Advances in Engineering Software 86 (2015) 70–79

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Acceleration of boundary element method by explicit vectorization

Michal Merta [*], Jan Zapletal

IT4Innovations National Supercomputing Center, VŠB-TU Ostrava, 17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
Dept. of Applied Mathematics, VŠB-TU Ostrava, 17. listopadu 15/2172, 708 33 Ostrava, Czech Republic

ARTICLE INFO

Article history:
Received 23 February 2015
Received in revised form 7 April 2015
Accepted 11 April 2015

Keywords:
Boundary element method
Sound scattering
Helmholtz equation
Vectorization
SIMD
OpenMP parallelization

ABSTRACT

Although parallelization of computationally intensive algorithms has become a standard with the scientific community, the possibility of in-core vectorization is often overlooked. With the development of modern HPC architectures, however, neglecting such programming techniques may lead to inefficient code hardly utilizing the theoretical performance of nowadays CPUs. The presented paper reports on explicit vectorization for quadratures stemming from the Galerkin formulation of boundary integral equations in 3D. To deal with the singular integral kernels, two common approaches including the semi-analytic and fully numerical schemes are used. We exploit modern SIMD (Single Instruction Multiple Data) instruction sets to speed up the assembly of system matrices based on both of these regularization techniques. The efficiency of the code is further increased by standard shared-memory parallelization techniques and is demonstrated on a set of numerical experiments.

1. Introduction

The boundary element method (BEM) is a counterpart to the finite element method (FEM) suitable for the solution of partial differential equations which can be formulated in the form of boundary integral equations. Since BEM reduces the given problem to the boundary of a computational domain, it is especially suitable for problems stated in unbounded domains, such as acoustic or electromagnetic wave scattering, or shape optimization.

System matrices arising from the classical BEM are dense and the method has quadratic computational and memory complexity with respect to the number of surface elements. Moreover, special quadrature methods are needed due to the singularities in the kernels of the boundary integrals [1,2], which further contribute to computational demands of the method. Several fast BEM approaches can be employed to reduce computational and memory requirements to almost linear. The common methods are based on the decomposition of the surface mesh into clusters and subsequent low rank approximation of matrix blocks corresponding to admissible pairs of clusters. Nonadmissible blocks are assembled in the standard way as full rank matrices. The fast multipole method (FMM) is based on the approximation of the system matrices by the multipole series expansion [3–5], whereas the adaptive cross approximation (ACA) assembles the low rank approximation from an algebraic point of view [6,2].

Regardless the above mentioned approximation techniques, there is still a need for an efficient assembly of nonadmissible matrix blocks or a certain number of rows and columns of admissible blocks in the case of ACA. Since these blocks are usually too small to be distributed among computational nodes by MPI, an OpenMP parallelization of the assembly is an obvious choice. In this paper, we discuss further acceleration of the process by means of vectorization of the quadrature over pairs of surface elements.

With new SIMD instruction sets available in modern processors the usage of vectorization becomes more important in scientific computation. Neglecting it may lead to inefficient code not capable of reaching the theoretical performance of current CPUs. The SSE instruction set introduced by Intel in 1999 provided eight 128-bit registers and enabled concurrent operations on four 32-bit single-precision floating point numbers. Its successors, SSE2–SSE4, extended this capability to support SIMD operations on two 64-bit double-precision floating point operands while incrementally adding more instructions. The AVX instruction set supported by Intel processors since 2011 extends the registers length from 128 bits to 256 bits and introduces a three-operand SIMD instruction format. Its capabilities are further extended by AVX2. The AVX-512 should provide registers with 512-bit length allowing for concurrent operation on eight 64-bit double precision numbers. Its support is announced for Intel's Knights Landing processor available in 2015 and for Intel's Skylake microprocessor architecture [7].

[*] Corresponding author at: IT4Innovations National Supercomputing Center, VŠB-TU Ostrava, 17. listopadu 15/2172, 708 33 Ostrava, Czech Republic.
E-mail addresses: michal.merta@vsb.cz (M. Merta), jan.zapletal@vsb.cz (J. Zapletal).

http://dx.doi.org/10.1016/j.advengsoft.2015.04.008
0965-9978/© 2015 Elsevier Ltd. All rights reserved.

To use the vector instructions the existing scalar code usually has to be modified. While the automatic loop vectorization provided by the compiler is not capable of vectorizing more complex loops often occurring in scientific codes, exploiting the supported intrinsic functions may lead to a confusing and hardly maintainable code. One of the possibilities avoiding these issues is to use a higher level library, such as VML from Intel's Math Kernel Library [8], VDT [9], or the Vc library [10], which is the main focus of this paper. The library provides a high level wrapper on SIMD intrinsics and enables explicit vectorization of C++ code. It is portable among various compilers and SIMD instruction sets and enables easy vectorization without the need for a major redesign of the existing object oriented C++ code.

The topic of the vectorization of the BEM computation has been presented in several publications. In [11] an example of automatic loop vectorization of Fortran boundary element computation is provided. The original routines are manually altered using techniques such as loop unrolling and loop reordering in order to enable the compiler to employ SIMD instructions. Although a reasonable speedup with respect to the non-vectorized version is obtained, modifications lead to a significantly more complex code. The interested reader may also consult [12] for a comprehensive presentation of BEM quadrature vectorization. The author provides a general overview of the SIMD parallelism, compares two approaches to handling data during the computation (inter- and intra-register operations), and presents results of numerical experiments with a code vectorized using intrinsic functions. However, the work does not discuss the treatment of singularities in the related surface integrals, which is one of the crucial tasks of BEM computations.

The structure of the paper is as follows. In the next section we provide a model problem on which we demonstrate the boundary element workflow. In Section 3, a short overview of our BEM library is provided, Section 4 discusses the vectorization of the computationally most demanding parts of the code. Finally, we provide results of numerical experiments and conclude.

2. Boundary element method for sound-hard scattering

In this section we present the model problem under consideration, derive the corresponding boundary integral equations and their Galerkin discretization.

2.1. Helmholtz boundary integral equation

We consider a time-harmonic scattering problem with a sound-soft obstacle modelled by a bounded Lipschitz domain $\Omega \subset \mathbb{R}^3$. The incident wave is of the form $u_\mathrm{i} := e^{\mathrm{i}\kappa\langle \boldsymbol{x},\boldsymbol{d}\rangle}$ with the imaginary unit i, the wave number $\kappa \in \mathbb{R}_+$, and a unit direction vector $\boldsymbol{d}$. The wave u_s scattered from Ω satisfies the exterior Dirichlet boundary value problem for the Helmholtz equation [13,2]

$$\begin{cases} \Delta u_\mathrm{s} + \kappa^2 u_\mathrm{s} = 0 & \text{in } \Omega^\mathrm{ext} := \mathbb{R}^3 \setminus \overline{\Omega}, \\ u_\mathrm{s} = g & \text{on } \partial\Omega, \\ \left|\left\langle \nabla u_\mathrm{s}(\boldsymbol{x}), \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right\rangle - \mathrm{i}\kappa u_\mathrm{s}(\boldsymbol{x})\right| = \mathcal{O}\left(\frac{1}{\|\boldsymbol{x}\|^2}\right) & \text{for } \|\boldsymbol{x}\| \to \infty. \end{cases} \tag{1}$$

The total wave field around the obstacle is given by $u_\mathrm{t} := u_\mathrm{i} + u_\mathrm{s}$. The Dirichlet condition in (1) is given by the negative of the incident wave $g := -u_\mathrm{i}$. Thus, the total wave field u_t vanishes on $\partial\Omega$. The Sommerfeld radiation condition ensures uniqueness of the solution $u_\mathrm{s} \in H^1_{\Delta,\mathrm{loc}}(\Omega^\mathrm{ext})$, which is equivalent to

$$u_\mathrm{s}|_{\widetilde{\Omega}} \in H^1_\Delta(\widetilde{\Omega}) := \left\{ u \in L^2(\widetilde{\Omega}) : \frac{\partial u}{\partial x_i} \in L^2(\widetilde{\Omega}) \wedge \Delta u \in L^2(\widetilde{\Omega}) \right\},$$

for all bounded domains $\widetilde{\Omega} \subset \Omega^\mathrm{ext}$ with the derivatives understood in the distributional sense.

To solve the boundary value problem (1) we use the direct approach via the representation formula [13,14,2,1,15]

$$u_\mathrm{s} = W_\kappa \gamma^{0,\mathrm{ext}} u_\mathrm{s} - \widetilde{V}_\kappa \gamma^{1,\mathrm{ext}} u_\mathrm{s} \quad \text{in } \Omega^\mathrm{ext}, \tag{2}$$

with the trace operators

$$\gamma^{0,\mathrm{ext}} : H^1_{\Delta,\mathrm{loc}}(\Omega^\mathrm{ext}) \to H^{1/2}(\partial\Omega), \quad \gamma^{0,\mathrm{ext}} u = u|_{\partial\Omega} \text{ for } u \in C^\infty(\overline{\Omega^\mathrm{ext}}),$$
$$\gamma^{1,\mathrm{ext}} : H^1_{\Delta,\mathrm{loc}}(\Omega^\mathrm{ext}) \to H^{-1/2}(\partial\Omega), \quad \gamma^{1,\mathrm{ext}} u = \frac{\partial u}{\partial \boldsymbol{n}} \text{ for } u \in C^\infty(\overline{\Omega^\mathrm{ext}}),$$

$\boldsymbol{n}$ denoting the unit exterior normal vector to Ω, the single-layer and double-layer potentials

$$\widetilde{V}_\kappa : H^{-1/2}(\partial\Omega) \to H^1_{\Delta,\mathrm{loc}}(\Omega^\mathrm{ext}),$$
$$(\widetilde{V}_\kappa q)(\boldsymbol{x}) := \int_{\partial\Omega} v_\kappa(\boldsymbol{x},\boldsymbol{y}) q(\boldsymbol{y})\, ds_{\boldsymbol{y}}, \tag{3}$$

$$W_\kappa : H^{1/2}(\partial\Omega) \to H^1_{\Delta,\mathrm{loc}}(\Omega^\mathrm{ext}),$$
$$(W_\kappa t)(\boldsymbol{x}) := \int_{\partial\Omega} \frac{\partial v_\kappa}{\partial \boldsymbol{n}_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y}) t(\boldsymbol{y})\, ds_{\boldsymbol{y}}, \tag{4}$$

and the fundamental solution of the Helmholtz equation in 3D

$$v_\kappa(\boldsymbol{x},\boldsymbol{y}) := \frac{1}{4\pi} \frac{e^{\mathrm{i}\kappa\|\boldsymbol{x}-\boldsymbol{y}\|}}{\|\boldsymbol{x}-\boldsymbol{y}\|}.$$

To evaluate the solution u_s in $\boldsymbol{x} \in \Omega^\mathrm{ext}$ using the formula (2) it is necessary to complete the Cauchy data $\gamma^{0,\mathrm{ext}} u_\mathrm{s}$, $\gamma^{1,\mathrm{ext}} u_\mathrm{s}$. Since the Dirichlet trace $\gamma^{0,\mathrm{ext}} u_\mathrm{s}$ is given by $g = -\gamma^{0,\mathrm{ext}} u_\mathrm{i}$, only the missing Neumann trace $\gamma^{1,\mathrm{ext}} u_\mathrm{s}$ needs to be computed. Applying the Dirichlet trace operator $\gamma^{0,\mathrm{ext}}$ to the representation formula (2) and using the well-known properties of the potential operators (3) and (4) (see, e.g., [14,2,15,1]) we obtain the boundary integral equation

$$(V_\kappa \gamma^{1,\mathrm{ext}} u_\mathrm{s})(\boldsymbol{x}) = -\frac{1}{2} g(\boldsymbol{x}) + (K_\kappa g)(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \partial\Omega, \tag{5}$$

with the single-layer and double-layer boundary integral operators

$$V_\kappa : H^{-1/2}(\partial\Omega) \to H^{1/2}(\partial\Omega), \quad (V_\kappa q)(\boldsymbol{x}) := \int_{\partial\Omega} v_\kappa(\boldsymbol{x},\boldsymbol{y}) q(\boldsymbol{y})\, ds_{\boldsymbol{y}}, \tag{6}$$

$$K_\kappa : H^{1/2}(\partial\Omega) \to H^{1/2}(\partial\Omega), \quad (K_\kappa t)(\boldsymbol{x}) := \int_{\partial\Omega} \frac{\partial v_\kappa}{\partial \boldsymbol{n}_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y}) t(\boldsymbol{y})\, ds_{\boldsymbol{y}}. \tag{7}$$

Note that contrary to the potentials (3) and (4), the functions $V_\kappa q$, $K_\kappa t$ are only defined on $\partial\Omega$.

The Galerkin formulation equivalent to (5) reads

$$\langle V_\kappa \gamma^{1,\mathrm{ext}} u_\mathrm{s}, s \rangle_{\partial\Omega} = \left\langle \left(-\frac{1}{2} I + K_\kappa \right) g, s \right\rangle_{\partial\Omega} \quad \text{for all } s \in H^{-1/2}(\partial\Omega). \tag{8}$$

2.2. Boundary element method

To discretize the Galerkin formulation (8) we triangulate the surface $\partial\Omega$ into E flat shape-regular open triangles τ_i, i.e.,

$$\partial\Omega \approx \bigcup_{n=1}^{E} \overline{\tau_i}.$$

To approximate the Cauchy data we use piecewise linear ansatz for the Dirichlet data $\gamma^{0,\mathrm{ext}} u_\mathrm{s}$ and piecewise constant ansatz for the Neumann data $\gamma^{1,\mathrm{ext}} u_\mathrm{s}$

$$\gamma^{0,\mathrm{ext}} u_\mathrm{s} = g \approx t := \sum_{i=1}^{N} t_j \varphi_j, \quad \gamma^{1,\mathrm{ext}} u_\mathrm{s} \approx s := \sum_{i=1}^{E} s_i \psi_i,$$

where N denotes the total number of mesh nodes. Using piecewise constant testing functions ψ_ℓ this results in the discrete system of linear equations