Formulating various BEM problems with an open source C++ BEM template library

Péter Fiala[1], Péter Rucz[1]

[1] *Laboratory of Acoustics and Studio Technologies, 1117 Budapest, Hungary, Email: fiala@last.hit.bme.hu*

Introduction

The paper introduces `NiHu`, an open source C++ template library for boundary elements [1]. The library is designed for the efficient discretisation of weighted residuals of boundary integral operators:

$$W_{ij} = \left\langle \varphi_i, (\mathcal{K} d_j)_\Gamma \right\rangle_F = \int_F \varphi_i(x) \int_\Gamma K(x,y) d_j(y) \, \mathrm{d}y \, \mathrm{d}x \qquad (1)$$

where $\mathcal{K}$ denotes a boundary integral operator defined with the kernel function $K(x, y)$, acting on the function space d_j over the boundary Γ, and φ_i denotes a discretised test function space defined over the test domain F. Weighted residual forms of this kind appear in several engineering applications related to the numerical solution of partial differential equations [2].

The general form (1) provides a high level of flexibility and generality in the definition of different BEM-related problems: For the particular case of steady state acoustics, the kernel function is the fundamental solution of the Helmholtz equation. The function space $d_j(y)$ describes a piecewise constant or isoparametric function space over the mesh of the problem's geometry. The selection of the test function space φ_i can define different solution formalisms. The case $\varphi_i = d_i$ defines a Galerkin BEM. Collocation is defined by $\varphi_i = \delta(x - x_i)$ where x_i denotes the nodal locations of d_i.

Software concepts

The toolbox consists of (1) a C++ template core that defines the process of matrix generation in a generic way in terms of component implementations, (2) a Component library that defines specialised components for specific aspects of BEM problems (e.g. element types, kernels for acoustic applications, or regularisation methods), and (3) a Matlab shell layer that provides pre- and post-processing functionalities and straightforward coupling with other software tools.

The two main configurations of `NiHu` are displayed in Figure 1. `NiHu` is responsible for the weighted residual matrix assembly, and is designed as a matrix-to-matrix application, where the inputs are mesh description matrices and the outputs are the weighted residual matrices. Linking with external mesh generating libraries and solvers can result in a standalone C++ BEM software. In particular, the built-in Matlab interface and the MEX compiler provides the means to compile the library directly under Matlab. In this case, `NiHu` works directly on Matlab's internally allocated matrices.

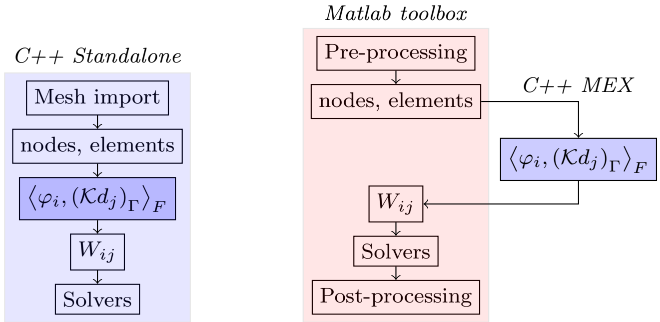

Figure 1: The two main configurations of `NiHu`: Serving as a standalone C++ application with external open source libraries or compiled under Matlab through the MEX interface

The 7-line example C++ code segment below demonstrates how the toolbox is used to discretise the double layer potential operator of the 3D Helmholtz equation into a BEM system matrix in a Galerkin context:

```
1 auto mesh = import_off_mesh("mymesh.off",
      quad_1_tag(), tria_1_tag());
2 auto &d = constant_view(mesh);
3 auto &phi = d; // Galerkin
4 unsigned N = mesh.get_num_dofs();
5 auto K = create_integral_operator(
      helmholtz_3D_DLP_kernel(1.));
6 MatrixXd W(N, N);
7 W << phi * K[d];
```

A collocational BEM is obtained by modifying line 3 to

```
4 auto &phi = dirac(d); // Collocation
```

`NiHu` provides a high level of flexibility and abstraction in the C++ source code. Efficiency is maintained by massively exploiting template metaprogramming features of the latest C++11 standard. The toolbox provides highly optimised low level mesh handling and numerical integration routines, templated support for heterogeneous problems, and optimised kernel evaluation for parallel kernel integrations.

Numerical examples

Although the main software feature is generality and flexibility, the first numerical example demonstrates a large-scale application of industrial size. The presented problem is an external 3D steady state acoustic scattering problem, where an acoustically rigid standing person mesh is illuminated by an acoustic point source, and the reflected wave field is computed. The applied formalism is that of Burton and Miller, where the single layer, double layer potentials and their hypersingular derivatives are discretised in order to assemble the BEM system.

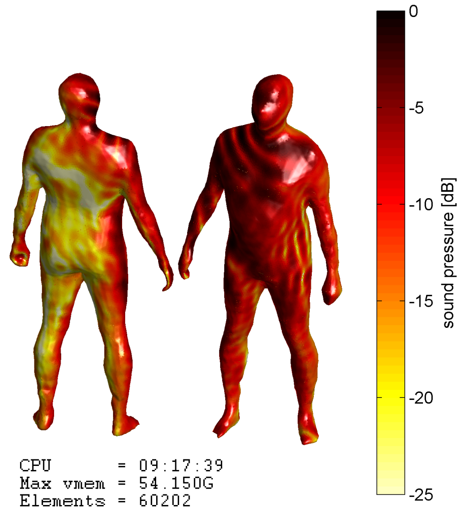

Figure 2: Pressure field around an acoustically rigid surface illuminated by a point source

Strongly and hypersingular boundary integrals are handled using Guiggiani's singularity subtraction method [3], implemented in a generic way in `NiHu`. The mesh consists of 60,202 elements, and is valid up to a Helmholtz number $\gamma = 120$. The applied solution formalism is collocation with piecewise constant function spaces. Computations were performed using a standalone C++ `NiHu` executable, utilising the iterative BiCGSTAB solver of the numerical linear algebra `Eigen`. The executable was compiled on a supercomputer providing sufficient RAM for storing the system matrices, and system matrix generation was performed using a single core of the machine. The resulting pressure field around the mesh is displayed in Figure 2. This example also demonstrates the portability of the library.

In its present form, the Component library includes the boundary operator kernels of steady state acoustics, as well as numerous recently published regularisation formalisms for singular integrals related to specific discretisation options [4, 5]. However, support for vector valued PDE is a built-in feature of the implementation, and `NiHu`'s Component library is designed to be easily customisable by researchers and developers.

This paper presents an example where Kelvin's displacement and traction fundamental solutions are incorporated into the `NiHu` framework, in order to provide a boundary element code for linear isotropic elastostatics. The new family of PDE can be incorporated into the library by (1) defining function objects computing the kernel expressions, (2) providing meta-information on the kernel's behaviour, like asymptotic behaviour, singular behaviour, symmetry, etc., and (3) defining function objects computing asymptotic expressions of the strongly and hypersingular kernel parts. The provided meta-data are processed during compilation by template metaprograms. The output of metaprograms are C++ classes optimised for the specific application case.

Figure 3 plots the displacement field of an elastic bar

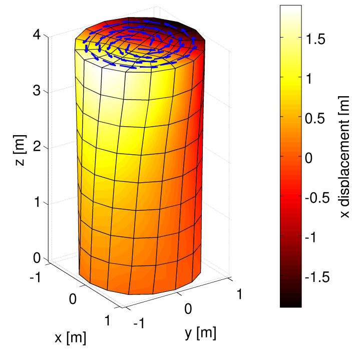

Figure 3: Demonstrative example for a vector valued boundary value problem, related to elastostatics

characterised by its Poisson's ratio, excited by tangential traction vectors on the top, zero traction vectors on the side and clamped at the bottom. The mesh is discretised using 9-noded quadratic quadrilateral elements, and the solution formalism is collocation with piecewise constant function spaces. Note that the implementation integrates the strongly singular traction kernel over curved elements in a generic way.

Conclusions

A newly developed generic BEM library, called `NiHu` has been introduced. The library is capable of discretising boundary integral operators for a large variety of BEM formalisms in a generic way, by reflecting abstract mathematics in the source and generating efficient code. The toolbox—accompanied by references, tutorials and supporting materials—can be downloaded from `http://last.hit.bme.hu/nihu`

References

[1] P. Fiala and P. Rucz. NiHu: an open source C++ BEM library. *Advances in Engineering Software*, 2014. submitted for publication.

[2] L. Gaul, M. Kögl, and M. Wagner. *Boundary element methods for engineers and scientists*. Springer, Stuttgart, Germany, 2003.

[3] Junjie Rong, Lihua Wen, and Jinyou Xiao. Efficiency improvement of the polar coordinate transformation for evaluating BEM singular integrals on curved elements. *Engineering Analysis with Boundary Elements*, 38:83–93, 2014.

[4] T. Matsumoto, C. Zheng, S. Harada, and T. Takahashi. Explicit evaluation of hypersingular boundary integral equation for 3-D Helmholtz equation discretized with constant triangular element. *Journal of Computational Science and Technology*, 4(3):194–206, 2010.

[5] M. Lenoir and N. Salles. Exact evaluation of singular and near-singular integrals in Galerkin BEM. In *Proceedings of ECCOMAS2012*, pages 1–20, Vienna, Austria, September 2012.