Applied Computer Science, vol. 17, no. 4, pp. 100–109
doi: 10.23743/acs-2021-32

Submitted: 2021-07-12 | Revised: 2021-08-14 | Accepted: 2021-09-20

Keywords: computer clusters, parallel computing, n-body problem

Tomasz NOWICKI [0000-0003-0752-2509]*,
Adam GREGOSIEWICZ [0000-0002-6702-8505]**,
Zbigniew ŁAGODOWSKI [0000-0003-1811-6151]**

PRODUCTIVITY OF A LOW-BUDGET COMPUTER CLUSTER APPLIED TO OVERCOME THE N-BODY PROBLEM

Abstract

The classical n-body problem in physics addresses the prediction of individual motions of a group of celestial bodies under gravitational forces and has been studied since Isaac Newton formulated his laws. Nowadays the n-body problem has been recognized in many more fields of science and engineering. Each problem of mutual interaction between objects forming a dynamic group is called as the n-body problem. The cost of the direct algorithm for the problem is $O(n^2)$ and is not acceptable from the practical point of view. For this reason cheaper algorithms have been developed successfully reducing the cost to O(nln(n)) or even O(n). Because further improvement of the algorithms is unlikely to happen it is the hardware solutions which can still accelerate the calculations. The obvious answer here is a computer cluster that can preform the calculations in parallel. This paper focuses on the performance of a low-budget computer cluster created on ad hoc basis applied to n-body problem calculation. In order to maintain engineering valuable results a real technical issue was selected to study. It was Discrete Vortex Method that is used for simulating air flows. The presented research included writing original computer code, building a computer cluster, preforming simulations and comparing the results.

1. INTRODUCTION

The n-body problem arises occasionally in physics and thus also in engineering. The prerequisite for its emergence is (1) description of a physical phenomenon by means of a dynamic and discrete set of particles, which (2) influence mutually in the relationship "each with everyone". Computer modelling of physical phenomena in this way is simple and so attractive from an engineering point of view. However, the simplicity and purity of the method carries time–consuming calculations resulted from the necessity of recalculating all

* Lublin University of Technology, Faculty of Electrical Engineering and Computer Science, Department of Computer Science, Poland, t.nowicki@pollub.pl
** Lublin University of Technology, Faculty of Electrical Engineering and Computer Science, Department of Mathematics, Poland, a.gregosiewicz@pollub.pl, z.lagodowski@pollub.pl

the mutual interactions at every step of the simulation. Engineers who applies n-body particle models in their practice always face the challenge of time-intensive computer calculation. It also should be noted that various particle models and methods (those in which the n-body problem occurs) are always accompanied by other additional complications that forms the individual computational specificity of them. Research presented in this article focuses on Discrete Vortex Method (DVM). The authors tried to answer the question what the efficiency of low-budget computer clusters can be when applied to DVM simulations.

1.1. The generalized n-body problem

Firstly, the n-body problem is going to be formulated in the simplest and general way (Fig. 1). For this purpose one should:

- define a metric space with a metric d and supply it with time t,
- spread at $t = 0$ a finite set of particles (called a discrete population or a discrete system) numbered $i = 1, 2, 3, \ldots, n$ in the space by determining their initial positions P_i and velocities u_i,
- abstract one common attribute of the particles, which intensity C determines the strength of mutual influence,
- chose a mutual influence function Q, which lets calculate influence from the particle j on i $Q_{ij} = Q(C_i, C_j, d_{ij})$,
- choose a velocity function u which lets calculate change in the velocity of the particle i $\Delta u_i = u(u_i, C_i, Q_1, Q_2, Q_3, \ldots, Q_n)$,
- choose a displacement function D, which lets calculate the change in a particle position after time Δt: $\Delta d_i = D(u_i, C_i, \Delta t)$,
- let the particles change their positions with time.

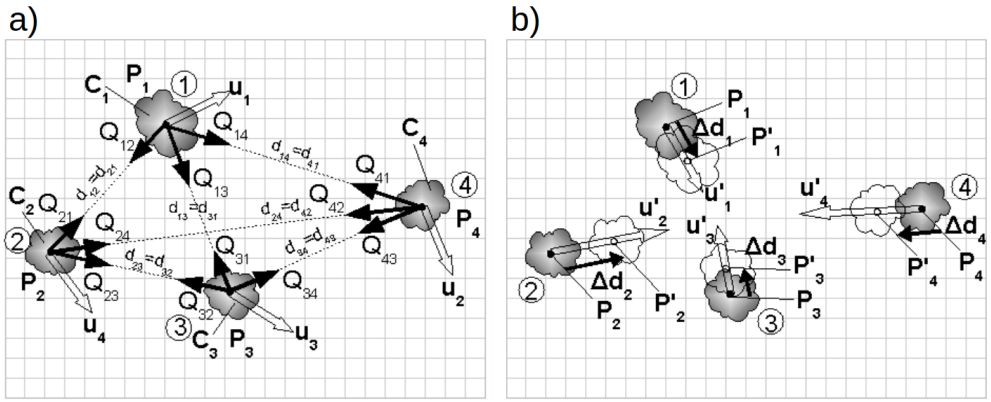

Fig. 1. A four-element discrete population (n = 4) in the "each to everyone" relation: a) the initial configuration, b) the change under mutual interactions; the description: i = 1, 2 ,3, 4 – particle numbers, C_i – the intensity of a attribute , P_i P_i' – the location in the space (before and after the change), u_i u_i' – velocity, Q_{ij} – the influence on particle i from j, Δd_i – the location change

Such a discrete population constantly reconfigures itself as time runs. All the particles continually influence each other and move in the space due to the influence. The movement results in a change of the mutual influences. The population equilibrium may or may not be achieved. Such a "numeric ecosystem" has the ability to reproduce a real phenomenon if constructed and interpreted in a correct way.

The n-body problem is considered resolved when is known the configuration of the population (positions and velocities of all the particles) at any time $t > 0$. It turns out that the n-body problem has in general no analytical solution. Subsequent configurations of the population may be determined only by direct simulations. There are three main groups of algorithms for n-body problem (Hockney & Eastwood, 1988): 1) particle-particle (P^2), 2) particle-mesh (PM) and 3) particle-particle-particle-mesh (P^3M). The P^2 algorithm is the simplest one and consists in calculating all single interactions in a direct way. It results in the numerical cost of $O(n^2)$, which is usually unacceptable in engineering practice. The PM algorithm employs a calculating mesh, which let reduce the cost to O(n) but also decreases accuracy of the results because small-scale local effects are not able to develop. The last P^3M algorithm is a combination of the previous two. For each particle a direct neighbourhood area is established within which the P^2 algorithm is used to calculate the influence from other particles from the neighbourhood. The influence from the remote particles are determined using the P^3M algorithm. The numerical cost of the last algorithm is $O(n \cdot ln(n))$. It is most frequently implemented and was used in the presented research.

1.2. The Discrete Vortex Method

The Discrete Vortex Method (DVM) (Lewis, 1991; Cottet & Koumoutsakos, 2000) is one of methods dedicated to computer simulating of turbulent fluid flows. The method was originated in the thirties of the twentieth century and has been applied successfully to fluid mechanics since then (Fig. 2).

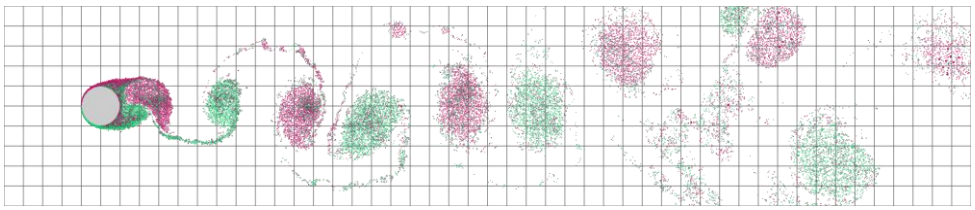

Fig. 2. Turbulent air flow over a cylinder with correctly developed vortex street as an example of DVM in action (Nowicki, 2012)

DVM is a numerical method developed for solving the Navier-Stokes equation (N-S) based on the Lagrangian model of a particle tracing. In DVM, the equation is solved by a direct computer simulation of a physical phenomena. A finite mesh known from finite element or finite volume methods is not applied in DVM. Artificial models of turbulence such as LES or k-ε are also not used. The most valuable feature of the method is its self-adaptability to geometry of computational task (Nowicki, 2015) and numeric stability. The biggest drawback of DVM is time consuming simulations come from the n-body problem.

Considering 2D euclidean areas of fluid flow and assuming a homogeneous dry air with a constant density, the following form of the N-S equation can be used to describe the phenomenon under interest:

$$\frac{\partial u}{\partial t} + (u\nabla)u = -\frac{1}{\rho}\nabla p + \nu\nabla^2 u \tag{1}$$

where: u – velocity field,
$(u\nabla)$– operator of the material derivative,
p – pressure field,
ρ – density of air,
ν – kinematic viscosity of air,
t – time.

Eq. (1) can be decomposed by calculating the rotation of the vector $\boldsymbol{u}$, which gives the so-called vorticity transport equation:

$$\frac{\partial \omega}{\partial t} + (u\nabla)\omega = \nu\nabla^2\omega \tag{2}$$

where: $\omega = \nabla \times u$ – vorticity field of the flow (treated as scalar for 2D flows).

The last eq. (2) is composed of two components: advection (3) and diffusion (4):

$$\frac{\partial \omega}{\partial t} + (u\nabla)\omega = 0 \tag{3}$$

$$\frac{\partial \omega}{\partial t} = \nu\nabla^2\omega \tag{4}$$

The separation (known as *Split Algorithm*) lets us treat the fluid flow as two simultaneous and independent phenomena: advection and diffusion, wherein only advection eq. (3) describes the vortex kinematics that leads to the n-body problem.

In DVM the computational particle is a discrete vortex. The abstracted attribute of the particle is its vorticity traditionally denoted by the letter Γ. The vorticity equals the value of circulation of velocity field over a contour (with element $\boldsymbol{dr}$) of an area from which the vorticity is reduced to a single point:

$$C = \Gamma = \oint_L u\, dr \tag{5}$$

The mutual influence function Q from particle j on i is given by a formula:

$$Q_{ij} = \Gamma_j \cdot K \times d_{ij} \tag{6}$$

where: $d_{ij} = P_j - P_i$- distance between vortexes as the metrics,
K – kernel articulating inverse-square law.

Since the influence function in DVM describes velocity field, there is no need to introduce an extra velocity function and:

$$u_i = \sum_{j=1,2,3,\ldots,n \wedge i \neq j} Q_{ij} \tag{6}$$

After the short glimpse of DVM given above it should be clear that the method incorporates the n-problem method.

1.3. Literature review

The results presented in this paper concern a numerical experiment carried out in 2007 (Nowicki, 2007). The aim of that experiment was to determine the performance of a low-cost computer cluster dedicated to DVM simulations. At the time, the method was in its early stage of development and such data was lacking. Today in 2021 the method can be still characterized as academic one because neither commercial nor open source software has been released yet. The interest of the method has not stopped as well, but its development is rather slow. In the period of 2007–2012 several hundred scientific papers on the subject have been published. About 300 can be found in the Scopus database, 100 in SpringerLink and 200 in ScienceDirect. The majority of published works concerns engineering applications of DVM or improving its accuracy. The problem of accelerating calculations appears extremely rarely and relates to modification of DVM algorithms rather than parallelization of calculations. And so, for example, Ricciardi, Wolf & Bimbato, 2017 studied the combination of exponential and power series expansions implemented using a divide and conquer strategy to accelerate the calculation while two years earlier he proposed fast multipole method algorithm to accelerate the expensive interactions of the discrete vortices (Ricciardi et al., 2015). The results of analysis on possibility of using fast matrix multiplication methods for the approximation of the velocity field when solving the system of differential equations describing the vorticity transport in an ideal incompressible fluid in Lagrangian coordinates can be found at Aparinov & Setukha (2009). Whereas Dynnikova (2009) explored the construction of a hierarchical structure of regions (tree) in order to accelerate the calculations. A different approach represents Huang, Su & Chen (2009), who introduced a concept of residual circulation in that sense that only a partial circulation of the vortex sheet is diffused into the flow field. The cited examples show that accelerating calculations with hardware methods has not been of interest to the researchers. Only Kuzima, Marchevsky & Moreva (2015) studied the speed-up in DVM calculations on multicore (using MPI and OpenMP) and graphic workstation (CUDA). She reported acceleration in calculations up to 40 times. On the other hand the interest in the classic n-body problem itself has not stopped. Despite the fact that today it is a well-recognized problem novel simulations are being preformed (e.g. Groen, Zwart, Ishiyama & Makino, 2011) and new software is being developed (e.g. Incardona, Leo, Zaluzhny, Ramaswamy & Sbalzarini, 2019).

Taking into account the above information any practical study on usage of computer clusters in the DVM should be in the field of interest of so called theoretical engineers. It happens very often that small research groups (at universities or in start-ups) ask themselves if it is worth to invest their time in building a computer cluster and creating parallel solvers in order to speed up calculations. The aim of this paper is to facilitate the answer to such questions in the case of Discrete Vortex Method. In this respect, the presented results remain still valid.

2. THE COMPUTER EXPERIMENT

The experiment was carried out in 2007 in a computer laboratory at Lublin University of Technology. The laboratory was equipped with 12 single-processor PCs connected with Fast Ethernet network. All the computers had the *AMD Atlon XP 1600+* 1.6GHz processor and 256MB RAM. The cluster was a symmetric one and was build according to Soan 2005. The *Ubuntu Linux 6.10* was used as an operating system and the *Mpich 2.0* as a communication layer. As a part of the experiment, three original DVM solvers were developed (see Suplement): *vorsym_s*, *vorsym_q* and *vorsym-p*. The program *vorsym_s* (vortex simulator slow) is a single-process and single-threaded program which implements the PP algorithm. The *vorsym_q* (quick) is also a single-process and single-threaded program but it implements the P^3M algorithm. Whereas the *vorsym_p* (parallel) solver is a multi-process (but still single-threaded) solver implementing the P^3M algorithm. The last program was run on the computer cluster using 4 or 9 nodes of it. (The number of nodes has been added in round brackets.) From the engineering point of view, the most important thing was to compare the execution time of calculations between *vorsym_q* along with *vorsym_p(4)* and *vorsym_p(9)*.

Tab. 1. The set of data used to perform the simulations

No.	File	Vortexes	Size		No.	File	Vortexes	Size
1	9.vrt	9	14 MiB		18	30k.vrt	29 929	45 GiB
2	25.vrt	25	38 MiB		19	40k.vrt	40 000	60 GiB
3	36.vrt	36	55 MiB		20	50k.vrt	49 729	74 GiB
4	49.vrt	49	75 MiB		21	60k.vrt	59 536	89 GiB
5	81.vrt	81	124 MiB		22	70k.vrt	69 696	104 GiB
6	100.vrt	100	153 MiB		23	80k.vrt	79 945	119 GiB
7	200.vrt	196	299 MiB		24	90k.vrt	90 000	134 GiB
8	300.vrt	289	441 MiB		25	100k.vrt	100 489	150 GiB
9	400.vrt	400	610 MiB		26	200k.vrt	200 704	299 GiB
10	500.vrt	484	739 MiB		27	300k.vrt	299 209	446 GiB
11	600.vrt	576	879 MiB		28	400k.vrt	399 424	595 GiB
12	700.vrt	676	1032 MiB		29	500k.vrt	499 849	745 GiB
13	800.vrt	784	1196 MiB		30	600k.vrt	600 625	895 GiB
14	900.vrt	900	1373 MiB		31	700k.vrt	700 569	1044 GiB
15	1k.vrt	1024	1563 MiB		32	800k.vrt	801 025	1194 GiB
16	10k.vrt	10 000	15 GiB		33	900k.vrt	900 601	1345 GiB
17	20k.vrt	19 881	30 GiB		34	1M.vrt	1 000 000	1.5 TiB

For the experiment 34 numerical samples were generated. They were files defining the initial conditions of the n-body DVM tasks. The samples differed in the number of vortex particles (Tab. 1). In each case the size of the computational space (domain) was of the same size of 100×100. Initially the vortexes were randomly and evenly distributed in the domain. The random Marsagil generator was used. The vortexes had also random strengths from –1.0 to 1.0. All the initial velocities were zero. All simulations were carried out with a constant step time equals to 0.01. The number of vortices in the domain was fixed. Vortices that crossed the domain boundary making their moves were returned to the domain from the opposite edge in such way that they continued their movement on the opposite side. The column *Size* gives the sizes of the output files for a 100 000 step simulation for each case.

3. RESULTS

The main aim of the simulations was to test the efficiency of the developed computational system considered as the computer cluster and dedicated solver *vorsym_p*. The simulations were carried out for all prepared files (Tab. 1). Depending on the size of a task a single simulation took form 3 to 10 000 steps due to time constraints. In order to normalize the obtained results, the average execution time of a single step was calculated. Results has been presented in a table (Tab. 2) and in a diagram (Fig. 3). The specimen numbers from Tab. 1. agrees with numbers from Tab. 2. Simulations for the specimens number form 22 to 34 were not performed with *vorsym_s* due to too long calculating times. Additionally, for the for *vorsym_q* and *vorsym_p* simulators number of computing subdomains were given. The subdomains were formed by dividing the square main domain to, also square, areas. The sides of the main domain were divided as follow: 2×2, 3×3, 4×4, …, 19×19, which resulted in 4, 9, 16, …, 361 subdomains respectively. They determined the calculating mesh of the P^3M algorithm. For the parallel *vorsym_p* solver firstly the main domain was divided into subdomains of distinct processes (4 or 9) then each process created its own subdomains according to the previously described rule.

A typical engineering problem solved using DVM requires at least tens of thousands discrete vortexes and performing about 100 000 simulation steps. Approximate times of completing such tasks were estimated on the basis of results from Tab. 2 and presented in Tab. 3. The results were valid in 2007 (for hardware reasons) but still clearly show significant reduction of computational time when a computer cluster is used for DVM. Whereas relative speed-ups of calculations presented on a diagram in the Fig. 4 has not outdated at all. It was noticed that 4 node cluster speeded up simulation 8 times and 9 node cluster – 22 times! The results are dubious but absolutely correct. So why did cooperation of n nodes caused acceleration greater than n times? The answer is the large amount of data generated at each calculating step and written into files. In the case of n nodes there were n different files on different hard dives instead of one big file on a single drive, which shortened the execution time of each step. Each node wrote only its own data. It was the amount of output data along with the specificity of n–body problem what determined the time of simulations. It is also the reason why obtained results does not correspond with those found in Kuzmina et al. (2015) where for small number of calculating cores linear acceleration was observed. Simply, the simulation time did not include data recording to files at each step of the simulation. This can not be avoided in engineering practice, which makes low-budget cluster very effective while deployed in DVM calculations. Such observation is very important to an engineer who has a task to shorten the time of DVM simulations as much as possible.

Tab. 2. Averaged time of performing a single simulation step for each file

No.	vorsym_s	vorsym_q		vorsym_p (4)		vorsym_p (9)	
	T [s]	N_{sub}	T [s]	N_{sub}	T [s]	N_{sub}	T [s]
1	0.00015	1	0.00029	1	0.00660	1	0.04860
2	0.00042	4	0.00072	1	0.00660	1	0.08000
3	0.00066	4	0.00091	1	0.00650	1	0.05000
4	0.00108	4	0.00111	4	0.00800	1	0.04900
5	0.00190	9	0.00146	4	0.00804	1	0.04800
6	0.00276	9	0.00158	4	0.00800	1	0.04600
7	0.01060	9	0.00280	4	0.01005	4	0.06670
8	0.02260	9	0.00490	4	0.01001	4	0.08230
9	0.04340	16	0.00803	9	0.01030	4	0.10076
10	0.06400	16	0.01100	9	0.01502	4	0.08330
11	0.09100	16	0.01403	9	0.01510	4	0.06000
12	0.14500	16	0.01900	9	0.01511	4	0.10040
13	0.16700	16	0.02201	9	0.01512	9	0.10108
14	0.21800	25	0.02700	9	0.01540	9	0.10204
15	0.30082	25	0.03012	9	0.01750	9	0.10160
16	27.1	64	0.9	36	0.3	25	0.1
17	106.3	100	2.7	49	0.8	36	0.5
18	241.7	121	5.1	64	1.0	36	0.7
19	432.0	144	7.8	64	1.3	49	1.0
20	685.0	144	10.7	81	1.7	49	1.3
21	952.0	169	14.3	81	2.0	64	1.3
22	-	169	18.1	100	2.7	64	1.7
23	-	196	21.7	100	3.0	64	2.0
24	-	225	27.0	100	3.5	64	2.3
25	-	225	30.7	121	4.3	81	2.7
26	-	324	89	169	12	100	5
27	-	400	159	196	23	121	9
28	-	441	245	225	32	144	13
29	-	484	341	256	44	169	17
30	-	529	452	289	57	196	22
31	-	576	547	289	73	196	27
32	-	625	697	324	87	225	33
33	-	625	829	324	104	225	38
34	-	279	994	361	122	225	44

Tab. 3. Estimated time of completing a 100 000 step DVM simulation

Solver	Number of vortexes in the simulation		
	20 000	**50 000**	**100 000**
vorsym_s	4 months	2 years	9 years
vorsym_q	3 days	12 days	36 days
vorsym_p(4)	22 hours	2 days	5 days
vorsym_p(9)	14 hours	1½ day	3 days

Obtained results may seem outdated nowadays due to the development in computer hardware since 2007. It is evident that in 2021 the simulations, if recreated would be completed in much shorter times even using the same computer code. It would be simply achieved by using faster CPUs and hard disc drives. That said, it is also evident that the new hardware nowadays could be used for bigger problems. In other words the discussion today would consider bigger task. Since the nature of the n-body problem has not changed it is still the writing of the output to hard discs which delays calculations significantly. What could improve the performance in this area is using multicore CPUs, which let delegate the writing tasks to a separate threads. It should be undoubtedly the first idea to be explored. Another way to overcome the problem of time-consuming n-body simulations could be using the GPUs technology that is much more affordable now, though this question is beyond the scope of this paper.

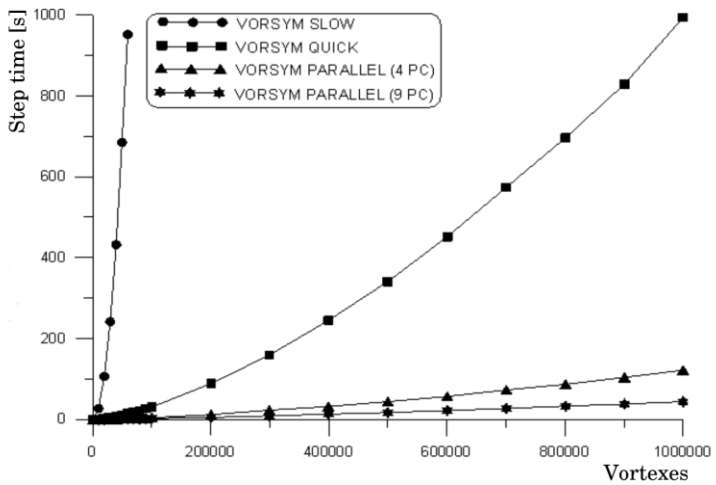

Fig. 3. Averaged time of performing a calculation step against the size of a task for different calculation methods

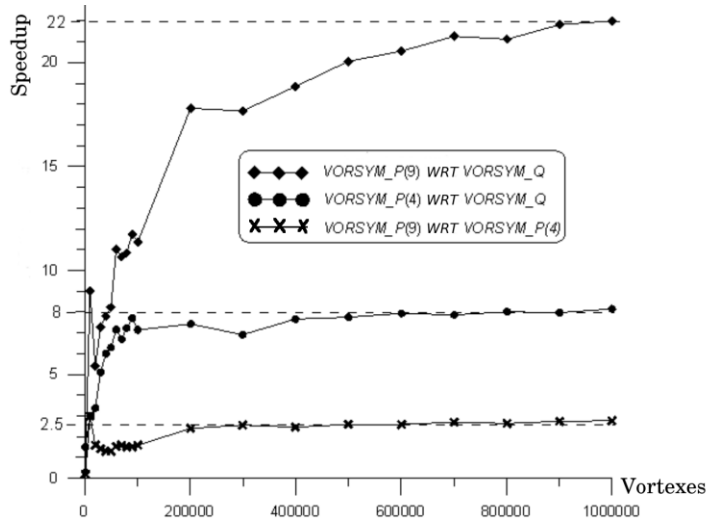

Fig. 4. Relative speedup of simulation (WRT abbr "with relation to")

4. CONCLUSIONS

In this paper results on the possibility of accelerating the Discrete Vortex Method computer simulations were presented. A low-budget computer cluster was build and a parallel solver was developed. Obtained acceleration of calculations exceeded the number of the cluster nodes due to division of the computing domain and separation of the output files. The paper deals with issues rarely described in the literature on the discrete vortex method.

Supplement

https://github.com/TomekNowicki/vorsym

REFERENCES

Aparinov, A. A., & Setukha, A. V. (2009). On the application of mosaic-skeleton approximations of matrices for the acceleration of computations in the vortex method for the three-dimensional Euler equations. *Differential Equations*, *45*, 1358. http://doi.org/10.1134/S0012266109090110

Cottet, G. H., & Koumoutsakos, P. D. (2000). *Vortex Methods Theory and Practice.* Cambridge University Press.

Dynnikova, G. Ya. (2009). Fast technique for solving the *N*-body problem in flow simulation by vortex methods. *Computational Mathematics and Mathematical Physics*, *49*, 1389–1396. http://doi.org/10.1134/S0965542509080090

Groen, D., Zwart, S. P., Ishiyama, T., & Makino, J. (2011). High Performance Gravitational N-body Simulations on a Planet-wide Distributed Supercomputer. *Computational Science & Discovery*, *4*(1), 015001. http://doi.org/10.1088/1749-4699/4/1/015001

Hockney, R. W., & Eastwood, J. W. (1988). *Computer Simulation Using Particles*. Taylor & Francis Group.

Huang, M. J., Su, H. X., & Chen, L. Ch. (2009). A fast resurrected core-spreading vortex method with no-slip boundary conditions. *Journal of Computational Physics*, *228*(6), 1916–1931. https://doi.org/10.1016/j.jcp.2008.11.026

Incardona, P., Leo, A., Zaluzhny, Y., Ramaswamy, R., & Sbalzarini, I. F. (2019). OpenFPM: A scalable open framework for particle and particle-mesh codes on parallel computers. *Computer Physics Communications*, *241*, 155–177. https://doi.org/10.1016/j.cpc.2019.03.007

Kuzmina, K., Marchevsky, I., & Moreva, V. (2015). Parallel Implementation of Vortex Element Method on CPUs and GPUs. *Procedia Computer Science*, *66*, 73–82. https://doi.org/10.1016/j.procs.2015.11.010

Lewis, R. I. (1991). *Vortex Element Methods for Fluid Dynamics of Engineering Systems.* Cambridge University Press.

Nowicki, T. (2007). *Algorytm równoległy dla problemu n-ciał* (Unpublished master thesis). Lublin University of Technology, Lublin. https://github.com/TomekNowicki/vorsym/blob/main/nowicki_n-body.pdf

Nowicki, T. (2012). *Wpływ sposobu realizacji warunków brzegowych w metodzie wirów dyskretnych na odpowiedź aeroelastyczną pomostów*. Politechnika Lubelska.

Nowicki, T. (2015). The Discrete Vortex Method for estimating how surface roughness affects aerodynamic drag acting on a long cylinder exposed to wind. *Technical Transactions, Civil Engineering*, *2-B*(12), 127–144. https://doi.org/10.4467/2353737XCT.15.129.4166

Ricciardi, T. R., Wolf, W. R., & Bimbato, A. M. (2017). A fast algorithm for simulation of periodic flows using discrete vortex particles. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *39*, 4555–4570. http://doi.org/10.1007/s40430-017-0902-x

Ricciardi, T., R., Bimbato, A. M., Wolf, W., R., Idelsohn, S. R., Sonzogni, V., Coutinho, A., Cruchaga, M., Lew, A., & Cerrolaza, M. (2015). Numerical simulation of vortex interactions using a fast multipole discrete particle method. *Proceedings Of The 1st Pan-american Congress On Computational Mechanics And Xi Argentine Congress On Computational Mechanics* (pp. 1065–1076). Barcelona: Int Center Numerical Methods Engineering.