AMORTIZED FINITE ELEMENT ANALYSIS FOR FAST PDE-CONSTRAINED OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing the parameters of partial differential equations (PDEs), i.e., PDE-constrained optimization (PDE-CO), allows us to model natural systems from observations or perform rational design of structures with complicated mechanical, thermal, or electromagnetic properties. However, PDE-CO is often computationally prohibitive due to the need to solve the PDE—typically via finite element analysis (FEA)—at each step of the optimization procedure. In this paper we propose amortized finite element analysis (AmorFEA), in which a neural network learns to produce accurate PDE solutions, while preserving many of the advantages of traditional finite element methods. As FEA is a variational procedure, AmorFEA is a direct analogue to popular amortized inference approaches in latent variable models, with the finite element basis acting as the variational family. AmorFEA can perform PDE-CO without the need to repeatedly solve the associated PDE, accelerating optimization when compared to a traditional workflow using FEA and the adjoint method.

1 INTRODUCTION

Partial differential equations (PDEs) are widely used to describe the properties of physical systems, including heat transfer, electromagnetics, and elasticity. PDE-constrained optimization (PDE-CO) addresses the situation in which an objective function must be minimized minimized or maximized, subject to the constraints of real-world physics as expressed by PDEs. Common examples include optimal design, optimal control, and the identification of parameters to relate a simulation to observed data (Rees et al., 2010; Ulbrich & van Bloemen Waanders, 2018). PDE-CO can be computationally challenging, however, for even moderately sized problems, as it is usually necessary to solve the associated PDE at every iteration of the outer-loop optimization, e.g., by the widely used finite element analysis (FEA) (Hughes, 2012).

We propose a two-stage optimization framework to efficiently tackle sequences of related PDE-constrained optimization problems. At the first stage, we introduce amortized finite element analysis (AmorFEA) to efficiently learn the physics governed by the PDE without requiring supervised data provided by expensive PDE solvers. We borrow the idea of *amortized optimization*, widely used in amortized variational inference (Kingma & Welling, 2013; Gershman & Goodman, 2014; Ravi & Beatson, 2018; Choi et al., 2019). By learning to jump directly to the FEA solution with a neural network, we obtain a surrogate model that is able to predict the solutions directly from the control parameters. At the second stage, we perform gradient-based PDE-CO using the AmorFEA-enabled neural surrogate model. During each optimization iteration, the gradient is efficiently computed via one forward and one backward pass through the neural network, instead of querying an expensive PDE solver as in the traditional adjoint method requires (Cao et al., 2003).

2 AMORTIZED FINITE ELEMENT ANALYSIS

Many physical systems governed by PDEs obey variational principles. For a given control field λ (e.g., heat source), the true solution field u (e.g., temperature) is the one that minimizes the total potential energy of the system. In a discretized FEA problem (see a full description of FEA in Appendix A), given a fixed control vector $\boldsymbol{\lambda} \in \mathbb{R}^m$, we find the solution vector (or the state vector) $\boldsymbol{u} \in \mathbb{R}^n$ by solving an optimization problem:

$$\min_{\boldsymbol{u}\in\mathbb{R}^n} \mathcal{L}(\boldsymbol{u},\boldsymbol{\lambda}), \tag{1}$$

where $\mathcal{L}(\boldsymbol{u},\boldsymbol{\lambda})$ denotes the total potential energy.

Our amortized finite element analysis (AmorFEA) approach reframes the per-control-vector optimization process into a shared regression problem. We use a neural network to build a deterministic mapping $g_\psi : \mathbb{R}^m \to \mathbb{R}^n$, whose weights ψ are learned by minimizing the expected potential energy:

$$\min_{\psi} \mathbb{E}_{p(\boldsymbol{\lambda})}[\mathcal{L}(g_\psi(\boldsymbol{\lambda}),\boldsymbol{\lambda})], \tag{2}$$

where $p(\boldsymbol{\lambda})$ is the distribution of typical control parameters associated with specific problems and we hope that $\boldsymbol{u} \approx \widehat{\boldsymbol{u}} = g_\psi(\boldsymbol{\lambda})$. Since FEA is an approximate variational procedure (in the general sense of minimizing a functional), it resembles variational inference in latent variable models in that the finite element basis functions are analogous to the variational family of distributions and the FEA solution vector mirrors the variational parameters.

Amortization Suboptimality For amortized variational inference, Cremer et al. (2018) introduced the notions of *approximation*, *amortization*, and *inference gap*. Similar to gaps for amortized inference, we propose the approximation, amortization and "inference" (total error) gaps for AmorFEA:

$$\Delta_{\text{ap}} = \mathbb{E}_{p(\boldsymbol{\lambda})}\Big[\mathcal{L}(\boldsymbol{u}^\star_{\boldsymbol{\lambda}},\boldsymbol{\lambda}) - \mathcal{L}(u^e,\lambda)\Big] \tag{3}$$

$$\Delta_{\text{am}} = \mathbb{E}_{p(\boldsymbol{\lambda})}\Big[\mathcal{L}(g_{\psi^\star}(\boldsymbol{\lambda}),\boldsymbol{\lambda}) - \mathcal{L}(\boldsymbol{u}^\star_{\boldsymbol{\lambda}},\boldsymbol{\lambda})\Big] \tag{4}$$

$$\Delta_{\text{inf}} = \Delta_{\text{ap}} + \Delta_{\text{am}}, \tag{5}$$

where $\psi^\star$ is the optimal solution to Eq. 2, $\boldsymbol{u}^\star_{\boldsymbol{\lambda}}$ is the optimal solution to Eq. 1 for a given $\boldsymbol{\lambda}$, and u^e is the exact solution to the governing PDE which is generally impossible to obtain. Note that we are abusing notation somewhat here in allowing $\mathcal{L}(\cdot,\cdot)$ to take inputs of either functions (e.g., normal font u) or FEA vectors (e.g., bold font $\boldsymbol{u}$).

PDE-constrained Optimization The discretized PDE-constrained optimization is formulated as

$$\min_{\boldsymbol{u}\in\mathbb{R}^n,\boldsymbol{\lambda}\in\mathbb{R}^m} \mathcal{J}(\boldsymbol{u},\boldsymbol{\lambda})$$
$$\text{s.t.} \quad \boldsymbol{c}(\boldsymbol{u},\boldsymbol{\lambda}) = 0, \tag{6}$$

where $\mathcal{J}(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the objective function and $\boldsymbol{c}(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the constraint function imposed by the governing PDE. A reduced formulation is often used to embed the PDE constraint as

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}^m} \widehat{\mathcal{J}}(\boldsymbol{\lambda}), \tag{7}$$

where $\widehat{\mathcal{J}}(\boldsymbol{\lambda}) := \mathcal{J}(\boldsymbol{u}(\boldsymbol{\lambda}),\boldsymbol{\lambda})$ and $\boldsymbol{u}(\boldsymbol{\lambda})$ is the implicit function arising from the solution to Eq. 1. Gradient-based optimization algorithms require the evaluation of the derivative of the objective function with respect to the control vector:

$$\frac{d\widehat{\mathcal{J}}}{d\boldsymbol{\lambda}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\frac{d\boldsymbol{u}}{d\boldsymbol{\lambda}} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{\lambda}}. \tag{8}$$

A common way to compute this gradient is to use the adjoint method Cao et al. (2003) (see Appendix B). We propose to accelerate PDE-CO with AmorFEA. AmorFEA yields a differentiable map from the control vector $\boldsymbol{\lambda}$ to the state vector $\boldsymbol{u}$, which does not require solving the governing PDE via traditional FEA, and which can be used to approximate the costly term $\frac{d\boldsymbol{u}}{d\boldsymbol{\lambda}}$ in Eq. 8. With the differentiable neural surrogate model, we formulate the PDE-constrained optimization problem as

$$\min_{\boldsymbol{\lambda}\in\mathbb{R}^m} \widetilde{\mathcal{J}}(\boldsymbol{\lambda}), \tag{9}$$

where $\widetilde{\mathcal{J}}(\boldsymbol{\lambda}) := \mathcal{J}(g_{\psi^\star}(\boldsymbol{\lambda}),\boldsymbol{\lambda})$ and $\psi^\star$ are the learned weights of the neural network. The derivative is given by

$$\frac{d\widetilde{\mathcal{J}}}{d\boldsymbol{\lambda}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\frac{dg_{\psi^\star}}{d\boldsymbol{\lambda}} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{\lambda}}. \tag{10}$$

We employ reverse-mode automatic differentiation, since $m \gg 1$. A summary for the procedures of AmorFEA along with AmorFEA based PDE-CO can be found in Fig. 1 by visualizing the computation graph. We demonstrate AmorFEA using a simple linear example in Appendix C.

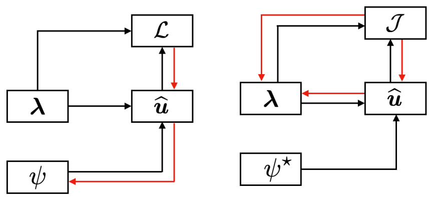

Figure 1: Computation graph for AmorFEA based PDE-CO. Red arrows denote automatic differentiation. Left: AmorFEA training for the surrogate model. Right: PDE-CO with the learned model.

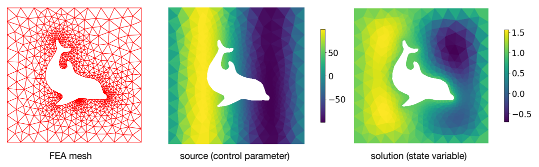

Figure 2: Setup of the problem. Left: the finite element mesh for a irregular domain with a dolphin-shaped hole. Middle: an example source field $\lambda(x) = 100\sin(2\pi x_1)$. Right: the solution field $u(x)$ associated with the source field.

3 Examples

Source Field Finding We consider a two-dimensional optimal source control problem simplified from superconductivity theory (De Melo et al., 1993). Mathematically, the problem is to minimize the functional

$$\mathcal{J}(u,\lambda) = \frac{1}{2}\int_{\Omega}(u-u_d)^2dx + \frac{\alpha}{2}\int_{\Omega}\lambda^2 dx \tag{11}$$

subject to a nonlinear Poisson's equation with Dirichlet boundary conditions:

$$\begin{aligned} -\Delta u + 10(u+u^3) &= \lambda \quad \text{in } \Omega, \\ u &= u_b \quad \text{on } \Gamma. \end{aligned} \tag{12}$$

Fig. 2 shows more details. Note that we employ an irregular domain with a dolphin-shaped hole in the middle. The irregular domain cannot be discretized using a structured mesh (a lattice). We record the wall-clock time in Table 1. As shown, AmorFEA based PDE-CO achieves similar optimized objective values compared with the adjoint method, but with significantly less computation time, excluding time to train the network.

Method	α	Optimized objective	Wall time [ms]	Iteration steps [#]
	10^{-6}	3.35×10^{-4}	700	13
Adjoint Method	10^{-3}	2.47×10^{-3}	600	11
	1	2.56×10^{-3}	337	6
	10^{-6}	3.40×10^{-4}	38	13
AmorFEA	10^{-3}	2.47×10^{-3}	37	11
	1	2.56×10^{-3}	16	7

Table 1: PDE-CO results for the adjoint method and AmorFEA. For different regularity coefficient α, AmorFEA achieves similar optimal objectives compared with the adjoint method, but with less wall time.

Inverse Kinematics of a Soft Robot We consider the control of a snake-like soft robot made of elastic material. Such robots represent promising solutions to minimally invasive surgery (Runciman et al., 2019). We control the static position of the robot by expanding or contracting "muscles" on the left and right sides of the robot, quantitatively described by an actuation field $\lambda(x)$ that controls the stretch ratio on the two sides. We focus on the middle point x_0 at the top of the robot and specify an

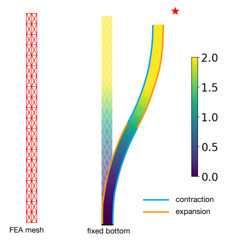

Figure 3: Setup of the problem. Left: the finite element mesh for the soft robot in an undeformed configuration. Right: with the actuation field $\lambda(x)$ set to be half-and-half for contraction and expansion, but upside down for the two sides, the robot can deform to a specific configuration. Colors indicate the displacement magnitude $\|u\|_2$.

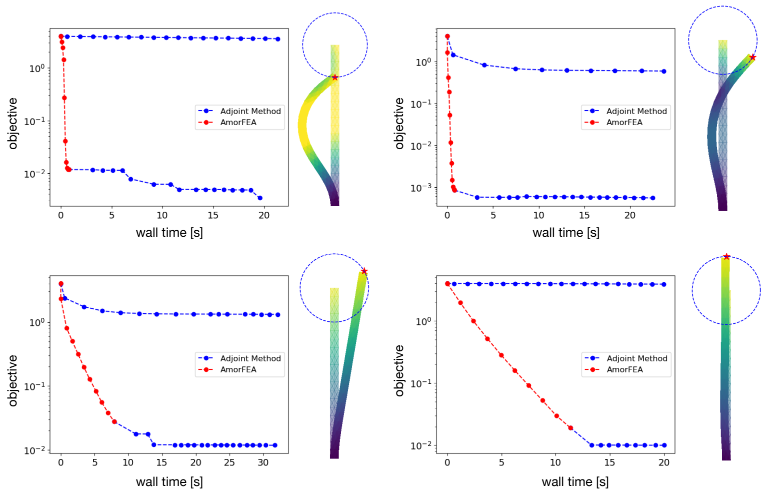

Figure 4: Wall time measurement for both the adjoint method and AmorFEA. There are four target displacements for the tip point displacement $u(x_0)$ to achieve. In these plots, one blue dot indicate one gradient descent step of the adjoint method, while one red dot indicate multiple gradient steps for AmorFEA.

arbitrary two-dimensional displacement u_0 that we hope this tip point can achieve by optimizing the actuation field $\lambda(x)$. Mathematically, the problem is to minimize the functional

$$\mathcal{J}(u,\lambda) = \|u(x_0) - u_0)\|_2^2 \tag{13}$$

subject to an equilibrium equation for hyperelastic material with appropriate boundary conditions:

$$\begin{aligned} \text{Div}\, P(u) &= 0 \quad \text{in } \Omega, \\ r(u,\lambda) &= 0 \quad \text{on } \Gamma, \end{aligned} \tag{14}$$

where "Div" is the divergence operator, P is the a second-order stress tensor and r is the boundary constraint. For more precise definitions of these terms, refer to Appendix D. Fig. 3 shows more details of the problem setup. We pick four equally-spaced target points on a circle centered at the robot tip and perform PDE-CO to find actuations. As shown in Fig. 4, we use gradient descent for both the adjoint method and AmorFEA. AmorFEA consumes significantly less time than the adjoint method.

4 Conclusions

In this work, we proposed a novel formulation (AmorFEA) that amortizes the solving process of classic finite element analysis. AmorFEA enables fast, differentiable prediction of PDE solutions, which accelerates PDE-constrained optimization. We quantitatively studied the amortization gap for both linear problems and nonlinear problems. Numerical experiments show that our method outperforms the traditional adjoint method on a per-iteration basis.

REFERENCES

Franco Brezzi and Michel Fortin. *Mixed and hybrid finite element methods*, volume 15. Springer Science & Business Media, 2012.

Yang Cao, Shengtai Li, Linda Petzold, and Radu Serban. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. *SIAM Journal on Scientific Computing*, 24(3):1076–1089, 2003.

Kristy Choi, Mike Wu, Noah Goodman, and Stefano Ermon. Meta-amortized variational inference and learning. *arXiv preprint arXiv:1902.01950*, 2019.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*, 2018.

CAR Sá De Melo, Mohit Randeria, and Jan R Engelbrecht. Crossover from BCS to Bose superconductivity: Transition temperature and time-dependent Ginzburg-Landau theory. *Physical Review Letters*, 71(19):3202, 1993.

Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

A Gerhard Holzapfel. Nonlinear solid mechanics ii. 2000.

Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pp. 971–980, 2017.

Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1): 112–147, 1998.

Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media, 2012.

Raymond W Ogden. *Non-linear elastic deformations*. Courier Corporation, 1997.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Sachin Ravi and Alex Beatson. Amortized Bayesian meta-learning. 2018.

Tyrone Rees, H Sue Dollar, and Andrew J Wathen. Optimal solvers for PDE-constrained optimization. *SIAM Journal on Scientific Computing*, 32(1):271–298, 2010.

Mark Runciman, Ara Darzi, and George P Mylonas. Soft robotics in minimally invasive surgery. *Soft robotics*, 6(4):423–443, 2019.

Michael Ulbrich and Bart van Bloemen Waanders. An introduction to partial differential equations constrained optimization. *Optimization and Engineering*, 19(3):515–520, 2018.

A Finite Element Analysis

The Finite Element Analysis (FEA) is arguably the most powerful approach known for the numerical solutions of problems characterized by partial differential equations (PDEs). We demonstrate the basic idea by considering the linear Poisson's equation as a model problem. As shown in Fig. 5, we start with the *strong formulation* of the problem and then introduce the *weak formulation* upon which the finite element approximation is built, namely, the *Galerkin weak formulation*. We further introduce the *minimization formulation* and the corresponding *Galerkin minimization formulation*, which is the cornerstone for the proposed amortized finite element analysis (AmorFEA).

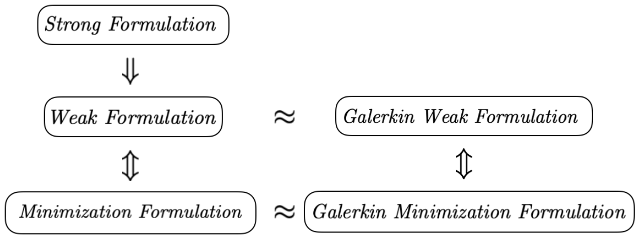

Figure 5: FEA roadmap.

The strong formulation of the Poisson's equation reads as

$$\begin{aligned} -\Delta u = \lambda \quad & \text{in } \Omega, \\ u = 0 \quad & \text{on } \Gamma. \end{aligned} \tag{15}$$

For simplicity, we have assumed homogenous Dirichlet boundary conditions. Instead of heuristically approximating the differential operator like the Finite Difference Method (FDM), FEA employs a more systematic approach by searching the best possible solution over a constructed finite-dimensional function space.

First, let us multiply a test function v for both sides of the PDE in Eq. 15, integrate over Ω, and use integration by parts to obtain the weak formulation: find $u \in \mathbb{V} := \{v \text{ sufficiently smooth} \mid v|_\Gamma = 0\}$ such that

$$a(u, v) = l(\lambda, v), \text{ for all } v \in \mathbb{V}, \tag{16}$$

where

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, dx, \quad l(\lambda, v) = \int_\Omega \lambda v \, dx. \tag{17}$$

Second, we construct a finite-dimensional subspace $\mathbb{V}_h \subset \mathbb{V}$ and $\mathbb{V}_h = \text{span}\{\phi_1, ..., \phi_n\}$ is the piece-wise polynomial function space. Note that for any function $v \in \mathbb{V}_h$, there is a unique representation $v = \sum_{i=1}^n v_i \phi_i$. We thus can define an isomorphism $\mathbb{V}_h \cong \mathbb{R}^n$ by $v = \sum_{i=1}^n v_i \phi_i \leftrightarrow \boldsymbol{v} = (v_1, ..., v_n)^\top$.

The Galerkin weak formulation yields an approximation to Eq. 16 by solving the following system of linear equations:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}, \tag{18}$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the stiffness matrix with $A_{ij} = a(\phi_i, \phi_j)$, $\boldsymbol{u} \in \mathbb{R}^n$ is the solution vector and $\boldsymbol{f} \in \mathbb{R}^n$ is the source vector with $f_i = l(\phi_i, \lambda)$. As a remark, if λ is also represented by piece-wise polynomial basis functions, we could further write $\boldsymbol{f} = \boldsymbol{B}\boldsymbol{\lambda}$ with $\boldsymbol{B} \in \mathbb{R}^{n \times m}$ and $B_{ij} = l(\phi_i, \phi_j)$. It is not difficult to verify that $\boldsymbol{A}$ is symmetric and positive definite. By solving Eq. 18, the FEM gives the best possible solution we can find in $\mathbb{V}_h$.

The minimization formulation states that

$$\min_{u \in \mathbb{V}} \mathcal{L}(u), \tag{19}$$

where

$$\mathcal{L}(u) = \frac{1}{2} \int_\Omega \nabla u \cdot \nabla u \, dx - \int_\Omega \lambda u \, dx. \tag{20}$$

By setting the functional derivative to be zero, we recover the weak formulation.

Finally, we introduce the Galerkin minimization formulation, which is the discretized version of the minimization formulation above. Replace $u \in \mathbb{V}$ in Eq. 19 with $u = \sum_{i=1}^{n} u_i \phi_i \in \mathbb{V}_h$, we get the following finite-dimensional optimization problem:

$$\min_{\boldsymbol{u} \in \mathbb{R}^n} \mathcal{L}(\boldsymbol{u}), \tag{21}$$

where

$$\mathcal{L}(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{u} - \boldsymbol{f}^\top \boldsymbol{u}. \tag{22}$$

The quadratic programming problem yields the same solution as Eq. 18. AmorFEA is based on the Galerkin minimization formulation. Note that for a nonlinear problem, we may have a more complicated nonlinear optimization problem rather than the quadratic programming.

For further details regarding FEA and its variational formulation, we refer readers to Brezzi & Fortin (2012).

B The Adjoint Method

Consider the same PDE-CO setup as introduced in Eq. 6. Taking the derivative of the constraint equations in Eq. 6 with respect to $\boldsymbol{\lambda}$ yields chains of Jacobian matrices:

$$\frac{d\boldsymbol{c}}{d\boldsymbol{\lambda}} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}}\frac{d\boldsymbol{u}}{d\boldsymbol{\lambda}} + \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}} = 0. \tag{23}$$

Hence,

$$\frac{d\boldsymbol{u}}{d\boldsymbol{\lambda}} = -\Big(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}}\Big)^{-1}\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}}. \tag{24}$$

Substitute Eq. 24 to Eq. 8, we obtain

$$\frac{d\widehat{\mathcal{J}}}{d\boldsymbol{\lambda}} = -\overbrace{\frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\Big(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}}\Big)^{-1}}^{\text{adjoint PDE}}\!\!\!\!\!\!\!\!\underbrace{\phantom{\Big(\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{u}}\Big)^{-1}}\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{\lambda}}}_{\text{tangent linear PDE}} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{\lambda}}. \tag{25}$$

Resembling the two different modes of automatic differentiation, we could choose to either solve the adjoint PDE first (reverse-mode) or the tangent linear PDE first (forward-mode). When the size of the control vector is larger than that of the objective (e.g., $m \gg 1$ in our case), it is more efficient to solve the adjoint PDE first, giving the name adjoint method (Cao et al., 2003).

C Linear Examples

An important and general class of PDE-CO problems impose a linear relationship between the control parameter and the state variable. These linear problems provide a useful testbed for examining the AmorFEA approach. We focus on the model problem in Appendix A, where the source vector $\boldsymbol{\lambda}$ is linearly related to the solution vector $\boldsymbol{u}$. We show that by employing a linear regression model and performing AmorFEA, we are able to fully recover the underlying physics and achieve an amortization gap of zero; this result is unsurprising due to the assumption of a linear relationship between $\boldsymbol{u}$ and $\boldsymbol{\lambda}$. Though simple to analyze, the linear case gives intuition about the proposed scheme. We also compare AmorFEA with supervised learning, where we run FEA simulations to obtain labeled data and train the linear model in a traditional fashion. FEA simulations are carried out using an open source Python package `FEniCS` (Logg et al., 2012). Neural network training is performed in `PyTorch` (Paszke et al., 2019). We show that both AmorFEA and supervised training have the same global optimality condition.

We use AmorFEA to train a single-layer network that predicts $\boldsymbol{u} \in \mathbb{R}^n$ from $\boldsymbol{\lambda} \in \mathbb{R}^m$, where $m = 811$ and $n = 721$ in this case. We assume the distribution over source terms $\boldsymbol{\lambda}$ is a uniform distribution on the hypercube $[-1,1]^m$. $10,000$ samples were drawn from this distribution to form the training data. FEA converts the minimization problem in Eq. 1 into a linear system to solve:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{B}\boldsymbol{\lambda}, \tag{26}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ arise from the assembly procedure of the weak form in FEA (see details in Appendix A). Let us denote $\boldsymbol{f} = \boldsymbol{B}\boldsymbol{\lambda}$. The linear model we employ has a weight matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ and no bias vector so that $\widehat{\boldsymbol{u}} = \boldsymbol{W}\boldsymbol{f}$. The potential energy in this case is therefore equivalent to the least-squares loss for the solution to the linear system, preconditioned by the stiffness matrix $\boldsymbol{A}$:

$$\mathcal{L}_a(\boldsymbol{W};\boldsymbol{f}) = \frac{1}{2}(\boldsymbol{W}\boldsymbol{f})^\top \boldsymbol{A}(\boldsymbol{W}\boldsymbol{f}) - \boldsymbol{f}^\top(\boldsymbol{W}\boldsymbol{f}). \tag{27}$$

As an instantiation of Eq. 2 using the Monte Carlo estimate of the expectation, AmorFEA leads to the following optimization problem:

$$\min_{\boldsymbol{W}\in\mathbb{R}^{n\times n}} \overline{\mathcal{L}}_a(\boldsymbol{W}), \tag{28}$$

where

$$\overline{\mathcal{L}}_a(\boldsymbol{W}) = \frac{1}{K}\sum_{k=1}^{K}\Big(\frac{1}{2}(\boldsymbol{W}\boldsymbol{f}_k)^\top \boldsymbol{A}(\boldsymbol{W}\boldsymbol{f}_k) - \boldsymbol{f}_k^\top(\boldsymbol{W}\boldsymbol{f}_k)\Big).$$

Next, we study supervised learning in the same environment. We expand the empirical data set $\mathcal{D} = \{\boldsymbol{\lambda}^{(i)}\}$ to $\mathcal{D}' = \{(\boldsymbol{\lambda},\boldsymbol{u})^{(i)}\}$ by running FEA simulations. The supervised loss function for each data point is defined as

$$\mathcal{L}_s(\boldsymbol{W};\boldsymbol{f}) = \frac{1}{2}\|\widehat{\boldsymbol{u}} - \boldsymbol{u}\|_2^2 = \frac{1}{2}\|\boldsymbol{W}\boldsymbol{f} - \boldsymbol{A}^{-1}\boldsymbol{f}\|_2^2. \tag{29}$$

Supervised training solves the following minimization problem:

$$\min_{\boldsymbol{W}\in\mathbb{R}^{n\times n}} \overline{\mathcal{L}}_s(\boldsymbol{W}), \tag{30}$$

where

$$\overline{\mathcal{L}}_s(\boldsymbol{W}) = \frac{1}{K}\sum_{k=1}^{K}\Big(\frac{1}{2}(\boldsymbol{W}\boldsymbol{f}_k - \boldsymbol{A}^{-1}\boldsymbol{f}_k)^\top(\boldsymbol{W}\boldsymbol{f}_k - \boldsymbol{A}^{-1}\boldsymbol{f}_k)\Big).$$

We show next that both $\overline{\mathcal{L}}_a(\boldsymbol{W})$ and $\overline{\mathcal{L}}_s(\boldsymbol{W})$ are convex. The following lemma makes the proof more concise.

Lemma 1. *A function $f:\mathbb{R}^n \to \mathbb{R}$ is convex if and only if the function $g:\mathbb{R}\to\mathbb{R}$ given by $g(t) = f(\boldsymbol{x} + t\boldsymbol{y})$ is convex (as a univariate function) for all $\boldsymbol{x}$ in domain of f and all $\boldsymbol{y}\in\mathbb{R}^n$. (The domain of g here is all t for which $\boldsymbol{x}+t\boldsymbol{y}$ is in the domain of f.)*

We first show for AmorFEA training:

Proposition 1. *The empirical loss function $\overline{\mathcal{L}}_a(\boldsymbol{W})$ of AmorFEA in the linear model is convex.*

Proof. We use Lemma 1 to show the function $\overline{\mathcal{L}}_a : \mathbb{R}^{n\times n} \to \mathbb{R}$ is convex. For any fixed $\boldsymbol{W}_1 \in \mathbb{R}^{n\times n}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{n\times n}$, let

$$\begin{aligned} g(t) =& \overline{\mathcal{L}}_a(\boldsymbol{W}_1 + t\boldsymbol{W}_2) \\ =& \frac{1}{K}\sum_{k=1}^{K}\Big(\frac{1}{2}\big((\boldsymbol{W}_1+t\boldsymbol{W}_2)\boldsymbol{f}_k\big)^\top \boldsymbol{A}\big((\boldsymbol{W}_1+t\boldsymbol{W}_2)\boldsymbol{f}_k\big) - \boldsymbol{f}_k^\top\big((\boldsymbol{W}_1+t\boldsymbol{W}_2)\boldsymbol{f}_k\big)\Big) \\ =& \Big(\frac{1}{2K}\sum_{k=1}^{K}\boldsymbol{f}_k^\top\boldsymbol{W}_2^\top\boldsymbol{A}\boldsymbol{W}_2\boldsymbol{f}_k\Big)t^2 + \Big(\frac{1}{2K}\sum_{k=1}^{K}\big(\boldsymbol{f}_k^\top\boldsymbol{W}_2^\top\boldsymbol{A}\boldsymbol{W}_1\boldsymbol{f}_k + \boldsymbol{f}_k^\top\boldsymbol{W}_1^\top\boldsymbol{A}\boldsymbol{W}_2\boldsymbol{f}_k - 2\boldsymbol{f}_k^\top\boldsymbol{W}_2\boldsymbol{f}_k\big)\Big)t \\ &- \Big(\frac{1}{2K}\sum_{k=1}^{K}\boldsymbol{f}_k^\top\boldsymbol{W}_1\boldsymbol{f}_k\Big)\Big) \end{aligned} \tag{31}$$

Since $\boldsymbol{A}$ is symmetric and positive definite, the coefficient of the quadratic term is always positive:

$$\frac{1}{2K}\sum_{k=1}^{K}\boldsymbol{f}_k^\top \boldsymbol{W}_2^\top \boldsymbol{A}\boldsymbol{W}_2\boldsymbol{f}_k = \frac{1}{2K}\sum_{k=1}^{K}\|\boldsymbol{A}^{\frac{1}{2}}\boldsymbol{W}_2\boldsymbol{f}_k\|_2^2 \geq 0. \tag{32}$$

This shows that $g(t)$ is always convex. □

We then show for supervising training:

Proposition 2. *The empirical loss function $\overline{\mathcal{L}}_s(\boldsymbol{W})$ of supervised learning in the linear model is convex.*

Proof. We use Lemma 1 to show the function $\overline{\mathcal{L}}_s : \mathbb{R}^{n\times n} \to \mathbb{R}$ is convex. For any fixed $\boldsymbol{W}_1 \in \mathbb{R}^{n\times n}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{n\times n}$, let

$$\begin{aligned} g(t) =& \overline{\mathcal{L}}_s(\boldsymbol{W}_1 + t\boldsymbol{W}_2)\\ =& \frac{1}{K}\sum_{k=1}^{K}\Big(\frac{1}{2}\big((\boldsymbol{W}_1 + t\boldsymbol{W}_2)\boldsymbol{f}_k - \boldsymbol{A}^{-1}\boldsymbol{f}_k\big)^\top\big((\boldsymbol{W}_1 + t\boldsymbol{W}_2)\boldsymbol{f}_k - \boldsymbol{A}^{-1}\boldsymbol{f}_k\big)\Big)\\ =& \Big(\frac{1}{2K}\sum_{k=1}^{K}\boldsymbol{f}_k^\top\boldsymbol{W}_2^\top\boldsymbol{W}_2\boldsymbol{f}_k\Big)t^2 - \Big(\frac{1}{2K}\sum_{k=1}^{K}\big(\boldsymbol{f}_k^\top(\boldsymbol{W}_1 - \boldsymbol{A}^{-1})^\top\boldsymbol{W}_2\boldsymbol{f}_k + \boldsymbol{f}_k^\top\boldsymbol{W}_2^\top(\boldsymbol{W}_1 - \boldsymbol{A}^{-1})\boldsymbol{f}_k\big)\Big)t\\ &+ \Big(\frac{1}{2K}\sum_{k=1}^{K}\boldsymbol{f}_k^\top(\boldsymbol{W}_1 - \boldsymbol{A}^{-1})^\top(\boldsymbol{W}_1 - \boldsymbol{A}^{-1})\boldsymbol{f}_k\big)\Big) \end{aligned} \tag{33}$$

The coefficient of the quadratic term is always positive:

$$\frac{1}{2K}\sum_{k=1}^{K}\boldsymbol{f}_k^\top \boldsymbol{W}_2^\top \boldsymbol{W}_2\boldsymbol{f}_k = \frac{1}{2K}\sum_{k=1}^{K}\|\boldsymbol{W}_2\boldsymbol{f}_k\|_2^2 \geq 0. \tag{34}$$

This shows that $g(t)$ is always convex. □

We also see that the first order condition (FOC) for AmorFEA gives

$$\frac{\partial\overline{\mathcal{L}}_a}{\partial\boldsymbol{W}} = (\boldsymbol{A}\boldsymbol{W} - \boldsymbol{I})\Big(\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{f}_k\boldsymbol{f}_k^\top\Big) = \boldsymbol{0}. \tag{35}$$

Similarly, the FOC for supervised training gives

$$\frac{\partial\overline{\mathcal{L}}_s}{\partial\boldsymbol{W}} = (\boldsymbol{W} - \boldsymbol{A}^{-1})\Big(\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{f}_k\boldsymbol{f}_k^\top\Big) = \boldsymbol{0}. \tag{36}$$

By Proposition 1 and Proposition 2 along with Eq. 35 and Eq. 36, we conclude that both AmorFEA and supervised learning achieve the global minimum with same condition $\boldsymbol{W} = \boldsymbol{A}^{-1}$. The only difference between these problems is the "physical" preconditioning from $\boldsymbol{A}$. As a supporting experiment, we show in Fig. 6 that both AmorFEA and supervised training have similar capabilities to recover the underlying model. Since the fully trained linear model is able to produce $\widehat{\boldsymbol{u}} = \boldsymbol{u}$, we have a zero amortization gap as computed by Eq. 4.

D NONLINEAR EXAMPLES

We examine two more realistic nonlinear settings and perform PDE-CO in this section. Here the solution vector $\boldsymbol{u}$ and the control vector $\boldsymbol{\lambda}$ have a nonlinear relationship, which naturally motivates the use a neural network as the predictive mapping.

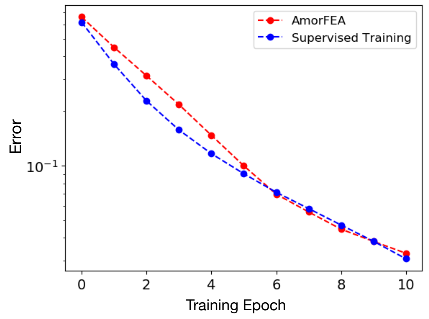

Figure 6: Normalized error versus training epochs. We define the normalized error $\epsilon = \frac{\|\boldsymbol{W}-\boldsymbol{A}^{-1}\|_\infty}{\|\boldsymbol{A}^{-1}\|_\infty}$.

Source Field Finding The potential energy is

$$\mathcal{L}(u,\lambda) = \frac{1}{2}\int_\Omega \|\nabla u\|_2^2 dx + \frac{1}{2}\int_\Omega (10u^2 + 5u^4)dx - \int_\Omega \lambda u dx. \tag{37}$$

The governing PDE (Eq. 12) can be derived by minimizing Eq. 37.

We set $\lambda(x) = 100\text{exp}(\frac{\|x-(0.1,0.1)\|_2^2}{0.02})$, solve the governing PDE by FEA and set the solution to be the desired field $u_d(x)$ for Eq. 11.

We use AmorFEA to train a neural network that predicts $\boldsymbol{u} \in \mathbb{R}^n$ from $\boldsymbol{\lambda} \in \mathbb{R}^m$, where $m = n = 759$ in this case. The assumed distribution over $\boldsymbol{\lambda}$ is constructed from a zero-mean Gaussian process given by

$$\begin{aligned} f(x) &\sim \mathcal{GP}\big(\mu(\cdot), k(\cdot,\cdot)\big) \\ \mu(x) &= 0 \\ k(x^{(i)}, x^{(j)}) &= \sigma^2 \text{exp}\Big(-\frac{\|x^{(i)} - x^{(j)}\|_2^2}{l^2}\Big), \end{aligned} \tag{38}$$

where we set the output variance $\sigma = 10^2$ and the lengthscale $l = 0.1$. To construct the training and testing data from this distribution, $30,000$ source terms were generated. Compared with supervised data generated by expensive FEA simulations, our data are almost free to obtain.

For the neural network model, we use a MLP with "scaled exponential linear units" (SELUs) for the activation functions (Klambauer et al., 2017). We perform a $90/10$ train-test split for our data. For the 10% test data, we run FEA simulation so that we can report test error by comparing the AmorFEA solutions with the FEA solutions. Since our data represent functions, it is more meaningful to define an error quantification metric for functions than to use simple mean squared error (MSE). We define the relative error ϵ using norms on the $\mathcal{L}^2(\Omega)$ space:

$$\epsilon = \mathbb{E}_{p(\boldsymbol{\lambda})}\Big[\frac{\|u^\star - u^a\|_{\mathcal{L}^2}}{\|u^\star\|_{\mathcal{L}^2}}\Big], \tag{39}$$

where $u^\star$ is the FEA solution and u^a is the AmorFEA solution. With FEA simulations available in the test set, we can also compute the amortization gap introduced in Eq. 4. Since analytical solutions are generally not possible, we cannot explicitly evaluate the approximation gap or the inference gap. However, the approximation gap is a well-studied subject in the literature on finite element analysis (Hughes, 2012).

We run experiments using MLPs with different layers and report the amortization gap and relative error in Table 2. The layers in all MLPs have equal widths. As shown in the table, a single-layer network cannot perfectly solve this nonlinear problem, in contrast to the linear problem in Section C. We observe that both the amortization gap and the relative error decrease with deeper neural networks. Using bigger neural networks may give us smaller amortization gaps, but there is a trade-off between training time and performance. We show next that a MLP with 2 hidden layers provides adequate accuracy for the PDE-CO problem while being trained cheaply.

	MLP-0	MLP-1	MLP-2
Δ_{am}	1.58×10^{-1}	1.47×10^{-2}	4.62×10^{-3}
ϵ	4.60×10^{-2}	9.44×10^{-3}	7.50×10^{-3}

Table 2: Test performance. The amortization gap Δ_{am} is computed according Eq. 4. The relative error ϵ is computed as Eq. 39. The number suffix to "MLP" refers to the number of hidden layers.

Inverse Kinematics of a Soft Robot We model the soft robot with a neo-Hookean hyperelastic solid (Ogden, 1997). The total potential energy is

$$\mathcal{L}(u, \lambda) = \int_{\Omega} W dx, \tag{40}$$

where the energy density W is defined for material bulk and shear moduli μ and κ as:

$$W = \frac{\mu}{2}\Big((\det F)^{-2/3}\text{tr}(FF^T) - 3\Big) + \frac{\kappa}{2}(\det F - 1)^2, \tag{41}$$

and F is the deformation gradient,

$$F = \nabla u + I. \tag{42}$$

Since our problem is in 2d, we have assumed plain strain condition. The governing PDE (Eq. 14) can be derived by minimization Eq. 40. The stress tensor P in Eq. 14 is known as the first Piola-Kirchoff stress and can be obtained by

$$P = \frac{\partial W}{\partial F}. \tag{43}$$

The boundary constraint $r(u, \lambda)$ in Eq. 14 is expressed over the bottom side boundary Γ_b, top side boundary Γ_t and the side boundaries Γ_s respectively as

$$u = 0 \quad \text{on } \Gamma_b, \tag{44}$$

$$P \cdot N = 0 \quad \text{on } \Gamma_t, \tag{45}$$

$$\Lambda = h(\lambda) \quad \text{on } \Gamma_s, \tag{46}$$

where N is the normal vector to the boundary, Λ is the stretch ratio (see definition in (Holzapfel, 2000)) and $h(\lambda) = \frac{1}{1+e^{-\lambda}} + \frac{1}{2}$ forces the range of Λ to $(0.5, 1.5)$ in order to avoid extreme and unrealistic deformations.

We use AmorFEA to train a neural network to predict $\boldsymbol{u} \in \mathbb{R}^n$ from $\boldsymbol{\lambda} \in \mathbb{R}^m$, where $m = 40$ and $n = 206$ in this case. The assumed distribution over actuation fields is constructed from a uniform distribution over the hypercube $[-a, a]^m$ with $a > 0$. $30,000$ training data were generated from this distribution. The potential energy $\mathcal{L}(u, \lambda)$ in this case is sensitive to the displacement u. For a displacement field that causes any overlap of the deformed robot, the energy will be infinite. We take a warm-start strategy to train the network by varying a from 0.1, 0.2, 0.4 until 0.8, and train the network successively. Training took $5 - 10$ minutes.

We additionally demonstrate that AmorFEA follows a similar optimization path to the adjoint method, by showing the objective function versus the number of gradient descent steps in Fig. 7 for each of the four cases. Wall time measurements in [s] of these four cases are $(119, 218, 133, 113)$ for the adjoint method and $(0.215, 0.210, 0.226, 0.249)$ for AmorFEA. For completeness, we also explored gradient-free techniques such as the Nelder–Mead method (Lagarias et al., 1998), but they typically converged to an unsatisfactory local minimum.

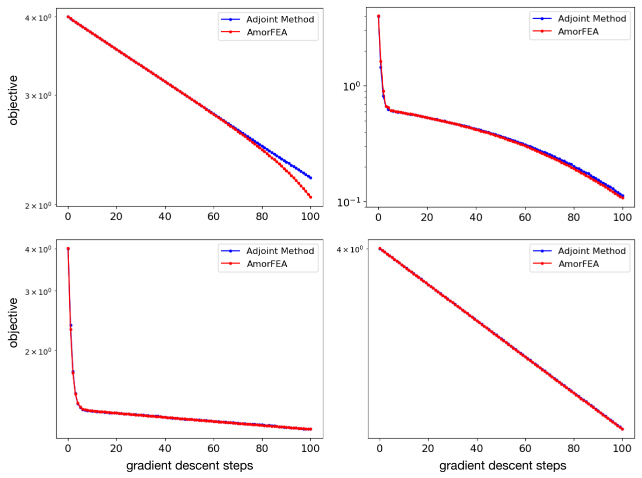

Figure 7: Objective function versus gradient descent steps for both AmorFEA and the adjoint method. The four cases correspond to the same scenarios in Fig. 4.