Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia IUTAM 7 (2013) 59 – 66

Procedia IUTAM

www.elsevier.com/locate/procedia

Topological Fluid Dynamics: Theory and Applications

Parallel computation of vortex tube reconnection using a graphics card and the 3D vortex–in–cell method

Henryk Kudela*, Andrzej Kosior

Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland

Abstract

In this paper we present numerical results for vortex tube reconnection for three different initial configurations: (i) initially straight orthogonally offset tubes, (ii) initially parallel counter-rotating vortices that are sinusoidally perturbed, and (iii) reconnecting vortex rings. Our aim is to detect some universal mechanism for core reconnection. The mixing of the fluid by the reconnection process is demonstrated by tracing passive markers placed initially near the vortex tubes involved in reconnection. For these numerical simulations we use the 3D vortex–in–cell (VIC) method. In order to overcome the problem of the large time required for single-processor calculations, we constructed a numerical code for the multiprocessor environment of a graphics card. We show that the VIC method is very well suited for parallel computation. We carefully tested the method by comparing the numerical results with theoretical results (for motion of a vortex ring and other published results). The computational speed obtained was nearly 50 times greater than for a single processor.

Selection and/or peer-review under responsibility of the Isaac Newton Institute for Mathematical Sciences, University of Cambridge

Keywords: reconnection; vortex tube; VIC method;

1. Introduction

Understanding the dynamics and mutual interaction of various types of vortical motions is the key ingredient in clarifying and controlling fluid motions. One of the most fundamental 3D vortical interactions is related to vortex tube reconnection. The primary source of information regarding the interaction of vortex structures has been flow visualization. However, visualization may not be able to reproduce properly vortical features of a flow, especially for long-time behavior [**?**, **?**]. It is considered that numerical simulation is the only satisfactory approach to study reconnection of two vortex tubes. As a numerical method we chose the 3D vortex–in–cell (VIC) method [**?**]. In this method particles that carry information about vorticity are used. Tracing the positions of these particles allows one to study how the vorticity evolves. It is well known that the velocity may be calculated from the vorticity distribution.

In the VIC method, after several time steps particles have a tendency to concentrate in areas where the velocity gradient is very high. This may lead to spurious vortex structures. To avoid this situation after an arbitrary number

* Corresponding author. Tel.: +48-71-320-2040 ; fax: +48-71-341-7708
E-mail address: henryk.kudela@pwr.wroc.pl

2210-9838
Selection and/or peer-review under responsibility of the Isaac Newton Institute for Mathematical Sciences, University of Cambridge
doi:10.1016/j.piutam.2013.03.008

of time steps, redistribution ('remeshing') of particles to regular positions is done. In 2D we found [**?**, **?**] that it is useful to remesh the particles at every time step. At the beginning the vortex particles are put on grid nodes. After displacement at each time step the intensities of the particles are redistributed again to the initial grid nodes. This strategy has several advantages such as the shortening of computational time and the accurate simulation of viscosity. In the present paper we implement this idea in 3D flow. Since the computations took a very long time on a single core, we found that the speed-up provided by parallel computing was necessary. The VIC method is very well suited for parallel computation. The redistribution process, which has to be done at each time step, has a local character and the computations for each particle can be done independently. Also each displacement of the vortex particles has a local character. So the whole set of particles can be divided into independent groups and operations over these groups can be done concurrently.

To speed-up calculations we decided to use the multicore architecture of the graphics card. Graphics Processing Units (GPUs) that were developed for video games provide cheap and easily accessible hardware for scientific calculations.

2. Equation of motion and description of the 3D VIC method

Equations of incompressible and viscous fluid motion have the following form:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\Delta\mathbf{u} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

where $\mathbf{u} = (u, v, w)$ is velocity vector, ρ is fluid density, p is pressure,ν is kinematic viscosity. The equation (1) can be transformed to the Helmholtz equation for vorticity evolution[**?**]:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u}\cdot\nabla)\boldsymbol{\omega} = (\boldsymbol{\omega}\cdot\nabla)\mathbf{u} + \nu\Delta\boldsymbol{\omega} \tag{3}$$

where $\boldsymbol{\omega} = \nabla\times\mathbf{u}$. From incompressibility (2) stems the existence of a vector potential $\mathbf{A}$, such that:

$$\mathbf{u} = \nabla\times\mathbf{A} \tag{4}$$

Assuming additionally that the vector $\mathbf{A}$ is incompressible ($\nabla\cdot\mathbf{A} = 0$) its components can be obtained by solution of the Poisson equation

$$\Delta A_i = -\omega_i, \qquad i = 1, 2, 3. \tag{5}$$

Solving (5) one is able to calculate the velocity by formula (4).

In the VIC method the viscous splitting algorithm is used. The solution is obtained in two steps: first, the inviscid Euler equation is solved.

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u}\cdot\nabla)\boldsymbol{\omega} = (\boldsymbol{\omega}\cdot\nabla)\mathbf{u}. \tag{6}$$

In the second step, the viscosity effect is simulated by solving the diffusion equation

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \nu\Delta\mathbf{u}, \qquad \boldsymbol{\omega}_{t=0} = \boldsymbol{\omega}_I. \tag{7}$$

where $\boldsymbol{\omega}_I$ is vorticity distribution obtained after the inviscid step.

For the solution of the equation (7) one can use any suitable method like the particle-strength-exchange (PSE) method [**?**] or the finite-difference method. In the present paper we solved the diffusion equation (7) using the implicit finite-difference scheme.

2.1. Description of the VIC method for the three-dimensional case

First we have to discretize our computational domain. To do this, we set up a regular 3D grid $(j_1\Delta x, j_2\Delta y, j_3\Delta z)$ $(j_1, j_2, j_3 = 1, 2, \ldots, N)$, where $\Delta x = \Delta y = \Delta z = h$. The same mesh will be used for solving the Poisson equation. The continuous field of vorticity is replaced by a discrete distribution of Dirac delta measures [**?**, **?**]

$$\boldsymbol{\omega}(\mathbf{x}) = \sum_{p=1}^{N} \boldsymbol{\alpha}_p(\mathbf{x}_p)\delta(\mathbf{x} - \mathbf{x}_p) \tag{8}$$

where $\boldsymbol{\alpha}_p$ means vector vorticity particle $\boldsymbol{\alpha}_p = (\alpha_{p1}, \alpha_{p2}, \alpha_{p3})$ at position $\mathbf{x}_p = (x_{p1}, x_{p2}, x_{p3})$. The domain of the flow is covered by a numerical $(N_x \times N_y \times N_z)$ mesh with equidistant spacing h, and the i-th component of the intensity vector particle $\boldsymbol{\alpha}_i$ is defined by the expression:

$$\alpha_i = \int_{V_p} \omega_i(x_1, x_2, x_3)\,\mathrm{d}\mathbf{x} \approx h^3\omega_i(\mathbf{x}_p), \quad \mathbf{x}_p \in V_p, \quad |V_p| = h^3 \tag{9}$$

where V_p is the cell volume with index p.

From the Helmholtz theorems[**?**], we know that vorticity is carried by the fluid, so the equation for vortex particle motion is

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p, t). \tag{10}$$

We must also take into account that due to the three-dimensionality of the vorticity field, the intensities of the particles are changed by the stretching effect, according to

$$\frac{d\boldsymbol{\alpha}_p}{dt} = [\nabla\mathbf{u}(\mathbf{x}_p, t)] \cdot \boldsymbol{\alpha}_p. \tag{11}$$

The velocity at the grid nodes was obtained by solving the Poisson equation (5) by the finite-difference method and (4). The algebraic systems obtained from discretisation of the Poisson equations (5) were solved by a multigrid method. The system of equations (10), (11) was solved by the 4th-order Runge–Kutta. After solution of equations (10) and (11), the remeshing of the particles to the grid nodes is done. Then the diffusion equation (7) is solved. The m-th component $(m = 1, 2, 3)$ of the vorticity vector was approximated by

$$\frac{\omega_m^{n+1} - \omega_m^n}{\Delta t} = \nu\left(\Lambda_{x_1}\omega_m^{n+1} + \Lambda_{x_2}\omega_m^{n+1} + \Lambda_{x_3}\omega_m^{n+1}\right) \quad \text{where} \quad \Lambda_{x_i}\omega = \frac{\omega_{i+1,j,k} - 2\omega_{i,j,k} + \omega_{i-1,j,k}}{\Delta x_i^2}, \tag{12}$$

where the index the n relates to the time level $t_n = n\Delta t$, and i, j, k indices numerate the grid nodes in the x_1, x_2, x_3 directions respectively. The resulting algebraic systems were solved by the conjugate-gradient method.

2.2. Redistribution of the intensities of the particles on grid nodes

In the VIC method, particles have a tendency to gather in regions with high velocity gradients. This can lead to inaccuracies, as the particles come too close to one another. To overcome this problem, particles have to be redistributed to regular positions, i.e. back to the nodes of the rectangular mesh. We also need the redistribution of the intensities of particles on grid nodes at each time step for solution of the Poisson equations (5). Simulation of viscosity is done by solving the diffusion equation using the numerical mesh. This is done using an interpolation:

$$\omega_j = \sum_p \tilde{\alpha}_{p_n}\varphi\left(\frac{\mathbf{x_j} - \tilde{\mathbf{x}}_p}{h}\right)h^{-3}, \tag{13}$$

where j is the index of the numerical mesh node, and p is the index of a particle.

Let us assume that $x \in \mathbb{R}$. In this work, we used the following interpolation kernel [**?**]

$$\varphi(x) = \begin{cases} (2 - 5x^2 + 3|x|^3)/2 & \text{if } 0 \le |x| \le 1, \\ (2 - |x|)^2(1 - |x|)/2 & \text{if } 1 \le |x| \le 2, \\ 0 & \text{if } 2 \le |x|. \end{cases} \tag{14}$$

For the 3D case, $\varphi = \varphi(x)\varphi(y)\varphi(z)$. We require our interpolation kernel to satisfy

$$\sum_p \varphi\left(\frac{x - x_p}{h}\right) \equiv 1. \tag{15}$$

The discrepancy between the old (distorted) and new (regular) particle distribution, can be measured by the difference

$$\sum_p \tilde{\alpha}_{p_n}\delta(x - \tilde{x}_p) - \sum_p \alpha_{p_n}\delta(x - x_p). \tag{16}$$

In multiplying (16) by a test function ϕ one can get [**?**]:

$$E = \sum_p \tilde{\alpha}_{p_n}\phi(\tilde{x}_p) - \sum_p \alpha_{p_n}\phi(x_p), \tag{17}$$

where $\tilde{\alpha}_{p_n}$ and $\tilde{x}_p$ are values from the old distribution. Using (13) we can write:

$$E = \sum_p \tilde{\alpha}_{p_n}\left[\phi(\tilde{x}_p) - \sum_j \phi(x_j)\varphi\left(\frac{x_j - \tilde{x}_p}{h}\right)\right] \tag{18}$$

To evaluate the error E in (18) we have to evaluate the function

$$f(x) = \phi(x) - \sum_j \phi(x_j)\varphi\left(\frac{x_j - x}{h}\right). \tag{19}$$

Using (15) the equation (19) can be rewritten as

$$f(x) = \sum_j [\phi(x) - \phi(x_j)]\varphi\left(\frac{x_j - x}{h}\right). \tag{20}$$

The Taylor expansion of ϕ yields:

$$f(x) = \sum_j \sum_k \left[\frac{1}{k!}(x_j - x)^k \frac{d^k\phi}{dx^k}\right]^k \varphi\left(\frac{x - x_j}{h}\right). \tag{21}$$

We may conclude, that if φ satisfies the following moment condition [**?**]:

$$\sum_j (x - x_j)^k \varphi\left(\frac{x - x_j}{h}\right) = 0 \qquad 1 \le |k| \le m - 1 \tag{22}$$

then

$$f(x) = O\left(h^m\right), \tag{23}$$

and the redistribution process will be of the order m. This means that the polynomial functions up to order m will be exactly represented by this interpolation. The kernel (14) used in this work is of order $m = 4$, [**?**].

2.3. Remarks on parallel computing

As can be seen from the description of the VIC method, most calculations were related to vortex particles. Calculations for individual particles are independent of other particles and the calculations can therefore be done in parallel. As there are myriads of vortex particles in the flow (even a few millions) a massively parallel environment is desirable. For our calculations we selected Graphics Processing Units (GPUs). They have a great computing power-to-cost ratio, but required a quite different approach to programming. Details of implementation of the VIC method can be found in the authors' paper [**?**]

3. Formulations of the numerical test problems and numerical results

First we study the 'prototypical reconnection geometry' (two orthogonally-offset tubes) of Zabusky & Melander [**?**], whose results we use as a test case for our program.

3.1. Reconnection of identical orthogonally-offset tubes

The initial position is shown in the frame $t = 0$ in figure 1. Each vortex tube has the form

$$\omega(r) = \omega_0 e^{-\frac{r^2}{a^2}}, \qquad \Gamma = \int_0^a \omega(r) 2\pi r \, dr, \qquad Re_\Gamma = \frac{\Gamma}{\nu}, \tag{24}$$

where $\omega_0 = 20$, $a = 3^{-1/2}$, $\Gamma = 14.73$, $Re_\Gamma = 1403$. The domain of the computation was a box $[2\pi \times 2\pi \times 2\pi]$, $\Delta t = 0.001$, $\Delta x_1 = \Delta x_2 = \Delta x_3 = 2\pi/N$, and $N = 128$. Figure 1 shows our numerical results, which show very good agreement in shape and time evolution with the result of ([**?**], fig.3).

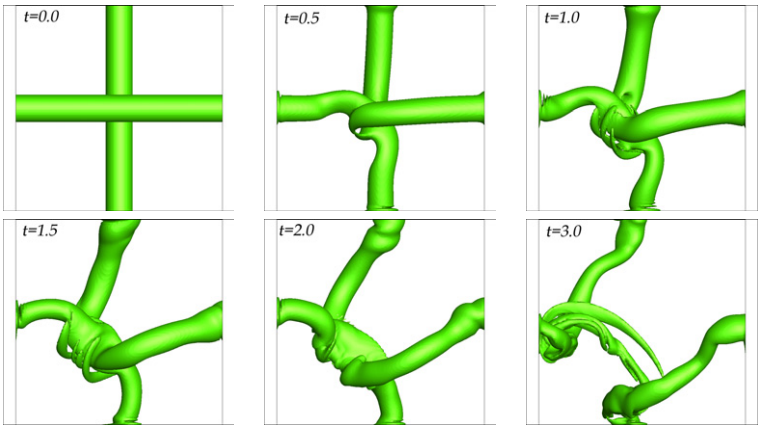

Fig. 1. Isosurface plot $|\omega| = 12$ for evolution of two identical orthogonally–offset vortex tubes. Comparison of numerical results of [**?**] - left and present work - right.

Figure 2 shows the transport of passive markers by the velocity induced by these interacting vortex tubes. Since the process of reconnection is very rapid, the loss of symmetry occurs at an early stage, and the stirring of the particles is very strong.

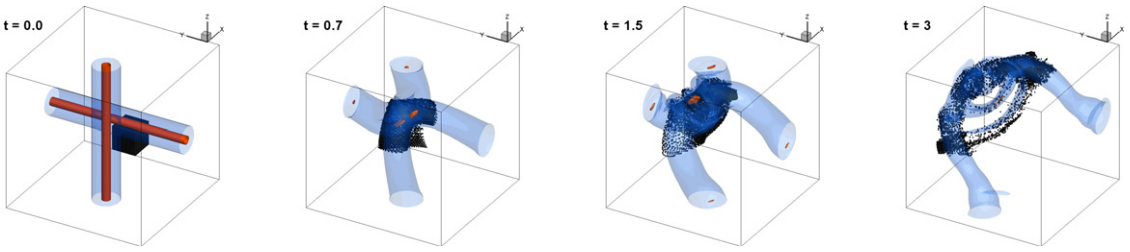

Fig. 2. Transport of passive markers by the velocity induced by two identical orthogonally–offset vortex tubes.

3.2. Reconnection of antiparallel tubes

For the tube geometry shown in figure 3, data is taken from Melander & Hussain[**?**] (see also [**?**]). We again use a Gaussian vorticity distribution in the core (24). In the computation we used $a = 0.666$, $\omega_0 = 20.0$, $\Gamma = 17.65$,

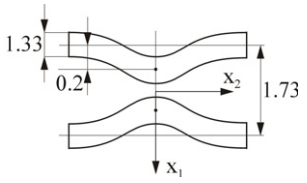

Fig. 3. Initial conditions of the antiparallel tubes with symmetric perturbation.

$Re_\Gamma = 1003$. The computational domain was $[4\pi \times 4\pi \times 4\pi]$, the number of nodes in each directions was $N = 128$, and $\Delta t = 0.01$. The quantitative agreement with the results of [**?**] is very good.

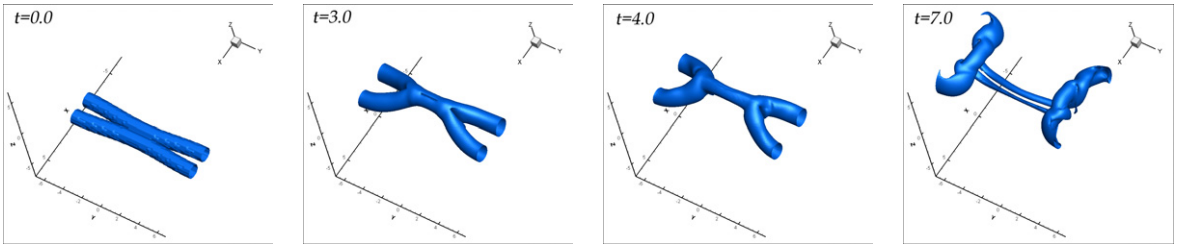

Fig. 4. Isosurface plot $|\omega| = 0.15\omega_0$ for evolution of the two antiparallel vortex tubes.

Figure 5 shows the time–dependence of the absolute maximum $|\omega_{max}|$ with the viscosity ν as a parameter. A similar time-dependence curve was reported also in [**?**]. On can notice that as ν decreases, the slope of the curve grows rapidly and the $|\omega_{max}|$ maximum is greater. This maximum appears at the end of reconnection and the start of the threading (frame $t = 4$ in figure 4).

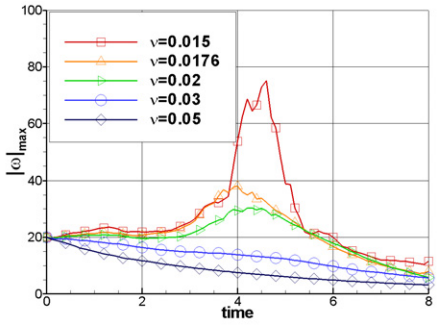

Fig. 5. Time-dependence of absolute $|\omega_{max}|$ with a viscosity as a parametr.

To see how the reconnection mixed the surrounding fluid, we used passive markers packed in a cuboid placed near the tubes (see figure 6). The reconnection intensified the mixing process because of the induced velocity component parallel to the axis of the initial vortex tubes.

3.3. Reconnection of two vortex rings

Initially, two identical circular vortex rings are set up as shown in figure 7, as in [**?**]. The radius of the rings is $R = 0.982$, the distance between their centers is $D = 3.65$, and the core radius is $a = 0.393$. A Gaussian vorticity distribution in the core was assumed, as in (24), with $\omega_0 = 23.8$, $\Gamma = 7.3$, $Re_\Gamma = 730$. The computational domain was $[4\pi \times 4\pi \times 4\pi]$, the number of nodes in each directions was $N = 128$, and $\Delta t = 0.01$. The numerical results are

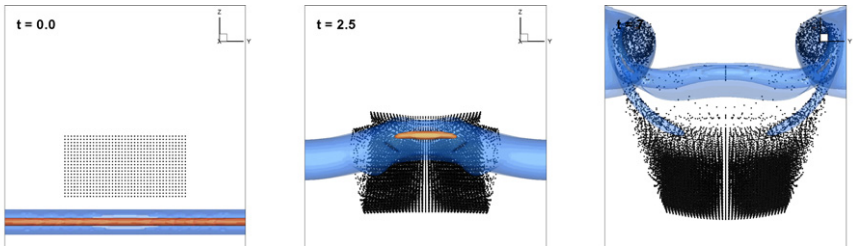

Fig. 6. Transport of passive markers by the velocity induced by two antiparallel vortex tubes seen from different points of view. To improve the visualization, in the right-most picture (perspective view) the cuboid of particles was cut in half.

presented in figure 8. The reconnection of the vortex tubes proceeds typically [**?**] (see also interesting DNS calculation [**?**]). One can observe at first the bridging, then the first and the second reconnection.

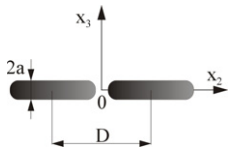

Fig. 7. Initial position of two rings seen from direction $(0, 1, 0)$.

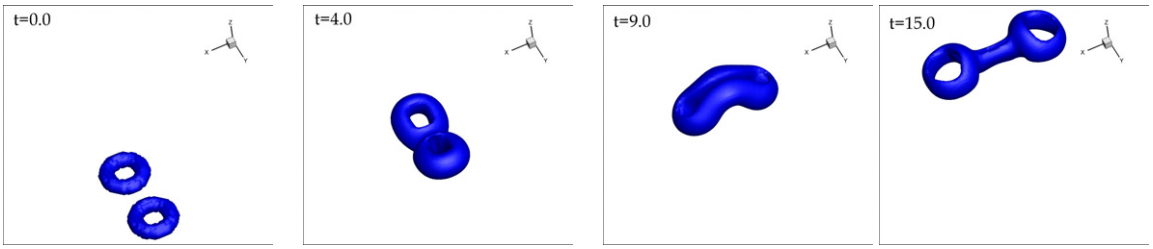

Fig. 8. Top view (left) and perspective view (right) of isosurfaces of vorticity $|\omega|$. The surfaces represent levels of 10% and 90% of instantaneous ω_{max}.

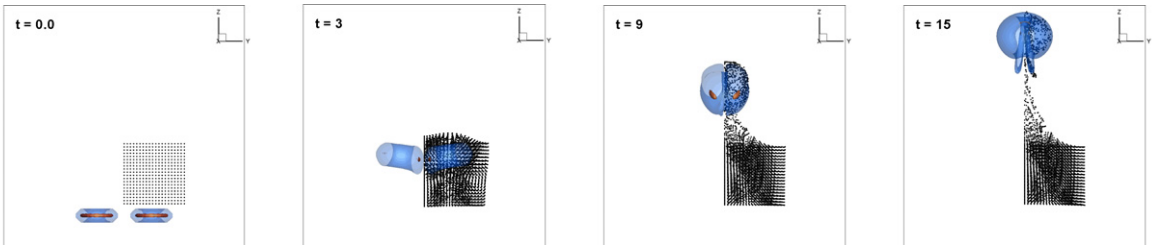

Fig. 9. Transport of passive markers by the velocity induced by the evolution of two vortex rings seen from direction $(1, 0, 0)$.

4. Closing remarks

From the results presented here, it can be seen that the reconnection process occurred in a similar manner in every test case. The common observation is that vortex reconnection takes place as a result of the cancellation of vorticity of opposite sense at the intersection of two colliding vortex tubes. Nevertheless one can notice some differences. The initial geometry of the vortex tubes influenced some details of reconnection. For example only in the vortex ring case were there two reconnections. The intensity of mixing is highest in the first case (orthogonally offset tubes).

Nowadays it may be noted that the computational power of single processors has stopped rising. Parallel architectures need to be used to speed up computation. Developing programs on GPUs is an interesting alternative to using the CPU. Thanks to hundreds of streaming processors working in parallel we can get results faster. Graphics cards are relatively cheap and easily accessible. An important element of parallel computations is choosing the right numerical algorithm to allow for the effective use of computer architecture. It is obvious that if one wants to have a good resolution of the physical phenomena, one has to use a fine numerical mesh in computations. This requires greater memory and computational time. To overcome these problems, one can use many graphics cards working in parallel. Properly used GPUs (memory management, parallel algorithms, etc.) allows programs to be executed much faster at relatively low cost.

Acknowledgements

One of the authors (A.K) was supported by a fellowship co-financed by the European Union within the European Social Fund.