Engineering Analysis with Boundary Elements 96 (2018) 194–208

Contents lists available at ScienceDirect

Engineering Analysis with Boundary Elements

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

Parallel and vectorized implementation of analytic evaluation of boundary integral operators

Jan Zapletal[a,b,*], Günther Of[c], Michal Merta[a,b]

[a] IT4Innovations, VŠB – Technical University of Ostrava 17. listopadu 2172/15, Ostrava-Poruba 708 33, Czech Republic
[b] Department of Applied Mathematics, VŠB – Technical University of Ostrava. 17. listopadu 2172/15, Ostrava-Poruba 708 33, Czech Republic
[c] Institute of Applied Mathematics, Graz University of Technology, Steyrergasse 30, Graz A-8010, Austria

ARTICLE INFO

MSC:
65N38
65Y05
68W10

Keywords:
Boundary element method
Quadrature
SIMD
Vectorization
Intel Xeon Phi
Many-core

ABSTRACT

In this paper, we describe an efficient analytic evaluation of boundary integral operators. Firstly, we concentrate on a novel approach based on the simultaneous evaluation of all three linear shape functions defined on a boundary triangle. This results in a speedup of 2.35–3.15 times compared to the old approach of separate evaluations. In the second part we comment on the OpenMP parallelized and vectorized implementation of the suggested formulae. The employed code optimizations include techniques such as data alignment and padding, array-of-structures to structure-of-arrays data transformation, or unit-strided memory accesses. The presented scalability results, with respect both to the number of threads employed and the width of the SIMD register obtained on an Intel® Xeon™ processor and two generations of the Intel® Xeon Phi™ family (co)processors, validate the performed optimizations and show that vectorization needs to be an inherent part of modern scientific codes.

1. Introduction

The computation of matrix entries and the evaluation of representation formula are of major importance in boundary element methods (BEM). On one hand, the often singular integrals have to be computed with sufficient accuracy to preserve important matrix properties and the optimal order of convergence. On the other hand, the computation has to be fast as this is a major part of the total computational time, even for fast boundary element methods. A popular approach is to use explicit analytical formulae for the evaluation of boundary integral operators. The related formulae have been topic of research for decades; recent publications discussing the topic include [1–10]. For approaches avoiding singular integrals see, e.g., [11–13]. In most cases, the formulae are provided for plane triangles and the kernel $|x-y|^{-1}$ and its derivatives. As the formulae are exact, they are obviously related. However, the knowledge of a formula is just part of the story as certain geometric settings lead to special cases in its evaluation which have to be handled with extra care in the implementation. Thus, a pure comparison of the formulae is not sufficient to rate the quality of the approaches.

In this paper, we use a carefully developed and extensively tested implementation based on the formulae in [5,10]. We focus on the evaluation of single- and double-layer potentials of the 3D Laplace kernel and linear shape functions. The formulae in [5,10] suggest choosing a local coordinate system in the plane triangle related to the considered linear shape function. Then, three independent computations are required to compute the integrals for the three linear shape functions of a single triangle. In this paper, we compute these three integrals at once, which reduces the computational effort to almost one third. To do so, we present additional analytic formulae which are related to the setting chosen in [5]. The formulae (2.13) and (2.14) for the double-layer potential with constant basis function were known but unpublished. The formulae (2.15) and (2.16) for the double-layer potential and some other linear basis function, as well as the corresponding formulae (2.26) and (2.28) for the single-layer potential are new in this setting. As we observed that all formulae of these three cases have major parts in common, we were able to elaborate the simultaneous computation of the potentials for all three linear shape functions of a triangle. These results are presented in Section 2.2.4 for the double-layer potential and in Section 2.3.4 for the single-layer potential. The results of Section 4.1 show good speedups for the related computational times, ranging from 2.35 to 3.15 with the new simultaneous computation.

The second part of the paper is devoted to the efficient implementation of the suggested evaluation routines for modern multi- and many-core (co)processors with wide SIMD registers. It has become more or less standard in scientific codes to utilize shared- and distributed-memory parallelism achieved by OpenMP and MPI, and thus to use the

* Corresponding author at: IT4Innovations, VŠB – Technical University of Ostrava, 17. listopadu 15/2172, 70833 Ostrava-Poruba, Czech Republic.
E-mail address: jan.zapletal@vsb.cz (J. Zapletal).

https://doi.org/10.1016/j.enganabound.2018.08.015
Received 24 May 2018; Received in revised form 22 August 2018; Accepted 29 August 2018
Available online 20 September 2018
0955-7997/

computational power of all available cores. However, in recent years the theoretical peak performance of CPUs has also been rising due to the capabilities of vector processing units able to perform simultaneous computations on vectors of data. This concept, known as Single Instruction Multiple Data (SIMD), is becoming increasingly important. Indeed, the newest AVX512 (Advanced Vector Extensions) instruction set is able to operate on 512 bits of floating-point data, which translates to 8 double-precision operands. Neglecting in-core vectorization can thus reduce the performance by a factor of 8 (or even 16 in single-precision arithmetic). Recently, several papers have been published dealing with many-core vectorized implementation of numerical methods, see [14,15] for GP-GPU accelerated numerical assembly of BEM matrices, [16] for 2D BEM for the Laplace equation, [17] for efficient quadrature routines in the context of the finite element method (FEM), [18,19] for stencil-based simulations of geophysical flows, [20–22] for the performance of CFD codes, and [23] for the acceleration of the finite element tearing and interconnecting (FETI) solver.

Vectorization can be achieved via different strategies. One option is the inline assembly code or compiler-specific intrinsic functions for compute-intensive kernels. Although these can achieve optimal speedup, the code is not portable between multiple architectures. A second option is to use wrapper libraries providing vector implementation of common mathematical functions in several vector instructions sets (including, e.g., SSE4.2, AVX2, or AVX512) resulting in a portable implementation. In [24] we describe the application of the Vc library [25] to both the semi-analytic and numerical BEM assembly. The VCL library [26] can be used in a similar fashion. In [27] we used OpenMP SIMD pragmas described by the OpenMP standard [28] for the vectorization of the regularized numerical assembly of BEM matrices. In contrast to [27], where we showed that the efficiency of this approach can get very close to the optimal values, in this paper we use OpenMP SIMD to accelerate the presented analytic evaluations. Although this approach is slightly less explicit than the methods mentioned above, the compiler is able to perform additional optimizations and can contribute to better performance.

This part of the paper is structured as follows; code optimizations employed for the efficient parallelization and vectorization of the semi-analytic assembly and the exact evaluation of the representation formula are presented in Section 3. In Sections 4.2 and 4.3 we provide results obtained on multi- and many-core architectures using the Intel Xeon and Xeon Phi (co)processors. The suggested rather simple threading approach leads to optimal speedup on all tested architectures, see Tables 4.2–4.4 for detailed results. For the performance of the vectorized code we refer to Tables 4.5–4.7, where one can see that changing the width of a SIMD vector processed simultaneously by vector processing units leads to significant speedups ranging from 4.95 to 7.75 for the matrix assembly and the evaluation of the representation formula, respectively. Taking into account all proposed techniques, i.e., the simultaneous evaluation for all shape functions and OpenMP threading and vectorization, the speedup with respect to the scalar sequential version reaches up to several hundred on Xeon and Xeon Phi.

As mentioned above, for simplicity we restrict our exposition to the Laplace kernel. However, the presented simultaneous evaluation of integral operators can be applied in the same manner for problems modelling wave scattering with the Helmholtz equation, see [5,10]. The boundary integrals are split into a singular part corresponding with the Laplace kernel, and a non-singular remainder that can be treated by a numerical scheme without further regularization. Moreover, the boundary element matrices related to the Lamé equation for linear elasticity problems can be built from sparse transformations of the Laplace matrices, see [5, Section C.2.3], for analytic evaluation in fracture propagation problems also consult [29]. Thus, the parallelization and vectorization techniques presented here can be applied in the same manner for a rather broad range of engineering problems.

Although we concentrate on the full assembly of BEM matrices in the presented paper, the developed techniques can easily be adapted for fast approaches. The adaptive cross approximation [5,30] is based on the full assembly of the so-called non-admissible blocks of the system matrix and low rank approximation of the far-field. The low rank approximation is built by assembling an appropriate subset of rows and columns building these blocks, i.e., the assembly routines are used in a very similar fashion. Numerical experiments comparing the parallelization and vectorization of full and sparsified BEM matrices have been presented in [27]. Similarly, the techniques presented here in the context of Galerkin approximation can be applied to collocation schemes without significant changes. Indeed, the numerical integration for the outer surface integral can be seen as an evaluation of the boundary integral operators in collocation points – here serving as quadrature points. In the paper we also concentrate on intra-node optimization in shared memory. For large scale experiments the distributed level of parallelism (achieved by MPI) has to be added to the method. This can be done, e.g., with the boundary element tearing and interconnecting (BETI) domain decomposition technique [31] in connection with the ESPRESO library [32] developed at IT4Innovations, or with the distributed version of the adaptive cross approximation (ACA) method [33,34].

2. Analytic evaluation of singular integrals

In the following we consider the Dirichlet boundary value problem for the Laplace equation in three spatial dimensions. We discuss analytical formulae to compute the single- and double-layer potentials for plane triangles and linear shape functions. In particular, we present some analytical formulae which are new in the setting of [5, Section C.2]. The presented complete set of formulae allows the simultaneous computation of the integral operators for the three linear shape functions of a triangle at once, thus reducing the computational times significantly.

2.1. Model problem

In particular, we solve

$$-\Delta u = 0 \text{ in } \Omega, \quad u = g \text{ on } \partial\Omega \tag{2.1}$$

where $\Omega \subset \mathbb{R}^3$ denotes a bounded Lipschitz domain and $g \in H^{1/2}(\partial\Omega)$ is the given Dirichlet datum. An explicit formula for the solution to (2.1) is given by, see, e.g., [35],

$$u(\widetilde{x}) = \int_{\partial\Omega} v(\widetilde{x}, y) w(y)\, \mathrm{d}s_y - \int_{\partial\Omega} \frac{\partial}{\partial n_y} v(\widetilde{x}, y) g(y)\, \mathrm{d}s_y \quad \text{for } \widetilde{x} \in \Omega \tag{2.2}$$

with $w := \partial u/\partial n$ and $v : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3$ denoting the fundamental solution to the Laplace equation in 3D, i.e.,

$$v(x, y) := \frac{1}{4\pi} \frac{1}{|x - y|}.$$

The unknown Neumann datum $w \in H^{-1/2}(\partial\Omega)$ can be determined by solving the weakly singular boundary integral equation obtained from (2.2) by taking the limit $\Omega \ni \widetilde{x} \to x \in \partial\Omega$,

$$Vw(x) = \frac{1}{2} g(x) + Kg(x) \quad \text{for almost all } x \in \partial\Omega \tag{2.3}$$

with the single- and double-layer boundary integral operators

$$V : H^{-1/2}(\partial\Omega) \to H^{1/2}(\partial\Omega), \; Vw(x) := \int_{\partial\Omega} v(x, y) w(y)\, \mathrm{d}s_y,$$

$$K : H^{1/2}(\partial\Omega) \to H^{1/2}(\partial\Omega), \; Kg(x) := \int_{\partial\Omega} \frac{\partial}{\partial n_y} v(x, y) g(y)\, \mathrm{d}s_y,$$

respectively. Both boundary integral operators are linear and bounded, and the $H^{-1/2}(\partial\Omega)$-ellipticity of V ensures unique solvability of (2.3), see, e.g., [35]. The variational formulation equivalent to the equation (2.3) used for the discretization by the boundary element method reads

$$\langle Vw, t \rangle_{\partial\Omega} = \left\langle \left(\frac{1}{2} I + K \right) g, t \right\rangle_{\partial\Omega} \quad \text{for all } t \in H^{-1/2}(\partial\Omega) \tag{2.4}$$