**Vibration soil isolation using a 3-D frequency domain BEM: GPGPU implementation.**

Alfredo Goldman1, Marco Dimas Gubitoso1, [Giuliano Augusto Faulin Belinassi](https://plus.google.com/u/0/114604157258770121122?prsrc=4)1

[gold@ime.usp.br](mailto:gold@ime.usp.br), [gubi@ime.usp.br](mailto:gubi@ime.usp.br), [giuliano.belinassi@usp.br](mailto:giuliano.belinassi@usp.br)

Ronaldo Carrion2

[rcarrion@usp.br](mailto:rcarrion@usp.br)

1 University of Sao Paulo – Institute of Mathematics and Statistics (IME/USP)

Rua do Matão, 1010, Butantã. CEP 05508-090, São Paulo – SP, Brasil

2 University of Sao Paulo – Polytechnic School (EPUSP)

Av. Prof. Mello Moraes, 2373, Butantã. CEP 05508-030, São Paulo – SP, Brasil

**Abstract.** This paper presents a Boundary Element formulation for dynamic response analysis … last text to write

**Keywords:** Boundary Element Method, GPGPU, High Performance Computing.

1. **INTRODUCTION**

The idea of mitigating vibrations transmitted by the ground in surrounding area of machine tools, trains or cars traffic and even mine blasting has long been studied. Vibration isolation may be accomplished by barriers which diffract the surface waves radiated from the vibration source and somehow reduce their amplitude. Among other possibilities, these barriers may be obtained by trenches (open or filled), subject addressed in this article.

As attested by (Beskos, Dasgupta, and Vardoulakis 1986), the analytical treatment of vibration isolation by wave barriers in the framework of linear elastodynamic is based on the theory of wave diffraction. (Barkan 1962) was the first to report some investigations for studying the effectiveness of wave barriers, however it was apparent that the analytical treatment of 3D elastic wave diffraction problems was confined to very simple geometries and idealized conditions and that realistic problems involving complex geometries, such as vibration isolation, could only be solved numerically.

This article reports a numerical investigation of vibration isolation from grounded transmitted waves generated by a soil load. The numerical code is written in Computer Unified Device Architecture (CUDA), for NVidia Graphics Processing Units (GPUs). The proposed methodology employs the frequency domain Direct version of the Boundary Element Method (DBEM) for the solution of vibration isolation problems by trenches in a 3D context and can deal with trenches of any arbitrary shape, but it is applied here to rectangular ones.

A comparative study for the case of vibration isolation by an open trench was presented by (Klein, Antes, and Le Houédec 1997) where a numerical simulation based on 3D DBEM was compared to an analytical solution and posteriorly to measured data constructed for full-scale testing and an important conclusion was the trench depth normalized with respect to the Rayleigh wave length.

Analyses developed by (Andersen and Nielsen 2005) on the study about reduction of ground vibration by means of barriers or soil improvement along a railway track indicated that open trenches are more efficient than infilled trenches or soil stiffening even at low frequencies noticing that the direction of the load is of primordial importance.. They coupled, in the frequency domain, a finite element-boundary element model of the track and subsoil, adopting a formulation in the moving frame of reference following the vehicle.

From the point of view of ground vibrations induced by the machine tools operation, (Alzawi and Hesham El Naggar 2011) proposed an innovative approach to construct GeoFoam trench as a wave barrier. Authors conducted A full-scale field experimental study to investigate the protective performance of both open and in-filled trench as well as to examine the influences of wall geometry and location from the vibratory source on the isolation efficiency. It was shown that both open and GeoFoam barriers can effectively reduce the transmitted waves.

Inserting a ground model as a fully saturated poroelastic half-space governed by Biot’s dynamic poroelastic theory, (Cao et al. 2012) proposed a semi-analytical study to investigate the screening efficiency of trenches to moving-load induced ground vibrations. Comparing two models of half-space, saturated poroelastic and a single-phase elastic, to the moving load-induced ground vibration, it was found that the discrepancy of the screening efficiencies between both models becomes significant when the load speed approaches to the Rayleigh wave speed of the ground surface.

Considering several fixed observation points, (Zoccali, Cantisani, and Loprencipe 2015) studied the mutual influence between trenches length and in-filled material type to mitigate ground-vibrations induced by trains. The analysis was performed using a finite element model, both in time and frequency domain, which was calibrated through comparison with in-situ measurements.

In view of Graphics Processing Units (GPUs), (Labaki, Ferreira, and Mesquita 2011) implemented the Direct version of the Boundary Element Method (DBEM) on a complementary GPU-CPU system adopting constant elements for the solution of 2D potential problems in which the efficiency of the implemented strategies was investigated. In this paper three important steps of BEM were evaluated, and it was observed that the point from which the GPU outperforms the CPU is function of the arithmetic intensity of each problem. In all three cases, however, the graphics hardware has shown to be more numerically efficient than the CPU as increasing number of elements and internal points.

Combining BEM with the fast multipole method (FMM), the fast multipole BEM (FMBEM), (Wang et al. 2013) proposed an adaptive scheme to solve large scale problems. Since Multipole Expansions (ME), Moment- to-Local (M2L) translation, Local Expansions (LE), and the Near Field Direct Computation (NFDC) are level independent, they are suitable for parallel computing. Authors showed that in some examples they reached speedup of 16.2. Based on the idea of rigid body motion method (RBMM) for the FMBEM, (Wang et al. 2015), once more achieved reasonable speedup for 3D elasticity problems.

On the analysis of GPU´s performance, (Carrion, Mesquita, and Ansoni 2015) developed a strategy to evaluate the dynamic response of frame-foundation-soil system. In this article, authors presented a DBEM to synthesize the 3D dynamic compliance matrix of a rigid and massless foundation interacting with unbounded soil profiles. The foundation compliance matrix was coupled to a frame, modeled by the Finite Element Method (FEM) leading to the dynamic response of a coupled frame-foundation-soil system. The performance analysis was done by comparing two codes: one was exclusively developed in C language and the other one in CUDA C (Compute Unified Device Architecture). The level of speedup was achieved by implementing some functions in the frame model, i.e., on the FEM based code.

Developing a simple strategy, (Vater, Betcke, and Dilba 2017) demonstrated how GPU-accelerated BEM routines can be used to accelerate fast boundary element formulations based on Hierarchical Matrices (H -Matrices) with ACA (Adaptive Cross Approximation). Have computed the scattered high-frequency sound field of a submarine to demonstrate the increase in overall application performance from moving to a GPU-based ACA assembly, authors based their methodology on offloading the CPU assembly of elements during the ACA assembly onto a GPU device and to use threading strategies across ACA blocks to create sufficient workload for the GPU. The proposed GPU strategy was designed such that it can be implemented in existing code with minimal changes to the surrounding application structure.

1. **Boundary Element formulation**

The frequency domain equations of motion for a 3D linear elastic, homogeneous and isotropic continuum body, known as Navier equations, can be expressed in terms of the displacement components as:

(1)

where *λ* and *μ* are Lamè constants, *ρ* is the mass density, *ω* is the circular frequency, are the components of body force and ; the point *x* has Cartesian coordinates *x*, *y* and *z*.

The auxiliary state used in the BE standard formulation is assumed to be the Full-Space Green’s Functions or Fundamental Solution with displacement and traction kernels given by and respectively. Thus, the differential Eq. (1) can be transformed into the integral Eq. (2) (Eringen and Suhubi 1975).

(2)

Equation (2) considers zero body forces; and represent displacements and tractions respectively of problem being solved. The collocation point is *ξ* and the field point is *x*. The elements of tensor are called integration free terms.

The solution of Eq. (2) is accomplished numerically. For this purpose, the boundary *S* of the body is discretized into a series of boundary elements over which displacements and tractions are assumed to be constant. Thus, a system of linear algebraic equations is obtained and can be written in matrix form as:

(3)

where and are square influence matrices consisting of elemental surface integrals with integrands the tensors and respectively.

After the boundary conditions are applied, the system can be solved to obtain all the unknown boundary values and consequently an approximate solution to the boundary value problem is obtained. Once the solutions at the boundary are obtained, Eq. (2) can be used to find interior displacements. The interior stresses can be obtained as in (Brebbia and Dominguez, 1989):

(4)

where the kernels and are listed in (Gaul and Fiedler 1993).

1. **Numerical Implemetation**

When , these integrals are regulars and integration is accomplished by using standard Gauss Quadrature. However, when , these integrals become singular due to the and singularities of the tensors and respectively and their integration are discussed below.

For kernel, which presents weak singularity, a particular treatment is applied to become possible the use of standard Gauss Quadrature (Dominguez 1993). The idea is to divide the quadrilateral element in triangular sub-elements. Then, each triangular sub-element is treated as a quadrilateral one with two corners collapsed at the collocation point. With this methodology the Jacobian of the transformation presents the order at the collocation point. This fact cancels the singularity at the mentioned point, which has the order . This approach has been introduced by (Lachat 1975) and has been further developed by (Telles 1987).

About kernel, an alternative to evaluate its integration is based on the rigid body motion. Nevertheless, this argument is only applicable to static problems and bounded domains. Below are described the way to apply this concept in dynamic problems and unbounded domains.

* 1. **Regularization of the Dynamic Kernel**

As the dynamic and static kernels present the same order of singularity, it is possible to regularize the singular integral by subtracting and adding the static kernel from the dynamic one. This strategy can be represented as follow.

Assuming the convention for the indexes:

*sta* = static problem

*dyn* = dynamic problem

Subtracting and adding the 3D static kernel to Eq. (2) results:

(5)

Integrals in Eq. (5) containing the difference between the dynamic and static kernels are no longer singular and can be evaluated by standard Gauss Quadrature. The integration of the static singular kernel is evaluated based on the rigid body motion concept.

* 1. **Treatment of Unbounded Domains**

Now this rigid body argument is extended to deal with unbounded half-space problems with the so-called Enclosing Elements. As the half-space discretization surface must be truncated at some point, a bounded body is created by adding a fictitious enclosing boundary which is discretized by the enclosing elements. Thus, it is possible to use the rigid body motion and determine all singular elements of the original half-space.

This idea has been used by (Ahmad and Banerjee 1988) and has been extended by (Araújo, Nishikava and Mansur 1997). (Carrion 2002) has also studied the enclosing elements in his Ph.D. thesis.

* 1. **GPGPU**

We used CUDA in order to access resources provided by NVidia GPUs with the objective to decrease the simulation time. CUDA is an extension to the C++ programming language that allows General Purpose computations in NVidia GPUs. CUDA uses the concept of kernels, which are functions to be called from the host (CPU) to be executed by the device (GPU).

CUDA kernels are organized into a set of blocks composed by a set of threads that cooperates with each other (Patterson and Hennessy 2007). CUDA memory hierarchy also reflects this structure, where Global memory is accessible by all threads; Local memory is private to a thread; and Shared memory is accessible by all threads in the same block (Patterson and Hennessy 2007).

Regarding the execution of threads within a block, once a block is allocated to a Streaming Multiprocessor, its threads are divided into sets of 32 units, called warps (Kirk and Wen-Mei 2016). So if a CUDA kernel is launched with 64 threads per block, threads [0-31] will be assigned to warp 0 and threads [32-63] will be assigned to warp 1. Inside a warp, a unique number between [0-31] is designed for each thread. Such number is called *lane*. Threads in the same warp are executed in the Single Instruction Multiple Data scheme, implying that if its control-flow diverges due an if-else statement, all threads within a warp will execute the if statement or all will execute the else statement. This behavior is not present when the divergence is between threads of distinct warps. (Kirk and Wen-Mei 2016)

We present the details about our implementation as pseudocode. Some parameters are shared between all algorithms and these are declared bellow: is the number of boundary elements; is the number of mesh elements; is the number of Gauss Quadrature point; **Thrust::reduce** is a function from Thrust library that reduces a vector with the respect to the sum; and **shfl\_down** is a instruction present since Kepler’s NVidia architecture that allows reduction of a variable among threads that are within same warp.

* + 1. **Assembly**

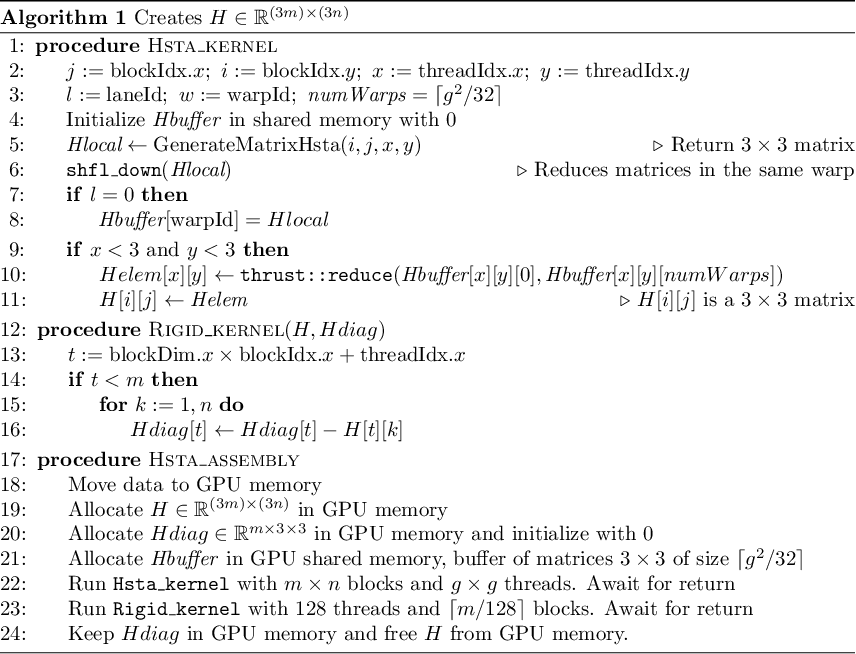
is assembled by computing the integral described in equation (5) by deploying the Gaussian Quadrature, and later by considering the rigid body motion. Algorithm 1 shows a pseudocode for creation in CUDA, where: (1) **HSTA\_KERNEL** constructs before the rigid body calculations; and (2) **RIGID\_KERNEL** is a CUDA kernel responsible for evaluating the rigid body movement.

Regarding **HSTA\_KERNEL**, **GenerateMatrixHsta** illustrates the evaluation of a single Gaussian point of , returning a single matrix. For completing the Gaussian Quadrature, we must sum all Gaussian Points associated with .

We now explain the details about our implementation of this CUDA kernel. Although a single call to **GenerateMatrixHsta** is , it does a considerable amount of floating point operations, justifying parallel calls to this procedure. Since it can be computed without control-flow divergence, we assigned each call to a CUDA thread. Then we assigned a block to compute all calls to **GenerateMatrixHsta** associated with , allowing us to use the low latency CUDA shared memory for the reduction required in Gaussian Quadrature.

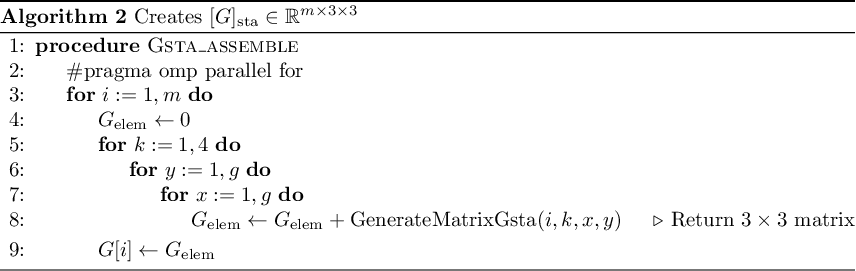
In order to complete the Gaussian Quadrature, we must sum all matrices that were returned by **GenerateMatrixHsta**. In total, we have matrices. We divided this step into two parts: (1) For all matrices that are within the same warp, we used the Kepler's **\_\_shfl\_down** intrinsic to reduce all these matrices into a single one, then we wrote that matrix into *HBuffer*, a buffer of matrices allocated in shared memory, to reduce the matrices between the threads that aren’t in the same warp. This resulted in a buffer of a size equal to the number of warps. Then we used 9 of these threads to reduce this buffer into a single matrix because one thread can be assigned to each entry. Finally, we wrote this matrix to its corresponding entry of . Launching the described kernel with blocks will generate the entire.

For the rigid body movement routine, we had to implement our own reduction because how the data is structured. This routine did not require many improvements because it consumed little time when compared with the assembly. We used 128 CUDA threads because it yielded best results. To minimize memory usage, we stored the reduced form of to a variable named *Hdiag.*



* + 1. **Assembly**

is assembled by computing the integrals described in equation (5), as illustrated by Algorithm 2. Since has no dependency relation with , and its creation complexity is , which is significantly lower than from creation, we opted to compute entirely in the CPU in parallel to creation in the GPU. For better usage of all cores within the CPU, we used OpenMP *parallel for* clause to compute each iteration of the loop at line 3 of Algorithm 2. We later send the result to the GPU.

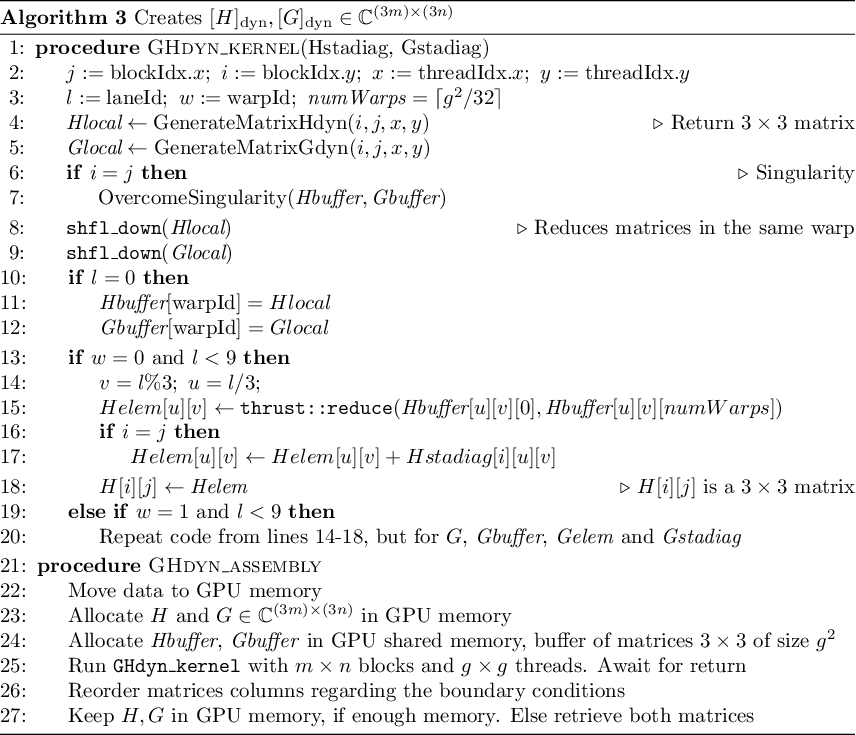


* + 1. **and Assembly**

and creations are one of the most floating point intensive parts of the software, requiring computations of several functions in the complex field. Some of these computations are shared between both these matrices, leading us to develop a single CUDA kernel to assemble both of them.

As for implementation as a CUDA kernel, we follow the same approach described in assembly. The differences are: We assigned one thread to compute one call to both **GenerateMatrixHdyn** and **GenerateMatrixGdyn**, and we made sure that reducing the buffer designed for the *H* matrices were done in parallel than the reduction of the *G* matrices.

Although the best performance can be archived by avoiding memory transferences across GPU-CPU, for a very large grid it is not possible to maintain both and in the Device simply because it does not have enough memory. In order to run such large grids, we implemented a memory swap logic that transfers the memory back to the CPU when there isn't sufficient memory, freeing GPU memory to proceed with the computation.



* + 1. **Dense Linear System solving**

Next step is solving the system . For multiplying , we used cuBLAS routines when fits entirely in the GPU, otherwise we used the OpenBLAS library, computing the product in the CPU. For indeed solving the system of equation, we used libMAGMA’s LU decomposition with partial pivoting when fits in GPU memory. This library implements such algorithm using GPU acceleration. When there isn’t enough GPU memory, we fall back to OpenBLAS LU’s decomposition.

* + 1. **Internal Points**

We did not implement in the GPU because our tests set the number of internal points to a margin that it would not worth the communication between the Host and Device, thus we used OpenMP directives for a better usage of multicore processors.

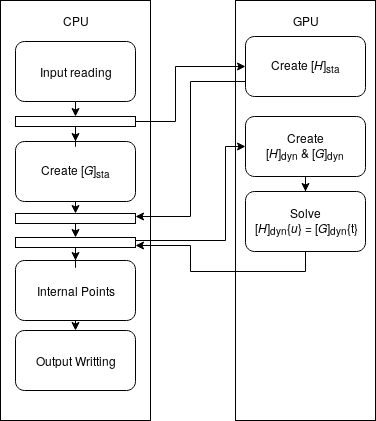


Figura : Code flow between CPU-GPU

Giuliano, please, write about GPGPU implementation

Code flowchart

|  |
| --- |
|  |

Input data reading Static purposes: function INPUTECE

[H]sta assembly: functions GHMATECE, NONSINGE, SINGGE and SOLFUNE

Input data reading Dynamic purposes: function INPUTECD

[H]dyn and [G]dyn assembly: functions GHMATECD, NONSINGD, SING\_DE, SOLFUND and SOLFUNDIF

System of equation solver: function (from LAPACK package)

Internal points: function INTEREC

Output data writing: function SIGMAEC

1. **Numerical Examples**
2. **Conclusion**

**Acknowledgments:**

Alfredo, if you want, write the agency which funded Giuliano´s scholarship.

1. **References**

Ahmad, S. and Banerjee, P. K. 1988, “Multi-Domain BEM for Two-Dimensional Problems of Elastodynamic”, *International Journal for Numerical Methods in Engineering*, Vol. 26, pp. 891-911.

Araújo, F. C., Nishikava, L., K. and Mansur, W. J. 1997. “On the Consideration of Enclosing Elements in 3D Elastodynamic Analysis with the BEM and BE/FE Coupled Process”, *XVIII CILAMCE*, Vol. I, pp. 423-431.

Barkan, D. D. 1962, “Dynamics of Bases and Foundations”, New York, McGraw-Hill.

Brebbia, C. A. and Dominguez, J. 1989. “Boundary Elements – An Introductory Course”, Computational Mechanics Publication.

Carrion, R. 2002. “Uma Implementação do Método dos Elementos de Contorno para Problemas Viscoelastodinâmicos Estacionários Tridimensionais em Domínios Abertos e Fechados”, Ph.D. Thesis, UNICAMP, Brasil (in portuguese).

Dominguez, J. 1993. “Boundary Elements in Dynamics”, Computational Mechanics Publications

Eringen, A. C. and Suhubi, E. S. 1975. “Elastodynamics - Linear Theory”, Vol. 2, New York, Academic Press.

Kirk, David B., and W. Hwu Wen-Mei. 2016. “Programming massively parallel processors: a hands-on approach”. Morgan kaufmann.

Patterson, D. A. and Hennessy, J. L. 2007. “Computer organization and design”. Morgan Kaufmann.

Telles, J. C. F. 1987. “A Self Adaptative Coordinate Transformation for Efficient Numerical Evaluation of General Boundary Element Integral”, *International Journal for Numerical Methods in Engineering*, Vol. 24:. 959-973.

Alzawi, Ashref, and M. Hesham El Naggar. 2011. “Full Scale Experimental Study on Vibration Scattering Using Open and in-Filled (GeoFoam) Wave Barriers.” *Soil Dynamics and Earthquake Engineering* 31(3): 306–17. http://dx.doi.org/10.1016/j.soildyn.2010.08.010.

Andersen, L., and S.R.K. Nielsen. 2005. “Reduction of Ground Vibration by Means of Barriers or Soil Improvement along a Railway Track.” *Soil Dynamics and Earthquake Engineering* 25(7–10): 701–16. http://linkinghub.elsevier.com/retrieve/pii/S0267726105000576.

Beskos, D. E., B. Dasgupta, and I. G. Vardoulakis. 1986. “Vibration Isolation Using Open or Filled Trenches.” *Computational Mechanics* 1(1): 43–63. http://link.springer.com/10.1007/BF00298637.

Cao, Zhigang, Yuanqiang Cai, Anders Boström, and Jianguo Zheng. 2012. “Semi-Analytical Analysis of the Isolation to Moving-Load Induced Ground Vibrations by Trenches on a Poroelastic Half-Space.” *Journal of Sound and Vibration* 331(4): 947–61. http://linkinghub.elsevier.com/retrieve/pii/S0022460X11007498.

Carrion, Ronaldo, Euclides Mesquita, and Jonas Laerte Ansoni. 2015. “Dynamic Response of a Frame-Foundation-Soil System: A Coupled BEM–FEM Procedure and a GPU Implementation.” *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 37(4): 1055–63. http://link.springer.com/10.1007/s40430-014-0230-3.

Gaul, L., and C. Fiedler. 1993. “Improved Calculation of Field Variables in the Domain Based on BEM.” *Engineering Analysis with Boundary Elements* 11(3): 257–64. http://linkinghub.elsevier.com/retrieve/pii/095579979390027I.

Klein, R., H. Antes, and D. Le Houédec. 1997. “Efficient 3D Modelling of Vibration Isolation by Open Trenches.” *Computers & Structures* 64(1–4): 809–17. http://linkinghub.elsevier.com/retrieve/pii/S004579499600418X.

Labaki, Josué, Luiz Otávio Saraiva Ferreira, and Euclides Mesquita. 2011. “Constant Boundary Elements on Graphics Hardware: A GPU-CPU Complementary Implementation.” *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 33(4): 475–82. http://www.scielo.br/scielo.php?script=sci\_arttext&pid=S1678-58782011000400011&lng=en&nrm=iso&tlng=en.

Vater, Kerstin, Timo Betcke, and Boris Dilba. 2017. “Simple and Efficient GPU Parallelization of Existing H-Matrix Accelerated BEM Code.” http://arxiv.org/abs/1711.01897.

Wang, Yingjun et al. 2013. “An Adaptive Dual-Information FMBEM for 3D Elasticity and Its GPU Implementation.” *Engineering Analysis with Boundary Elements* 37(2): 236–49. http://dx.doi.org/10.1016/j.enganabound.2012.09.012.

———. 2015. “Graphics Processing Unit (GPU) Accelerated Fast Multipole BEM with Level-Skip M2L for 3D Elasticity Problems.” *Advances in Engineering Software* 82: 105–18. http://dx.doi.org/10.1016/j.advengsoft.2015.01.002.

Zoccali, Pablo, Giuseppe Cantisani, and Giuseppe Loprencipe. 2015. “Ground-Vibrations Induced by Trains: Filled Trenches Mitigation Capacity and Length Influence.” *Construction and Building Materials* 74: 1–8. http://dx.doi.org/10.1016/j.conbuildmat.2014.09.083.