FAST METHODS FOR VORTEX INFLUENCE COMPUTATION IN MESHLESS LAGRANGIAN VORTEX METHODS FOR 2D INCOMPRESSIBLE FLOWS SIMULATION

DARIA LEONOVA[1], ILIA MARCHEVSKY[1,2] & EVGENIYA RYATINA[1,2]
[1]Bauman Moscow State Technical University, Russia
[2]Ivannikov Institute for System Programming of the RAS, Russia

ABSTRACT

Vortex methods are a powerful tool for solving engineering problems of incompressible flow simulation around airfoils. The vorticity is considered as a primary computed variable. According to the Navier–Stokes equations written down in Helmholtz-type form (for 2D case), new vorticity is generated only on the surface line of an airfoil. Its intensity is unknown and can be found from the solution of the boundary integral equation resulting from the no-slip boundary condition satisfaction. The right-hand side of the integral equation in the simplest case depends on the incident flow velocity and vorticity distribution in the flow domain. For the velocity field computation, it is necessary to take into account the influence of all the vortices, which simulate the vorticity distribution in the flow. These vortices are moving in the flow with velocities calculated as sums of point-to-point vortex influences, so the computation complexity of such operation is proportional to N^2 where N is the number of vortices. In practice, N can have the order of tens or hundreds of thousands, up to a million, so the application of the "direct" method for velocities calculation becomes impossible. In this paper, two fast methods having logarithmic (proportional to $N\log N$) computational complexity are implemented and investigated. The first method is an analogue of the Barnes–Hut fast method for the N-body problem. The second one is based on fast solution of the Poisson's equation with respect to the stream function on rather coarse mesh by using the fast Fourier transform technique with some special procedure for the results correction. Numerical complexity estimations for both methods are derived. Their sequential and parallel implementations are developed. Numerical experiments show that the FFT-based method is more efficient. Total acceleration compared with the "direct" method is over 1000 times for 500,000 vortex elements.

Keywords: vortex methods, viscous incompressible media, vortex influence calculation, fast methods, Barnes–Hut-type method, Poisson equation, fast Fourier transform.

1 INTRODUCTION

In many engineering applications the problem of two-dimensional outer gas and fluid flows simulation with small subsonic speeds appears. If the flow compressibility can be neglected, the so-called vortex methods [1], [2] can be very efficient in comparison to well-known mesh methods for solving such problems, especially when we deal with large body displacements. We consider pure Lagrangian meshless modification of vortex methods, namely Viscous Vortex Domains method [3]; its main idea is considering the vorticity as a primary computed variable. The vorticity in the flow moves with the velocity which is superposition of the convective velocity and diffusive one caused by viscosity influence. It means that at every time step there are no vorticity generation in the flow domain (only the motion of existing one); new vorticity is generated on the airfoil surface line. Vorticity distribution in 2D vortex methods is simulated by a set of elementary vorticity carriers – vortex elements. Each of them is described by the position in the flow domain $\vec{r}$, small constant radius ε and circulation Γ ,which is also assumed to be constant. The airfoil surface line K can be modelled with the

WIT Transactions on Engineering Sciences, Vol 126, © 2019 WIT Press
www.witpress.com, ISSN 1743-3533 (on-line)
doi:10.2495/BE420231

free vortex sheet of unknown intensity [4], so the problem is reduced to the solution of the boundary integral equation with respect to the vortex sheet intensity $\gamma(\vec{r})$ following from the no-slip boundary condition satisfaction for the tangential velocities components [5]

$$\oint_K \frac{\vec{n}(\vec{r})\cdot(\vec{r}-\vec{\xi})}{2\pi\,|\,\vec{r}-\vec{\xi}\,|^2}\gamma(\vec{\xi})dl_{\xi} - \frac{\gamma(\vec{r})}{2} = f_{\tau}(\vec{r}) \qquad \vec{r}\in K, \tag{1}$$

where $\vec{n}(\vec{r})$ is unit normal vector to the airfoil surface line *K*. This equation has to be solved at every time step. The right-hand side of eqn (1) is a known function which depends only on the incident flow velocity $\vec{V}_{\infty}$ and vorticity distribution $\Omega(\vec{r})$ in the flow domain in the simplest case of immovable and non-deformable airfoil:

$$f_{\tau}(\vec{r}) = -\vec{\tau}(\vec{r})\cdot\left(\vec{V}_{\infty} + \int_S \frac{\vec{k}\times(\vec{r}-\vec{\xi})}{2\pi\,|\,\vec{r}-\vec{\xi}\,|^2}\Omega(\vec{\xi})dS_{\xi}\right), \tag{2}$$

where $\vec{\tau}(\vec{r})$ is unit tangent vector to the airfoil surface line, $\vec{k}$ is unit vector orthogonal to the flow plane, $\vec{k} = \vec{n}(\vec{r})\times\vec{\tau}(\vec{r})$. According to the mentioned vorticity distribution representation through a set of *n* vortex elements with circulations Γ_i, positions $\vec{r}_i$ and radius ε, the integral overflow domain in eqn (2) can be replaced by a sum over all vortex elements:

$$\int_S \frac{\vec{k}\times(\vec{r}-\vec{\xi})}{2\pi\,|\,\vec{r}-\vec{\xi}\,|^2}\Omega(\vec{\xi})dS_{\xi} \approx \sum_{i=1}^{n} \frac{\vec{k}\times(\vec{r}-\vec{r}_i)}{2\pi\left\{|\,\vec{r}-\vec{r}_i\,|^2,\ \varepsilon^2\right\}}\Gamma_i \qquad \vec{r}\in K, \tag{3}$$

After solving the integral eqn (1), all the vortex elements in the flow should be moved to new positions. Their movement is described by the system of differential equations

$$\frac{d\vec{r}_i}{dt} = \vec{V}(\vec{r}_i) + \vec{W}(\vec{r}_i),$$

where $\vec{W}(\vec{r})$ is the diffusive velocity, proportional to the viscosity coefficient [3]. This system is normally solved by Euler's explicit method. The convective velocity $\vec{V}(\vec{r}_i)$ is calculated in the similar way as the right-hand side eqn (3), but now it is necessary to calculate such a sum for all positions $\vec{r}_i$ of the vortex elements:

$$\vec{V}(\vec{r}_i) = \sum_{j=1}^{n} \frac{\vec{k}\times(\vec{r}_i-\vec{r}_j)}{2\pi\left\{|\,\vec{r}_i-\vec{r}_j\,|^2,\ \varepsilon^2\right\}}\Gamma_j, \quad i = 1,\ldots,n. \tag{4}$$

Vortex elements influences are inversely proportional to the distances to the corresponding vortex elements. So, at every time step it is necessary to calculate the vortex influence according to eqn (4) for every vortex element in the flow domain, that leads to the computational complexity proportional to n^2, where *n* is the number of vortex elements. This problem is similar to the *n*-body gravitational problem, where the calculating of the attractive forces between all the bodies is necessary.

It should be noted that vortex influence calculation is the most time-consuming operation in the whole algorithm of the vortex method; in some cases it can reach about 70% of the total execute time. In order to provide flow simulation with rather high accuracy, it is

necessary to consider large number of vortex elements – up to hundreds of thousands. Thus, the "direct" method of vortex influence calculation becomes impossible. The usage of multiprocessor systems and Nvidia graphic accelerators allows for solving this problem only partly. In Kuzmina et al. [6] it is shown that parallel technologies provide significant acceleration of the whole algorithm, especially using combined technologies MPI+CUDA. However, for some problems time of computations remains unacceptably high, for example, in case when the number of vortices exceeds about 100 000.

The aim of this research is implementation of the fast algorithms for vortex influence calculation permitting one to reduce the computational complexity of this operation. Two fast methods of logarithmic (proportional to $n \log n$) computational complexity for vortex influence calculation are considered: the Barnes–Hut-type method and the method based on fast solution of the Poisson's equation with respect to the stream function on rather coarse mesh using fast Fourier transform technique with further solution correction.

2 THE BARNES–HUT-TYPE METHOD

The Barnes–Hut method [7] initially had been developed for the gravitational *n*-body problem and later adapted to vortex methods by Dynnikova [8]. The main idea of this method is that the influence of the groups of closely adjacent vortex elements on another such groups located far apart, can be calculated approximately using linearized formulae [8].

For this purpose, in the flow domain the hierarchical tree-structure of rectangular space domains (cells), containing all vortex elements, is constructed. The zero-level cell contains all vortex elements, and then it is divided across its long side into two first-level cells, each is reduced horizontally and vertically according to its vortices in order to exclude empty area. Next, similarly the second-level cells are constructed, etc. (Fig. 1). Such procedure continues until the target level is achieved or the cell contains single vortex.

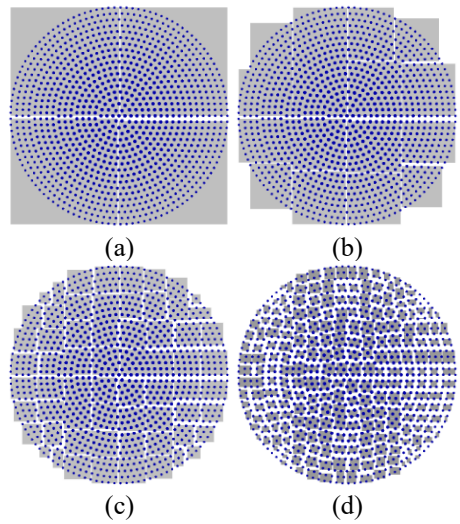


Figure 1: Cells of the (a) 2nd; (b) 4th; (c) 6th; and (d) 8th levels.

The set of terminal vertices of the resulting tree contains all the vortex elements and for each cell the tree traversal is carried out according to the scheme in Fig. 2. Moreover, the

transition to lower level cells (red arrows) is carried out only if the "proximity criterion" for the current cell is satisfied (the distance between cells centers is compared with the sum of their sides).

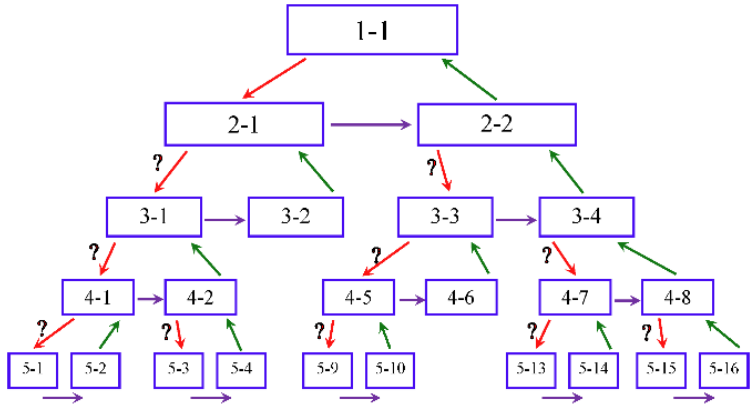


Figure 2: Tree traversal scheme.

So, the Barnes–Hut-type algorithm of vortex influence computation at every time step of flow simulation consists of the following stages:

1. Zero-level cell formation for all vortex elements existing at the current time step.
2. Tree structure construction; determination of the terminal vertices.
3. Calculation of the properties for every tree-cell such as centers of positive and negative vorticity and total circulations (also positive and negative).
4. For every terminal tree cell 3 operations are performed:
   a. tree traversal in order to determine far-spaced cells from the current one and calculation of the coefficients in linearized representation of the vortex influence;
   b. calculation of the vortex influence from closely adjacent vortex elements (placed in the so-called "neighboring" zone) according to the direct eqn (4);
   c. accumulation of the influences calculated approximately (a) and exactly (b).

The "proximity criterion" used for "neighboring" cells determination depends on the adjustable parameter θ (usually $0<\theta\leq 1$) which determines the accuracy of the method. For small values of this parameter ($\theta \ll 1$) the Barnes–Hut method provides high accuracy, but it also has high computational complexity; increase of the parameter θ leads to decreasing both the computational complexity and accuracy of the method. Numerical experiments show, that the value of the parameter θ should not exceed 0.2…0.4 in order to provide the relative error at the level of 0.2%.

It should be noted, that number of tree levels k is also an adjustable parameter; it practically does not influence the accuracy of the method, but the computational complexity of the algorithm depends on its value significantly – deviation on 2–3 levels from the optimal value leads to increase of calculation time in 2 or more times. In Table 1, the relative time of the Barnes–Hut-type algorithm execution is given for the problems with large number of vortex elements (n = 100,000, n = 250,000, n = 500,000, n = 1,000,000), where the ratio of

times of computations is given for non-optimal and optimal numbers of tree levels (denoted as k^*); in brackets the acceleration of the algorithm compared to the "direct" Biot–Savart method (eqn (4)) is shown. Note, that hereinafter the algorithm execution time means the vortex influence calculation at one time step.

Table 1: Time of computations ratio for the Barnes–Hut method at optimal and non-optimal number of tree levels and different number of vortices.

Relative calculation time t_k/t_{k^*} (k^* is an optimal number of tree levels)							
n = 100,000 k^* = 14		n = 250,000 k^* = 15		n = 500,000 k^* = 16		n = 1,000,000 k^* = 17	
K	t_k/t_{k^*}	k	t_k/t_{k^*}	k	t_k/t_{k^*}	K	t_k/t_{k^*}
12	1.76	13	2.02	14	1.78	15	1.91
13	1.20	14	1.19	15	1.23	16	1.10
14	1.00 (22.3)	15	1.00 (40.5)	16	1.00 (64.8)	17	1.00 (92.8)
15	1.36	16	1.17	17	1.25	18	1.24
16	2.18	17	1.92	18	2.08	19	2.11

In order to estimate parallel properties of the algorithm (its scalability), taking in mind the possibility of its parallel version development, in Table 2 the contributions in percent of the above mentioned main operations to the total execution time of the Barnes–Hut algorithm are shown for the model problem with n = 1,000,000 vortex elements and different numbers of tree levels k. Here the operation *1* is the operation of tree construction (stage 2), operation *2* is coefficients calculation (stage 4a), operation *3* is direct influence calculation in neighboring zone (stage 4b), t_1 is calculation time for the Barnes–Hut method, t_2 is calculation time for the direct method, s is the achieved acceleration.

Table 2: Contributions of the main operations in the Barnes–Hut algorithm at different number of tree levels.

Levels	Operation *1*	Operation *2*	Operation *3*	Other ops.
15	0.95%	2.82	96.22	0.01
16	1.69%	11.58	86.70	0.03
17	1.90%	30.59	67.44	0.07
18	1.57%	60.61	37.77	0.05
19	1.00%	78.53	20.43	0.04

It is seen that with increasing number of levels the point-to-point stage of calculations (Operation *3*) is decreased, while that the stage of coefficients calculation (Operation *2*) increases.

The parallel version of the Barnes–Hut method for vortex influence computation was implemented using both MPI (Message Passing Interface, for cluster systems) and OpenMP (Open Multi-Processing, for systems with shared memory) technologies. Taking into account that stages 4a–4c of the Barnes–Hut algorithm can be executed independently for the terminal tree cells, the simplest way of parallelization is to split the terminal cells onto MPI-processes, including additional parallelization using OpenMP technology for each process. Due to small contribution of the tree construction procedure (less than 2%, according to Table 2), it is not

parallelized and executed simultaneously by all the MPI processes. The achieved values of acceleration are shown in Table 3 for the problem with large number of vortex elements (n = 1,000,000) for one, two and three cluster nodes. It is seen, that the most efficient is hybrid (MPI + OpenMP) parallelization.

Table 3: Acceleration results using MPI and OpenMP technologies for the Barnes–Hut algorithm.

Number of nodes	1	2	3
1 OpenMP thread per node (per MPI process)			
Time, sec	106.13	54.02	36.67
Acceleration, times	1.00	1.96	2.89
4 OpenMP threads per node (per MPI process)			
Time, sec	30.84	16.82	12.11
Acceleration, times	3.44	6.31	8.76
4 MPI processes per node			
Time, sec	33.58	18.20	13.17
Acceleration, times	3.16	5.83	8.06

Taking into account, that the optimal variant of the Barnes–Hut method (for 17 tree levels) provides 92,8 times acceleration in comparison to the direct method in sequential mode (Table 2), now we obtain, that its parallel version, being run on 3 nodes, provides total acceleration of about 810 times. Thus, the total time of computation can be reduced from approximately 165 minutes (2.75 hours) to about 12 seconds.

However, despite the similarity the problem of vortex influence calculation and n-body gravitational problem, there is a significant difference. In gravitational problem due to 3D problem statement, the attractive force, caused by one body decreases proportionally to the squared distance, whilst the two-dimensional problem statement in vortex methods leads to influence decreasing with the first degree of the distance only, so the efficiency of the Barnes–Hut algorithm in vortex method is lower in comparison to the original gravitational problem. Thus, it is interesting to investigate other possible ways to approximate fast vortex influence calculation.

3 THE FFT-BASED METHOD

This method is based on the similar idea as the Barnes–Hut method: the influence of the vortex elements, placed if far-spaced regions can be calculated approximately. Such influence can be calculated fast due to the possibility of the Poisson's equation fast solution with respect to the stream-function using fast Fourier transform (FFT) technique [9].

The velocity can be represented through the stream-function as $\vec{V}(\vec{r}) = \nabla \times \left(\psi(\vec{r})\vec{k}\right)$, where the stream-function ψ satisfies the Poisson's equation with known right-hand side

$$\Delta \psi = -\Omega(\vec{r}). \tag{5}$$

The solution of such equation can be represented as the convolution of the Green's function $G(\vec{r})$ and the right-hand side, so for the velocity $\vec{V}(\vec{r})$ we obtain the following eqn:

$$\vec{V}(\vec{r}) = \nabla \times \left(\psi(\vec{r})\vec{k}\right) = \int_S \left(\nabla \times G(\vec{r}-\vec{\xi})\vec{k}\right)\Omega(\vec{\xi})dS_\xi = \\ = \int_S \frac{\vec{k}\times(\vec{r}-\vec{\xi})}{2\pi \mid \vec{r}-\vec{\xi} \mid^2}\Omega(\vec{\xi})dS_\xi = \int_S \vec{Q}(\vec{r}-\vec{\xi})\,\Omega(\vec{\xi})dS_\xi, \quad (6)$$

If rectangular uniform mesh is introduced, which contains $M_x \times M_y$ nodes in horizontal and vertical directions, respectively, the integral in eqn (6) can be approximately replaced with a sum over all the mesh nodes:

$$\int_S \vec{Q}(\vec{r}-\vec{\xi})\Omega(\vec{\xi})dS_\xi \approx \sum_{i=0}^{M_x-1}\sum_{j=0}^{M_y-1}\vec{Q}(\vec{r}-\vec{r}_{ij})\Gamma_{ij}, \quad (7)$$

where Γ_{ij} are the nodal values of vorticity, i.e., the vorticity should be projected somehow onto the nodes in such a way, that the following condition is satisfied:

$$\int_S \Omega(\vec{\xi})dS_\xi = \sum_{i=0}^{M_x-1}\sum_{j=0}^{M_y-1}\Gamma_{ij}. \quad (8)$$

We consider rather coarse mesh, that means $M_x M_y \ll n$, and now describe briefly the main stages of the FFT-based fast method of vortex influence calculation:

1. To calculate the nodal circulations Γ_{ij} using some smoothing operator, for example, introduced in Monaghan [11].
2. To calculate vector kernel components
$$\vec{Q}(\vec{r}) = \frac{\vec{k}\times\vec{r}}{2\pi \mid \vec{r} \mid^2} = \{Q_x, Q_y\}$$
at mesh nodes. For the origin point $\vec{r} = \{0, 0\}$ we assume $\vec{Q}(\vec{r}) = \{0, 0\}$. Nodal values of Q_x and Q_y form matrices Q^x and Q^y, respectively.
3. To quadruple the mesh domain is by its extension to the right and upper directions (reflection technique) in order to satisfy the boundary conditions (zero velocity on infinity). In additional nodes the matrices Q^x and Q^y are extended symmetrically and skew-symmetrically [10], circulation values Γ_{ij} are assumed to be zero. All following stages should be performed for obtained extended matrices.
4. To perform two-dimensional fast Fourier transform [9] for the abovementioned matrices Q^x, Q^y, Γ; the resulting complex-valued images are $\hat{Q}^x$, $\hat{Q}^y$, $\hat{\Gamma}$.
5. According to the well-known feature of the FFT, the nodal values of eqn (7) image form the matrix, which components are calculated as elementwise multiplications of two matrices, so as a result we obtain two matrices, consist of the images of the velocity components nodal values: $\hat{V}^x = \hat{Q}^x \circ \hat{\Gamma}$ and $\hat{V}^y = \hat{Q}^y \circ \hat{\Gamma}$.
6. To perform inverse fast Fourier transform procedure and obtain matrices of nodal values of horizontal and vertical velocity components V^x and V^y.
7. To interpolate the velocity nodal values onto vortices placed in the corresponding mesh cells.

Due to coarse mesh usage, a significant error occurs. In particular cell it is caused mainly by significant error in influence calculation from the vortex elements, placed in this cell itself and in the neighboring zone, so the correction procedure should be performed. It is necessary to exclude the "incorrect" influence calculated at the Poisson's equation solution only from

the vortices from neighboring zone and add the "correct" influence, calculated point-to-point using eqn (4). So, the following algorithm [10] should be performed for each mesh cell.

1. Calculation of the correction matrix:
   a) the vortex of unit circulation is placed in arbitrary mesh cell;
   b) using the same smoothing operator, which have been used for nodal values Γ_{ij} calculation, its circulation is redistributed onto 16 nearest nodes (black points in Fig. 3), the resulting nodal circulations vector is $\{\Gamma\}$;

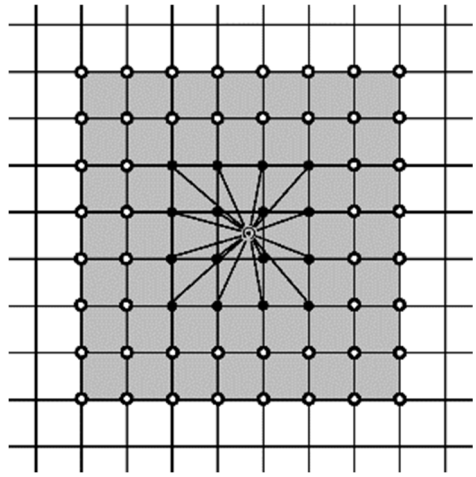


Figure 3: Neighboring zone of the cell.

   c) the steps 4–6 of the basic algorithm are performed for this "sub-problem"; as a result we obtain the velocities at 64 nearest nodes (black and white points in Fig. 3). Due to 2D problem statement, the resulting vector $\{V\}$ has 128 components;
   d) based on assumption of linear dependence between velocity and circulation vectors we obtain the matrix equality $\{V\} = [C]\{\Gamma\}$, where $[C]$ is correction matrix having size 128×16;
   e) the 15 similar sub-problems are considered in order to construct the system of equations for correction matrix components; note, that for every sub-problem the unit vortex is placed in different position to avoid a singular matrix.
2. After constructing a correction matrix $[C]$, the influences of the vortices placed in the neighboring zone of each cell can be excluded. For this purpose, for each cell the circulations of the vortices contained in it are redistributed onto 16 nearest nodes; the resulting vector is $\{V\} = [C]\{\Gamma\}$ which means the current cell influence on its neighboring zone. Corresponding nodal velocities are kept save for each cell and they are summarized with analogous influences from other cells in neighboring zone (for the cells, which do not refer to the neighboring zone such influence will be equal to zero). Thus, for all the cells we can exclude incorrectly calculated influence of the vortices places in the neighboring zones.
3. For each cell now it is necessary to add the correctly (exactly) calculated vortex influence from its neighboring zone according to the Biot–Savart law (eqn (4)):

$$\vec{V}_{neigh}(\vec{r}_i) = \sum_{j \in B_i \cup B_i'} \frac{\vec{k} \times (\vec{r}_i - \vec{r}_j)}{2\pi \left\{ |\vec{r}_i - \vec{r}_j|^2, \varepsilon^2 \right\}} \Gamma_j \qquad i \in B_i, \tag{9}$$

where B_i means the set of vortices in the *i*-th cell; B_i' means the set of vortices in its neighboring zone to the *i*-th cell (not including the *i*-th cell itself).

3.1 Numerical experiment

The model problem with 500 vortex elements was considered (Fig. 4). The mesh 16 x 16 was considered for the numerical solution of the Poisson's equation vortices circulations were chosen arbitrary ($-0.01 \le \Gamma_i \le 0.01$).

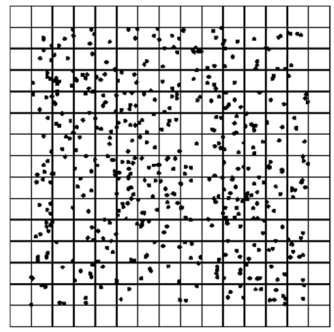


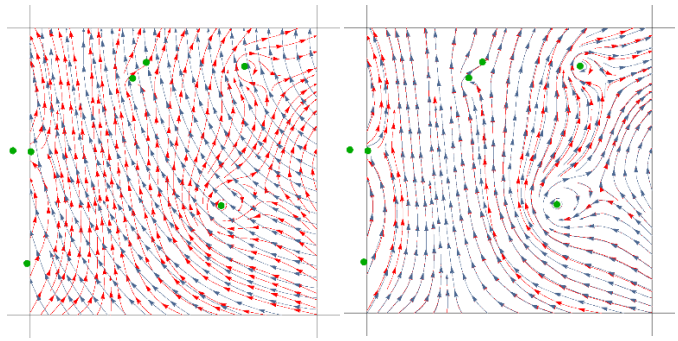
Figure 4: Model problem statement: vortex elements distributed uniformly in the unit square.

At Fig. 5 the streamlines of the velocity field are shown for the central mesh cell before the correction procedure performing (Fig.5 (a)) and after this procedure (Fig. 5(b)). Here the streamlines, obtained using the direct eqn (4), are shown in red color; the streamlines obtained using the FFT-based method are shown in blue color. It is apparent that velocities obtained using the fast solution of the Poisson's equation (without correction) are quite different from the real ones calculated directly point-to-point, while the correction permits to achieve quite accurate results.

Calculation for the vortices in the central cell:

$$\delta V_i = \frac{|\vec{V}_{i,exact} - \vec{V}_{i,FFT}|}{\max\limits_j |\vec{V}_{j,exact}|} \qquad i, j = 1, \ldots, n_{cell}. \tag{10}$$

A quantitative empirical estimation was obtained also for the accuracy of velocities where δV_i is relative error the velocities calculation; $\vec{V}_{i,exact}$ – exact value of the velocity of the *i*-th vortex element, calculated "directly" according to eqn (4); $\vec{V}_{i,FFT}$ – approximate value, calculated using the FFT-based method. The maximum in the denominator is calculated over the vortices in the central cell. Numerical results for different sizes of the neighboring zone, i.e., numbers of cell layers around each cell, are shown in Table 4. Zero number of cell layers corresponds to the algorithm without correction stage.



(a) (b)

Figure 5: Velocities streamlines in the central mesh cell. (a) Before correction procedure; and (b) After it.

Table 4: Relative error of the FFT-based method.

Number of cell layers	0	1	2	3	4
$\max \delta V_i$	0.2840	0.0326	0.0024	0.0017	0.0003

It is seen, that the error of the algorithm without correction exceeds 20%, which makes the algorithm inapplicable in practice. Three cell layers in the neighboring zone (i.e., the neighboring zone in this case consists of 49 cells including the central cell itself) provides the error level less than 0.2%. Further neighboring zone size increase provides smaller error: 0.03% for 4 cell layers, however the neighboring zone here consists of 81 mesh cells that leads to significant growth of the algorithm computational complexity.

At the same time number of numerical experiments, performed in order to investigate the properties of the vortex methods algorithms, shows the error level of 0.2% is acceptable for the majority of problems, being solved by vortex methods. This level of the error is comparable with the error, arising from the boundary integral equation discretization, which determines the vorticity generation on the airfoil surface line. Therefore, it seems to be optimal from point of view of the ratio between the method accuracy and computational time to choose the size of the neighboring zone to be equal 3 cell layers, as it is shown in Fig. 3.

3.2 Complexity estimation of the algorithm

Note, that the accuracy of the FFT-based method practically does not depend on the mesh size, used for solving of the Poisson's problem, however it influences significantly the computational complexity of the method in the whole.

In order to estimate the computational complexity, we assume that n vortex elements are distributed uniformly in the square domain, mesh cells are squares, and there are equal numbers of the mesh nodes (equal to M) in horizontal and vertical directions. We assume additionally, that there are no vortex elements in the outer layer of the cells, similarly to shown in Fig. 4: this permits to use known smoothing kernel M_4 [11] for vorticity redistribution from the vortex elements onto mesh nodes and satisfy eqn (8).

Taking into account only the operations of multiplication and division, it is possible to estimate the complexities of the following algorithm stages:

1. Circulations calculation at mesh nodes (redistribution) and correction procedure.
2. Fast Fourier Transform and inverse Fourier transform; we take into account, that the FFT-based algorithm is implemented at every time step, so for the fixed mesh it is possible to compute matrices Q^x, Q^y and perform 2D FFT for them (resulting matrices $\hat{Q}^x$, $\hat{Q}^y$) only once. Thus, at every time step the FFT should performed only for matrix Γ and two inverse transformations should be performed for matrices $\hat{V}^x$ and $\hat{V}^y$ in order to obtain nodal values of velocity components.
3. Interpolation procedure from the mesh nodes onto the vortex elements.

The theoretical complexities estimations are the following:

$$\left.\begin{aligned} Q_1 &= 192n + 2048(M-2)^2 \\ Q_2 &= 8M^2(7 + 6\log_2 M) \\ Q_3 &= 9n + 245n^2(M-2)^{-2} \end{aligned}\right\}. \tag{11}$$

It is also important to estimate the mentioned operations contributions to the method complexity; it significantly depends on the number of mesh nodes. With nodes number increasing the contributions Q_1 and Q_2 decrease, while Q_3 increases significantly. As an example, we consider the problem with n = 1 000 000 vortex elements and numbers of mesh nodes M = 128 and M = 512. The nodes numbers are chosen equal to power of two (2^k) because that makes it possible to perform the FFT procedure optimally. The comparison of the numerical results (blue columns) and theoretical estimations (red columns) of the operations Q_1, Q_2 and Q_3 contributions are shown in the diagrams in Fig. 6. A good agreement between the numerical results and theoretical estimations is seen.

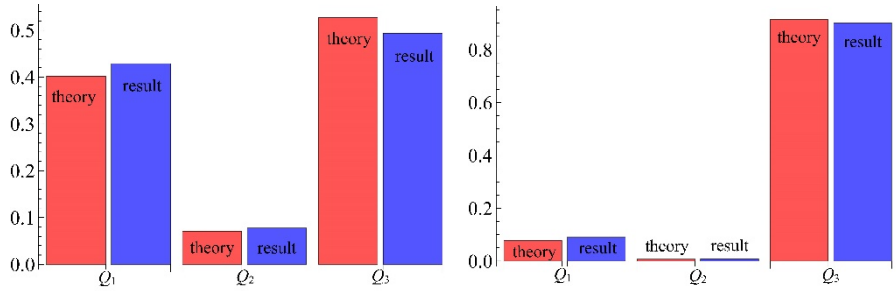


Figure 6: The main operations contributions in the FFT-based algorithm for the meshes consist of M = 128 nodes (a) and M = 512 nodes (b) in both directions.

The parallel version of the FFT-based algorithm was implemented using both MPI and OpenMP technologies, similarly to the above discussed parallelization scheme for the Barnes–Hut method. Note, that in the present implementation the FFT transformation (direct and inverse) procedure is performed sequentially, by using the Eigen library (http://eigen.tuxfamily.org). The achieved values of acceleration are shown in Table 5 for the problem with large number of vortex elements (n = 1,000,000) for one, two and three cluster nodes.

Table 5: Acceleration using MPI and OpenMP technologies for the FFT-based algorithm.

Nodes number	1	2	3
1 OpenMP thread per node (per MPI process)			
Time, sec	4.52	2.82	2.24
Acceleration, times	1.00	1.60	2.02
4 OpenMP threads per node (per MPI process)			
Time, sec	2.12	1,64	1.41
Acceleration, times	2.13	2.76	3.21
4 MPI processes per node			
Time, sec	1.79	1.48	1.40
Acceleration, times	2.53	3.05	3.23

Thus, the parallel implementation of the FFT-based method, being run on 3 nodes provides total acceleration of about 7000 times, that means that the total time of computation can be reduced from approximately 165 minutes (2.75 hours) to about 1.4 seconds. Note, that MPI parallelization is more efficient in comparison to hybrid (MPI + OpenMP) approach.

4 COMPARISON OF THE TWO FAST METHODS

In order to compare the efficiency of two fast methods for vortex influence calculation, four different problems with large number of vortex elements (n_1 = 100,000, n_2 = 250,000, n_3 = 500,000, n_4 = 1,000,000) were considered. The values of the adjustable parameters of both methods (coefficient θ in proximity criterion and number of tree levels k in the Barnes–Hut method; number of cell layers in the neighboring zone and number of nodes in the mesh in the FFT-based method) were chosen optimally; the relative error (10) in all cases was less than 0.2%. In Table 6 time of computations in seconds is given for all the mentioned problems (all the computations are performed in sequential mode).

Table 6: Time of computations (in sec.) for all the methods for different number of vortices.

	n_1 = 100,000	n_2 = 250,000	n_3 = 500,000	n_4 = 1,000,000
Biot-Savart (direct) method	98.5	615.6	2462.2	9848.9
Barnes–Hut method	4.43	15.21	38.02	106.13
FFT-based method	0.47	1.10	2.47	4.52

The achieved acceleration is shown in Table 7 in comparison to the direct method (computations according to the Biot–Savart law (eqn 4)). It is clearly seen that the FFT-based method is much more efficient in comparison to the Barnes–Hut-type method; for example, time of computations is nearly the same for $n_1 = 100\,000$ using the Barnes–Hut method and for $n_4 = 1\,000\,000$ using the FFT-based method.

Table 7: Fast methods acceleration in times.

	n_1 = 100,000	n_2 = 250,000	n_3 = 500,000	n_4 = 1,000,000
Barnes–Hut method	22	40	65	93
FFT-based method	210	560	997	2179

5 CONCLUSIONS

In the present paper two different methods of vortex influence computation were implemented and investigated. Both methods have the logarithmic ($n \log n$) computational complexity and can be efficiently applied in case the number of vortex elements n is about few hundreds of thousands, up to one million. Both methods have some adjustable parameters, such as the coefficient in the proximity criterion in the Barnes–Hut method and the number of cell layers in the neighboring node in the FFT-based method, which effect the ratio between the accuracy and time of computations. The other parameters – number of tree levels in the Barnes–Hut method and mesh size in the FFT-based method practically do not affect the accuracy, but their values determine the computational complexity of the corresponding algorithms. It is found, that the FFT-based method is at least 10 times more efficient in comparison to the Barnes–Hut method. Parallel implementations of both methods are developed using OpenMP and MPI technologies; their efficiencies are investigated for a small number of processors/cores.

ACKNOWLEDGEMENTS

The research was supported by the Russian Foundation for Basic Research (project No. 18-31-20051, "Development of new algorithms based on vortex methods for flow around airfoils simulation and coupled hydroelastic problems solving and their efficient software implementation for multiprocessor computers of different architectures").

REFERENCES

[1] Lifanov, I.K., *Singular Integral Equations and Discrete Vortices*, Utrecht, p. 475, 1996.

[2] Cottet, G.H. & Koumoutsakos, P.D., *Vortex Methods: Theory and Practice*, Cambridge University Press, p. 328, 2000.

[3] Andronov, P.R., Guvernyuk, S.V. & Dynnikova, G.Y., *Vortex Calculation Methods of Non-Stationary Hydrodynamical Loadings*, Moscow University Press, p. 184, 2006.

[4] Tokaty, G.A., *A History and Philosophy of Fluid Mechanics*, Courier Corporation, p. 241, 1994.

[5] Kempka, S.N., Glass, M.W., Peery, J.S., Strickland J.H. & Ingber, M.S., Accuracy considerations for implementing velocity boundary conditions in vorticity formulations. Sandia report, Sand96-0583 UC-700, p. 52, 1996.

[6] Kuzmina, K.S., Marchevsky, I.K. & Ryatina, E.P., On CPU and GPU parallelization of VM2D code for 2D flows simulation using vortex method. *Proceedings of the 6th European Conference on Computational Mechanics (ECCM 6) and 7th European Conference on Computational Fluid Dynamics (ECFD 7)*, 2018, www.eccm-ecfd2018.org/admin/files/filePaper/p2045.pdf. Accessed on: 10 Mar. 2019.

[7] Barnes, J. & Hut, P., A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, **324**(4), pp. 446–449, 1986.

[8] Dynnikova, G.Y., Fast technique for solving the N-body problem in flow simulation by vortex methods. *Computational Mathematics and Mathematical Physics*, **49**(8), pp. 1389–1396, 2009.

[9] Nussbaumer, H.J., *Fast Fourier Transform and Convolution Algorithms*, Springer Verlag, p. 286, 1982.

[10] Morgenthal, G. & Walther, J.H., An immersed interface method for the Vortex-In-Cell algorithm. *Computers and Structures*, **85**, pp. 712–726, 2007.

[11] Monaghan, J.J., Extrapolating B-splines for interpolation. *Journal of Computational Physics*, **60**(2), pp. 253–262, 1985.