materials

MDPI

Article

A Simple Matlab Code for Material Design Optimization Using Reduced Order Models

George Kazakis † and **Nikos D. Lagaros** *,†

Institute of Structural Analysis and Antiseismic Research, School of Civil Engineering, National Technical University of Athens, 9, Heroon Polytechniou Str., Zografou Campus, GR-15780 Athens, Greece; kzkgeorge@gmail.com

* Correspondence: nlagaros@central.ntua.gr; Tel.: +30-210-772-2625

† These authors contributed equally to this work.

Abstract: The main part of the computational cost required for solving the problem of optimal material design with extreme properties using a topology optimization formulation is devoted to solving the equilibrium system of equations derived through the implementation of the finite element method (FEM). To reduce this computational cost, among other methodologies, various model order reduction (MOR) approaches can be utilized. In this work, a simple Matlab code for solving the topology optimization for the design of materials combined with three different model order reduction approaches is presented. The three MOR approaches presented in the code implementation are the proper orthogonal decomposition (POD), the on-the-fly reduced order model construction and the approximate reanalysis (AR) following the combined approximations approach. The complete code, containing all participating functions (including the changes made to the original ones), is provided.

Keywords: topology optimization; microstructure; homogenization; Matlab; reduced order models; reduced basis; on-the-fly construction; POD; approximate reanalysis

Citation: Kazakis, G.; Lagaros, N.D. A Simple Matlab Code for Material Design Optimization Using Reduced Order Models. *Materials* **2022**, *15*, 4972. https://doi.org/10.3390/ma15144972

Academic Editor: Luciano Lamberti

Received: 28 May 2022
Accepted: 14 July 2022
Published: 17 July 2022

1. Introduction

The basic theory for the implementation of topology optimization in material design was presented first in 1994 by Sigmund [1], followed by Sigmund and Torquato in 1997 [2] and by Gigiansky and Sigmund in 2000 [3]. Since then, many other studies have been published dealing with a variety of different material optimization problem formulations. Neves et al. [4] and Fujii et al. [5] used the density-based approach to design periodic microstructures for optimal elastic properties. Guest and Prévost [6] dealt with the topology optimization of the fluid flows in the design of porous periodic materials. Challis et al. [7], Amstutz et al. [8] and Gao et al. [9] proposed level set-based approaches for the design of microstructures, and Huang et al. in [10,11] presented a Bi-directional Evolutionary Structural Optimization (BESO)-method-based approach for the optimal design of periodic microstructures. A detailed review of the different methodologies in the optimal design of materials together with a description of the variety of the approaches presented so far to deal with the topology optimization of the macro design concurrently with the micro design can be seen in [12].

In the past, due to the increased computational effort required for solving the topology optimization problem, various methodologies along different directions (approximate reanalysis, model order reductions, machine learning, etc.) have been presented. Indicatively, Kirsch and Paralambros [13] first proposed a unified approach to structural reanalysis using the combined approximations in topology optimization. Wang et al. [14] presented a methodology of recycling search spaces in iterative solvers during the optimization procedure. Amir et al. [15] proposed an approximate reanalysis approach in topology

optimization based on the combined approximations approach and the use of approximations for dealing with the solution of the analysis problem, generated by a Krylov subspace iterative solver [16]. In addition, in [17], Amir et al. addressed the computational cost of the robust topology optimization formulation. Gogu [18] presented an on-the-fly approach for the construction of the reduced order model. Alaimo in [19] proposed an reduced order model approach where a reduced basis is created based on the functional principal component analysis (FPCA). Ferro et al. [20] proposed a proper orthogonal decomposition (POD) approach where the stages of the SIMP method were used as reduced basis vectors during the optimization procedure. Senne et al. [21] proposed a combination of the approximate reanalysis technique with the sequential piecewise linear programming method, and Xiao et al. [22] proposed a reduced order modeling approach which constructed the reduced basis using the proper orthogonal decomposition (POD) approach. Meanwhile, in the same direction, to reduce the computational effort, various machine learning methodologies have been presented, and the precursor of these was a study by the authors [23].

So far, many Matlab code implementations of the topology optimization formulation have been presented in various publications. For the density-based approach, the first code was the so-called top99 [24] implementation that was followed by the top88 one [25]. Both Matlab codes were dealing with the 2D topology optimization problem formulation. Liu et al. [26] and Ferrari [27] presented an extension of the density-based approach into the 3D space, with Ferrari [27] suggesting code modifications for achieving better performance. Talischi et al. [28] and Chi et al. [29] expanded the 2D and 3D density-based approaches by using the capability to deal with unstructured meshes as well. Amir et al. in [30] presented a code implementation for improving the computational cost of the topology optimization procedure using the multi-grid, preconditioned conjugated gradients solver (MGCG). Huang et al. [31] presented an Evolutionary Structural Optimization (ESO) topology optimization code implementation based on the top99 for the 2D space. Wang et al. [32] and Challis [33] published code implementations that rely on the level set approach for the topology optimization for 2D problem formulations. Otomori et al. [34] and Wei et al. [35] also presented level set-based code implementations for the topology optimization using the reaction diffusion equation and radial basis functions, respectively. In 2019, an integration of a topology-optimization procedure with SAP2000, well-known commercial software for analysis and design of structural systems, was presented by the authors [36]. In addition, Gao et al. in [37] presented IgaTop, a topology optimization formulation using isogeometric analysis.

Subsequently, several code implementations were also presented that were dealing with the homogenization-based topology optimization approaches. Specifically, numerical homogenization implemented for 2D and 3D material design was presented in studies [38,39]. In addition, an energy-based homogenization approach combined with the optimal design of materials was presented in [40]. In this study, a topology-optimization-based Matlab code implementation is presented that deals with the problem of material design at the microstructure level, assisted by model order reduction (MOR) approaches. In particular, the topology optimization procedure is combined with the proper orthogonal decomposition (POD), the on-the-fly reduced order model construction and the approximate reanalysis following the combined approximations approach. Although the code provided covers the case of microscale material design for 2D design domains, it can easily be extended to 3D design domains by modifying the homogenization part to produce the 3D elasticity tensor of the unit cell and extend the problem formulation's description to handle both macro and micro scales. The implementation of the MOR approaches is independent of the dimensionality of the formulation due to being applied in the solution part of the finite element analysis performed at the macro scale.

The layout of the work is composed of four sections accompanied by Sections 1 and 6. In particular, a short description of the optimal design problem of materials is presented in Section 2; subsequently, in Section 3 the theoretical part of the

integration of the model order reduction methodologies into the material optimization problem is provided. The detailed description of the most critical parts of the Matlab code's implementation is provided in Section 4, followed by test examples in Section 4 where the ease of use of the code is presented.

2. Optimal Design of Materials

The formulations of the topology optimization (TO) problem used for the design of materials are expressed as the optimal distribution of material volume fraction into the unit cell design domain so that the structural response is optimized. Thus, compared to the original TO problem, the design variables are different, from density values X of the finite elements used to discretize the macro design domain to densities x of the finite elements discretizing the micro-unit cell design domain. In this scope, it can be seen that the optimization procedure performed involves two different scales; the macro scale and the micro one (Figure 1). The design variables as well as the volume constraint are defined at the micro scale, where as the objective function is set on the macro scale; however, it is still expressed as a function of x. The transition between the two scales is achieved through the elasticity tensor by means of the homogenization method [38].

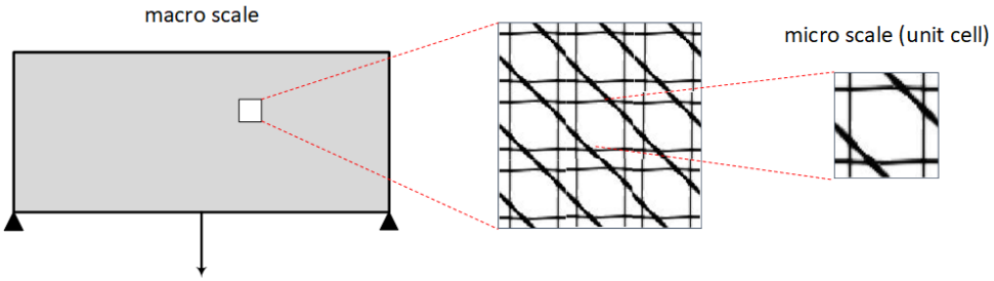

Figure 1. Schematic representation of the periodic unit cell (micro scale) inside the macro structure (macro scale).

The mathematical formulation of the typical topology optimization problem is thus changed according to the following expression of Equation (1).

$$\begin{aligned} C(x) &= F^T \cdot U(x) \\ s.t. & \\ F &= K \cdot U(x) \\ V(x)/V_0 &= f \\ 0 \leq & x_e \leq 1 \end{aligned} \tag{1}$$

where $C(x)$ is the compliance, F is the load vector, $U(x)$ is the resulting displacements from the structural analysis, $F = K \cdot U$ is the linear system of equations derived from the finite element method, $V(x)$ is the material volume resulting from the densities x, V_0 is the full domain material volume and f is the volume fraction applied as a constraint. The derivative of the objective function is obtained using the adjoint method [41] as described in the following expression of Equation (2):

$$\frac{\partial C}{\partial x_e} = -U^T \cdot \frac{\partial K}{\partial x_e} \cdot U \tag{2}$$

while $\partial K/\partial x_e$ is calculated using the following expression of Equation (3):

$$\frac{\partial K}{\partial x_e} = \frac{\partial C^H}{\partial x_e} \cdot K_0 \tag{3}$$

where C^H is the homogenized elasticity tensor. According to the homogenization theory, the elasticity tensor is obtained by applying unit strains globally to the unit cell domain as well as locally to the finite elements used to discretize the unit cell domain and then using the following expression of Equation (4).

$$C^H_{i,j} = \frac{1}{V}\sum_{e=1}^{N}\int_{x_e}(u_e^{0(i)} - u_e^{(i)}) \cdot k_e^0 \cdot (u_e^{0(j)} - u_e^{(J)})dV_e \tag{4}$$

where superscript 0 denotes the globally applied strains, thus u^0 are the displacement fields resulting from the globally applied unit strains, u are the displacement fields resulting from the locally applied unit strains and C^H is the elasticity tensor. If Equation (4) is then differentiated with respect to y_e, the derivative of the elasticity tensor can be obtained using the following expression of Equation (5):

$$\frac{\partial C^H}{\partial x_e} = \frac{\partial E}{\partial x_e} \cdot \frac{1}{V}\int_{x_e}(u_e^{0(i)} - u_e^{(i)}) \cdot k_e^0 \cdot (u_e^{0(j)} - u_e^{(J)})dV_e \tag{5}$$

Furthermore, the derivative of the Young modulus with respect to each unit cell element density is obtained using the modified SIMP approach [42]. Thus, the Young modulus as a function of density value x is defined using the following expression of Equation (6):

$$E(x) = E_{min} + x^p \cdot (E_0 - E_{min}) \tag{6}$$

and the derivative from the expression of Equation (7):

$$\frac{\partial E}{\partial x_e} = p \cdot x^{p-1} \cdot (E_0 - E_{min}) \tag{7}$$

3. Model Order Reduction in Material Optimization

The main focus of the model order reduction (MOR) approaches is to reduce the computational cost required for solving the linear system of equations formulated from the finite element method (FEM). This is achieved by creating a reduced basis model and finding an approximate solution instead of solving the full-order system of equations. The creation of the reduced basis system of equations is accomplished by substituting first the displacement vector U in the finite element equilibrium system of equations with an approximation vector $\Phi \cdot y$. Matrix $\Phi = \{\Phi_i, \ldots, \Phi_m\}$ consists of the reduced basis vectors, and its first dimension corresponds to the dimension of vector U. Its second dimension refers to a small number (e.g., 5 to 10) chosen as the number of the reduced basis vectors. The left and right hand sides of the resulting equation is then multiplied by Φ^T, as shown in the following expressions of Equation (8).

$$\begin{aligned} F = K \cdot U \rightleftharpoons \\ F \approx K \cdot \Phi \cdot y \rightleftharpoons \\ \Phi^T \cdot F \approx \Phi^T \cdot K \cdot \Phi \cdot y \rightleftharpoons \\ F_{eff} \approx K_{eff} \cdot y \end{aligned} \tag{8}$$

where F is the load vector, U is the displacement vector, K is the stiffness matrix, F_{eff} is the reduced approximation of the load vector, K_{eff} is the reduced approximation of the stiffness matrix and y is the reduced basis displacement vector. In general, to assess the accuracy of the projected solution, the residual load can be obtained and divided by the norm of the original load vector, as shown in the following Equation (9).

$$e^2 = \frac{\|K \cdot \Phi \cdot a - F\|^2}{\|F\|^2} \tag{9}$$

Thus, the most important part of the procedure is the creation of the reduced basis vectors that represent the main variation in most of the reduced model approaches. In the following subsections, the methodology as well as the creation of the reduced basis vectors of three MOR approaches will be presented. The three approaches implemented are the proper orthogonal decomposition (POD), the on-the-fly reduced order model construction and the approximate reanalysis following the combined approximations approach. In most MOR approaches, the approximation of the displacement field is taken into account in the calculation of the sensitivities through the expression of Equation (10).

$$\frac{\partial C}{\partial x_e} = -y^T \cdot \Phi^T \cdot \frac{\partial K}{\partial x_e} \cdot \Phi \cdot y - \sum_{i=1}^{N_b} \lambda_i^T \cdot \frac{\partial K_i}{\partial x_e} \cdot U_i \tag{10}$$

where the first term of the expression corresponds to the sensitivity calculated from the approximate solution and the second part denotes the adjustment term which corrects the sensitivity calculation, taking into account that the solution is a approximation. The term λ_i is the solution vector of the following expression of Equation (11) for each reduced basis vector.

$$K_i \cdot \lambda_i = 2 \cdot y_i \cdot (F - K \cdot \Phi \cdot y) \tag{11}$$

U_i used in Equation (10) and K_i of Equation (11) denote the displacement vector and stiffness matrix of each reduced basis vector, respectively. Aiming to simplify the code implementation of the adopted MOR approaches into the optimal material design procedure that is described below, the sensitivity adjustment term in not taken into account.

3.1. Proper Orthogonal Decomposition

According to the proper orthogonal decomposition (POD) approach, the construction of the reduced basis vectors is achieved by means of the singular value decomposition (SVD) factorization methodology. In particular, a small number (e.g., five to ten) of optimization iterations is performed first that is equal to the number of the reduced basis vectors, in which the full-scale system equations are solved and the resulting displacement vectors are stored in the matrix A. Thus, matrix A is composed of different snapshots of the displacement field in the early phases of the optimization procedure. Before applying the SVD factorization methodology to the matrix A, the mean of the displacement snapshot of the final reduced basis vector is subtracted from A, as described in [22]. Then, SVD methodology is applied to matrix A and three different matrices are generated, as shown in the following Equation (12):

$$A = \overline{U} \cdot \Sigma \cdot V' \tag{12}$$

Matrix $\overline{U}$ in general contains information about the spacial correlation of the snapshots of matrix A. Matrix Σ is a diagonal matrix containing the weight coefficients denoting the importance of each column of matrix $\overline{U}$, and finally V contains the corresponding time dynamics of each of the columns of matrix $\overline{U}$. The columns of matrix $\overline{U}$ are also called the POD modes and are used as the reduced basis vectors Φ_i consisting of matrix Φ. Thus, in the POD approach, the reduced basis matrix Φ coincides with the first matrix (i.e., $\overline{U}$) of the SVD of matrix A.

$$\Phi = \overline{U} \tag{13}$$

Subsequently, given the creation of the reduced basis matrix Φ in each optimization iteration, the displacement vector is obtained using the constructed reduced model and then projected to the full scale. The accuracy of each new solution in validated using Equation (9), and if the deviation is too large, a full-scale finite element analysis (FEA) is performed and the matrix A is updated with the new snapshot of displacements (mth

column of matrix A), removing the earliest generated one (i.e., the first column of matrix A). A new SVD is performed on the updated variant of A and a new reduced basis matrix Φ is used for the next iterations.

3.2. On-the-Fly Reduced Order Model Construction

Similarly to the POD approach, according to the on-the-fly approach a number of optimization iterations are performed first in order to generate displacements snapshots of the early optimization stages. Then, the reduced basis vectors are created based on the Gram–Schmidt orthogonalization methodology which is applied onto the displacement snapshots of the early optimization stages, as follows: for the first reduced basis vector, only the first displacement snapshot is utilized, in which a normalization is performed following the expression of Equation (14):

$$\Phi_1 = \frac{U_1}{\|U_1\|} \tag{14}$$

For the next reduced basis vectors, the following Gram–Schmidt orthogonalization procedure is applied, taking into account all previous reduced basis vectors, as shown in the following expression of Equation (15).

$$\widehat{\Phi}_{i+1} = U_{i+1} - \sum_{j=1}^{i} \langle U_{i+1}, \Phi_j \rangle \Phi_j \tag{15}$$

Subsequently, the new $\widehat{\Phi}_{i+1}$ is normalized (as denoted in Equation (16) and the resulting vector is added to the reduced basis matrix.

$$\Phi_{i+1} = \frac{\widehat{\Phi}_{i+1}}{\|\widehat{\Phi}_{i+1}\|} \tag{16}$$

Following the same steps as described for the POD approach, when the reduced basis matrix Φ is constructed, the subsequent optimization iterations rely on approximate displacement fields obtained using the reduced basis matrix and the accuracy of every new reduced basis based FEA in assessed using the expression of Equation (9). If the accuracy is not acceptable, a new reduced basis vector is created by means of a full-scale FEA and using the previously described procedure. Then, matrix Φ is updated by removing the earliest generated reduced basis vector (i.e., first column of matrix Φ) and adding the new one as the mth column of matrix Φ.

3.3. Approximate Reanalysis

In contrast to the POD and on-the-fly approaches, in the approximate reanalysis, one the reduced basis vectors is not created based on displacement snapshots obtained from the initial optimization iterations. Instead, new reduced basis vectors are created in each optimization iteration. These reduced basis vectors are based only on a single snapshot of the displacement field obtained by solving the full-scale system of equations; the displacement field snapshot is updated during the optimization procedure. The fist reduced basis vector is equal to the displacement snapshot used as the basis of the reduced order model, and thus is defined using the following expression of Equation (17):

$$\Phi_1 = U_1 = K_0^{-1} \cdot F \tag{17}$$

Using Φ_1 and K_0 as the basis of each reduced basis matrix Φ, at each iteration, a new set of reduced basis vectors is constructed. Each vector is obtained using the following expression of Equation (18):

$$K_0 \cdot U_i = F - \Delta K \cdot U_{i-1} \quad i = 2 \ldots i_{max} \tag{18}$$

where $\Phi_i = U_i$, ΔK is the difference between the original stiffness matrix K_0 and the one corresponding to the current iteration. The size of the reduced basis is not the same for every iteration; after the creation of each reduced basis vector the accuracy of the solution is validated using Equation (9), and if it is below a certain threshold, the accuracy of the solution is accepted. A maximum size of reduced basis vectors is also provided. The update of the first reduce basis vector is usually performed after either a fixed number of iterations or after the change in the design variables or the compliance is significant. For a more detail review of the approximate reanalysis approach, the reader is referred to [15].

4. The Matlab Code Implementation

Part of the implementation of the methodologies described previously into a Matlab code is based on two existing codes. For the homogenization part, the basis was the Matlab code presented by Andreassen in [38], whereas for the topology optimization part, the basis was the Matlab code presented also by Andreassen in [25]. For efficiency, in the following sections only the parts of the code modified and the logic behind these modifications will be presented, starting from the part of the homogenization method and then to the topology optimization part. The code implementation presented here is composed of nine Matlab files. These are: homogenize function (i.e., *homogenize.m* Matlab file) that implements the homogenization procedure, elementMatVec function (i.e., *elementMatVec.m* Matlab file) that is used to compute the element load vectors and stiffness matrix, Q4elementStiffnessMatrix function (i.e., *Q4elementStiffnessMatrix.m* Matlab file) that is used for performing a similar role to the elementMatVec function, interpolate function (i.e., *interpolate.m* Matlab file) that performs the SIMP interpolation scheme, UCOpt function (i.e., *UCOpt.m* Matlab file) that performs the material topology optimization procedure, and three additional Matlab files containing the procedures of the corresponding MOR approaches, i.e., *pod.m* Matlab file containing the pod function, *onthefly.m* Matlab file containing the onthefly function and *ar.m* Matlab file containing the ar function.

4.1. Homogenization Code Implementation (Matlab File "homogenize.m")

In this section, the modifications made to the homogenization Matlab files will be presented. For a more in-depth description of the functionalities of the original Matlab homogenization code, the reader is referred to [38]. There were two main modifications of the current implementation compared to the original function, denoted as homogenize, that is used for implementing the homogenization method, originally presented by Andreassen [38]. The first modification refers to the transition from the lame parameters to the Poisson ratio and Young modulus parameters, and the second one to the addition of the derivative of the homogenized tensor dC^H/dy_e with respect to the densities at the unit cell level.

Input parameters: The input arguments of the new implementation of the homogenize function are the following:

```
1 function [CH,DCH] = homogenize(lx, ly, E, nu, dE, phi)
```

where lame parameters as well as the mapping parameters are replaced by matrix E containing the values of the Young modulus for every finite element used to discretize the unit cell domain, matrix dE contains the derivative of the Young modulus based on the modified SIMP approach, following the expression of Equation (7) and the Poisson ratio nu that is the same for all finite elements. In addition, an extra output argument was added to the method called DCH which is the derivative of the elasticity tensor from the expression of Equation (5), calculated using the parameter matrix dE.

Initialization: Due to the elimination of the mapping variable, the number of elements along the directions of the abscissa and ordinate of the unit cell are taken from the size of matrix E, and thus *Line* 4 (of the original code in [38] function) was slightly modified to take the number of elements from matrix E, as follows:

```
4 [nely, nelx] = size(E);
```

By using the Young modulus and Poisson ratio, the need for decomposing into two parts the loading vectors and stiffness matrix as described in [38] is not required. Thus, function elementMatVec was modified to return three loading vectors corresponding to the three different unit strains, as shown in the following expression of Equation (19) and the element stiffness matrix computed using the Poisson ratio parameters without the Young modulus.

$$f_e^i = \int_{V_e} B_e^T \cdot C_e \cdot e^i dV_e \tag{19}$$

Thus, *Line* 9 (of the original code in [38]) was modified to have the following form:

```
9 [ke, fe] = elementMatVec(dx/2, dy/2, phi, nu);
```

Assembly of the stiffness matrix and loading vectors: elementMatVec function (see Matlab file "*elementMatVec.m*") was modified to compute the element loading vectors and stiffness matrix using the Poisson ratio by changing the first *Line* of the function to compute the elasticity tensor from Poisson ratio and Young modulus of one instead of the Lame parameters as presented below:

```
2 A = [1 nu 0; nu 1 0; 0 0 (1-nu)/2];
3 C = 1/(1-nu^2)*A;
```

Thus, in the last *lines* of the function where the loading vectors and stiffness matrix are computed, the *lines* are

```
34 % Element matrices
35 ke = ke + weight*(B' * C * B);
36 % Element Loads
37 fe = fe + weight*(B' * C * diag([1 1 1]));
```

In the assembly of the global stiffness matrix part of the homogenize function, *Lines* 34 and 35 (of the original code in [38]) are removed due to Young modulus already being a matrix, and *Line* 37 (of the original code in [38]) is modified to multiply the element stiffness matrix with a vector of the Young modulus, as shown below:

```
37 sK = ke(:)*E(:).';
```

Moving now to the creation of the global loading vector, *Line* 41 (of the original code in [38]) is replaced by a simple multiplication of the element loading vector with the element Young modulus.

```
41 sF = fe(:)*E(:).';
```

Due to the element loading vectors as well as the element stiffness matrix no longer being separated into two parts, *Lines* 53 and 54 (of the original code in [38]) are removed from the code. In addition, an extra line is added bellow the initialization of the elasticity tensor, initializing the derivative of the elasticity tensor. During the iterative procedure performed from *Lines* 64 to 75 (of the original code in [38]), the parameters *sumLambda* and *sumMu* are replaced with the parameter *sumYoung* which is obtained in the same way using *ke* instead of *keLambda* and *keMu*. An extra procedure is added to compute the derivative of the elasticity tensor in which the variable *sumYoung* is multiplied with the derivative of the Young modulus and then added to the cell variable *DCH*. *DCH* is a cell variable of a size of the number of elements, and contains the 3×3 derivative of the elasticity tensor of each element. The new iterative procedure is presented below:

```
64 for i =1:3
65   for j = 1:3
66     sumYoung = ((chi0(:,:,i) - chi(edofMat+(i-1)*ndof))*ke).*...
67     (chi0(:,:,j) - chi(edofMat+(j-1)*ndof));
68     sumYoung = reshape(sum(sumYoung,2), nely, nelx);
```

```
69      % Homogenized elasticity tensor
70      CH(i,j) = 1/cellVolume*sum(sum(E.*sumYoung));
71      finalSum = dE.*sumYoung;
72      for k=1:nely
73        for l=1:nelx
74          DCH{k,l}(i,j) = 1/cellVolume*finalSum(k,l);
75        end
76      end
77    end
78  end
```

4.2. Topology Optimization Code Implementation

In this section, all modification applied to the top88 code published in [25] will be presented. The aim of all modifications was to transfer the code implementation from the conventional topology optimization formulation into the optimal design of materials. The new function used to perform the optimization procedure is called UCOpt. In this function, in addition to the input parameters already present in the original top88 code, four extra parameters were added. Due to the two different scales (micro and macro), two parameters (i.e., lx, ly) representing the dimensions along the directions of the abscissa and ordinate of the macro domain, respectively, were added for the case of macro scale, and two parameters (i.e., nlx, nly) representing the number of elements along the directions of the abscissa and ordinate of the unit cell were added for the micro scale. Thus, the resulting function is presented below:

```
2  function UCOpt(lx,ly,nelx,nely,nlx,nly,volfrac,penal,rmin,ft)
```

As for the creation of the stiffness matrix variable KE, a new function is utilized. This function takes into consideration the length of the finite element along the directions of the abscissa and ordinate in the form of dx and dy, as well as the elasticity tensor instead of the Young modulus and the Poisson ratio used in the original code. This change is applied to enable the creation of the stiffness matrix from the homogenized elasticity tensor created by the homogenization function. In addition, the creation of the element stiffness matrix is performed inside the optimization procedure before the finite element analysis. Moving to the initialization of the design variables, in *Lines* 40 to 47 (of the UCOpt function), the initialization of the design variable is performed, in which instead of mapping the volume fraction to all densities and circle of zero densities is created in the centre of the unit cell, and all other densities are set to one, as shown in Figure 2.

Figure 2. Initial unit cell.

This is achieved using the following iterative procedure:

```
x = ones(nly,nlx);
for i = 1:nlx
  for j = 1:nly
    if sqrt((i-nlx/2-0.5)^2+(j-nly/2-0.5)^2) < min(nlx,nly)/3
      x(j,i) = 0;
    end
  end
end
```

Moving to the optimization loop, two additional steps are added before performing the finite element analysis part. In the first step, a function called interpolate is utilized to compute the Young modulus and its derivative with respect to the design variables using the expressions of Equation (6) for the element Young modulus and Equation (7) for the corresponding derivative. During the second step, the resulting Young modulus and its derivative are provided to the homogenize function, which in turn produces the elasticity tensor and its derivative for each element consisting the unit cell. Moving to the finite element analysis part, the function Q4elementstiffnessMatrix is utilized to obtain the element stiffness matrix from the homogenized elasticity tensor, which in turn is used to perform the finite element analysis and compute the macro-domain displacements. For the computation of the sensitivities, an iterative procedure is utilized, looping for each element of the micro domain to create a different stiffness matrix for each unit cell element based on each element's derivative of the homogenized elasticity tensor. Then, the variable *ce* is computed in the same manner as in the original code, resulting in the computation of the derivative *dc*.

```
for i = 1:nly
  for j = 1:nlx
    dKE = Q4elementStiffnessMatrix(lx/nelx/2,ly/nely/2,90,DCH{i,
        j});
    ce = reshape(sum((U(edofMat)*dKE).*U(edofMat),2),nely,nelx);
    c = c + sum(sum(ce));
    dc(i,j) = - sum(sum(ce));
  end
end
```

4.3. Model Order Reduction: Code Implementation

In this section, the implementation of the three model order reduction approaches will be presented. Aiming to create an easy integration of the three approaches into the UCOpt function presented in the previous section, the usage of the class structure is opted for the three MOR approaches. Thus, each approach is created as a single Matlab class object containing three common functions denoted as solve, fea and counts, respectively. The solve function is implemented differently for each class, while it is used by the UCOpt function in order to compute every set of displacements during the optimization procedure. The two other functions are the same for all three classes and they are used to perform the full-scale finite element analyses (function fea) and to return the number of full- and reduced-scale iterations performed (function counts). Since the class properties are modified during the optimization procedure, all MOR classes inherit from the handle a Matlab class. In order to use the MOR classes in the UCOpt function, an extra parameter is used called *p*, representing the MOR class, while *Line* 62 (of the pod class) is modified to call the solve function of the *p* class, as presented below:

```
%   U(freedofs) = K(freedofs,freedofs)\F(freedofs);
U(freedofs) = p.solve(K(freedofs,freedofs),F(freedofs));
```

4.3.1. POD: Code Implementation (Matlab File "pod.m")

The pod class was developed for the code implementation of POD approach, which except for the constructor function, requires seven properties and three functions. Out of these properties, three refer to iteration trackers, i.e., parameters labeled as *loop*, *fll* and *rdc* tracking the total number of TO iterations performed, the total number of full finite element analyses and the total number of reduced basis finite element analyses, respectively. The forth parameter refers to the tolerance *tol* that represents the residual force tolerance used as a criterion for updating the reduced basis vectors after the creation of the reduced basis. The remaining three parameters correspond to the number of the reduced basis vectors *Nb*, the reduced basis matrix *fi* and a matrix containing the displacement snapshots *A* that is used to create the reduced basis matrix.

The implementation of the POD approach is performed inside the solve function. The solve function is separated into two main sections. The first section is executed during the first iterations for creating the first variant of the reduced basis matrix *fi*, as shown below:

```
U = obj.fea(K,F);
obj.A(:,obj.loop) = U;
if obj.loop == obj.Nb
    obj.A = obj.A - mean(U);
    [obj.fi,~,~] = svd(obj.A,'econ');
end
```

The second section is executed after the first creation of the reduced basis. In particular, in this section the reduced displacement field *y* is calculated, projecting it to the full scale of the displacement field *U*. Then, it is determined if the forces residual is acceptable. If the forces' residual is deemed not acceptable, then the reduced basis is updated using a new set of displacement snapshots. The implementation is presented below:

```
y = obj.fi'*K*obj.fi \ obj.fi'*F;
U = obj.fi*y;
dF = K*U-F;
res = norm(dF);
if res > obj.tol
  U = obj.fea(K,F);
   obj.A(:,1) = [];
   obj.A(:,obj.Nb) = U;
   obj.A = obj.A - mean(U);
   [obj.fi,~,~] = svd(obj.A,'econ');
else
   obj.rdc = obj.rdc + 1;
end
```

For the creation of the reduced basis matrix *fi*, the svd function of Matlab is utilized selecting the *'econ'* option for generating an economy-size decomposition of matrix *A*.

4.3.2. On-the-Fly: Code Implementation (Matlab File "onthefly.m")

In the implementation of the on-the-fly reduced order model approach, the number of properties required by the corresponding class is reduced from seven to six, basically removing only the displacement snapshot matrix *A*. All other properties remain the same as those used in the POD implementation of the corresponding class. In the same manner as in the POD class, the implementation of the on-the-fly approach requires the use of the solve function. The on-the-fly implementation is also separated into two main parts. In the first one where the reduced basis matrix *fi* is computed, the norm function of Matlab is utilized to perform the normalization of the displacement field vector, whereas the procedure is exactly as described in Section 3.2 where the theoretical description of the on-the-fly reduced order model approach is provided.

```
if obj.loop == 1
  U = obj.fea(K,F);
  obj.fi(:,obj.loop) = U/norm(U);
elseif obj.loop <= obj.Nb
  U = obj.fea(K,F);
  Uorth = U - obj.fi*(obj.fi'*U);
  obj.fi(:,obj.loop) = Uorth/norm(Uorth);
else
```

In the second part of the on-the-fly implementation, the procedure mirrors that of the POD implementation where instead of the svd function of Matlab, the update of the reduced basis matrix is performed as described in the expressions of Equations (15) and (16) in *Lines* 42 to 45 (of the onthefly class).

```
y = obj.fi'*K*obj.fi \ obj.fi'*F;
U = obj.fi*y;
dF = K*U-F;
res = norm(dF);
if res > obj.tol
  U = obj.fea(K,F);
  obj.fi(:,1) = [];
  Uorth = U - obj.fi*(obj.fi'*U);
  obj.fi(:,obj.Nb) = Uorth/norm(Uorth);
else
  obj.rdc = obj.rdc + 1;
end
```

4.3.3. Approximate Reanalysis: Code Implementation (Matlab File "ar.m")

To keep the same structure of the code implementation for the approximate reanalysis approach as that of the previously presented two MOR approaches, the displacement snapshots are updated in a fixed number of iterations without taking into account the change in the objective function or the design variables. For the implementation of the approximate reanalysis approach, two new properties were added compared with the implementation of the on-the-fly approach. These properties correspond to the stiffness matrix $K0$ of the full-scale FEA and to a counter rf that keeps record of how often a new full-scale FEA will be performed.

As far as the solve function goes, its first part, i.e., *Lines* 32 to 35 (of the ar class), deals with the initialization of the reduced basis matrix fi. As discussed earlier, at the beginning of this section the criterion for the update of this procedure is simplified. More specifically, the update of the reduced basis matrix takes place in the first iteration and then after a fix number of iterations specified by the class variable rf, as shown bellow:

```
if (obj.loop == 1) || (mod(obj.loop,obj.rf) == 0)
  U = obj.fea(K,F);
  obj.fi(:,1) = U;
  obj.K0 = K;
else
```

In the second part of the solve function, the reduced displacement vector is obtained. In more detail, in *Line* 37, the difference between the stiffness matrices (dK) is computed. Then, a while loop is implemented (see *Lines* 41 to 49 of the ar class) which builds the reduced basis matrix fi until either the maximum number of reduced basis vectors is reached or the residual is smaller than the tolerance value tol predefined.

```
else
  dK = K-obj.K0;
  U = obj.fi(:,1);
```

```
  res = 100;
  s = 1;
  while (res > obj.tol) && (s <= obj.Nb)
    s = s + 1;
    U = obj.K0\(F - dK*U);
    obj.fi(:,s) = U;
    y = obj.fi'*K*obj.fi \ obj.fi'*F;
    U = obj.fi*y;
    dF = K*U-F;
    res = norm(dF);
  end
  obj.rdc = obj.rdc + 1;
end
```

5. Test Examples

In this section, three simple test examples will be presented in order to demonstrate the ease of use of the proposed topology optimization Matlab code and how the three MOR approaches are integrated in order to assist the search procedure. The first test example refers to a simple bridge problem, the second one refers to the cantilever beam problem and the third one corresponds also to a cantilever beam problem with the load applied at the central right side of the domain. The macro domains for all test examples are schematically presented in Figure 3. In all test examples, the number of the iterations required, the number of full-scale FEAs and the final objective function value achieved are presented when the three MOR approaches are implemented, as well as the case without the application of any MOR approach.

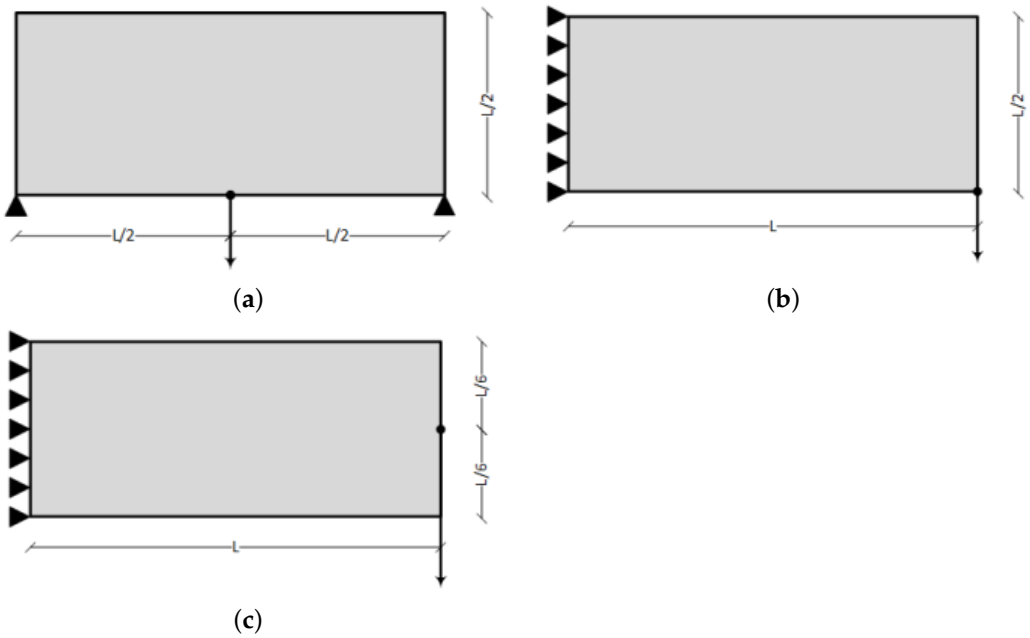

Figure 3. Test examples considered. (**a**) Bridge test example. (**b**) Cantilever beam 1 test example. (**c**) Cantilever beam 2 test example.

5.1. Bridge Test Example

The implementation of the MBB beam test example with respect to the load vector and fixed degrees of freedom is the default implementation of the UCOpt function. The optimization parameters were a grid of 300×150 finite elements in the directions of the abscissa and ordinate for the discretization of the macro domain and a grid of 50×50 finite elements in the directions of the abscissa and ordinate for the discretization of the micro

domain. A target volume fraction of 40%, penalization factor for the SIMP approach of 3, filter radius of 1.5 and application only of a sensitivity filter (option $ft = 1$) were chosen. As far as the three MOR approaches go, the size of the reduced basis was chosen to be 8 for the POD and on-the-fly approaches and 10 for the approximate reanalysis, the tolerance for the update was set equal to 0.01 for all approaches and the update frequency for the approximate reanalysis was set to every six iterations. The script implementation for the POD-assisted optimization implementation is presented below:

```
p = pod(8, 0.01); % for the proper orthogonal decomposition
    approach
UCOpt(10,10,200,100,50,50,0.5,3,1.5,1,p);
```

For the implementation of the other two MOR approaches, changes are only required in the first *line* where the MOR is created, as follows:

```
p=onthefly(8, 0.01); % for the on-the-fly approach
```

and

```
p=ar(10,0.01,6);      % for the approximate reanalysis approach
```

The results obtained of every MOR approach as well as the implementation without MOR assistance are presented in Table 1.

Table 1. Bridge test example: Results of each MOR approach as well as the classic implementation.

Approach	Total itrns	Full FEAs	Compliance
FEA	26	26	105.14
POD	26	8	105.14
on-the-fly	26	8	105.14
AR	26	5	105.14

On the results of topology optimization achieved, in terms of unit cell structure, for the various implementations (with and without MOR), minor differences are observed, while the compliance value resulting from the four implementations is the same. Thus, only one of the results achieved, and in particular, the one obtained by means of the POD approach, is presented in Figure 4.

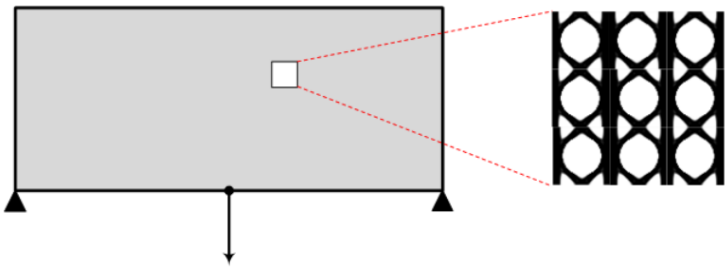

Figure 4. Optimized periodic unit cell for the bridge test example macro structure.

5.2. Cantilever Beam 1 Test Example

For the implementation of the first cantilever beam test example, changes to the load vector and fixed degrees of freedom should be made. In more detail, *Lines* 13 and 14 of the UCOpt function should be changed, as presented below:

```
F = sparse(2*(nelx+1)*(nely+1),1,-1,2*(nely+1)*(nelx+1),1);
fixeddofs = 1:2*(nely+1);
```

The optimization parameters were a grid of 200 × 100 finite elements in the directions of the abscissa and ordinate for the discretization of the macro domain and a grid of 50 × 50

finite elements in the the directions of the abscissa and ordinate for the discretization of the micro domain. Similarly to the first test example, a target volume fraction of 50%, penalization factor for the SIMP approach of 3, filter radius of 1.5 and application only of the sensitivity filter (i.e., option $ft = 1$) were chosen. For the MOR approaches, the size of the reduced basis was chosen to be 4 for the POD and on-the-fly approaches and 10 for the approximate reanalysis, the tolerance for the update was set equal to 0.01 for all approaches and the update frequency for the approximate reanalysis was set to every five iterations. The results obtained for the first cantilever beam test example are presented below in Table 2.

Table 2. Cantilever Beam 1: Results of each MOR approach as well as the classic implementation.

Approach	Total TOP itrns	Full FEAs	Compliance
FEA	107	107	257.3
POD	104	20	257.3
on-the-fly	101	21	257.3
AR	107	22	257.3

Similarly to the first test example, on the results of topology optimization achieved, in terms of unit cell structure, for the various implementations (with and without MOR) minor differences are observed, while the compliance value resulting from the four implementations is the same. Thus, only one of the results achieved, and in particular, the one obtained by means of the on-the-fly approach, is presented in Figure 5.

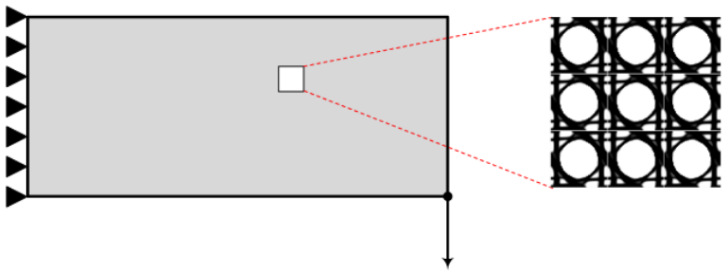

Figure 5. Optimized periodic unit cell for the cantilever beam 1 test example macro structure.

5.3. Cantilever Beam 2 Test Example

For the implementation of the second cantilever beam test example (labeled as cantilever beam 2), changes to the load vector and fixed degrees of freedom should be applied. In more detail, *Lines* 13 and 14 (of the UCOpt) function should be changed as presented below:

```
13 F = sparse(2*((nely+1)*nelx + (nely+1) - ceil(1/2*nely))
       ,1,-1,2*(nely+1)*(nelx+1),1);
14 fixeddofs = 1:1:2*(nely+1);
```

The optimization parameters were a grid of 300×100 finite elements in the directions of the abscissa and ordinate for the discretization of the macro domain and a grid of 50×50 finite elements in the the directions of the abscissa and ordinate for the discretization of the micro domain. Similarly to the first test example, a target volume fraction of 50%, penalization factor for the SIMP approach of 3, filter radius of 1.5 and application only of the sensitivity filter (i.e., option $ft = 1$) were chosen. For the MOR approaches, the size of the reduced basis was chosen to be 4 for the POD and on-the-fly approaches and 10 for the approximate reanalysis, the tolerance for the update was set equal to 0.01 for all approaches and the update frequency for the approximate reanalysis was set to every five iterations. The results obtained for the second cantilever beam test example are presented below in Table 3.

Table 3. Cantilever Beam 2: Results of each MOR approach as well as the classic implementation.

Approach	Total TOP itrns	Full FEAs	Compliance
FEA	84	84	1369.0
POD	83	16	1369.1
on-the-fly	84	16	1368.9
AR	84	17	1368.9

Similarly to the remarks reported for the first two examples presented before, on the results of topology optimization achieved, in terms of unit cell structure, for the various implementations (with and without MOR) observed minor differences are observed, while the compliance value resulting from the four implementations is the same. Thus, only one of the results achieved, and in particular, the one obtained by means of the approximate reanalysis approach, is presented in Figure 6.

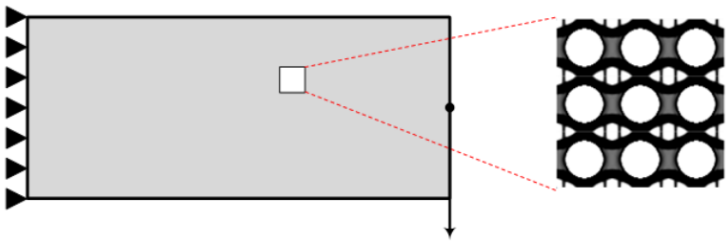

Figure 6. Optimized periodic unit cell for the cantilever beam 2 test example macro structure.

6. Conclusions

The scope of this work is to present an open source numerical implementation of a methodology dealing with the optimal design of material structure using the theories of topology optimization and homogenization as well as the application of reduced order models. The code presented is written in Matlab, and two of the functions are partially based on existing well-known codes, published on the topology optimization and homogenization formulations. The implementation of the three model order reduction (MOR) approaches is simple, and the aim is to provide the means to integrate such models in any type of topology optimization problem formulation. Although the code implementation of the topology optimization part is based on a 2D space variant, it can easily be extended to the 3D space as well without the need to modify any of the three MOR classes presented in this study. The authors would be happy to receive suggested improvements that can be implemented in the public domain of the *UCOpt* codes.

Author Contributions: Conceptualization, G.K. and N.D.L.; methodology, G.K. and N.D.L.; software, G.K.; validation, G.K.; formal analysis, G.K.; investigation, G.K.; writing—original draft preparation, G.K. and N.D.L.; writing—review and editing, G.K. and N.D.L.; visualization, G.K.; supervision, N.D.L.; project administration, N.D.L.; funding acquisition, N.D.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been financed by the ADDOPTML project: "ADDitively Manufactured OPTimized Structures by means of Machine Learning" (No: 101007595).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Please address the corresponding author by his e-mail address nlagaros@central.ntua.gr. The complete code, containing all participating functions (including the changes made to the original ones), is listed in https://github.com/nikoslagaros/TOPcodes, (accessed on 14 June 2022) .

Acknowledgments: This research has been supported by the ADDOPTML project: "ADDitively Manufactured OPTimized Structures by means of Machine Learning" (No: 101007595) belonging

to the Marie Skłodowska-Curie Actions (MSCA) Research and Innovation Staff Exchange (RISE) H2020-MSCA-RISE-2020.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BESO	Bi-directional Evolutionary Structural Optimization
FE	Finite Element
FEA	Finite Element Analysis
FEM	Finite Element Method
MOR	Model Order Reduction
POD	Proper Orthogonal Decomposition
SIMP	Solid Isotropic Material with Penalization
STO	Structural Topology Optimization
SVD	Singular Value Decomposition
TO	Topology Optimization
TOP	Topology Optimization Problem

References

1. Sigmund, O. Materials with prescribed constitutive parameters: An inverse homogenization problem. *Int. J. Solid Struct.* **1994**, *31*, 2313–2329. [CrossRef]
2. Sigmund, O.; Torquato, S. Design of materials with extreme thermal expansion using a three-phase topology optimization. *J. Mech. Phys. Solids* **1997**, *45*, 1037–1067. [CrossRef]
3. Gibiansky, L.V.; Sigmund, O. Multiphase composites with extremal bulk modulus. *J. Mech. Phys. Solids* **2000**, *48*, 461–498. [CrossRef]
4. Neves, M.M.; Rodrigues, H.; Guedes, J. Optimal design of periodic linear elastic microstructures. *Comput. Struct.* **2000**, *76*, 421–429. [CrossRef]
5. Fujii, D.; Chen, B.; Kikuchi, N. Composite material design of two-dimensional structures using the homogenization design method. *Int. J. Numer. Methods Eng.* **2001**, *50*, 2031–2051. [CrossRef]
6. Guest, J.K.; Prévost, J.H. Design of maximum permeability material structures. *Comput. Method Appl. Mech. Eng.* **2007**, *196*, 1006–1017. [CrossRef]
7. Challis, V.J.; Roberts, A.P.; Wilkins, A.H. Design of three dimensional isotropic microstructures for maximized stiffness and conductivity. *Int. J. Solids Struct.* **2008**, *45*, 4130–4146. [CrossRef]
8. Amstutz, S.; Giusti, S.M.; Novotny, A.A.; Souza Neto, E.A. Topological derivative for multi-scale linear elasticity models applied to the synthesis of microstructures. *Int. J. Numer. Methods Eng.* **2010**, *84*, 733–756. [CrossRef]
9. Gao, J.; Li, H.; Gao, L.; Xiao, M. Topological shape optimization of 3D micro-structured materials using energy-based homogenization method. *Adv. Eng. Softw.* **2018**, *116*, 89–102. [CrossRef]
10. Huang, X.; Radman, A.; Xie, Y. Topological design of microstructures of cellular materials for maximum bulk or shear modulus. *Comput. Mater. Sci.* **2011**, *50*, 1861–1870. [CrossRef]
11. Huang, X.; Zhou, S.; Xie, Y.; Li, Q. Topology oprimization of microstructures of cellular materials and composites for macrostructures. *Comput. Mater. Sci.* **2013**, *67*, 397–407. [CrossRef]
12. Wu, J.; Sigmund, O.; Groen, J. Topology optimization of multi-scale structures: A review. *Struct. Multidiscip. Optim.* **2021**, *63*, 1455–1480. [CrossRef]
13. Kirsch, U.; Papalambros, P.Y. Structural reanalysis for topological modifications—A unified approach. *Struct. Multidiscip. Optim.* **2001**, *21*, 333–344. [CrossRef]
14. Wang, S.; Sturler, E. Paulino, G.H. Large-scale topology optimization using preconditioned Krylov subspace methods with recycling. *Int. J. Numer. Methods Eng.* **2006**, *69*, 2441–2468. [CrossRef]
15. Amir, O.; Bendsøe, M.P.; Sigmund, O. Approximate reanalysis in topology optimization. *Int. J. Numer. Methods Eng.* **2008**, *78*, 1474–1491. [CrossRef]
16. Amir, O.; Stolpe, M.; Sigmund, O. Efficient use of iterative solvers in nested topology optimization. *Struct. Multidiscip. Optim.* **2009**, *42*, 55–72. [CrossRef]
17. Amir, O.; Sigmund, O.; Lazarov, B.S.; Schevenels, M. Efficient reanalysis techniques for robust topology optimization. *Comput. Methods Appl. Mech. Eng.* **2012**, *245-246*, 217–231. [CrossRef]
18. Gogu, C. Improving the efficiency of large scale topology optimization through on-the-fly reduced order model construction. *Int. J. Numer. Methods Eng.* **2014**, *101*, 281–304. [CrossRef]
19. Alaimo, G.; Auricchio, F.; Bianchini, I.; Lanzarone, E. Applying functional principal components to structural topology optimization. *Int. J. Numer. Methods Eng.* **2018**, *115*, 189–208. [CrossRef]

20. Ferro, N.; Micheletti, S.; Perotto, S. POD-assisted strategies for structural topology optimization. *Comput. Math. Appl.* **2019**, *77*, 2804–2820. [CrossRef]
21. Senne, T.A.; Gomes, F.A.M.; Santos, S.A. On the approximate reanalysis technique in topology optimization. *Optim. Eng.* **2019**, *20*, 251–275. [CrossRef]
22. Xiao, M.; Lu, D. Breitkopf, P.; Raghavan, B.; Dutta, S.; Zhang, W. On-the-fly model reduction for large-scale structural topology optimization using principal components analysis. *Struct. Multidiscip. Optim.* **2020**, *62*, 209–230. [CrossRef]
23. Kallioras, N.A.; Kazakis, G.; Lagaros, N.D. Accelerated topology optimization by means of deep learning. *Struct. Multidiscip. Optim.* **2020**, *63*, 1185–1212. [CrossRef]
24. Sigmund, O. A 99 line topology optimization code written in Matlab. *Struct. Multidiscip. Optim.* **2001**, *21*, 120–127. [CrossRef]
25. Andreassen, E.; Clausen, A.; Schevenels, M. Lazarov, B.; Sigmund, O. Efficient topology optimization in MATLAB using 88 lines of code. *Struct. Multidiscip. Optim.* **2010**, *43*, 1–16. [CrossRef]
26. Liu, K.; Tovar, A. An efficient 3D topology optimization code written in Matlab. *Struct. Multidiscip. Optim.* **2014**, *50*, 1175–1196. [CrossRef]
27. Ferrari, F.; Sigmund, O.; Guest, J.K. Topology optimization with linearized buckling criteria in 250 lines of Matlab. *Struct. Multidiscip. Optim.* **2021**, *63*, 3045–3066. [CrossRef]
28. Talischi, C.; Paulino, G.H.; Pereira, A.; Menezes, I.F.M. PolyTop: A Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes. *Struct. Multidiscip. Optim.* **2012**, *45*, 329–357. [CrossRef]
29. Chi, H.; Pereira, A.; Meneze, I.F.M.; Paulino, G.H. Virtual element method (VEM)-based topology optimization: An intergrated framework. *Struct. Multidiscip. Optim.* **2020**, *62*, 1089–1114. [CrossRef]
30. Amir, O.; Aage, N.; Lazarov, B.S. On multigrid-CG for efficient topology optimization. *Struct. Multidiscip. Optim.* **2014**, *419*, 815–829. [CrossRef]
31. Huang, X.; Xie, Y. A futher review of ESO type methods for topology optimization. *Struct. Multidiscip. Optim.* **2010**, *41*, 671–683. [CrossRef]
32. Wang, M.; Wang, X.; Guo, D. A level set method for structural topology optimization. *Comput. Methods Appl. Mech. Eng.* **2003**, *192*, 227–246. [CrossRef]
33. Challis, V.J. A discrete level-set topology optimization code written in Matlab. *Struct. Multidiscip. Optim.* **2010**, *41*, 453–464. [CrossRef]
34. Otomori, M.; Yamada, T.; Izui, K.; Nishiwaki, S. Matlab code for a level-set based topology optimization method using a reaction diffusion equation. *Struct. Multidiscip. Optim.* **2014**, *51*, 1159–1172. [CrossRef]
35. Wei, P.; Li, Z.; Li, X.; Wang, M.Y. An 88-line MATLAB code for the parameterized level set method based topology optimization using radial basis functions. *Struct. Multidiscip. Optim.* **2018**, *58*, 831–849. [CrossRef]
36. Lagaros, N.D.; Vasileiou, N.; Kazakis, G. A C# code for solving 3D topology optimization problems using SAP2000. *Optim. Eng.* **2019**, *20*, 1–35. [CrossRef]
37. Gao, J.; Wang, L.; Luo, Z.; Gao, L. IgaTop: An implementation of topology optimization for structures using IGA in MATLAB. *Struct. Multidiscip. Optim.* **2021**, *64*, 1669–1700. [CrossRef]
38. Andreassen, E.; Andreasen, C.S. How to determine composite material properties using numerical homogenization. *Comput. Mater. Sci.* **2014**, *83*, 488–495. [CrossRef]
39. Dong, G.; Tang, Y.; Zhao, Y. A 149 Line Homogenization Code for Three-Dimensional Cellular Materials Written in Matlab. *J. Eng. Mater. Technol.* **2019**, *141*, 555. [CrossRef]
40. Xia, L.; Breitkopf, P. Design of materials using topology optimization and energy-based homogenization approach in Matlab. *Struct. Multidiscip. Optim.* **2015**, *52*, 1229–1241. [CrossRef]
41. Bendsøe, M.P.; Sigmund, O. *Topology Optimization: Theory, Methods and Applications*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2004. [CrossRef]
42. Sigmund, O. Morphology-based black and white filters for topology optimization. *Struct. Multidiscip. Optim.* **2007**, *33*, 401–424. [CrossRef]