User manual for the `TopOptSLP3D` program: a Matlab implementation to efficiently solve three-dimensional structural topology optimization problems

Alfredo Vitorino
Francisco A. M. Gomes

Introduction

This is a beginner tutorial on how to use the MATLAB$^{©}$ implementation developed by Alfredo Vitorino and Francisco A. M. Gomes to solve three-dimensional structural topology optimization problems. More details about the algorithm can be found in the article "An efficient topology optimization algorithm for large-scale three-dimensional structures".

The text is organized as follows. Section 1 presents a summary of the topology optimization problem with the numerical methods employed in the algorithm and some implementation features. A download and setup guide can be consulted in Section 2. Sections 3 and 4 show how to use the two main files of the implementation: the function `EnterData`, which is used to input the problem data, and the script `TopOpt3D`, which is used to set up the algorithm parameters and run the program to solve the problem. Finally, in Section 5, we present brief descriptions of the other functions.

Contents

1 Problem description and implementation features

Structural topology optimization is a mathematical methodology that helps in selecting a suitable initial shape for a new structure, facilitating and speeding up its design. The aim is to decide how the material should be distributed within a domain so that the structure has the greatest possible rigidity, supporting the application of external loads without suffering excessive displacements and deformations, remaining in static equilibrium and satisfying a maximum volume constraint.

The essential data required to define a particular problem are: the domain in which the structure must be contained, the supports that keep the structure in equilibrium and the external loads applied. Eventually, the structure may also have predefined void or solid regions. All of these data can be inputed using the function `EnterData`, as described in Section 3. Other information, like the maximum volume of the structure and the algorithm parameters, can be defined in the script `TopOpt3D`, as explained in Section 4.

We apply the *finite element method* to discretize the domain and to approximate the displacements of the structure. We consider that the domain has the shape of a right rectangular prism and we use elements that also have this shape, called rectangular prismatic elements. The elements can be of the Lagrange and serendipity families, with shape functions of degree 1, 2 and 3. Our implementation also features an *adaptive strategy to increase the degree of the elements* whenever a more accurate approximation of the displacements is required.

The optimal topology of the structure is determined by the best distribution of material densities in the finite element mesh and the *SIMP model* is used to reduce the intermediate densities. To avoid the occurrence of checkerboard patterns, we apply a *weighted average density filter*. Furthermore, a very small positive value for the Young's modulus of the void is employed to prevent the global stiffness matrix from becoming singular.

The optimization problem is solved by a globally convergent *sequential linear programming (SLP)* method with a stopping criterion based on first-order optimality conditions.

The SLP method can be combined with a *multiresolution* scheme, that employs different discretizations to deal with displacement, design and density variables, allowing high-resolution structures to be obtained at a low computational cost.

For solving the equilibrium linear systems, it is possible to use the *Cholesky factorization* or the *preconditioned conjugate gradient method* with four available preconditioners: the diagonal of the global stiffness matrix, the incomplete Cholesky factorization of the global stiffness matrix, an *algebraic multigrid* method and a *geometric multigrid* method. It is also possible to solve the problem without explicitly generate the global stiffness matrix, using the alternative functions in the folder named "WOK".

Moreover, a *density projection strategy* based on the gradient vector information can be applied to obtain structures composed by fully solid or void regions.

2 Download and setup

The implementation files can be downloaded from this repository. In Matlab, open the folder with the files in the current path or add it to the search path of the current Matlab section. Then, you are ready to solve structural topology optimization problems.

Example

The so called MBB beam is a classic topology optimization problem. As illustrated in Figure 1, it has supports in the four bottom corners and a concentrated load applied downwards at the center of the top face of the domain. The structure is symmetric with respect to both the xy and the yz planes.

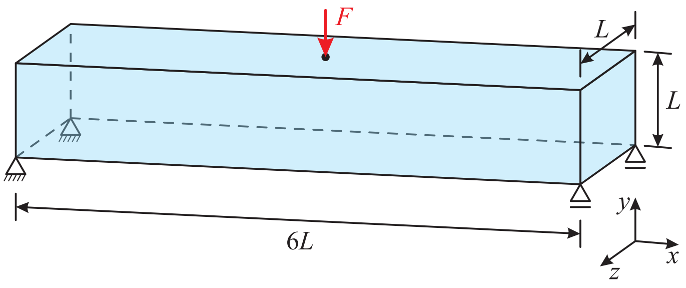
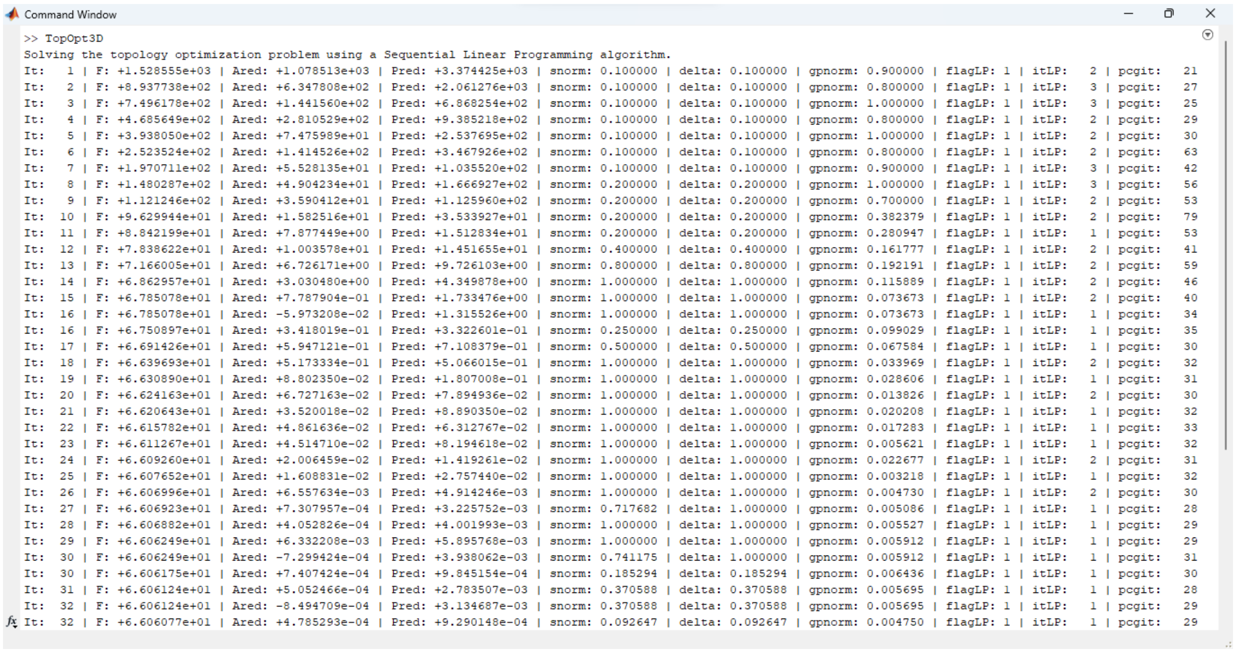


Figure 1: Initial design domain of the MBB beam problem.

Once you download the files, you can test the program in Matlab by running the script `TopOpt3D`. With the settled parameters, the program should load the problem data in the file ‘`mbb96x16x16quarter.mat`’ inside the folder “Tests” and solve this problem, which corresponds to a MBB beam problem with `96x16x16` elements in the mesh. Indeed, we consider only the symmetric part of the structure, i.e. one quarter of the structure with `48x16x8` elements.

A summary of the numerical results obtained at each iteration of the sequential linear programming algorithm will be displayed in the command window, as in Figure 2.



```
>> TopOpt3D
Solving the topology optimization problem using a Sequential Linear Programming algorithm.
It:   1 | F: +1.528555e+03 | Ared: +1.078513e+03 | Pred: +3.374425e+03 | snorm: 0.100000 | delta: 0.100000 | gpnorm: 0.900000 | flagLP: 1 | itLP:   2 | pcgit:   21
It:   2 | F: +8.937738e+02 | Ared: +6.347808e+02 | Pred: +2.061276e+03 | snorm: 0.100000 | delta: 0.100000 | gpnorm: 0.800000 | flagLP: 1 | itLP:   3 | pcgit:   27
It:   3 | F: +7.496178e+02 | Ared: +1.441560e+02 | Pred: +6.868254e+02 | snorm: 0.100000 | delta: 0.100000 | gpnorm: 1.000000 | flagLP: 1 | itLP:   3 | pcgit:   25
It:   4 | F: +4.685649e+02 | Ared: +2.810529e+02 | Pred: +9.385218e+02 | snorm: 0.100000 | delta: 0.100000 | gpnorm: 0.800000 | flagLP: 1 | itLP:   2 | pcgit:   29
It:   5 | F: +3.938050e+02 | Ared: +7.475989e+01 | Pred: +2.537695e+02 | snorm: 0.100000 | delta: 0.100000 | gpnorm: 1.000000 | flagLP: 1 | itLP:   2 | pcgit:   30
It:   6 | F: +2.523524e+02 | Ared: +1.414526e+02 | Pred: +3.467926e+02 | snorm: 0.100000 | delta: 0.100000 | gpnorm: 0.800000 | flagLP: 1 | itLP:   2 | pcgit:   63
It:   7 | F: +1.970711e+02 | Ared: +5.528135e+01 | Pred: +1.035520e+02 | snorm: 0.100000 | delta: 0.100000 | gpnorm: 0.900000 | flagLP: 1 | itLP:   3 | pcgit:   42
It:   8 | F: +1.480287e+02 | Ared: +4.904234e+01 | Pred: +1.666927e+02 | snorm: 0.200000 | delta: 0.200000 | gpnorm: 1.000000 | flagLP: 1 | itLP:   3 | pcgit:   56
It:   9 | F: +1.121246e+02 | Ared: +3.590412e+01 | Pred: +1.125960e+02 | snorm: 0.200000 | delta: 0.200000 | gpnorm: 0.700000 | flagLP: 1 | itLP:   2 | pcgit:   53
It:  10 | F: +9.629944e+01 | Ared: +1.582516e+01 | Pred: +3.533927e+01 | snorm: 0.200000 | delta: 0.200000 | gpnorm: 0.382379 | flagLP: 1 | itLP:   2 | pcgit:   79
It:  11 | F: +8.842199e+01 | Ared: +7.877449e+00 | Pred: +1.512834e+01 | snorm: 0.200000 | delta: 0.200000 | gpnorm: 0.280947 | flagLP: 1 | itLP:   1 | pcgit:   53
It:  12 | F: +7.838622e+01 | Ared: +1.003578e+01 | Pred: +1.451655e+01 | snorm: 0.400000 | delta: 0.400000 | gpnorm: 0.161777 | flagLP: 1 | itLP:   2 | pcgit:   41
It:  13 | F: +7.166005e+01 | Ared: +6.726171e+00 | Pred: +9.726103e+00 | snorm: 0.800000 | delta: 0.800000 | gpnorm: 0.192191 | flagLP: 1 | itLP:   2 | pcgit:   59
It:  14 | F: +6.862957e+01 | Ared: +3.030480e+00 | Pred: +4.349878e+00 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.115889 | flagLP: 1 | itLP:   2 | pcgit:   46
It:  15 | F: +6.785078e+01 | Ared: +7.787904e-01 | Pred: +1.733476e+00 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.073673 | flagLP: 1 | itLP:   2 | pcgit:   40
It:  16 | F: +6.785078e+01 | Ared: -5.973208e-02 | Pred: +1.315526e+00 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.073673 | flagLP: 1 | itLP:   1 | pcgit:   34
It:  16 | F: +6.750897e+01 | Ared: +3.418019e-01 | Pred: +3.322601e-01 | snorm: 0.250000 | delta: 0.250000 | gpnorm: 0.099029 | flagLP: 1 | itLP:   1 | pcgit:   35
It:  17 | F: +6.691426e+01 | Ared: +5.947121e-01 | Pred: +7.108379e-01 | snorm: 0.500000 | delta: 0.500000 | gpnorm: 0.067584 | flagLP: 1 | itLP:   1 | pcgit:   30
It:  18 | F: +6.639693e+01 | Ared: +5.173334e-01 | Pred: +5.066015e-01 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.033969 | flagLP: 1 | itLP:   2 | pcgit:   32
It:  19 | F: +6.630890e+01 | Ared: +8.802350e-02 | Pred: +1.807008e-01 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.028606 | flagLP: 1 | itLP:   1 | pcgit:   31
It:  20 | F: +6.624163e+01 | Ared: +6.727163e-02 | Pred: +7.894936e-02 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.013826 | flagLP: 1 | itLP:   2 | pcgit:   30
It:  21 | F: +6.620643e+01 | Ared: +3.520018e-02 | Pred: +8.890350e-02 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.020208 | flagLP: 1 | itLP:   1 | pcgit:   32
It:  22 | F: +6.615782e+01 | Ared: +4.861636e-02 | Pred: +6.312767e-02 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.017283 | flagLP: 1 | itLP:   1 | pcgit:   33
It:  23 | F: +6.611267e+01 | Ared: +4.514710e-02 | Pred: +8.194618e-02 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.005621 | flagLP: 1 | itLP:   1 | pcgit:   32
It:  24 | F: +6.609260e+01 | Ared: +2.006459e-02 | Pred: +1.419261e-02 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.022677 | flagLP: 1 | itLP:   2 | pcgit:   31
It:  25 | F: +6.607652e+01 | Ared: +1.608831e-02 | Pred: +2.757440e-02 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.003218 | flagLP: 1 | itLP:   1 | pcgit:   32
It:  26 | F: +6.606996e+01 | Ared: +6.557634e-03 | Pred: +4.914246e-03 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.004730 | flagLP: 1 | itLP:   2 | pcgit:   30
It:  27 | F: +6.606923e+01 | Ared: +7.307957e-04 | Pred: +3.225752e-03 | snorm: 0.717682 | delta: 1.000000 | gpnorm: 0.005086 | flagLP: 1 | itLP:   1 | pcgit:   28
It:  28 | F: +6.606882e+01 | Ared: +4.052826e-04 | Pred: +4.001993e-03 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.005527 | flagLP: 1 | itLP:   1 | pcgit:   29
It:  29 | F: +6.606249e+01 | Ared: +6.332208e-03 | Pred: +5.895768e-03 | snorm: 1.000000 | delta: 1.000000 | gpnorm: 0.005912 | flagLP: 1 | itLP:   1 | pcgit:   29
It:  30 | F: +6.606249e+01 | Ared: -7.299424e-04 | Pred: +3.938062e-03 | snorm: 0.741175 | delta: 1.000000 | gpnorm: 0.005912 | flagLP: 1 | itLP:   1 | pcgit:   31
It:  30 | F: +6.606175e+01 | Ared: +7.407424e-04 | Pred: +9.845154e-04 | snorm: 0.185294 | delta: 0.185294 | gpnorm: 0.006436 | flagLP: 1 | itLP:   1 | pcgit:   30
It:  31 | F: +6.606124e+01 | Ared: +5.052466e-04 | Pred: +2.783507e-03 | snorm: 0.370588 | delta: 0.370588 | gpnorm: 0.005695 | flagLP: 1 | itLP:   1 | pcgit:   28
It:  32 | F: +6.606124e+01 | Ared: -8.494709e-04 | Pred: +3.134687e-03 | snorm: 0.370588 | delta: 0.370588 | gpnorm: 0.005695 | flagLP: 1 | itLP:   1 | pcgit:   29
It:  32 | F: +6.606077e+01 | Ared: +4.785293e-04 | Pred: +9.290148e-04 | snorm: 0.092647 | delta: 0.092647 | gpnorm: 0.004750 | flagLP: 1 | itLP:   1 | pcgit:   29
```

Figure 2: Matlab command window when solving the `mbb96x16x16` problem.

When the program finishes solving the problem, it will open a figure window with a 3D view of the optimal structure obtained, as shown in Figure 3.

The folder “Tests” also contains a variety of other predefined examples that can be tested. The three-dimensional problems include: the cantilever beam (`cb`), the MBB beam (`mbb`), the transmission tower (`tt`), the engine bracket (`eb`), the half cube with concentrated load (`hc`), the building with torsion (`bt`), the L-shaped beam (`ls`) and the bridge (`bd`).

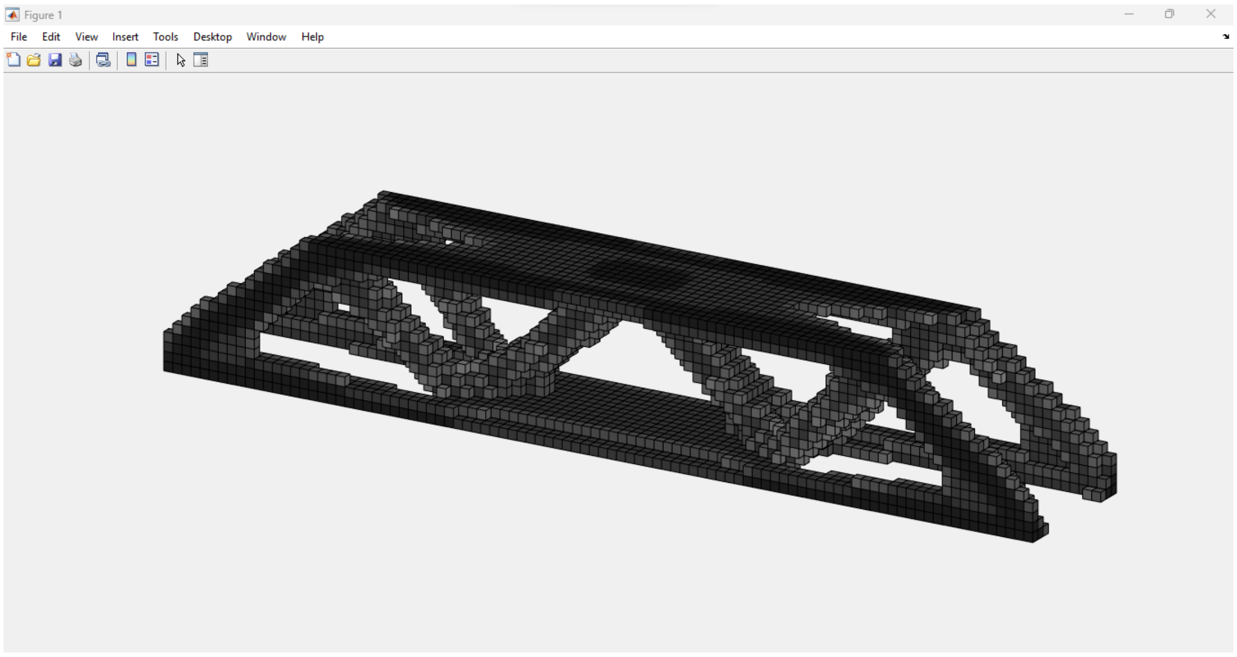


Figure 3: Optimal structure obtained for the `mbb96x16x16` problem.

For each test, it is necessary to change the line `load('Tests/mbb96x16x16quarter.mat')` in script `TopOp3D` with the proper file name, as well as change other parameters, like the maximum volume that the structure can have (`volfrac`) and the filter radius (`rmin`), as desired. A discussion about each parameter that can be defined in script `TopOpt3D` will be presented in Section 4. Next, we show how you can create your own problem data.

3 The function `EnterData`

To input a new problem data, you can use the function called `EnterData`. This function is used to create the variables with information about the domain where the structure must be contained, the finite element mesh, the material properties, the supports, the external loads and the predefined void or solid regions.

Once you run the function `EnterData` in Matlab, the program will wait until you input the variables, one at a time, in the command window.

Dimensions and material properties

First, you should inform the domain dimensions.

```
» Length:

» Height:

» Width:
```

For each dimension, type a positive real number and press “Enter” on your keyboard.

Then, you should inform the finite element mesh dimensions.

```
» Number of elements in the length(x) direction:

» Number of elements in the height(y) direction:

» Number of elements in the width(z) direction:
```

These must be positive integer numbers.

The next information required are some material properties.

```
» Young’s modulus:

» Poisson’s ratio:
```

The Young’s modulus must be a positive real number and the Poisson’s ratio must be a real number between 0 and 1.

Supports

Next, you should enter the data about the supports. We remark that it is necessary to enter at least one support and the supports must prevent the movement in each direction (x, y and z) of at least one point of the domain, so that the structure can remain in equilibrium and the problem can be solved.

Before entering each support data, you will be asked the following question.

```
» Continue adding supports? (1 = Yes, 0 = No)
```

Input the number 1 to add a new support or the number 0 to stop adding supports to the problem. To insert a new support, you first need to inform the geometric position of the support within the domain.

```
» Initial x-coordinate of the support:

» Final x-coordinate of the support:

» Initial y-coordinate of the support:

» Final y-coordinate of the support:

» Initial z-coordinate of the support:

» Final z-coordinate of the support:
```

These support coordinates are related to the dimensions (length, height and width) previously defined for the domain. The x-coordinate must be a real number from 0 to the domain length, the y-coordinate must be a real number from 0 to the domain height and the z-coordinate must be a real number from 0 to the domain width. If the support is not distributed in some direction, just input the same initial and final coordinates in that direction.

After, you should inform whether or not the support prevent the movement in each direction.

```
    » Does the support prevent the movement in the x direction?
      (1 = Yes, 0 = No)

    » Does the support prevent the movement in the y direction?
      (1 = Yes, 0 = No)

    » Does the support prevent the movement in the z direction?
      (1 = Yes, 0 = No)
```

Input the number 1 if the support prevent the movement in the given direction or the number 0 otherwise.

External loads

The next requested data are the external loads applied. As in the case of supports, before entering each load data, you will be asked the following question.

```
    » Continue adding loads? (1 = Yes, 0 = No)
```

Input the number 1 to add a new load or the number 0 to stop adding loads to the problem. To insert a new load, you first need to inform the load vector components, which gives the load intensity in each direction.

```
    » x-component of load:

    » y-component of load:

    » z-component of load:
```

These values can be any real number.

Then, you should inform whether or not the load is distributed in each direction.

```
    » Is the load distributed in the x direction? (1 = Yes, 0 = No)

    » Is the load distributed in the y direction? (1 = Yes, 0 = No)

    » Is the load distributed in the z direction? (1 = Yes, 0 = No)
```

Again, input the number 1 to answer "yes" or the number 0 to answer "no" in each case. We note that our implementation cannot deal with volumetric loads, i.e. loads distributed in all three directions at the same time.

Next, you should inform the geometric position of the load within the domain. If the load is distributed in a given direction, you will be asked for the initial and final coordinates of the load application. Otherwise, you just need to input one coordinate of the load application in that direction. For instance, if the load is distributed only in the x direction, the inputs will be as follows.

```
    » Initial x-coordinate of load application:

    » Final x-coordinate of load application:

    » y-coordinate of load application:

    » z-coordinate of load application:
```

The x-coordinate must be a real number from 0 to the domain length, the y-coordinate must be a real number from 0 to the domain height and the z-coordinate must be a real number from 0 to the domain width.

Void or solid regions

If the problem has predefined void or solid regions in the domain, you can inform them next. Before entering each void region, you will be asked the following question.

```
    » Continue adding void regions? (1 = Yes, 0 = No)
```

Input the number 1 to add a new void region or the number 0 to stop adding void regions to the problem. If you wish to add a void region, then you should inform the geometric position of that region within the domain as done for the supports.

```
    » Initial x-coordinate of the void region:

    » Final x-coordinate of the void region:

    » Initial y-coordinate of the void region:

    » Final y-coordinate of the void region:

    » Initial z-coordinate of the void region:

    » Final z-coordinate of the void region:
```

The way to insert solid regions is the same as to insert void regions. Before entering each solid region, you will be asked the following question.

```
    » Continue adding solid regions? (1 = Yes, 0 = No)
```

Input the number 1 to add a new solid region or the number 0 to stop adding solid regions to the problem. If you wish to add a solid region, then you should inform the geometric position of that region within the domain as done for the supports.

```
» Initial x-coordinate of the solid region:

» Final x-coordinate of the solid region:

» Initial y-coordinate of the solid region:

» Final y-coordinate of the solid region:

» Initial z-coordinate of the solid region:

» Final z-coordinate of the solid region:
```

For void and solid regions, the x-coordinate must be a real number from 0 to the domain length, the y-coordinate must be a real number from 0 to the domain height and the z-coordinate must be a real number from 0 to the domain width.

Finally, you can save the variables in a '`.mat`' file and load it substituting the line `load('Tests/mbb96x16x16quarter.mat')` with the proper file name in script `TopOpt3D` to be able to solve the new problem.

4 The script `TopOpt3D`

In addition to the problem data previously defined using the function `EnterData`, the algorithm parameters and options can be defined by changing the values of the variables in the script `TopOpt3D`. Tables 1 – 9 show all variables that can be changed, a brief description and the possible values that can be assigned to each variable.

Table 1: General parameters and options defined in the script `TopOpt3D`.

Variable	**Description**	**Possible values**
`volfrac`	Maximum volume fraction that the structure can occupy	Real number between 0 and 1
`rmin`	Filter radius	Positive real number
`p`	Penalty parameter for the SIMP model	Positive real number
`p123`	Defines if the continuation strategy will be used to adapt the parameter `p`	• `true`: set `p` to 1, 2 and 3 • `false`: use only one value of `p`
`femin`	Stiffness factor of the void material ($E_{min} = \texttt{femin} \times E_0$)	Positive real number (close to 0)
`opfilter`	Filter option	• `0`: no filter • `1`: weighted average density filter • `2`: average density filter
`volineq`	Defines the type of the volume constraint	• `true`: less than or equal to • `false`: equal to

Table 2: Finite element parameters and options defined in the script `TopOpt3D`.

Variable	**Description**	**Possible values**
`elem.deg`	Element degree	1, 2 or 3
`elem.type`	Element type	• `1`: Lagrange • `2`: serendipity
`elem.increasedeg`	Defines if the adaptive strategy to increase the degree of the elements will be used	• `true`: increase the degree of the elements • `false`: do not increase the degree of the elements
`elem.maxdeg`	Maximum element degree	1, 2 or 3
`elem.fix`	Defines the way to choose variables to be fixed when increasing the element degree	• `0`: do not choose any variable • `1`: choose at the first iteration • `2`: choose in all iterations • `3`: choose every `elem.fixit` iterations • `4`: choose at the first iteration and at iteration `elem.fixit`
`elem.fixit`	Number of outer iterations to be accomplished until choose again the variables that will be fixed	Positive integer number
`elem.fixtl`	Tolerance to find void elements	Positive real number (close to 0)
`elem.fixtu`	Tolerance to find solid elements	Positive real number (close to 1)
`elem.fixnb`	Defines if all void elements will be considered when suppressing variables from the problem	• `true`: choose all void elements • `false`: choose only the ones surrounded by void elements
`elem.fixdofs`	Defines if the displacements of void elements will be suppressed from the problem	• `0`: do not fix displacements • `1`: fix the displacements of nodes not shared by neighbors • `2`: fix the displacements of all nodes of the void elements
`elem.fixdsgn`	Defines the way to choose void elements	• `true`: look for the ones full of approximately 0 design variables • `false`: look for the ones full of approximately 0 densities
`elem.ltdsgn`	Max. value a neighbor may have when fixing design variables to 0	Real number between 0 and 1
`elem.utdsgn`	Max. value a neighbor may have when fixing design variables to 1	Real number between 0 and 1
`elem.tolgdsgn`	Tolerance for the gradient when fixing design variables	Positive real number (close to 0)

Table 3: SLP algorithm parameters and options defined in the script `TopOpt3D`.

Variable	**Description**	**Possible values**
`slp.tolS`	Threshold for the step norm	Positive real number (close to 0)
`slp.tolG`	Threshold for the projected gradient norm	Positive real number (close to 0)
`slp.tolF`	Threshold for the actual reduction of the objective function	Positive real number (close to 0)
`slp.maxcount`	Number of times that the projected gradient norm (or step norm) needs to reach a value less than or equal to the threshold before stop the algorithm	Positive integer number
`slp.maxiter`	Maximum number of outer iterations	Positive integer number
`slp.maxit12`	Maximum number of outer iterations when using the continuation strategy (`p123 = true`) with the SIMP penalty parameter `p` equals to 1 and 2	Positive integer number
`slp.lpsolver`	Linear programming solver	• `0`: dual simplex • `1`: interior point method
`slp.delta0`	Initial trust region radius	Positive real number
`slp.eta`	Parameter used to accept the step (it is accepted if $A_{red} \geq$ `slp.eta` $\times P_{red}$)	Real number between 0 and 1
`slp.rho`	Parameter used to increase the trust region radius (δ is increased if $A_{red} \geq$ `slp.rho` $\times P_{red}$)	Real number between 0 and 1
`slp.alphaR`	Factor used to decrease the trust region radius when the step is rejected	Positive real number
`slp.alphaA`	Factor used to increase the trust region radius when the step is accepted	Positive real number

Table 4: Options for considering symmetries defined in the script `TopOpt3D`.

Variable	**Description**	**Possible values**
`sym.xy`	Defines if the problem domain has a symmetry with respect to the xy plane	• `true`: consider the symmetry • `false`: do not consider the symmetry
`sym.yz`	Defines if the problem domain has a symmetry with respect to the yz plane	• `true`: consider the symmetry • `false`: do not consider the symmetry
`sym.xz`	Defines if the problem domain has a symmetry with respect to the xz plane	• `true`: consider the symmetry • `false`: do not consider the symmetry

Table 5: Density projection strategy parameters and options defined in the script `TopOpt3D`.

Variable	**Description**	**Possible values**
`prj.op`	Defines if the projection strategy will be applied to round the densities to 0 or 1 at the end of the SLP algorithm	• `true`: use the density projection • `false`: do not use the density projection
`prj.nearlim`	Defines if the densities will be rounded to the nearest limit prior to using the gradient strategy	• `true`: round the densities • `false`: don't round the densities
`prj.maxang`	Largest angle (in degrees) allowed to verify the angle condition	Real number between 0 and 90
`prj.maxit`	Max. number of projection attempts	Positive integer number
`prj.tolV`	Tolerance for the volume constraint violation after the density projection	Positive real number
`prj.tolN`	Tolerance for the change between consecutive projected solutions	Positive real number
`prj.opfilter`	Filter used after the projection	• `0`: no filter • `1`: weighted average density filter • `2`: average density filter
`prj.rmin`	Filter radius after the projection	Positive real number
`prj.gthresh`	Gradient threshold	Positive real number (close to 0)
`prj.ut`	Upper limit threshold (densities greater than or equal to `prj.ut` are always rounded to 1)	Real number between 0 and 1
`prj.lt`	Lower limit threshold (densities lower than or equal to `prj.lt` are always rounded to 0)	Real number between 0 and 1
`prj.vumin`	Smallest density value that can be rounded to 1	Real number between 0 and 1
`prj.vlmax`	Largest density value that can be rounded to 0	Real number between 0 and 1
`prj.heavi`	Defines if the smooth Heaviside projection will be applied to round the densities	• `true`: use the Heaviside projection • `false`: do not use the Heaviside projection
`prj.eta`	Initial value of the parameter η used to compute the Heaviside function	Positive real number
`prj.beta`	Initial value of the parameter β used to compute the Heaviside function	Positive real number
`prj.dbeta`	Factor to increase the parameter β	Positive real number
`prj.betamax`	Max. value of the parameter β	Positive real number
`prj.deg`	Defines if linear finite elements will be used on the calls to the algorithm after projecting the densities	• `true`: use linear elements • `false`: keep the original element degree

Table 6: Linear system solver parameters and options defined in the script `TopOpt3D`.

Variable	**Description**	**Possible values**
`opsolver`	Linear system solver option	• `0`: Cholesky factorization • `1`: Conjugate gradient method (PCG) with preconditioner being the diagonal of the matrix • `2`: PCG with preconditioner being the incomplete Cholesky factorization • `3`: PCG with preconditioner being the algebraic multigrid method • `4`: PCG with preconditioner being the geometric multigrid method
`pcgp.opu0`	Defines if the previous solution will be adopted as an initial guess for the conjugate gradient method	• `true`: use the previous solution as initial guess • `false`: use the zero vector as initial guess
`pcgp.tolPCG`	Tolerance for the convergence of the conjugate gradient method	Positive real number (close to 0)
`pcgp.maxiterPCG`	Max. number of iterations for the conjugate gradient method	Positive integer number
`genK`	Defines if the global stiffness matrix will be explicitly generated	• `true`: explicitly generate the global stiffness matrix • `false`: do not generate the global stiffness matrix

Table 7: Multiresolution parameters and options defined in the script `TopOpt3D`.

Variable	**Description**	**Possible values**
`mr.op`	Defines if the multiresolution method will be used	• `true`: use multiresolution • `false`: do not use multiresolution
`mr.n`	Number of density elements on each direction of each displacement element	Positive integer number
`mr.d`	Number of design variables on each direction of each displacement element	Positive integer number
`mr.x0`	Defines if the solution of the problem on the coarsest mesh will be used as initial guess for multiresolution	• `true`: use the solution on the coarsest mesh • `false`: do not use the solution on the coarsest mesh
`mr.interp`	Defines if the displacements will be interpolated to calculate the gradient	• `true`: interpolate the displacements • `false`: do not interpolate the displacements

Table 8: Multigrid parameters and options defined in the script `TopOpt3D`.

Variable	**Description**	**Possible values**
`mg.ngrids`	Number of grids	Positive integer number
`mg.cycle`	Cycle type	• `0`: V cycle • `1`: W cycle • `2`: Full multigrid V cycle
`mg.smoother`	Smoother method	• `0`: Jacobi • `1`: Gauss-Seidel (SOR) • `2`: SSOR
`mg.omega`	Smoother relaxation parameter	Real number in $[0, 2)$
`mg.smoothiter1`	Number of pre-smooth iterations	Positive integer number
`mg.smoothiter2`	Number of post-smooth iterations	Positive integer number
`mg.tolMG`	Tolerance for the conjugate gradient method with multigrid	Positive real number (close to 0)
`mg.maxiterMG`	Maximum number of iterations for the PCG method with multigrid	Positive integer number
`mg.theta`	Parameter to calculate the strength of connection in algebraic multigrid	Positive real number
`mg.optheta`	Defines how `mg.theta` will be used	• `0`: use `mg.theta` as a tolerance • `1`: use `mg.theta` as the maximum number of strong connections per node
`mg.nt`	Number of test space vectors in algebraic multigrid	Positive integer number
`mg.tolr`	Tolerance for the test space generation	Positive real number (close to 0)
`mg.tolq`	Tolerance for the test space generation	Positive real number (close to 0)
`mg.itp`	Maximum number of iterations in the prolongation DPLS	Positive integer number
`mg.kappa`	Tolerance for the condition number in the prolongation DPLS	Positive real number
`mg.opcond`	Preconditioner option for the test space generation and for PCG to solve the system on the coarsest grid	• `0`: diagonal • `1`: incomplete Cholesky factorization
`mg.nmaxchol`	Maximum system dimension to be able to use the Cholesky factorization on the coarsest grid	Positive integer number
`mg.opchol`	Action adopted when the dimension of the system on the coarsest grid is greater than `mg.nmaxchol`	• `0`: increase `mg.ngrids` • `1`: use the conjugate gradient method
`mg.Anbatch`	Batch used in the computation of the matrix `An` when `genK = false`	Positive integer number

Table 9: Options to calculate extra results defined in the script `TopOpt3D`.

Variable	Description	Possible values
`exop.correctF`	Calculate the objective function value on the finest mesh with linear elements when multiresolution or an element degree greater than 1 is used	`true` or `false`
`exop.postopt`	Solve the problem again on the finest mesh with the multiresolution solution as initial guess	`true` or `false`
`exop.roundsol`	Round the greatest densities to 1 until filling the volume and the other densities to 0, then calculate the objective function value of the rounded solution	`true` or `false`
`exop.solidsol`	Calculate the objective function of the fully solid structure (with all densities equal to 1)	`true` or `false`

Default and recommended parameters

The default parameters and options settled in the script `TopOpt3D` are the following. These are recommended values to solve the problem `mbb96x16x16` as in the example of Section 2.

```
volfrac = 0.2;
rmin = 1.5;
p = 3;
p123 = false;
femin = 1.0e-6;
opfilter = 1;
volineq = true;
elem.deg = 1;
elem.type = 1;
elem.increasedeg = false;
elem.maxdeg = 2;
elem.fix = 4;
elem.fixit = 5;
elem.fixtl = 1e-6;
elem.fixtu = 0.9;
elem.fixnb = false;
elem.fixdofs = 1;
elem.fixdsgn = false;
elem.ltdsgn = 0.3;
elem.utdsgn = 0.7;
elem.tolgdsgn = 1e-6;
slp.tolS = 1e-4;
slp.tolG = 1e-3;
slp.tolF = 5e-2;
slp.maxcount = 3;
slp.maxiter = 500;
slp.maxit12 = 15;
slp.lpsolver = 0;
slp.delta0 = 0.1;
slp.eta = 0.1;
slp.rho = 0.5;
slp.alphaR = 0.25;
slp.alphaA = 2.0;
prj.op = false;
prj.nearlim = true;
prj.maxang = 89.9;
prj.maxit = 10;
prj.tolV = 0.005;
prj.tolN = 0.01;
prj.opfilter = 1;
prj.rmin = 1.1;
prj.gthresh = 1e-4;
prj.ut = 0.95;
prj.lt = 0.05;
prj.vumin = 0.7;
prj.vlmax = 0.3;
prj.heavi = true;
prj.eta = 0.25;
prj.dbeta = 2.0;
prj.beta = 1.0;
prj.betamax = 1000.0;
prj.deg = false;
sym.xy = true;
sym.yz = true;
sym.xz = false;
opsolver = 4;
pcgp.opu0 = true;
pcgp.tolPCG = 1e-8;
pcgp.maxiterPCG = 10000;
genK = true;
mg.ngrids = 3;
mg.cycle = 1;
mg.smoother = 0;
mg.omega = 0.5;
mg.smoothiter1 = 1;
mg.smoothiter2 = 1;
mg.tolMG = 1e-8;
mg.maxiterMG = 5000;
mg.theta = 20;
mg.optheta = 1;
mg.nt = 30;
mg.tolr = 1e-1;
mg.tolq = 1e-2;
mg.itp = 2;
mg.kappa = 100;
mg.opcond = 1;
mg.nmaxchol = 60000;
mg.opchol = 0;
mg.Anbatch = 4096;          mr.x0 = false;              exop.roundsol = false;
mr.op = false;              mr.interp = false;          exop.solidsol = false;
mr.n = 2;                   exop.correctF = false;
mr.d = 2;                   exop.postopt = false;
```

The parameters highlighted in blue may be changed depending on the problem to be solved. These parameters were changed to perform the different tests presented in the article "An efficient topology optimization algorithm for large-scale three-dimensional structures".

To obtain high-resolution structures, it is recommended to use the multiresolution scheme setting `mr.op = true`. For example, to obtain the MBB beam with `288x48x48` density elements, either load the file '`mbb288x48x48quarter.mat`' and solve this problem with the traditional method or load the file '`mbb96x16x16quarter.mat`' and solve this problem with multiresolution setting `mr.op = true` and `mr.n = 3`. The second option is more efficient.

It is also recommended to set `elem.increasedeg = true` in order to obtain better quality solutions with the multiresolution method if the filter radius is small (`rmin < 1`). Furthermore, you can set `prj.op = true` to obtain the final structure with fully solid or void regions.

5 Other functions

The implementation covers all of the structural topology optimization process, including finite element analysis, filter application, sequential linear programming method, multigrid methods, multiresolution, density projection strategies and more. The files `EnterData` and `TopOpt3D` were explained in Sections 3 and 4, respectively. Short descriptions of the other functions of the program (in alphabetical order) are given bellow.

- `ApplyFilter`: applies the filter to the density elements.
- `Coarsening`: constructs the coarse grid for the algebraic multigrid method.
- `ConvertDofs`: adapts the number of nodes, indexes of the degrees of freedom and the vector of nodal loads according to the polynomial degree of the shape functions.
- `ConvertDofsMR`: adapts the number of nodes, indexes of the degrees of freedom and the vector of nodal loads to the density mesh used in multiresolution.
- `CorrectFunctionValue`: calculates the "correct" objective function value, solving the equilibrium linear system on the finest mesh with linear finite elements. This may be used to better compare the function value obtained by multiresolution with that obtained by the traditional method.
- `DegreesOfFreedom`: obtains the indexes of the degrees of freedom for each element.
- `Display3D`: creates a 3D view of the optimal structure using the element densities vector.
- `Elem2Fix`: choose the elements that will have fixed variables when solving the problem again increasing the element degree in the adaptive strategy.
- `ElemNeighborhood`: finds the neighborhood and the weight factors of each finite element, for the filter application.

- `ElemNeighborhoodMR`: finds the neighborhood between the density elements and design variable elements, for the projection of design variables to densities in multiresolution.
- `ElemStiffMatrix`: constructs the finite element stiffness matrix analytically (works only for linear elements).
- `ElemStiffMatrixGQ`: constructs the element stiffness matrix using Gaussian quadrature.
- `ElemStiffMatrixGQMR`: calculates the stiffness integrands using Gaussian quadrature to numerically construct the element stiffness matrix for multiresolution.
- `ExtraResults`: gathers additional results based on the solution obtained by the algorithm.
- `FMVCycle`: applies the full multigrid V-Cycle recursively.
- `GlobalStiffMatrix`: constructs the global stiffness matrix.
- `Grad`: calculates the gradient vector of the objective function.
- `GradProj`: applies the density projection strategy to round the densities to 0 or 1.
- `HeavisideProj`: updates the β and η parameters used to compute the Heaviside projection of the density vector.
- `HeavRoot`: calculates the value of the Heaviside function.
- `InterpolateDisp`: approximates the nodal displacements for a density element using interpolation of the nodal displacements on the coarsest mesh for multiresolution.
- `NodalDisp`: calculates the nodal displacements solving the equilibrium linear system. This function is called when `opsolver` is equal to `0`, `1` or `2` (not using multigrid).
- `Prolongation`: obtains the prolongation matrix for the geometric multigrid method.
- `ProlongationDPLS`: obtains the prolongation matrix for the algebraic multigrid method using the Dynamic Pattern Least Squares method.
- `ProlongationS`: obtains the prolongation matrix for the geometric multigrid method when using serendipity elements.
- `SetupAMG`: performs the setup phase for the algebraic multigrid method.
- `SetupGMG`: performs the setup phase for the geometric multigrid method.
- `SetupStiffMatrix`: obtains the row and column indexes with nonzero elements (sparse pattern) of the global stiffness matrix and its permutation vector.
- `Smooth`: performs iterations of the smoother method of multigrid to solve the linear system.
- `SolveMG`: applies the conjugate gradient method with multigrid as preconditioner to solve the equilibrium linear system.
- `StartMR`: creates the density and design variable meshes data and gets the initial design variables vector for multiresolution.

- `StartStrSLP`: initializes the sequential linear programming algorithm to solve the topology optimization problem. This function realizes several checks and initial calculations before calling the SLP algorithm.
- `StrConnections`: obtains the strength of connections for each node in the algebraic multigrid method.
- `StrSLP:` applies the sequential linear programming (SLP) method to solve the topology optimization problem.
- `TestSpace`: constructs the test space matrix solving the generalized eigenvalue problem for the algebraic multigrid method.
- `TotalStr`: generates the complete element densities vector of the optimal structure based on the symmetries.
- `VCycle`: applies the multigrid V Cycle recursively.
- `WCycle`: applies the multigrid W Cycle recursively.

Furthermore, there are alternative functions in the folder "WOK" with which it is possible to solve the problem without explicitly generate the global stiffness matrix. This option, however, is less efficient and should be used only if the problem is really huge and the computational memory is insufficient to generate the global stiffness matrix.