Boundary Element Software for 3D Linear Elasticity

BESLE

User's Guide Version 1.0

Andres F. Galvis
Daniel M. Prada
Lucas S. Moura
Cecilia Zavaglia
Jamie M. Foster
Paulo Sollero
Luiz C. Wrobel

October 2020

References

In case that you are using BESLE version 1.0 please cite:

[1] A. F. Galvis, D. M Prada, L. S. Moura, C. Zavaglia, J. M. Foster, P. Sollero, and L. C. Wrobel. BESLE: Boundary Element Software for 3D Linear Elasticity, *Comput. Phys. Commun.*, 2021.

[2] A. F. Galvis, P. A. Santos-Flórez, P. Sollero, M. de Koning, and L. C. Wrobel. Multiscale model of the role of grain boundary structures in the dynamic intergranular failure of polycrystal aggregates, *Comput Methods. Appl. Mech. Eng.*, vol. 362 p. 112868, 2020.

Contents

Introduction

BESLE is an open-source code to solve 3D elasticity problems under quasi-static, inertial, and high-rate boundary conditions. It uses the 3D elastostatic and elastodynamic boundary element method with the fundamental solutions based on double Fourier series. The software supports simulations of isotropic and anisotropic bodies considering, if necessary, several domains for modeling heterogeneous materials composed of different constituents. A distinguishing feature of BESLE is that it carries out customized solid simulations in a straightforward configuration only using the surface mesh information. The software includes with external sub-packages for creating a material database, options for mesh generation, and diverse ways for the configuration of the boundary conditions.

1.1 Features

BESLE attempts to be the first open-source code based on boundary elements for the above-mentioned applications in 3D linear elasticity. Classical limitations using BEM have appeared since the first developments on BEM, such as the number of degrees of freedom (DOF) caused by the fully populated non-symmetrical incidence matrices. To deal with this issue an MPI parallel algorithm described in [1, 2] was incorporated offering an efficient solution for this critical step of the BEM solver. Furthermore, a numerical approach recently emerged, the adaptive cross approximation, otherwise known as hierarchical matrices [3–5] and the high-performance BEM adaptations described in [6, 7]. These improvements conduce to BESLE, if an adequate balance between the number of DOFs and subdomains is set, to deal with large scale problems.

The main code is implemented in Fortran 90 modules that can be compiled and modified from a setup subroutine utility following the instruction provided in this User's Guide. BESLE employs external free codes and complementary software such as Multifrontal Massively Parallel Sparse (MUMPS) [8, 9], ScaLAPACK, SCOTCH, BLAS, BLACS, and LAPACK. The user must download and install them according to the instructions given in the INSTALL file. In order to run the different examples shown in the User's Guide, this application requires the previous installation of MPI. Moreover, BESLE accounts only with a parallel version, and therefore, at least 2 threads must be indicated in its execution. Additionally, for simulations with a large number of degrees of freedom, MUMPS should be installed in its 64-bit version.

Several simulation analyses can be set in BESLE: i) static, quasi-static, dynamic, and transient regimes, using ii) morphologies with a single or multiple domains with iii) isotropic and anisotropic materials. Moreover, it supports datasets of Dirichlet and

Neumann boundary conditions, and elastic material properties. The construction of a 3D surface mesh, composed of single or multiple domains is a complex task and there are not many surface generators available. For this purpose, BESLE includes a module to generate the mesh and boundary conditions based on 3ds Max[1]. Additionally, the analysis of geometries containing cracks was implemented in BESLE, and hence, the stress concentration and stress intensity factors (SIF) can be evaluated. Results of SIF for mode I have been reported in [10, 11] using BESLE and Ncorr[2].

BESLE is available to be executed in Desktop and Server Ubuntu versions. In general, the main BEM package is supported by five sub-codes: i) solver, ii) general mesh generator based on 3ds Max, iii) polycrystal structure generator, iv) MATLAB box mesh generator, and v) material database. The user is encouraged to read the User's Guide and to explore the examples therein. Finally, for post-processing, *.vtk files are exported ready to visualise in ParaView.

1.2 Package

The software files to be download are

INSTALL	README	VERSION	Makefile
Make.inc/	doc/	src/	Mesh/
Material/	Cracks/	Body_forces/	Examples/

Make.inc/	contains all the Makefile.inc necessary for the instalation of MUMPS, SCOTCH and ScaLAPACK.
doc/	contains the users' guide in pdf format.
src/	source *.f90 files of the main BESLE code.
Mesh/	options for the mesh generation are provided in this folder i) General using 3dxMax and ii) Polycrystal and iii) MATLAB box.
Material/	is the folder that contains a serial Frotran code to generate the material database.
Cracks/	contains the *.dat file with pre-cracks.
Body_forces/	contains the *.dat file with a body force vector.
Examples/	contains the mesh, material and simulation examples.

Additional contents by the time of installation:

MUMPS/	The MUMPS solver must be installed in its 64-bit version.
voro++/	3D Voronoi structure generator.
triangle/	two-dimensional quality mesh generator and delaunay triangulator.

[1] 3D modelling and rendering software for design visualisation, games, and animation.
[2] 2D digital image correlation MATLAB program.

The voro++ [12] and triangle [13] libraries have to be installed in case the user wants to reproduce artificial polycrystalline structures. If this is not the case, their installation could be skipped. After BESLE compilation, temporary folders are generated as follows.

obj/	contains the *.o and *.mod files produced by the general compilation of BESLE.
OOC/	is the folder that MUMPS uses to write data while the system of equations is being solved. After a successful solution, MUMPS deletes these files itself.
Results/	contains the .vtk files for visualization using ParaView.

Finally, the sub-packages material, general mesh, polycrystal structures also generate similar temporary files after compilation.

1.3 Installation

This software must be installed in an Ubuntu distribution due to its parallel (distributed memory MPI based) implementation.

MPI

It is recomended to install "libopenmpi-dev" using the Synaptic Package Manager, or using terminal with

```
~$ sudo apt-get install libopenmpi-dev
```

an additional option is MPICH using

```
~$ sudo apt-get install mpich
```

BLAS, LAPACK and zlib1g

```
~$ sudo apt-get install libblas-dev liblapack-dev zlib1g-dev
```

MUMPS

In order to install the parallel sparse direct solver, the user needs to fill the download request submission in MUMPS. The folder must be renamed simply as MUMPS/ and added within the BESLE/ root. According to the installation instructions of MUMPS, the latest released versions of the ScaLAPACK and SCOTCH libraries are necessary. Both must be renamed simply as scotch/ and scalapack/, respectively. These packages must be added to the BESLE/MUMPS/ root. In order to install the SCOTCH library, the user needs to take the Makefile.inc.scotch from the BESLE/Make.inc folder copy and paste it into BESLE/MUMPS/scotch/src/ with Makefile.inc name. Then, install with

```
~/BESLE/MUMPS/scotch/src$ make ptscotch
~/BESLE/MUMPS/scotch/src$ make scotch
~/BESLE/MUMPS/scotch/src$ make esmumps
~/BESLE/MUMPS/scotch/src$ make ptesmumps
```

ptscotch might require to specify the root of "mpi.h" at line 123 in the common.h file located in the BESLE/MUMPS/scotch/src/libscotch folder. Usually, the root is

```
#include </usr/include/mpi/mpi.h>
```

This depends on the computer and the user must verify the root. To install the ScaLAPACK library, the user needs to take the SLmake.inc.scalapack from the BESLE/Make.inc folder, copy and paste it into BESLE/MUMPS/scalapack/ with SLmake.inc name. Then, install with

```
~/BESLE/MUMPS/scalapack$ make
```

To install the MUMPS solver, the user needs to take the Makefile.inc.MUMPS_Install from the BESLE/Make.inc folder, copy and paste it into BESLE/MUMPS/ with Makefile.inc name. Then, install with

```
~/BESLE/MUMPS$ make all
```

take the Makefile.inc.MUMPS_run from the BESLE/Make.inc folder, copy and paste it into BESLE/MUMPS/ with Makefile.inc name.

finally, to build the BESLE application

```
~/BESLE$ make
```

installation completed !

Polycrystal aggregates

In order to simulate the mechanical behavior of polycrystalline materials, a mesh generator is provided in BESLE/Mesh/Polycrystal, the code requires the previous installation of the Voro++ to generate the crystalline structure, which can be found in Voro++. The user needs to download the latest released version, in this case, change the folder name simply as voro++ and add it to the BESLE/Mesh/Polycrystal/ root. Now, the user can install the Voro++ using

```
~/BESLE/Mesh/Polycrystal/voro++$ make
~/BESLE/Mesh/Polycrystal/voro++$ sudo make install
```

Next, the discretization of each aggregate is carried out using the Triangle mesh generator, which can be downloaded from on Triangle. The user needs to add the folder called triangle to BESLE/Mesh/Polycrystal/ root. Then, to build the application

```
~/BESLE/Mesh/Polycrystal$ make
```

installation completed !

1.4 Acknowledgements

The authors would like to thank to the University of Campinas (Brazil), Brunel University London (UK), and the University of Portsmouth (UK) for the facilities and structure provided to develop this version of BESLE. The project was funded by the National Council for Scientific and Technological Development-CNPq (Grant Numbers: 312493/2013-4, 154283/2014-2 and 312536/2017-8), and the Brazilian Coordination for the Improvement of Higher Education Personnel-CAPES (Grant Number: 435214/2019-01).

This material is based upon work supported by the Air Force Office of Scientific Research-AFOSR under Award Numbers FA9550-18-1-0113 and FA9550-20-1-0133.

Andres F. Galvis was supported by the EPSRC New Investigator Award "Multiscale modelling of mechanical deterioration in lithium-ion batteries" Grant number EP/T000775/1.

Luiz C. Wrobel also thanks the CNPq for his personal financial support (Grant Number: 303770/2019-8).

Computational resources were provided by the Center for Computational Engineering and Science-CCES at the University of Campinas funded by the São Paulo Research Foundation-FAPESP (Grant Number: 2013/08293-7).

Boundary element method

The BEM formulation allows modelling high gradients of different mechanical fields only using the domain's surface information. The surface discretisation leads to a reduction in the number of DOFs used in the model. BEM requires a fundamental solution (Green's function), which depends on the analysed field. In the modeling of isotropic and general anisotropic solids, the displacement fundamental solution based on double Fourier series is implemented, see section 4. For the case of different constitutive domains, the multidomain algorithm needs to be incorporated. The inclusion of dynamic effects such as those caused by body forces or high-rate boundary conditions are also presented here. In order to develop the dynamic analysis, it is necessary for the transformation of the domain integral into a boundary integral, through the application of the dual reciprocity BEM formulation (DRBEM). Furthermore, Dirichlet and Neumann boundary conditions can be applied to the model.

2.1 Integral formulation

The boundary integral equation expresses the relation of the displacement u_i and traction t_i on a surface Γ using the known fundamental solutions for displacement $U_{ik}(\mathbf{x}', \mathbf{x})$ and traction $T_{ik}(\mathbf{x}', \mathbf{x})$. For homogeneous elastic bodies, the boundary integral equation considering the body forces on the domain Ω is given by

$$c_{ik}(\mathbf{x}')u_i(\mathbf{x}') + \int_\Gamma T_{ik}(\mathbf{x}', \mathbf{x})u_i(\mathbf{x})\,\mathrm{d}\Gamma = \int_\Gamma U_{ik}(\mathbf{x}', \mathbf{x})t_i(\mathbf{x})\,\mathrm{d}\Gamma + \int_\Omega \rho\ddot{u}_i U_{ik}(\mathbf{x}', \mathbf{x})\,\mathrm{d}\Omega \quad , \tag{1}$$

where $(\mathbf{x}')$ and $(\mathbf{x})$ are the source and field points respectively. $c_{ik}(\mathbf{x}')$ is $\delta_{ik}/2$ for a smooth surface boundary at the source point and ρ is the mass density. For transient analysis, the body forces are caused by the acceleration field $\ddot{u}_i$. Thus, in order to apply the BEM formulation, it is required the transformation of this domain integral into a boundary or surface integral. In BESLE, the DRBEM is implemented to pursue this transformation. The acceleration field of the domain integral in Eq. (1) can be represented by

$$\rho\ddot{u}_i = \sum_{j=1}^{M} f_{mk}^j(x)\,\alpha_m^j \quad , \tag{2}$$

where α_m^j are unknown coefficients and f_{mk}^j are M radial functions that have to fulfill the equilibrium equation, in the D'Alembert sense

$$C_{mnrs}\hat{u}_{rk,ns} = f_{mk}^j \quad . \tag{3}$$

The term $\hat{u}_{rk}$ is the particular solution to solve Eq. (3). Substituting Eq. (2) into the fourth integral of Eq. (1) gives

$$\int_\Omega \rho \ddot{u}_i U_{ik} \mathrm{d}\Omega = \sum_{j=1}^{M} \alpha_n^j \int_\Omega U_{ik} f_{kn}^j \mathrm{d}\Omega \quad . \tag{4}$$

The reciprocal integral relation can also be obtained between the fundamental solutions and the particular solution as

$$c_{ik}\hat{u}_{kn}^j + \int_\Gamma T_{ik}\hat{u}_{kn}^j \mathrm{d}\Gamma = \int_\Gamma U_{ik}\hat{t}_{kn}^j \mathrm{d}\Gamma + \int_\Omega U_{ik} f_{kn}^j \mathrm{d}\Omega \quad , \tag{5}$$

by substituting Eq. (5) into Eq. (4) and then into Eq. (1), the integral equation results in

$$c_{ik}u_i + \int_\Gamma T_{ik}u_i \mathrm{d}\Gamma = \int_\Gamma U_{ik}t_i \mathrm{d}\Gamma + \sum_{j=1}^{M} \alpha_n^j \left\{ c_{ik}\hat{u}_{kn}^j - \int_\Gamma U_{ik}\hat{t}_{kn}^j \mathrm{d}\Gamma + \int_\Gamma T_{ik}\hat{u}_{kn}^j \mathrm{d}\Gamma \right\} \quad , \tag{6}$$

where the particular solution $\hat{u}_{in}^m$ is a radial function [14] expressed as

$$\hat{u}_{rk} = \delta_{kn}(r^2 + r^3) \quad , \tag{7}$$

and its derivatives

$$\begin{aligned} \hat{u}_{rk,l} &= \delta_{rk}(2r + 3r^2)r_{,l} \quad , \\ \hat{u}_{rk,lj} &= \delta_{rk}\left((2 + 3r)\delta_{lj} + 3r r_{,j} r_{,l}\right) \quad , \end{aligned} \tag{8}$$

the particular solution $\hat{t}_{rk}^j$ can be evaluated using

$$\hat{t}_{ik} = (\hat{\sigma}_{il} n_l)_k \quad , \tag{9}$$

where $\hat{\sigma}_{il}$ is the particular solution of stresses and n_l is the outward normal vector on the surface at the field point. Finally, using the generalized Hooke's law

$$(\hat{\sigma}_{il})_k = C_{ilmn}(\hat{u}_{mk,n} + \hat{u}_{nk,m})/2 \quad , \tag{10}$$

the complete transformation from Eq. (1) to the Eq. (6) can be carried out now.

2.2 Discretisation

The Eq. (6) must be discretised into surface elements, a different type of elements can be used such as quadrilateral or triangular, each of them can be implemented usually as linear or quadratic [15]. In BESLE, linear three-node discontinuous triangular boundary elements are used. Discontinuous elements offer advantages in the implementation of the multidomain algorithm because there are no shared nodes by more than two domains. Rewriting the Eq. (6) the solid is represented by

$$
\begin{aligned}
c_{ik}u_i + \int_\Gamma T_{ik}u_i \, \mathrm{d}\Gamma = \int_\Gamma U_{ik}t_i \, \mathrm{d}\Gamma \\
+ \sum_{j=1}^{M} \alpha_n^j \left(c_{ik}\hat{u}_{kn}^j + \int_\Gamma T_{ik}\hat{u}_{kn}^j \mathrm{d}\Gamma - \int_\Gamma U_{ik}\hat{t}_{kn}^j \mathrm{d}\Gamma \right) \quad .
\end{aligned}
\tag{11}
$$

The linear three-node discontinuous element is shown in Fig. 1, where the three nodes are function of the two intrinsic parametric (η, ξ) coordinates. Thus, the position of all nodes in the element is controlled by the parametric distance λ.

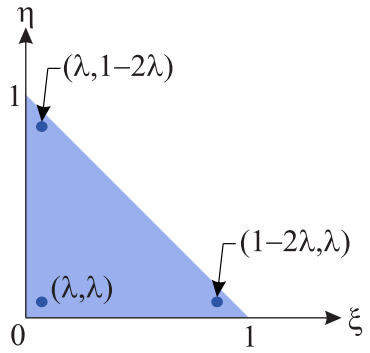


Figure 1: Linear three-node discontinuous element.

The surface response is interpolated within an element from the nodal values, using shape functions $h^{(1)}$ $h^{(2)}$ $h^{(3)}$ corresponding to the nodes 1, 2, and 3 respectively. The interpolated field X expressed in terms of the known nodal values $X^{(k)}$ is

$$
X(\xi, \eta) = \sum_{k=1}^{3} h^{(k)}(\xi, \eta) X^{(k)} \quad , \tag{12}
$$

the first derivative of the interpolated field, in terms of the shape functions, can be written as

$$
\begin{aligned}
\frac{\partial X(\xi, \eta)}{\partial \xi} &= \sum_{k=1}^{3} \frac{\partial h^{(k)}(\xi, \eta)}{\partial \xi} X^{(k)} \quad , \\
\frac{\partial X(\xi, \eta)}{\partial \eta} &= \sum_{k=1}^{3} \frac{\partial h^{(k)}(\xi, \eta)}{\partial \eta} X^{(k)} \quad .
\end{aligned}
\tag{13}
$$

The shape functions of the three-node discontinuous elements are evaluated numerically following the procedure shown in [15], where these shape functions are obtained from the known shape functions of the continuous three-node triangular element Fig. 2.

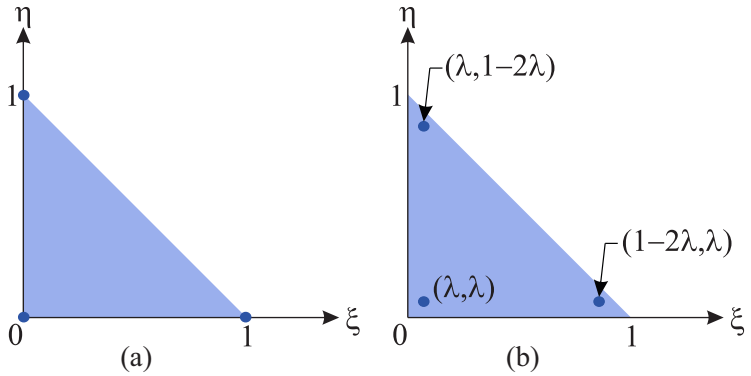


Figure 2: Linear three-node elements: (a) continuous and (b) discontinuous.

The known shape functions of the continuous three-node triangular element are

$$\begin{aligned} N^{(1)} &= \xi \quad , \\ N^{(2)} &= \eta \quad , \\ N^{(3)} &= 1 - \xi - \eta \quad . \end{aligned} \tag{14}$$

The interpolated field X expressed in terms of the known node values $X^{(k)}$ are

$$X(\xi, \eta) = \sum_{k=1}^{3} N^{(k)}(\xi, \eta) X^{(k)} \quad . \tag{15}$$

Now, assuming w_j as the interpolated values of the new discontinuous element and forcing the Eq. (15)

$$w_j = \sum_{k=1}^{3} N^{(k)}(\xi, \eta) X^{(k)} \quad , \tag{16}$$

expanding the matrix form

$$\left\{ \begin{array}{c} w_1 \\ w_2 \\ w_3 \end{array} \right\} = \left[\begin{array}{ccc} N^{(1)}(1-2\lambda, \lambda) & N^{(2)}(1-2\lambda, \lambda) & N^{(3)}(1-2\lambda, \lambda) \\ N^{(1)}(\lambda, 1-2\lambda) & N^{(2)}(\lambda, 1-2\lambda) & N^{(3)}(\lambda, 1-2\lambda) \\ N^{(1)}(\lambda, \lambda) & N^{(2)}(\lambda, \lambda) & N^{(3)}(\lambda, \lambda) \end{array} \right] = \left\{ \begin{array}{c} X^{(1)} \\ X^{(2)} \\ X^{(3)} \end{array} \right\} \quad , \tag{17}$$

or

$$\{w\} = [L]\{X\} \quad , \tag{18}$$

where Eq. (18) can be solved for the X values as

$$\{X\} = [L]^{-1}\{w\} = [G]\{w\} \quad . \tag{19}$$

Substituting Eq. (19) into Eq. (15) the formula for the matrix interpolation functions of the discontinuous element is obtained

$$X(\xi,\eta)=\sum_{k=1}^{3}N^{(k)}(\xi,\eta)X^{(k)}=[N]\{X\}=[N][G]\{w\}=[H]\{w\} \quad , \tag{20}$$

where $[H]=[h^{(1)}h^{(2)}h^{(3)}]$. The derivatives of the shape functions with respect to the intrinsic coordinates ξ and η need to be expressed as

$$\begin{aligned}\frac{\partial x_j(\xi,\eta)}{\partial\xi}&=\sum_{k=1}^{3}\frac{\partial N^{(k)}(\xi,\eta)}{\partial\xi}x_j^{(k)}=[N]_{,\xi}[G]\{w_j\}=[H]_{,\xi}\{w_j\} \quad ,\\ \frac{\partial x_j(\xi,\eta)}{\partial\eta}&=\sum_{k=1}^{3}\frac{\partial N^{(k)}(\xi,\eta)}{\partial\eta}x_j^{(k)}=[N]_{,\eta}[G]\{w_j\}=[H]_{,\eta}\{w_j\} \quad ,\end{aligned} \tag{21}$$

the Jacobian and the normal vectors are given by Eq. (22).

$$J=0.5\left(J_1^2+J_2^2+J_3^2\right)^{0.5}=0.5\left(J_kJ_k\right)^{0.5} \quad , \tag{22}$$

where

$$\begin{aligned}J_1&=\frac{\partial x_2}{\partial\xi}\frac{\partial x_3}{\partial\eta}-\frac{\partial x_3}{\partial\xi}\frac{\partial x_2}{\partial\eta} \quad ,\\ J_2&=\frac{\partial x_3}{\partial\xi}\frac{\partial x_1}{\partial\eta}-\frac{\partial x_1}{\partial\xi}\frac{\partial x_3}{\partial\eta} \quad ,\\ J_3&=\frac{\partial x_1}{\partial\xi}\frac{\partial x_2}{\partial\eta}-\frac{\partial x_2}{\partial\xi}\frac{\partial x_1}{\partial\eta} \quad .\end{aligned} \tag{23}$$

The terms x_k represent the coordinates of the actual element, and the normal vector is expressed as

$$n_k=J_kJ^{-1} \quad . \tag{24}$$

From the Eq. (15) to Eq. (21) the w_j terms contain the three coordinates of the nodes of the discontinuous elements, the matrix $[G]$ is the same in Eq. (20) and Eq. (21). Finally the matrix $[L]$ can be evaluated numerically. Therefore, rewriting Eq. (11) gives

$$\begin{aligned}c_{ik}u_i+\sum_{\mathfrak{e}=1}^{N_\mathfrak{e}}\int_{\Gamma_\mathfrak{e}}T_{ik}\left(\sum_{\mathfrak{n}=1}^{N_\mathfrak{n}^\mathfrak{e}}h^\mathfrak{n}{u_i}^\mathfrak{n}\right)\mathrm{d}\Gamma_\mathfrak{e}&=\sum_{\mathfrak{e}=1}^{N_\mathfrak{e}}\int_{\Gamma_\mathfrak{e}}U_{ik}\left(\sum_{\mathfrak{n}=1}^{N_\mathfrak{n}^\mathfrak{e}}h^\mathfrak{n}{t_i}^\mathfrak{n}\right)\mathrm{d}\Gamma_\mathfrak{e}\\ &+\sum_{j=1}^{M}\alpha_n^{j,\mathfrak{g}}\left(c_{ik}\hat{u}_{kn}^j+\sum_{\mathfrak{e}=1}^{N_\mathfrak{e}}\int_{\Gamma_\mathfrak{e}}T_{ik}\hat{u}_{kn}^j\mathrm{d}\Gamma_\mathfrak{e}-\sum_{\mathfrak{e}=1}^{N_\mathfrak{e}}\int_{\Gamma_\mathfrak{e}}U_{ik}\hat{t}_{kn}^j\mathrm{d}\Gamma_\mathfrak{e}\right) \quad .\end{aligned} \tag{25}$$

In Eq. (25), the surface is divided by the elements $\mathfrak{e}$ and the nodes $\mathfrak{n}$ in the element. The integration of the Eq. (25) is carried out by the numerical Gauss integration, for triangles following the procedures presented in [15]. Thus, the Eq. (25) becomes

$$c_{ik}u_i + \sum_{\mathfrak{e}=1}^{N_\mathfrak{e}} \left(\sum_{\mathfrak{n}=1}^{N_\mathfrak{n}} \int_0^1 \int_0^{1-\eta} T_{ik} h^\mathfrak{n} J \mathrm{d}\xi \mathrm{d}\eta \right) u_i{}^\mathfrak{n} = \sum_{\mathfrak{e}=1}^{N_\mathfrak{e}} \left(\sum_{\mathfrak{n}=1}^{N_\mathfrak{n}} \int_0^1 \int_0^{1-\eta} U_{ik} h^\mathfrak{n} J \mathrm{d}\xi \mathrm{d}\eta \right) t_i{}^\mathfrak{n}$$
$$+ \sum_{j=1}^{M} \alpha_n^j \left[c_{ik}\hat{u}_{kn}^j + \sum_{\mathfrak{e}=1}^{N_\mathfrak{e}} \left(\sum_{\mathfrak{n}=1}^{N_\mathfrak{n}^\mathfrak{e}} \int_0^1 \int_0^{1-\eta} T_{ik} h^\mathfrak{n} J \mathrm{d}\xi \mathrm{d}\eta \right) \hat{u}_{kn}^j \right. \tag{26}$$
$$\left. - \sum_{\mathfrak{e}=1}^{N_\mathfrak{e}} \left(\sum_{\mathfrak{n}=1}^{N_\mathfrak{n}^\mathfrak{e}} \int_0^1 \int_0^{1-\eta} U_{ik} h^\mathfrak{n} J \mathrm{d}\xi \mathrm{d}\eta \right) \hat{t}_{kn}^j \right] \quad ,$$

where the integration of the traction fundamental solution is defined as

$$c_{ik} + \sum_{\mathfrak{e}=1}^{N_\mathfrak{e}} \left(\sum_{\mathfrak{n}=1}^{N_\mathfrak{n}^\mathfrak{e}} \int_0^1 \int_0^{1-\eta} T_{ik} h^\mathfrak{n} J \mathrm{d}\xi d\eta \right) = \mathbf{H} \quad , \tag{27}$$

and the integration of the displacement fundamental solution is

$$\sum_{\mathfrak{e}=1}^{N_\mathfrak{e}} \left(\sum_{\mathfrak{n}=1}^{N_\mathfrak{n}^\mathfrak{e}} \int_0^1 \int_0^{1-\eta} U_{ik} h^\mathfrak{n} J \mathrm{d}\xi d\eta \right) = \mathbf{G} \quad , \tag{28}$$

rewriting Eq. (26) in the matrix form

$$\mathbf{Hu} = \mathbf{Gu} + \sum_{j=1}^{M} \boldsymbol{\alpha}^j \left(\mathbf{H}\hat{\mathbf{u}}^j - \mathbf{G}\hat{\mathbf{t}}^j \right) \quad . \tag{29}$$

The summation in Eq. (29), is over the total number of nodes M of the physical problem. Then, the final equation in matrix form over a single domain is

$$\mathbf{Hu} = \mathbf{Gt} + \left(\mathbf{H}\hat{\mathbf{U}} - \mathbf{G}\hat{\mathbf{T}} \right) \boldsymbol{\alpha} \quad . \tag{30}$$

From Eq. (2) in its matrix form, the term $\boldsymbol{\alpha}$ can be expressed as

$$\boldsymbol{\alpha} = \rho \mathbf{E}\ddot{\mathbf{u}} \quad , \tag{31}$$

where $\mathbf{E}$ is the inverse of the matrix $\mathbf{F}$, the matrix $\mathbf{F}$ contains all the components of f_{mk}^j. Substituting the Eq. (31) into Eq. (30)

$$\mathbf{Hu} = \mathbf{Gt} + \rho \left(\mathbf{H}\hat{\mathbf{U}} - \mathbf{G}\hat{\mathbf{T}} \right) \mathbf{E}\ddot{\mathbf{u}} \quad . \tag{32}$$

In this formulation, the mass matrix can be defined as $\mathbf{M} = \rho \left(\mathbf{G}\hat{\mathbf{T}} - \mathbf{H}\hat{\mathbf{U}} \right) \mathbf{E}$, the final form of Eq. (32) is

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{Hu} = \mathbf{Gt} \quad . \tag{33}$$

At this point, the formulation considers the inertial effects of the body caused by its own acceleration. Moreover, body forces can be added to the right-hand term in Eq. (33).

2.3 Time domain integration

The transient regime analysis of 3D solids requires the time-domain integration of Eq. (33). The Houbolt's algorithm [16] is an appropriate implicit method to be used coupled to the DRBEM [17,18]. The time-dependent solution of Eq. (33) is obtained at every instant $\tau+\Delta\tau$. Therefore, the acceleration of the body can be expressed as

$$\ddot{\mathbf{u}}_{\tau+\Delta\tau} = \frac{1}{\Delta\tau^2}\left(2\mathbf{u}_{\tau+\Delta\tau} - 5\mathbf{u}_{\tau} + 4\mathbf{u}_{\tau-\Delta\tau} - \mathbf{u}_{\tau-2\Delta\tau}\right) \quad , \tag{34}$$

the matrix Eq. (33) at instant $\tau + \Delta\tau$ is

$$\mathbf{M}\ddot{\mathbf{u}}_{\tau+\Delta\tau} + \mathbf{H}\mathbf{u}_{\tau+\Delta\tau} = \mathbf{G}\mathbf{t}_{\tau+\Delta\tau} \quad . \tag{35}$$

The substitution of the Eq. (34) into Eq. (35) leads to Eq. (36), which is the response at instant $\tau + \Delta\tau$ using the information of the last three time steps

$$\left[\frac{2}{\Delta\tau^2}\mathbf{M} + \mathbf{H}\right]\mathbf{u}_{\tau+\Delta\tau} = \mathbf{G}\mathbf{t}_{\tau+\Delta\tau} + \frac{1}{\Delta\tau^2}\mathbf{M}\left(5\mathbf{u}_{\tau} - 4\mathbf{u}_{\tau-\Delta\tau} + \mathbf{u}_{\tau-2\Delta\tau}\right) \quad . \tag{36}$$

In Eq. (36), vectors $\mathbf{u}_{\tau+\Delta\tau}$ and $\mathbf{t}_{\tau+\Delta\tau}$ are the displacement and traction fields at the instant $\tau + \Delta\tau$, respectively.

2.4 Multidomain

For heterogeneous solids with more than one domain where the boundary conditions are imposed in the external domains, the displacement compatibility and traction equilibrium must be applied at the interfaces, as shown in the following equation

$$\begin{aligned} \mathbf{u}_i^j &= \mathbf{u}_j^i \quad , \\ \mathbf{t}_i^j &= -\mathbf{t}_j^i \quad . \end{aligned} \tag{37}$$

where the i and j indices represent the $\mathbf{\Omega}_i$ and $\mathbf{\Omega}_j$ domains. Fig. 3 shows an interface, being $\mathbf{\Gamma}_i$ and $\mathbf{\Gamma}_j$ related to the surfaces, and $\mathbf{n}$ to the outward normal vector.

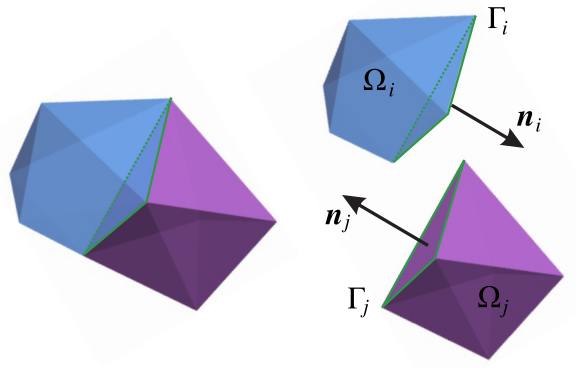


Figure 3: Interfaces.

The multidomain algorithm divides the matrices into blocks. The first corresponds to the elements where the boundary conditions are applied in the external surfaces defined as

$$\mathbf{A} = \left[\frac{2}{\Delta\tau^2}\mathbf{M} + \mathbf{H}_{bc}\right] \quad , \qquad \mathbf{B} = \mathbf{G}_{bc} \quad , \tag{38}$$

where $\mathbf{H}_{bc}$ and $\mathbf{G}_{bc}$ are the blocks after the exchange of columns when known displacement boundary conditions are imposed. For the internal elements that belong to the interfaces, the matrix is

$$\mathbf{F} = \left[\frac{2}{\Delta\tau^2}\mathbf{M} + \mathbf{H}_{I}\right] \quad , \tag{39}$$

and

$$\mathbf{u}_{\alpha} = \frac{1}{\Delta\tau^2}\left(5\mathbf{u}_{\tau} - 4\mathbf{u}_{\tau-\Delta\tau} + \mathbf{u}_{\tau-2\Delta\tau}\right) \quad . \tag{40}$$

For this illustrative case of two domains, Fig. 3, the final system of equations after the application of the boundary conditions is

$$\begin{aligned}\begin{bmatrix} \mathbf{A}_i & \mathbf{F}_i^j & -\mathbf{G}_i^j & 0 \\ 0 & \mathbf{F}_j^i & \mathbf{G}_j^i & \mathbf{A}_j \end{bmatrix} \begin{Bmatrix} \mathbf{x}_i \\ \mathbf{u}_i^j \\ \mathbf{t}_i^j \\ \mathbf{x}_j \end{Bmatrix}_{\tau+\Delta\tau} &= \begin{bmatrix} \mathbf{B}_i & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{B}_j \end{bmatrix} \begin{Bmatrix} \mathbf{k}_i^{bc} \\ 0 \\ 0 \\ \mathbf{k}_j^{bc} \end{Bmatrix}_{\tau+\Delta\tau} + \\ &+ \begin{bmatrix} \mathbf{M}_i & \mathbf{M}_i^j & 0 & 0 \\ 0 & \mathbf{M}_j^i & 0 & \mathbf{M}_j \end{bmatrix} \begin{Bmatrix} \mathbf{u}_{\alpha i} \\ \mathbf{u}_{\alpha i}^j \\ 0 \\ \mathbf{u}_{\alpha j} \end{Bmatrix} \quad , \end{aligned} \tag{41}$$

where $\mathbf{A}_i$, $\mathbf{B}_i$, and $\mathbf{M}_i$ belong to the boundary of the heterogeneous solid; $\mathbf{A}_i$ and $\mathbf{B}_i$ are the blocks corresponding to $\mathbf{H}_i$ and $\mathbf{G}_i$ where the load conditions are applied. The blocks $\mathbf{F}_i^j$ and $\mathbf{G}_i^j$ are the interfaces between ith and jth domains. Vectors $\mathbf{x}_i$ represent all the unknown traction and displacements to be evaluated in the elements corresponding to the boundaries and the vectors $\mathbf{k}_i^{bc}$ are the known boundary conditions applied. At the interfaces, displacement $\mathbf{u}_i^j$ and traction $\mathbf{t}_i^j$ are evaluated. The last three-step displacement responses are defined by $\mathbf{u}_{\alpha i}$, where $\mathbf{u}_{\alpha j}$ is for blocks belonging to the boundaries and $\mathbf{u}_{\alpha i}^j$ for blocks in the interfaces. A general algorithm to assemble the system of equation in multidomain applications can be found in [15, 19]. The final matrix equation is

$$\mathcal{A}\mathbf{x}_{\tau+\Delta\tau} = \mathcal{B}\mathbf{k}_{\tau+\Delta\tau}^{bc} + \mathcal{M}\mathbf{u}_{\alpha} \quad . \tag{42}$$

When elastostatic problems are treated, the mass terms are avoided and the system becomes time-independent, in that case $\mathbf{F} = \mathbf{H}_{bc}$ and $\mathbf{B} = \mathbf{G}_{bc}$. The final general matrix equation for elastostatic problems is

$$\mathcal{A}\mathbf{x} = \mathcal{B}\mathbf{k}^{bc} \quad , \tag{43}$$

if body forces $\mathbf{f}$ are acting on the solid

$$\mathcal{A}\mathbf{x} = \mathcal{B}\mathbf{k}^{bc} + \mathcal{M}\mathbf{f} \quad , \tag{44}$$

At this point, the BEM solution corresponds to the displacement and traction fields. The stress and strain tensor fields are evaluated following the procedure described by Kane [15] at each time or load step as a secondary response. Finally, to solve the large system of equations, a Multifrontal Massively Parallel Sparse-MUMPS [8, 9] direct solver is used. MUMPS implements a direct method based on a multifrontal approach, which performs a Gaussian factorization. This solver is configured for 64-bit general unsymmetrical real matrices using the out of core option. Some computational aspects are pointed out in [1, 2].

Mesh

The mesh generation is a primary task in the simulation process, and as expected, BESLE requires a specific input data file of the surface mesh. First, some definitions need to be introduced. The file contains the number of domains/regions `nreg`, the matrix of `POINTS`, matrix of `ELEMENTS` or connectivity, matrix of `NORMAL_VECTORS`, and the vector of elements per domain/region `EL_reg`. The matrix of `ELEMENTS` has three columns that identify each node of a triangle element with an integer number. Each node corresponds to a column in the matrix of `POINTS` which also has three columns with the x, y, and z coordinates in double precision. Further `nreg` and `EL_reg` are integer variables, and `NORMAL_VECTORS` is defined in double precision. The structure of the mesh file is easy to build using the following format

```
nreg
nPoints ! matrix of POINTS
x_1  y_1  z_1
x_2  y_2  z_2
x_3  y_3  z_3
.
.
nElements ! matrix of ELEMENTS
node_1_el_1   node_2_el_1   node_3_el_1
node_1_el_2   node_2_el_2   node_3_el_2
node_1_el_3   node_2_el_3   node_3_el_3
.
.
nElements  ! matrix of NORMAL_VECTORS
n_x_el_1   n_y_el_1   n_y_el_1
n_x_el_2   n_y_el_2   n_y_el_2
n_x_el_3   n_y_el_3   n_y_el_3
.
.
nreg   ! El_reg matrix
nElements_reg_1
nElements_reg_2
nElements_reg_3
.
.
nElements_reg_n
```

Table 1: Structure of the mesh file.

BESLE includes three ways for mesh generation, a `Fortran` sub-package based on 3ds MAX that also can define general meshes and boundary conditions for several cases. Further, a `C++/C` sub-package to artificially reproduce polycrystal aggregates. Finally, a basic `MATLAB` box generator is also provided. After the construction of the mesh, `*.obj` or `*.vtk` files are available to be checked using `ParaView`.

3.1 General

A BEM mesh and its boundary conditions could be generated using multimedia software like 3ds Max (Autodesk) or Blender [20,21]. The process has to be made in such a way that each "Object" matches a physical region/domain and the surfaces where the BCs are applied, should be generated using the "detach" tool in order to preserve the numerical precision of the mesh coordinates. The mesh and each set of elements where a specific BC will be applied should be individually exported into a file in `*.obj` format. The export options have to be set as follows: Faces=triangles, Target=PC/Win, Precision=5, Optimize Vertex = checked and everything else has to be non-checked. Special care should be taken with regards to numerical precision as the multimedia software uses real numbers of 4 bytes. Then, there is a limited number of available significant digits which means the further you get from origin and the more digits in front of the decimal point, the less precision you get. Furthermore, multimedia software tends to duplicate vertices and edges during the mesh modeling process which could generate errors. Therefore, it is recommended to use the tool "Weld", into the "Edit Poly" modifier, across the mesh modelling.

A `Fortran` sub-package to transform the input data and the BCs files from `*.obj` to `*.dat` is located in the `~/BESLE/Mesh/General` root, and the user's script `Set_parameters.f90` is in the `~/BESLE/Mesh/General/src/` folder. In order to compile the package, in the terminal

```
~/BESLE/Mesh/General$ make all
```

or just `make` should work. Before the compilation, the user's script must be configured. At any subsequent modification, the code is recompiled by

```
~/BESLE/Mesh/General$ make recompile
```

if there is something wrong during configuration, the next compilation is carried out using

```
~/BESLE/Mesh/General$ make fix
```

and, to execute

```
~/BESLE/Mesh/General$ ./Mesh
```

Finally, `make clean` to delete all the files and compilation. Now learning by examples, a case is prepared for a simulation of a vertebra under realistic boundary conditions. The data is saved in a `"*.dat"` file by default in the `Meshes/DAT/` folder, with `filename_out` defined by the user. As the code is implemented with double precision, `*.d0` or `*d0` should be added to the quantities. The necessary files to run the vertebra L2 and composite examples are located at `~/BESLE/Examples/Mesh/`, the user just need to replace them into the `~/BESLE/Mesh/`.

The vertebra mesh, shown in Fig. 4, was modeled as follows. Firstly, it was segmented from a fully anonymized CT of the 3DSlicer database[3] [22]. Thereafter, it was exported as `"*.stl"` and finally tuned into 3ds Max.

[3] https://www.slicer.org/wiki/CitingSlicer

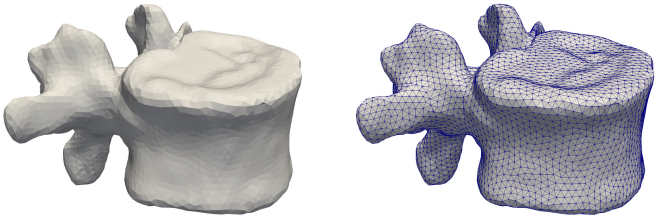


Figure 4: Vertebra L2: mesh of 21612 boundary elements.

The model of the vertebra L2 was divided into trabecular and cortical bones by segmenting the trabecular tissue from the CT and then, superimposing it with the vertebra model. The anisotropic properties of cortical bone need to be oriented regarding the normal direction of the bone surface. To approached this, the cortical domain was split into 15 new domains, which were used to evaluate a trending normal direction of each region and then, oriented its stiffness tensor. The 15 cortical and the trabecular domains are shown in Fig. 5.

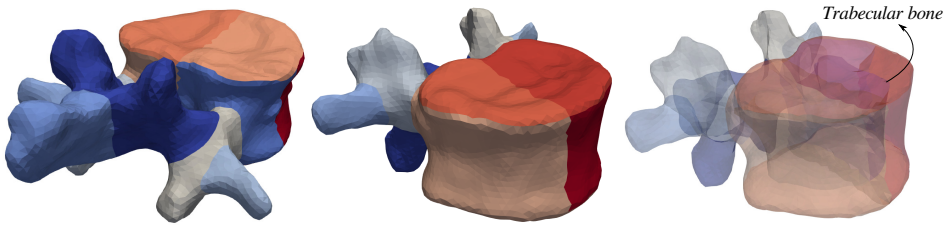


Figure 5: Vertebra L2: cortical (external) and trabecular (internal) domains.

As shown in Table 2, the idea is to obtain the data for the input mesh of the vertebra L2. The `filename_in` is defined by the user and corresponds to the `"*.obj"` file exported from 3ds Max.

```
1 SUBROUTINE Setup
2
3     filename_in = "Meshes/Obj/mesh.obj"
4     filename_out = "Input_data.dat"
5
6 END SUBROUTINE Setup
```

Table 2: Input data mesh of a vertebra L2.

This mesh generator includes the possibiliy to configure several options of boundary conditions using the same `Set_parameters.f90` script. From the individually `"BCs_i.obj"` file exported for each desired BC located in `BCSfilesPlace_in`, the new input data is saved in a `"BCs_i.dat"` file by default in the `BCs/BCs_DAT/` folder. To facilitate the use of external codes, the `"BCs_i.dat"` must have the following structure

```
nElements
Element_1 of the group
Element_2 of the group
Element_3 of the group
.
.
Element n of the group
TypeX        TypeY       TypeZ
valueX_1    valueY_1    valueZ_1
valueX_2    valueY_2    valueZ_2
valueX_3    valueY_3    valueZ_3
.
.
valueX_n    valueY_n    valueZ_n
```

Table 3: Structure of the boundary condition file.

The `nElements`, `Element_i of the group` are integer variables. The `Type` in each component is an integer value that depends on the boundary condition, it is `0` for displacement, and `1` for traction. If the user wants to apply a displacement in the normal direction to the surface `(TypeX, TypeY, TypeZ) = (2,0,0)`, and `(valueX_i, valueY_i, valueZ_i) = (u_n,0,0)` being u_n the magnitude of the normal displacement. The instruction to apply a traction in the normal direction to the surface `(TypeX, TypeY, TypeZ) = (3,0,0)`, and `(valueX_i, valueY_i, valueZ_i) = (t_n,0,0)` where t_n is the he magnitude of the normal traction in double precision.

In the first example in Table 4, the variable `AnalysisType` can take three reserved values `"elastostatic"`, `"quasi-elastostatic"`, and `"elastodynamic"`. This example defines two boundary conditions in `NumBCS` . Therefore, two new files are generated `"BCs_1.dat"` and `"BCs_2.dat"`.

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="elastostatic"
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%EParam%staticBCxVal=0.d0
    BC(1)%EParam%staticBCyVal=0.d0
    BC(1)%EParam%staticBCzVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="free"
    BC(2)%BCzType="free"
    BC(2)%EParam%staticBCxVal=100.d0
END SUBROUTINE Setup
```

Table 4: Elastostatic BCs independently applied on x, y and, z axes.

The `BC(i)%DirType` set as `"xyz"` indicates the boundary condition is applied independently in the x, y and, z axes. The BCs are defined by `BC(i)%BCxType`, `BC(i)%BCxType`, and `BC(i)%BCzType` respectively. The value of each of the boundary conditions is specified using `BC(i)%EParam%staticBCxVal`, `BC(i)%EParam%staticBCyVal`, and `BC(i)%EParam%staticBCzVal`. As observed in Table 4, the `"free"` are by default as traction zero and its value does not need to be defined. Additional examples are presented for different values of traction in Table 5(a), or mixing traction and displacement in the same group of BCs, Table 5(b).

(a)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="elastostatic"
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%EParam%staticBCxVal=0.d0
    BC(1)%EParam%staticBCyVal=0.d0
    BC(1)%EParam%staticBCzVal=0.d0

    BC(2)%BCxType="traction"
    BC(2)%BCyType="traction"
    BC(2)%BCzType="traction"
    BC(2)%EParam%staticBCxVal=100.d0
    BC(2)%EParam%staticBCyVal=200.d0
    BC(2)%EParam%staticBCzVal=300.d0

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="elastostatic"
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%EParam%staticBCxVal=0.d0
    BC(1)%EParam%staticBCyVal=0.d0
    BC(1)%EParam%staticBCzVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="displacement"
    BC(2)%BCzType="displacement"
    BC(2)%EParam%staticBCxVal=100.d0
    BC(2)%EParam%staticBCyVal=1.d0
    BC(2)%EParam%staticBCzVal=1.d0

END SUBROUTINE Setup
```

Table 5: Additional elastostatic cases, (a) different values of traction in each direction and (b) mixing traction and displacement in the same group of BCs.

Now, some configurations are given for the case of `"quasi-elastostatic"`, Table 6. The difference here is the number of load steps specified by the `BC(i)%Nsteps` variable. In this case, new variables appear `BC(i)%FuncxType`, `BC(i)%FuncyType`, and `BC(i)%FunczType`. The BCs will follow a `"linear"` function (ramp) throughout the simulation steps starting from a minimum value of zero to a maximum value define by `BC(1)%LParam%LinearBCxMaxVal`, `BC(1)%LParam%LinearBCyMaxVal`, and `BC(1)%LParam%LinearBCzMaxVal` in each direction. Again for this case, the `"free"` condition is defined by default in Table 6, and different values of traction mixing traction and displacement in the same group of BCs, Table 6(b).

(a)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="free"
    BC(2)%BCzType="free"
    BC(2)%FuncxType="linear"
    BC(2)%LParam%LinearBCxMaxVal=100.d0

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="traction"
    BC(2)%BCzType="displacement"
    BC(2)%FuncxType="linear"
    BC(2)%FuncyType="linear"
    BC(2)%FunczType="linear"
    BC(2)%LParam%LinearBCxMaxVal=100.d0
    BC(2)%LParam%LinearBCyMaxVal=100.d0
    BC(2)%LParam%LinearBCzMaxVal=1.d0

END SUBROUTINE Setup
```

Table 6: Quasi-elastostatic cases, (a) traction and free surface conditions and (b) mixing traction and displacement in the same group of BCs.

The next example in Table 7(a) illustrates how to apply BCs that follow a `"quadratic"` function in each axis as

$$\begin{aligned} Ax^2 + Bx + C &\rightarrow x \in (SOx, Sfx) \ , \\ Ay^2 + By + C &\rightarrow y \in (SOy, Sfy) \ , \\ Az^2 + Bz + C &\rightarrow z \in (SOz, Sfz) \ , \end{aligned} \tag{45}$$

where the coefficients are defined using the `BC(i)%QParam%QuadAj`, `BC(i)%QParam%QuadBj`, and `BC(i)%QParam%QuadCj` variables with j being x, y or z. Similarly, for the space interval (SOj, Sfj) the variables `BC(i)%QParam%QuadSOj`, and `BC(i)%QParam%QuadSfj` are employed with j being x, y or z. Then, the example in Table 7(a) shows that multiple conditions are combined, such as a boundary condition of traction quadratically applied in the x and y directions, and a displacement linearly applied in the z direction.

(a)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="traction"
    BC(2)%BCzType="displacement"
    BC(2)%FuncxType="quadratic"
    BC(2)%FuncyType="quadratic"
    BC(2)%FunczType="linear"
    BC(2)%QParam%QuadSOx=0.d0
    BC(2)%QParam%QuadSfx=10.d0
    BC(2)%QParam%QuadAx=1.d0
    BC(2)%QParam%QuadBx=0.d0
    BC(2)%QParam%QuadCx=0.d0
    BC(2)%QParam%QuadSOy=0.d0
    BC(2)%QParam%QuadSfy=10.d0
    BC(2)%QParam%QuadAy=1.d0
    BC(2)%QParam%QuadBy=0.d0
    BC(2)%QParam%QuadCy=0.d0
    BC(2)%LParam%LinearBCzMaxVal=1.d0

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="traction"
    BC(2)%BCzType="displacement"
    BC(2)%FuncxType="sine"
    BC(2)%FuncyType="sine"
    BC(2)%FunczType="linear"
    BC(2)%SParam%SineSOx=0.d0
    BC(2)%SParam%SineSfx=2.d0*pi
    BC(2)%SParam%SineAx=100.d0
    BC(2)%SParam%SineBx=1.d0
    BC(2)%SParam%SineCx=0.d0
    BC(2)%SParam%SineSOy=0.d0
    BC(2)%SParam%SineSfy=2.d0*pi
    BC(2)%SParam%SineAy=100.d0
    BC(2)%SParam%SineBy=1.d0
    BC(2)%SParam%SineCy=0.d0
    BC(2)%LParam%LinearBCzMaxVal=1

END SUBROUTINE Setup
```

Table 7: Quasi-elastostatic cases, (a) using a linear and a quadractic function, and (b) using a linear and a sinusoidal function.

An additional default function, the application of a sinusoidal load is shown in Table 7(b). The example illustrates how to apply BCs that follow a `"Sine"` function in each axis as

$$\begin{aligned} A\sin(B\theta_x + C) &\rightarrow \theta_x \in (SOx, Sfx) \ , \\ A\sin(B\theta_y + C) &\rightarrow \theta_y \in (SOy, Sfy) \ , \\ A\sin(B\theta_z + C) &\rightarrow \theta_z \in (SOz, Sfz) \ . \end{aligned} \tag{46}$$

Similarly as used for `"quadratic"` functions, in the `"sine"` case, the coefficients are defined using the `BC(i)%QParam%SineAj`, `BC(i)%QParam%SineBj`, and `BC(i)%QParam%SineCj` with j being x, y or z. For the space interval (SOj, Sfj) the variables `BC(i)%QParam%SineSOj`, and `BC(i)%QParam%SineSfj` are employed with j being x, y or z.

Then, the example in Table 7(b) also shows multiple conditions being imposed, such as a traction applied using a sine function, in the x and y directions, and a displacement linearly applied in the z direction. For illustrative purposes, some additional possibilities are shown in Table 8.

(a)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="traction"
    BC(2)%BCzType="displacement"
    BC(2)%FuncxType="sine"
    BC(2)%FuncyType="quadratic"
    BC(2)%FunczType="linear"
    BC(2)%SParam%SineSOx=0.d0
    BC(2)%SParam%SineSfx=2.d0*pi
    BC(2)%SParam%SineAx=100.d0
    BC(2)%SParam%SineBx=1.d0
    BC(2)%SParam%SineCx=0.d0
    BC(2)%QParam%QuadSOy=0.d0
    BC(2)%QParam%QuadSfy=10.d0
    BC(2)%QParam%QuadAy=1.d0
    BC(2)%QParam%QuadBy=0.d0
    BC(2)%QParam%QuadCy=0.d0
    BC(2)%LParam%LinearBCzMaxVal=1

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 3

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="free"
    BC(2)%BCzType="free"
    BC(2)%FuncxType="sine"
    BC(2)%SParam%SineSOx=0.d0
    BC(2)%SParam%SineSfx=2.d0*pi
    BC(2)%SParam%SineAx=100.d0
    BC(2)%SParam%SineBx=1.d0
    BC(2)%SParam%SineCx=0.d0

    BC(3)%DirType="xyz"
    BC(3)%BCxType="free"
    BC(3)%BCyType="traction"
    BC(3)%BCzType="free"
    BC(3)%FuncyType="sine"
    BC(3)%SParam%SineSOy=0.d0
    BC(3)%SParam%SineSfy=2.d0*pi
    BC(3)%SParam%SineAy=100.d0
    BC(3)%SParam%SineBy=1.d0
    BC(3)%SParam%SineCy=0.d0

END SUBROUTINE Setup
```

Table 8: Additional quasi-elastostatic cases, (a) mixing linear, quatratic, and sinusoidal functions with traction and displacements; and (b) three groups of boundary conditions.

There are situations where it is necessary to apply BCs in the normal direction to the surface. An `"elastostatic"` case is shown in Table 9(a), and configured by the `BC(i)%DirType` variable as `"normal"`. The type of boundary condition can be `"traction"` or `"displacement"` defined by the `BC(i)%BCnType` variable. Only one value for this condition is required in

`BC(i)%EParam%staticBCnVal` variable.

(a)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="elastostatic"
    NumBCS = 3

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%EParam%staticBCxVal=0.d0
    BC(1)%EParam%staticBCyVal=0.d0
    BC(1)%EParam%staticBCzVal=0.d0

    BC(2)%DirType="normal"
    BC(2)%BCnType="traction"
    BC(2)%EParam%staticBCnVal=100.d0

    BC(3)%DirType="normal"
    BC(3)%BCnType="displacement"
    BC(3)%EParam%staticBCnVal=1.d0

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 5

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="normal"
    BC(2)%BCnType="traction"
    BC(2)%FuncnType="sine"
    BC(2)%SParam%SineSOn=0.d0
    BC(2)%SParam%SineSfn=2.d0*pi
    BC(2)%SParam%SineAn=100.d0
    BC(2)%SParam%SineBn=1.d0
    BC(2)%SParam%SineCn=0.d0

    BC(3)%DirType="normal"
    BC(3)%BCnType="displacement"
    BC(3)%FuncnType="sine"
    BC(3)%SParam%SineSOn=0.d0
    BC(3)%SParam%SineSfn=2.d0*pi
    BC(3)%SParam%SineAn=1.d0
    BC(3)%SParam%SineBn=1.d0
    BC(3)%SParam%SineCn=0.d0

    BC(4)%DirType="normal"
    BC(4)%BCnType="traction"
    BC(4)%FuncnType="linear"
    BC(4)%LParam%LinearBCnMaxVal=100.d0

    BC(5)%DirType="normal"
    BC(5)%BCnType="traction"
    BC(5)%FuncnType="quadratic"
    BC(5)%QParam%QuadSOn=0.d0
    BC(5)%QParam%QuadSfn=10.d0
    BC(5)%QParam%QuadAn=1.d0
    BC(5)%QParam%QuadBn=0.d0
    BC(5)%QParam%QuadCn=0.d0

END SUBROUTINE Setup
```

Table 9: Additional quasi-elastostatic cases, (a) mixing linear, quatratic, and sinusoidal functions with traction and displacements; and (b) three groups of boundary conditions.

For `"elastostatic"` BCs, the `"normal"` case also can follow specific functions as explained

for the `"xyz"` definition. As appreciated in Table 9(b), the only difference is the sintaxis that is for the normal direction.

An additional `BC(i)%DirType` is introduced in Table 10(a), as `"load"`. It corresponds to a force in Newton units with `BC(i)%EParam%staticBClVal` magnitude and direction as specified for the `"elastostatic"` case by the `BC(i)%LoadDirection(j)` vector, where $j = 1, 2, 3$.

(a)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="elastostatic"
    NumBCS = 3

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%EParam%staticBCxVal=0.d0
    BC(1)%EParam%staticBCyVal=0.d0
    BC(1)%EParam%staticBCzVal=0.d0

    BC(2)%DirType="normal"
    BC(2)%BCnType="traction"
    BC(2)%EParam%staticBCnVal=100.d0

    BC(3)%DirType="load"
    BC(3)%EParam%staticBClVal=1200.d0
    BC(3)%LoadDirection(1)=1.d0
    BC(3)%LoadDirection(2)=1.d0
    BC(3)%LoadDirection(3)=0.d0

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 4

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="load"
    BC(2)%FunclType="sine"
    BC(2)%SParam%SineSOl=0.d0
    BC(2)%SParam%SineSfl=2.d0*pi
    BC(2)%SParam%SineAl=1200.d0
    BC(2)%SParam%SineBl=1.d0
    BC(2)%SParam%SineCl=0.d0
    BC(2)%LoadDirection(1)=1.d0
    BC(2)%LoadDirection(2)=1.d0
    BC(2)%LoadDirection(3)=0.d0

    BC(3)%DirType="load"
    BC(3)%FunclType="quadratic"
    BC(3)%QParam%QuadSfl=34.d0
    BC(3)%QParam%QuadSOl=0.d0
    BC(3)%QParam%QuadAl=1.d0
    BC(3)%QParam%QuadBl=1.d0
    BC(3)%QParam%SineCl=0.d0
    BC(3)%LoadDirection(1)=1.d0
    BC(3)%LoadDirection(2)=1.d0
    BC(3)%LoadDirection(3)=0.d0

    BC(4)%DirType="load"
    BC(4)%FunclType="linear"
    BC(4)%LParam%LinearBClMaxVal=1200
    BC(4)%LoadDirection(1)=1.d0
    BC(4)%LoadDirection(2)=1.d0
    BC(4)%LoadDirection(3)=0.d0
END SUBROUTINE Setup
```

Table 10: Examples including the load direction type, (a) elastostatic and (b) quasi-elastostatic.

Both examples in Table 10 show how to combine the different types and directions of boundary conditions for "elastostatic" and "quasi-elastostatic" simulations. It is worth noting that in the last case, the "load" condition also works with the different BC(i)%FunclType previously defined. The remaining default case of BC(i)%FunclType is the "heaviside" boundary condition, Table 11(a). It also works for all the BC(i)%DirType options. This kind of condition is usually applied in "elastodynamic" simulations.

(a)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="elastodynamic"
    Nsteps = 50
    NumBCS = 3

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="displacement"
    BC(2)%BCyType="free"
    BC(2)%BCzType="traction"
    BC(2)%FuncxType="heaviside"
    BC(2)%FunczType="heaviside"
    BC(2)%HParam%HeaviBCxVal=1.d0
    BC(2)%HParam%HeaviBCyVal=100.d0

    BC(3)%DirType="normal"
    BC(3)%BCnType="displacement"
    BC(3)%FuncnType="heaviside"
    BC(3)%HParam%HeaviBCnVal=1.d0

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup
    filename_in = "Meshes/Obj/mesh.obj"
    filename_out = "Input_data.dat"

    AnalysisType="quasi-elastostatic"
    Nsteps = 50
    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0.d0
    BC(1)%LParam%LinearBCyMaxVal=0.d0
    BC(1)%LParam%LinearBCzMaxVal=0.d0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="displacement"
    BC(2)%BCyType="displacement"
    BC(2)%BCzType="displacement"
    BC(2)%FuncxType="custom_1"
    BC(2)%FuncyType="custom_2"
    BC(2)%FunczType="custom_3"

END SUBROUTINE Setup
```

Table 11: Examples, (a) Elastodynamic step function and (b) custom functions.

All the "quasi-elastostatic" scripts can be used for "elastodynamic" simulations with the same sintaxis. As observed, some default functions were included such as linear, quadratic, and sinusoidal. In order to turn this mesh and boundary conditions generator more general, three custom functions were incorporated in the Custom_function.f90 module in the ~/BESLE/Mesh/General/src/ folder. The user is free to add or modifify any custom function to be applied as shown in Table 11(b). The "custom_1" function is the cubic equation, the "custom_2" represents an harmonic piece-wise function, and "custom_3" is an exponential piece-wise function.

In order to clarify the concept and the way of generating mesh and BCs to be simulated using BESLE, the user is encouraged to run the illustrative examples. The first is a matrix-fiber composite shown in Figure 6. This mesh is composed of 57 domains, 18 fibres divided into 54 Γ_i^f sub-fibres, and 3 additional Ω_i^M domains for the matrix for convenience. The BCs are prepared for a `"quasi-elastostatic"` simulation of 50 steps where variable s is the load step.

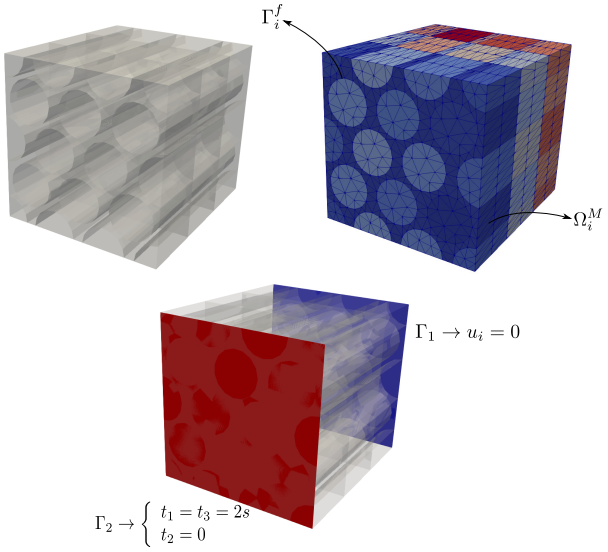


Figure 6: Composite, mesh of 8640 elements and boundary conditions.

The BCs to be applied to the vertebra L2 was set assuming that it supported an upper body weight of 54 kg, and the selection of the BCs elements was based on the loading described by Jong *et al.* [23]. Figure 7 shows the fixed BC (Γ_1), the vertebral body BC (Γ_2) and the articular processes BCs (Γ_3 and Γ_4). The BCs were prepared for a `"quasi-elastostatic"` simulation of 4 steps where the variable θ varies $\pi/2$ at each step load.

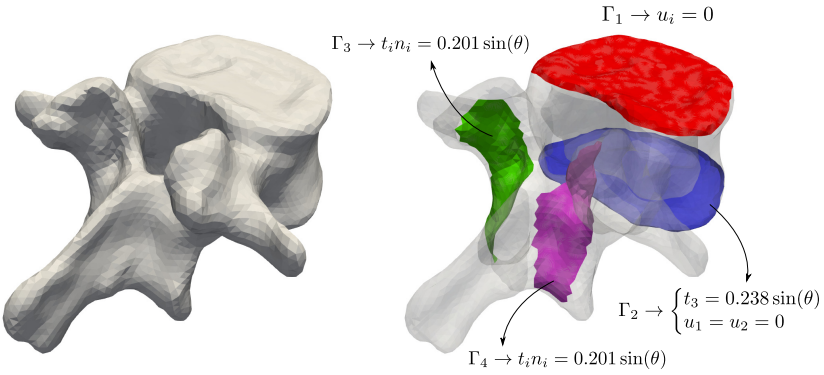


Figure 7: Vertebra L2, mesh of 21612 elements and boundary conditions.

Finally, it is reiterated that the necessary files to run these examples are located at `~/BESLE/Examples/Mesh/`, the user just needs to replace them into the `~/BESLE/Mesh/General/`.

3.2 Polycrystal aggregates

Polycrystalline materials are represented by the aggregate of several crystals or grains with proper crystalline orientations and planes. At the microscale, each grain is usually modeled as linear elastic anisotropic material as presented in the majority of metallic and ceramic crystalline materials. Artificial polycrystalline morphologies are available to be generated and discretised using the sub-package `Polycrystal` located at `~/BESLE/Mesh/` root.

This mesh generator is a `C++/C` multi-compiler code based on the algorithm and procedure presented in [1, 2, 24]. The data mesh is generated by default in the `Mesh.dat` file at the `~/BESLE/Mesh/Polycrystal/Export/` folder, and the user's script `Set_parameters.f90` is in the `~/BESLE/Mesh/Polycrystal/src/` folder. In order to compile the package, in the terminal

```
~/BESLE/Mesh/Polycrystal$ make all
```

or just `make` should work. Before the compilation, the user's script must be configured. At any subsequent modification, the code is recompiled by

```
~/BESLE/Mesh/Polycrystal$ make recompile
```

if there is something wrong during configuration, the next compilation is carried out using

```
~/BESLE/Mesh/Polycrystal$ make fix
```

and, to execute

```
~/BESLE/Mesh/Polycrystal$ ./Structure && ./Mesh
```

Finally, `make clean` to delete all the files and compilation. Now learning with an example, one polycrystalline structure is prepared for simulation. The necessary files to run the polycrystal example are located at `~/BESLE/Examples/Mesh/`, the user just needs to replace them into the `~/BESLE/Mesh/Polycrystal/`. The script shown in Table 12 requires the input of three integer variables to define the number of grains in each direction of the box, hence, the total number of grains is given by `ngrains_x*ngrains_y*ngrains_z`. The box size is defined by three double variables `x_max`, `y_max`, and `z_max`.

```
void Setup()
{

    // Grains in each coordinate
    ngrains_x=3;
    ngrains_y=4;
    ngrains_z=8;

    // Box dimensions
    x_max = 15.0;
    y_max = 15.0;
    z_max = 45.0;

    // Random geometry
    stochastic = 0;

    // Mesh density parameter
    dm = 1;
}
```

Table 12: Script for polycrystal aggregates.

Further, two additional variables are set as `stochastic` and `dm`. The first is to build random polycrystalline structures if its value is `1`, Alternatively, the value `0` guarantees that the same structure is generated at every compilation. The other integer parameter `dm` is the mesh density and its lower value is `1`. Figure 8 presents a polycrystalline material constituted by 96 grains in a box with the dimensions specified by the script and the level `1` of discretization.

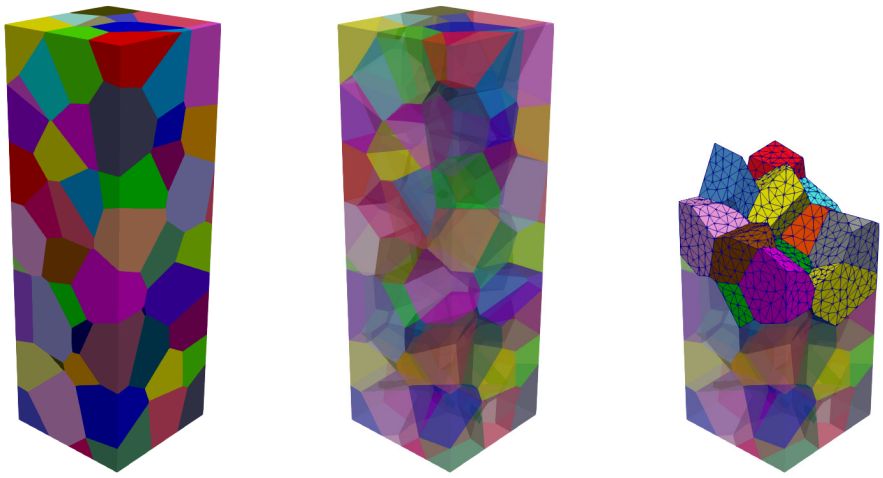
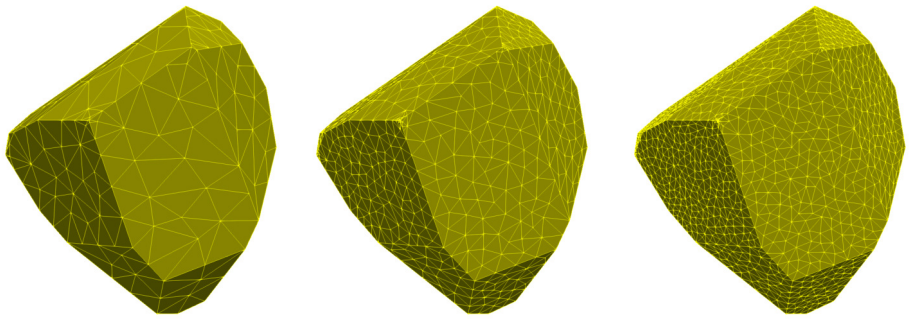


Figure 8: Polycrystal aggregate, 96 grains and 22767 boundary elements.

In order to illustrate the effect of the `dm` parameter, in Fig. 9, three cases are shown for different values of `dm`. It can be appreciated how the refinement increases with the increment of `dm`.



(a) (b) (c)

Figure 9: Mesh density parameter, (a) $d_m = 1$, 399 elements; (b) $d_m = 2$, 1478 elements; and (c) $d_m = 3$, 3250 elements.

Finally, the mesh can be checked in `ParaView` with the `Mesh.vtk` file. The boundary conditions to analyse the mechanical behavior of these materials are imposed directly by BESLE in the user's script.

3.3 Box

This is a basic `MATLAB` mesh generator for box geometries that can be composed by one or multiple domains. It is located at the `~/BESLE/Mesh/` root, and the data mesh is generated by default in a `*.dat` file at the `~/BESLE/Mesh/box/` folder with the `file` name, Table 13. The necessary files to run the this body force example are located at `~/BESLE/Examples/Mesh/`, the user just need to replace them into the `~/BESLE/Mesh/Box/`. The user's script containing the sintaxis is named `Mesh_Generator.m`.

(a)

```
file = 'Input_data';

a = 35; % Dimension in x
b = 35; % Dimension in y
c = 60; % Dimension in z

nreg_a = 1; % Regions in x
nreg_b = 1; % Regions in y
nreg_c = 1; % Regions in z

nel_a = 6; % Elements in x / region
nel_b = 4; % Elements in y / region
nel_c = 12;% Elements in z / region
```

(b)

```
file = 'Input_data';

a = 35; % Dimension in x
b = 35; % Dimension in y
c = 60; % Dimension in z

nreg_a = 3; % Regions in x
nreg_b = 2; % Regions in y
nreg_c = 6; % Regions in z

nel_a = 2; % Elements in x / region
nel_b = 2; % Elements in y / region
nel_c = 2; % Elements in z / region
```

Table 13: Box meshes, (a) single domain and (b) 36 domains.

The first example in Table 13(a), is for one domain box as `nreg_i=1` with the number of elements in each direction defined by `nel_i`. The other case shown in Table 13(b) generates a

box split into unit cells following the `nreg_i` variable, each one with the number of elements indicated by `nel_i`. Both examples are illustrated in the following figure

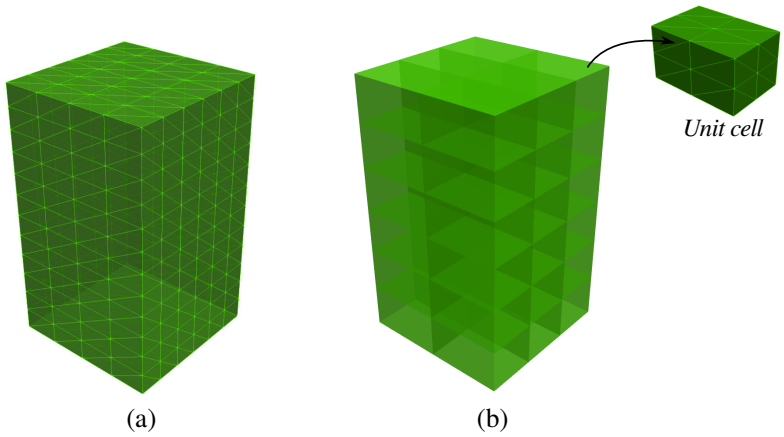


Figure 10: Box mesh, (a) single domain, 576 elements; and (b) 36 domains, 1728 elements.

Finally, the mesh can be checked in `ParaView` with the `"*.vtk"` file. The boundary conditions to analyse the mechanical behavior of materials with this shape are especified directly by BESLE in the user's script.

Material

Several options for modelling elastic materials are implemented in BESLE, it is possible to consider isotropic or anisotropic materials in specimens with only one domain. Furthermore, for heterogeneous materials, useful miscellaneous lists were included to facilitate the modelling of specimens with a large number of domains. First, a background of the fundamental solutions is presented, and then, the computational aspects and guidance for running the external sub-package of materials is shown with nine illustrative examples.

4.1 Fundamental solutions

The BEM formulation incorporates the elastic material using the fundamental solutions which, in BESLE, are based on double Fourier series as developed in [25]. The displacement fundamental solution $\mathbf{U}(\mathbf{x}', \mathbf{x})$ is considered as the response in x_i direction at the field point $\mathbf{x}$ owing to the perturbation produced by a unit load in the x_j direction at the source point $\mathbf{x}'$ in a homogeneous infinite body. Several efforts were made to produce an explicit form to evaluate $\mathbf{U}(\mathbf{x}', \mathbf{x})$, such as the explicit form to compute the Green's function proposed by Ting and Lee [26]. The authors presented the Green's function in terms of the Barnett-Lothe tensor [27, 28] $\mathbf{H}(\theta, \phi)$ as follows

$$\mathbf{U}(r, \theta, \phi) = \frac{1}{4\pi r}\mathbf{H}(\theta, \phi) \quad , \tag{47}$$

where, the Bartnett-Lothe tensor, $\mathbf{H}(\theta, \phi)$ in terms of Stroh's eigenvalues p_i, are given by

$$\mathbf{H}(\theta, \phi) = \frac{1}{|\boldsymbol{\kappa}|}\sum_{n=0}^{4} q_n \hat{\mathbf{\Gamma}}^{(n)} \quad , \tag{48}$$

and q_n is defined by

$$q_n = \begin{cases} \dfrac{-1}{2\beta_1\beta_2\beta_3}\left[\Re\left\{\displaystyle\sum_{t=1}^{3}\frac{p_t^n}{(p_t - \bar{p}_{t+1})(p_t - \bar{p}_{t+2})}\right\} - \delta_{n2}\right] & \text{for } n = 0, 1, 2 \\ \dfrac{-1}{2\beta_1\beta_2\beta_3}\left[\Re\left\{\displaystyle\sum_{t=1}^{3}\frac{p_t^{n-2}\bar{p}_{t+1}\bar{p}_{t+2}}{(p_t - \bar{p}_{t+1})(p_t - \bar{p}_{t+2})}\right\}\right] & \text{for } n = 3, 4 \end{cases} , \tag{49}$$

The Barnett-Lothe tensor $\mathbf{H}(\theta, \phi)$ depends only on the spherical angles (θ, ϕ), and it is a contour integral around a unit circle $|\mathbf{n}^*|$ on an oblique plane at field point $\mathbf{x}$. The unit vector $|\mathbf{n}^*|$ on the oblique plane, the two mutually orthogonal vectors $\mathbf{n}$ and $\mathbf{m}$ are

$$\begin{array}{l} |\mathbf{n}^*| = \mathbf{n}\cos\psi + \mathbf{m}\sin\psi \\ \mathbf{n} = (\cos\phi\cos\theta, \cos\phi\sin\theta, -\sin\phi) \\ \mathbf{m} = (-\sin\theta, \cos\theta, 0) \end{array} \left\{ \begin{array}{l} \psi \text{ Arbitrary} \\ 0 \leq \theta < 2\pi \\ 0 \leq \phi \leq \pi \end{array} \right. . \tag{50}$$

The $[\mathbf{n}, \mathbf{m}, \mathbf{n}_r]$ form the right-handed triad, where $\mathbf{n}_r = \mathbf{x}/r$ and ψ is an arbitrary parameter. The fourth-order tensor $\hat{\mathbf{\Gamma}}^{(n)}$ is the adjoint of the matrix $\mathbf{\Gamma}(p)$ defined as

$$\hat{\Gamma}^{(n)}_{ij} = \tilde{\Gamma}^{(n)}_{(i+1)(j+1)(i+2)(j+2)} - \tilde{\Gamma}^{(n)}_{(i+1)(j+2)(i+2)(j+1)}, \ \ (i,j = 1,2,3) \ \ , \tag{51}$$

$$\mathbf{\Gamma}(p) = \mathbf{Q} + p\mathbf{V} + p^2\boldsymbol{\kappa} \ \ , \tag{52}$$

where

$$\mathbf{V} = (\mathbf{R} + \mathbf{R}^T) \ \ , \tag{53}$$

and

$$\mathbf{Q} \equiv Q_{ik} = C_{ijks} n_j n_s, \quad \mathbf{R} \equiv R_{ik} = C_{ijks} n_j m_s \ \ , \tag{54}$$

the remaining term of (48), $\boldsymbol{\kappa}$ is given by

$$\boldsymbol{\kappa} \equiv \kappa_{ik} = C_{ijks} m_j m_s \ \ . \tag{55}$$

In Eq. (49), the Stroh's eigenvalues are the roots of the sextic equation obtained by the determinant $|\mathbf{\Gamma}(p)| = 0$, composed by three pairs of complex conjugates, written as

$$p_u = \alpha_v + \mathbf{i}\beta_v, \ \beta_v > 0, \ \ (v = 1,2,3) \ \ . \tag{56}$$

Finally, the tensor $\hat{\mathbf{\Gamma}}^{(n)}$ is expressed in a reduced form after some algebraic manipulation [29] as

$$\begin{array}{l} \tilde{\Gamma}^{(4)}_{pqrs} = \kappa_{pq}\kappa_{rs} \ \ , \\ \tilde{\Gamma}^{(3)}_{pqrs} = V_{pq}\kappa_{rs} + \kappa_{pq}V_{rs} \ \ , \\ \tilde{\Gamma}^{(2)}_{pqrs} = \kappa_{pq}Q_{rs} + \kappa_{rs}Q_{pq} + V_{pq}V_{rs} \ \ , \\ \tilde{\Gamma}^{(1)}_{pqrs} = \kappa_{pq}Q_{rs} + \kappa_{rs}Q_{pq} \ \ , \\ \tilde{\Gamma}^{(0)}_{pqrs} = Q_{pq}Q_{rs} \ \ . \end{array} \tag{57}$$

In order to obtain the displacement fundamental solution and its derivatives, the Bartnett-Lothe tensor $\mathbf{H}(\theta, \phi)$ is expressed in terms of double Fourier series in the spherical coordinate system, and it is given by

$$H_{uv}(\theta, \phi) = \sum_{m=-\alpha}^{\alpha} \sum_{n=-\alpha}^{\alpha} \lambda^{(m,n)}_{uv} e^{\mathbf{i}(m\theta + n\phi)} \ (u, v = 1,2,3) \ \ , \tag{58}$$

where, α must be sufficiently large for the convergence of the series. The unknown Fourier coefficients $\lambda^{(m,n)}_{uv}$ are evaluated by

$$\lambda_{uv}^{(m,n)} = \frac{1}{4\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} H_{uv}\left(\theta,\phi\right)e^{-\mathbf{i}(m\theta+n\phi)}d\theta d\phi \quad , \tag{59}$$

in Eq. (59) using the Euler's formula, $\lambda_{uv}^{(m,n)}$ and $\lambda_{uv}^{(-m,-n)}$ are complex conjugates, thus

$$\lambda_{uv}^{(m,n)} = \bar{\lambda}_{uv}^{(-m,-n)}. \tag{60}$$

The Fourier coefficients $\lambda_{uv}^{(m,n)}$ can be separated into the real part $R_{uv}^{(m,n)}$ and the imaginary part $I_{uv}^{(m,n)}$ as

$$\lambda_{uv}^{(m,n)} = R_{uv}^{(m,n)} + \mathbf{i}I_{uv}^{(m,n)} \quad , \tag{61}$$

substituting Eq. (60) into Eq. (58) and using the Euler's formula, gives

$$H_{uv}\left(\theta,\phi\right) = \sum_{m=-\alpha}^{\alpha}\sum_{n=-\alpha}^{\alpha} h_{uv}^{(m,n)}\left(\theta,\phi\right) \quad (u,v=1,2,3) \quad , \tag{62}$$

where

$$h_{uv}^{(m,n)}\left(\theta,\phi\right) = R_{uv}^{(m,n)}\cos\left(m\theta+n\phi\right) - I_{uv}^{(m,n)}\sin\left(m\theta+n\phi\right) \quad , \tag{63}$$

using the condition in Eq. (60), the Eq. (62) can be simplified as

$$\begin{aligned} H_{uv}\left(\theta,\phi\right) = 2\sum_{m=1}^{\alpha}\left\{\sum_{n=1}^{\alpha} h_{uv}^{(m,n)}\left(\theta,\phi\right) + \sum_{m=-\alpha}^{-1} h_{uv}^{(m,n)}\left(\theta,\phi\right)\right\} \\ +2\sum_{n=1}^{\alpha} h_{uv}^{(0,n)}\left(\theta,\phi\right) + 2\sum_{m=1}^{\alpha} h_{uv}^{(m,0)}\left(\theta,\phi\right) + R_{uv}^{(0,0)} \quad , \end{aligned} \tag{64}$$

substituting Eq. (63) into Eq. (64) and finally into Eq. (47), the fundamental solution $\mathbf{U}$ is expressed by

$$U_{uv}\left(r,\theta,\phi\right) = \frac{1}{2\pi r}\left\{\begin{array}{l} \displaystyle\sum_{m=1}^{\alpha}\sum_{n=1}^{\alpha}\left[\begin{array}{l}\left(\tilde{R}_{uv}^{(m,n)}\cos m\theta - \tilde{I}_{uv}^{(m,n)}\sin m\theta\right)\cos n\phi \\ -\left(\hat{R}_{uv}^{(m,n)}\sin m\theta - \hat{I}_{uv}^{(m,n)}\cos m\theta\right)\sin n\phi\end{array}\right] \\ \displaystyle+\sum_{m=1}^{\alpha}\left(\begin{array}{l} R_{uv}^{(0,m)}\cos m\phi - I_{uv}^{(0,m)}\sin m\phi \\ +R_{uv}^{(m,0)}\cos m\theta - I_{uv}^{(m,0)}\sin m\theta\end{array}\right) + \frac{R_{uv}^{(0,0)}}{2}\end{array}\right\} \quad , \tag{65}$$

the components $\tilde{R}_{uv}^{(m,n)}$, $\hat{R}_{uv}^{(m,n)}$, $\tilde{I}_{uv}^{(m,n)}$ and $\hat{I}_{uv}^{(m,n)}$ are

$$\begin{aligned} \tilde{R}_{uv}^{(m,n)} = R_{uv}^{(m,n)} + R_{uv}^{(m,-n)}, \quad \hat{R}_{uv}^{(m,n)} = R_{uv}^{(m,n)} - R_{uv}^{(m,-n)}, \\ \tilde{I}_{uv}^{(m,n)} = I_{uv}^{(m,n)} + I_{uv}^{(m,-n)}, \quad \hat{I}_{uv}^{(m,n)} = I_{uv}^{(m,n)} - I_{uv}^{(m,-n)} \quad . \end{aligned} \tag{66}$$

The first order derivative of the displacement fundamental solution $\mathbf{U}'$ is evaluated using the chain rule

$$\mathbf{U}' \equiv U_{uv,l} = \frac{\partial U_{uv}}{\partial r}\frac{\partial r}{\partial x_l} + \frac{\partial U_{uv}}{\partial \theta}\frac{\partial \theta}{\partial x_l} + \frac{\partial U_{uv}}{\partial \phi}\frac{\partial \phi}{\partial x_l} \quad . \tag{67}$$

Applying direct partial differentiation of Eq. (58) with respect to the spherical coordinates using the Eq. (67), gives

$$U_{uv,l}\left(r,\theta,\phi\right) = \frac{1}{4\pi r^2}\left\{\begin{array}{l} \displaystyle\sum_{m=-\alpha}^{\alpha}\sum_{n=-\alpha}^{\alpha} \lambda_{uv}^{(m,n)} e^{\mathbf{i}(m\theta+n\phi)} \left[\begin{array}{c} -\cos\theta(\sin\phi - \mathbf{i}n\cos\phi) \\ -\mathbf{i}m\sin\theta/\sin\phi \end{array}\right] \\ \displaystyle\sum_{m=-\alpha}^{\alpha}\sum_{n=-\alpha}^{\alpha} \lambda_{uv}^{(m,n)} e^{\mathbf{i}(m\theta+n\phi)} \left[\begin{array}{c} -\sin\theta(\sin\phi - \mathbf{i}n\cos\phi) \\ +\mathbf{i}m\cos\theta/\sin\phi \end{array}\right] \\ \displaystyle\sum_{m=-\alpha}^{\alpha}\sum_{n=-\alpha}^{\alpha} \lambda_{uv}^{(m,n)} e^{\mathbf{i}(m\theta+n\phi)} \left[-(\cos\phi + \mathbf{i}n\sin\phi)\right] \end{array}\right\} \quad . \tag{68}$$

Following a procedure like the one used to obtain the Eq. (65), the expression for the derivative of the displacement fundamental solution is

$$U_{uv,l} = \frac{1}{2\pi r^2}\left\{\begin{array}{l} -\omega_l(\theta,\phi)\left[\begin{array}{c} \displaystyle\sum_{m=1}^{\alpha}\sum_{n=1}^{\alpha}\left(\widehat{\Gamma}_{uv}^{(m,n)}(\theta)\cos n\phi - \breve{\Gamma}_{uv}^{(m,n)}(\theta)\sin n\phi\right) \\ \displaystyle\sum_{m=1}^{\alpha}\left(\widehat{\gamma}_{uv}^{(m,n)}(\theta) + \breve{\gamma}_{uv}^{(m,n)}(\phi)\right) + \frac{R_{uv}^{(0,0)}}{2} \end{array}\right] \\ -\omega_l'(\theta,\phi)\left[\begin{array}{c} \displaystyle\sum_{m=1}^{\alpha}\sum_{n=1}^{\alpha} m\left(\tilde{\Gamma}_{uv}^{(m,n)}(\theta)\cos n\phi - \hat{\Gamma}_{uv}^{(m,n)}(\theta)\sin n\phi\right) \\ \displaystyle\sum_{m=1}^{\alpha} m\tilde{\gamma}_{uv}^{m}(\theta) \end{array}\right] \\ -\omega_l''(\theta,\phi)\left[\begin{array}{c} \displaystyle\sum_{m=1}^{\alpha}\sum_{n=1}^{\alpha} n\left(\widehat{\Gamma}_{uv}^{(m,n)}(\theta)\sin n\phi - \widehat{\Gamma}_{uv}^{(m,n)}(\theta)\cos n\phi\right) \\ \displaystyle\sum_{m=1}^{\alpha} m\hat{\gamma}_{uv}^{m}(\phi) \end{array}\right] \end{array}\right\} \quad , \tag{69}$$

where $\omega_l(\theta,\phi)$, $\omega_l'(\theta,\phi)$ and $\omega_l''(\theta,\phi)$ represent the following spatial differentiations

$$\omega_l(\theta,\phi) = r\frac{\partial r}{\partial x_l} = \left\{\begin{array}{ll} \sin\phi\cos\theta, & (\text{for } l=1) \\ \sin\phi\cos\theta, & (\text{for } l=2) \\ \cos\phi, & (\text{for } l=3) \end{array}\right. \quad , \tag{70}$$

$$\omega_l'(\theta,\phi) = r\frac{\partial \theta}{\partial x_l} = \left\{\begin{array}{ll} -\sin\theta/\sin\phi, & (\text{for } l=1) \\ \cos\theta/\sin\phi, & (\text{for } l=2) \\ 0, & (\text{for } l=3) \end{array}\right. \quad , \tag{71}$$

$$\omega_l''(\theta,\phi) = r\frac{\partial \phi}{\partial x_l} = \begin{cases} \cos\phi\cos\theta, & (\text{for } l=1) \\ \cos\phi\sin\theta, & (\text{for } l=2) \\ -\sin\phi, & (\text{for } l=3) \end{cases} \quad , \tag{72}$$

and

$$\begin{aligned}
\overset{\frown}{\Gamma}{}_{uv}^{(m,n)}(\theta) &= \tilde{R}_{uv}^{(m,n)}\cos m\theta - \tilde{I}_{uv}^{(m,n)}\sin m\theta \quad , \\
\overset{\smile}{\Gamma}{}_{uv}^{(m,n)}(\theta) &= \hat{R}_{uv}^{(m,n)}\sin m\theta + \hat{I}_{uv}^{(m,n)}\cos m\theta \quad , \\
\tilde{\Gamma}_{uv}^{(m,n)}(\theta) &= \tilde{R}_{uv}^{(m,n)}\sin m\theta + \tilde{I}_{uv}^{(m,n)}\cos m\theta \quad , \\
\hat{\Gamma}_{uv}^{(m,n)}(\theta) &= \hat{R}_{uv}^{(m,n)}\cos m\theta - \hat{I}_{uv}^{(m,n)}\sin m\theta \quad , \\
\overset{\frown}{\gamma}{}_{uv}^{m}(\theta) &= R_{uv}^{(m,0)}\cos m\theta - I_{uv}^{(m,0)}\sin m\theta \quad , \\
\overset{\smile}{\gamma}{}_{uv}^{m}(\phi) &= R_{uv}^{(0,m)}\cos m\phi - I_{uv}^{(0,m)}\sin m\phi \quad , \\
\tilde{\gamma}_{uv}^{m}(\theta) &= R_{uv}^{(m,0)}\sin m\theta + I_{uv}^{(m,0)}\cos m\theta \quad , \\
\hat{\gamma}_{uv}^{(m,n)}(\theta) &= R_{uv}^{(0,m)}\sin m\phi + I_{uv}^{(0,m)}\cos m\phi \quad ,
\end{aligned} \tag{73}$$

finally, the traction fundamental solution T_{ij} can be evaluated by

$$T_{ij} = (\sigma_{ik}n_k)_j \quad , \tag{74}$$

where σ_{ik} is the fundamental solution of stresses and n_k is the outward normal vector on the surface at the field point. Using the generalized Hooke's law

$$(\sigma_{ik})_j = C_{ikmn}(U_{mj,n} + U_{nj,m})/2 \quad . \tag{75}$$

A numerical singularity occurs in the formulation of this fundamental solution in the evaluation of its derivative $\mathbf{U}'$, in the special case when the field and source point are both over the x_3-axis at $\phi = 0$ or $\phi = \pi$ for $l = 1$ and $l = 2$. This condition occurs because the spherical angle θ becomes ill-conditioned at these locations. Tant et al. [25] suggest a scheme to remove this singularity, imposing a small perturbation for ϕ, such as $\phi = 10^{-6}$ and $\theta = 0$ for $l = 1$. In case of $l = 2$, it is selected $\theta = \pi/2$.

4.2 Run Fourier coefficients

BESLE requires, as material input parameters, the Fourier coefficients and the stiffness tensor that are generated in a separated database. From Eqs. (65) and (73), and including the stiffness tensor, the parameters that need to be exported are

$$\begin{gathered}
\tilde{R}_{uv}^{(m,n)}, \quad \hat{R}_{uv}^{(m,n)}, \quad \tilde{I}_{uv}^{(m,n)}, \quad \hat{I}_{uv}^{(m,n)} \quad , \\
R_{uv}^{(0,m)}, \quad I_{uv}^{(0,m)}, \quad R_{uv}^{(m,0)}, \quad I_{uv}^{(m,0)} \quad , \\
R_{uv}^{(0,0)}, \quad C_{ijkl} \quad .
\end{gathered}$$

The algorithm of this `Fortran` sub-package is explained in more detail in [1,2]. It uses a listing scheme for some special cases, where the user can define several materials from external files that contain lists of material properties. This sub-package is located in the `~/BESLE/Material/` root, and the user's script `Set_parameters.f90` is at the `~~/BESLE/Material/src/` folder. In order to compile the package, in the terminal

```
~/BESLE/Material$ make all
```

or just `make` should work. Before the compilation, the user's script must be configured. At any subsequent modification, the code is recompiled by

```
~/BESLE/Material$ make recompile
```

if something goes wrong during configuration, the next compilation is carried out using

```
~/BESLE/Material$ make fix
```

and, to execute

```
~/BESLE/Material$ ./F_Coeff
```

Finally, `make clean` deletes all the files and compilation. Now learning by examples, some cases are explored, where all the syntax and commands are shown. The data is saved in a `"*.dat"` file at the `fileplace` with `file_name` defined by the user. It is worth noting that, the user should be aware of the units, and for all examples, properties in MPa are used. Furthermore, as the code is implemented with double precision, `*.d0` or `*d0` should be added to the quantities. The necessary files to run the examples are located at `~/BESLE/Examples/Material/`, the user just needs to replace them into the `~/BESLE/Material/` depending on each case.

In the first example shown in Table 14(a), the idea is to obtain the data for isotropic iron characterized by only two elastic proterties, the Young's modulus and Poisson's ratio. `Material` is a string variable and its values are reserved, it defines for the simulation if the material is `"Isotropic"`or `"Anisotropic"`.

(a)

```
SUBROUTINE Setup

    fileplace="Data/Isotropic/"
    file_name="Material_7.dat"

    Material="Isotropic"

    E=210000.d0; nu=0.3d0;

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup

    fileplace="Data/Anisotropic/"
    file_name="Material_1.dat"

    Material="Anisotropic"
    Lattice="cubic"

    C11=230000.d0;
    C12=135000.d0;
    C44=117000.d0;

END SUBROUTINE Setup
```

Table 14: Examples, (a) isotropic iron and (b) BCC iron.

In the second example in Table 14(b), the material this time is the BCC iron that is anisotropic and is described by three independent elastic constants C_{11}, C_{12}, and C_{44}. A new string variable appears, `Lattice`, that defines the crystal system of the material. In this case, anisotropic iron is a BCC material belonging to the `"cubic"` system. As well-known, the stiffness tensor in anisotropic materials depends on its symmetries. Therefore, the `Lattice` variable accounts with some other crystal systems as hexagonal closed package `"hcp"`, `"trigonal"`, and `"full"` for the most general case.

The third example in Table 15(a), illustrates how to work again with anisotropic materials in this case a `"hcp"` zinc that has five independent elastic constants C_{11}, C_{12}, C_{13}, C_{33}, and C_{44}. If the material is rotated with a specific orientation, by default the code proceeds with the $x-y-z$ convention and the `theta_x`, `theta_y`, and `theta_z` should be given in degrees. It is worth noting that these angles are positive counterclockwise.

(a)

```
SUBROUTINE Setup

    fileplace="Data/Anisotropic/"
    file_name="Material_8.dat"

    Material="Anisotropic"
    Lattice="hcp"

    C11=165000.d0; C12=31100.d0;
    C13=50000.d0; C33=61800.d0;
    C44=33600.d0;

    theta_x=-125.d0;
    theta_y=10.d0;
    theta_z=332.d0;

END SUBROUTINE Setup
```

(b)

```
SUBROUTINE Setup

    fileplace="Data/Anisotropic/"
    file_name="Material_13.dat"

    Material="Anisotropic"
    Lattice="trigonal"

    C11=87600.d0; C12=6070.d0;
    C13=13300.d0; C14=17300.d0;
    C33=106800.d0; C44=57200.d0;

    z_x_z = 1
    theta_z=-15.d0;
    theta_x=135.d0;

END SUBROUTINE Setup
```

Table 15: Examples, (a) hcp zinc oriented with $x-y-z$ convention, and (b) trigonal Alpha Quartz oriented with $z-x-z$ convention.

In Table 15(b), an additional case is shown for a material with a `"trigonal"` crystal system that accounts with six independent elastic constants C_{11}, C_{12}, C_{13}, C_{14}, C_{33}, and C_{44}. The difference here is the specification of the orientation following the $z-x-z$ convention, and the variable `z_x_z` is activated. Further, only two angles should be defined `theta_z` and `theta_x` for this case.

In the last examples, input data for single domain materials were generated. Now, consider a heterogeneous material with multiple domains or constituents. In order to facilitate this procedure, henceforth, the `file_name` variable now corresponds to the name of an input file containing the necessary information located at `fileplace`. By default, the resulting files are saved with the sequence names of `"Mat_i.dat"` also at the `fileplace`.

For illustrative purposes in Table 16, the fifth example to generate data for a material composed of multiple isotropic domains, `"Multiple_iso"`, is shown. Hence, the file `"E_v_constants.dat"` contains the list of Young's moduli and Poisson's ratios, the number of domains are specified by `n_materials`.

```
SUBROUTINE Setup
    n_materials=8

    fileplace="Data/Isotropic/Multiple/Simulation_1/"
    file_name="E_v_constants.dat"

    Material="Multiple_iso"
END SUBROUTINE Setup
```

Table 16: Material composed of multiple isotropic domains

The same case for a material composed of multiple anisotropic domains `"Multiple_aniso"` is shown in Table 17. This sixth example differs from the last, in the input file `"C_tensors.dat"` that contains the twenty one elastic constants of the stiffness tensor C_{ijkl} at each line for each material as C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}, C_{22}, C_{23}, C_{24}, C_{25}, C_{26}, C_{33}, C_{34}, C_{35}, C_{36}, C_{44}, C_{45}, C_{46}, C_{55}, C_{56}, and C_{66}. This is a general case in which isotropic tensors could also be included.

```
SUBROUTINE Setup
    n_materials=8

    fileplace="Data/Anisotropic/Multiple/Simulation_1/"
    file_name="C_tensors.dat"

    Material = "Multiple_aniso"
END SUBROUTINE Setup
```

Table 17: Material composed of multiple anisotropic domains

There are heterogeneous materials composed by the same anisotropic stiffness however with different orientation in each phase, e.g. polycrystals or trabecular bones. Hence, to generate the necessary database, the example seven is presented in Table 18. In this case, `"Angles.dat"` is the input file containing a list of `theta_x` and `theta_z` angles in each line, because `z_x_z` is activated. This time, the material is anisotropic FCC `"cubic"` copper with three independent elastic constants.

```
SUBROUTINE Setup

    n_materials=100

    fileplace="Data/Anisotropic/Orientations/Cu/"
    file_name="Angles.dat"

    Material="Anisotropic"
    Lattice="cubic"

    C11=168000.d0;
    C12=121400.d0;
    C44=75400.d0

    z_x_z=1

END SUBROUTINE Setup
```

Table 18: Polycrystalline FCC copper with $z-x-z$ orientations.

In example eight in Table 19, data for polycrystalline `"hcp"` zinc is generated again for 100 different crystalline orientations following the default $x-y-z$ convention. In this case the `"Angles.dat"` file contains a list of `theta_x`, `theta_y`, and `theta_z` angles in each line.

```
SUBROUTINE Setup

    n_materials=100

    fileplace="Data/Anisotropic/Orientations/Zn/"
    file_name="Angles.dat"

    Material="Anisotropic"
    Lattice="hcp"

    C11=165000.d0; C12=31100.d0;
    C13=50000.d0; C33=61800.d0;
    C44=33600.d0

END SUBROUTINE Setup
```

Table 19: Polycrystalline hcp zinc with $x-y-z$ orientations.

Finally, the example nine, where a special case is shown in Table 20. Data for a Lamellar bone tissue with a fully populated stiffness tensor is generated. In this case the `Lattice` variable is set as `"full"` and the twenty one elastic constants should be defined.

```
SUBROUTINE Setup

    n_materials=100

    fileplace="Data/Anisotropic/Orientations/Lbt/"
    file_name="Angles.dat"

    Material='Anisotropic'
    Lattice='full'

    C11=22.571d0;     C12=10.326d0;
    C13=9.451d0;      C14=0.003051d0;
    C15=0.00219d0;    C16=-0.134d0;
    C22=22.922d0;     C23=9.622d0;
    C24=0.003051d0;   C25=1.947d0;
    C26=-0.139d0;     C33=25.262d0;
    C34=-0.003821d0; C35=-0.001656d0;
    C36=-0.07954d0;   C44=4.954d0;
    C45=-0.106d0;     C46=0.004102d0;
    C55=4.833d0;      C56=-0.009525d0;
    C66=5.444d0;

END SUBROUTINE Setup
```

Table 20: Lamellar bone tissue with $x-y-z$ orientations.

This sub-package offers the advantage to create large material databases, that could be used at any time reducing the computational cost for simulations that need to be run more than once. This is accomplished through the functionalities to reproduce materials composed of different constituents using the several list schemes presented in the examples.

Physical models

BESLE has the capability for modelling several physical problems on solids. As mentioned before isotropic and anisotropic materials can be simulated further one and multiple domains for the case of heterogeneous media. Similarly, BESLE is run from its main user's script `Set_parameters.f90` that is located at the `~/BESLE/src/` folder. As the package works only on a distributed memory architecture, assuming that the installation was successful, to build the BESLE application

```
~/BESLE$ make
```

Before the compilation, the user's script must be configured. At any subsequent modification, the code is recompiled by

```
~/BESLE$ make recompile
```

if there is something wrong during configuration, the next compilation is carried out using

```
~/BESLE$ make fix
```

and, to execute

```
~/BESLE$ mpirun -np <nthreads> ./BESLE
```

where `<nthreads>` indicates the number of cores/threads defined by the user, which in this version `threads > 1` . Finally, `make clean` to delete all the files and compilation. Now learning by examples, some cases are explored in the subsequent subsections, where all the syntax and commands are shown. The necessary files to execute these examples are located in the `~/BESLE/Examples/Physical Models` folder. Before the examples, some default variables are explained here, where the user can decide how to customise the simulation. We recall Fig. 1 which is shown again in Fig. 11.

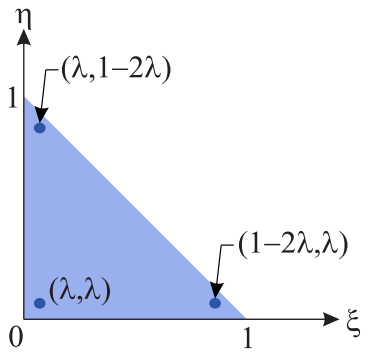


Figure 11: Linear three-node discontinuous element.

After trials of different values of `lambda` (λ), by default this variable was set in `0.155d0`. The user is free to change this parameter if required by the application.

The integrals in Eq. (26) are carried out by using Gauss-quadrature, in BESLE some levels of integration refinement are provided to be configured depending on the application. For the non-singular integration, there are two possibilities of distribution of the Gauss rule for triangles, using 7 and 13 Gauss integration points [15]. Then, the `npoints_rule_NSI` variable was set by default as `13`. In order to improve the quality of the non-singular integration, the boundary element can be split into sub-triangles. The `int_type` variable is assigned for this purpose, it can take values such as 1, 4, 8, and 16. However, by default `int_type` is set as `4` in the software, which means the non-singular integration will pursue with 52 Gauss points.

Quasi-singularity problems appear in the BEM when the radius of the integration of the fundamental solutions is very small, Eqs. (65) and (69), e.g. in bonded joins where the adhesive use to have $O(10^{-1})$ mm of thickness, details in [10, 11]. In those cases, it is recommended to increase the number of Gauss points used for the integration. Hence the `int_type` and `npoints_rule_NSI` variables should take the greatest available values in BESLE.

For the singular integration, a customised division of the boundary element was implemented for each case when the source and field points coincide at the same node. Here, the `npoints_rule_SI` variable is `13` by default. However, this rule integration can work also with 7 Gauss points.

The MUMPS solver includes the option to configure the amount of RAM to be used for each thread, if needed the user can set it using the `working_memory` variable. By default, this is defined with the value of 13 in GB units. It is important to consider that the solver works without the master thread, therefore, the total available memory must be divided into ($nthreads$ - 1).

In addition to the MUMPS diagnostics, in BESLE two main error situations were included in the debug process. Due to the allocation limit of 32-bits Fortran and depending on the available RAM, for really large problems the number of degrees of freedom could exceed its limit that was established approximately as $0.70 \cdot (nthreads) \cdot (2^{31} - 1)$ non-zero entries of the main general system of equations, the matrix $\mathcal{A}$ in Eqs. (42)-(44). In this case, a message will appear in the terminal indicating the number of additional DOF in each rank. To solve this, the user must run the simulation again increasing the number of threads if available, or reduce the mesh if possible.

A second error situation occurs when there are overlap elements or repeated nodes that make the minimum segment of all the boundary elements be zero. In that case, the mesh must be reviewed and edited. Finally, as a recommendation and good practice, no more than 20 boundary elements mush converge to the same node.

5.1 Quasi-static behavior of polycrystal aggregates

In this model, consider a 96-grain zinc polycrystal aggregate with dimensions $L_x = 15\ \mu$m, $L_y = 15\ \mu$m and $L_z = 45\ \mu$m, Fig. 12. In this case, each grain has its proper crystalline orientation from the reference five independent elastic constants $C_{11} = 165$ GPa, $C_{12} = 31.1$ GPa, $C_{13} = 50$ GPa, $C_{33} = 61.8$ GPa, and $C_{44} = 33.6$ GPa. Therefore, the stiffness tensor C_{ijkl} should be rotated following the $x - y - z$ convention.

The specimen is fully constrained in all axes $u_i = 0$ at the bottom in the Γ_5 surface. When nothing is defined by the user, the default boundary condition corresponds to a free surface $t_i = 0$ imposed in this model in the lateral surfaces. At the top in the Γ_6 surface, a ramp $u_3 = 0.0025s\,\mu$m displacement is applied where s takes the value of the current load step $0 < s \leq 20$. Hence, as 20

load steps are defined in this simulation, the u_3 displacement goes from 0.0 μm to a maximum of 0.05 μm. Finally, the set of boundary conditions are complete with the free surface conditions at the top $t_1 = 0$ and $t_2 = 0$.

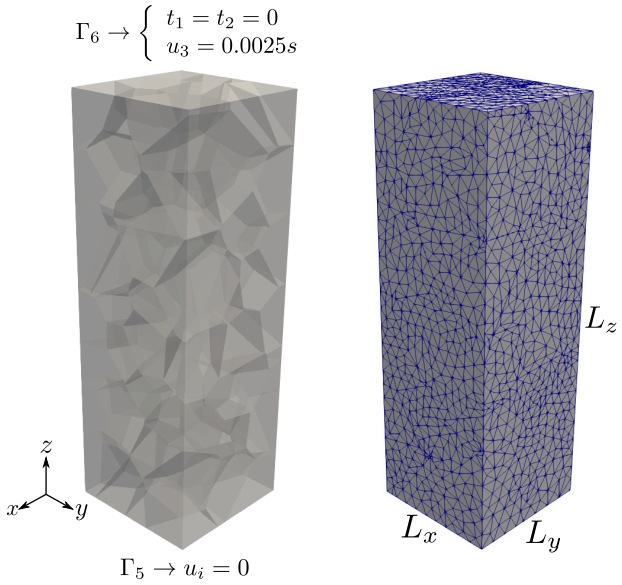


Figure 12: Model of a 96-grains polycrystalline material discretised with 22767 boundary elements.

First, the polycrystal generator is used to build the structure using the `Set_parameters.cc` script as shown in Table 12. Note that for convenience, the specimen size is defined in the $O(10)$. This is to avoid problems with the double precision code specially in the interface generation process. Then, the mesh is saved in the `"Mesh.dat"` file at the `~/BESLE/Mesh/Polycrystal/Export/` root.

```
void Setup()
{
    // Grains in each coordinate
    ngrains_x=3;
    ngrains_y=4;
    ngrains_z=8;

    // Box dimensions
    x_max = 15;
    y_max = 15;
    z_max = 45;

    // Random geometry
    stochastic = 0;

    // Mesh density parameter
    dm = 1;
}
```

Table 21: Script for generating a 96-grains polycrystalline structure.

If this is the first time using hcp zinc, the material package is employed for generating the polycrystalline `"hcp"` zinc database. It must contain about 100 different crystalline orientations following the default $x-y-z$ convention. In this case the `"Angles.dat"` file contains the list of `theta_x`, `theta_y`, and `theta_z` angles in each line. The material database requires to identify each domain and assign the correct elastic properties. For this, the file `List_Mat.dat` located at the `~/BESLE/Material/Data/Anisotropic/Orientations/Zn/` root is used. It contains the list of numbers that correspond to each `"Mat_i.dat"` file. Finally, the `Set_parameters.f90` script as shown in Table 22 is configured.

```
SUBROUTINE Setup

    n_materials=100

    fileplace="Data/Anisotropic/Orientations/Zn/"
    file_name="Angles.dat"

    Material="Anisotropic"
    Lattice="hcp"

    C11=165000.d0; C12=31100.d0;
    C13=50000.d0; C33=61800.d0;
    C44=33600.d0

END SUBROUTINE Setup
```

Table 22: Polycrystalline hcp zinc with $x-y-z$ orientations.

Now the configuration of BESLE `Set_parameters.f90` script is illustrated in Table 24. First, the `mesh_file` and `fileplace_mesh` variables to indicate the name and from where the mesh file should be read, respectively.

In case of a material with different constituents, the variable `n_Materials` should be used as greater than `1`, otherwise it is `0` by default. For the material definition, similar to the mesh there are two variables the `Material_coefficients_database` and `fileplace_material` that correspond to the name and from where the database files should be read.

As the mesh was generated using $O(10)$ order in its dimensions or in this case in (μm) units, the entire mesh should be scaled using the variable `scale_size_1` with the value of `0.001d0` to get it in (mm) units. The default value of this variable is `1`. It is worth noting that the material properties were obtained in (MPa) units being compatible with the geometrical dimensions in mm.

In this problem, no mass or inertial effects are acting on the simulation as the time scale is large. Then, the `quasi_static` variable should be activated with the number `1`. Also the number of `static_steps` must also be defined.

If the specimen has a box shape as shown in Fig. 12, BESLE includes a straightforward way to configure the boundary conditions manually within the main `Set_parameters.f90` script. Consider the variable `box_face_i`, where `i = 1, 2,..., 6` are the six surfaces of the box which represent a vector as follows

$$\text{box\_face\_i} = \left[value_x,\ type_x,\ value_y,\ type_y,\ value_z,\ type_z \right] \quad , \tag{76}$$

where

$$type_j \rightarrow \begin{cases} 0 \text{ for displacement} & , \\ 1 \text{ for traction} & , \end{cases} \tag{77}$$

By default, the box surfaces are set from 1 to 6 as presented in Table 23.

i	Γ_i
1	$x = 0$
2	$y = 0$
3	$x = x_{max}$
4	$y = y_{max}$
5	$z = 0$
6	$z = z_{max}$

Table 23: Box surfaces.

According to the last definitions, the `box_face_5` variable mimics the fully constrained in all axes $u_i = 0$ at the bottom. the `box_face_6` variable imposes the ramp $u_3 = 0.0025s$ μm displacement and the free surface conditions at the top $t_1 = 0$ and $t_2 = 0$. In this simulation a `"ramp"` is intended to be applied given by the `load_profile` variable. For this type of load, the user must impose the maximum applied load in mm units for consistency. The `load_profile` variable also accounts with some other type of loads, the `"heaviside"` and `"harmonic"` functions.

```
MODULE Set_parameters
USE Global_variables
CONTAINS
    SUBROUTINE Setup

        mesh_file = 'Mesh'
        fileplace_mesh = 'Mesh/Polycrystal/Export/'

        n_Materials = 96
        material_coefficients_database = 'Zn'
        fileplace_material = "Material/Data/Anisotropic/Orientations/"
        scale_size_1 = 0.001d0

        quasi_static = 1
        static_steps = 20

        load_profile='ramp'
        box_face_5 = (/0.d0,0.d0,0.d0,0.d0,0.d0,0.d0/)
        box_face_6 = (/0.d0,1.d0,0.d0,1.d0,0.00005d0,0.d0/)

        crack=1

        scale_results = 10
        results_file = 'Results'
        fileplace_results = 'Results/'

    END SUBROUTINE Setup
END MODULE Set_parameters
```

Table 24: Quasi-static behaior of polycrystal aggregates.

In order to impose a `"Heaviside"` load, it is only necessary to include the magnitude of the displacement or traction in the `box_face_i` variable. The `"harmonic"` case requires two addtional definitions the frequency `omega` and the `phase`. Then in the former case the boundary conditions will follow the $A\sin(\omega t+\phi)$ function, where the wave amplitude is again established for displacement or traction in the `box_face_i` variable.

Cracks can be incorporated in the physical model, through the `crack` variable being `1`, its default values is `0`. The user defines virtual boxes across the specimen in which all the interfaces contained within a specific box will be separated before the simulation. When `crack=1`, BESLE reads the `Input_cracks.dat` file located at the `~/BESLE/Cracks/` root with the following format

n					
x^1_{min}	x^1_{max}	y^1_{min}	y^1_{max}	z^1_{min}	z^1_{max}
x^2_{min}	x^2_{max}	y^2_{min}	y^2_{max}	z^2_{min}	z^2_{max}
.					
.					
.					
x^n_{min}	x^n_{max}	y^n_{min}	y^n_{max}	z^n_{min}	z^n_{max}

Table 25: Definition of virtual boxes for the incorporation of cracks.

In Table 25, n indicates the number of virtual boxes. Hence, in this model, a set of cracks were included within a specific region defined by the following virtual box expressed in mm units

1					
0.006	0.016	0.006	0.016	0.015	0.045

Table 26: Inclusion of cracks in the polycrstal aggregate.

The `scale_results` variable is to scale the displacement field for improving the visualisation in terms on how the specimen is being deformed. Finally, the `scale_results` indicates the name of the `*.vtk` file that is saved at the `results` folder. In ParaView, the user can get the visualisation of the displacement u_i field, the total displacement u_t, stress σ_{ij} and strain ε_{ij} tensors, and the von Mises σ_{VM} stress at every time or load step.

Figures 13 and 14 show the displacement field and σ_{33} stress component at different levels of load, respectively. The influence of the initial crack included in the model can be appreciated. In this version, crack propagation formulation was not incorporated, however, the inclusion of cracks can be a useful way for validating SIF results of simple specimens. Quasi-static analyses of the anisotropy of these materials using this BEM formulation can be studied in detail in [1, 2].

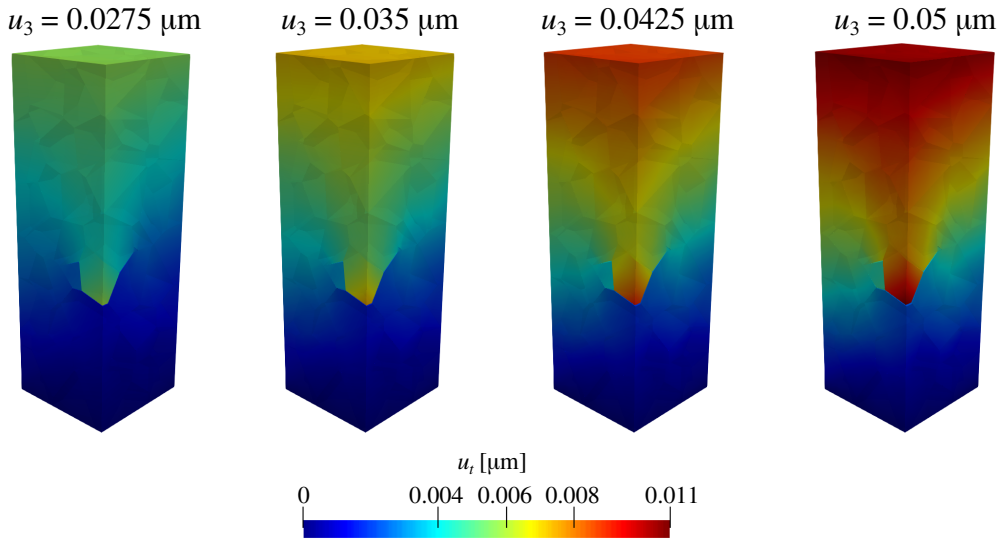


Figure 13: Total displacement.

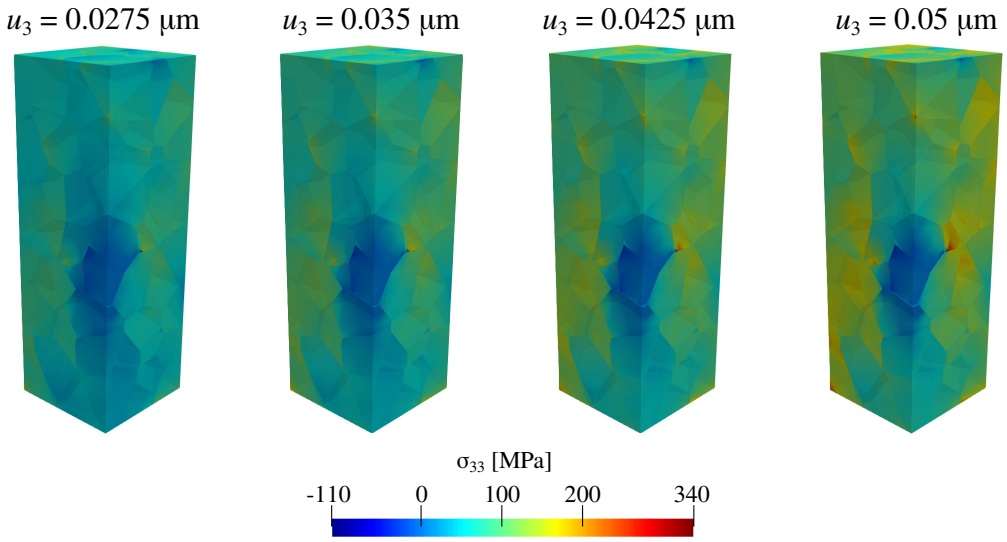


Figure 14: σ_{33} stress component.

Using several ParaView tools, the user can explore the results graphically in detail. These results are consistent in magnitude and units within the limits of the elasticity regime at this microscopic simulation.

5.2 Transient behavior of polycrystal aggregates

Consider a 96-grain iron polycrystal aggregate with dimensions $L_x = 15$ μm, $L_y = 15$ μm and $L_z = 45$ μm, Fig. 15. The polycrystalline structure was generated following the same procedure described in the last example, Table 21. In this case, each grain has its proper crystalline orientation from the reference three independent elastic constants $C_{11} = 230$ GPa, $C_{12} = 135$ GPa, and $C_{44} = 117$ GPa. Therefore, the stiffness tensor C_{ijkl} should be rotated following the $x - y - z$ convention.

The specimen is fully constrained in all axes $u_i = 0$ at the bottom in the Γ_5 surface. Free surface $t_i = 0$ boundary conditions were imposed in the lateral surfaces. At the top in the Γ_6 surface, a Heaviside $t_3(\tau) = 200$ MPa traction is applied. In order to capture the transient regime and dynamic effects, high-rate boundary conditions are applied in a period of 0.25 μs. Hence, the simulation comprises 50 steps of 5 ns. Finally, the set of boundary conditions are complete with the free surface conditions at the top $t_1 = 0$ and $t_2 = 0$. Similar simulations on polycrystal aggregates were carried out in more detail in [2, 30, 31].

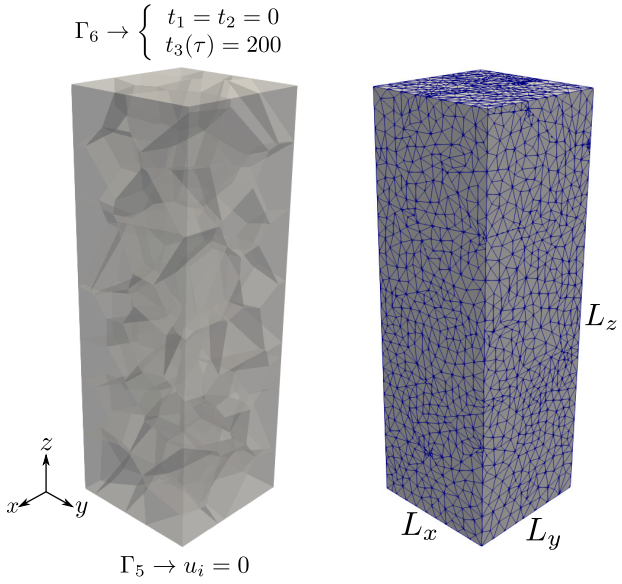


Figure 15: Model of a 96-grains polycrystalline material discretised with 22767 boundary elements.

If this is the first time using BCC iron, the material package is employed for generating the polycrystalline `"cubic"` iron database. It must contain about 100 different crystalline orientations following the default $x - y - z$ convention. In this case the `"Angles.dat"` file contains the list of `theta_x`, `theta_y`, and `theta_z` angles in each line. For this purpose, the `Set_parameters.f90` script as shown in Table 27 is configured.

```
SUBROUTINE Setup

    n_materials=100

    fileplace="Data/Anisotropic/Orientations/Fe/"
    file_name="Angles.dat"

    Material="Anisotropic"
    Lattice="cubic"

    C11=230000.d0;
    C12=135000.d0;
    C44=117000.d0;

END SUBROUTINE Setup
```

Table 27: Polycrystalline BCC iron with $x-y-z$ orientations.

Now the configuration of BESLE `Set_parameters.f90` script is illustrated in Table 28. The first different variable here is `scale_prop_mat`, it facilitates the scale of the elastic constraints of the C_{ijkl} tensor. For convenience and numerical stability, this simulation is configured in mm units.

```
MODULE Set_parameters
!-------------------------------------------------------------!
USE Global_variables
!-------------------------------------------------------------!
CONTAINS

    SUBROUTINE Setup

        mesh_file = 'Mesh'
        fileplace_mesh = 'Mesh/Polycrystal/Export/'

        n_Materials = 96
        material_coefficients_database = 'Fe'
        fileplace_material = "Material/Data/Anisotropic/Orientations/"
        scale_prop_mat = 1.d0*(1e3)
        scale_size_1 = 0.001d0

        transient = 1
        time_steps = 50
        dt = 5e-9
        density = 7874.d0/(1000**3)

        box_face_5 = (/0.d0,0.d0,0.d0,0.d0,0.d0,0.d0/)
        box_face_6 = (/0.d0,1.d0,0.d0,1.d0,200.d0*(1e3),1.d0/)

        scale_results = 200
        results_file = 'Results'
        fileplace_results = 'Results/'

    END SUBROUTINE Setup

END MODULE Set_parameters
```

Table 28: Transient behavior of polycrystal aggregates.

In order to get the consistent units, according to the analytical solution of a bar under a

Heaviside load [32, 33], the natural frequency of the nth vibration mode is given by

$$\omega_n = (2n-1)\frac{\pi}{2}\sqrt{\frac{EA}{mL^2}} \quad n = 1, 2, \ldots \quad , \tag{78}$$

where E is the Young's modulus, A is the cross-sectional area, L represents the length of the bar, and m is the mass per unit length. The dimensional analysis of the natural frequency leads to $E \cdot (10^3)$ kg/mm $\cdot$ s^2 if E is originally given in MPa units. Therefore, all the length units in the simulation are correctly expressed in 10^3 units using `scale_prop_mat`, the default value of this variable is `1`. Further, the 10^3 factor must be also used in all the traction boundary conditions if they are initially expressed in MPa units. In this problem, the mass or inertial effects will be captured by the simulation as the time scale is very small. Then, the `transient` variable should be activated with the number `1`. Also the number of `time_steps`, the time step `dt`, and the mass `density` (in kg/mm^3) must be defined. Figure 16 shows the displacement field at different instants of time.

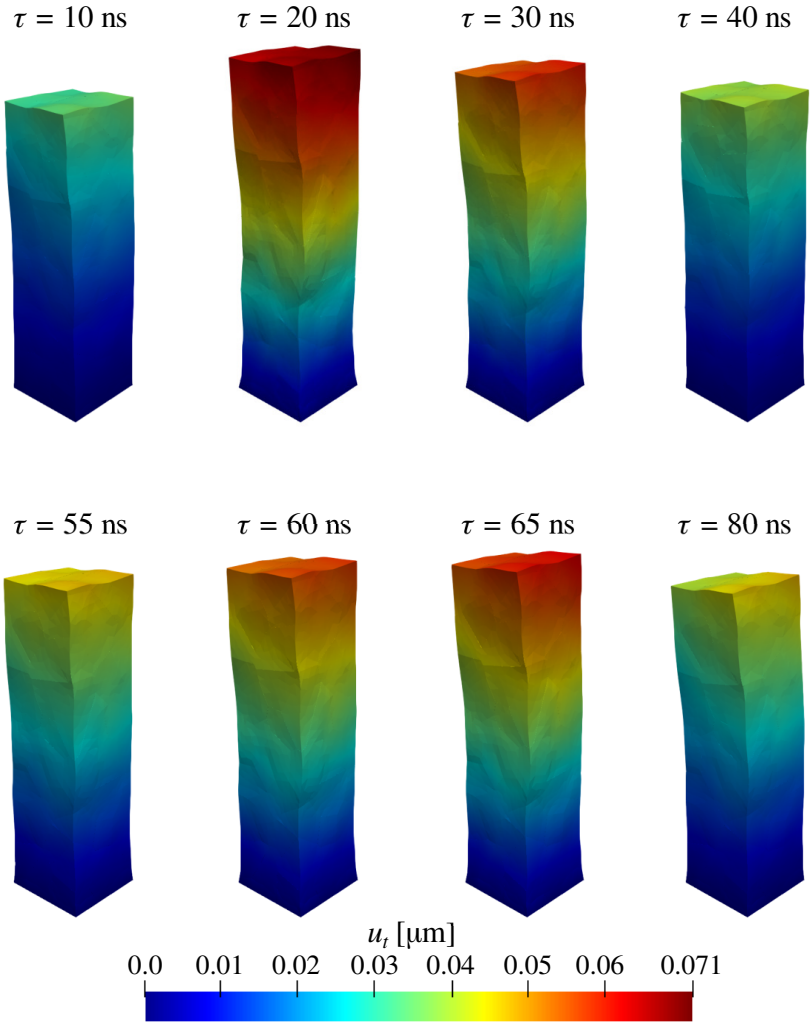


Figure 16: Total displacement.

In this case, the anisotropic deformation of the specimen is easily appreciated as `scale_results` is large compared to the last example. Due to high-rate boundary conditions, the stress and strain waves propagate along with the specimen. As the load is a Heaviside function, these waves reflect in the body without damping.

5.3 Body forces acting on isotropic materials

Consider a simple box specimen with dimensions $L_x = 15$ mm, $L_y = 15$ mm and $L_z = 30$ mm, Fig. 17. The material is constituted by one domain of isotropic iron with Young's modulus $E = 210$ GPa and Poisson ratio $\nu = 0.3$. It is fully constrained in all axes $u_i = 0$ at the bottom in the Γ_5 surface. When nothing is defined by the user, the default boundary condition corresponds to a free surface $t_i = 0$ condition imposed on the lateral and top surfaces. In this model, the effect of the body force ρg is observed, being the density $\rho = 7874\,\mathrm{ks/m^3}$ and $g = 9.81\,\mathrm{m/s^2}$ the acceleration of gravity. Therefore, it is expected to obtain the deformation of the solid under its own weight.

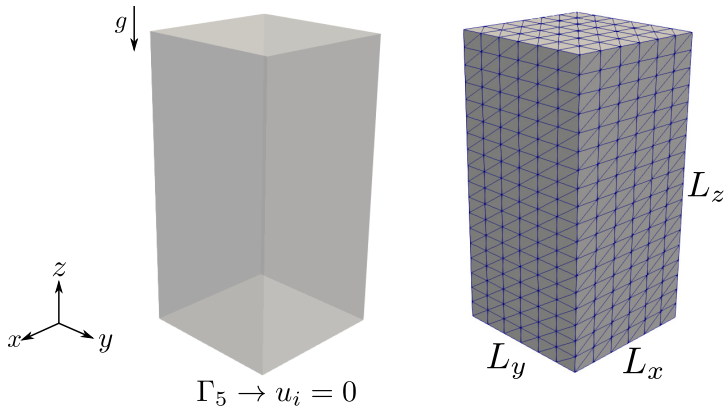


Figure 17: Box model of one domain discretised with 1008 boundary elements.

First, the box generator is used to build the mesh using the `Mesh_Generator.m` script as shown in Table 29. Then, the mesh is saved in the `"Input_data.dat"` file at the `~/BESLE/Mesh/Box/` root.

```
1
2  file = 'Input_data';
3
4  a = 15; % Dimension in x
5  b = 15; % Dimension in y
6  c = 30; % Dimension in z
7
8  nreg_a = 1; % Regions in x
9  nreg_b = 1; % Regions in y
10 nreg_c = 1; % Regions in z
11
12 nel_a = 6; % Elements in x / region
13 nel_b = 6; % Elements in y / region
14 nel_c = 18; % Elements in z / region
```

Table 29: Script for generating a box model of one domain.

If this is the first time using isotropic iron, the material package is employed for generating the "Material_7.dat" iron database. For this purpose, the `Set_parameters.f90` script is shown in Table 30.

```
SUBROUTINE Setup

    fileplace="Data/Isotropic/"
    file_name="Material_7.dat"

    Material="Isotropic"

    E=210000.d0; nu=0.3d0;

END SUBROUTINE Setup
```

Table 30: Isotropic iron.

Now the configuration of BESLE `Set_parameters.f90` script is illustrated in Table 31. First, the `mesh_file` and `fileplace_mesh` variables to indicate the name and from where the mesh file should be read, respectively.

```
MODULE Set_parameters
!-------------------------------------------------------------!
USE Global_variables
!-------------------------------------------------------------!
CONTAINS

    SUBROUTINE Setup

        mesh_file = 'Input_data'
        fileplace_mesh = 'Mesh/Box/'

        material_coefficients_file = 'Material_7'
        fileplace_material = "Material/Data/Isotropic/"

        bodyforces = 1
        bodyforce = (/0.d0,0.d0,-7.72439d0*(1e-5)/)

        density = 7874.d0/(1000**3)

        quasi_static = 1
        static_steps = 1

        box_face_5 = (/0.d0,0.d0,0.d0,0.d0,0.d0,0.d0/)

        scale_results = 4e7
        results_file = 'Results'
        fileplace_results = 'Results/'

    END SUBROUTINE Setup

END MODULE Set_parameters
```

Table 31: Body forces acting on isotropic materials.

In this case, the simulation requires only one material, then, similar to the mesh, there are

two variables the `material_coefficients_file` and `fileplace_material` that correspond to the name and from where the database files should be read.

As the mesh was generated using $O(10)$ order, it is directly in mm units. The mesh does not need to be scaled. It is worth noting that the material properties were obtained in (MPa) units being compatible with the geometrical dimensions in mm.

In this problem, the mass or inertial effects are taken into account in the simulation activating the `bodyforces` variable. If a constant body force is being applied within a `quasi-static` or `transient` simulation it can be defined in each axis using the `bodyforce` variable. The body force caused by the accelelation of gravity is $F_{bf} = (0, 0, -\rho g)$, where $\rho g = -7.72439(10^{-5})$ N/mm^3. A non-constant body force can also be applied if the user set the `type_bf` as `1` in that case `bodyforce` is deleted from the script. Therefore, BESLE will read the `"Body_forces.dat"` file located at the `~/BESLE/Body_forces/` root. This file contains a list of values for each component of the body force. The number of body force steps must be the same as defined by `static_steps` or `time_steps`. Figure 18 shows the displacement field components and the von Mises stress caused by the acceleration of gravity.

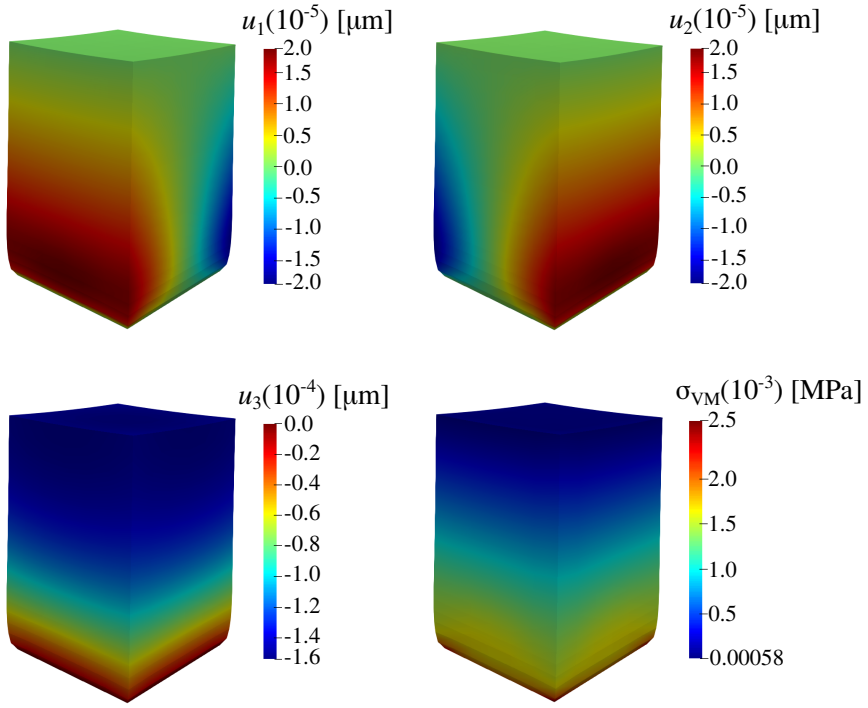


Figure 18: Displacement components and von Mises stress.

The isotropy of homogeneous iron leads to a symmetric deformation on the x-y plane of the specimen, and the displacement is mainly along its vertical axis.

5.4 Body forces acting on anisotropic materials

This is a similar case to the last example, consider a simple box specimen with dimensions $L_x = 15$ mm, $L_y = 15$ mm and $L_z = 30$ mm, Fig. 19. The material is constituted by three domains of the same trigonal alpha-quartz with six independent elastic constants $C_{11} = 87.6$ GPa, $C_{12} = 6.07$ GPa, $C_{13} = 13.3$ GPa, $C_{14} = 17.3$ GPa, $C_{33} = 106.8$ GPa, and $C_{44} = 57.2$ GPa. The C_{ijkl} tensor will be rotated $\theta_z = -30°$, $\theta_y = -30°$, and $\theta_z = -60°$ to get a new material orientation. The specimen is fully constrained in all axes $u_i = 0$ at the bottom in the Γ_5 surface. When nothing is defined by the user, the default boundary condition corresponds to a free surface $t_i = 0$ imposed on the lateral and top surfaces. In this model, the effect of the body force ρg is observed, being the density $\rho = 2648\ \mathrm{ks/m^3}$ and the acceleration of gravity $g = 9.81\ \mathrm{m/s^2}$. Therefore, it is expected to obtain the deformation of the solid under its own weight.

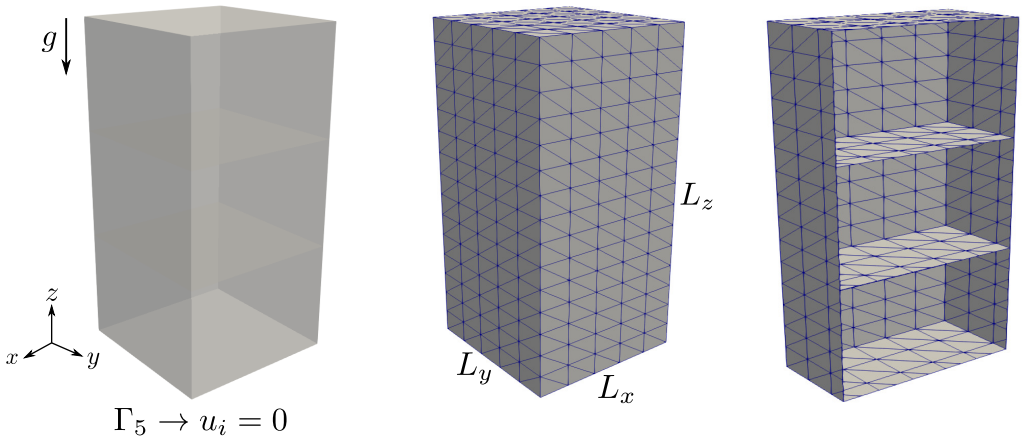


Figure 19: Box model of three domains discretised with 900 boundary elements.

The box generator is used to build the mesh using the `Mesh_Generator.m` script as shown in Table 29. Then, the mesh is saved in the `"Input_data.dat"` file at the `~/BESLE/Mesh/Box/` root.

```
file = 'Input_data';

a = 15; % Dimension in x
b = 15; % Dimension in y
c = 30; % Dimension in z

nreg_a = 1; % Regions in x
nreg_b = 1; % Regions in y
nreg_c = 3; % Regions in z

nel_a = 5; % Elements in x / region
nel_b = 5; % Elements in y / region
nel_c = 5; % Elements in z / region
```

Table 32: Script for generating a box model of three domains.

If this is the first time using this material, the material package is employed for generating the `"Material_4.dat"` alpha-quartz database. For this purpose, the `Set_parameters.f90` script as

shown in Table 33 is configured.

```
SUBROUTINE Setup

    fileplace="Data/Anisotropic/"
    file_name="Material_4.dat"

    Material="Anisotropic"
    Lattice="trigonal"

    C11=87600.d0; C12=6070.d0;
    C13=13300.d0; C14=17300.d0;
    C33=106800.d0; C44=57200.d0;

    theta_x=-30.d0;
    theta_y=-45.d0;
    theta_z=-60.d0;

END SUBROUTINE Setup
```

Table 33: Trigonal alpha-quartz oriented with $x-y-z$ convention.

Now the configuration of BESLE `Set_parameters.f90` script is illustrated in Table 34. First, the `mesh_file` and `fileplace_mesh` variables to indicate the name and from where the mesh file should be read, respectively.

```
MODULE Set_parameters
!-------------------------------------------------------------!
USE Global_variables
!-------------------------------------------------------------!
CONTAINS

    SUBROUTINE Setup

        mesh_file = 'Input_data'
        fileplace_mesh = 'Mesh/Box/'

        material_coefficients_file = 'Material_4'
        fileplace_material = "Material/Data/Anisotropic/"

        bodyforces = 1
        bodyforce = (/0.d0,0.d0,-2.5976d0*(1e-5)/)

        density = 2648.d0/(1000**3)

        quasi_static = 1
        static_steps = 1

        box_face_5 = (/0.d0,0.d0,0.d0,0.d0,0.d0,0.d0/)

        scale_results = 1.5e8
        results_file = 'Results'
        fileplace_results = 'Results/'

    END SUBROUTINE Setup

END MODULE Set_parameters
```

Table 34: Body forces acting on anisotropic materials.

In this case, the simulation requires the same material for the three domains, then, similar to the mesh there are two variables, the `material_coefficients_file` and `fileplace_material` that correspond to the name and from where the database files should be read. As the mesh was generated using $O(10)$ order it is directly in mm units, then, the mesh does not need to be scaled. It is worth noting that the material properties were obtained in (MPa) units being compatible with the geometrical dimensions in mm. The body force caused by the acceleration of gravity is $F_{bf} = (0, 0, -\rho g)$, where $\rho g = -2.5976(10^{-5})$ N/mm^3. Figure 20 shows the displacement u_3 component and the σ_{33} stress caused by the acceleration of gravity

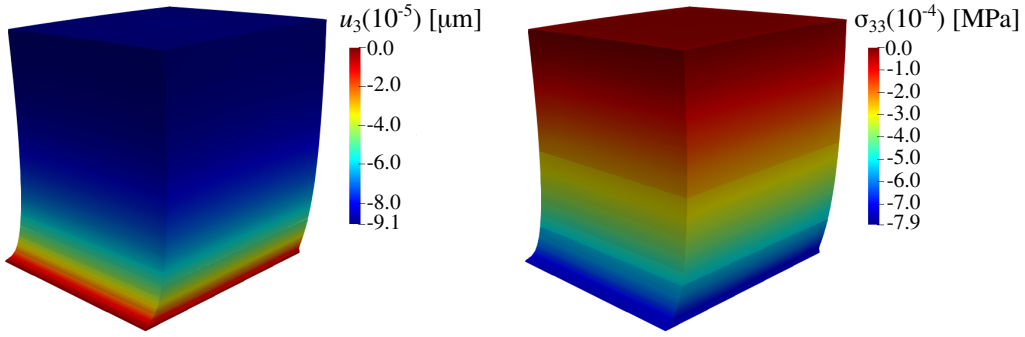


Figure 20: Displacement u_3 component and σ_{33} stress.

Here, in contrast to the last case, due to the level of anisotropy of the alpha-quartz material, the deformation is not symmetric on the $x - y$ plane of the specimen.

5.5 Composite material modelling

In this model, a carbon/epoxy composite material of 18 fibres of 7 μm diameter is simulated. The morphology was built using 3ds Max as described in the General mesh and boundary conditions generator. The specimen dimensions are $L_x = 51.3\,\mu$m, $L_y = 51.3\,\mu$m and $L_z = 51.3\,\mu$m, Fig. 21. Two isotropic domains are involved, the carbon fibres with Young's modulus $E = 241$ GPa and Poisson ratio $\nu = 0.26$, and the epoxy with Young's modulus $E = 5.10$ GPa and Poisson ratio $\nu = 0.40$.

The specimen is fully constrained in all axes $u_i = 0$ at the Γ_1 surface. When nothing is defined by the user, the default boundary condition corresponds to a free surface $t_i = 0$ imposed in this model in the lateral surfaces. At the Γ_2 surface, a ramp $t_1 = t_3 = 2s$ MPa traction is applied, where s represents the current load step $0 < s \leq 50$. Then the maximum applied traction is 100 MPa. Finally, the free surface condition $t_2 = 0$ must be imposed to get a well posed problem.

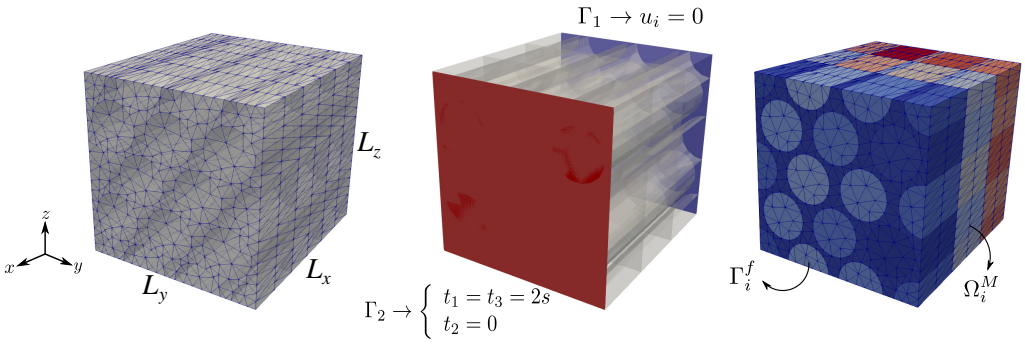


Figure 21: Composite model of 57 domains discretised with 8640 boundary elements.

Despite the fact the solid is a box specimen, this time the boundary conditions will be imposed through the General mesh generator. In Table 35, the `Set_parameters.f90` script located at the `~/BESLE/Mesh/General` root is configured to reproduce the mentioned boundary conditions.

```
SUBROUTINE Setup

    filename_in = 'Meshes/Obj/mesh.obj'
    filename_out = 'Input_data.dat'
    BCSfilesPlace_in = 'BCs/BCs_Obj/'

    AnalysisType="quasi-elastostatic"
    Nsteps=50

    NumBCS = 2

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0
    BC(1)%LParam%LinearBCyMaxVal=0
    BC(1)%LParam%LinearBCzMaxVal=0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="traction"
    BC(2)%BCyType="free"
    BC(2)%BCzType="traction"
    BC(2)%FuncxType="linear"
    BC(2)%FunczType="linear"
    BC(2)%LParam%LinearBCxMaxVal=100
    BC(2)%LParam%LinearBCzMaxVal=100

END SUBROUTINE Setup
```

Table 35: Customised boundary conditions.

If this is the first time using the carbon and epoxy elastic properties, the material package is employed for generating `"Mat_1.dat"` and `"Mat_2.dat"` material database for fibre and epoxy, respectively. For this purpose, the `Set_parameters.f90` script as shown in Table 36 is configured.

The mesh of this fibre-matrix composite shown in Figure 21 is composed of 57 domains, 18 fibres divided into 54 Γ_i^f sub-fibres and 3 additional Ω_i^M domains for the matrix for convenience, Fig. 21. It means, the material database requires to identify each domain and assign the correct elastic properties. For this, the file `List_Mat.dat` located at the `~/BESLE/Material/Data/Others/Composite/` root is used. It contains the list of numbers that correspond to each `"Mat_i.dat"` file.

Fibre	Epoxy

`SUBROUTINE Setup`
 	
`fileplace = "Data/Others/Composite/"`
`file_name = "Mat_1.dat"`
 	
`Material = 'Isotropic'`
 	
`E = 241000.d0;`
`nu = 0.26d0;`
 	
`END SUBROUTINE Setup`
	`SUBROUTINE Setup`
	
	`fileplace = "Data/Others/Composite/"`
	`file_name = "Mat_2.dat"`
	
	`Material = 'Isotropic'`
	
	`E = 5100.d0;`
	`nu = 0.40d0;`
	
	`END SUBROUTINE Setup`

Table 36: Fibre and epoxy material databases.

Now the configuration of BESLE `Set_parameters.f90` script is illustrated in Table 37. First, the `mesh_file` and `fileplace_mesh` variables to indicate the name and from where the mesh file should be read, respectively.

```
MODULE Set_parameters
!-------------------------------------------------------------!
USE Global_variables
!-------------------------------------------------------------!
CONTAINS
    SUBROUTINE Setup

        mesh_file = 'Input_data'
        fileplace_mesh = 'Mesh/General/Meshes/DAT/'

        n_Materials = 57
        material_coefficients_database = 'Composite'
        scale_size_1 = 0.001d0

        quasi_static = 1
        static_steps = 50

        bc_groups = 2
        fileplace_BCs = 'Mesh/General/BCs/BCs_DAT/'

        scale_results = 200
        results_file = 'Results'
        fileplace_results = 'Results/'

    END SUBROUTINE Setup
END MODULE Set_parameters
```

Table 37: Quasi-static simulation of a composite material.

According to the script in Table 35, there are 2 groups of boundary conditions being applied saved in the `BCs_i.dat` files located at the `fileplace_BCs` root. They are incorporated in the simulation using the `bc_groups` taking this number as its value. Figure 22 shows the ε_{11}, ε_{22}, and ε_{33} strain components. Also the displacement u_1 component at the 13 load step where $t_1 = t_3 = 20$ MPa.

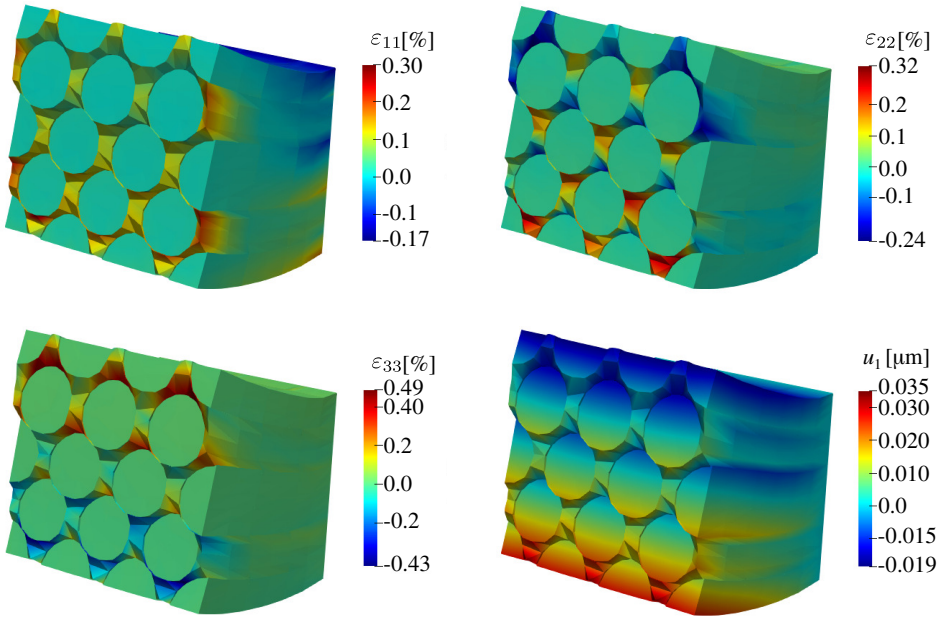


Figure 22: Strain and displacement components under $t_1 = t_3 = 20$ MPa.

An interesting fact is the stiffness ratio between these materials as the ε_{11}, ε_{22}, and ε_{33} strain components of the fibres are approximately zero. Then, the deformation is absorbed by the epoxy.

5.6 Vertebra L2 modelling

This example attempts to illustrate a realistic model of a vertebra L2 shown in Fig. 23. The mesh was modeled as follows: firstly, it is segmented the vertebra L2 from a fully anonymized CT of the 3DSlicer database[4] [22]. Thereafter, it was exported as `"*.stl"` and finally tuned into 3ds Max.

The model of the vertebra L2 is divided into the trabecular and cortical bones by segmenting the trabecular tissue from the CT and then, superimposing it with the vertebra model. The anisotropic properties of cortical bone need to be oriented regarding the normal direction of the bone surface. To approximate this, the cortical domain is split into 15 new domains, which are used to evaluate a trending normal direction of each region and then, oriented its stiffness tensor. The 15 cortical and the trabecular domains are shown in Figs. 5 and 23.

[4]https://www.slicer.org/wiki/CitingSlicer

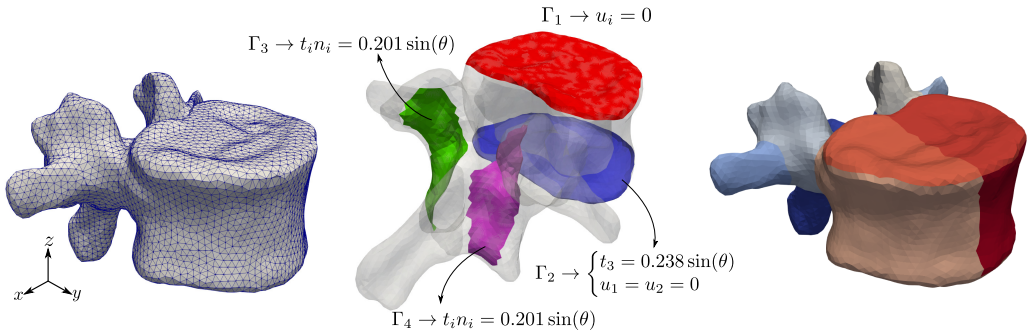


Figure 23: Vertebra L2 model of 15 domains discretised with 21612 boundary elements.

The boundary conditions to be applied to the vertebra L2 are set assuming that it supports an upper body weight of 54 kg, and the selection of the boundary elements is based on the loading described by Jong *et al.* [23]. Figure 23 shows the fully constrained at (Γ_1), the vertebral body at (Γ_2) and the articular processes at (Γ_3 and Γ_4). The BCs were prepared for a `"quasi-elastostatic"` simulation of 4 steps where the variable θ varies $\pi/2$ at each step load. In this case, the boundary conditions will be imposed through the General mesh generator. In Table 38, the `Set_parameters.f90` script located at the `~/BESLE/Mesh/General` root is configured to reproduce the mentioned boundary conditions.

```
SUBROUTINE Setup

    filename_in = 'Meshes/Obj/mesh.obj'
    filename_out = 'Input_data.dat'
    BCSfilesPlace_in = 'BCs/BCs_Obj/'

    AnalysisType="quasi-elastostatic"
    Nsteps=4

    NumBCS = 4
    MeshNumPress = 4

    BC(1)%DirType="xyz"
    BC(1)%BCxType="displacement"
    BC(1)%BCyType="displacement"
    BC(1)%BCzType="displacement"
    BC(1)%FuncxType="linear"
    BC(1)%FuncyType="linear"
    BC(1)%FunczType="linear"
    BC(1)%LParam%LinearBCxMaxVal=0
    BC(1)%LParam%LinearBCyMaxVal=0
    BC(1)%LParam%LinearBCzMaxVal=0

    BC(2)%DirType="xyz"
    BC(2)%BCxType="displacement"
    BC(2)%BCyType="displacement"
    BC(2)%BCzType="traction"
    BC(2)%FuncxType="linear"
    BC(2)%FuncyType="linear"
    BC(2)%FunczType="sine"
    BC(2)%LParam%LinearBCxMaxVal=0
    BC(2)%LParam%LinearBCyMaxVal=0
```

Table 38: Customised boundary conditions.

```
1
2       BC(2)%SParam%SineAz=0.268d0
3       BC(2)%SParam%SineBz=1.d0
4       BC(2)%SParam%SineCz=0.d0
5       BC(2)%SParam%SineSOz=0.d0
6       BC(2)%SParam%SineSfz=pi/2.d0
7
8       BC(3)%DirType="normal"
9       BC(3)%BCnType="traction"
10      BC(3)%FuncnType="sine"
11      BC(3)%SParam%SineAn=0.201d0
12      BC(3)%SParam%SineBn=1.d0
13      BC(3)%SParam%SineCn=0.d0
14      BC(3)%SParam%SineSOn=0.d0
15      BC(3)%SParam%SineSfn=pi/2.d0
16
17      BC(4)%DirType="normal"
18      BC(4)%BCnType="traction"
19      BC(4)%FuncnType="sine"
20      BC(4)%SParam%SineAn=0.201d0
21      BC(4)%SParam%SineBn=1.d0
22      BC(4)%SParam%SineCn=0.d0
23      BC(4)%SParam%SineSOn=0.d0
24      BC(4)%SParam%SineSfn=pi/2.d0
25
26  END SUBROUTINE Setup
```

Table 38: Customised boundary conditions. (continuation)

As mentioned, there are two materials involved here, the trabecular and cortical bones. The first one is incorporated in the model in its original reference orientation using the `Set_parameters.f90` script as shown in Table 39 from the material package.

```
1  SUBROUTINE Setup
2
3       fileplace = "Data/Others/Vertebra/"
4       file_name = "Mat_11.dat"
5
6       Material = 'Anisotropic'
7       Lattice = 'full'
8
9       C11=0.3822d0*(1e3); C12=0.0955d0*(1e3);
10      C13=0.0955d0*(1e3); C22=0.3822d0*(1e3);
11      C23=0.0955d0*(1e3); C33=0.3822d0*(1e3);
12      C44=0.1433d0*(1e3); C55=0.1433d0*(1e3);
13      C66=0.1433d0*(1e3);
14
15 END SUBROUTINE Setup
```

Table 39: trabecula material.

The second material corresponds to a list of $x-y-z$ orientations applied on the reference stifness tensor of the cortical bone. For this purpose, the `"Angles.dat"` file is provided with the rotation angles following the instructions in the `"1.info.txt"` file both located at the `~/BESLE/Material/Data/Others/Vertrbra` root. Finally, the `Set_parameters.f90` script as presented in Table 40 is configured.

```
SUBROUTINE Setup

    n_materials=11

    fileplace="Data/Others/Vertebra/"
    file_name="Angles.dat"

    Material='Anisotropic'
    Lattice='full'

    C11=23.8319d0*(1e3); C12=10.3887d0*(1e3);
    C13= 10.1493d0*(1e3); C14= 0.0221d0*(1e3);
    C15=  0.0205d0*(1e3); C16= -0.0277d0*(1e3);
    C22=22.4503d0*(1e3); C23=  9.6802d0*(1e3);
    C24= 0.0027d0*(1e3); C25= -0.0247d0*(1e3);
    C26= -0.0851d0*(1e3); C33= 26.0334d0*(1e3);
    C34= 0.0029d0*(1e3); C35=  0.0016d0*(1e3);
    C36=  0.0002d0*(1e3); C44= 5.5510d0*(1e3);
    C45=  0.0122d0*(1e3); C46= -0.0079d0*(1e3);
    C55=  5.6694d0*(1e3); C56=  0.0063d0*(1e3);
    C66=  6.0325d0*(1e3);

END SUBROUTINE Setup
```

Table 40: cortical material.

Now the configuration of BESLE `Set_parameters.f90` script is illustrated in Table 41. First, the `mesh_file` and `fileplace_mesh` variables to indicate the name and from where the mesh file should be read, respectively.

```
MODULE Set_parameters
!-------------------------------------------------------------!
USE Global_variables
!-------------------------------------------------------------!
CONTAINS

    SUBROUTINE Setup

        mesh_file = 'Input_data'
        fileplace_mesh = 'Mesh/General/Meshes/DAT/'

        n_Materials = 15
        material_coefficients_database = 'Vertebra'
        fileplace_material = "Material/Data/Others/"

        quasi_static = 1
        Static_steps = 4

        bc_groups = 4
        fileplace_BCs = 'Mesh/General/BCs/BCs_DAT/'

        scale_results = 10
        results_file = 'Results'
        fileplace_results = 'Results/'

    END SUBROUTINE Setup

END MODULE Set_parameters
```

Table 41: Simulation of a L2 vertebra.

The configuration of this simulation is similar to the last example for the composite material, the user can check the differences to update the script. Finally, results are presented in Fig. 24.

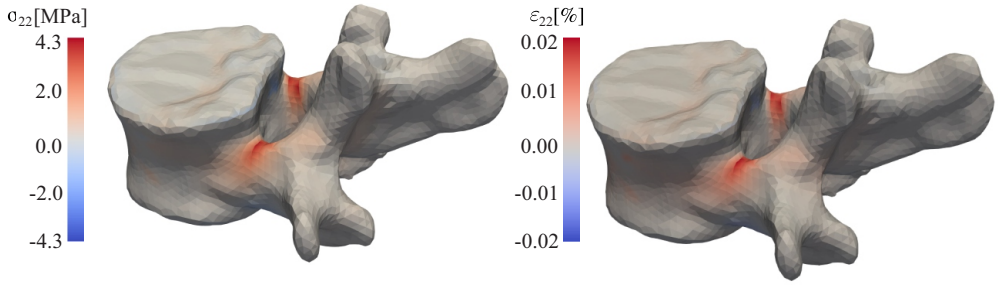


Figure 24: Vertebra L2 results at the four load step.

High levels of stress and strain were evidenced in the vertebral arch, specifically in the pedicle, and in the vertebral body. This is consistent with the results reported by Jong *et al.* [23]. Nevertheless, a stress concentration were observed in the subdomain interfaces due to stress shielding. Then, it is recommended, when anisotropic tissues are being modelled, to subdivide it into small pieces to grant a smooth transition of its elastic properties.

References

[1] A. F. Galvis, R. Q. Rodriguez, and P. Sollero, "Analysis of three-dimensional hexagonal and cubic polycrystals using the boundary element method," *Mechanics of Materials*, vol. 117, pp. 58–72, 2018.

[2] A. F. Galvis, *Multiscale Modeling of Dynamic Failure in 3D Polycrystalline Materials using BEM and MD*. PhD thesis, School of Mechanical Engineering, University of Campinas, 2019.

[3] I. Benedetti, M. H. Aliabadi, and G. Davì, "A fast 3D dual boundary element method based on hierarchical matrices," *International Journal of Solids and Structures*, vol. 45, pp. 2355–2376, 2008.

[4] R. Q. Rodríguez, C. L. Tan, P. Sollero, and E. L. Albuquerque, "Analysis of 3D anisotropic solids using fundamental solutions based on Fourier series and the adaptive cross approximation method," *Computer Modeling in Engineering & Sciences*, vol. 102, no. 5, pp. 359–372, 2014.

[5] A. M. Haider and M. Schanz, "Adaptive Cross Approximation for BEM in Elasticity," *Journal of Theoretical and Computational Acoustics*, vol. 2019, no. 1, p. 1850060, 2019.

[6] Z. Yao, X. Zheng, H. Yuan, and J. Feng, "Research progress of high-performance BEM and investigation on convergence of GMRES in local stress analysis of slender real thin-plate beams," *Engineering Computations*, vol. 36, no. 8, pp. 2530–2556, 2019.

[7] A. Ayala, X. Claeys, and L. Grigori, "Linear-time CUR approximation of BEM matrices," *Journal of Computational and Applied Mathematics*, vol. 368, p. 112528, 2020.

[8] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 15–41, 2001.

[9] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet, "Hybrid scheduling for the parallel solution of linear systems," *Parallel Computing*, vol. 32, no. 2, pp. 136–156, 2006.

[10] L. S. Moura, A. F. Galvis, E. L. Albuquerque, and P. Sollero, "Failure analysis of adhesively bonded composite joints using 3D BEM formulation," in *Advances in Boundary Element & Meshless Techniques*, vol. 20, pp. 128–134.

[11] L. S. Moura, "Numerical and experimental analysis of bonded composites," Master's thesis, University of Campinas, 2019.

[12] C. H. Rycroft, "Voro++: A three-dimensional Voronoi cell library in C++," *Chaos*, vol. 19, pp. 041111–1, 2009.

[13] J. R. Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*. Springer-Verlag, 1996.

[14] M. Kögl and L. Gaul, "A 3-D boundary element method for dynamic analysis of anisotropic elastic solids," *CMES: Computer Modeling in Engineering & Science*, vol. 1, no. 4, pp. 27–43, 2000.

[15] J. H. Kane, *Boundary Element Analysis In Engineering Continuum Mechanics*. New Yersey: Prentice-Hall, 1994.

[16] J. C. Houbolt, "A reccurrence matrix solution for the dynamic response of elastic aircraft," *Journal of Aeronautical and Science*, vol. 17 (9), pp. 540–550, 1950.

[17] J. Dominguez, *Boundary Elements in Dynamics*. Southampton: Computational Mechanics Publications, 1993.

[18] E. L. Albuquerque, P. Sollero, and M. H. Aliabadi, "The boundary element method applied to time dependent problems in anisotropic materials," *International Journal of Solids and Structures*, vol. 39, pp. 1405–1422, 2002.

[19] J. T. Katsikadelis, *Boundary Elements: Theory and Applications*. Oxford: Elsevier, 2002.

[20] D. M. Prada, A. F. Galvis, A. C. Alcântara, and P. Sollero, "3D Boundary element meshing for multiscale bone anisotropic analysis," *European Journal of Computational Mechanics*, vol. 27, no. 5-6, pp. 425–442, 2017.

[21] D. M. Prada, "Multiscale anisotropic elastostatic homogenization of healthy and osteoporotic bone tissue using 3D BEM," Master's thesis, University of Campinas, 2019.

[22] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J.-C. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, J. Buatti, S. Aylward, J. V. Miller, S. Pieper, and R. Kikinis, "3D Slicer as an image computing platform for the quantitative imaging network," *Magnetic Resonance Imaging*, vol. 30, no. 9, pp. 1323–41, 2012.

[23] J. K. Shin, T. S. Goh, M.-S. Kim, K. Kim, M. J. Shin, S. M. Son, H. J. Lee, J. S. Lee, and C.-S. Lee, "Stress and strain analyses of single and segmental lumbar spines based on an accurate finite element model for vertebrae," *Biomedical Research*, vol. 29, pp. 464–471, 2018.

[24] I. Benedetti and M. H. Aliabadi, "A three-dimensional grain boundary formulation for microstructural modeling of polycrystalline materials," *Computational Materials Science*, vol. 67, pp. 249–260, 2013.

[25] C. L. Tan, Y. C. Shiah, and C. Y. Wang, "Boundary element elastic stress analysis of 3D generally anisotropic solids using fundamental solutions based on Fourier series," *International Journal of Solids and Structures*, vol. 50, pp. 2701–2711, 2013.

[26] T. C. T. Ting and V. G. Lee, "The three-dimensional elastostatic Green's function for general anisotropic linear elastic solids," *The Quarterly Journal of Mechanics & Applied Mathematics*, vol. 50, pp. 407–426, 1997.

[27] J. L. Synge, *The Hypercircle in Mathematical Physics.* Cambridge University Press, Cambridge, 1957.

[28] D. M. Barnett, "The precise evaluation of derivatives of the anisotropic elastic Green's functions," *Physics State Solid*, vol. 49, pp. 741–748, 1972.

[29] Y. C. Shiah, C. L. Tan, and V. G. Lee, "Evaluation of explicit-form fundamental solutions for displacements and stresses in 3D anisotropic elastic solids," *CMES: Computer Modeling in Engineering & Science*, vol. 34, no. 3, pp. 205–226, 2008.

[30] A. F. Galvis, R. Q. Rodriguez, and P. Sollero, "Dynamic analysis of three-dimensional polycrystalline materials using the boundary element method," *Computers & Structures*, vol. 200, pp. 11–20, 2018.

[31] A. F. Galvis, P. A. Santos-Flórez, P. Sollero, M. de Koning, and L. C. Wrobel, "Multiscale model of the role of grain boundary structures in the dynamic intergranular failure of polycrystal aggregates," *Computer Methods in Applied Mechanics and Engineering*, vol. 362, p. 112868, 2020.

[32] R. Clough and J. Penzien, *Dynamic of Structures.* Berkeley: Computers & Structures, Inc, Third ed., 2003.

[33] L. Gaul, M. Kögl, and M. Wagner, *Boundary Element Method for Engineers and Scientists.* Berlin: Springer, 2003.