Title

Implementation of a Plug-and-Play Reusable Level-Set Topology Optimization Framework via COMSOL Multiphysics

Authors

Guibert, Alexandre T
Hyun, Jaeyub
Neofytou, Andreas
et al.

Publication Date

2023-01-23

DOI

10.2514/6.2023-1675

AIAA SciTech Forum
23-27 January 2023, National Harbor, MD & Online
AIAA SCITECH 2023 Forum

10.2514/6.2023-1675

Implementation of a plug-and-play reusable level-set topology optimization framework via COMSOL Multiphysics

Alexandre T. Guibert[1], Jaeyub Hyun[2], Andreas Neofytou[3] and H. Alicia Kim[4]
University of California San Diego, San Diego, California, 92093, USA

This work is aimed at developing a workflow to use state-of-the-art level-set topology optimization (LSTO) methods with COMSOL Multiphysics. The graphical user interface (GUI) and solver of COMSOL Multiphysics are used to simplify the implementation of the considered physics problem. A physics model file (.m file) is then extracted from COMSOL Multiphysics, which can be called from and adjusted in MATLAB to make an interface with the level-set module. The LSTO modules written in C++ are imported into MATLAB to interface with the physics model extracted from COMSOL Multiphysics. The element sensitivities are computed in COMSOL Multiphysics and combined with the boundary perturbation method to calculate the boundary point (or shape) sensitivities using the discrete adjoint method. The capabilities of the proposed workflow are demonstrated with numerical examples. The proposed workflow alleviates the difficulty of implementing LSTO, especially in the case of coupled multiphysics problems.

I. Introduction

Topology Optimization (TO) is a design optimization method that aims to produce a design by modifying the material distribution in the design domain based on one or more objectives and constraints. Many variants of TO have been developed thus far, such as the density-based TO, boundary-based TO or bi-directional evolutionary structural optimization (BESO) [1]. Among those, the level-set topology optimization (LSTO) method is a popular boundary-based TO technique. The key idea is to represent the geometry of the part using an implicit level-set function and to update this function by solving an advection equation, so-called the Hamilton-Jacobi equation. The level-set function handles naturally the topological changes occurring during the optimization. In contrast to the other methods, the LSTO method [2] does not require filtering. However, LSTO is not widely available in commercial software. For more information regarding the availability of LSTO, a comprehensive review of the different LSTO codes is presented in [3]. Additionally, open-source software usually requires the user to be a proficient programmer in the language the tool was developed in, and this can present a challenging entry barrier to new users to LSTO.

In this paper, we present an efficient workflow to address this challenge and make LSTO more accessible. The model is first created in the graphical user interface (GUI) of COMSOL Multiphysics, and a physics model file (.m file) is easily extracted from the model created in the GUI, which is directly utilized in MATLAB. MATLAB has been chosen as the interface environment for its ease of use and popularity in engineering. COMSOL Multiphysics is integrated with MATLAB via its LiveLink feature. Only a few lines of code in MATLAB are then needed to call the LSTO modules from MATLAB. The workflow has been developed to offer the user a seamless experience and lower the entry barrier. The authors believe that this workflow is especially appropriate for newcomers and for educators, who are interested in utilizing the LSTO for coupled multiphysics problems.

[1] Structural Engineering Distinguished Fellow, Structural Engineering Department, AIAA Student Member
[2] Project scientist, Structural Engineering Department
[3] Post-Doctoral Researcher, Structural Engineering Department
[4] Jacobs Scholar Chair Professor, Structural Engineering Department, AIAA Associate Fellow

II. The Level-Set Topology Optimization Method

The LSTO method is a powerful gradient-based topology optimization technique where the structure is represented by an implicit function φ, for which a signed distance function is usually used. The LSTO method here does not require filtering and/or projection scheme as the structural boundary is clearly defined. The boundary of the structure is implicitly described as $\{x \mid \varphi(x) = 0\}$ where x is a point in the design domain and the overall structure is described as $\{x \mid \varphi(x) \geq 0\}$. At each iteration of optimization, the boundary is updated via the following Hamilton-Jacobi equation.

$$\frac{\mathrm{d}\varphi}{\mathrm{dt}} + V_n|\nabla\varphi| = 0 \tag{1}$$

where V_n is the design velocity normal to the boundary and t is the pseudo time. The design velocity is then computed through a linearized sub-optimization problem as presented in [4]. For more information on LSTO, the reader is referred to [5] and [6] where the method is described in detail.

Our LSTO method is written in C++ and imported into MATLAB using the MEX Package [7]. The element sensitivities of any functions of interest with respect to the design variables are computed using COMSOL Multiphysics' inherent symbolic differentiation capability. They are then passed to the LSTO modules where a boundary perturbation scheme is used for calculating the boundary point sensitivities. More details on the sensitivity analysis scheme can be found in [8]. The overall architecture is presented in Fig. 1.

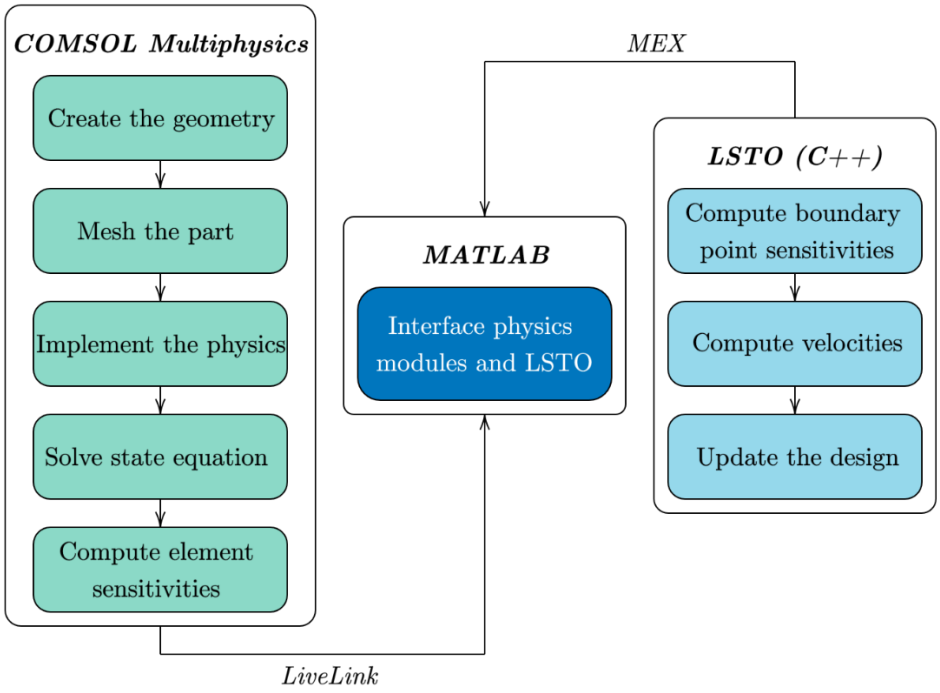

Fig. 1. Overall architecture of the proposed LSTO framework with COMSOL Multiphysics

III. Implementation of LSTO in MATLAB

The physics model is created in COMSOL Multiphysics GUI and saved as a MATLAB file using COMSOL Multiphysics LiveLink feature [9]. A few lines are then added for optimization and the analysis portion of the code is placed in an optimization loop to carry out the finite element analysis at each iteration. The design is updated with the Hamilton-Jacobi equation (1) and a fixed grid is used during the analysis so that remeshing of the domain is not needed. The material property of each finite element is updated through a material interpolation scheme specific to the discipline considered. The lines of code added for optimization are detailed in this section and the optimization workflow is presented in Fig. 2. Four different steps are needed: computation of the element sensitivities, computation of the boundary point sensitivities based on the element sensitivities, computation of the velocities and update of the current design. Each of these are presented in the following paragraphs.

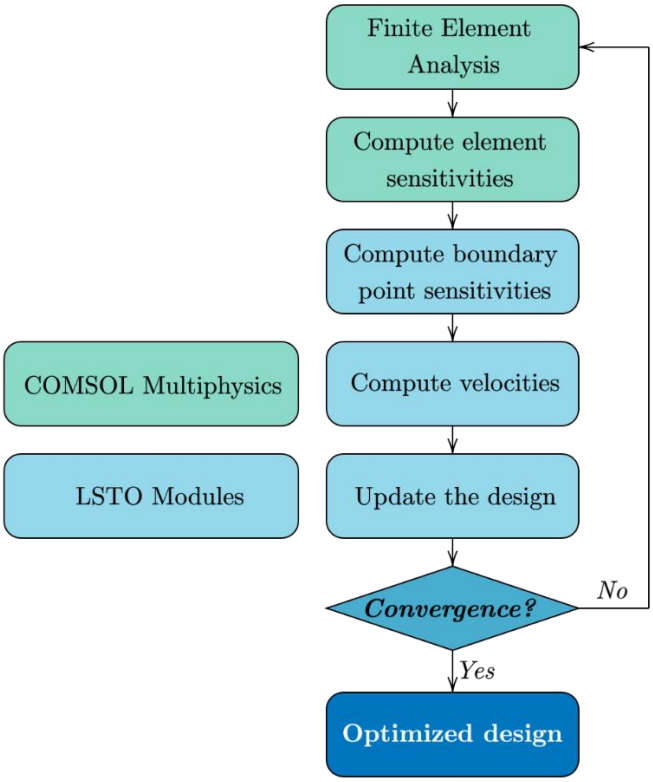

Fig. 2. Workflow of the LSTO process implemented in MATLAB

The first step is to compute the element sensitivities. COMSOL Multiphysics built-in function `fsens()` is used to evaluate the element sensitivities with the adjoint method. The user first adds a sensitivity study in COMSOL Multiphysics GUI and defines a control variable, i.e., a design variable which corresponds to the element density (ED) in our LSTO. Then, the element sensitivities can be evaluated in the following fashion with MATLAB:

```
J = mphint2(model,'comp1.solid.Ws','volume','selection',[1]);
df_drho_COMSOL = mphgetu(model, 'type', 'fsens(ED)');
df_drho_COMSOL = df_drho_COMSOL(ED_index_set);
df_drho_LSM = df_drho_COMSOL(physics_to_LSM_map);
```

Here, the element sensitivities are computed for an objective of structural compliance `J` defined as an integral of the strain energy over the entire domain. Note that the computation of the objective value is not strictly necessary as only the sensitivities are needed for the objective function, but the magnitude of the objective can be used to monitor the convergence history [10]. The mapping between the level-set indices and COMSOL Multiphysics indices is done through a MATLAB method developed in-house that returns `physics_to_LSM_map` and `ED_index_set` is the indices corresponding to the sensitivity analysis.

The second step is to compute the boundary point sensitivities with the discrete adjoint method. This is done with a C++ in-house function `MapSensitivities` which is general for any function of interest and is called in MATLAB as follows

```
df_dbpt = lsm.MapSensitivities(df_drho_LSM, true);
dg_dbpt = lsm.MapVolumeSensitivities();
```

The C++ function `MapVolumeSensitivities` is specifically for volume constraint or objective. Once the boundary point sensitivities have been computed, a suboptimization problem is solved using the C++ function `Solve` that returns the velocities needed to update the design. In addition, the maximum allowed movement of a boundary point is limited for numerical stabilities with the `SetLimits` function based on the CFL condition. Finally, the level-set function is updated using Eq. (1) discretized. These two steps are presented in the following.

```
up = zeros(1,length(df_dbpt)); low = zeros(1,length(df_dbpt));
curr_cons_vals = lsm.GetVolume();
for i=1:length(df_dbpt)
    up(i) = move_limit; low(i) = -move_limit;
end
opt.SetLimits(up, low);
velocities = opt.Solve(df_dbpt, dg_dbpt, curr_cons_vals, false);
lsm.Update(velocities, move_limit, false);
```

The entire optimization loop used for an LSTO example is presented below.

```
n_iterations = 0; max_it = 200; move_limit = 0.1;
opt = myMex.OptimizerWrapper(1, volfrac, 2);
while n_iterations < max_it
    n_iterations = n_iterations + 1;
    % Step 1: Solve the model: Forward analysis
    ED_LSM = lsm.CalculateElementDensities(true);
    for i = 1:length(ED_LSM)
        if ED_LSM(i) < 1e-3
            ED_LSM(i) = 1e-3;
        end
    end
    ED_COMSOL = ED_LSM(LSM_to_physics_map); % Sort the element densities
    % Assign the sorted element densities to COMSOL
    U_sol = mphgetu(model, 'type', 'sol');
    U_sol(ED_index_set) = ED_COMSOL;
    model.sol('sol1').setU(U_sol);
    model.sol('sol1').createSolution;
    model.sol('sol1').feature('v1').set('initmethod','sol');
    model.sol('sol1').feature('v1').set('initsol','sol1');
    model.sol('sol1').feature('v1').set('notsolmethod','sol');
    model.sol('sol1').feature('v1').set('notsol','sol1');
    model.sol('sol1').runAll; % Perform FEA with the assigned element densities
    % Step 2: Calculate the objective function & its sensitivities
    J = mphint2(model,'comp1.solid.Ws','volume','selection',[1]);
    df_drho_COMSOL = mphgetu(model, 'type', 'fsens(ED)');
    df_drho_COMSOL = df_drho_COMSOL(ED_index_set);
    df_drho_LSM = df_drho_COMSOL(physics_to_LSM_map);
    % Step 3: Compute the boundary point sensitivities
    df_dbpt = lsm.MapSensitivities(df_drho_LSM, true);
    dg_dbpt = lsm.MapVolumeSensitivities();
    % Step 4: Solve the sub-optimization problem (for design velocities)
    up = zeros(1,length(df_dbpt)); low = zeros(1,length(df_dbpt));
    curr_cons_vals = lsm.GetVolume();
    for i=1:length(df_dbpt)
        up(i) = move_limit; low(i) = -move_limit;
    end
    opt.SetLimits(up, low);
    velocities = opt.Solve(df_dbpt, dg_dbpt, curr_cons_vals, false);
    % Step 5: Update the level-set function and write STL file
    lsm.Update(velocities, move_limit, false);
    lsm.WriteStl(n_iterations, STLfiles_path, 'levelset');
end
```

IV. Numerical examples

The capability of the LSTO workflow is demonstrated with 3 examples. As the first example, a three-dimensional (3D) cantilever beam is optimized for structural functionality. Then, a heat transfer problem is considered, and the design for a fluid flow problem is optimized.

A. Structural optimization

We consider a cantilever beam made of aluminum and linear elasticity is assumed. The Young's modulus of the *e*th finite element E_e is defined as $E_e = \chi E_0$ where E_0 is the Young's modulus of the solid, and χ is the element density (ED) defined as the fraction of the element that lies in the solid phase, i.e., where $\varphi \geq 0$. For numerical stability, the values of E_e less or equal than $E_0 \times 10^{-3}$ are maintained at this threshold with an *if* statement in MATLAB. The configuration of the problem is presented in Fig. 3.

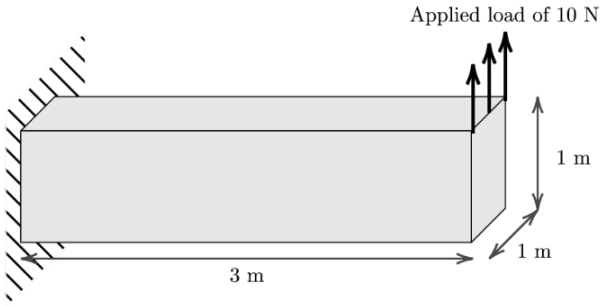

Fig. 3. Configuration of the structural problem

The cantilever beam is optimized to minimize structural compliance with a volume constraint to obtain a stiff and lightweight structure. The optimization problem can be stated as,

$$\begin{aligned} & \text{Find } \Omega \\ \min \quad & C = \boldsymbol{u}^T \mathbf{K} \boldsymbol{u} \\ \text{subject to} \quad & V - (VF)V_0 \leq 0 \\ & \mathbf{Ku} = \mathbf{F} \end{aligned} \tag{2}$$

where $\boldsymbol{u}$ is the vector of nodal displacement, **K** is the stiffness matrix, V is the current global volume, VF is the target volume fraction, V_0 is the global volume of entire design domain, and **F** is the vector of applied force. The target volume fraction is fixed to 20 % and the optimization results are presented in Fig. 4. As expected, the results are similar to an I-beam with similar features such as a web and flanges.

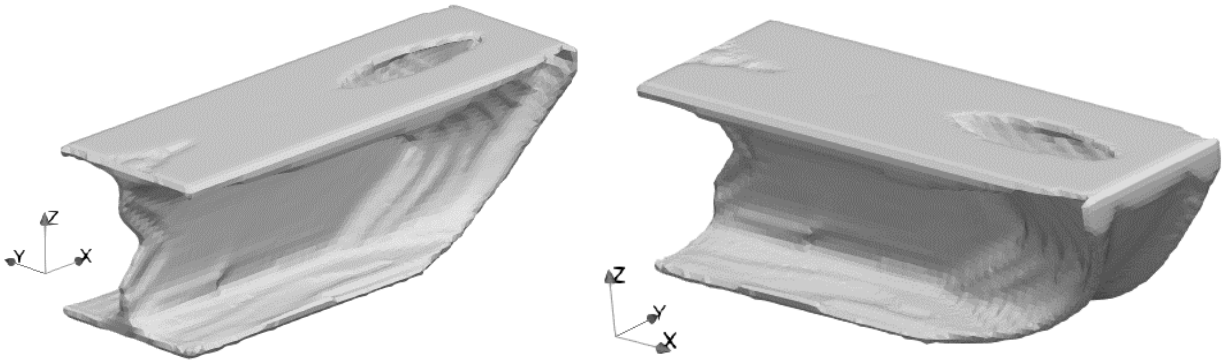

Fig. 4. Optimization results

B. Heat transfer optimization

As the second problem, we consider optimization of a heat exchanger, and conduction is modeled. The thermal conductivity of the eth finite element κ_e is defined as $\kappa_e = \chi\kappa_0$ where κ_0 is the thermal conductivity of the solid. Again, for numerical stability, the element thermal conductivity is constrained to be greater than or equal to $\kappa_0 \times 10^{-3}$. The configuration of the problem is presented in Fig. 5.

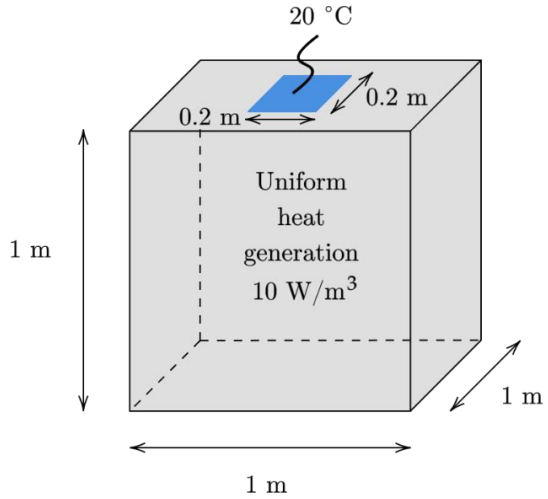

Fig. 5. Configuration of the thermal problem

The optimization is formulated such that the thermal compliance is being minimized and the maximum volume is enforced. This formulation can be interpreted as the minimization of the stored thermal energy for a given volume. The optimization problem can be stated as,

$$\begin{aligned} &\text{Find } \Omega \\ \min \quad & C_{Th} = \boldsymbol{T}^{\boldsymbol{T}}\mathbf{K}_{\mathbf{T}}\boldsymbol{T} \\ \text{subject to} \quad & V - (VF)V_0 \leq 0 \\ & \mathbf{K}_{\boldsymbol{T}}\boldsymbol{T} = \mathbf{F}_{\boldsymbol{T}} \end{aligned} \tag{3}$$

where $\boldsymbol{T}$ is the vector of nodal temperatures, $\mathbf{K}_{\mathbf{T}}$ is the conductivity matrix, and $\mathbf{F}_{\boldsymbol{T}}$ is the vector of applied heat. The volume fraction is fixed to 15 % and the optimization results are presented in Fig. 6. As anticipated, we observe that the optimizer creates conduction paths to transfer heat throughout the domain to the heat sink. The results are similar to the one reported in [11].

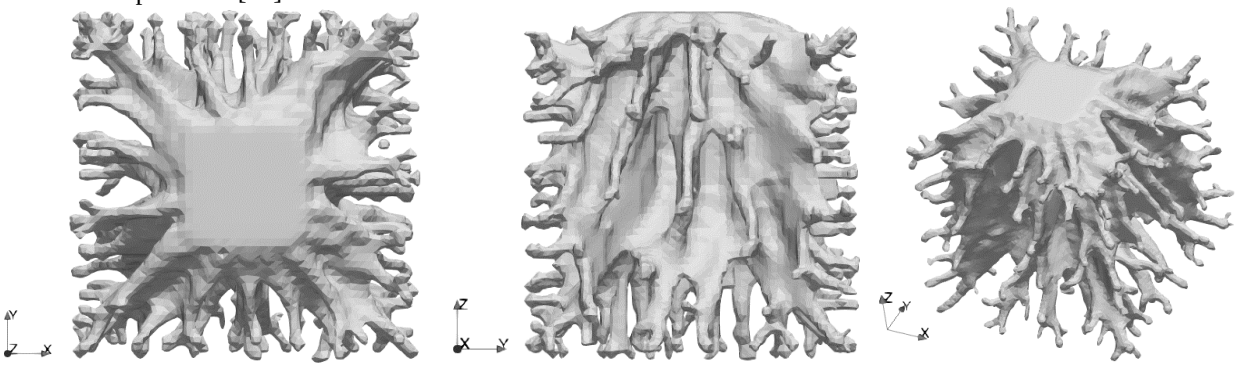

Fig. 6. Optimization results for the heat transfer problem

C. Fluid flow optimization

Finally, we consider the optimization of a fluid problem and compare the optimization results with two different inlet velocities. The laminar incompressible flow model is used, and the Brinkman penalization term is employed. The permeability of the material is interpolated using a RAMP scheme such that the inverse permeability of the solid phase is large and thus the velocity of the fluid is null in that region, where 10^5 is used in this paper. For more details regarding the interpolation scheme, please refer to [12]. The configuration of the problem is presented in Fig. 7.

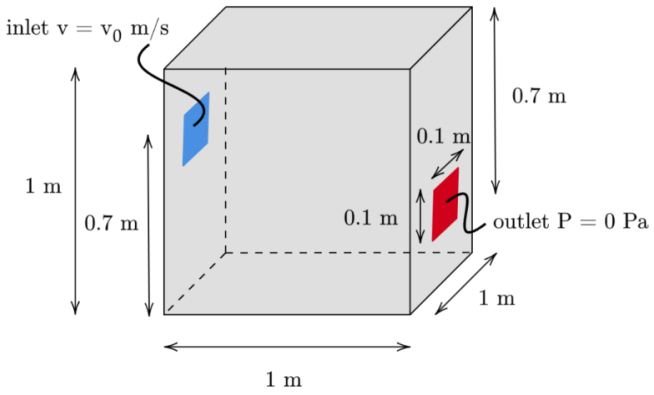

Fig. 7. Configuration of the fluid flow problem

The pressure drop is minimized so that the difference in pressure between inlet and outlet in the domain is minimized and once again the volume constrained. Thus, the optimization problem can be stated as,

$$\begin{aligned} & \text{Find } \Omega \\ \min \quad & J = P_d \\ \text{subject to} \quad & V - (VF)V_0 \leq 0 \\ & \begin{bmatrix} \mathbf{K_{uu}} + \mathbf{C_{uu}} + \mathbf{A_{uu}} & \mathbf{K_{uP}} \\ \mathbf{K_{uP}}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u} \\ \boldsymbol{P} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{F_u} \\ \mathbf{0} \end{Bmatrix} \end{aligned} \tag{3}$$

where P_d is the pressure drop, $\mathbf{K_{uu}}$ is the stiffness matrix for velocity, $\mathbf{C_{uu}}$ is the convective matrix, $\mathbf{A_{uu}}$ is the artificial damping matrix, $\mathbf{K_{uP}}$ is the coupling matrix, $\boldsymbol{u}$ is the vector of nodal velocities, $\boldsymbol{P}$ is the vector of nodal pressures, and $\mathbf{F_u}$ is the vector of external forces.

The optimization results are presented in Fig. 8 for two cases: $v_0 = 0.002\ m/s$ and $v_0 = 0.0005\ m/s$. The optimized result at a lower velocity creates a shorter channel from inlet to outlet whereas at a higher velocity the inertia of the fluid has a greater impact and creates a wider channel.

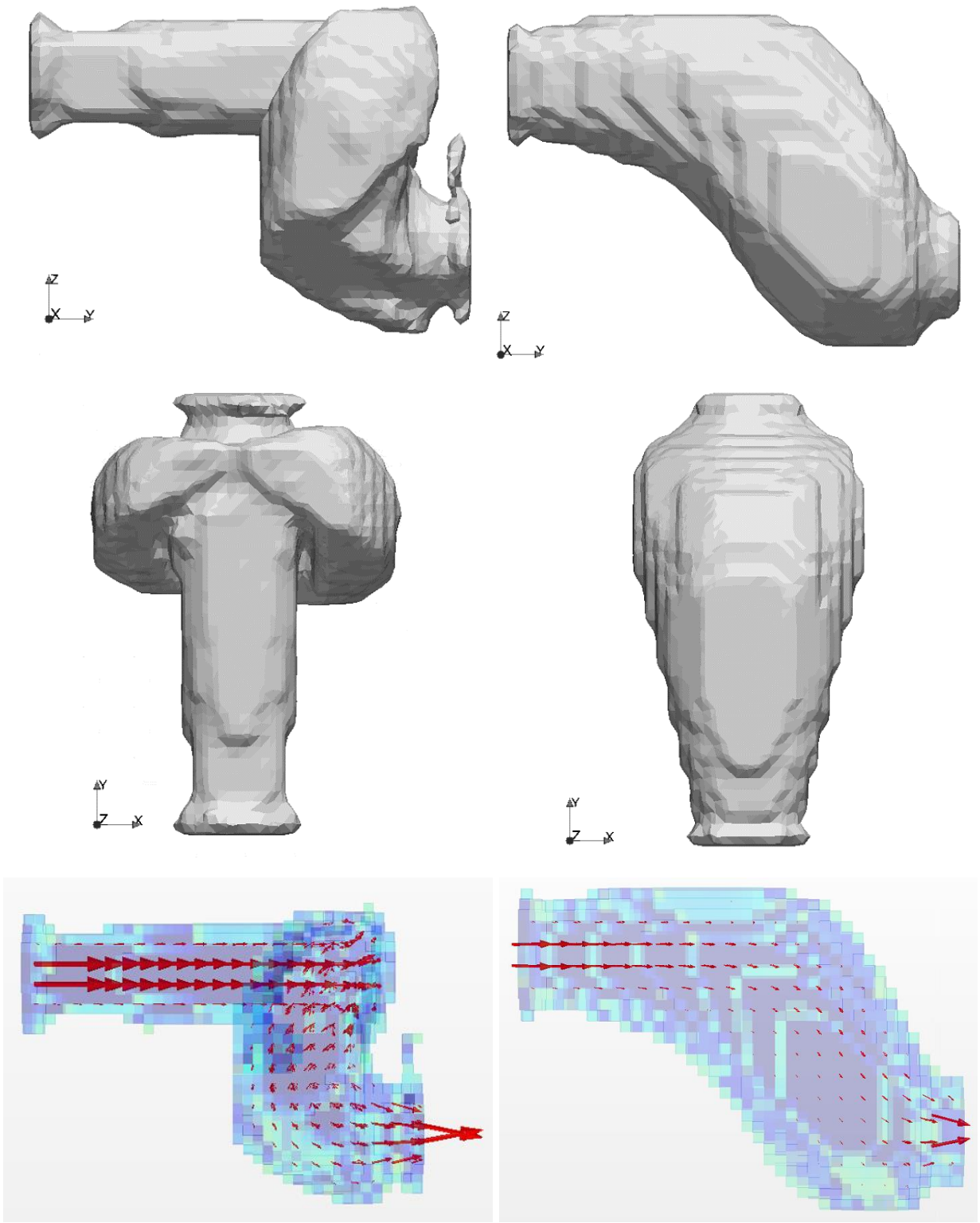

Fig. 8. Optimization results for the fluid flow problem and associated velocity field. Left, $v_0 = 0.002\ m/s$ and right, $v_0 = 0.0005\ m/s$.

V. Conclusion

In this work, we introduce an easy-to-use plug-and-play optimization workflow that allows the user to utilize state-of-the-art LSTO in C++ with any physics model created in the GUI of COMSOL Multiphysics. For ease of communication between the LSTO modules and the physics model from COMSOL Multiphysics, MATLAB is used as the interface between the two. The workflow was developed in such a way that only a few lines need to be modified from one model to another. Additionally, the capabilities of the framework were demonstrated with three numerical examples from three different governing equations, namely structural mechanics, heat transfer conduction, and fluid flow. This proposed workflow would make it easy to add new functionalities via customized functions either in MATLAB or in the LSTO modules in the C++ code directly. We believe that the proposed workflow will improve the accessibility of LSTO for newcomers in the field of TO and can foster cross-disciplinary collaborations amongst multidisciplinary sciences.

Acknowledgement

We gratefully acknowledge the support of Honda Performance Development and Samsung Display.

References

[1] Sigmund, O., Maute, K. “Topology optimization approaches.” *Structural and Multidisciplinary Optimization*, Vol. 48, 2013, pp. 1031-1055, doi.org/10.1007/s00158-013-0978-6

[2] Multiscale, Multiphysics Design Optimization website, http://m2do.ucsd.edu/software/, accessed 25 September 2022

[3] Wang, C., Zhao, Z., Zhou, M., Sigmund, O., and Zhang, X. S. “A Comprehensive review of educational articles on structural and multidisciplinary optimization.” *Structural and Multidisciplinary Optimization*, Vol. 64, 2021, pp. 2827-2880, doi:10.1007/s00158-021-03050-7.

[4] Dunning, P., Kim, H. A., “Introducing the sequential linear programming level-set method for topology optimization.” *Structural and Multidisciplinary Optimization*, Vol. 51, 2015, pp. 631-643, doi: 10.1007/s00158-014-1174-z

[5] Allaire, G., Jouve, F., Toader, A-M., “Structural optimization using sensitivity analysis and a level-set method. ” *Journal of Computational Physics*, Vol. 194, 2004, pp. 363-393, doi: 10.1016/j.jcp.2003.09.032

[6] Van Dijk, N.P., Maute, K., Langelaar, M., Van Keulen, F., “Level-set methods for structural topology optimization: a review.” *Structural and Multidisciplinary Optimization*, Vol. 48, 2013, pp. 437-472, doi: 10.1007/s00158-013-0912-y

[7] Brian Wong (2022). mexPackage (https://www.mathworks.com/matlabcentral/fileexchange/78655-mexpackage), MATLAB Central File Exchange. Retrieved 15 September 2022.

[8] Kambampati, S., Chung, H., and Kim, H. A. “A Discrete Adjoint Based Level Set Topology Optimization Method for Stress Constraints.” *Computer Methods in Applied Mechanics and Engineering*, Vol. 377, 2021, pp. 113563, doi:10.1016/j.cma.2020.113563.

[9] LiveLink for MATLAB, https://www.comsol.com/livelink-for-matlab, accessed 25 September 2022

[10] Sivapuram R., Dunning P.D. and Kim, H.A. "Simultaneous material and structural optimization by multiscale topology optimization." *Structural and multidisciplinary optimization*, Vol. 54, 2016, pp. 1267-1281, doi: 10.1007/s00158-016-1519-x

[11] Kambampati, S., Jauregui, C., Museth, K., Kim, H. A. “Large-scale level set topology optimization for elasticity and heat conduction.” *Structural and Multidisciplinary Optimization*, Vol. 61, 2020, pp. 19-38, doi.org/10.1007/s00158-019-02440-2

[12] Hyun, J., Wang, S., Yang, S., “Topology optimization of the shear thinning non-Newtonian fluidic systems for minimizing wall shear stress.” *Computers & Mathematics with Applications*, Vol. 67, 2014, pp. 1154-1170, doi.org/10.1016/j.camwa.2013.12.013