 doi:10.20944/preprints202210.0071.v1

MULTIPHYSICS TOPOLOGY OPTIMIZATION USING AUTOMATIC DIFFERENTIATION AND ADJOINT METHOD

A PREPRINT

Junyan He
Department of Mechanical Science and Engineering
University of Illinois at Urbana-Champaign

Diab W. Abueidda*
National Center for Supercomputing Applications
Department of Mechanical Science and Engineering
University of Illinois at Urbana-Champaign

ABSTRACT

An essential step in solving any topology optimization problem is determining the sensitivities of the objective function and optimization constraints. Unfortunately, these sensitivities are usually derived and implemented manually. Nontrivial objective functions and constraints, especially with the involvement of material and geometric nonlinearities, need strenuous mathematical derivation, leading to error-prone implementation. Another intriguing approach to finding sensitivities is automatic differentiation. This paper uses the automatic differentiation and adjoint method to find the sensitivities for two multiphysics topology optimization problems: 1) thermoelasticity and 2) piezoelectricity. This approach is not limited to these examples and can be easily extended to other single- or multi-physics topology optimization problems.

Keywords FEniCS · JAX · Sensitivity analysis · Topology optimization

1 Introduction

Topology optimization (TO) provides a systematic framework for the design process, which allocates material in a physical domain by optimizing an objective function while accounting for design constraints [1]. With enhanced computer performance and the development of finite element methods (FEM), TO has gained increasing interest from researchers and industries, especially with the recent improvements in the additive manufacturing realm [2, 3]. TO algorithms have been implemented for various material and structural optimization applications, including thermoelasticity [4, 5], plasticity [6, 7], heat conduction and convection systems [8, 9], wave propagation [10, 11], and others.

Recently, scientific machine learning has become of high interest to many fields, and TO is no exception. Several researchers have used deep learning to develop surrogate models for different TO problems [12, 13]. Although such models can infer the optimized designs almost instantly after having the model trained, they usually require high overhead costs to generate large data to train the model. Moreover, even after generating the data and training the model, the model's generalizability is questionable.

Finding the sensitivities of an objective function and design constraints is a pillar in gradient-based TO frameworks. The sensitivities can be usually determined using one of four methods [14]: 1) numerical differentiation, 2) symbolic differentiation, 3) manual derivation and implementation, and 4) automatic differentiation (AD). The numerical scheme is based on the finite difference method and is usually simple. Still, it is very time-consuming to the extent that it is not practical for many scenarios. The symbolic approach is limited to simple expressions, while manual derivation and implementation are arduous and error-prone, especially with the involvement of non-trivial objectives and/or constraints. Researchers have recently used automatic differentiation to quantify the sensitivities for TO problems [15].

*abueidd2@illinois.edu

Although automatic differentiation, also known as algorithmic differentiation, has been known for some time [16, 17], its impact still has not achieved its full potential, according to Griewank and Walther [18]. AD is a set of processes based on the chain rule to compute derivatives of a function using a computer program. AD helps evaluate the Jacobian of an arbitrarily complicated differentiable function by partitioning it into a series of simple and easily differentiable operations. By repeatedly applying the chain rule of derivative calculus to these operations, derivatives of arbitrary order can be computed automatically and accurately to working precision. AD is currently used in various areas such as numerical methods, inverse problems, design optimization, uncertainty quantification, and data assimilation.

Although using AD in TO stays limited to a few studies in the literature, it has been recently proven successful for several applications [15]. Dilgen et al. [19] presented an approach for TO of turbulent flow systems, where the gradients are computed using automatic differentiation. Additionally, Nørgaard et al. [20] used automatic differentiation to solve a non-trivial unsteady flow TO problem. In this work, we extend the application of automatic differentiation to multiphysics TO problems.

2 Methods

2.1 Thermoelasticity

A commonly encountered multi-physics problem in engineering is thermoelasticity accounting for thermal strains, which relates heat transfer with mechanical deformation. In the absence of any mechanical body load, the strong form of the governing equations reads:

$$\begin{aligned} \nabla \cdot \boldsymbol{\sigma} &= \mathbf{0}, \quad \forall \boldsymbol{X} \in \Omega, \\ -\nabla \cdot \boldsymbol{q} + g &= 0, \quad \forall \boldsymbol{X} \in \Omega, \\ \boldsymbol{u} &= \bar{\boldsymbol{u}}, \quad \forall \boldsymbol{X} \in \partial\Omega_u, \\ T &= \bar{T}, \quad \forall \boldsymbol{X} \in \partial\Omega_T, \end{aligned} \tag{1}$$

where g is the volumetric heat generation. In the small deformation setting, the strain $\boldsymbol{\epsilon}$ is given by:

$$\boldsymbol{\epsilon} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T). \tag{2}$$

For linear elastic, isotropic materials in the presence of thermal strain, the stress $\boldsymbol{\sigma}$ is given by:

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\epsilon} + \left[\lambda tr(\boldsymbol{\epsilon}) - \alpha(3\lambda + 2\mu)(T - T_{ref})\right] \boldsymbol{I}, \tag{3}$$

where α is the coefficient of thermal expansion. The heat flux vector $\boldsymbol{q}$ is given by Fourier's law:

$$\boldsymbol{q} = -k\nabla T, \tag{4}$$

where k is the thermal conductivity of the material.

To suppress the presence of intermediate element densities in TO, a SIMP-like scheme is used to scale the stress and heat flux:

$$\begin{aligned} \boldsymbol{\sigma}^*(\boldsymbol{\rho}) &= \boldsymbol{\sigma} \circ [\epsilon + (1-\epsilon)\boldsymbol{\rho}^p] \\ \boldsymbol{q}^*(\boldsymbol{\rho}) &= \boldsymbol{q} \circ [\epsilon + (1-\epsilon)\boldsymbol{\rho}^p], \end{aligned} \tag{5}$$

where the scaled quantities are denoted by a superscript *. $\epsilon = 1 \times 10^{-3}$ is the minimum scaling factor for numerical stability to ensure that the global stiffness matrix is non-singular. $p = 3$ is the SIMP penalization exponent, and $\circ$ denotes the element-wise Hadamard product.

For simplicity, we used linear Lagrange finite elements for the displacements and temperature degrees of freedom (DOF). These DOFs are defined at the nodes of the finite element. Linear discontinuous Lagrange finite elements are used to represent the element design density, which is defined at the center of the finite element. The weak form of the coupled thermomechanical problem stated as in a residual form, reads:

$$R(\boldsymbol{u}, T, \boldsymbol{\rho})_i = \int_\Omega \left[\nabla \boldsymbol{w}_i \cdot \boldsymbol{\sigma}^*(\boldsymbol{u}, \boldsymbol{\rho}) + \nabla v_i \cdot \boldsymbol{q}^*(T, \boldsymbol{\rho})\right] dA + \int_\Omega v_i \cdot g \, dA = 0, \tag{6}$$

where $\boldsymbol{w}_i$, v_i denote the i^{th} test functions for the displacement DOFs and temperature DOFs, respectively. Newton's method is used to solve the problem $\boldsymbol{R} = \mathbf{0}$:

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\lambda}} \Delta\boldsymbol{\lambda} = -\boldsymbol{R}, \tag{7}$$

where $\boldsymbol{\lambda} = [\boldsymbol{u}, T]^T$ denotes all nodal DOFs. $\Delta\boldsymbol{\lambda}$ is the change in DOF values in the current Newton step, and $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{\lambda}}$ is the Jacobian obtained by applying AD on the residual vector $\boldsymbol{R}$.

For heat transfer and mechanical deformation, two common objective functions can be considered. They are the thermal (C_{th}) and mechanical (C_{me}) compliances:

$$\begin{aligned} C_{\text{th}} &= -\int_{\Omega} \boldsymbol{q}^* \cdot \nabla T \, dA \\ C_{\text{me}} &= \int_{\Omega} \boldsymbol{\sigma}^* : \boldsymbol{\epsilon} \, dA. \end{aligned} \tag{8}$$

In the multiphysics setting, we considered the following normalized objective function f [21]:

$$f(\boldsymbol{u}, T, \boldsymbol{\rho}) = w_1 \bar{C}_{\text{th}} + w_2 \bar{C}_{\text{me}} = w_1 \frac{C_{\text{th}} - C_{\text{th}}^f}{C_{\text{th}}^0 - C_{\text{th}}^f} + w_2 \frac{C_{\text{me}} - C_{\text{me}}^f}{C_{\text{me}}^0 - C_{\text{me}}^f}, \tag{9}$$

where $\bar{C}_{\text{th}}$ and $\bar{C}_{\text{me}}$ denote the normalized thermal and mechanical compliances, respectively. w_1 and w_2 denote the weights for the two single-physics objective functions. The weights range from 0 to 1 and satisfy the constraint $w_1 + w_2 = 1$. The Pareto front of the problem can be generated by solving the TO problem repeatedly with different values of w_1. Quantities marked with superscripts ^{f} and ^{0} denote the final (optimized) and initial values obtained from single-physics TOs by setting the corresponding weight to 1. Using this normalized objective function, we state the multiphysics TO problem as:

$$\begin{aligned} \min_{\boldsymbol{\rho}} \; & f(\boldsymbol{u}, T, \boldsymbol{\rho}) \\ \text{s.t. } & \boldsymbol{R}(\boldsymbol{u}, T, \boldsymbol{\rho}) = \boldsymbol{0} \\ & V_c = \frac{1}{N} \sum_{i=1}^{N} \rho_i - \bar{V}_f \leq 0 \\ & 0 \leq \rho_e \leq 1, \, \forall e = 1 \cdots N, \end{aligned} \tag{10}$$

where N is the number of finite elements in the mesh and $\bar{V}_f$ is the target volume fraction. The gradient-based method of moving asymptote (MMA) method [22] is used to update the density. As such, the density-space gradients $\frac{\partial f}{\partial \boldsymbol{\rho}}$ and $\frac{\partial V_c}{\partial \boldsymbol{\rho}}$ are needed by the MMA solver, and are obtained from AD using JAX [23] directly, and no analytical sensitivity analysis is needed. To suppress checkerboarding, density filtering is applied. The entries of the normalized filter matrix $\boldsymbol{H}$, which is denoted by $\bar{q}_{ij}$, are given by:

$$\begin{aligned} q_{ij} &= \max(r_{min} - ||\boldsymbol{X}_i - \boldsymbol{X}_j||), \\ \bar{q}_{ij} &= \frac{1}{\sum_{k=1}^{N} q_{ik}} q_{ij}. \end{aligned} \tag{11}$$

where r_{min} is the filter radius.

2.2 Piezoelectricity

Another commonly seen multiphysics problem with great practical interest is piezoelectricity, which can be used in designing piezoelectric actuators and energy harvesters. In this example, we consider the design of a 2D piezoelectric actuator subjected to an applied voltage. The planar actuator is assumed to lie in the XY plane, and is under plane stress assumption. The thickness of the plate in the Z-direction is t. An external voltage $\bar{V}$ is applied in the Z direction on the Z+ face, and the Z- face has a constant voltage of 0. Further, it is assumed that the voltage is uniform in the XY plane, and varies linearly in the Z direction. These assumptions follow from the work of Homayouni-Amlashi et al. [24]. In the absence of any body force and externally applied traction, the strong form of the problem can be written as:

$$\begin{aligned} \nabla \cdot \boldsymbol{\sigma} &= \boldsymbol{0}, \quad \forall \boldsymbol{X} \in \Omega, \\ \nabla \cdot \boldsymbol{D} &= 0, \quad \forall \boldsymbol{X} \in \Omega, \\ \boldsymbol{u} &= \bar{\boldsymbol{u}}, \quad \forall \boldsymbol{X} \in \partial\Omega_u, \end{aligned} \tag{12}$$

where $\boldsymbol{u}$, $\boldsymbol{\sigma}$, and $\boldsymbol{D}$ denote the displacement, stress, and electric displacement, respectively. Under the plane-stress and uniform in-plane voltage assumption, the constitutive laws for $\boldsymbol{\sigma}$ and $\boldsymbol{D}$ can be simplified. For piezoelectric actuation under an applied voltage, the constitutive law for a linear elastic material is given in its 2D reduced form by [25]:

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ D_3 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & 0 & -e_{31} \\ C_{12} & C_{22} & 0 & -e_{31} \\ 0 & 0 & C_{33} & 0 \\ e_{31} & e_{31} & 0 & p_{33} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ E_3 \end{bmatrix}, \tag{13}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0071.v1

where C_{ij} , e_{ij}, p_{33} denote the plane stress stiffness coefficients, piezoelectric coefficients, and material permittivity, respectively. $E_3 = \bar{V}/t$ is the electric field strength in the Z-direction.

In this work, we used the modified SIMP scheme proposed in [26] to scale the material properties as a function of the element design density, which is specialized to piezoelectric materials:

$$\begin{aligned} C^*_{ij}(\rho) &= C_{ij,min} + (C_{ij} - C_{ij,min})\rho^{p_c} \\ e^*_{31}(\rho) &= e_{31,min} + (e_{31} - e_{31,min})\rho^{p_e}, \end{aligned} \tag{14}$$

where $C_{ij,min} = e_{31,min} = 1 \times 10^{-6}$ is the minimum value of the material moduli. $p_c = 3$ and $p_e = 4$ are the penalty exponents for the elastic and piezoelectric constants. The SIMP-penalized stress $\boldsymbol{\sigma}^*(\rho)$ can be calculated by using $C^*_{ij}(\rho)$ and $e^*_{31}(\rho)$ in Eq. 13.

Similar to Section 2.1, linear Lagrange and discontinuous Lagrange finite elements are used for the nodal displacements and design density, respectively. The weak form of the problem in residual form reads:

$$R(\boldsymbol{u}, \boldsymbol{\rho})_i = \int_\Omega \nabla \boldsymbol{w}_i \cdot \boldsymbol{\sigma}^*(\boldsymbol{u}, \boldsymbol{\rho})\, dA = 0. \tag{15}$$

which is solved using Newton's method with the Jacobian computed from AD. The density-space gradients are obtained using AD once the displacement vector is found.

A commonly used objective function when designing actuators is to maximize the force on the output node of the mechanism. Numerically, this is achieved by connecting a linear spring of stiffness k to the output node, so that the reaction force on the output node is given by:

$$F_{out} = k u_{out}, \tag{16}$$

where u_{out} is the displacement of the output node along the output direction. Using this objective function, the TO problem can be stated as:

$$\begin{aligned} \min_{\boldsymbol{\rho}} \; & -F_{out}(\boldsymbol{u}, \bar{V}, \boldsymbol{\rho}) \\ \text{s.t. } & \boldsymbol{R}(\boldsymbol{u}, \bar{V}, \boldsymbol{\rho}) = \boldsymbol{0} \\ & V_c = \frac{1}{N}\sum_{i=1}^{N} \rho_i - \bar{V}_f \leq 0 \\ & 0 \leq \rho_e \leq 1, \, \forall e = 1 \cdots N. \end{aligned} \tag{17}$$

2.3 The FEniCS-JAX-based framework

The current framework is based on the *jax-fenics-adjoint* package by Yashchuk et. al [27], which combines the general purpose FEM code FEniCS [28] and the Python-based AD package JAX [23]. We demonstrate the core workflow of the framework using the thermoelasticity example (Section 2.1) with code snippets.

First, we define a quadrilateral mesh for the computational domain using:

```
import fenics_adjoint as fa
M = fa.RectangleMesh.create([fn.Point(0.0, 0.0), fn.Point(L, h)],\\
    [nelx, nely], fn.CellType.Type.quadrilateral)
```

where L, h, $nelx$, $nely$ denote the dimensions and number of elements along the X- and Y-directions. Note that the *fenics_adjoint* package provides many other built-in functions to construct meshes for more complicated domains, and even supports importing a previously generated mesh. Then, we define the appropriate FE function spaces for the element density, nodal displacement, and nodal temperature given a mesh M:

```
import fenics as fn
C   = fn.FunctionSpace(mesh, "DG", 0)             # Design density
Vue = fn.VectorElement('CG', mesh.ufl_cell(), 2) # Displacement DOFs, vector
Vte = fn.FiniteElement('CG', mesh.ufl_cell(), 1) # Temperature DOFs, scalar
V   = fn.FunctionSpace(mesh, fn.MixedElement([Vue, Vte])) # Mixed finite element space
```

We define a function forward(rho) to solve the forward coupled FE problem given the current density vector $\boldsymbol{\rho}$:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0071.v1


```
import ufl
solve_templates    = (fa.Function(C),)
@build_jax_fem_eval(solve_templates)
def forward( rho ):
    # Get trial and test functions
    dU           = fn.TrialFunction(V)
    (du, dTheta) = fn.split(dU)
    U_           = fn.TestFunction(V)
    (u_, Theta_) = fn.split(U_)

    flux   = -K * grad(dTheta) * simp( rho ) # Heat flux, Eq. (4)
    strain = sym(grad(du)) # Strain, Eq. (2)
    # Stress, Eq. (3)
    stress = ( (lmbda*tr(strain)- alpha*(3*lmbda+2*mu)*dTheta) * Identity(2) +\\
        2.0*mu*strain ) * simp( rho )

    # Mechanical residual
    mech_form  = inner( stress, sym(grad(u_)) ) * dx - dot(b, u_) * ds
    # Thermal residual
    therm_form = -inner(flux, grad(Theta_)) * dx - g * Theta_ * dx
    # Total residual, Eq. (6)
    F = mech_form + therm_form

    # Solve
    U = fa.Function(V)
    fa.solve(ufl.lhs(F) == ufl.rhs(F), U, bcs)
    return U
```

Note that a succinct line:

```
    fa.solve( ufl.lhs(F) == ufl.rhs(F), U, bcs)
```

derives the Jacobian matrix by AD using the definition of F, and solves the resulting discrete problem. Then, we define the objective function and volume constraint for TO:

```
import numpy as np
from MMA import *
assemble_templates = (fa.Function(W), fa.Function(A))
@build_jax_fem_eval(assemble_templates)
def obj_function( rho_raw ):
    rho = H @ rho_raw # Apply density filter
    w = forward(x) # Solve
    u, T   = fn.split(w) # Split components

    # Compliance calculation
    J_heat = g * T * dx # Thermal compliance, Eq. (8)-1
    strain = sym(grad(u))
    stress = ( (lmbda*tr(strain)- alpha*(3*lmbda+2*mu)*T) * Identity(2) +\\
        2.0*mu*strain ) * simp( rho )
    J_mech = inner( stress, sym(grad(u)) ) * dx # Mechanical compliance, Eq. (8)-2

    # Normalize components, Eq. (9)
    coeff1 = weight_mech / ( f_mech_ini - f_mech_opt )
    mech_part = coeff1 * J_mech
    coeff2 = weight_thermo / ( f_thermal_ini - f_thermal_opt )
    thermal_part = coeff2 * J_heat

    return fa.assemble( mech_part + thermal_part )
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0071.v1


```
def volume_constraint( rho ):
    return np.mean( rho ) - Vf
```

Finally, we define the function handles to obtain the derivative of the objective function and the volume constraint through JAX and sets up the MMA solver. A Python implementation of the MMA solver by Chandrasekhar et al. [15] was used.

```
def min_f( rho_raw ):
    value, grad = jax.value_and_grad(obj_function)( rho_raw )
    return np.array(value), np.array(grad)

def min_volcon( rho_raw ):
    value, grad = jax.value_and_grad(volume_constraint)( rho_raw )
    return np.array(value), np.array(grad)

# Perform TO
x0 = np.ones(A.dim()) * Vf # Initial condition
mech_part = 0.5; weight_thermo = 0.5
optimizationParams = {'maxIters':40,'minIters':40,'relTol':0.001} # MMA settings
final_rho , final_obj = optimize( x0 , optimizationParams , min_f , min_volcon )
```

3 Results and discussion

In this section, we present two numerical examples that demonstrate our AD-based TO framework for multi-physics problems. In the first example, we plot the Pareto front for a thermo-mechanical bi-objective TO problem. In the second example, we design a piezoelectric actuator and study how the optimized mechanism design changes with the spring stiffness k.

3.1 Thermoelasticity and the Pareto front

Consider a rectangular domain of size 3×1 m^2, discretized into a 150×50 mesh of 2D bi-linear quadrilateral finite elements. Unit thickness was assumed in the out-of-plane direction. The domain was fully fixed on the left edge and was subjected to a downward point load $P = 10000$ at the center of its right edge. A uniform heat generation $g = 2$ J/m^3 is applied, and a constant temperature $T = 0$ is maintained on a portion of the left edge where $0.4 \leq Y \leq 0.6$ m. The material properties used in this example are $E = 70000$ MPa, $\nu = 0.3$, and $k = 10$ W/(m·K). A filter radius of 0.06 m was used, which is equivalent to 3 element lengths. 11 uniformly spaced values for w_1 were generated from the range $[0, 1]$ to show how the optimized design changes with w_1 and to plot the Pareto front [29]. A total of 40 TO iterations were conducted for each w_1 value, with a target volume fraction of 0.4. The Pareto front for the bi-objective problem and optimized designs generated with different w_1 values are shown in Fig. 1. The time required to solve the forward FE analysis and the backward adjoint analysis (done automatically through AD) is compared in Fig. 2.

From Fig. 1, we see that the generated Pareto front is qualitatively similar to the thermoelasticity problem studied in the work of [21] (see Fig. 23 therein). The final designs generated with different w_1 values are distinct, showing that the AD-based TO framework can trace the difference in the objective function, and reflect it in the final optimized designs. When comparing the time required for the forward and backward analyses, we notice that both are very comparable for all 11 different values of w_2. This is because the *jax-fenics-adjoint* package internally assembles the adjoint problem and obtains the adjoint vector by solving a separate linear system, which is of equal size to the forward FE problem. This is a major disadvantage of this current approach, as the adjoint analysis, although completely automated, is computationally as expensive as the forward analysis. However, the fact that the adjoint analysis is automated can be very beneficial when the contributing physics becomes nonlinear, whereas analytical sensitivity analysis is time-consuming and error-prone.

3.2 Design of a piezoelectric actuator

In this example, we consider a rectangular domain of size 1×0.5 m^2, with a thickness $t = 1 \times 10^{-4}$ m. The in-plane geometry was discretized into a 130×65 mesh. Mirror symmetry ($u_y = 0$) is applied at the bottom edge. The geometry is fixed on its left edge. The applied voltage is $\bar{V} = 1$ V. The output location of the actuator mechanism is the bottom-right corner element (its two nodes on the right edge), which is connected to a 'spring'. However, as a discrete spring is not supported in FEniCS, we connected an additional quadrilateral finite element to the right of the output

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0071.v1

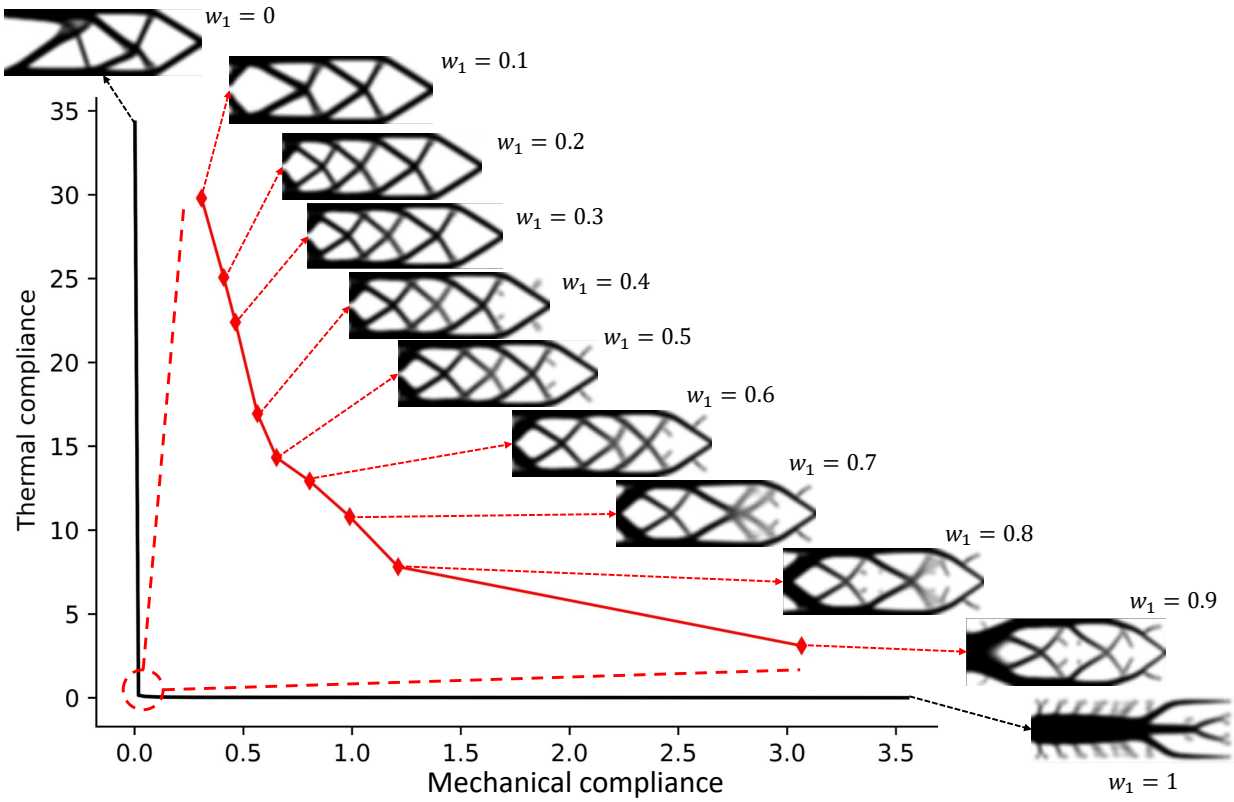


Figure 1: Pareto front for the 2D thermoelasticity problem. Final optimized designs at different weight values are also shown.

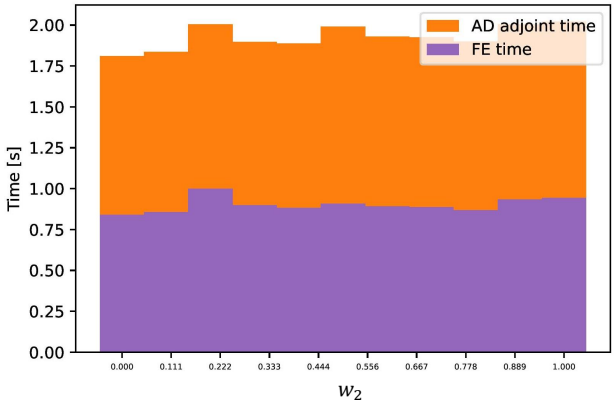
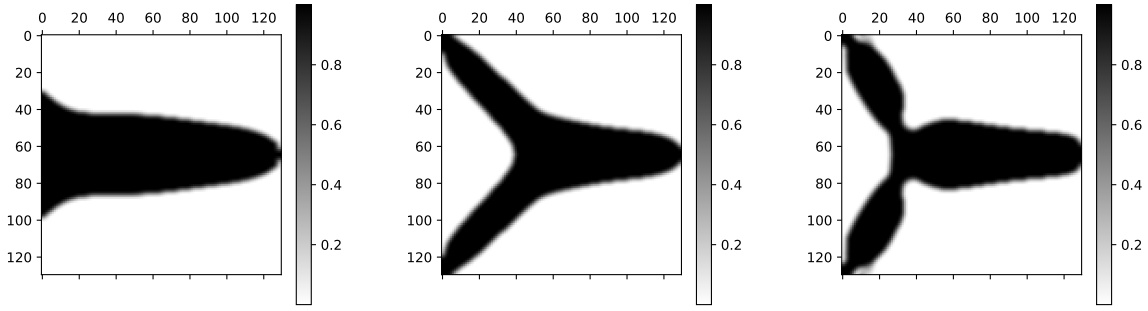


Figure 2: Time required for the forward FE analysis and backward adjoint analysis, for different values of w_2.

element and fixed the right edge of the additional element, which effectively simulates the addition of a discrete spring. The stiffness of the 'spring' can be adjusted by assigning different design densities to the additional element. Three different density values were simulated for the additional element, they were 1, 0.3, and 0.15. TO settings are identical to the previous example. The material properties used in this example were adopted from [24] and are presented in Table 1. The final, optimized geometries are shown in Fig. 3; note that we have applied symmetry to plot the full design. With different output 'spring' stiffness, the actuator designs changed. The three generated designs are qualitatively

Table 1: Material properties used in the piezoelectric actuator example

	C_{11} [Pa]	C_{22} [Pa]	C_{12} [Pa]	C_{33} [Pa]	e_{31} [Pa · m/V]	p_{33} [F/m]
Value	9.1187×10^{10}	C_{11}	3.0025×10^{10}	3.0581×10^{10}	-14.9091	7.84×10^{-9}



(a) $\rho_{\text{spring}} = 1$ (b) $\rho_{\text{spring}} = 0.3$ (c) $\rho_{\text{spring}} = 0.15$

Figure 3: Optimized piezoelectric actuators generated from different output spring stiffness.

similar to those presented in [24], indicating that the AD-based TO framework can generate designs similar to those with analytical sensitivity analysis. Again, the advantage of the current work is that the user only needs to define the forward FE analysis and the calculation of the objective function and constraints, and there is no need to analytically derive the density-space gradients used in TO.

4 Conclusions and future directions

This work is an educational paper that advocates the use of automatic differentiation in multiphysics topology optimization problems. Automatic differentiation is beneficial as it frees the user from having to analytically derive the sensitivity of the objective function with respect to the design density, whose calculation can be time-consuming and error-prone. With the help of automatic differentiation, the user only needs to define the forward finite element problem, the objective function, and the design constraints. The current framework then applies automatic differentiation to the user-defined objective and constraints to perform gradient-based topology optimization using the MMA solver.

The current framework is Python-based, which is based on the *jax-fenics-adjoint* package. It combines the versatility and wide user support of the FEniCS platform with the power of the JAX package for automatic differentiation. We demonstrate our framework using two common multiphysics examples: the first one being a bi-objective thermoelasticity optimization, and the second one being the design of a piezoelectric actuator. In both examples, we showed that the current framework generates similar designs compared to previous works that were based on analytical sensitivity analysis. We note that the time required for the adjoint analysis is comparable to that of the forward finite element analysis time. In the future, we plan to apply the current framework to other multiphysics problems such as fluid-structure interaction and coupled thermo-fluid problems and to investigate the use of this framework with nonlinear materials and under large deformation settings.

Acknowledgment

The authors would like to thank the National Center for Supercomputing Applications (NCSA) Industry Program and the Center for Artificial Intelligence Innovation.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

[1] M. P. Bendsoe, O. Sigmund, Topology Optimization: Theory, Methods, and Applications, Springer Science & Business Media, 2003.

[2] K. V. Wong, A. Hernandez, A review of additive manufacturing, International scholarly research notices 2012.

[3] D. W. Abueidda, M. Elhebeary, C.-S. A. Shiang, R. K. A. Al-Rub, I. M. Jasiuk, Compression and buckling of microarchitectured neovius-lattice, Extreme Mechanics Letters 37 (2020) 100688.

[4] H. Rodrigues, P. Fernandes, A material based model for topology optimization of thermoelastic structures, International Journal for Numerical Methods in Engineering 38 (12) (1995) 1951–1965.

[5] H. Chung, O. Amir, H. A. Kim, Level-set topology optimization considering nonlinear thermoelasticity, Computer Methods in Applied Mechanics and Engineering 361 (2020) 112735.

[6] D. W. Abueidda, Z. Kang, S. Koric, K. A. James, I. M. Jasiuk, Topology optimization for three-dimensional elastoplastic architected materials using a path-dependent adjoint method, International Journal for Numerical Methods in Engineering 122 (8) (2021) 1889–1910.

[7] M. Wallin, V. Jönsson, E. Wingren, Topology optimization based on finite strain plasticity, Structural and multidisciplinary optimization 54 (4) (2016) 783–793.

[8] S. Yan, F. Wang, O. Sigmund, On the non-optimality of tree structures for heat conduction, International Journal of Heat and Mass Transfer 122 (2018) 660–680.

[9] J. Alexandersen, O. Sigmund, N. Aage, Large scale three-dimensional topology optimisation of heat sinks cooled by natural convection, International Journal of Heat and Mass Transfer 100 (2016) 876–891.

[10] J. Dahl, J. S. Jensen, O. Sigmund, Topology optimization for transient wave propagation problems in one dimension, Structural and Multidisciplinary Optimization 36 (6) (2008) 585–595.

[11] B. S. Lazarov, R. Matzen, Y. Elesin, Topology optimization of pulse shaping filters using the hilbert transform envelope extraction, Structural and Multidisciplinary Optimization 44 (3) (2011) 409–419.

[12] D. W. Abueidda, S. Koric, N. A. Sobh, Topology optimization of 2d structures with nonlinearities using deep learning, Computers & Structures 237 (2020) 106283.

[13] F. V. Senhora, H. Chi, Y. Zhang, L. Mirabella, T. L. E. Tang, G. H. Paulino, Machine learning for topology optimization: Physics-based learning through an independent training strategy, Computer Methods in Applied Mechanics and Engineering 398 (2022) 115116.

[14] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, Journal of Marchine Learning Research 18 (2018) 1–43.

[15] A. Chandrasekhar, S. Sridhara, K. Suresh, Auto: a framework for automatic differentiation in topology optimization, Structural and Multidisciplinary Optimization 64 (6) (2021) 4355–4365.

[16] A. Griewank, A. Walther, Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation, ACM Transactions on Mathematical Software (TOMS) 26 (1) (2000) 19–45.

[17] Q. Wang, P. Moin, G. Iaccarino, Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation, SIAM Journal on Scientific Computing 31 (4) (2009) 2549–2567.

[18] A. Griewank, A. Walther, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, SIAM, 2008.

[19] C. B. Dilgen, S. B. Dilgen, D. R. Fuhrman, O. Sigmund, B. S. Lazarov, Topology optimization of turbulent flows, Computer Methods in Applied Mechanics and Engineering 331 (2018) 363–393.

[20] S. A. Nørgaard, M. Sagebaum, N. R. Gauger, B. S. Lazarov, Applications of automatic differentiation in topology optimization, Structural and Multidisciplinary Optimization 56 (5) (2017) 1135–1146.

[21] M. A. Ali, M. Shimoda, Toward multiphysics multiscale concurrent topology optimization for lightweight structures with high heat conductivity and high stiffness using matlab, Structural and Multidisciplinary Optimization 65 (7) (2022) 1–26.

[22] K. Svanberg, The method of moving asymptotes—a new method for structural optimization, International Journal for Numerical Methods in Engineering 24 (2) (1987) 359–373.

[23] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018). URL `http://github.com/google/jax`

[24] A. Homayouni-Amlashi, T. Schlinquer, A. Mohand-Ousaid, M. Rakotondrabe, 2D topology optimization matlab codes for piezoelectric actuators and energy harvesters, Structural and Multidisciplinary Optimization 63 (2) (2021) 983–1014.

[25] C. D. M. Junior, A. Erturk, D. J. Inman, An electromechanical finite element model for piezoelectric energy harvester plates, Journal of Sound and Vibration 327 (1-2) (2009) 9–25.

[26] M. Kögl, E. C. Silva, Topology optimization of smart structures: design of piezoelectric plate and shell actuators, Smart materials and Structures 14 (2) (2005) 387.

[27] I. Yashchuk, C. Carroll, S. S. Sankaran, jax-fenics-adjoint, `https://github.com/IvanYashchuk/jax-fenics-adjoint` (2021).

[28] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, G. N. Wells, The fenics project version 1.5, Archive of Numerical Software 3 (100).

[29] I. Y. Kim, O. L. De Weck, Adaptive weighted-sum method for bi-objective optimization: Pareto front generation, Structural and Multidisciplinary Optimization 29 (2) (2005) 149–158.