Parallel Interpolation in FSI Problems Using Radial Basis Functions and Problem Size Reduction*

Sergey Kopysov, Igor Kuzmin, Alexander Novikov, Nikita Nedozhogin, and Leonid Tonkov

Institute of Mechanics, Ural Branch of the Russian Academy of Sciences, Izhevsk, Russia
{s.kopysov,i.m.kuzmin}@gmail.com, sc_work@mail.ru, nedozhogin@inbox.ru, tnk@udman.ru

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 physical properties (pressure, displacement) across two meshes. In this work, we propose to accelerate the interpolation process by the method of radial basis functions using the matrix-free solution of the equation system on a GPU.

Keywords: parallel computing · hybrid HPC platforms · fluid-structure interaction · radial basis functions · layer-by-layer partitioning

1 Introduction

Main interpolation methods on non-matching meshes for fluid-structure interaction (FSI) simulations are overviewed in [1, 4]. We consider the one based on radial basis functions (RBF) [2], where, the coefficients of the interpolant are found from the system of linear equations with the matrix formed by means of a radial basis function.

The choice of the function determines the conditioning and density of the matrix, and, as a result, the computational complexity of solving the system of equations. The advantages of the RBF interpolation are the following:

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

This paper is structured as follows. Section 2 briefly describes the RBF interpolation scheme for the FSI problem. The next section presents a new approach

* This work is supported by the Russian Foundation for Basic Research (projects: 16-37-00060-mol_a, 16-01-00129-a, 17-01-00402-a).

based on layer-by-layer mesh partitioning for reducing the problem size. The fourth section describes a matrix-free solution of the interpolation problem on a GPU.

2 RBF Interpolation for FSI Problems

Consider the problem of interpolation by the method of radial basis functions, in the context of pressure interpolation in the FSI problem. Let Ω be the mesh approximation of the domain with the boundary $\Gamma_\Omega = \partial\Omega$ and the given pressure p_{Γ_Ω}. The mesh approximation of the domain with the interpolated pressure is denoted by Φ with the boundary $\Gamma_\Phi = \partial\Phi$. The domains Ω and Φ have a common part of boundary (interface part), i.e. $\Gamma_\Omega \cap \Gamma_\Phi \neq \{\emptyset\}$. The pressure interpolation between these meshes can be expressed in the matrix form as follows:

$$\begin{bmatrix} W_{\Gamma_\Omega\Gamma_\Omega} & P_{\Gamma_\Omega} \\ P^T_{\Gamma_\Omega} & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} p_{\Gamma_\Omega} \\ 0 \end{bmatrix} \quad \text{or} \quad A\,\gamma = b \ , \tag{1}$$

and the target pressure vector p_{Γ_Φ} is obtained by the matrix-vector product

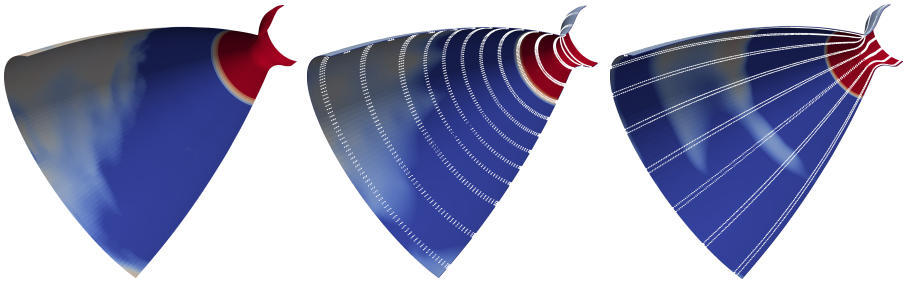
$$p_{\Gamma_\Phi} = [W_{\Gamma_\Phi\Gamma_\Omega} P_{\Gamma_\Phi}] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \ , \tag{2}$$

where $W_{\Gamma_\Omega\Gamma_\Omega}$, $W_{\Gamma_\Phi\Gamma_\Omega}$ are the $n_{\Gamma_\Omega} \times n_{\Gamma_\Omega}$ and $n_{\Gamma_\Phi} \times n_{\Gamma_\Omega}$ matrices, consisting of the elements with values equal $\phi(\,\|\mathrm{x}^i_{\Gamma_\Omega} - \mathrm{x}^j_{\Gamma_\Omega}\|_2\,)$ and $\phi(\,\|\mathrm{x}^i_{\Gamma_\Phi} - \mathrm{x}^j_{\Gamma_\Omega}\|_2\,)$ respectively; n_{Γ_Ω} and n_{Γ_Φ} are the numbers of interpolation points on the interface boundary of the domains; α, β are the coefficients of the interpolant; x^i — the vector of coordinates of the interpolation points.

The function $\phi(\|\mathrm{x}\|_2)$ reduce to a scalar function of the Euclidean norm $\|\mathrm{x}\|_2$ of their vector argument x, i.e.: they are radial in the sense $\phi(\|\mathrm{x}\|_2) = \phi(r)$, $\mathrm{x} \in \mathbb{R}^3$ for the "radius" $r = \|\mathrm{x}\|_2$ with a scalar function $\phi : \mathbb{R} \to \mathbb{R}$. This makes their use for highdimensional reconstruction problems very efficient, and it induces invariance under orthogonal transformations.

There are two types of radial basis functions: basis functions with global and compact support. The basis functions with global support are Gaussian ($\phi(r) = e^{-c\,r^2}$, $r \geq 0$, $c > 0$), inverse multiquadric ($\phi(r) = (r^2 + c^2)^{-1/2}$, $r \geq 0$, $c > 0$), thin plate spline ($\phi(r) = r^2 \log r$, $r \geq 0$), cubic ($\phi(r) = r^3$, $r \geq 0$). The basis functions with compact support have the form: $\phi(r) = (c - r)^2$, $r \geq 0$, $c > 0$, $\phi(r) = (c - r)^4\,(4r + 1)$, $r \geq 0$, $c > 0$, etc. Further, the constant c is considered from the point of view of the influence on the error and the rate of convergence of the solution of the system (1) on the example of the global basis function $\phi(r) = e^{-c\,r^2}$.

Solving the system of equations (1) is the most computationally expensive part of the interpolation. In [3], 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 optimized by (i) reducing the system; (ii) choosing constant c and (iii) parallelizing the steps of the preconditioning and solution of sparse/dense systems of equations.



(a) (b) (c)

Fig. 1. (a) The initial pressure distribution, (b) the result of interpolation when partitioned into 15 layers parallel to the directrix and (c) partitioned into 15 layers parallel to the generatrix

3 Reducing the Size of the System of Equations

In this section, we demonstrate reducing the size of the system of equations for the fluid-structure interaction of a supersonic flow with a nozzle wall that has a high geometric expansion ratio [7]. 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 [6]. 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.

Let us divide the interface part Γ_Ω of the mesh Ω into layers. To do this, we use the neighborhood criterion where any two mesh cells are considered adjacent if they have at least one common node.

The mesh Γ_Ω is the discrete approximation of the rotation surface with the closed directrix. To form layers in parallel to the directrix Γ_Ω (see Fig. 1 (b)) or along the surface generatrix (see Fig. 1 (c)), we use the algorithm proposed in [6], choosing the first layer of partitioning in the appropriate directions. Further, to reduce the number of interpolation points, we choose only a few layers of the fliud-structure interface surface.

The quality of interpolation is compared for the global $\phi(r) = e^{-c\,r^2}$, using different partitions, numbers of layers and constant c. 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 last column, the evaluation of the interpolation quality is given for all possible interpolation points (28800) of Γ_Ω. Therefore, the results shown in this column do not depend on the partitioning.

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 (Fig. 1).

Table 1. Relative error of the pressure interpolation for $e^{-c\,r^2}$, %

n_{Γ_Ω}	1800			5700			9600			28800		
Radial partitioning												
c	0.1	1.0	10	0.1	1.0	10	0.1	1.0	10	0.1	1.0	10
iter	46	33	10	95	81	24	153	121	43	540	421	121
error	1.05	1.59	24.3	0.38	0.38	1.48	0.17	0.16	0.41	0.005	0.58	0.038
Longitudinal partitioning												
c	0.1	1.0	10	0.1	1.0	10	0.1	1.0	10	0.1	1.0	10
iter	205	156	49	318	231	69	403	285	85	540	421	121
error	4.14	2.36	37.1	0.26	0.15	3.18	0.07	0.04	0.69	0.005	0.58	0.038

In addition, the table 1 the convergence of the solution of the system (1) depends on the choice of interpolation points. So, when choosing points based on longitudinal partitioning, the number of iterations for solving the resulting system of equations is twice as large as for a radial partition, regardless of the number of equations. With an increase in the constant c, the number of iterations decreases, but the interpolation error increases. A similar situation is typical for local basis functions. The best interpolation is achieved for the radial partitioning of the domain with $c = 0.1$. It allows to reduce the number of equations in system (1) by a factor of 15 with the acceptable quality of the interpolation.

4 Matrix-Free Solving of Interpolation System on GPU

One of the specific features of the system (1) 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 (i) using several GPUs, thereby increasing the total memory available for the system solution; (ii) 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 [5].

Let us consider in more detail the MFA computing expenses. Table 2 shows the time of the sequential and parallel formation of the matrix A of the system (1). In the MFA, the formation time is excluded. For comparison, the time of the solution of the system with an assembled matrix is given. The time of copying the matrix A of the system (1) to the GPU memory is also presented. In addition, the time is given for solving the system with the use of both the algorithm with an assembled matrix and the matrix-free solution algorithm.

The CPU parallelization is carried out with OpenMP. The solution of the system of equations on several GPUs is carried out by CUDA in conjunction with

Table 2. The execution time of the interpolation for e^{-cr^2}, $c = 1.0$, s

		Number of equation			
		960	9600	19200	28800
Forming A	$1 \times$ CPU	0.098	5.655	33.91	60.85
	$8 \times$ CPU	0.018	0.907	6.166	12.16
	MFA	0.0	0.0	0.0	0.0
Copy of A to GPU	$1 \times$ GPU	0.009	0.312	0.842	—
	$2 \times$ GPU	0.006	0.125	0.433	1.393
	MFA	0.0	0.0	0.0	0.0
System solution $A\gamma = b$	$1 \times$ CPU	3.343	592.8	4732	10359
	$8 \times$ CPU	0.271	86.82	946.6	2273
	$1 \times$ GPU	0.288	2.282	12.11	—
	$2 \times$ GPU	1.262	2.354	8.643	16.19
	Matrix-free algorithm				
	$8 \times$ CPU	1.712	249.5	1489	4492
	$1 \times$ GPU	0.471	14.01	91.59	191.1
	$2 \times$ GPU	1.226	8.664	47.03	95.58
Total time	$1 \times$ CPU	3.438	598.5	4765	10419
	$8 \times$ CPU	0.288	86.82	952.7	2285
	$1 \times$ GPU	0.317	3.53	19.12	—
	$2 \times$ GPU	1.283	3.38	15.24	29.74
	Matrix-free algorithm				
	$8 \times$ CPU	1.712	249.6	1489	4492
	$1 \times$ GPU	0.472	14.01	91.59	191.1
	$2 \times$ GPU	1.226	8.664	47.03	95.58

OpenMP. The system of equations is solved by the conjugate gradient method with the diagonal preconditioner [5]. 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× quad-core Intel Xeon processor E5-2609, 2×GeForce GTX 980 with 4Gb GDDR.

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. In the last line of Table 2, the 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 $8 \times$ 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 $n_{\Gamma_\Omega} \times 3$. The required memory size for solving the system with the assembled matrix is $n_{\Gamma_\Omega} \times n_{\Gamma_\Omega}$. 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 $n_{\Gamma_\Omega}/3$. 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 $n_{\Gamma_\Omega} \times 3$ (the MFA) and six vectors $n_{\Gamma_\Omega} \times 1$. Thus, for solving the system using the MFA and double precision arithmetic, $n_{\Gamma_\Omega} \times (3+6) \times 8$ bytes are required. Consequently, the maximum size of a system for the GPU with a 4Gb 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 Conclusion

For solving the problems on unstructured meshes, choosing the data points affects the quality of interpolation. The obtained results show that the interpolation on the irregularly distributed data is of high quality and applicable for meshes of super-large dimensions. The variation of the constant c allows us to choosing the optimal ratio of the interpolation error and the time of its construction.

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 minimizing the cost of additional computations associated with the formation of the matrix elements.

References

1. Berndt, M., Breil, J., Galera, S., Kucharik, M., Maire, P.H., Shashkov, M.: Two-step hybrid conservative remapping for multimaterial arbitrary LagrangianEulerian methods. Journal of Computational Physics 230(17), 66646687 (2011)
2. Boer, A.D., der Shoot, M.V., Bijl, H.: Mesh deformation based on radial basis function interpolation. Computer and Structures 85, 784795 (2007)
3. De Boer, A., Van der Schoot, M.S., Bijl, H.: New method for mesh moving based on radial basis function interpolation. In: ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006. Delft University of Technology; European Community on Computational Methods in Applied Sciences (ECCOMAS) (2006)
4. Farrell, P., Piggott, M., Pain, C., Gorman, G., Wilson, C.: Conservative interpolation between unstructured meshes via supermesh construction. CMAME 198(33-36), 26322642 (2009)
5. Kopysov, S.P., Kuzmin, I.M., Nedozhogin, N., Novikov, A.K., Sagdeeva, Y.A.: Scalable hybrid implementation of the Schur complement method for multi-GPU systems. Journal of Supercomputing 69, 8188 (2014)
6. Novikov, A., Piminova, N., Kopysov, S., Sagdeeva, Y.: Layer-by-Layer Partitioning of Finite Element Meshes for Multicore Architectures. Communications in Computer and Information Science 687, 106117 (2016)
7. Wang, T.S., Zhao, X., Zhang, S.: Aeroelastic Modeling of a Nozzle Startup Transient. In: Journal of Propulsion and Power. vol. 30 (2013)