Radial rasis function for non-matching mesh interpolation in parallel solving FSI problem

Sergey Kopysov, Igor Kuzmin*,
Alexander Novikov and Nikita Nedozhogin

Laboratory of Computing Technologies,
Udmurt Federal Research Center,
The Ural Branch of the Russian Academy of Sciences,
Izhevsk, Russia
Email: s.kopysov@gmail.com
Email: imkuzmin@gmail.com
Email: sc_work@mail.ru
Email: nedozhogin@inbox.ru
*Corresponding author

Leonid Tonkov

Department of Computational Mechanics,
Institute of Mathematics, Information Technologies and Physics,
Udmurt State University,
Izhevsk, Russia
Email: letonkov@gmail.com

Abstract: In strongly coupled fluid-structure interaction simulations, the fluid dynamics and solid dynamics problems are solved independently on their own meshes. Therefore, it becomes necessary to interpolate the physical properties (pressure, displacement) across two meshes. In the present paper, we propose to accelerate the interpolation process by the method of radial basis functions using the matrix-free solution of the system of equations on a GPU. Also, we reduce the number of equations in the system by using an adaptive algorithm for choosing interpolation points.

Keywords: parallel computing; hybrid HPC platforms; fluid-structure interaction; FSI; radial basis functions; RBF; layer-by-layer partitioning.

Reference to this paper should be made as follows: Kopysov, S., Kuzmin, I., Novikov, A., Nedozhogin, N. and Tonkov, L. (2019) 'Radial rasis function for non-matching mesh interpolation in parallel solving FSI problem', *Int. J. Engineering Systems Modelling and Simulation*, Vol. 11, No. 1, pp.19–25.

Biographical notes: Sergey Kopysov received his PhD in Mathematical simulation, numerical methods and software from the Institute of Mathematical Modeling Russian Academy of Sciences in 1992. Since then, he has been with the Institute of Applied Mechanics, Russian Academy of Sciences, and as the Director of Department of Computing and Information Technologies since 1995. He received Doctor of Physics and Mathematics in Mathematical Simulation, Numerical Methods and Software in 2007 and is a Full Professor in 2013. He has authored and co-authored more than 100 academic papers. His research interests include decomposition methods, parallel computing, adaptive finite element method, methods for constructing computational meshes.

Igor Kuzmin received his MSc in Applied Mathematic from the Udmurt State University in 2009. He is currently a Post-Graduate Research Fellow in the Udmurt Federal Research Center, Russian Academy of Sciences. His research interests include high-performance computing, parallel algorithms, fluid-structure interaction problems.

Aleksandr Novikov received his Engineer qualification in Engines from the Izhevsk State Technical University in 1994, the PhD in Mathematical Simulation, Numerical Methods and Software from the Institute of Mathematical Modeling Russian Academy of Sciences in 2005. He is currently a Postdoctoral Research Fellow in the Udmurt Federal Research Center, Russian Academy of Sciences. His research interests include numerical methods, finite element method, parallel algorithms, HPC, unstructured mesh generation and adaptation.

Nikita Nedozhogin received his MSc in Applied Mathematic and Computer Science from the Udmurt State University in 2011. He is currently a Post-Graduate Research Fellow in the Udmurt Federal Research Center, Russian Academy of Sciences. His research interests include decomposition method, parallel programming and linear algebra.

Leonid Tonkov received his Engineer qualification in Engines from the Izhevsk State Technical University in 1993, the PhD in Gas Dynamics from the Institute of Applied Mechanics Ural Branch of Russian Academy of Sciences in 2000. He is currently a Postdoctoral Research Fellow in Udmurt Federal Research Center, Russian Academy of Sciences. His research interests include numerical methods, computational mathematics, fluid dynamic problems, algorithms, software and hardware of parallel computing.

1 Introduction

Several approaches are used to simplify algorithms for the numerical solving of the fluid-structure interaction (FSI) problem such as the decreasing of the problem size, *algorithms accelerating the computations*, the analysis methods and the use of the properties of subtasks.

Let us consider the variants of decreasing the computational costs at the interpolation of data between non-matching meshes: the decrease of the interpolation data; the parallelisation of algorithms taking into account the geometry of the extended boundaries of axisymmetric bodies.

The main methods for interpolation on non-matching meshes for the FSI simulations are surveyed in Berndt et al. (2011) and Farrell et al. (2009). We consider a method based on radial basis functions (RBF) (De Boer et al., 2007), where the coefficients of the interpolant are found from the system of equations, the matrix of which is formed using a RBF. The choice of the function determines the condition number and density of the matrix, and, as a result, the computational complexity of solving the system of equations.

The RBF interpolation has the following advantages:

- it does not require any mesh connectivity information
- it requires solving a sparse system of equations, especially with the compact basis functions
- it can be efficiently parallelised.

This paper is structured as follows. Section 2 briefly describes the RBF interpolation scheme for the FSI problem. Section 3 presents a new approach based on layer-by-layer mesh partitioning for reducing the problem size. Section 4 describes a matrix-free solution of the interpolation problem on a GPU. Finally Section 5 will be the conclusions.

2 RBF interpolation for FSI problems

Let P^d_{Q-1} be the d-dimensional space of the polynomials of a degree smaller than $Q-1$ and $p_1, \ldots, p_q$ be a basis in this space. The main idea of the RBF method is to find the required interpolation function as a linear combination of the following functions:

$$w(x_i) = \sum_{j=1}^{n} \alpha_j \phi(\|x_i - x_j\|) + \sum_{l=1}^{q} \beta_l p_l(x_i), \tag{1}$$

where q is additional degrees of freedom and the coefficients α_i and the polynomials $p_l(x_i)$ satisfy

$$\sum_{j=1}^{n} \alpha_j p_l(x_j) = 0, \quad 1 \le l \le q. \tag{2}$$

The solution of the system exists and is unique if

$$\begin{aligned} &p(x_j) = 0, \text{ for all } 1 \le j \le n \text{ and} \\ &p \in P^d_{Q-1} \text{ implies } p = 0. \end{aligned} \tag{3}$$

The system of equations (1)–(3) is always solvable if ϕ is a positive-definite RBF.

Let Ω be the domain with the given pressure p_Ω. The domain with the required pressure is denoted by Φ. The pressure interpolation between the meshes can be expressed in the matrix form as follows:

$$\begin{bmatrix} W_{\Omega\Omega} & P_\Omega \\ P_\Omega^T & O \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} p_\Omega \\ 0 \end{bmatrix} \quad \text{or} \quad A\,\gamma = b, \tag{4}$$

here $W_{\Omega\Omega}$ is the $n_\Omega \times n_\Omega$ matrix, consisting of the elements $\phi(\|\mathrm{x}^i_\Omega - \mathrm{x}^j_\Omega\|)$ and $1 \le i, j \le n_\Omega$; P_Ω is the column matrix consisting of the elements $\left[1\ \mathrm{x}^i_\Omega\right]$; α, β are the coefficients of the interpolant; n_Ω is the number of interpolation points of the domain. The target pressure vector p_Φ is obtained by the matrix-vector product

$$p_\Phi = [W_{\Phi\Omega} P_\Phi] \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{5}$$

where $W_{\Phi\Omega}$ is the $n_\Phi \times n_\Omega$ matrix, consisting of the elements $\phi(\|\mathrm{x}^i_\Phi - \mathrm{x}^j_\Omega\|)$, $1 \le i \le n_\Phi$ and $1 \le j \le n_\Omega$; n_Φ is the number of interpolation points on the domain Φ; P_Φ is the column matrix $\left[1\ \mathrm{x}^i_\Phi\right]$. The dimension of the matrices P_Ω and P_Φ depends on the type of basis functions. For example, the dimension of the matrices $W_{\Omega\Omega}$ and $W_{\Phi\Omega}$ for the global RBF thin-plate spline is $3 \times n_\Omega$ and $3 \times n_\Phi$, respectively.

Solving the system of equation (4) is the most computationally expensive part of the interpolation. In

De Boer et al. (2006), it was shown that the choice of basis functions affected both the quality of the interpolation and the solution time. The functions providing more accurate interpolation may require a large amount of time for the solution. The computational cost can be optimised by

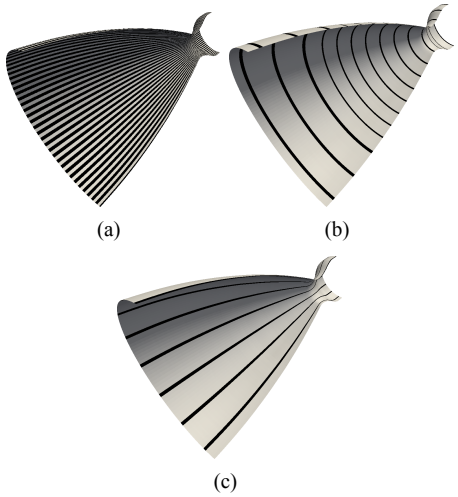
1 reducing the system

2 parallelising the steps of the pre-conditioning and solution of sparse/dense systems of equations.

3 Reducing the size of the system of equations

In this section, we demonstrate reducing the size of the system of equations for the FSI of a supersonic flow with a nozzle wall that has a high geometric expansion ratio (Zhao et al., 2013). The boundary along which the computational data are interpolated is quite long and the pressure is irregularly distributed along the boundary Ω (the nozzle wall). The solution of the above problems is considered within the framework of the layer-by-layer mesh partitioning method proposed in our previous work (Novikov et al., 2016). The method provides a conflict-free data access during the parallel summation of the components of the finite element vectors in the shared memory of the multi-core computing systems.

Figure 1 (a) The interface boundary partitioned into 150 layers (b) The partition into 15 layers parallel to the directrix (c) The partition into 15 layers by the surface generatrix



(a) (b) (c)

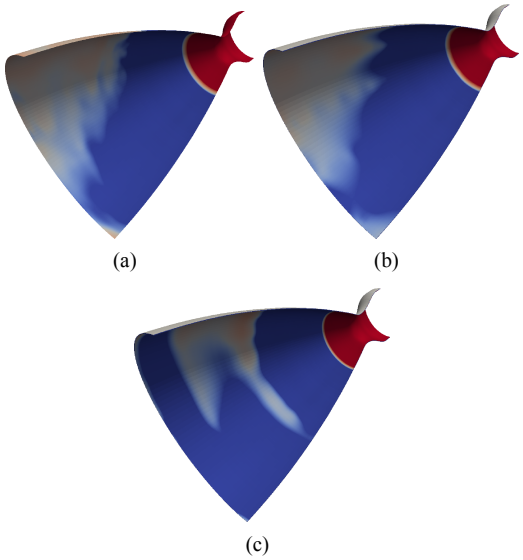
Let us divide the interface part $\Gamma_{\Omega^h} = \partial\Omega^h$ of the mesh Ω^h into layers. To do this, we use the neighbourhood criterion where any two mesh cells are considered adjacent if they have at least one common node.

The considered physical area and the computational mesh are symmetrical. Therefore, the choice of the initial set of interpolation points is carried out in accordance with the distribution of the layers. Here, there are two possibilities for selecting layers: along the generatrix and along the directrix.

The mesh Γ_{Ω^h} is the discrete description of the rotation surface Γ_Ω with the closed directrix. To form layers in parallel to the directrix Γ_Ω [see Figure 1(b)] or along the surface generatrix [see Figure 1(c)], we use the algorithm proposed in Novikov et al. (2016).

The layer-by-layer partitioning is used to reduce the number of interpolation points. To construct the interpolant, we choose those layers of the interface surface which most accurately represent the distribution of the interpolated data (pressure). Figure 1 shows the partitioning of the surface mesh into 150 layers. The dark layers correspond to 15 layers involved in the pressure interpolation [Figures 1(b) and 1(c)].

Figure 2 (a) The given pressure distribution at the boundary of the domain Ω (b) The pressure distribution obtained at the use of 15 out of 150 layers at the radial partition (c) At the partition along the generatrix (see online version for colours)



(a) (b) (c)

The quality of interpolation is compared for the local $\phi(\|\mathrm{x}\|) = 1 - \|\mathrm{x}\|$, $\phi(\|\mathrm{x}\|) = (1 - \|\mathrm{x}\|)^2$, the global $\phi(\|\mathrm{x}\|) = e^{-\|\mathrm{x}\|^2}$, $\phi(\|\mathrm{x}\|) = \|\mathrm{x}\|^2 \log \|\mathrm{x}\|$ basis functions and inverse distance weighting (IDW) (Shepard, 1968), using different partitions and numbers of layers. The quality of the pressure interpolation can be estimated as the relative error computed by the ratio of the norms of the resultant forces of the pressure on the interface boundary.

Table 1 shows the results for the pressure interpolation in parallel to the directrix (Radial partitioning) and along the surface generatrix (longitudinal partitioning). In the second column, the evaluation of the interpolation quality is given for all possible interpolation points of Γ_{Ω^h}.

The quality of the interpolation with the data reduction depends not only on the number of interpolation points but also on the choice of the points (Figure 2). When compact basis functions in the form $(1 - \|\mathrm{x}\|)^2$ are used, the orientation of the pressure distribution after the interpolation depends on the partitioning. The best interpolation is achieved for the radial partitioning of the domain. The RBF method using the global basis function $\phi(\mathrm{x}) = \|\mathrm{x}\|^2 \log \|\mathrm{x}\|$ gives the best results. It allows to

reduce the number of equations in system (4) by a factor of 15 with the acceptable quality of the interpolation. The IDW interpolation gives the greatest error, even in the case of the full data.

The obtained matrix A is ill-conditioned. When using global basis functions, the condition number depends on the number and location of the interpolation points, as well as on the type of the basis functions. The distance between the interpolation points influences the condition number. As the number of interpolation points increases, the distance between them decreases, and the condition number increases.

Table 1 Relative error of the pressure interpolation, %

n_Ω	28,800	9,600	5,760	2,880	960
Longitudinal distribution					
$1 - \|\mathrm{x}\|$	0.24	3.79	10.4	15.2	349.6
$(1 - \|\mathrm{x}\|)^2$	0.78	7.79	69.3	74.3	365.4
$\mathrm{IDW}_{p=3}$	14.2	50.6	105	189	407.0
$e^{-\|\mathrm{x}\|^2}$	0.54	1.18	2.68	8.04	147.6
$\|\mathrm{x}\|^2 \log \|\mathrm{x}\|$	0.01	0.23	1.07	5.95	28.61
Radial distribution					
$1 - \|\mathrm{x}\|$	0.24	0.34	0.92	10.8	214.3
$(1 - \|\mathrm{x}\|)^2$	0.78	1.58	9.34	15.9	36.5
$\mathrm{IDW}_{p=3}$	14.2	53.9	111	205	534.2
$e^{-\|\mathrm{x}\|^2}$	0.54	1.07	3.99	5.41	101.6
$\|\mathrm{x}\|^2 \log \|\mathrm{x}\|$	0.01	0.01	0.21	1.13	21.6
Adaptive distribution					
$1 - \|\mathrm{x}\|$	0.24	1.23	1.14	0.93	4.89
$(1 - \|\mathrm{x}\|)^2$	0.78	0.67	1.75	2.41	1360
$\mathrm{IDW}_{p=3}$	14.2	26.1	27.3	30.8	85.1
$e^{-\|\mathrm{x}\|^2}$	0.54	0.94	1.12	0.82	9.61
$\|\mathrm{x}\|^2 \log \|\mathrm{x}\|$	0.01	0.37	1.25	1.31	3.65

For interpolation it is practical to use an adaptive algorithm. It should be noted that the interpolation error is not the only reason for its use. In De Marchi et al. (2005), it was shown that in the RBF-based interpolation, the optimal distribution for the case of a two-dimensional space was the asymptotically uniform distribution of points. In the considered case, at the initial non-uniform distribution of the data, the adaptive choice of the interpolation points is preferable since it allows obtaining the interpolation point distribution providing a minimum error. Note that in the case of the asymptotically uniformly distributed interpolation points, theoretically, an arbitrarily high error can be obtained for some functions; however, for others it is unattainable. Even the use of greedy algorithms (Rendall and Allen, 2009) does not provide an optimal interpolation by the RBF method for an arbitrary function.

The aim of the adaptive algorithm is to achieve a sufficiently small error for the interpolation with the use of $n_a \ll n_\Omega$ points only, where n_Ω is the initial number of interpolation points.

The adaptive algorithm starts with a very small number of points and then refines the dataset by adding new points of the interpolation where the observed interpolation error is largest. The algorithm constructs an interpolant for the reduced set of points, which reproduces the interpolation function in certain tolerance limits. Furthermore, limits can be imposed on the number of iterations and the total number of interpolation points (n_a).

3.1 *Adaptive algorithm for selecting interpolation points*

Let us

1 select an initial set of layers and solve $A\gamma = b$ for n_a init points

2 form and solve the linear system of the interpolation coefficients for the initial layers $A\gamma = b$

3 evaluate the interpolant at all n_Ω data points $p_\Phi = A\gamma$

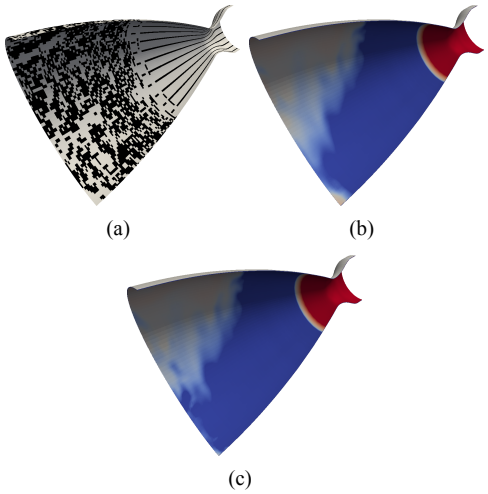
4 compute the residual vector or find errors $E(\mathrm{x}) = p(\mathrm{x}) - p_\Omega(\mathrm{x})$

5 check the stopping criteria, and if they are not satisfied, increase the iteration count and add new points with the largest error $E(\mathrm{x})$

6 the adaptation is successfully completed if the residual is smaller than the given tolerance.

In the fluid dynamics problem on a flow in a supersonic over expanded nozzle, the line $\ell = \{(\arg\min_x p(x, \varphi_*), \varphi_*) \in \Gamma : 0 \le \varphi_* < 2\pi\}$ divides the set of interpolation points Ω into Γ_1, where $p = p(x)$ is the axial symmetry region, and Γ_2, where $p = p(x, \varphi)$ [Figures 3, 4 and 5(a)].

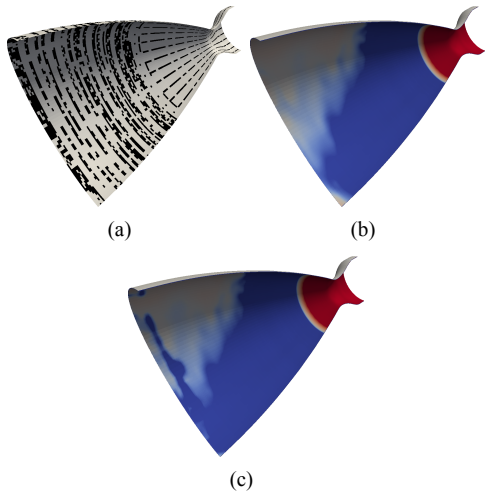
Figure 3 Pressure distribution, (a) adaptive distribution with n_Ω = 9,600 (b) $e^{-\|x\|^2}$ (c) $1 - \|x\|$ (see online version for colours)



Let us apply the above adaptive algorithm to the pressure interpolation for the considered FSI problem. At the first step, we apply it to the layers to which the set of interpolation points is divided, i.e., we reduce the set of interpolation points along the coordinate x. Thus, from

all the layers we leave the minimum number of layers satisfying the error of interpolation. At the second step, we apply this algorithm to each remaining layer to select the interpolation points with the largest error in the coordinate φ. Thus, we reduce the number of the interpolation points preserving the interpolation error.

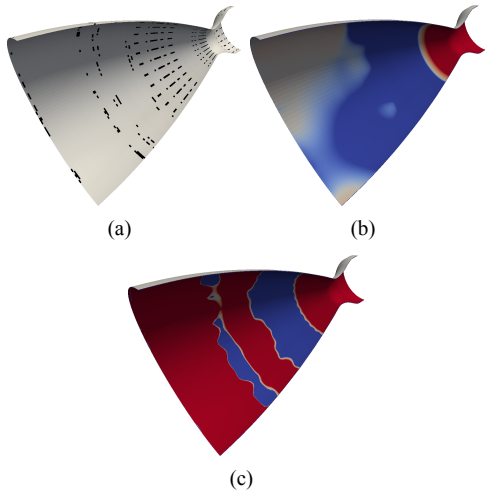
Figure 4 Pressure distribution, (a) adaptive distribution with n_Ω = 5,760 (b) $e^{-\|x\|^2}$ (c) $1 - \|x\|$ (see online version for colours)



(a) (b)

(c)

Figure 5 Pressure distribution, (a) adaptive distribution with n_Ω = 960 (b) $e^{-\|x\|^2}$ (c) $1 - \|x\|$ (see online version for colours)



(a) (b)

(c)

In addition to the pressure $p = p(x, \varphi)$ determined on Ω, the gradient grad p is also known. It is used as an indicator of the addition of interpolation points. Layers and interpolation points are added to the minimal set of points when they are located in regions of the largest gradient.

In the absence of any additional information on the distribution of the interpolated data (e.g., in the case of the pressure interpolation it is a gradient), the indicator of the interpolation error is formed on the basis of local basis functions or by the IDW method.

Table 1 shows the error of the pressure interpolation using the adaptive distribution of interpolation points. It should be noted that the adaptive choice of interpolation points increases of the condition number of the matrix of the system of equation (4) and thus, the iterative solution process converges slowly.

The adaptive distribution of the interpolation points using the local basis function $1 - \|x\|$ substantially reduces the error for any number of interpolation points, and another local basis function $(1 - \|x\|)^2$ has a large error for a small number of points. The Gaussian global function [Figure 5(b)] shows good results for a small number of the adaptively chosen points n_Ω = 960.

4 Matrix-free solution of interpolation problem on GPU

One of the specific features of the system (4) is a dense matrix, which imposes some restrictions on the GPU use due to the small capacity of the available GPU memory. The problem can be resolved by

1 using several GPUs, thereby increasing the total memory available for the system solution

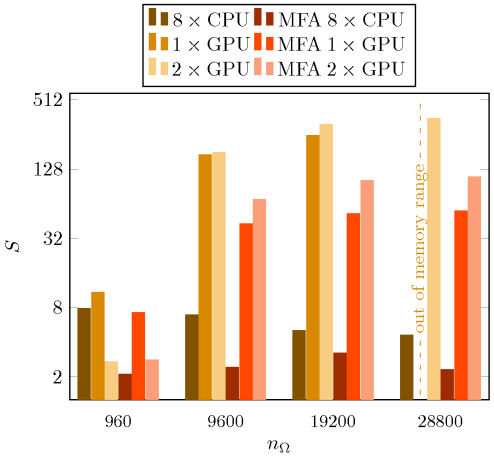
2 solving the system of equations without the formation of a matrix (matrix-free algorithm).

In this case, the matrix elements are computed as they are required in the algorithm of the system solution. The solution of the system by the RBF method is possible without the formation of a matrix. since the matrix elements are computed by the chosen basis function. This improves the data locality and arithmetic intensity for matrices and vectors. The memory requirements and CPU-GPU communications are reduced. The efficiency of the algorithm can be improved if multi-GPUs are used in the similar way to that in Kopysov et al. (2014).

Let us consider in more detail the MFA computing expenses. In the MFA, the formation time is excluded. The time to copy the matrix into the GPU memory is additionally taken into account when using the algorithm with the assembled matrix. In addition, Figure 6 shows the acceleration s for solving the system with the use of both the algorithm with an assembled matrix and the matrix-free solution algorithm. The acceleration is computed as $S = t_1/t_p$, where t_1 – the time of execution of task per one CPU thread and t_p – the time required to complete the task on processors (by using CPU, GPU or their mixture).

The CPU parallelisation is carried out with OpenMP. The solution of the system of equations on several GPUs is carried out by CUDA in conjunction with OpenMP. The system of equations is solved by the conjugate gradient method with the diagonal preconditioner (Kopysov et al., 2014). The precision is equal to 10^{-6}. In the computations, double-precision arithmetic is used. The analysis and performance estimations are performed on a computing node consisting of 2 $\times$ quad – core Intel Xeon processor E5-2609, 2 $\times$ GeForce GTX 980 with 4 GB GDDR.

Figure 6 Accelerating of interpolation, obtained by using different approaches for function $e^{-\|x\|^2}$ (see online version for colours)



When the system of equations is solved using the assembled matrix on the CPU, the step of the matrix formation is added. The use of GPU increases the cost due to the necessity of copying the data to the GPU. When the system is solved using the MFA the cost is not increased because there in no need to copy the data. Figure 6 shows, the reducing of total time is given for each of the above approaches.

The numerical computations show that the use of eight CPU threads within one computing node reduces the solution time almost by a factor of seven. One GPU allows to speed up solving the system by a factor of 250 compared with one CPU thead and by a factor of 50 compared with 8 × CPU. The GPU efficiency increases with the increase of the system size. Using two GPUs reduces the time by a factor of 1.5 compared with one GPU and by a factor of 350 compared with the CPU. With an increase in the number of GPUs, the strong scalability can be provided only when the sizes of the submatrices on each GPU are preserved

The matrix-free solution of the system using 8 × CPU reduces the solution time by a factor of 2.5. However, the solution with the assembled matrix is twice as fast as the matrix-free solution. When one GPU is used, the time for the matrix-free solution of the system of equations is 5 times larger than that for the solution with the assembled matrix, and in the case of using two GPUs, the matrix-free solution is 3 times longer. The speedup obtained at the use of one CPU thread is 55 times smaller than that when using one GPU and 110 times smaller than that when using two GPUs. It should be noted that the use of local basis functions with an introduced radius of influence increases the MFA efficiency.

Let us estimate the maximum size of the system, which can be solved using the MFA on a one GPU. For the matrix A formation, the coordinates of the interpolation points are used. Then for interpolation in a three-dimensional space, it is necessary to allocate memory for the vector of coordinates of length equal to $n_\Omega \times 3$. The required memory size for solving the system with the assembled matrix is $n_\Omega \times n_\Omega$. The remaining vectors participating in the conjugate gradient method coincide for both algorithms. Thus, the memory size for interpolating the mesh data in the three-dimensional space is decreased by a factor of n_Ω / 3. The maximum system size solved by the MFA increases by the same factor. The algorithm of the conjugate gradient method with a diagonal preconditioner involves the use of memory to store a matrix of size $n_\Omega \times 3$ (the MFA) and six vectors $n_\Omega \times 1$. Thus, for solving the system using the MFA and double precision arithmetic, $n_\Omega \times (3 + 6) \times 8$ bytes are required. Consequently, the maximum size of a system for the GPU with a 4 GB GDDR is about 6×10^8 equations. Using two graphics cards, the possible size of the system is increased to 1.2×10^9 equations. Thus, for the dense matrices obtained on the basis of global basis functions, a parallel method of conjugate gradients is constructed. The computations are distributed among several GPUs. The use of the matrix-free approach makes it possible to remove any limitations on the amount of memory.

5 Conclusions

We propose an algorithm for constructing a uniform distribution of interpolation points on an unstructured mesh in the interpolation based on RBF. For solving the problems on unstructured meshes, the adaptive algorithm is proposed for sampling the data for the interpolation based on the RBF method. The obtained results show that the interpolation on the uniformly distributed data is of high quality and applicable for meshes of super-large dimensions. The use of the adaptive data reduction based on the layer-by-layer partition of the mesh makes it possible to reduce the number of interpolation points, but requires additional information about the data. At the same time, the quality of the interpolation for the irregularly-spaced data is preserved both for global and for local basis functions. Using a matrix-free algorithm on large meshes significantly reduces the memory costs associated with the formation of the interpolation matrix. At the same time, the computation locality of the matrix-vector product computations increases when solving the system of equations by iterative methods. The solution of systems with dense matrices by the MFA on the CPU does not lead to any significant time reductions. Since the time of the matrix formation is less than 1% of the solution time, the use of the MFA in conjunction with the CPU is inefficient. The matrix-free approach is most effective when using a GPU, especially when it is not possible to achieve a large reduction of points without the interpolation quality loss. Using a GPU for solving larger systems of equations allows minimising the cost of additional computations associated with the formation of the matrix elements.

A further increase in the efficiency of the MFA is associated with a decrease in the number of iterations of the algorithm for solving the system of equation by constructing effective parallel preconditioners and by using local basis functions with the radius of influence.

Acknowledgements

This work is supported by the Russian Foundation for Basic Research (projects: 17-01-00402 a, 16-01-00129 a, 16-41-180276 r_ural_a).

References

Berndt, M., Breil, J., Galera, S., Kucharik, M., Maire, P-H. and Shashkov, M. (2011) 'Two-step hybrid conservative remapping for multimaterial arbitrary Lagrangian–Eulerian methods', *Journal of Computational Physics*, Vol. 230, No. 17, pp.6664–6687.

De Boer, A., Van der Schoot, M.S. and Bijl, H. (2006) 'New method for mesh moving based on radial basis function interpolation', in *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, European Community on Computational Methods in Applied Sciences (ECCOMAS)*, Delft University of Technology, Egmond aan Zee, The Netherlands, 5–8 September.

De Boer, A., Van der Schoot, M.S. and Bijl, H. (2007) 'Mesh deformation based on radial basis function interpolation', *Computer and Structures*, Vol. 85, pp.784–795.

De Marchi, S., Schaback, R. and Wendland, H. (2005) 'Near-optimal data-independent point locations for radial basis function interpolation', *Advances in Computational Mathematics*, Vol. 23, No. 3, pp.317–330.

Farrell, P.E., Piggott, M.D., Pain, C.C., Gorman, G.J. and Wilson, C.R. (2009) 'Conservative interpolation between unstructured meshes via supermesh construction', *CMAME*, Vol. 198, Nos. 33–36, pp.2632–2642.

Kopysov, S.P., Kuzmin, I.M., Nedozhogin, N.S., Novikov, A.K. and Sagdeeva, Y.A. (2014) 'Scalable hybrid implementation of the Schur complement method for multi-GPU systems', *Journal of Supercomputing*, Vol. 69, pp.81–88.

Novikov, A., Piminova, N., Kopysov, S. and Sagdeeva, Y. (2016) 'Layer-by-layer partitioning of finite element meshes for multicore architectures', *Communications in Computer and Information Science*, Vol. 687, pp.106–117.

Rendall, T.C.S. and Allen, C.B. (2009) 'Efficient mesh motion using radial basis functions with data reduction algorithms', *J. Comput. Phys.*, Vol. 228, No. 17, pp.6231–6249.

Shepard, D. (1968) 'A two-dimensional interpolation function for irregularly-spaced data', in *Proceedings of the 1968 23rd ACM National Conference, ACM'68*, ACM, pp.517–524.

Wang, T-S., Zhao, X. and Zhang, S. (2013) 'Aeroelastic modeling of a nozzle startup transient', in *Journal of Propulsion and Power*, Vol. 30.

Zhao, X., Bayuuk, S. and Zhang, S. (2013) 'Aeroelastic response of rocket nozzles to asymmetric thrust loading', in *Computers and Fluids*, Vol. 76, pp.128–148.