Implementing a 3D model for simulations in mechanics

Lasko Laskov
Informatics Department
New Bulgarian University
Sofia, Bulgaria
llaskov@nbu.bg

***Abstract*—A large category of problems that arise in physics require an efficient and robust implementation of a three-dimensional model that is specific and cannot be found in the existing software systems. Such models rely on 3D geometric objects representation to perform various simulations of different physical phenomena.**

Divers applications are based on 3D geometric objects, among which material deformation under certain conditions, heat propagation in a given environment, and as in our case, wave propagation simulation. Of course, the variety of examples that can be given is vast.

In this paper we present an implementation of a 3D model as part of our solution of a 3D elastodynamic problem for wave propagation in continuously inhomogenous half-space. The model is optimized to serve as basis of boundary integral equation method (BIEM), but the approach can be applied in various applications that require 3D geometric objects representation.

***Keywords*—model validation and analysis, three-dimensional modelling, numerical integration, object-oriented programming**

I. Introduction

Vast variety of tasks in different fields in practice raise the need of three-dimensional (3D) problems solutions in physics [4]. The diversity of such problems is not a subject of our research, but it's worth mentioning rigid body dynamics, harmonic motions of bodies, studying elastic properties of materials, and many other topics that are part of a classical course in mechanics.

3D geometric objects and shapes are a subject in various fields in computer science as well. Some of these fields are based on computer graphics [9], [10], and have numerous practical applications in 3D images generation, 3D models and animation. Gliding examples are implementation of 3D game engines, modelling for the needs of architecture and engineering, virtual reality.

The *reverse problem* in which a 3D scene is actually analyzed rather than generated, gives rise to various challenging tasks. Many methods from image processing [7], and pattern recognition [5] are focused on them in various state-of-the art applications, such as aerial and satellite image analysis, motion detection, robotics motion control, etc.

Solution of many of the problems that are subject of research in physics, and in particular in mechanics, can involve the compiling of a 3D geometric model for the needs of a computer simulation. In some of these cases the existing software systems cannot propose a complete solution, and special approach may be required. Exactly this is the case of our goal to develop a software system that solves a 3D elastodynamic problem for wave propagation in continuously inhomogenous half-space. The main method involved in our model is the *boundary integral equation method* (BIEM) [1] [3], and the 3D geometric representation must be optimized to serve efficiently its needs.

In this paper we present the implementation of our 3D geometric model. Even though our approach is designed to serve the needs of the BIEM, it can be adapted in various other applications that involve 3D modelling for problems is the field of mechanics.

This work is supported by research Grant IB-RA2014-178-EnTranEmiss from the Federal Ministry of Education Research in Germany.

II. Geometric model

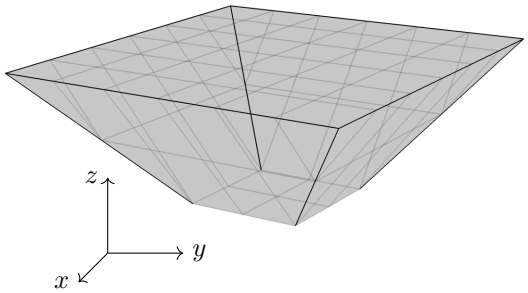
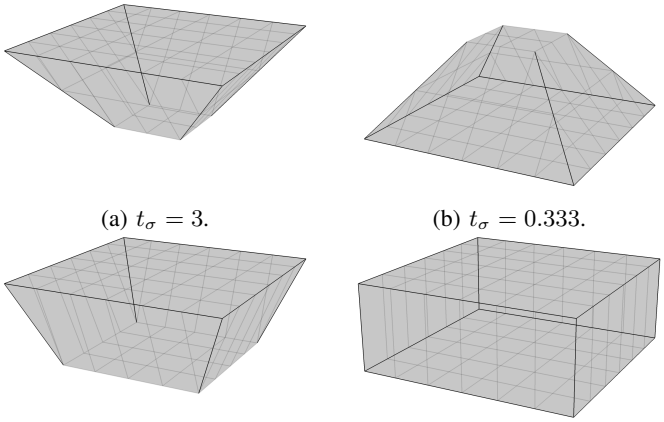


Fig. 1: Truncated square pyramid $\mathcal{P}$ located in the coordinate system O_{xyz}. The top base is larger than the bottom base.

The geometry that is implemented for the purposes of our simulation represents a 3D segment of a given environment with particular physical properties defined in Cartesian coordinate system O_{xyz}. The representation of the physical properties is described latter in Section III-A. The concrete 3D solid, which we will examine in this paper, is based on a truncated square pyramid, denoted with $\mathcal{P}$ (see Fig. 1). In the general case we will assume that the top base of $\mathcal{P}$ is larger than the bottom base, so $\mathcal{P}$ can be viewed as if it is rotated upside down.



(a) $t_\sigma = 3$. (b) $t_\sigma = 0.333$.

(c) $t_\sigma = 2$. (d) $t_\sigma = 1$.

Fig. 2: Example solids representative of the family generated by different the input parameter t_s of the model.

All the input parameters are flexible enough to determine the exact shape of $\mathcal{P}$, and can adjust it to fit the needs of the different experiments that must be conducted. We can easily modify the bases to be rectangles, instead of squares, or using particular input values we can degenerate the truncated square pyramid to a simple cuboid (see Fig. 2d), which by the way is required in some of the simulations. To accomplish the needed flexibility of the model, the following parameters are set:

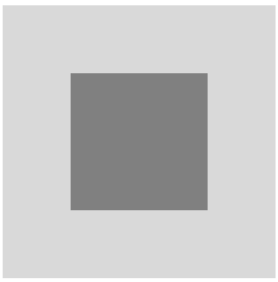
- a is the size parameter that basically is used to determine the size of the entire geometry. It represents the size of the bottom base of $\mathcal{P}$, and is used to calculate the other parameters of the geometry by the given scale factors below;
- h_σ is the scale factor to calculate the height of $\mathcal{P}$. The actual height is then $h = ah_\sigma$;
- t_σ is the scale factor to calculate the top base of $\mathcal{P}$. The side of the top base is given by $t = at_\sigma$.

The described simple parametric representation allows us to define a whole family of solids, for example by just by varying the parameter t_σ as in the Fig. 2 above.

Here we must clarify that in addition to the 3D segment that is defined by the square truncated pyramid $\mathcal{P}$, we must include a region in the plane where the top base $\mathcal{P}_t$ lies (see Fig. 3a). We will denote the rectangular area around $\mathcal{P}_t$ with $\mathcal{E}$, and we will label it *extended top*. $\mathcal{E}$ is composed by the following rectangular sub-areas that are placed counterclockwise around $\mathcal{P}_t$: $\mathcal{E}_A$, $\mathcal{E}_B$, $\mathcal{E}_C$, $\mathcal{E}_D$, $\mathcal{E}_E$, $\mathcal{E}_F$, $\mathcal{E}_G$ and $\mathcal{E}_H$. The side of each of them is obtained based on the side e of the whole area $\mathcal{E}$, which is calculated again using a scale factor e_σ, and it is given by $e = ae_\sigma$. Note that the definition of $\mathcal{E}$ is required by the BIEM which is applied latter on the 3D model.

The square truncated pyramid $\mathcal{P}$ together with the extended top $\mathcal{E}$ form the 3D solid $\bar{\mathcal{P}}$ that represents the segment of the 3D space used in our model, and:

$$\bar{\mathcal{P}} = \mathcal{P} \cup \mathcal{E}. \tag{1}$$



(a) Dark area denotes $\mathcal{P}_t$, while light is $\mathcal{E}$.

$\mathcal{E}_G$	$\mathcal{E}_F$	$\mathcal{E}_E$
$\mathcal{E}_H$	$\mathcal{P}_t$	$\mathcal{E}_D$
$\mathcal{E}_A$	$\mathcal{E}_B$	$\mathcal{E}_C$

(b) Scheme of the top base $\mathcal{P}_t$ and extended top.

Fig. 3: Extended area around the top of $\mathcal{P}$.

Furthermore, each surface of $\bar{\mathcal{P}}$, including each of the eight sub-areas of $\mathcal{E}$, must be divided in a mesh of *boundary elements* (BEs), which are used in BIEM. To calculate the location of each BE in space, each surface of $\bar{\mathcal{P}}$ is covered with a mesh, whose intersection points determine the vertices of the BEs. The mesh is constructed by dividing each side of the surface equally on a number of m segments.

On Fig. 4a is given a scheme of a square-shaped surface covered with a mesh calculated by dividing each side by $m = 3$ equal segments. Such surfaces can be the bottom base, the top base $\mathcal{P}_t$, and each of the extended top $\mathcal{E}$ sub-areas. In this case, the boundary elements are also shaped as squares.

The lateral faces of $\mathcal{P}$, however, are in the shape of trapezoids in the general case, and the BEs formed by the mesh on their surface are also shaped in the form of trapezoids (see Fig. 4b).

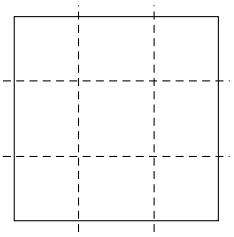
Each separate BE (see Fig. 4c) is defined by the coordinates of its vertices in O_{xyz}, and contains the coordinates of a special point, called the *nodal point* s, that plays an important role in integration step in BIEM. Note that the location of s must be shifted from the centroid of the BE by a small offset δ that is determined individually for each separate BE. Each BE has also a normal vector $\vec{n}$ associated with it that is also used by the BIEM algorithms.

The above definition of the BEs on the surfaces of $\bar{\mathcal{P}}$ determine the geometry that is needed for the model. $\bar{\mathcal{P}}$ is fully defined as a set of BEs with the coordinates of their vertices in O_{xyz}, nodal points and normal vectors.

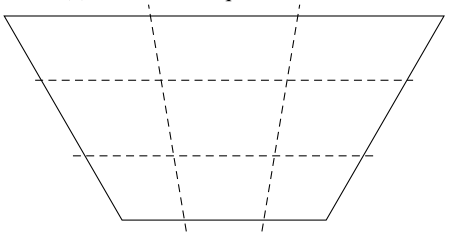
III. Implementation

The geometric model described in Section II is a part of our software system implemented using C++ programming language, and built using Gnu Compiler Collection (GCC). For visual components we are using the Qt framework [6], while the plots of the 3D solids are implemented using the MathGL library [12] which is a cross-platform collection of scientific plotting routines. For numerical methods primitives, we are adopting the Gnu Scientific Library (GSL) [8] which consists of powerful implementation of various mathematical routines.

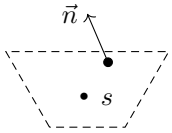
The module of our software that involves the implementation of the 3D model is called `Model`, and is composed by



(a) Mesh on a square surface.



(b) Mesh on a trapezoid surface.



(c) BE with nodal point s and normal vector $\vec{n}$.

Fig. 4: Mesh dividing surfaces of $\bar{\mathcal{P}}$ to calculate BEs locations in space. Each side of the surface is divided in equal segments, in this case the number of segments is $m = 3$.

six user-defined classes, plus a special class that implements the exceptions thrown by classes members. The graph of the classes is given on Fig. 5, where the solid rectangles denote user-defined classes, and rectangles in gray color denote library classes.

Shortly, the purpose of the classes is:

- `Physics` – implements physical parameters of the model;
- `Vector3` – algebraic concept of a vector in 3D, and is also used to represent a point in O_{xyz};
- `Be` – implements boundary element data type;
- `Shape3` – general idea of a 3D shape used as base class for classes implementing a particular solid;
- `Pyramid` – square truncated pyramid;
- `Graphics3` – visual graphs of 3D solids, derived derived from MathGL library class `mglDraw`;
- `BiemException` – a `logic_exception` class designed for the needs of the components of the system.

A. Physical parameters

The geometry of the model determines two domains: the exterior of $\bar{\mathcal{P}}$ denoted by Ω_0, and the interior of $\bar{\mathcal{P}}$ denoted by Ω_1. For each of them the following physical parameters are defined by the class `Physics`:

1) Lamé's first and second parameters λ and μ are elastic moduli which are used in mechanics to describe the linear elasticity of an isotropic material (see for example [11], p. 333).

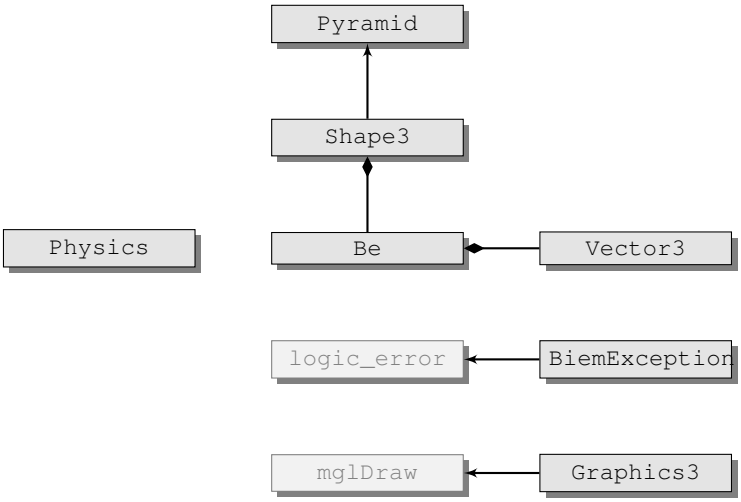


Fig. 5: Graph of the classes that implement the 3D model. The gray color denotes library classes, the arrow denotes inheritance, while diamond shows class composition.

2) Environment density ρ.
3) Primary wave v_p and secondary wave v_s velocities.
4) First and second wave numbers k_1 and k_2.

The expressions based on v_p and v_s are calculated and stored by the class `Physics`:

$$\frac{v_p}{v_s},\ \frac{v_s}{v_p},\ \left(\frac{v_p}{v_s}\right)^2,\ \text{and}\ \left(\frac{v_s}{v_p}\right)^2. \tag{2}$$

B. BEs and flat data representation

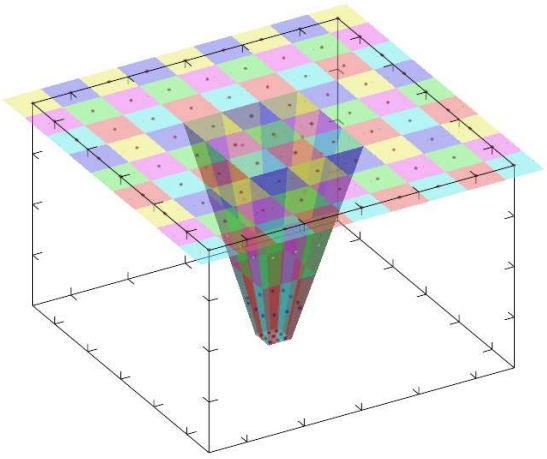
As described in Section II, each BE is defined by the coordinates of its four vertices in O_{xyz}, and stores two more special components:

- nodal point s, that is the shifted by δ centroid of the BE quadrilateral, where δ is determined individually for each separate BE;
- the normal vector $\vec{n}$.

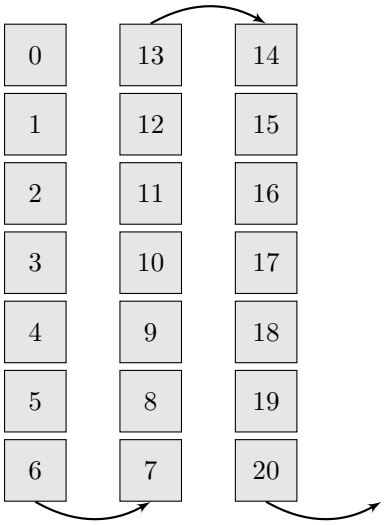
The class `Be` stores the above components, and also provides transformation from O_{xyz} coordinates to intrinsic coordinates for the BE, partial derivatives, and the Jacobean function of the BE in intrinsic coordinates.

The above implementation of the `Be` data type is the basis of our 3D model. The class `Shape3` provides the general concept of a 3D solid as a one-dimensional sequence of BEs that represents the discrete mesh that covers the surfaces of the object. Note that the representation of a multi-dimensional data in the form of a one-dimensional array is a classical approach that is referred to as *flat data* or *linear indexing*, and is commonly applied in many computer science tasks, for example digital images representation (see [7], p. 70).

`Shape3` stores the number of surfaces of the 3D solid, the exact discretization of the mesh, and the flat data representation of the BEs sequence. Knowing the consecutive number of the concrete BE in the sequence, it can restore its exact location on the 3D solid surface. This is the approach how the



(a) BEs that comprise $\bar{\mathcal{P}}$ generated by our software.



(b) Fragment of flat data representation.

Fig. 6: $\bar{\mathcal{P}}$ is fully described by the set of BEs that are stored in the form of flat data.

whole 3D model is encoded as one-dimensional array (see Fig. 6b), which subsequently is used by the algorithms of BIEM.

The exact shape of the 3D solid is determined by classes that inhere from `Shape3`, an in this particular example this is the class `Pyramid` that defines the square truncated pyramid, based on the input parameters a, h_σ, t_σ, and e_σ. For example, the geometry generated by our software for $a = 100$, $h_\sigma = 2$, $t_\sigma = 5$, $e_\sigma = 3$, and mesh discretization $m = 3$ is given on Fig. 6a. Note that the graph actually contains only the plots of each of the BEs, that drawn together form the shape of $\bar{\mathcal{P}}$ defined by the input parameters. Fig. 6b is a scheme that presents a fragment of the flat data representation of $\bar{\mathcal{P}}$ as a one-dimensional sequence of BEs, where the index of a given BEs unambiguously defines its location on the surface of the 3D sold.

IV. Conclusion

The implementation of the 3D model that we present here is optimized to serve the purposes of the boundary integral equation method (BIEM) for solution of a 3D elastodynamic problem for wave propagation in continuously inhomogenous half-space. The flexibility of the presented approach comes mainly from the flat-data representation that is based on one-dimensional sequence of BEs, and the logical separation of its storage from the concrete 3D solid that is implemented. This separation is provided by the inheritance hierarchy of the classes that comprise the implementation of the 3D geometry. The input parameters that determine the exact 3D shape a, h_σ, t_σ, and e_σ, together with the discretization parameter m that determines the number of BEs on each surface of the solid, provide the capability of the model to run experiments with broad family of shapes. Our implementation allows the representation of a given solid width variety of number of BEs, which is an important feature for BIEM algorithms: for example a surface can be discretized in single, 4, 9, 16, and so on, number of BEs.

Our implementation has been verified on a broad variety of representatives of 3D solids, such as those depicted on Fig. 2, and BIEM has been tested based on our model.

As future work, the model will be extended to broader types 3D shapes, for example to an approximation of a sphere that is again composed by a set of BEs. Also, similar approach can be adopted by similar systems that solve 3D problems in physics. Another direction for improvement of our work is the application of parallel programming.

References

[1] Brebbia, C. A., J. C. F. Telles, L. Wrobel Boundary Element Techniques: Theory and Applications in Engineering. Springer-Verlag Berlin Heidelberg, ISBN 9783642488627, 1984.

[2] Burden, R. L., J. D. Faires Numerical Analysis. Brooks/Cole, Cengage Learning, ISBN 9780538735643, 2011.

[3] Constanda, C., D. Doty, W. Hamill Boundary Integral Equation Methods and Numerical Solutions. Developments in Mathematics, Springer International Publishing, ISBN 978331926300, 2016, 54–62.

[4] Dourmashkin, P. A. Classical Mechanics: MIT 8.01 Course Notes. Wiley Custom Learning Solutions, ISBN 1118952804, 9781118952801, 2014.

[5] Duda, R. O., Hart, E. Peter, D. G. Stork Pattern Classification, 2nd Edition. Wiley-Interscience, ISBN 9780471056690, 2000.

[6] Eng, L. Z. Qt5 C++ GUI Programming Cookbook, 2nd Edition. PACKT Publishing, ISBN 9781789803822,178980382, 2019.

[7] Gonzalez, R. C., R. E. Woods Digital Image Processing, 4th Edition. Pearson, ISBN 9353062985, 9789353062989, 2018.

[8] Gough, B. GNU Scientific Library Reference Manual - Third Edition. Network Theory Ltd., ISBN 0954612078, 2009.

[9] Foley, D. J., A. V. Dam, S. K. Feiner, J. F. Hughes, R. L. Phillips Introduction to Computer Graphics. Addison-Wesley, ISBN 0201609215, 9780201609219, 1994.

[10] Foley, D. J., A. V. Dam, S. K. Feiner, J. F. Hughes Computer Graphics: Principles and Practice. Addison-Wesley Professional, ISBN 0201848406, 9780201848403, 1996.

[11] Salencon, J. Handbook of Continuum Mechanics: General Concepts - Thermoelasticity Springer Science & Business Media, ISBN 3540414436, 9783540414438, 2001.

[12] Balakin, A.A. http://mathgl.sourceforge.net/doc_en/index.html