ACCELERATION OF BEM WITH THE CROSS APPROXIMATION FOR DETERMINATION OF BOUNDARY VORTICITY

JAN TIBAUT[1], LEOPOLD ŠKERGET[2] & JURE RAVNIK[1]
[1]Faculty of Mechanical engineering, University of Maribor, Slovenia.
[2]Wessex Institute, Ashurst Lodge, Southampton.

ABSTRACT

In this paper, we present a fast boundary element method (BEM) algorithm for the solution of the velocity-vorticity formulation of the Navier-Stokes equations. The Navier-Stokes equations govern incompressible fluid flow, which is inherently nonlinear and when discretizised by BEM requires the discretization of the domain and calculation of domain integrals. The computational demands of such method scale with $O(N^2)$, where N is the number of boundary nodes. To accelerate the solution process and reduce the computational demand, we present two different approaches, the subdomain method and an approximation procedure with hierarchical structure. Several approximation techniques exist, such as multipole approximation methods FMM (fast multiple method), SVD (singular value decomposition method), wavelet transform method and a cross approximation method. In this paper, we present the cross approximation method in combination with the hierarchical H-structure. The cross approximation method can reduce the computational demands from $O(N^2)$ to O(N log N). There are many forms of the cross approximation, like the algebraic cross approximation and the hybrid cross approximation. Here, we applied the algebraic cross approximation form. The main advantage is that we did not need to evaluate the integral and then to change it with a degenerate kernel function. The cross approximation algorithm was used to solve the kinematics equation for unknown boundary vorticity values. Results show that an increasing of the compression rate has a negative influence on the solution accuracy. On the other hand, the solution accuracy increases with computational grid density. Tests were performed using the 3D lid-driven cavity test case with Reynolds numbers up to 1000. Solution accuracy was similar for all Reynolds numbers considered. In conclusion, the tests showed that our implementation of the algebraic cross approximation for the acceleration of the solution of the kinematics equation can be applied to decrease the computational demands and to accelerate the BEM.

Keywords: adaptive cross approximation, boundary element method, boundary-domain integral method, fast algorithms, hierarchical structure, H-matrix, kinematics equation, lid-driven cavity, velocity-vorticity formulation

1 INTRODUCTION

The boundary element method (BEM) is a numerical method for solving partial differential equations (PDE). The numerical method is based on the Greens second theorem. Thus, only the discretization of the boundary is necessary [1]. However, due to the non-local nature of the fundamental solution, the resulting system of linear equations is fully populated. Due to this, the computational demands of the method scale with the square of the number of boundary unknowns $O(N^2)$.

To reduce computational demand, a number of accelerating procedures were proposed by diverse authors, which aim to reduce the cost from $O(N^2)$ to O(N logN) or even to O(N). We present the cross approximation algorithm in combination with a hierarchical structure for the acceleration of the solution process. The hierarchical structure or H-structure is a procedure, which divides the full matrix formulation into smaller matrices, based on the shape of the numerical domain. A geometrical condition filters the matrix parts on which an approximation algorithm is applied and the matrix parts which are not admissible. Börm *et al*. [2] presented the application of hierarchical matrices and the bounding box and bisection method

ISSN: 2046-0546 (paper format), ISSN: 2046-0554 (online), http://www.witpress.com/journals
DOI: 10.2495/CMEM-V6-N6-1097-1107

for building cluster trees. Hackbusch [3] wrote the arithmetic based on H-matrices. A different hierarchical structure is an H^2-matrix formulation. Börm [4] used the H^2-matrix formulation to construct efficient approximations of discretized integral formulation.

There are different authors who present different forms and variants of the cross approximation method. Bebendorf [5] introduced the fully pivoting adaptive cross approximation algorithm (ACA). Rjasanow and Steibach [6] employed the partly pivoting adaptive cross approximation algorithm (ACA+). Tamayo [7] presented a special algebraic form named the multilevel adaptive cross approximation (MLACA). The MLACA is a procedure that divides the matrix into a number of levels and applies the ACA algorithm on each level. Bebendorf [8] used the cross approximation algorithm on a collocation BEM and Rjasanov [9] applied it for Galerkin BEM. Börm and Grasedyck [10] introduced the hybrid cross approximation of integral operators, where they used the H-matrix representation of finite element stiffens matrix.

Many authors used the cross approximation method in different applications. In all publications, the approximation procedure was introduced to accelerate the BEM. The most often used form of the cross approximation is the adaptive cross approximation. Schröder *et al*. [11] used the adaptive cross approximation algorithm to solve the electromagnetic field for a known current distribution. Kurz and Rjasanow [12] used the ACA algorithm to accelerate a coupled BEM and finite element numerical method to solve the distribution of a symmetric electromagnetic field. Grytsenko and Galybin [13] used the ACA to solve a large number of cracks on a plate with the, singular integral method. Maerten [14] used the adaptive cross approximation to solve a 3D elasticity problem. Wei *et al*. [15] solved the Laplace equation to predict the temperature distribution in a 2D domain by combining ACA with the singular boundary method.

The purpose of this work is to implement the cross approximation algorithm with the H-structure to speed up a BEM-based fluid flow solver. The fluid flow solver [16, 17] has been developed for the velocity-vorticity formulation of the Navier-Stokes equations. Our main motivation was to decrease the memory storage demands. We tested the application and assessed its usefulness using the lid-driven cavity benchmark test case [18].

2 GOVERNING EQUATIONS

The fluid flow is defined by two equations, the kinematic and the transport vorticity equation. The two equations are formulated in the velocity-vorticity formulation of the Navier-Stokes equations. Let $\vec{v}$ denote the velocity field and $\vec{\omega} = \vec{\nabla} \times \vec{v}$ the vorticity field. With this consideration the kinematics equation may be specified from the continuity equation as [16]:

$$\nabla^2 \vec{v} + \vec{\nabla} \times \vec{\omega} = 0. \tag{1}$$

For a non-compressible fluid, the velocity field and the vorticity field are divergence free. Eqn (1) represents a connection between the velocity and vorticity vector field. The fluid movement governs the vorticity transport equation, written in a non-dimensional form, from the momentum equation as:

$$\frac{\partial \vec{\omega}}{\partial t} + \left(\vec{v} \cdot \vec{\nabla}\right)\vec{\omega} = \left(\vec{\omega} \cdot \vec{\nabla}\right)\vec{v} + \frac{1}{Re}\nabla^2 \vec{\omega}, \tag{2}$$

where $Re = \dfrac{vl}{\nu}$ is the Reynolds number and l the characteristic length scale. The fluid viscosity ν and fluid density ρ are considered constant.

The system of equations that is formed with the discretization process is solved with an iterative algorithm proposed by Ravnik *et al*. [17]. In this paper, we present an accelerated version of the estimation of boundary vorticity. In the second step, the vorticity momentum equation is solved with the sub-domain BEM to solve the domain velocity. Domain decomposition cannot be used for the solution of the kinematics equation due to the Biot-Savart law. Because of this, we used the cross approximation technique in combination with the H-structure to solve the boundary vorticity in eqn (1).

2.1 The kinematics equation for the boundary vorticity

The kinematic equation solves the vorticity field of the fluid flow, which is on the edge of the domain. Let us consider a domain Ω with a position vector, $\vec{r} \in \mathbb{R}^3$ and with a boundary $\Gamma = \delta\Omega$. The integral form of the kinematics eqn (1) without derivatives of the velocity and vorticity fields takes the following form [17]:

$$c\left(\vec{\xi}\right)\vec{v}\left(\vec{\xi}\right) + \int \vec{v}\vec{\nabla}u^* d = \int \vec{v} \times \left(\vec{n} \times \vec{\nabla}\right) u^* d + \int_{'} \left(\vec{\omega} \times \vec{\nabla}u^*\right) d', \tag{3}$$

where $u^*\left(\vec{r},\vec{\xi}\right) = \dfrac{1}{4\pi\left|\vec{r}-\vec{\xi}\right|}$ is the fundamental solution of the Laplace equation and $\vec{\xi}$ is the source point. In order to use the kinematics equation to solve the boundary vorticity values, we must rewrite eqn (3) in a tangential form by multiplying the system with a normal in the source point $\vec{n}\left(\vec{\xi}\right)$. This yields the following integral equation,

$$\begin{aligned} c\left(\vec{\xi}\right)\vec{n}\left(\vec{\xi}\right)\times\vec{v}\left(\vec{\xi}\right) + \vec{n}\left(\vec{\xi}\right)\times\int_{\Gamma}\vec{v}\vec{\nabla}u^* \cdot \vec{n} d\Gamma \\ = \vec{n}\left(\vec{\xi}\right)\times\int_{\Gamma}\vec{v}\times\left(\vec{n}\times\vec{\nabla}\right)u^* d\Gamma + \vec{n}\left(\xi\right)\times\int_{\Omega}\left(\vec{\omega}\times\vec{\nabla}u^*\right) d\Omega. \end{aligned} \tag{4}$$

The discretization process, divides the domain into domain elements $\Omega = \sum_{i=1}^{d}\Omega_i$ and the boundary into boundary elements $\Gamma = \sum_{i=1}^{b}\Gamma_i$, where d is the number of domain elements and b the number of boundary elements. With this, we can write the discrete form of the kinematic equation. Each domain part Ω_i is defined with 27 nodes ϕ_i and each boundary part Γ_i is defined with 9 nodes Φ_i. A quadratic interpolation function was used for the boundary and domain nodes.

For each collocation node at the boundary $\vec{\xi}_f$, we calculate the following integrals over boundary element c:

$$h_{fcl} = \delta_{fcl} c\left(\xi_f\right) + \sum_{l=1}^{9}\int_{c} \phi_l n_i \frac{\partial u^*}{\partial x_i} d_c, \tag{5}$$

$$h_{fcl}^{ij} = \sum_{l=1}^{9}\int_{c} \phi_l \left[n_i \frac{\partial u^*}{\partial x_j} - n_j \frac{\partial u^*}{\partial x_i} \right] d_c. \tag{6}$$

and the following integrals over domain elements e

$$d_{fel}^{i} = \sum_{l=1}^{27} \int_{\Omega_e} \phi_l \frac{\partial u^*}{\partial x_i} d\Omega_e . \tag{7}$$

The indexes i, j in eqns (5)–(7) stand for the coordinates x, y, z. For the integral in eqn (6), $i \neq j$. The character δ_{fcl} is the Kronecker delta, which is equal to 1 when source point and the grid point are equal.

In order to separate the unknown boundary vorticity values, the vorticity vector can be written as a sum of boundary and domain vorticity as $\{\omega_i\} = \{\omega_i\}_\Gamma + \{\omega_i\}_{\Omega\setminus\Gamma}$. Vector $\{\omega_i\}_\Gamma$ includes the boundary vorticity and vector $\{\omega_i\}_{\Omega\setminus\Gamma}$ includes the vorticity at nodes in the domain. Hence, we can write the system of equations like this:

$$\begin{aligned}
&\left([n_i][D_i]+[n_j][D_j]+[n_k][D_k]\right)\{\omega_i\}_\Gamma \\
&\quad = \left([n_j][H_{ki}]-[n_k][H_{ij}]\right)\{v_i\}+\left([n_j][H]-[n_k][H_{jk}]\right)\{v_k\} \\
&\quad -\left([n_i][H]+[n_j][H_{jk}]\right)\{v_j\}+[n_k][D_x]\{\omega_k\}_\Gamma+[n_j][D_i]\{\omega_j\}_\Gamma \\
&\quad +[n_i][D_i]\{\omega_i\}_\Gamma-\left([n_j][D_j]_{\Omega\setminus\Gamma}+[n_k][D_k]_{\Omega\setminus\Gamma}\right)\{\omega_i\}_{\Omega\setminus\Gamma} \\
&\quad +[n_j][D_i]_{\Omega\setminus\Gamma}\{\omega_j\}_{\Omega\setminus\Gamma}+[n_k][D_i]_{\Omega\setminus\Gamma}\{\omega_k\}_{\Omega\setminus\Gamma}
\end{aligned} \tag{8}$$

Indexes i, j, k in eqn (8) are the coordinates i=x, j=y, k=z. From eqn (8), we write a system of three linear equations. In the first equation, the combination of indexes is i, j, k in the second the combination indexes is j, k, i and in the last one the combination is k, i, j.

Matrices [n_x], [n_y], [n_z] are diagonal matrices that contain the components of the normal vector $\vec{n} = \{n_x, n_y, n_z\}$. The size of matrices $[H]$ and $[H_{ij}]$ is $n \times n$, where n is the number grid nodes on the boundary. The first domain matrices $[D_i]$ have the size $n \times n$, while the second $[D_i]_{\Omega\setminus\Gamma}$ are of the size $n \times m$, where i is the coordinate *x, y, z* and m is the number of domain nodes.

As the size of $[D_i]_{\Omega/\Gamma}$ domain matrices is one order of magnitude greater than all other matrices, we implemented the cross approximation algorithm for these matrices.

3 APPROXIMATION OF THE DOMAIN MATRIX

Let us consider one of the domain integral matrices $[D_i]_{\Omega\setminus\Gamma}$ and denote it as D. Its size is $n \times m$ and its entries are real numbers $D \in \mathbb{R}^{n\times m}$. To approximate the domain matrix D first, an H-matrix structure has to be built. The H-matrix structure is built from the block cluster tree. The block cluster tree is a combination of two cluster trees. The first cluster tree is built from the domain elements and the second one is built from the boundary elements. From the two cluster trees, a block cluster tree is formed. Each cluster of the block cluster tree is tested on a geometrical condition, which is based on the shape of the domain. Considering the geometrical condition, we can split the matrix $D_{n\times m}$ into smaller matrices $\hat{D}|_{\hat{n}\times\hat{m}}$. Matrix parts that fulfil the geometrical condition can be approximated.

3.1 Construction of the block cluster tree

A block cluster tree is a combination of the domain and boundary clusters. The computational domain is split into smaller boxes, which we will denote as clusters. In each level of the

domain cluster tree, the number of clusters increases and the size of the boxes decreases. Clusters that are at the end of the cluster tree are called leaves. The cluster tree that is built from the computational domain is denoted as T_I. The total number of levels p defines the depth of a cluster tree. The following algorithm below was used to build the domain cluster tree T_I:

- for all levels i=1 to p
- for all clusters $I_k^{(i)}$ at the level, i
- find $I_k^{(i)}$'s eight neighbouring clusters and join them into a new cluster at level $i+1$

For each domain cluster in T_I cluster tree, we define a bounding box $Q_{dm} \subseteq \mathrm{IR}^3$, which includes all domain nodes in the cluster as proposed by Börm [2]. Similarly, for each boundary cluster in the T_J cluster tree, we define a bounding box $Q_{bn} \subseteq \mathbb{R}^3$ from the boundary nodes in the cluster. In Fig. 1, we present a block cluster of the block cluster tree.

3.2 H structure formulation

From the block cluster tree, we build an H-matrix, by applying a geometrical condition. The block cluster tree includes several parts. We decide which cluster is admissible by defining a geometrical condition. The geometrical condition is:

$$min\left\{dim\left(Q_{dm}\right), dim\left(Q_{bn}\right)\right\} \leq \eta\, dist\left(Q_{dm}, Q_{bn}\right). \tag{9}$$

Expressions $dim\left(Q_{dm}\right) = \left|\vec{r}_A - \vec{r}_B\right|$ and $dim\left(Q_{bn}\right) = \left|\vec{r}_C - \vec{r}_D\right|$ are the diameters of boundary and domain bounding boxes. The distance $dist\left(Q_{dm}, Q_{bn}\right)$ is the minimal distance between nodes in the boundary and domain clusters. All expressions are illustrated in Fig. 1. Parameter η is the admissibility parameter, which is defined by the user. This parameter enables the user to vary the geometrical condition and to control the number of admissible block clusters.

Block clusters which fulfil the geometrical condition, are admissible. Thus, an approximation procedure can be applied. However, block clusters, which do not meet the geometrical condition, are split into smaller blocks.

Leaves, which meet the geometrical condition in eqn (10), are admissible leaves. Leaves that do not fulfil the geometrical condition are not admissible and they have to be maintained

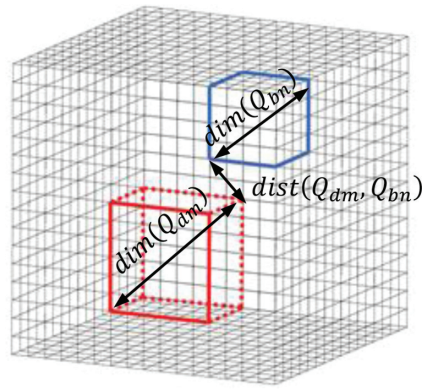

Figure 1: An illustration of a block cluster, which is tested for admissibility.

in their original form. In our simulations, we used $\eta = 1$ based on the recommendation given by Wei *et al*. [15].

3.3 Cross approximation

Cross approximation is a method that approximates matrices $\hat{D}_{\hat{n}\times\hat{m}}$, from the full matrix formulation to an RK-form. Every matrix that has a rank k>1, can be written into the RK-matrix formulation using the approximation rank r in the following way [2]:

$$\tilde{D}_{r(\hat{n}+\hat{m})} \approx \hat{A}\hat{B}^T, \hat{A} \in \mathbb{R}^{\hat{n}\times r}, \hat{B} \in \mathbb{R}^{r\times\hat{m}}. \tag{10}$$

To store the *RK*-matrix into memory and perform matrix-vector multiplications, we need $O(\hat{n}\hat{m})$ computational resources [10]. The approximation rank r is the number of rows in matrix A and the number of columns in matrix B^T. In this paper, we present the full pivoting cross approximation method. Full pivoting means that the algorithm evaluates each element in the matrix and picks the highest valued element, to determine a cross, which is the building block of the RK-matrix formulation. The cross approximation algorithm with full pivoting is written bellow:

1. $\hat{R}_0 = \hat{D}_{\hat{n}\times\hat{m}}$
2. *The loop* $u = 0,1,2,3,\ldots,r$

$$(2.1)\left(i^*, j^*\right)^u = Argmax\left|\hat{R}^u\right|$$

$$(2.2)\gamma^{u+1} = \left(\hat{R}^u_{i^*,j^*}\right)^{-1}$$

$$(2.3)a^{u+1} = \gamma^{u+1}\hat{R}^u_{i,j^*}, b^{u+1} = \left(\hat{R}^u_{i^*,j}\right)^T$$

$$(2.4)\hat{R}^{u+1} = \hat{R}^u - a^{u+1}b^{u+1}$$

In point (1) of the algorithm, we define the residual matrix. The second point (2) is a start of a loop that performs the approximation. In eqn (2.1), the largest element in the matrix $\hat{R}^u$ is found, with the full pivoting procedure. The full pivoting algorithm needs to perform $O(\hat{n}\hat{m})$ pivoting steps. To reduce the number of pivoting steps Rjasanow and Steibach [6] proposed a partly pivoting cross approximation method (ACA+). However, the partly pivoting algorithm demands more computational resources. Because of this, the algorithm is less desirable. Next in eqn (2.2), the reverse of the largest element $\hat{R}^u_{i^*,j^*}$ is calculated. In eqn (2.3) vectors a^{u+1} and b^{u+1} are defined. These two vectors build a cross around the element $\hat{R}^u_{i^*,j^*}$. Eqn (2.4) calculates a new residual matrix $\hat{R}^{u+1}$.

We propose to use a user defined compression factor a, which defines the approximation rank r in the following way:

$$r = \alpha \cdot k\ , \tag{11}$$

where k is the rank of the matrix $\hat{D}_{r(\hat{n}+\hat{m})}$. Based on r, we can calculate the compression ratio φ

$$\varphi = \frac{\sum_i \hat{n}_i \cdot \hat{m}_i + \sum_j r_j\left(\hat{n}_j + \hat{m}_j\right)}{n \cdot m} \tag{12}$$

which, is the memory ratio of the approximated matrix versus the original matrix. The counter is the sum of the matrix elements in admissible block clusters and leaves $\tilde{D}_{r(\hat{n}+\hat{m})}$ plus the elements in inadmissible matrix parts $\hat{D}_{\hat{n}\times\hat{m}}$. And the denominator is the number of elements in the original matrix $D_{n\times m}$.

4 NUMERICAL MODEL AND EXPERIMENTS

Flow in a 3D lid-driven cavity is one of the standard test cases used in the development of flow solvers. The domain as well as the boundary conditions are unambiguously defined and do not change with the Reynolds number.

We tested the proposed algorithm using the lid-driven cavity problem [18]. The domain was a unit cube. We sought steady state flow solutions using three different grid densities, having 25^3, 41^3 and 49^3 nodes. The grid was structured and non-uniform. The size of hexahedral elements changed from the boundary to the middle elements by a length ratio 1:6.5.

The side and bottom walls of the cube are no-slip walls. The top wall is moving with a velocity, which is determined from the Reynolds number. The boundary vorticity is obtained by the solution of the accelerated kinematics equation, except for the $\omega_x = 0$ on the left and right wall, $\omega_y = 0$ on the front and back wall and $\omega_z = 0$ on the top and bottom walls. The flow was solved for three Reynolds numbers Re=100, 400 and 1000. Boundary conditions are shown in Fig. 2. The linear system of equations was solved with the LU-decomposition method. Iterations were stopped when the difference between subsequent iteration of all field functions dropped below 10^{-6}. The maximum number of iterations for each run was 8000.

The cross approximation was used on the matrices $[D_x]_{\Omega/\Gamma}$, $[D_y]_{\Omega/\Gamma}$, $[D_z]_{\Omega/\Gamma}$ in eqn (8). In order to measure the influence of the approximation on the result, find its dependence on the flow structure, and grid density, we used two norms. We measured the difference between boundary vorticity values as:

$$RMS_{\omega} = \left(\sum_{j=x,y,z} \frac{\sum_i \left(\omega_{ji} - \omega_{Aji}\right)^2}{\sum_i \left(\omega_{Aji}\right)^2} \right)^{\frac{1}{2}} \tag{13}$$

where ω_{ji} is the j^{th} component of vorticity in i^{th} boundary node calculated without an approximation and the ω_{Aji} its approximated counterpart.

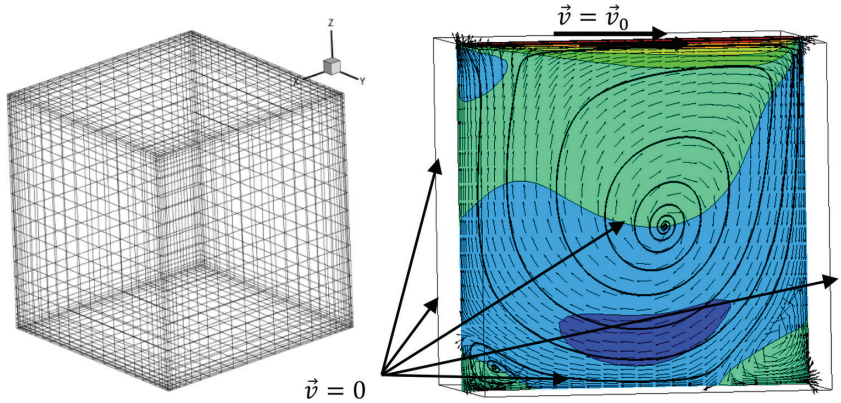

Figure 2: Flow structure and boundary conditions for the lid driven cavity at Re = 400 on the right and a 25^3 grid on the left.

5 RESULTS AND DISCUSSION

In this section, we illustrate the impact of the cross approximation method on the solution of the boundary vorticity. The Reynolds number was changed in an interval from 100 to 1000 to see how a different compression rate α effect the norm RMS_{ω}. All simulations were performed on an Intel Xenon 64-bit processor with 64 GB of memory space. Table 1 is showing the amount of memory that is needed to store one of the matrices $[D_i]_{\Omega/\Gamma}$. We observed that the space to store the original matrix formulation increases quadratically with the number of boundary unknowns, while the approximated formulation exhibits a quasilinear dependence.

In Table 2, we observed the number of iterations ($Niter_{\alpha}$) that was necessary to obtain a solution of the kinematics equation. We noticed that with an increasing grid density and Reynolds number the number of iterations has increased. This is because the non-linearity of the problem increases with a higher Reynolds number and a better-resolved grid. However, the compression rate did not have an impact on the number of iterations, if α was between 1.0 and 0.5. When the compression rate was lowered below 0.5, the number of iterations started to increase. Thus, the compression rate has an influence on the number of iterations to solve the boundary vorticity.

In Fig. 3, we illustrate how the norm RMS_{ω} is dependent of the compression ratio φ and different Reynolds numbers. The RMS increases with a higher compression ratio. At a given Reynolds number, we observe that the norms are lower for a fine grid compared to course grid. This means that the simulation with a fine grid is able to use less memory storage to reach the same accuracy of results compared to a simulation on a coarse grid. Thus, the difference in the boundary vorticity is dependent of the grid density.

Using the norm, we focused on the impact of the cross approximation algorithm on the solution of the boundary vorticity values. The approximation of the boundary values effects

Table 1: Memory demands for storing one of the matrices [Di]Ω/Γ

	$ORG_{\alpha=1.0}$	$RAM_{\alpha=0.5}$[MB]	$RAM_{\alpha=0.2}$[MB]	$RAM_{\alpha=0.1}$[MB]
25^3	320	239	122	82
41^3	4345	3083	1833	1424
49^3	8828	5859	3171	2294

Table 2: Number of iteration to solve the kinematics equation with the proposed algorithm at different compression ratios α.

Re=100	$ORG_{\alpha=1.0}$	$Niter_{\alpha=0.8}$	$Niter_{\alpha=0.5}$	$Niter_{\alpha=0.2}$	$Niter_{\alpha=0.1}$
25^3	728	728	728	728	821
41^3	766	766	765	769	841
49^3	800	800	799	830	939
Re=1000	$ORG_{\alpha=1.0}$	$Niter_{\alpha=0.8}$	$Niter_{\alpha=0.5}$	$Niter_{\alpha=0.2}$	$Niter_{\alpha=0.1}$
25^3	2000	2184	2184	1896	8000
41^3	2401	2401	2401	2204	2331
49^3	2561	2561	2561	2679	3748

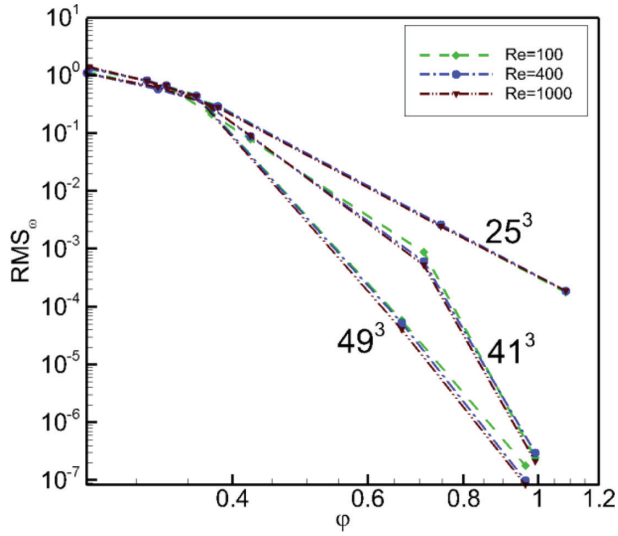

Figure 3: Influence of the Reynolds number and the grid density on the accuracy of the solution of the boundary vorticity values RMS$_\omega$.

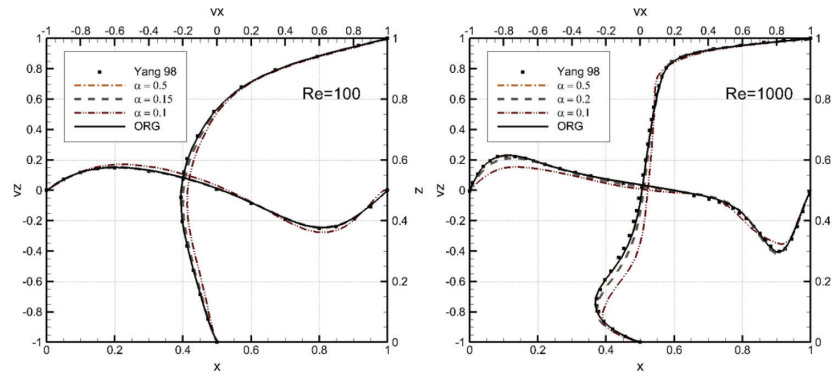

Figure 4: Velocity profiles vx(z) and vz(x) at the y=0.5 plane, for different compression ratio and Reynolds number. The grid density was 49^3.

the hole velocity field. In Fig. 4, we show the velocity profile through the centre of the cavity. We compared the benchmark results of Yang [18] with our method. We observed very good agreement between the benchmark result and our results obtained without the approximation method. When the approximation method is used, the difference in the profile increases. We also observed that the difference rises with a higher Reynolds number, and declines with a finer grid density.

6 CONCLUSION

In this study, we developed an accelerated algorithm based on BEM, for solving the kinematics equation. The developed algorithm used the H-matrix structure and the cross approximation method to decrease the computational demand of the BEM-based solver for incompressible

fluid flow. We have shown that the cross approximation algorithm can be successfully applied to solve the kinematics equation. The implementation was tested using the 3D lid-driven cavity test case with the Reynolds number up to 1000 using several computational grids.

Results showed that the chosen solution accuracy of a selected grid density was dependent on the compression rate. However, also the number of iterations that were necessary to obtain a solution increased with a lower compression rate. A finer grid allows more compression. We plan to control the inaccuracy introduced into the boundary vorticity by the approximation by specifying the rank of the rank of the approximated matrix parts.

However, the elliptic integral kernel, which is in the integral form of the kinematics equation, is not very well suited for approximation. We can conclude that the usage of the cross approximation enables the usage of the flow solver on a finer grid at a reduced computational cost and thus allows the simulation of more complex flow structures.

ACKNOWLEDGEMENTS

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0196).

REFERENCES

[1] Wrobel, L.C., *The boundary element method*, John Willey and Sons, 2002.

[2] Börm, S., Grasedyck, L. & Hackbusch, W., Introduction to hierarchical matrices with applications. *Engineering Analysis with Boundary Elements*, **27**(5), pp. 405–422, 2003. https://doi.org/10.1016/s0955-7997(02)00152-2

[3] Hackbusch, W., A sparse matrix arithmetic based on H -matrices. *Computing*, **108**, pp. 89–108, 1999. https://doi.org/10.1007/s006070050015

[4] Börm, S., Approximation of integral operators by H2-matrices with adaptive bases. *Computing*, **74**(3), pp. 249–271, 2005. https://doi.org/10.1007/s00607-004-0106-y

[5] Bebendorf, M., Approximation of boundary element matrices. *Numerische Mathematik*, **86**(4), pp. 565–589, 2000. https://doi.org/10.1007/pl00005410

[6] Rjasanow, S. & Steinbach, O., *The fast solution of boundary integral equations,* Springer Science+Business Media, 2007.

[7] Tamayo, J.M., Heldring, A. & Rius, J.M., Multilevel adaptive cross approximation (MLACA), *IEEE Antennas and Propagation Society International Symposium Antennas and Propagation,* pp. 2008–2011, 2009.

[8] Bebendorf, M. & Rjasanow, S., Adaptive low-rank approximation of collocation matrices. *Computing*, **70**(1), pp. 1–24, 2003. https://doi.org/10.1007/s00607-002-1469-6

[9] Rjasanow, S., Adaptive cross approximation of dense matrices, *IABEM 2002 Symposium*, pp. 1–12, 2002.

[10] Börm, S. & Grasedyck, L., Hybrid cross approximation of integral operators, *Numerische Mathematik*, **101**(2), pp. 221–249, 2005. https://doi.org/10.1007/s00211-005-0618-1

[11] Schröder, A., Brüns, H.-D. & Schuster, C., Fast evaluation of electromagnetic fields using a parallelized adaptive cross approximation. *Antennas and Propagation, IEEE Transactions on*, **62**(5), pp. 2818–2822, 2014. https://doi.org/10.1109/tap.2014.2303819

[12] Kurz, S., Rain, O. & Rjasanow, S., Application of the adaptive cross approximation technique for the coupled BE-FE solution of symmetric electromagnetic problems. *Computational Mechanics*, **32**(4), pp. 423–429, 2003.
https://doi.org/10.1007/s00466-003-0511-7

[13] Grytsenko, T. & Galybin, A.N., Numerical analysis of multi-crack large-scale plane problems with adaptive cross approximation and hierarchical matrices. *Engineering Analysis with Boundary Elements*, **34**(5), pp. 501–510, 2010.
https://doi.org/10.1016/j.enganabound.2009.12.001

[14] Maerten, F., Adaptive cross-approximation applied to the solution of system of equations and post-processing for 3D elastostatic problems using the boundary element method. *Engineering Analysis with Boundary Elements*, **34**(5), pp. 483–491, 2010.
https://doi.org/10.1016/j.enganabound.2009.10.016

[15] Wei, X., Chen, B., Chen, S. & Yin, S., An ACA-SBM for some 2D steady-state heat conduction problems. *Engineering Analysis with Boundary Elements*, **71**, pp. 101–111, 2016.
https://doi.org/10.1016/j.enganabound.2016.07.012

[16] Škerget, L., Hriberšek, M. & Žunic, Z., Natural convection flows in complex cavities by BEM. *International Journal of Numerical Methods for Heat & Fluid Flow*, 13(6), pp. 720–735, 2003.
https://doi.org/10.1108/09615530310498394

[17] Ravnik, J., Škerget, L. & Zunič, Z., Velocity-vorticity formulation for 3D natural convection in an inclined enclosure by BEM. *International Journal of Heat and Mass Transfer*, **51**(17–18), pp. 4517–4527, 2008.
https://doi.org/10.1016/j.ijheatmasstransfer.2008.01.018

[18] Yang, J., Yang, S., Chen, Y. & Hsu, C., Implicit weighted ENO schemes for the three-dimensional incompressible Navier – Stokes equations. *Journal of Computational Physics*, **487**, pp. 464–487, 1998.
https://doi.org/10.1006/jcph.1998.6062