stokes decomposition theorem, 1845)
- both components are in general partially induced by *shear*
- *Triple Decomposition Method* (TDM) [Kolář (2007)] presents a novel approach to decompose relative motion near a point into **three** rather than two elementary components introducing *shear* besides *residual strain* and *residual rigid rotation*
- in TDM, velocity gradient tensor $\nabla \mathbf{u}$ is decomposed as

$$\nabla \mathbf{u} = \mathbf{S}_{RES} + \mathbf{\Omega}_{RES} + \nabla \mathbf{u}_{SH}$$

  where
  - $\mathbf{S}_{RES}$ ... residual strain tensor
  - $\mathbf{\Omega}_{RES}$ ... residual vorticity tensor
  - $\nabla \mathbf{u}_{SH}$ ... shear component

Principles of Triple Decomposition Method

The *pure shearing motion* is described by the *pure shear tensor* which appears in a suitable *reference frame* (i.e. under certain orthogonal transformations) as an *asymmetric matrix A*

$$a_{i,j} = 0 \text{ or } a_{j,i} = 0, i, j = 1, 2, 3.$$

- $\mathbf{S}_{RES} + \mathbf{\Omega}_{RES}$ form *residual tensor*,

$$\nabla \mathbf{u} = \begin{pmatrix} \text{residual} \\ \text{tensor} \end{pmatrix} + \begin{pmatrix} \text{shear} \\ \text{tensor} \end{pmatrix}$$

- the residual tensor consists of diagonal terms and symmetric or antisymmetric parts of the off-diagonal terms of $\nabla \mathbf{u}$, the 'leftovers' are moved to shear tensor

$$\begin{pmatrix} \text{res.} \\ \text{ten.} \end{pmatrix} = \begin{pmatrix} u_x & \operatorname{sgn}(u_y)\min(|u_y|, |v_x|) & \cdot \\ \operatorname{sgn}(v_x)\min(|u_y|, |v_x|) & v_y & \cdot \\ \cdot & \cdot & w_z \end{pmatrix}$$

- maximum of *effective shear* is extracted when norm of the *residual tensor* attains minimum

Example decomposition [Kolář (2007)]

$$\underbrace{\begin{pmatrix} -1 & 15 & 17 \\ 3 & 8 & -14 \\ -26 & -14 & -5 \end{pmatrix}}_{\nabla \mathbf{u}} = \underbrace{\begin{pmatrix} -1 & 3 & 17 \\ 3 & 8 & -14 \\ -17 & -14 & -5 \end{pmatrix}}_{residual\ \ tensor} + \underbrace{\begin{pmatrix} 0 & 12 & 0 \\ 0 & 0 & 0 \\ -9 & 0 & 0 \end{pmatrix}}_{\nabla \mathbf{u}_{SH}}$$

Triple Decomposition Method

- using standard strain-rate and vorticity tensors $\mathbf{S}$ and $\boldsymbol{\Omega}$, the following expression holds [Kolář (2007)]

$$||\nabla \mathbf{u}||^2 = \left|\left|\left(\begin{array}{c} \text{residual} \\ \text{tensor} \end{array} \right)\right|\right|^2 + 4(|S_{12}\Omega_{12}| + |S_{23}\Omega_{23}| + |S_{31}\Omega_{31}|).$$

- minimum of the residual tensor norm corresponds to the reference frame where maximum of the term

$$|S_{12}\Omega_{12}| + |S_{23}\Omega_{23}| + |S_{31}\Omega_{31}|$$

is attained.

Basic Reference Frame (BRF)

- the previously defined frame is called *basic reference frame (BRF)* and its determination can be stated as the problem: *Find orthogonal matrix* $\mathbf{Q}_{BRF}$ *such that*

$$\max_{\mathbf{Q}}(|S_{12}\Omega_{12}| + |S_{23}\Omega_{23}| + |S_{31}\Omega_{31}|) \tag{1}$$

  is attained, where

$$\begin{aligned} \mathbf{A} &= \mathbf{Q}(\nabla\mathbf{u})\mathbf{Q}^T, \\ \mathbf{S} &= \frac{1}{2}\left(\mathbf{A} + \mathbf{A}^T\right), \\ \mathbf{\Omega} &= \frac{1}{2}\left(\mathbf{A} - \mathbf{A}^T\right). \end{aligned}$$

- in this optimization problem, $\mathbf{Q}$ is constructed from three angles of rotating frame along axes, $Q = f(\alpha, \beta, \gamma)$, $0 \le \alpha < \pi$, $0 \le \beta < \pi$, $0 \le \gamma \le \pi/2$

Basic Reference Frame (BRF)

- BRF is in general different and independent for each point of the flow field – whole range of angles must be taken into account at every position when looking for it
- to find BRF numerically, ranges of angles are uniformly discretized with reasonable (angle) step size and the expression (1) is evaluated for combinations of α_i, β_j and γ_k
- while finer step size allows better localization of the BRF, it results in increase of required objective function evaluations since there is $1/\Delta\alpha^3$ dependence on the step size $\Delta\alpha$ (e.g. 2,981,251 evaluations for step size 1 deg)

TDM for vortex identification

Once approximation to $\mathbf{Q}_{BRF}$ is known, we determine the residual vorticity by

1. transform the velocity gradient into BRF by

$$\mathbf{A} = \mathbf{Q}_{BRF}(\nabla\mathbf{u})\mathbf{Q}_{BRF}^{T},$$

2. get residual tensor $\mathbf{A}_{RES}$ from $\mathbf{A}$
3. perform double decomposition to obtain *residual* strain-rate tensor $\mathbf{S}_{RES}$ and *residual* vorticity tensor $\mathbf{\Omega}_{RES}$

$$\begin{aligned}\mathbf{S}_{RES} &= \frac{1}{2}\left(\mathbf{A}_{RES} + \mathbf{A}_{RES}^{T}\right),\\ \mathbf{\Omega}_{RES} &= \frac{1}{2}\left(\mathbf{A}_{RES} - \mathbf{A}_{RES}^{T}\right).\end{aligned}$$

4. a *vortex* is defined as a connected region where $\|\mathbf{\Omega}_{RES}\| > 0$

- TDM eliminates the effect of shear on vorticity
- capable of describing a vortex in incompressible as well as compressible fluids

TDM in 2D

In 2D – vortex axis is always peripendicular to flow plane – the method simplifies due to one axis of BRF set perpendicular to the plane to the following formulas:

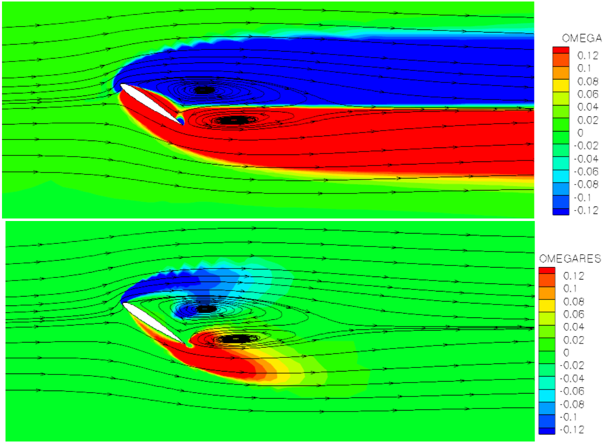
$$\omega_{RES} = \begin{cases} 0 & \text{for} \quad |s| \geq |\omega| \\ \mathrm{sgn}(\omega)(|\omega| - |s|) & \text{for} \quad |s| \leq |\omega| \end{cases} ,$$

where e.g. for incompressible flow

$$\begin{aligned} |s| &= \left(\sqrt{4\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2} \right) /2, \\ \omega &= \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) /2. \end{aligned}$$

Outline

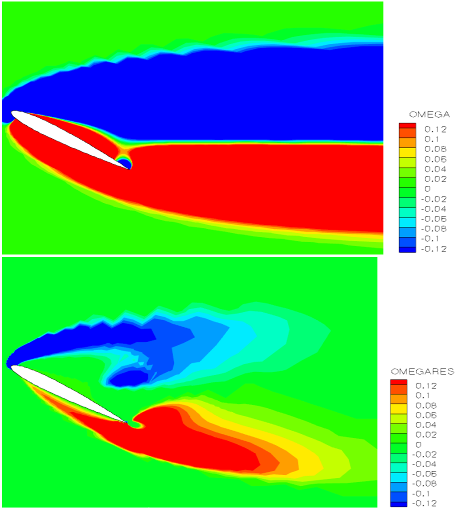
Numerical example in 2D



Vorticity defined by 'standard' two-dimensional vorticity tensor component ω (top) and by *residual vorticity* (bottom) in flow past NACA 0012 airfoil, $\alpha = 34^\circ$, Re $= 100$

FEM data by Dr. Jaroslav Novotný

Numerical example in 2D

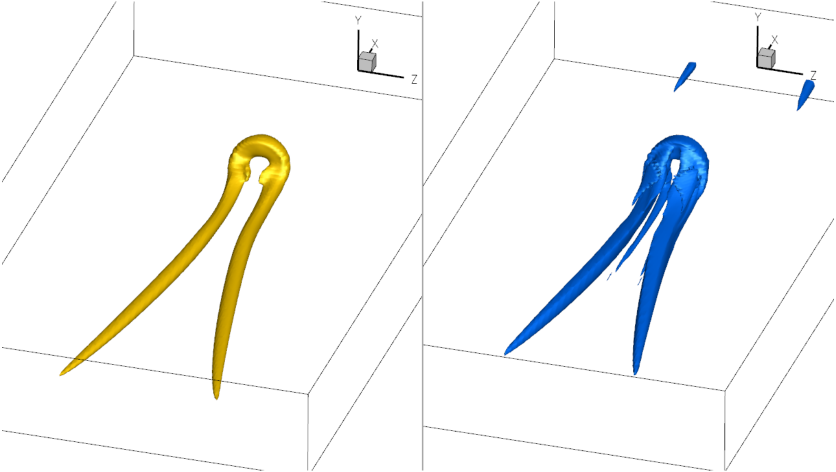


Details of vorticity defined by 'standard' two-dimensional vorticity tensor component ω (top) and by *residual vorticity* (bottom) in flow past NACA 0012 airfoil, $\alpha = 34^\circ$, Re $= 100$

Numerical example in 3D

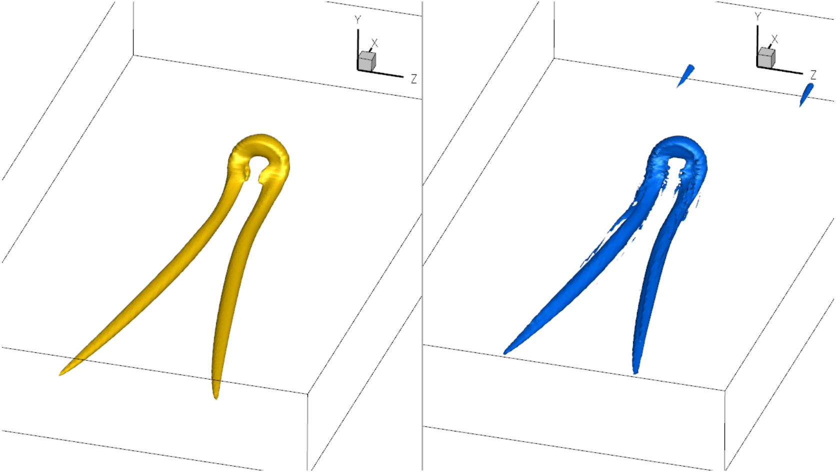
- in 2D, TDM is very competitive, in 3D it poses a practical issue of computational costs due to the neccessity of finding the BRF, experiments have shown that in general cases the choice of step of angles may have a large impact on the quality of the identification and thus the step size of 1-10 degrees is recommended
- BRF is specific for each data point – 2,981,251 possibilities for step size 1 deg
- BRF for each point is independent – presents an *embarassingly parallel* computation – specific properties of GPU can be exploited to speed-up calculations
- flow data for 3D DNS simulations are due to kindness of the team of Prof. Rist from IAG Stuttgart

Numerical example in 3D



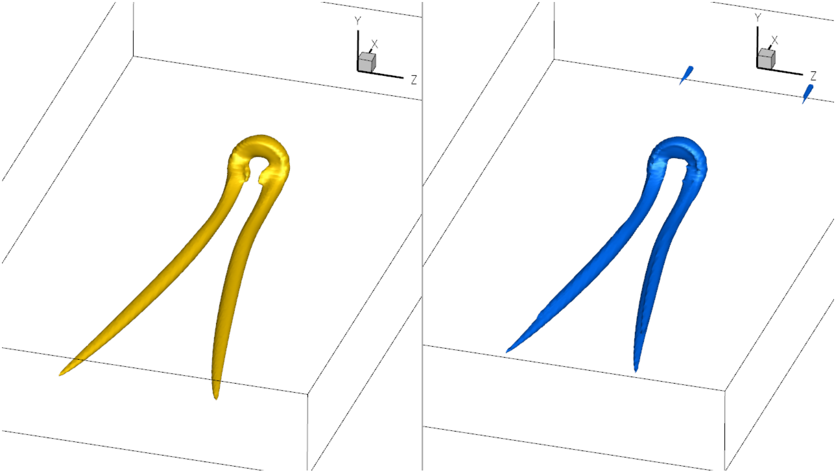
Ω-vortex in boundary layer, comparison of λ_2-method at 4.6% treshold (left) and residual vorticity at 13% treshold, BRF search step 90 deg (right)

Numerical example in 3D



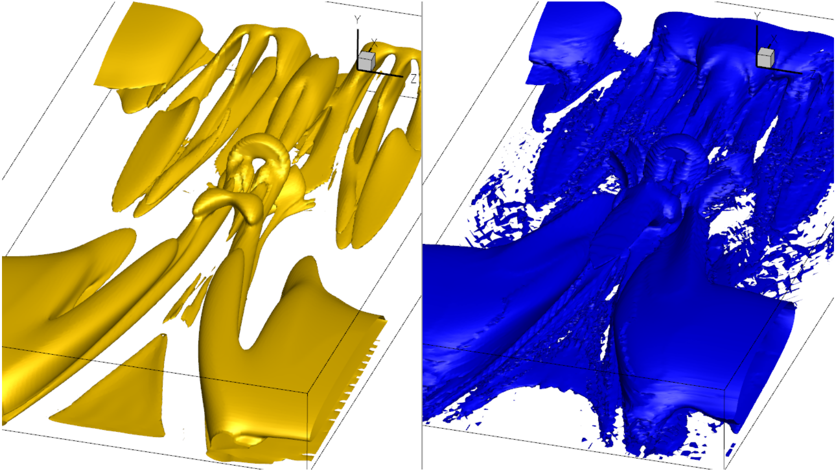
Ω-vortex in boundary layer, comparison of λ_2-method at 4.6% treshold (left) and residual vorticity at 13% treshold, BRF search step 30 deg (right)

Numerical example in 3D



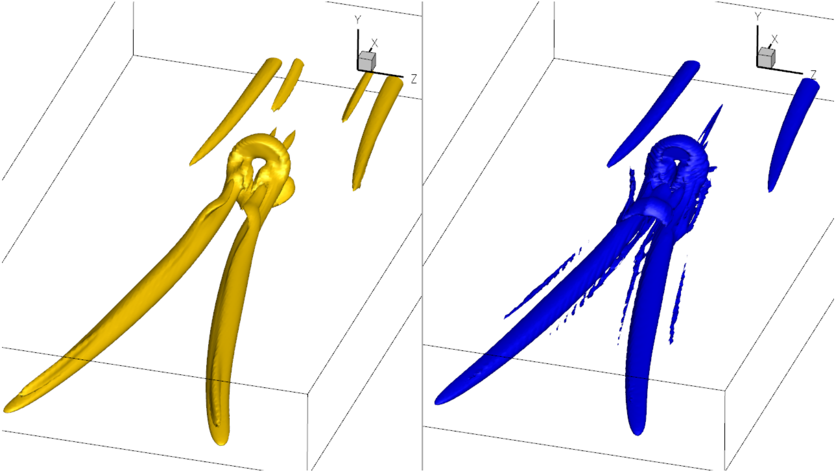
Ω-vortex in boundary layer, comparison of λ_2-method at 4.6% treshold (left) and residual vorticity at 13% treshold, BRF search step 1 deg (right)

Numerical example in 3D



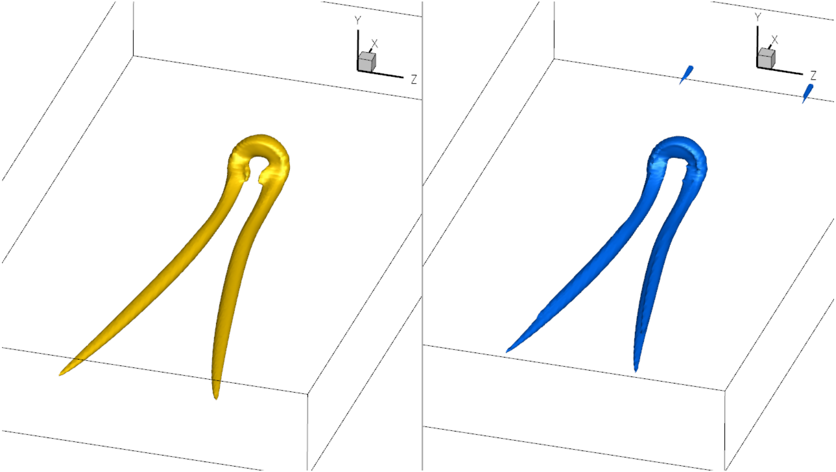
Ω-vortex in boundary layer, comparison of λ_2-method at 0.03% treshold (left) and residual vorticity at 0.2% treshold, BRF search step 1 deg (right)

Numerical example in 3D



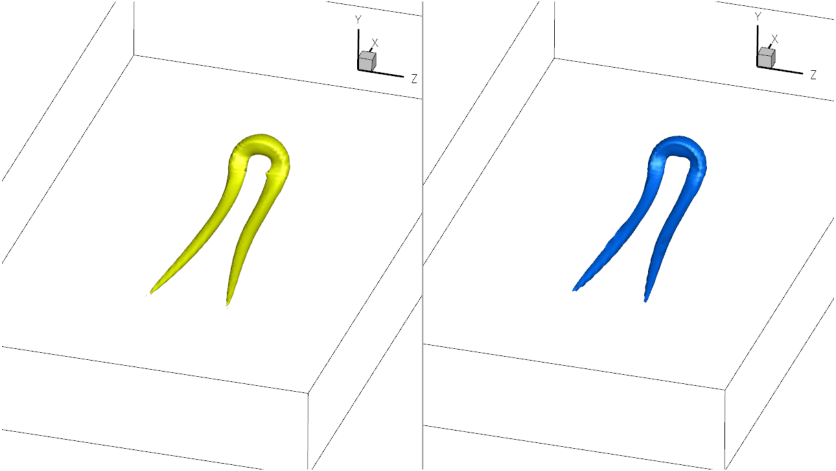
Ω-vortex in boundary layer, comparison of λ_2-method at 0.76% treshold (left) and residual vorticity at 2.2% treshold, BRF search step 1 deg (right)

Numerical example in 3D



Ω-vortex in boundary layer, comparison of λ_2-method at 4.6% treshold (left) and residual vorticity at 13% treshold, BRF search step 1 deg (right)

Numerical example in 3D



Ω-vortex in boundary layer, comparison of λ_2-method at 7.6% treshold (left) and residual vorticity at 19.4% treshold, BRF search step 1 deg (right)

Computational times

step	import (s)	BRF (s)		$\|\Omega\|_{RES}$	λ_2 (s)
		GF 9650M GT	GTX 480		
90	15.6	0.4	0.05	1.4	4.9
45		2.0	0.13		
30		6.7	0.34		
10		172.8	8.0		
5		1,373.1	63.9		
2		4,093.7	997.8		
1		n/a	2,220.6		

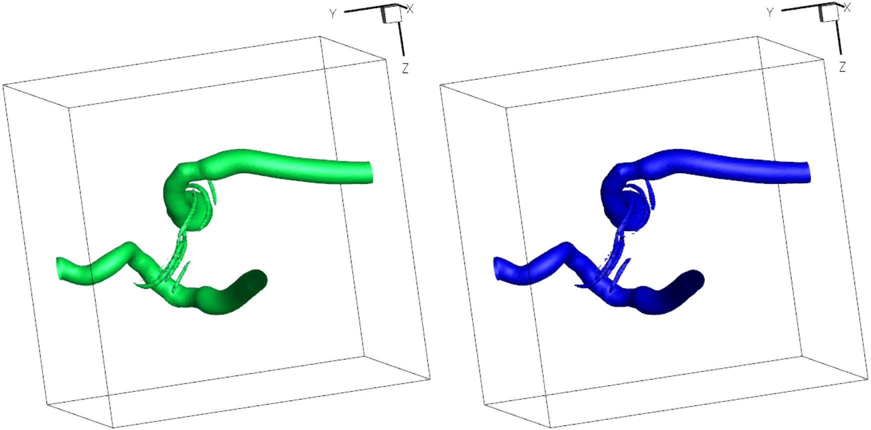
- GF 9650M GT ... NVIDIA GeForce 9650M GT (my laptop)
- GTX 480 ... NVIDIA GeForce GTX 480 (recently acquired by Institute of Mathematics)
- computational times on Tesla T10 GPU (UC Denver) comparable to GTX 480

Computational times

Serial computation of BRF on laptop CPU (Intel Core Duo 2.53 GHz):

- for step size 2 deg, one point takes 142 ms
- the problem presented consists of $175 \times 101 \times 129 = 2{,}280{,}075$ points
- the serial computation would take about 90 hours on CPU of my laptop
- it would take about **28 days** with step size 1 deg

Numerical example in 3D



Burgers vortex, Ma $= 0.3$, comparison of λ_2-method at 3.9% treshold (left) and residual vorticity at 16% treshold, BRF search step 1 deg (right)

Outline

Conclusion

- Triple Decomposition Method (TDM) appears to be competitive with standard vortex identification methods in terms of quality of vortex identification
- TDM may be costly procedure for fine resolution
- it offers a good intuitive insight into the process of vortex identification
- the method is subject to ongoing research and some related methods are being developed [Kolář, Moses, Šístek (2010)]
- determination of basic reference frame is a 'dream problem' to be addressed by GPU computing
- achieved speed-up 80 (within a laptop) or 340 (on the new generation GPU chip) made from the TDM method
  a candidate for fine analysis of vortical structures in a complex flow field
- code *Vortex Analyzer* available to public at my website `http://www.math.cas.cz/~sistek`

Acknowledgements

- authors are very grateful to Prof. Ulrich Rist and Dr. Kudret Baysal, IAG Stuttgart for providing the DNS data sets used in the present paper
- research has been supported by Grant Agency of Czech Academy of Sciences under grant IAA200600801

References

[1] Jeong, J., Hussain, F.: On the identification of a vortex. J. Fluid Mech. **285**, 69-94 (1995)

[2] Kolář, V.: Vortex identification: new requirements and limitations. Int. J. Heat Fluid Flow **28**(4), 638 - 652 (2007)

[3] Kolář, V., Moses, P., Šístek, J.: Local corotation of line segments and vortex identification. In: Mallinson, G. and Cater, J. (eds.) Proceedings of the 17th Australasian Fluid Mechanics Conference, Auckland, New Zealand, 2010.

[4] Kolář, V., Moses, P., Šístek, J.: Triple Decomposition Method for Vortex Identification in Two-Dimensional and Three-Dimensional Flows, To appear in *Computational Fluid Dynamics 2010*, Proceedings of The 6th ICCFD Conference, St. Petersburg, Russia, July 12–16, 2010.