Computer Methods in Applied Mechanics and Engineering 451 (2026) 118663

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Differentiable lagrangian shock hydrodynamics with application to stable shock acceleration of density interfaces

Kevin Korner[a], Brandon Talamini[a], Julian Andrej[a], Michael Tupek[a], William Moses[b], Daniel Tortorelli[a], Robert Rieben[a], Tzanio Kolev[a], Jamie Bramwell[a], Daniel White[a], Jonathan Belof[a], William Schill[a]

[a] Lawrence Livermore National Laboratory, 7000 East Ave, Livermore, 94550, California, USA
[b] University of Illinois Urbana-Champaign, 506 S. Wright St., Urbana, 61801, Illinois, USA

ARTICLE INFO

Keywords:
Topology optimization
Hydrophysics
Richtmyer-Meshkov instability
Shock shaping
Interfaces

ABSTRACT

We develop a gradient based optimization approach for the equations of compressible, Lagrangian hydrodynamics and demonstrate how it can be employed to automatically uncover strategies to control hydrodynamic instabilities arising from shock acceleration of density interfaces. Strategies for controlling the Richtmyer-Meshkov instability (RMI) are of great benefit for inertial confinement fusion (ICF) where shock interactions with many small imperfections in the density interface lead to instabilities which rapidly grow over time. These instabilities lead to mixing which, in the case of laser driven ICF, quenches the runaway fusion process ruining the potential for positive energy return. We demonstrate that control of these instabilities can be achieved by optimization of initial conditions with (> 100) parameters. Optimizing over a large parameter space like this is not possible with gradient-free optimization strategies.

This requires computation of the gradient of the outputs of a numerical solution to the equations of Lagrangian hydrodynamics with respect to the inputs.

We show that the efficient computation of these gradients is made possible via a judicious application of (i) adjoint methods, the exact formal representation of sensitivities involving partial differential equations, and (ii) automatic differentiation (AD), the algorithmic calculation of derivatives of functions.

Careful regularization of multiple operators including artificial viscosity and timestep control is required.

We perform design optimization of > 100 parameter energy field driving the Richtmyer Meshkov instability showing significant suppression while simultaneously enhancing the acceleration of the interface relative to a nominal baseline case.

1. Introduction

Shock hydrodynamics describe the behavior of some of the most complex phenomena in science and technology. Examples abound including supernovae in astrophysics and inertial or magnetic confinement fusion in the laboratory. The physical processes that may occur are manifold; however, much of the complexity may be thought of as arising from dynamical instabilities. By way of example, the Richtmyer-Meshkov instability (RMI) occurs when a shock wave – a discontinuous jump in the thermodynamic state of the material – impinges on an interface between two materials of differing densities [1,2]. The resulting dynamics are famously unconditionally unstable resulting in *jetting* which rips up the interface and mixes the materials together. For a comprehensive review of RMI, see [3,4] and the references therein. RMI has held a critical role in both scientific and technological applications including astrophysics [5], mining [6], many applications of fluid transport [7] including scramjets [5], and laser driven inertial confinement

https://doi.org/10.1016/j.cma.2025.118663
Received 23 September 2025; Received in revised form 19 November 2025; Accepted 8 December 2025
Available online 31 December 2025

fusion (ICF) [8,9] such as is pursued at the National Ignition Facility (NIF). Control over RMI induced jetting is thus a grand challenge.

Significant recent effort has gone into optimization involving hydrodynamic control over jetting including gas gun driven experiments [10], and explosively driven linear shaped charges [11,12]. These strategies typically proceed by identifying a *small* number of parameters describing a physical system of interest, performing a large suite of calculations, fitting a machine-learned model to predict certain interesting quantities from a simulation, and then using the resulting model to optimize or otherwise interrogate the behavior of the system [13,14]. It is even possible to develop an analytic solution [15] to a reduced and simplified version of this problem. A closely related class of problems is the inference of material parameters from complex extreme pressure materials science experiments such as those conducted on pulsed power [16,17] or laser [18] platforms. These techniques are powerful but they are currently limited to a small number of design variables; however, the general case – i.e. control over an arbitrary interface with potentially > 100 degrees of freedom remains unsolved. The fundamental reason for this has to do with basic characteristics of optimization theory; optimization is very expensive for large numbers of variables in the absence of gradient information and the computation of the objective function gradients in these problems rely on the *derivative of the hydrocode outputs with respect to their inputs*. In general optimization problems where the forward solve requires significant computation tends to be limited to O(10) optimization parameters while the inclusion of gradients can readily handle far greater than O(100) degrees of freedom much more efficiently (see [19] and the references therein). A brief demonstration of the scaling benefit of gradients can be found in Section (A.1). Computing this gradient in the context of Lagrangian Shock Hydrodynamics is a challenging task and is one of the key contributions of this paper.

The subject of gradient based optimization involving partial differential equations has seen extensive study. For instance, topology optimization is a field of engineering and mathematics which deals with optimizing the material layout within a given design space to achieve the best performance under specified conditions and has been applied to studying many different problems [20–22]. Basic adjoint calculations can be found in a variety of texts (see, for instance, Akerson [21], Plessix [23], Zhang and Sandu [24], Zhang et al. [25,26]). This has seen significant recent development by some of the authors [27] for nonlinear problems in solid mechanics. Recently, a number of authors have begun applying these techniques to challenging problems in high energy physics [28].

In this article, we present a computational approach that efficiently computes the derivatives of predicted outputs from a high-order finite element Lagrangian hydrodynamics code with respect to its inputs, using a combination of adjoint theory and automatic differentiation. We apply this method to a complex interface stability problem, which requires differentiating the time-stepping update, the (partially) assembled force vector, and the zero-dimensional physics at the quadrature points.

The remainder of this article will be organized as follows: (i) gradient based optimization of time dependent problems (Section 2), (ii) discretization by finite elements of Lagrangian shock hydrodynamics (Section 3), and (iii) the application of this technique to the suppression of RMI suggesting a tantalizing pathway to stable shock acceleration of density interfaces (Section 4).

2. Gradient based optimization of time dependent problems

In this section, we introduce several concepts that are necessary for our ultimate goal of computing gradients of a Lagrangian hydrodynamics discretization. First, we discuss the two main strategies used to develop adjoints of time dependent problems: differentiate then discretize and discretize then differentiate. We argue for the latter case; however, we include the fundamental framwork for both as they provide valuable insight into the structure of the problem. Next we discuss methods for verifying the computation of gradients by checking the error order of convergence. We then introduce automatic differentiation (AD), quantities of interest (QoIs) and check-pointing for time dependent problems. Finally, we consider a simple example involving multi-particle systems to illustrate the combination of these components.

2.1. Differentiate then discretize

One method of conducting an adjoint calculation is to ***differentiate then discretize*** [23]. This approach is also sometimes refered to as the *infdim* approach because all continuous fields are varied on their respective function spaces. We find a system of equations which can be solved to find the derivative, then we discretize the system in time in order to do calculations. We have a design field β and a response y that are linked through an initial boundary value problem IBVP such that

$$\dot{y}(t,\beta) = f(y(t,\beta),\beta,t), \tag{1}$$

$$y(0,\beta) = y_0\,. \tag{2}$$

As seen above, y is both an implicit and explicit function of β, with the exception of the initial condition y_0, which may be an explicit function of β. We refer to (1) as the *primal analysis* and y as the *primal response*. Upon solving the IBVP, we evaluate a Quantity of Interest (QoI) O that might be the cost or constraint function in an optimization, or the error function in an inverse or identification analysis. We define

$$O(\beta) = \int_0^T o(y(t,\beta),\beta)dt + \hat{o}(y(T,\beta),\beta)\,, \tag{3}$$

where the integrand o, $\hat{o}$ is a differentiable function of y and β. In this notation, o is a running objective while $\hat{o}$ is a terminal objective. The solution of the primal problem for y and integration of O generally proceed in tandem. We call this the forward pass. To solve the optimization/inverse/identification problem we use nonlinear programming algorithms to determine the β that minimizes the cost/error function. These iterative algorithms require gradients/variations of the QoIs to efficiently update β and to verify optimality. The variation of our QoI becomes

$$\delta O(\beta,\delta\beta) = \int_0^T \left(\frac{\partial o}{\partial y}(y(t,\beta),\beta)\cdot \delta y(t,\beta,\delta\beta) + \frac{\partial o}{\partial \beta}(y(t,\beta),\beta)\cdot\delta\beta \right) dt + \frac{\partial \hat{o}}{\partial y}\cdot \delta y(T,\beta,\delta\beta) + \frac{\partial \hat{o}}{\partial \beta}\cdot\delta\beta\,. \tag{4}$$

We use the adjoint method to eliminate the implicit response variation $\delta y(\beta,\delta\beta)$ in the above expression. In this method, we augment (3) such that

$$O(\beta) = \int_0^T o(y(t,\beta),\beta)dt + \hat{o}(y(T,\beta),\beta) + \int_0^T \lambda(t)\cdot(\dot{y}(t,\beta) - f(y(t,\beta),\beta,t))dt \tag{5}$$

The *adjoint response* λ, which is presently an arbitrary field, is used to convert the vector valued dynamics from Eq. (1) into a scalar and integrate it into the objective. Because the solution y satisfies the dynamics, the augmented term equals zero. We integrate the product of Eq. (1) and λ over time to leverage the equality (1) at all time $t \in [0,T]$. Taking the variation of the above, integrating by parts, rearranging and dropping the arguments for conciseness gives

$$\delta O = \int_0^T \left(\frac{\partial o}{\partial y}\cdot\delta y + \frac{\partial o}{\partial \beta} \right) dt + \frac{\partial \hat{o}}{\partial y}\cdot\delta y(T) + \frac{\partial \hat{o}}{\partial \beta}\cdot\delta\beta + \tag{6}$$

$$\int_0^T \lambda\cdot\left(\delta\dot{y} - \frac{\partial f}{\partial y}\cdot\delta y - \frac{\partial f}{\partial \beta}\cdot\delta\beta\right)dt \tag{7}$$

$$= \int_0^T \left(\frac{\partial o}{\partial y}\cdot\delta y + \frac{\partial o}{\partial \beta} \right) dt + \frac{\partial \hat{o}}{\partial y}\cdot\delta y(T) + \frac{\partial \hat{o}}{\partial \beta}\cdot\delta\beta + \tag{8}$$

$$\int_0^T \left[\delta y\cdot\left(-\dot{\lambda} - \left(\frac{\partial f}{\partial y}\right)^T\cdot\lambda\right) - \lambda\cdot\frac{\partial f}{\partial \beta}\cdot\delta\beta \right] dt + \delta y\cdot\lambda|_0^T \tag{9}$$

$$= \int_0^T \left(\frac{\partial o}{\partial \beta}\cdot\delta\beta - \lambda\cdot\frac{\partial f}{\partial \beta}\cdot\delta\beta \right) dt - \delta y(0)\cdot\lambda(0) + \frac{\partial \hat{o}}{\partial \beta}\cdot\delta\beta + \tag{10}$$

$$\int_0^T \delta y\cdot\left(\frac{\partial o}{\partial y} - \dot{\lambda} - \left(\frac{\partial f}{\partial y}\right)^T\cdot\lambda\right)dt + \delta y(T)\cdot\left(\frac{\partial \hat{o}}{\partial y}(y(T,\beta),\beta) + \lambda(T)\right) \tag{11}$$

where we note that $\delta y(0)$ equals the explicity known variation $\delta y_0(\beta,\delta\beta)$, i.e. β can describe y_0 as well as the function f. To eliminate the implicitly defined variation δy and $\delta y(T)$ we now require heretofore arbitrary adjoint variable λ to solve the *adjoint* terminal value boundary value problem

$$\dot{\lambda} = \frac{\partial o}{\partial y} - \lambda\cdot\frac{\partial f}{\partial y} \tag{12}$$

$$\lambda(T) = -\frac{\partial \hat{o}}{\partial y}(y(T,\beta),\beta)\,. \tag{13}$$

In this way, the sensitivity (6) reduces to

$$\delta O = \int_0^T \left(\frac{\partial o}{\partial \beta}\cdot\delta\beta - \lambda\cdot\frac{\partial f}{\partial \beta}\cdot\delta\beta \right) dt - \delta y_0\cdot\lambda(0) + \frac{\partial \hat{o}}{\partial \beta}\cdot\delta\beta\,. \tag{14}$$

The solution of the adjoint *terminal* value problem for λ and *backward* integration of δO generally proceed in tandom. We call this the *reverse pass*.

Due to the fact that no discretization was necessary in order to describe both the forward and reverse problems mathematically, it presents itself nicely for theoretical work in the subject. This makes it ideal for studying properties of solutions using analytical methods. Unfortunately, the vast majority of problems are not analytically integrable in time, so, in order to represent the solution, we must choose time integrators for both the forward and reverse passes as well as any integrators we use for calculating QoIs.

2.2. Discretize then differentiate

In this work, we argue its best to ***discretize then differentiate*** for the case of nonlinear partial differential equations representing temporal evolution of conserved physical quantities. We cannot generally solve (1) analytically so we resort to some discretization scheme, e.g. Crank-Nicolson, Runge-Kutta, etc. so that (1) becomes

$$y_k = f_k(y_{k-1},\beta,\Delta t), \tag{15}$$

$$y_0 = y_0\,, \tag{16}$$

where the subscripts k and $k-1$ refer to the times that the quantities are evaluated, e.g. $y_k = y(t_k)$ contains the degrees-of-freedom that are used to approximate $y(t_k)$, Δt is the fixed time-step interval, and β contains the parameters that are used to discretize β. Note that f_k is the discrete timestep update which is found by composing the dynamics from Eq. (1) with the particular time integration scheme. The above equations hold for explicit dynamics, however they can easily be modified to account for implicit updates. We likewise cannot integrate the QoI analytically so we again resort to a discretization scheme, e.g. trapezoid, Simpson, etc. so that (3) becomes

$$O(\beta) = \sum_{i=1}^{N} o_k(y_k, \beta, \Delta t)\,. \tag{17}$$

Similar remarks apply for the resolution of the adjoint problem (6) and the evaluation of the sensitivity which is now the derivative $\frac{\partial O}{\partial \beta}$ rather than the variation δO.

Note that in the previous analysis, we require consistency in space-time discretization schemes for primal analysis, evaluation of the objective O, the adjoint analysis, and evaluation of $\frac{\partial O}{\partial \beta}$. If the integration schemes for each of these operations are not chosen to be internally consistent (i.e. the spatial discretization is identical and the reverse time integration is appropriately adjoint to the forward time integrator), then each calculation will generate a discretization error which will propagate through to the gradient, see [29] and the references therein. The notion of a "good" or accurate derivative will be discussed in Section (2.4.1).

In the *discretize and differentiate* adjoint sensitivity analysis, we follow the same steps as in the infdim formulation, but we replace the infdim (1) with its discretized counterparts. Mimicking the infdim analysis with the augment QoI becomes

$$O_N(\beta) = \sum_{k=1}^{N} o_k(y_k, \beta, \Delta t) + \sum_{k=1}^{N} \lambda_k^T (y_k - f_k(y_{k-1}, \beta, \Delta t)) \tag{18}$$

Note that we omit an explicit terminal objective because it can be integrated into the sum. Differentiating with respect to β and rearranging leads to

$$\frac{\partial O}{\partial \beta} = \sum_{k=1}^{N} \left(\frac{\partial o_k}{\partial \beta}(y_k, \beta, \Delta t) - \lambda_k^T \frac{\partial f_k}{\partial \beta}(y_{k-1}, \beta, \Delta t) \right) - \lambda_1^T \frac{\partial f_1}{\partial y_0}(y_0, \beta, \Delta t) \frac{\partial y_0}{\partial \beta} + \tag{19}$$

$$\left(\frac{\partial y_N}{\partial \beta} \right)^T \left[\left(\frac{\partial o_N}{\partial y_N}(y_N, \beta, \Delta T) \right)^T + \lambda_N \right] + \tag{20}$$

$$\sum_{k=1}^{N-1} \left(\frac{\partial y_k}{\partial \beta} \right)^T \left[\left(\frac{\partial o_k}{\partial y_k}(y_k, \beta, \Delta t) \right)^T + \lambda_k - \left(\frac{\partial f_{k+1}}{\partial y_k}(y_k, \beta, \Delta t) \right)^T \lambda_{k+1} \right]. \tag{21}$$

Intermediate steps of the above calculation can be found in Section (A.2) We eliminate the terms involving implicitly defined derivatives $\frac{\partial y_k}{\partial \beta}$ by solving the terminal value adjoint problem: at t_N we equate

$$\lambda_N = -\left(\frac{\partial o_N}{\partial y_N}(y_N, \beta, \Delta t) \right)^T \tag{22}$$

and then we proceed backward in time evaluating $y_{N-1}, y_{N-2}, \dots$ where

$$\lambda_k = \left(\frac{\partial f_{k+1}}{\partial y_k} \right)^T \lambda_{k+1} - \left(\frac{\partial o_k}{\partial y_k}(y_k, \beta, \Delta t) \right)^T \tag{23}$$

Upon computing λ_k, the sensitivity reduces to the readily evaluated expression

$$\frac{\partial O}{\partial \beta} = \sum_{i=1}^{N} \left(\frac{\partial o_k}{\partial \beta}(y_k, \beta, \Delta t) - \lambda_k^T \frac{\partial f_k}{\partial \beta}(y_{k-1}, \beta, \Delta t) \right) - \lambda_1^T \frac{\partial f_1}{\partial y_0}(y_0, \beta, \Delta t) \frac{\partial y_0}{\partial \beta}\,. \tag{24}$$

We now summarize the computations. In the forward pass, we assign the initial conditions $y_0 = y_0$ and then, for each time step t_k for $k = 1, 2, \dots, N$, we evaluate y_k from (15) and accumulate the QoI of (17). In the reverse pass, we assign the terminal condition $\lambda_N = -\left(\frac{\partial o_N}{\partial y_N}(y_N, \beta, \Delta t)^T \right)$ and then, for each time step t_k for $k = N-1, N-2, \dots, 1$, we evaluate the adjoint repsonse λ_k from (23) and accumulate the QoI sensitivity of (24). If the initial condition y_0 is a function of β, then we lastly subtract $\lambda_1^T \frac{\partial f_1}{\partial y_0}(y_0, \beta, \Delta t) \frac{\partial y_0}{\partial \beta}$ from the sensitivity. The main benefit of the *discretize then differentiate* approach lies in the automatic consistency of all the discretization methods used. In particular, we require numerical consistency in how the objective functions are evaluated, the forward time integration scheme, and particularly the reverse time integration scheme. As will be shown in Section (2.4.1), if integration schemes do not match, non-zero errors will propagate through our solutions. This is avoided when discretizing then differentiating as all the integration schemes are kept consistent and there is no choice in backward integration scheme.

2.3. Graph network approach.

The *graph network* approach builds from the discretize then differentiate sensitivity analysis in the previous section. To begin we express (17) O_N using the forward calculation as

$$O_k(\beta) = O_{k-1} + o_k(y_N, \beta, \Delta t)\,, \tag{25}$$

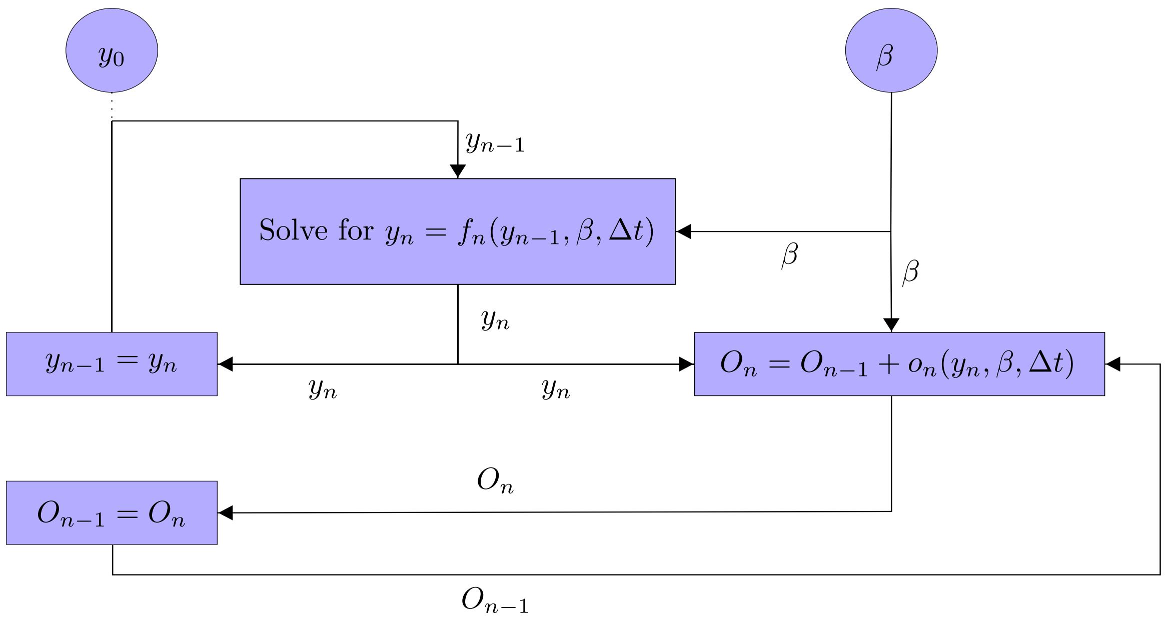


Fig. 1. Forward pass: QoI computation.

for $k = 1, 2, \ldots, N$ with the understanding that $O_0 = 0$. In this way, the forward pass can be summarized via the *graph* depicted in Fig. 1.

Indeed, as just mentioned, in the forward pass we assign the initial condition $y_0 = y_0$ and then for each time step t_k for $k = 1, 2, \ldots, N$ we evaluate y_k from (15) and accumulate the QoI of (25) (Fig. 2).

In calculating the gradients of the objective with respect to our intitial states and design parameters, we slightly modify the previously discussed *discretize then differentiate* reverse pass. Mathematically, the *graph* and *discretize then differentiate* approaches are identical, we merely introduce some notational conveniences and intermediate quantities. The main benefit of a *graph network* approach is that complex sets of function calls can be considered as nodes of a graph. Then, we simply need to traverse the graph backwards to calculate derivatives. This process allows for additional flexibility when considering complex physics and dynamical systems. In so far as the notation is concerned, we use the over-line to denote *summands* of the partial derivatives of O_N wrt. the overlined quantities quantities, e.g. all $\bar{\beta}$ are summed to evaluate $\sum \bar{\beta} = \dfrac{\partial O_N}{\partial \beta}$, likewise all $\bar{y}_k$ are summed to evalutate $\sum \bar{y}_k = \dfrac{\partial O_N}{\partial y_k}$. Note that the $\sum \bar{y}_k = \dfrac{\partial O_N}{\partial y_k}$ are not know; they are annihilated by the backward recursion adjoint sensitivity whereupon the are replaced by $\bar{\beta}$ and $\bar{y}_{k-1}$ summands to $\dfrac{\partial O_N}{\partial \beta}$ and $\dfrac{\partial O_N}{\partial y_{k-1}}$. The unknown sensitivity $\sum \bar{y}_{k-1} = \dfrac{\partial O_N}{\partial y_{k-1}}$ from time step t_k is reset for time step t_{k-1} and subsequently eliminated whereas the sensitvity summands $\bar{\beta}$ are continually added for all time steps. These backward recursions continue until we are left with $\dfrac{\partial O_N}{\partial \beta}$ and $\dfrac{\partial O_N}{\partial y_0} = \dfrac{\partial O_N}{\partial y_0}$, the latter being the sensitivity wrt. the initial conditions. Note that in the reverse pass, sums occur at the nodes where information splits in the forward pass. For instance, in the forward pass, the output y_k of the $y_k = f_k(y_{k-1}, \beta, \Delta t)$ analysis block is input into both the $O_k = O_{k-1} + o_k(y_k, \beta, \Delta t)$ QoI accumulation block and the $y_{k-1} = y_k$ increment block; in the reverse pass $\bar{y}_k$ is output from QoI accumulation and increment blocks and summed before it is input to the analysis block.

The beauty of the graph is that it automatically organizes the sensitivity computations. For example, the analysis block takes as input $(y_{k-1}, \beta, \Delta t)$ and spits out y_k. In the sensitivity analysis, the input to this block is $\sum \bar{y} = \dfrac{\partial O_N}{\partial y_k}$ and the output is $\bar{y}_{k-1}$ and $\bar{\beta}$ (as Δt is a constant). The derivatives $\bar{y}_k$ and $\bar{\beta}$ can be evaluated with AD. The exact implementation of AD at this level depends somewhat on the type and scale of the problem. In simple problems, the whole $\bar{y}_k$ and $\bar{\beta}$ can be found by writing the forward problem as a simple function call and using a code based automatic differentiation library to parse the derivatives of these calls. In more complex problems, such as those involving HPC, it is important to control the scale where software based AD tools are used in order to avoid holding large states in memory and avoid non-differentiable function calls such as scatter and gather operations. We demonstrate the specific implementation for large scale problems in Section 4 where automatic differentiation libraries are used in scalable and

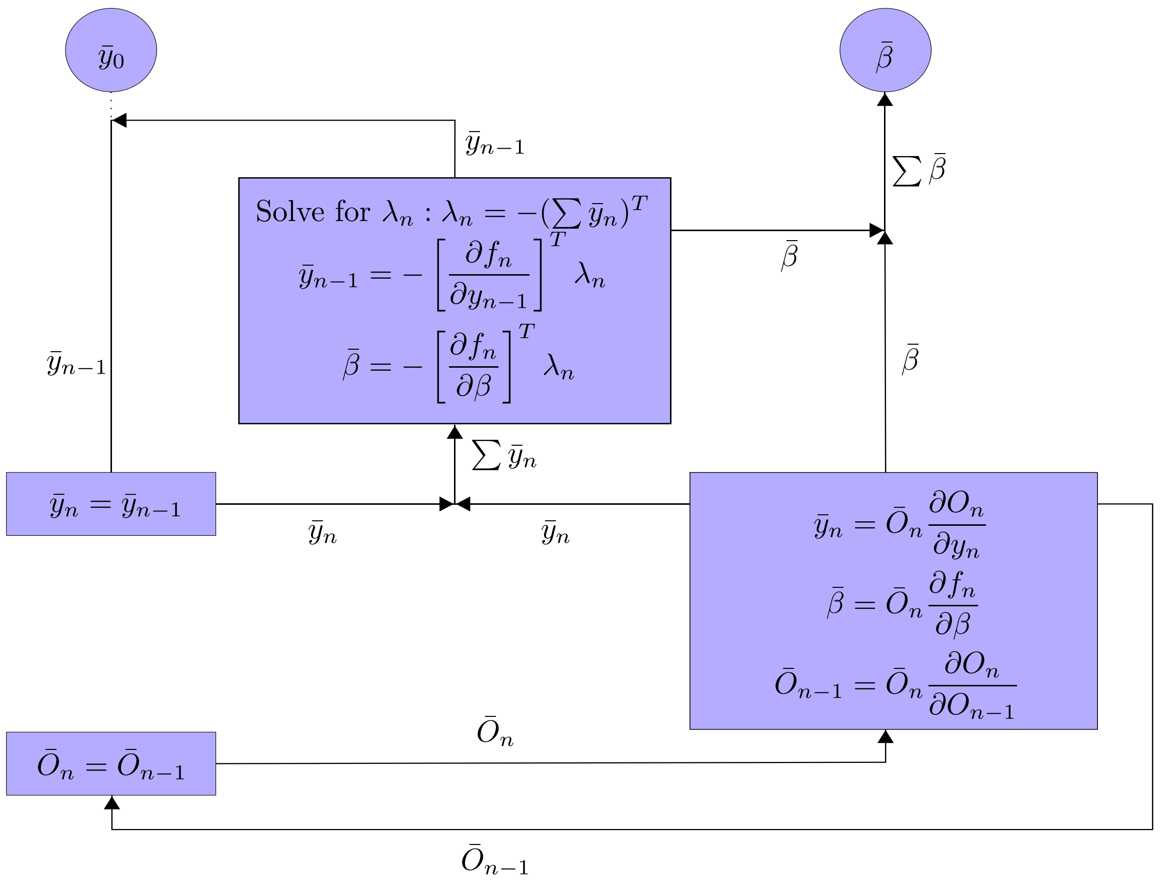

$\bar{y}_0$
$\bar{\beta}$
$\bar{y}_{n-1}$
$\sum \bar{\beta}$
Solve for $\lambda_n : \lambda_n = -(\sum \bar{y}_n)^T$
$\bar{y}_{n-1} = -\left[\frac{\partial f_n}{\partial y_{n-1}}\right]^T \lambda_n$
$\bar{\beta} = -\left[\frac{\partial f_n}{\partial \beta}\right]^T \lambda_n$
$\bar{\beta}$
$\bar{\beta}$
$\bar{y}_{n-1}$
$\sum \bar{y}_n$
$\bar{y}_n = \bar{y}_{n-1}$
$\bar{y}_n$
$\bar{y}_n$
$\bar{y}_n = \bar{O}_n \frac{\partial O_n}{\partial y_n}$
$\bar{\beta} = \bar{O}_n \frac{\partial f_n}{\partial \beta}$
$\bar{O}_{n-1} = \bar{O}_n \frac{\partial O_n}{\partial O_{n-1}}$
$\bar{O}_n$
$\bar{O}_n = \bar{O}_{n-1}$
$\bar{O}_{n-1}$

Fig. 2. Reverse pass: Adjoint computation.

memory efficient ways. Summarizing, the graph organizes the sequence of the computations for the sensitivity analysis and AD evaluates the necessary derivatives for the sensitivity analysis; thus fully automating the sensitivity analysis.

2.4. Computational considerations

While the above calculations are fundamental for building and studying the sensitivity analysis of a dynamical system, there are aspects we consider outside of the scope of the sensitivity analysis.

2.4.1. Assessment of gradient behavior

Our metric for what we consider to be an “accurate” gradient is whether it satisfies the famous Taylor remainder convergence test. Assume we are given a function f and another function df which is claimed to be the derivative of f. We can verify this claim by Taylor expanding it and checking the convergence.

$$f(x + h\delta x) = f(x) + hdf(x) \cdot \delta x + O(h^2)\,, \tag{26}$$

if we choose δx to be normalized and $h \in \mathbb{R}$ to be small. This can also be written as

$$T(x, h, \delta x) = \|f(x + h\delta x) - f(x) - hdf(x) \cdot \delta x\| = O(h^2)\,. \tag{27}$$

Therefore, in order to verify whether the given function df is the derivative of the function f, we verify not only that the error goes to zero, but that we have quadratic convergence of the left hand side of Eq. 27. Additionally, we verify that the direct error in the gradient ($hT(x, h, \delta x)$) also goes to zero.

To demonstrate, consider a nonlinear spring with dynamics given by $\ddot{y}(t) + ky(t) + \alpha y(t)^3 + b\dot{y}(t) = 0$. The parameters k, α, and β are the spring constant, hardening, and viscosity parameters, respectively. For simplicity, we set $k = \alpha = b = 1$. To integrate the system in time, we use a forward Euler integration scheme where $y_{i+1} = y_i + \Delta t f(y_i)$ For the backwards solve of the *differentiate then*

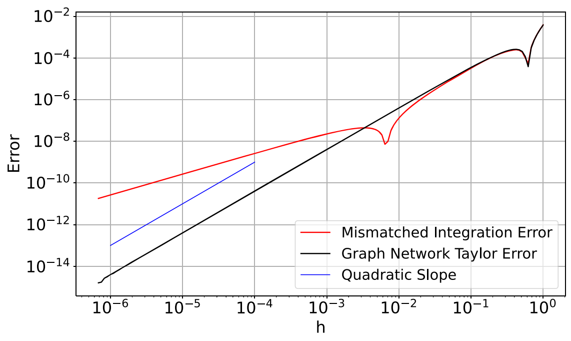


Fig. 3. Error for the Taylor test for various perturbation magnitudes h. The red line demonstrates incorrect scaling when mismatched integration schemes are used to calculate derivatives over a timestep. The black line (using the *graph network* approach) shows agreement with the blue line, indicating correct derivatives. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

discretize approach, we use a backwards Euler scheme which uses the forward step $\left(\lambda_{i-1} = \lambda_i + \Delta t \lambda_i \cdot \frac{\partial f}{\partial y}(y_i)\right)$. Note that this choice of backwards integration scheme is equivalent to if we had used an implicit forward method; however, because we use standard forward Euler, we introduce a numerical inconsistency.

We integrate this system with initial conditions of $y(0) = 1.1$, $\dot{y}(0) = 5.0$ and integrate until $T = 10$, define a scalar objective function $O(y_f) = y(T) + \dot{y}(T)$, and conduct the above Taylor test and plot the Taylor error in Fig. 3. We note that the *graph network* approach is numerically equivalent to the *discretize then differentiate* approach and thus gives the same gradient values. As can be seen, the *graph network* has proper convergence compared to that of the *differentiate then discretize* approach. Additionally, the *differentiate then discretize* gradient doesn't properly converge in the correct sense. This can present many problems in higher order optimization schemes, as accurate gradients are necessary.

Note that the discrepancy came in the choice of backwards integrator for the reverse problem in the *differentiate then discretize* method. While choosing the "correct" integrator when using a forward Euler method can easily be remedied, when using more complex time integrators, namely higher order Runge-Kutta schemes, the choice is not as obvious.

Various works, see [30–33], discuss the topic of consistency of discrete adjoint time integration schemes. While there are analytic solutions for some cases, namely one-step integration schemes, a general solution for complex multi-step methods is not known to exist, especially in the case of adaptive timestepping.

The *graph network* approach, consequently, presents a significant advantage over the *differentiate then discretize* approach in that the choice of integrator for the reverse pass is exactly specified.

2.4.2. Automatic differentiation

Automatic differentiation (also known as "auto-diff", "auto-grad", or simply "AD") is a computational technique used to efficiently and accurately evaluate derivatives of mathematical functions. It plays a crucial role in various fields such as machine learning, optimization, physics, simulations, and scientific computing. The most popular AD libraries are PyTorch [34], TensorFlow [35], and Jax [36], specifically due to their integrations into popular machine learning libraries.

The basic idea behind AD is to decompose functions into a sequence of elementary operations (such as addition, multiplication, exponentiation, etc.), for which the derivatives are well known. Then by applying the chain rule recursively to these operations, AD can compute the derivative of the entire function with respect to its input variables. This technology enables rapid prototyping of tools involving differentiation, and can readily differentiate complex functions.

AD is extremely useful when developing analytical tools. Consider the case where AD is currently being used extensively: machine learning. Without AD tools, users of ML packages such as PyTorch and Tensorflow would have to manually specify gradients of their loss functions with respect to the neural network architecture. While this is not an impossible task, it would put severe limitations on who can use these tools, how quickly different models can be tested, and introduce many avenues for error. Arguably, the integration of automatic differentiation is the impetus which allowed the field to thrive as it has today.

AD is also important in expanding the scope of problems we can study. In many cases, especially problems with iteration, composition, recursion, branches, and complex algebras, taking derivatives by hand is not feasible and, as with the previous point, extremely prone to error. A much more convenient approach is to define computational methods which can handle the complexity and accurately traverse the computational graphs.

Automatic differentiation, however, is not without its faults and drawbacks. Often, naive implementations of AD can lead to inefficient run times and massive memory usages. This is because, unless otherwise specified, the tool must hold the entire computational graph, including sensitivities, in memory all at once. In large scale problems, particularly those involving dynamic simulations of hydrodynamic systems, this is cost prohibitive. As a result, we must control how we apply automatic differentiation and carefully consider various aspects such as memory allocation and code structure.

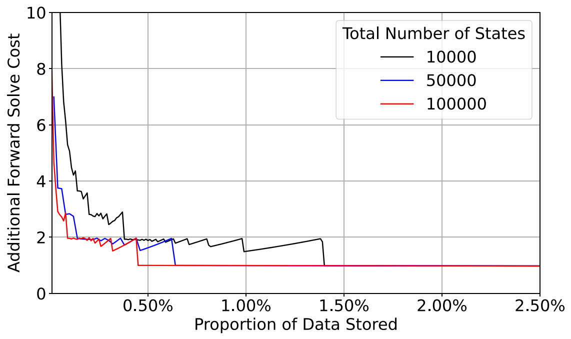


Fig. 4. Amount of additional compute time added (measured in terms of the time of the forward solve) added given a memory budget written as a percentage of the total number of states from the forward solve. The different lines are for different numbers of total states from the forward solve.

For the following work, we use the automatic differentiation library Enzyme ([37,38]). This framework presents a few benefits. First, LLVM compilers allow us to use various programming languages in conjunction with this tool, opening up opportunities such as Julia and C++. Next, compile time optimization creates fast and reliable tools which are essential to generate code which can run at large scales on HPC.

For the work in Section 3, we use the MFEM library ([39]) due to its excellent scaling to HPC as well as various features such as partial assembly, sum factorization, native support for arbitrarily high order elements. Additionally, it is written in C++, which allows us to use Clang compilers (https://llvm.org/) to link with the Enzyme library for automatic differentiation. A differentiable finite element library named ∂FEM ([40]) is currently in production to natively support automatic differentiation through the finite element function call stack.

2.4.3. *Data storage/checkpointing*

When calculating adjoints for nonlinear dynamial systems it is necessary to access the state data of every timestep from the forward solve during the reverse pass. This presents an issue in complex problems with many time steps and fine discretizations, as memory (RAM) can quickly become a limitation. For example, a typical hydrodynamic problem can involve solving for a system with a few million parameters. This system may require $O(1M)$ time steps to fully resolve. Assuming we are storing each parameter in double precision,

$$\left(\frac{8 \text{ bytes}}{\text{parameter}}\right)\left(10^6 \text{ parameters}\right)\left(10^6 \text{ time steps}\right) \approx 8 \text{ terabytes}.$$

This, unfortunately, is already exceeding the memory resources of current compute systems, and the problem becomes worse with more complex multiphysics and long duration simulations.

One option for addressing this issue is to selectively store data to disk and load it into RAM when needed. This process is often inefficient in practice due to the relatively low bandwidth and high latency of disk reads and writes, leading to bad performance of both the forward and reverse solves. Another option is the use a checkpointing scheme. The strategy here is to only save a relatively small number of simulation states in (relatively higher bandwidth) RAM. On the reverse pass, the data for each state going back in time is required for the adjoint calculation. When a state is reached that is not currently stored in RAM, the most recently saved checkpoint is fetched from RAM and the subsequent data states are recomputed by integrating forward in time again until the desired state is available. Algorithms for constructing an optimal schedule for checkpointing and fetching to minimize the required number of recompute steps have been developed in: Griewank and Walther [41], which provides an optimal method when the number of time steps is known up front; Wang et al. [42], which has good performance even when the number of steps is initially unknown (e.g., when the stable timestep evolves as the simulation progresses); and Herrmann and [43] (and other references within), where different levels of available memory with varying costs and resources are considered. Here we use the dynamic checkpointing algorithm from Wang et al. [42] to eventually allow for the possibility of varying timesteps, and we are currently only checkpointing to a single level of memory, namely, RAM. The tradeoff between memory and compute time in our framework can be seen in Fig. 4. When traversing the states backwards in order to accumulate the gradients, we will reach states which are not saved in the checkpoint buffer. When this occurs, we load the last checkpointed state and reconstruct the current needed state, saving and recalculating as necessary as per the checkpointing algorithm. In all cases shown, by storing only 1.5% of the total data in an optimal manner, we incur an additional cost of about 1 forward solve for recomputing the required states. The additional cost decays roughly linearly to zero in the limit where 100% of the data is stored.

2.4.4. *Gradient based optimization*

The above methods allow us to calculate the derivative of some objective function with respect to the initial (generalized) state. Access to this gradient, coupled with a gradient descent algorithm, allows us to minimize (or maximize) the given objective function.

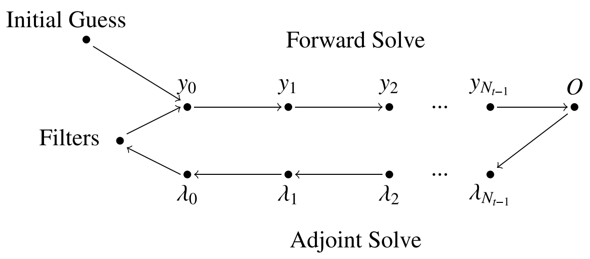


Fig. 5. Graph outline of the optimization cycle starting with an initial guess y_0. The solution is propagated from y_0 to O. If the objective is not sufficiently converged, then the bottom path is taken to calculate the gradient. Then, the filters can be applied then the initial guess can be updated. This process can be iterated until convergence.

For example, given an initial state y_0, we can calculate the forward solve to evaluate $O(y_0)$, then the adjoint solve to evaluate $\frac{\partial O}{\partial y_0}$. We then update the initial state with

$$y_0 \leftarrow y_0 - \alpha \frac{\partial O}{\partial y_0},$$

where α is a chosen gradient descent parameter. An outline of this problem structure can be seen in (Fig. 5).

Many of these optimization problems follow a similar structure (i) problem setup which defines physics and initial conditions, (ii) forward pass which advances time-stepping algorithm, (iii) objective function which maps final state(s) to a scalar value which we wish to minimize, and (iv) adjoint calculation via reverse pass where the gradient is accumulated.

2.4.5. Filters

After completing the adjoint solve loop as shown in Fig. 5, there is a step where filters are applied. A filter, in this case, is any operation which changes the full gradient vector. A few common uses of filters for optimization problems can be regularity (enforcing continuity or differentiability), subsampling (reducing the DOFs to a subset), or physical constraints. Historically this this step was used to eliminate the ubiquitous checkerboarding modes present in density based design optimization [44]. The choice of filter is problem dependent and outside of the scope of this work, however it plays a vital role in the performance and implementation of many optimization algorithms [45]. A more complete discussion of filters in PDE constrained optimization can be found in Bourdin [45], Sigmund [46], Zhou et al. [47] and many more works.

2.4.6. Example: optimizing multi-particle systems

For an introductory example, we introduce a system of interacting particles with two-body interactions. We label particles with degrees of freedom

$$\begin{aligned} \text{Position} &- \mathbf{x} \\ \text{Velocity} &- \mathbf{v} \\ \text{Charge} &- q \end{aligned}$$

Each particle is being acted upon by Coulomb and gravitational forces of the form

$$\mathbf{F}_i = \sum_{j\neq i} \frac{q_i q_j \mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^3} - g\mathbf{e}_2,$$

where the subscripts i and j indicate individual particles, $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, g is the gravitational constant, and $\mathbf{e}_2$ is the direction of the gravitational force. The inclusion of the Coulomb interaction causes the dynamics of the system to be nonlinear. The dynamics of the system are given by

$$\begin{aligned} \dot{\mathbf{x}}_i &= \mathbf{v}_i \\ \dot{\mathbf{v}}_i &= \mathbf{F}_i \\ \dot{q}_i &= 0 \end{aligned}$$

Particles are initialized at random (non-overlapping) locations with zero initial velocity and uniform charge $q_i = 1.0$. The forward trajectory can be found by composing the physics with a time integration scheme. We choose a 4th order Runge-Kutta scheme summarized by

$$y_{k+1} = y_k + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{28}$$

$$k_1 = f(t_k, y_k), \tag{29}$$

$$k_2 = f\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2}k_1\right), \tag{30}$$

$$k_3 = f\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2} k_2\right), \tag{31}$$

$$k_4 = f\left(t_k + \Delta t, y_k + \Delta t k_3\right). \tag{32}$$

As an illustrative objective function, let's define a quantity of interest of the form

$$O = \frac{1}{2} \sum_{i=0}^{N_p - 1} \|\mathbf{x}_i^f - \mathbf{x}_i^*\|^2$$

where $\mathbf{x}_i^f$ is the final position of the particle, $\mathbf{x}_i^* = R(\cos\left(\frac{2\pi i}{N_p}\right)\mathbf{e}_1 - \sin\left(\frac{2\pi i}{N_p}\right)\mathbf{e}_2)$ and R is the radius of the circle we wish to target. The goal is to minimize the above function with respect to the initial state variables. We will specifically consider the case where we are only allowed to control the initial velocity of each particle and not the position or charge. The initial derivative of the objective function can be found either through manual calculation or by using automatic differentiation.

The adjoint of this integration scheme can be summarized as

$$\begin{aligned}
M &= \lambda \cdot y_{k+1} \\
\bar{M} &= 1 \\
\bar{y}_{k+1} &= \bar{M}\lambda \\
\bar{k}_4 &= \frac{\Delta t}{6} \bar{y}_{k+1} \\
\bar{k}_3 &= \Delta t \bar{k}_4 \cdot \frac{\partial f}{\partial y}(t_k + \Delta t, y_k + \Delta t k_3) + \frac{\Delta t}{3} \bar{y}_{k+1} \\
\bar{k}_2 &= \frac{\Delta t}{2} \bar{k}_3 \cdot \frac{\partial f}{\partial y}\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2} k_2\right) + \frac{\Delta t}{3} \bar{y}_{k+1} \\
\bar{k}_1 &= \frac{\Delta t}{2} \bar{k}_2 \cdot \frac{\partial f}{\partial y}\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2} k_1\right) + \frac{\Delta t}{6} \bar{y}_{k+1} \\
\bar{y}_k &= \bar{y}_{k+1} + \bar{k}_4 \cdot \frac{\partial f}{\partial y}\left(t_k + \Delta t, y_k + \Delta t k_3\right) + \bar{k}_3 \cdot \frac{\partial f}{\partial y}\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2} k_2\right) \\
&\quad + \bar{k}_2 \cdot \frac{\partial f}{\partial y}\left(t_k + \frac{\Delta t}{2}, y_k + \frac{\Delta t}{2} k_1\right) + \bar{k}_1 \cdot \frac{\partial f}{\partial y}\left(t_k, y_k\right).
\end{aligned}$$

By calculating $\bar{y}_k$ using the above algorithm, we have the incremental adjoint. Note that the above can be implemented manually, or calculated using an automatic differentiation package, defining custom gradients for the adjoints of the physics. By accumulating the incremental adjoint from the final time step to the initial time step using methods described in 2.3, we obtain the gradient with respect to the objective function. Using the methods described in the previous sections, we begin with a set of initial states

$$\mathbf{x}_i, \mathbf{v}_i, q_i$$

We can simulate the forward problem by using the 4th Order Runge-Kutta described in Eq. 28. In the solve, we cache all of the states in the forward solve. We then calculate the objective function and its initial gradient (with respect to the final state). Then, apply the incremental adjoint to march backwards in time (using the cached data) until the initial time step. At the end of this process, we have the gradient of the objective function with respect to the initial positions, velocities, and charges. We can then use an optimization algorithm, in our case conjugate gradient descent, to update the initial conditions. Because we only wish to optimize with respect to the initial velocities, we only update those quantities and zero out the perturbations of the positions and charges. This process can be repeated until convergence. The results are demonstrated in Fig. 6.

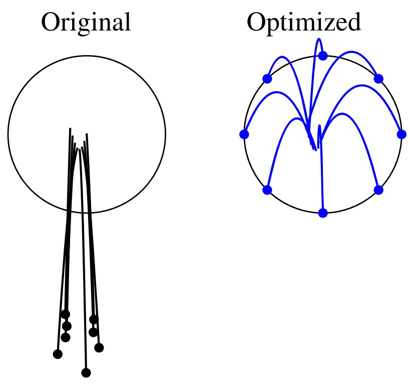
While optimization of the positions of a system of particles seems straightforward, it actually illuminates many features of dynamic optimization. Inspection of the numerical solution of this particular particle system gives chaotic results in the informal sense that seemingly small changes may yield significantly different outcomes. We have found that there is often an interesting sweet spot in such hyperbolic systems wherein usage of gradient information is exceptionally meaningful; systems which are either very chaotic or not chaotic at all are rather boring from a gradient optimization perspective, whereas systems which are somewhat chaotic can be readily controlled employing gradient optimization. While we have no formal proof of this behavior, we suspect that this trend is actually quite general.

3. Finite element method discretization of Lagrangian hydrodynamics

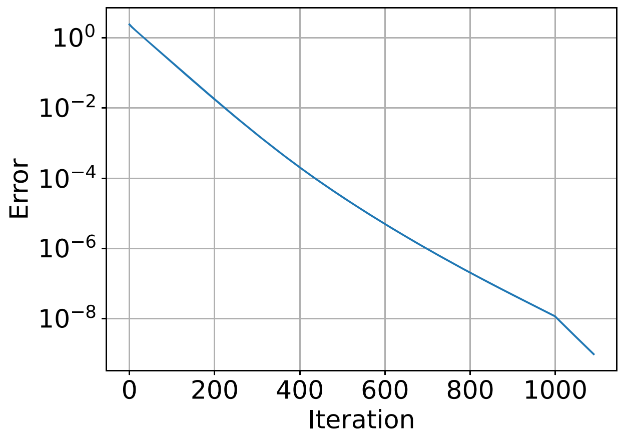
In this section, we apply the techniques discussed in the previous section to develop a method for computing adjoints of the equations of Lagrangian Hydrodynamics. We focus on a particular high-order discretization method which is described in Dobrev et al. [48]. This requires special considerations for features like artificial viscosity and time step estimates. We review the spatial discretization of these equations into finite element bases, and the computation of the adjoint of each of these terms.

3.1. Summary of equations

The equations of Lagrangian Hydrodynamics (see [48,49] for considerably more background and detail) describe the flow of continuous matter under the action of extreme pressures and energy deposition. The differential forms of the equations of motion, as



(a)



(b)

Fig. 6. (a) Plots of the unoptimized (black) and optimized solutions (blue). (b) Plot of the error in the conjugate gradient solve demonstrating linear slope in the log-y scale. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

defined in an Eulerian reference frame are:

$$\text{Momentum conservation}: \quad \rho\frac{dv}{dt} = \nabla\cdot\sigma, \tag{33}$$

$$\text{Mass conservation}: \quad \frac{1}{\rho}\frac{d\rho}{dt} = -\nabla\cdot v, \tag{34}$$

$$\text{Energy conservation}: \quad \rho\frac{de}{dt} = \sigma : \nabla v, \tag{35}$$

$$\text{Equation of motion}: \quad \frac{dx}{dt} = v, \tag{36}$$

$$\text{Stress Relation}: \quad \sigma = -pI + \sigma_v, \tag{37}$$

where v is the material velocity, e is the internal energy per unit density, ρ is the current density, p is the pressure calculated through an equation of state (EOS), σ is the stress, and σ_v is an artificial viscosity Additionally, ∇ is the spatial differential operator with respect to the current configuration. Note that the stress is often a nonlinear function of the state and therefore the dynamics are nonlinear. The artificial viscosity regularizes strong shocks which otherwise would have a thickness significantly below the spatial resolution of the computational mesh [50]. This topic will be discussed in more detail in Section 3.2.

To generate the weak form, we multiply by test functions ϕ_i^v and ϕ_i^e then integrate over the domain Ω where the superscripts v and e indicate the kinematic and thermodynamic function spaces, respectively. For brevity, we only demonstrate the momentum conservation equation with similar steps for the energy conservation equation. We have

$$\int_\Omega \rho\frac{dv}{dt}\cdot\phi^v dx = \int_{\Omega(t)} (\nabla\cdot\sigma)\cdot\phi^v dx \tag{38}$$

$$= -\int_{\Omega(t)} \sigma : \nabla\phi^v dx \tag{39}$$

$$= -\int_{\Omega_0} \sigma : \left(\nabla_X\phi^v F^{-1}\right) J dX \tag{40}$$

$$= -\int_{\Omega_0} \left(J\sigma F^{-T}\right) : \nabla_X\phi^v dX \tag{41}$$

$$= -\int_{\Omega_0} P : \nabla_X\phi^v dX \tag{42}$$

where Ω_0 is the initial/reference configuration, $F = \nabla_X(x)$ is the deformation gradient, $J = \det(F)$ is the Jacobian, and P is the Piola-Kirchhoff stress. Each spatial dimension is discretized using 3rd order H^1 elements in kinematic variables (x, v) and 2nd order L^2 elements with positive Bernstein polynomials in the thermodynamic variable e.

The pull-back operation into the reference configuration carries a few advantages. First, the mass matrix has constant coefficients in time. This significantly reduces the total amount of computation because the mass matrix only needs to be calculated and inverted once and can be re-used as long the mesh is not moving relative to the reference configuration. A consequence of this is that the relation for density becomes algebraic through mass conservation as $\det(F)\rho = \rho_0$ where ρ, ρ_0 are the densities in the current and reference configuations, respectively, and we no longer need to solve for the mass conservation component of Eq. (33). Next, anisotropic material properties do not need to be advected as material points have fixed locations in the mesh.

We use a Mie-Guneisen equation of state of the form

$$p = \frac{\rho_0 C_0^2 \chi}{(1 - s\chi)^2}\left(1 - \frac{\Gamma_0}{2}\chi\right) + \rho_0 \Gamma_0 e\,, \qquad \chi = 1 - \frac{\rho_0}{\rho}$$

where Γ_0 is the Gruneisen parameter and s is the linear Hugoniot slope coefficient.

3.2. Artificial viscosity

As stated in the previous section, artificial viscosity is critically important when considering shock propagation in order to prevent spurious oscillations without damping out the features of the shock. Various works, see [50–52], discuss the formulations and derivations. We use the particular form of artificial viscosity [48,52] given by

$$\sigma_v = 0.75\rho(\gamma_1 l c + \gamma_2 l |\Delta v|) H(\Delta v) \mathrm{sym}(\nabla v) \tag{43}$$

where γ_i are strength parameters, l is a length scale associated with an element ($l = l_0 \det(F)^{1/\mathrm{dim}}$), l_0 is the initial length scale, c is the wave speed in the element, $\Delta v = \mathrm{tr}(\nabla v) l$ is the velocity jump across the element, and H is the Heaviside function. This form has been demonstrated to sufficiently resolve the shock front while not overly damping the rest of the behavior; although we note that for truly high-order dissipation, which we will not consider here. Additional limiting is required, such as the hyperviscosity treatment shown in Bello-Maldonado et al. [53]. The choice of artificial viscosity, however, does not change the mathematical formulation, as the automatic differentiation tools allow for seamless transition between functional forms without having to recalculate derivatives.

One complication of this function in the context of calculating adjoints is the compression switch. In standard methods, we use a Heaviside function which is both non-differentiable, but also discontinuous. This is partially remedied by being multiplied by $\mathrm{sym}(\nabla v)$, which raises this to being continuous but non-differentiable. We will discuss the issue of non-differentiability in a future work. For now, we remedy this by replacing all non-differentiability with a suitable smoothing function. In the case of the Heaviside function, we use a sigmoid scaled with the wave speed

$$H(-x) \to \mathrm{sigmoid}\left(-\frac{x}{h}\right).$$

Additionally, the absolute value in the quadratic term must replaced with a soft absolute value. This is done by

$$|x| \to \mathrm{softabs}(x, h) = \mathrm{silu}\left(\frac{x}{h}\right) + \mathrm{silu}\left(\frac{-x}{h}\right)$$

where silu is the sigmoid linear unit and h is a length scale. Exact definitions of these functions can be seen in Eq. (A.6).

In both these cases we choose $h = 0.2c$ to properly scale the transition in the smoothed regions to properly account for the behavior yet maintain differentiability where c is a representative wave speed.

3.3. Vector Jacobian products with automatic differentiation

Many of the above rely on a *vector-Jacobian product* (VJP) to propagate and accumulate the gradient. In many cases we will use this synonymously with the term *adjoint product*. Recall from Section (2.2) that we require the computation of the VJP of the time derivatives. To generate the VJP, we construct the global inner product

$$M = \lambda \cdot \dot{y} = \sum_i \lambda_i^x \dot{x}_i + \sum_i \lambda_i^v \dot{v}_i + \sum_i \lambda_i^e \dot{e}_i \tag{44}$$

where the vector values refer to the discrete state vectors associated with those degrees of freedom.

If we plug in the semi-discrete forms of the equations of motion, we have

$$M = \sum_i \lambda_i^x v_i - \sum_{ij} \lambda_i^v M_{v,ij}^{-1} \int_{\Omega_0} \left(P : \nabla_X \phi_j^v\right) dX + \sum_{ij} \lambda_i^e M_{e,ij}^{-1} \int_{\Omega_0} (P : \nabla_X v) \phi_e^j dX \tag{45}$$

$$= \sum_i \lambda_i^x v_i - \int_{\Omega_0} (P : \nabla_X \tilde{\lambda}_v) dX + \int_{\Omega_0} (P : \nabla_X v) \tilde{\lambda}_e dX \tag{46}$$

$$= \sum_i \lambda_i^x v_i + \int_{\Omega_0} (P : (\tilde{\lambda}_e \nabla_X v - \nabla_X \tilde{\lambda}_v)) dX \tag{47}$$

$$= \sum_i \lambda_i^x v_i + \int_{\Omega_0} m\, dX\,, \tag{48}$$

where we $\tilde{\lambda}_i = \sum_j M_{ij}^{-1} \lambda_j$ and $m = P : \left(\tilde{\lambda}_e \nabla_X v - \nabla_X \tilde{\lambda}_v\right)$ is the local inner product. Note that we project the discrete adjoints ξ_i onto their continuous space counterparts ξ by using the shape functions associated with those fields. By taking the derivative of M with respect to the primal fields, we are able to construct the VJP of the time derivatives.

For example, if we take the derivative with respect to the velocity state variable, we have

$$\frac{\partial M}{\partial v_i} = \lambda_i^x + \int_{\Omega_0} \left(\frac{\partial m}{\partial v} \cdot \phi_i^v + \frac{\partial m}{\partial \nabla_X v} : \nabla_X \phi_i^v\right) dX\,. \tag{49}$$

The above expression allows us to immediately see the benefit of constructing m. First, we have turned the adjoint product calculation into a term which looks like a force call. This allows us to take advantage the standard finite element framework which exists to calculate linear forms and calculate adjoints in a matrix-free manner. Next, the derivatives of m are now done at the quadrature points and are entirely local and parallelizable. *This is ideal for automatic differentiation as we no longer have to consider information transfer across compute nodes in our computational graph.* This treatment of adjoints can be generalized and can be see in Section A.4 and the "correctness" is verified in Section A.5.

4. Suppression of Richmyer-Meshkov instability induced jetting

As noted in the introduction, RMI plays an important role in many scientific and engineering applications. Depending on the use case, either enhancement or suppression of the RMI jet may be desired. In this section, we develop a computational model of RMI, multi-objective functions to describe what we are after in terms of stable shock acceleration, the adjoint computation, some specific discussion of tracer particles practically needed to implement our specific objective function, and results showing RMI suppression.

4.1. *Forward pass*

The domain is separated into two regions. The left is high density ($\rho_0 = 10$) and the right is low density ($\rho_0 = 1$). A subset of the left domain ($\Omega_1 \subset \Omega$ s.t. $X < 1$) is the controllable domain and is initialized to a higher internal energy state $e(X \in \Omega_1, 0) = 0.15$ with the rest $e(X \notin \Omega_1, 0) = 0.0$. The top, left, and bottom boundaries allow for sliding boundary conditions while the right boundary is completely free.

A shock wave is generated due to the internal energy of the left side being higher than the rest of the domain. This causes a high pressure region with a sharp interface against a low pressure region, resulting in a force along the interface. As the high energy region expands, it generates a shock wave which propagates through the high density region. Once it hits the interface, baroclinic torque is generated due to the misalignment between the pressure gradient and the density gradient. This torque causes the system to evolve in such a way that the interface will invert itself and continue to grow. We use the 4th order Runge-Kutta for time integration and solve until $t = 7$.

The goal is to design the profile of the internal energy in Ω_1 to minimize the RMI jet length at a particular time.

4.2. *Objective function*

In order to minimize the jet length, we need a metric which takes the state of the system as an input and returns a single scalar value. By minimizing said functional, we obtain better results (i.e. reducing the jet length). Additionally, we would like to push the interface and not remove all accelerations (i.e. we want the optimizer to avoid the trivial solution of no acceleration). As a result, we would also like to include a term in the objective that increases the velocity of the interface while decreasing the objective value. In order to accomplish this, we introduce Lagrangian tracer particles. These are virtual particles which are linked to particular material points. One point of complexity is that, although tracer particles traditionally can be defined completely locally to the thread that "owns" that point, this is not true when conducting adjoint calculations. Here, we must collect the data from all the tracer particles onto a root thread in order to take derivatives of mathematical operations between the tracers, then propagate those derivatives back to the relevant threads. This will become clear in Eq. (50) as simply calculating that objective requires all of the tracer data to be known to a single thread. The total gradient of this objective (with respect to the tracer point values) then needs to be distributed to the threads which take "ownership" over the data.

We introduce a notation where x_i are the X components of the deformation of particles i, v_i are the X components of the velocities of particles i, $x_{\text{outer}} = \text{ave}(x_2, x_3)$, $v_{\text{ave}} = \text{ave}(v_1, v_2, v_3)$; then define the terminal objective

$$O = \frac{1}{2}\lambda_1 (x_1 - x_{\text{outer}})^2 + \frac{\lambda_2}{\delta + |v_{\text{ave}}|} \tag{50}$$

where $\lambda_i \geq 0$ are scaling factors and all measurements are taken at the final timestep. By inspection, it can be seen that this function is minimized when

$$\begin{aligned} x_{\text{outer}} &= x_1\,, && \text{Flatten the interface} \\ v_{\text{ave}} &= \pm\infty\,, && \text{Accelerate interface} \end{aligned}$$

when $\lambda_i > 0$.

In our particular case, we would like to push $v_{\text{ave}} \to +\infty$ as opposed to the other side. This is remedied by picking a suitable initial guess and using a local optimizer to push the solution towards one solution as opposed to the other. If using completely random initial conditions or a global minimizer, then a different objective function may be necessary. Also, note that Lagrangian tracer particles use local data to construct their state, as a result, we can generally write that

$$\mathbf{x}_i = \mathbf{x}_i(X, x, v, e), \tag{51}$$

$$\mathbf{v}_i = \mathbf{v}_i(X, x, v, e), \tag{52}$$

$$e_i = e_i(X, x, v, e)\,, \tag{53}$$

where the terms on the left are features of the tracer particles and the equations on the right are functions which project the global state vectors to the tracer data.

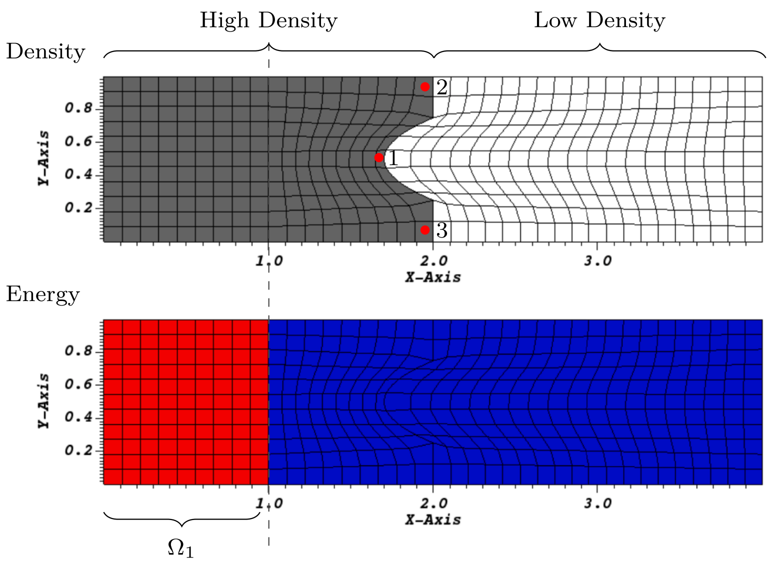


Fig. 7. Domain and initial energy configuration for the RMI case study. The nodes (1, 2 ,3) indicate where the tracer particles are placed (used in Eq. (50)).

4.3. Adjoint calculation

As a result of the tracer definition in (51), we can calculate the gradient of the objective as

$$\frac{\partial O}{\partial X} = \sum_i \left(\frac{\partial O}{\partial \mathbf{x}_i} \cdot \frac{\partial \mathbf{x}_i}{\partial X} + \frac{\partial O}{\partial \mathbf{v}_i} \cdot \frac{\partial \mathbf{v}_i}{\partial X} + \frac{\partial O}{\partial e_i} \frac{\partial e_i}{\partial X} \right),$$
$$\frac{\partial O}{\partial x} = \sum_i \left(\frac{\partial O}{\partial \mathbf{x}_i} \cdot \frac{\partial \mathbf{x}_i}{\partial x} + \frac{\partial O}{\partial \mathbf{v}_i} \cdot \frac{\partial \mathbf{v}_i}{\partial x} + \frac{\partial O}{\partial e_i} \frac{\partial e_i}{\partial x} \right),$$
$$\frac{\partial O}{\partial v} = \sum_i \left(\frac{\partial O}{\partial \mathbf{x}_i} \cdot \frac{\partial \mathbf{x}_i}{\partial v} + \frac{\partial O}{\partial \mathbf{v}_i} \cdot \frac{\partial \mathbf{v}_i}{\partial v} + \frac{\partial O}{\partial e_i} \frac{\partial e_i}{\partial v} \right),$$
$$\frac{\partial O}{\partial e} = \sum_i \left(\frac{\partial O}{\partial \mathbf{x}_i} \cdot \frac{\partial \mathbf{x}_i}{\partial e} + \frac{\partial O}{\partial \mathbf{v}_i} \cdot \frac{\partial \mathbf{v}_i}{\partial e} + \frac{\partial O}{\partial e_i} \frac{\partial e_i}{\partial e} \right)$$

Note that the relations above tend to be simple, but we maintain all the terms for the sake of generality. Similar to previously discussed methods, we calculate the adjoints of the time integration scheme and of the physics separately and compose them in order to find the incremental adjoint. By stepping backwards in time until the initial timestep, we can accumulate the adjoint of the objective with respect to the entire initial state X, x, v, e. From here, we mask the adjoint to only include the components in Ω_1. We note that, although we are using tracer particles and our objective function is dependent on only a few state variables, the overall adjoint structure will propagate the sensitivities to the whole domain as we move backwards in time, so the sensitivity *does not* maintain the sparsity of the initial variations.

An example of the forward and adjoint loops are shown in Fig. 8. We plot the pressure evolution from the forward pass; however, we cache all the data needed to recreate the state using the checkpointing methods from Section 2.4.3. Then, we calculate our objective function (see Eq. 50) and its gradient with respect to the final state. Performing adjoint calculations, we step backwards in time, recreating the state as needed using the checkpoints. We visualize the adjoint of the energy field (masked by the domain of control Ω_1) as we step backward in time. At the final time step of the adjoint solve ($t = 0$) we are left with the gradient of our objective with respect to the initial state within Ω_1. After subsampling our full gradient vector to the sub-domain Ω_1, we are left with about 400 degrees of freedom. We then update our initial state using *gradient descent* with the provided perturbation. Visualization of the adjoint fields is extremely useful when trying to understand the system's sensitivities. Additionally, these fields provide vital information which designers can use to improve designs ad hoc.

4.4. Results

(Fig. 9) summarizes the results of the optimization procedure described in the previous sections.

We note a few features of the solution. First, consider the bifurcation of the initial energy into left and right "hot spots". These regions will cause expansion in both of these locations, causing a complex shock front. Notably, it actually splits the shock into two different features. First, it will split the shock so it hits the bottom and top faces harder than the middle inclusion. This will temper

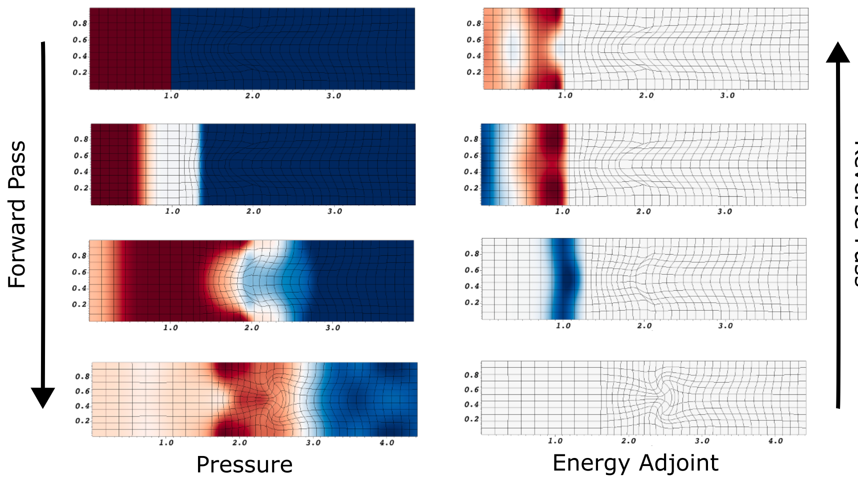


Fig. 8. Visualization of pressure and adjoint fields through the solve. The top plots are solutions at time $t = 0$ while the bottom are the final steps at $t = 7$. Following the result from the top left and working down, we integrate forward in time. Then from the bottom left to bottom right, we calculate the objective and initial gradient. Moving up, we integrate backwards in time to solve for the adjoint energy field to calculate the gradient. Note that the system has the density interface seen in Fig. 7.

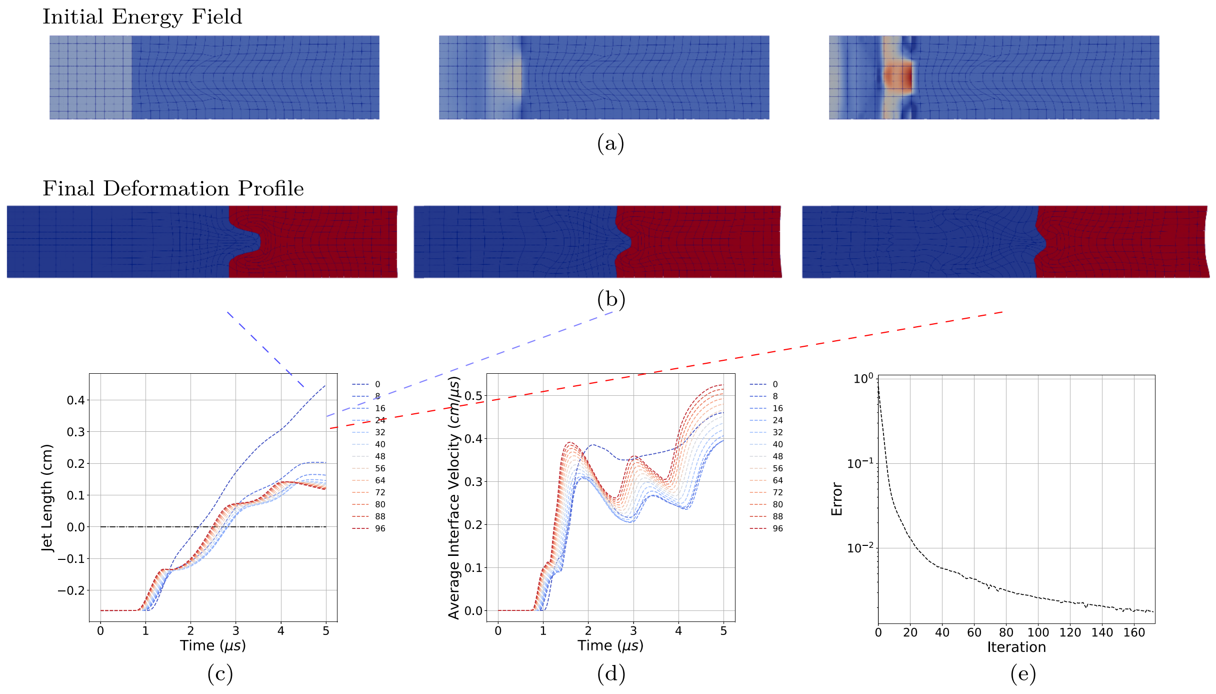


Fig. 9. Demonstration of mitigation of RMI through the gradient descent procedure. (a) The initial energy profile at three different stages (initial guess, 8 steps, 88 steps). (b) The deformation profile at the final timestep for each of those same three stages. (c) The evolution of the jet length over time for different iterations of gradient descent. (d) The average interface velocity over time for different iterations. (e) The change in objective function for each iteration.

RMI growth to flatten the interface. Next, a second, stronger shock wave hits the (now flattened) interface accelerating it more. This type of energy distribution is intuitive in hindsight, but we find it remarkable that this was discovered automatically through gradient based optimization.

The optimization algorithm can be seen to go through multiple stages. This is due to the competing objectives given in Eq. (50). Initially, the priority is interface flattening. Then, when that error is sufficiently low, the interface velocity is slowly increased. This behavior can be seen in (Fig. 9) as the final interface velocity initial is lowered, then, while maintaining a flat interface, the interface velocity is raised. Eventually, the interface velocity even surpasses that of the initial guess. Remarkably, this demonstrates that utilizing gradient optimization, RMI suppression is readily achievable without a decreasing the intensity of the drive. Moreover, as

RMI is a general, challenging prototype for hydrodynamic behavior, this example demonstrates the ability of this class of optimization methods to be used in practical applications involving Lagrangian hydrodynamics.

4.5. *Energy constrained optimization*

A potential issue with the above results (with regard to the energy field) is that the optimized solution may be infeasible for multiple reasons. Two factors we consider here are (1) we want to operate within a particular energy budget and (2) the initial energies cannot be negative locally. The first condition follows from the fact that we do not want to produce solutions which require arbitrary large or small amounts of energy in order to produce. In practical problems, such as laser drives, this is represented by a maximal laser intensity allowed in the specifications.

The second condition is a consequence of physical principals of thermodynamics. Specifically, it is impossible to create a system with negative absolute temperature anywhere. Thus, we want all our designs to remain in the strictly positive regime.

In mathematical terms, we have two separate conditions we would like to satisfy:

$$\int_{\Omega|_{t=0}} e d\Omega = C, \tag{54}$$

$$e(X, t = 0) \geq 0\,. \tag{55}$$

The first enforces that the total energy remains constant in the optimization procedure and the second that the energy remains in the feasible regime. This is a limitation, not of the optimization procedure, but of the formulation of the physics we are trying to model. As a result, incorporating both of these constraints into an optimization procedure is necessary to produce both feasible and practical results.

4.5.1. *Equality constraint*

Consider a constraint given by the equation

$$g(y) = 0\,. \tag{56}$$

We require that a perturbation δy also satisfies the constraint, i.e.,

$$g(y + \delta y) = 0\,.$$

Assuming small perturbations, we can Taylor expand the above equation to get

$$\frac{\partial g}{\partial y} \cdot \delta y = g_y \cdot \delta y = 0\,. \tag{57}$$

As a result, we can define a new perturbation $\delta\tilde{y}$ such that

$$\delta\tilde{y} = \left(I - \frac{1}{\left|g_y\right|^2} g_y \otimes g_y \right) \delta y\,. \tag{58}$$

For the specific case where we want to keep the total internal energy constant, we have the constraint

$$g(\mathbf{e}) = \int_\Omega e d\Omega \tag{59}$$

Taking the derivative as above, we have

$$\frac{\partial g}{\partial e_i} = \int_\Omega \frac{\partial e}{\partial e_i} d\Omega = \int_\Omega \phi_i d\Omega\,. \tag{60}$$

Conveniently, this derivative is constant with respect to the state variables, owing to the linearity of the energy conservation constraint. As a result, the application of the projection method described above will satisfy the constraint exactly. For shorthand, we will refer to the constraint projection as $\delta\hat{e} = P(\delta e)$.

4.5.2. *Inequality constraint*

The second constraint we want to satisfy is that the initial energy is locally non-negative. Because we are considering functions $e \in L^2(\Omega, t)$ with positive Bernstein polynomials, we can take advantage of the property that values of e within the element will be extremal at node values [54]. Therefore, it is sufficient to apply the constraint on the discretization e_i. The rectification of values is not a unique process; however, we choose the simplest as

$$\delta\hat{e} = \mathrm{ReLU}(e + \delta e) - e\,, \tag{61}$$

where $\mathrm{ReLU}(x) = \max(0, x)$. This transformation takes a perturbation δe and energy state e as inputs, rectifies the sum of the two, then returns the perturbation such that all values of the resulting energy are non-negative. Note that we can equivalently verify that the updated state maintains energy positivity. For shorthand, we will refer to this transformation as

$$\delta\hat{e} = R(\delta e, e)\,, \tag{62}$$

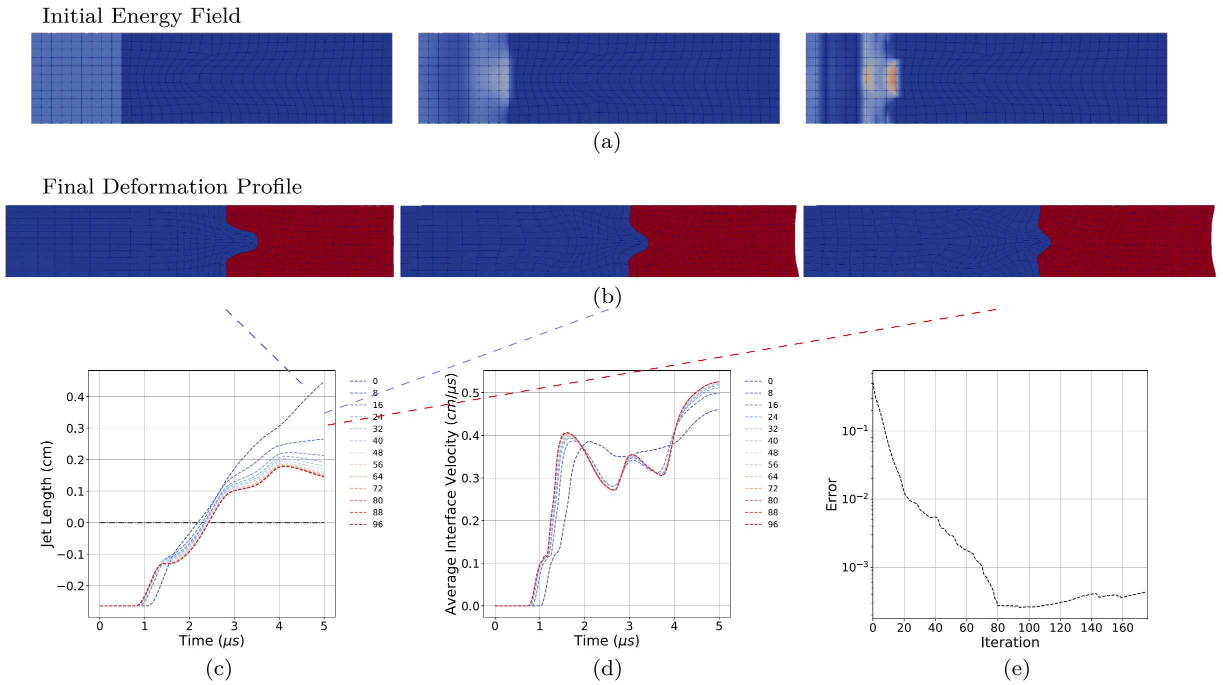


Fig. 10. Demonstration of mitigation of RMI through the gradient descent procedure. (a) The initial energy profile at three different stages (initial guess, 8 steps, 88 steps). (b) The deformation profile at the final timestep for each of those same three stages. (c) The evolution of the jet length over time for different iterations of gradient descent. (d) The average interface velocity over time for different iterations. (e) The change in objective function for each iteration.

4.5.3. *Combination*

There are many ways to satisfy the above two constraints in an optimization paradigm. The standard is to define Lagrange multipliers for both constraints, then evaluate the KKT conditions to ensure feasibility of the perturbations. We take a different approach. Because the dimensionality of the inequality constraint can be quite high, we choose to modify the perturbations to ensure that the constraints remain satisfied. We use the following algorithm to enforce the constraint

```
1. Solve the adjoint problem to obtain an initial $\delta e$
2. Set a *tolerance* value for constraint violation
3. While *error* is greater than *tolerance*
   (a) Solve $\delta e \leftarrow R(\delta e, e)$
   (b) Solve $\delta e \leftarrow P(\delta e)$
   (c) Evaluate infeasibility $error = E(e + \delta e)$
```

where we use an infeasibility error $E(x) = \max(\text{ReLU}(-x))$ which is effectively a L^∞ norm on the error. Additionally, because the projection operator P is always applied second, we ensure that the resulting perturbation exactly satisfies energy conservation. We note that the projection step may break the positivity constraint and thus we need to verify that the infeasibility error is sufficiently small.

4.5.4. *Results*

We repeat the optimization procedure described in Section 4.4 with the additional energy constraints described in Section 4.5.3.

It can be seen in Fig. 10 that the addition of energy conservation and non-negativity constraints does indeed change the structure of the optimized solution. In particular, the solution avoids large amounts of energy locally above and below the hot spot in the center. Generally, designing for experiments will often include constraints such as this in order to generate feasible designs. More information about this solution process can be seen in Fig. A.11–A.13.

5. Conclusion

In this article, we have developed a computational strategy for efficiently differentiating the predicted outputs of a high-order finite element Lagrangian hydrodynamics code with respect to its inputs via a combination of adjoint theory and automatic differentiation; further, we have applied this to a challenging problem of interface stability. This involved individually differentiating the time-stepping update, the (partially-)assembled force vector, and zero-dimensional physics (quadrature point update). The above demonstrations required various simultaneous developments (i) an efficient checkpointing algorithm and (ii) efficient vector-matrix product strategy for the time updates. (iii) We needed a proper regularization of the artificial viscosity which provides computational regularity for studying shocks – discontinuities in the thermodynamic state of the material - which gives this area of physics challenging character. In order to efficiently represent the derivative of the finite element force vector, we have leveraged (iv) partial assembly to compute the action of the resulting operator; this enables efficient computation of vector-matrix products in the time-stepping and optimization. Finally, we have utilized an efficient implementation of automatic differentiation at the quadrature point level to account for the extensive complexity of material function calls; this allows us to use the extreme human-time efficiency in terms of functional complexity of AD while simultaneously retaining computational efficiency of adjoint methods.

We have applied this efficient computation of gradients to optimization of the historically challenging problem in hydrodynamics of stable shock-acceleration of density interfaces; this application is critically important to major scientific challenges in fusion energy experiments such as those conducted at the national ignition facility. In these applications, the process of confining the fusion fuel to ignition conditions invariably involves launching multiple shock waves via a laser energy source which pass through density interfaces (typically diamond - DT). We show that for a prototype problem of this phenomenon called RMI, the gradient based optimization rapidly tailors a complex spatially dependent high dimensional energy drive which simultaneously suppresses the instability and accelerates the interface to a higher velocity than the baseline case. In addition, we apply various constraints to our solutions with the optimization procedure in order to more accurately represent both physical and practical limitations of experimental methods. This simultaneous achievement is remarkable and demonstrates the value of bringing gradient based optimization to bear on problems involving computational hydrodynamics.

We close with some speculation on the future of this field; we advocate that there are many important tasks to do. First, we have shown this for Lagrangian Hydrodynamics; state-of-the-art codes utilize arbitrary Lagrangian-Eulerian (ALE) strategies to manage the computational mesh via the introduction of a remap step which moves the materials appropriately to a new mesh. The absence of such a strategy tends to lead to mesh tangling. A key near-term topic of research is the differentiation of this step. This is made challenging since formally, one must differentiate the remap, the re-mesh, and consider the multi-material case which involves discontinuous material interfaces. Second, hydrodynamics oftentimes does not operate alone. In general, we must track many additional state variables associated with, for instance, phase transitions, material strength, or other multiphysics. Tracking the dependencies in the resulting differentiation calculation in a robust and extensible manner is challenging. Third, we believe that there are many interesting problems in hydrodynamics that should be considered, for instance, the authors intend to eventually conduct realistic simulations of NIF laser driven shots aiming to optimize capsule shape to account for well known and persistent laser drive asymmetries. Finally, there is ample opportunity to apply these techniques to machine learning and uncertainty quantification. A cheap gradient evaluation, as we have provided in this article, can be utilized by Sobolev learning strategies [55] and Hamiltonian Monte Carlo [56].

CRediT authorship contribution statement

Kevin Korner: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Brandon Talamini:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization; **Julian Andrej:** Writing – review & editing, Writing – original draft, Software, Investigation, Formal analysis, Conceptualization; **Michael Tupek:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization; **William Moses:** Software, Conceptualization; **Daniel Tortorelli:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization; **Robert Rieben:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization; **Tzanio Kolev:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization; **Jamie Bramwell:** Writing – review & editing, Methodology, Conceptualization; **Daniel White:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization; **Jonathan Belof:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization; **William Schill:** Writing – review & editing, Writing – original draft, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Data availability

The authors do not have permission to share data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. This work was supported by the LLNL-LDRD Program under Project no. 21-SI-006, Project no. 24-ERD-005, and Project no. 26-SI-002. The Lawrence Livermore National Security journal number is LLNL-JRNL-872441. We thank Libby Glascoe, John Edwards, and Teresa Bailey for their outstanding programmatic leadership and advocacy for this topic. We thank Sam Mish and Robert Carson for numerous interesting discussions.

Appendix A. Supplemental information

A.1. Scaling with derivatives

Consider a dth order polynomial in n dimensions. This will have $C(d+n,d)$ coefficients associated with it. In order to fully define the coefficients from an unknown function, we will then require $C(d+n,d)$ individual function evaluations. In order to consider the cost benefit of calculating derivatives, we compare the cost of fully defining the polynomial if, every time we do a forward solve, we also calculate the gradient at that point. The forward solve gives one equation while the derivative gives n additional equations. If we were to match a 2nd order polynomial in 10 dimensions, we would require 66 forward solves. With gradients we would require 6 forward solves and 6 gradient solves. In 100 dimensions, this becomes 5151 forward solves but including gradient calculations requires 51 forward and gradient. We calculate the number of forward and gradient solves necessary to fully solve the system by solving when $N(1+n) = C(d+n,d)$. We then find the ratio of number of solves necessary for just the forward solve vs number of forward + gradient solves with

$$\text{Cost Ratio} = \frac{C(d+n,d)}{2\,N} = \frac{1+n}{2}\,.$$

As a result we see that the cost of evaluating with just forward solves scales linearly with the number of dimensions we are solving in.

There is additional benefit that the gradient solved returns the steepest descent direction which is particularly useful in optimization when you are not trying to fully characterize the functional form.

A.2. Infdim math

$$O_N(\beta) = \sum_{k=1}^{N} o_k(y_k,\beta,\Delta t) + \sum_{k=1}^{N} \lambda_k^T (y_k - f_k(y_{k-1},\beta,\Delta t)) \tag{A.1}$$

Taking the derivative with respect to β we have

$$\frac{\partial O_N}{\partial \beta} = \sum_{k=1}^{N} \left(\frac{\partial o_k}{\partial y_k}\frac{\partial y_k}{\partial \beta} + \frac{\partial o_k}{\partial \beta} \right) + \sum_{k=1}^{N} \lambda_k^T \left(\frac{\partial y_k}{\partial \beta} - \frac{\partial f_k}{\partial y_{k-1}}\frac{\partial y_{k-1}}{\partial \beta} - \frac{\partial f_k}{\partial \beta} \right) \tag{A.2}$$

Adjusting indices in the second term gives

$$\frac{\partial O_N}{\partial \beta} = \sum_{k=1}^{N} \left(\frac{\partial o_k}{\partial y_k}\frac{\partial y_k}{\partial \beta} + \frac{\partial o_k}{\partial \beta} \right) + \sum_{k=1}^{N} \lambda_k^T \left(\frac{\partial y_k}{\partial \beta} - \frac{\partial f_k}{\partial \beta} \right) - \sum_{k=0}^{N-1} \lambda_{k+1}^T \frac{\partial f_{k+1}}{\partial y_k}\frac{\partial y_k}{\partial \beta} \tag{A.3}$$

Expanding out the sums to combine the internal summation,

$$\frac{\partial O_N}{\partial \beta} = \frac{\partial o_N}{\partial y_N}\frac{\partial y_N}{\partial \beta} + \frac{\partial o_N}{\partial \beta} + \lambda_N^T \left(\frac{\partial y_N}{\partial \beta} - \frac{\partial f_N}{\partial \beta} \right) + \sum_{k=1}^{N-1} \left(\frac{\partial o_k}{\partial y_k}\frac{\partial y_k}{\partial \beta} + \frac{\partial o_k}{\partial \beta} + \lambda_k^T \left(\frac{\partial y_k}{\partial \beta} - \frac{\partial f_k}{\partial \beta} \right) - \lambda_{k+1}^T \frac{\partial f_{k+1}}{\partial y_k}\frac{\partial y_k}{\partial \beta} \right) - \lambda_1^T \frac{\partial f_1}{\partial y_0}\frac{\partial y_0}{\partial \beta} \tag{A.4}$$

Grouping terms that contain $\partial y_k/\partial \beta$, we have

$$\frac{\partial O_N}{\partial \beta} = \left(\frac{\partial o_N}{\partial y_N} + \lambda_N^T \right)\frac{\partial y_N}{\partial \beta} + \sum_{k=1}^{N-1} \left(\frac{\partial o_k}{\partial y_k} + \lambda_k^T - \lambda_{k+1}^T \frac{\partial f_{k+1}}{\partial y} \right)\frac{\partial y_k}{\partial \beta} + \sum_{k=1}^{N} \left(\frac{\partial o_k}{\partial \beta} - \lambda_k^T \frac{\partial f_k}{\partial \beta} \right) - \lambda_1^T \frac{\partial f_1}{\partial y_0}\frac{\partial y_0}{\partial \beta} \tag{A.5}$$

A.3. Continuous analogues of non-differentiable functions

In some cases we smooth non-differentiable or discontinuous functions. Some functions used are

$$\text{sigmoid}(x) = \frac{1}{1+\exp(-x)} \tag{A.6}$$

$$\text{silu}(x) = x * \text{sigmoid}(x) \tag{A.7}$$

A.4. Adjoints of generic finite element systems

Consider a generic forcing function of the form

$$F_i = \int_\Omega \big(f(x, y(x), \nabla y(x)) \cdot \phi_i(x) + f'(x, y(x), \nabla y(x)) \cdot \nabla \phi_i(x) \big) dx \tag{A.8}$$

Given that we have terms which involve functions y, ϕ and their derivatives ∇y, $\nabla \phi$, we ensure we are working in an appropriate function space, such as H^1. If we consider the inner product of this term and some arbitrary adjoint variable, we have

$$M = \sum_i \lambda_i F_i = \sum_i \lambda_i \int_\Omega \big(f(x, y(x), \nabla y(x) \cdot \phi_i(x) + f'(x, y(x, \nabla y(x)) \cdot \nabla \phi_i(x))) \big) dx \tag{A.9}$$

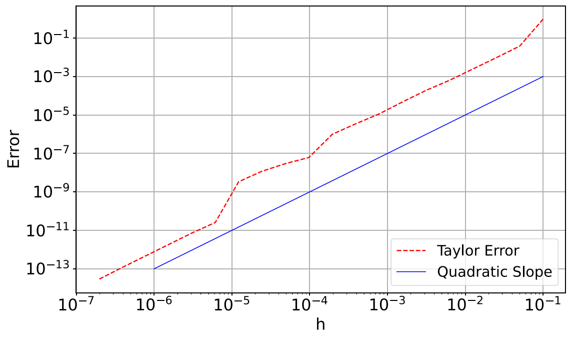


Fig. A.11. Taylor test of Lagrangian hydrodynamics.

$$= \int_\Omega \big(f(x, y(x), \nabla y(x)) \cdot \lambda(x) + f'(x, y(x), \nabla y(x)) \cdot \nabla\lambda(x)\big)dx\,. \tag{A.10}$$

Combining the integrand into a single variable as $m(x, y, \nabla y, \lambda, \nabla\lambda) = f(x, y, \nabla y) \cdot \lambda + f'(x, y, \nabla y) \cdot \nabla\lambda$, we write this integral in the simplest form

$$M = \sum_i \lambda_i F_i = \int_\Omega m(x, y(x), \nabla y(x), \lambda(x), \nabla\lambda(x))dx \tag{A.11}$$

If we now take the derivative with respect to the primal discretized field variable y_i, we have

$$\frac{\partial}{\partial y_i}\left(\sum_k \xi_k F_k\right) = \sum_k \xi_k \frac{\partial F_k}{\partial y_i} = \int_\Omega \left(\frac{\partial m}{\partial y} \cdot \phi_i(x) + \frac{\partial m}{\partial \nabla y} \cdot \nabla\phi_i(x)\right)dx \tag{A.12}$$

where we use that $y(x) = \sum_i y_i\phi_i(x)$ as the standard Galerkin expansion. As mentioned in the main text, this form has several advantages.

1. The vector-Jacobian product involved in the adjoint calculation becomes matrix free,
2. The calculation of m is local and parallelizable to calculations at quadrature points,
3. The calculations of the derivatives $\frac{\partial m}{\partial y}$ and $\frac{\partial m}{\partial \nabla y}$ are entirely parallelized and utilize the same force calculation structure present in all finite element codes.

This allows us to construct computationally efficient adjoint products which are necessary in computationally intensive problems such as hydrocodes.

A.5. Taylor test of hydrodynamic system

We verify the consistency of gradients by conducting a Taylor test on the Lagrangian hydrodynamic system.

The process is outlined in Section 2.4.1. We define a functional form $f(v, y) = v \cdot \dot{y}$ where v is a random vector and y is the current state (x, v, e). The state is then perturbed using $y = y_0 + h\delta y$ for various h and δy is another random vector which conforms to boundary conditions.

A.6. Mesh convergence

We verify the convergence with respect to the mesh by visualizing the adjoint field.

A.7. Lower energy results

We also run the results with the initial energy initialized to $e(\Omega_1, 0) = 0.5$.

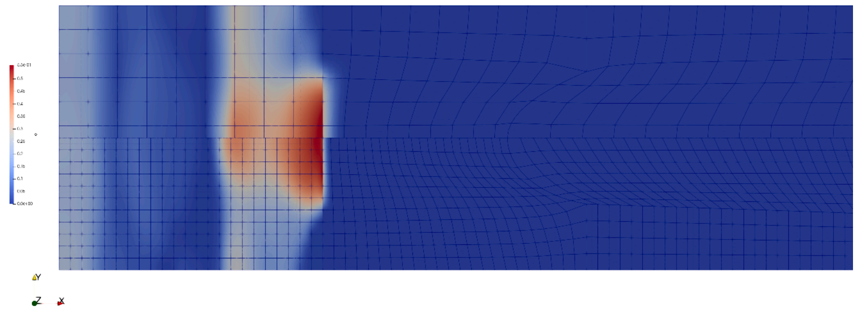


Fig. A.12. Solution comparison for different meshes.

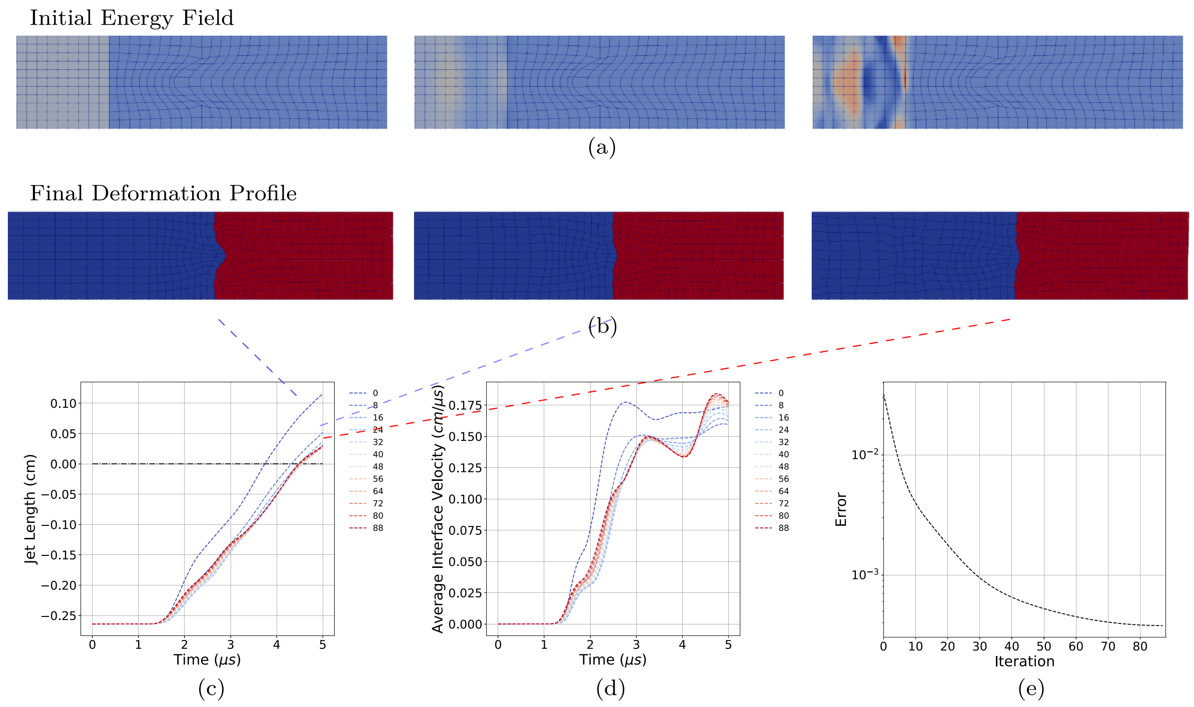


Fig. A.13. Demonstration of mitigation of RMI through the gradient descent procedure. (a) The initial energy profile at three different stages (initial guess, 8 steps, 88 steps). (b) The deformation profile at the final timestep for each of those same three stages. (c) The evolution of the jet length over time for different iterations of gradient descent. (d) The average interface velocity over time for different iterations. (e) The change in objective function for each iteration.

References

[1] R.D. Richtmyer, Taylor instability in shock acceleration of compressible fluids, Commun. Pure Appl. Math. 13 (2) (1960) 297–319. https://doi.org/10.1002/cpa.3160130207
[2] G.I. Taylor, The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. I, Proceed. Roy. Soc. Lond. Ser. A. Math. Phys. Sci. 201 (1065) (1997) 192–196. https://doi.org/10.1098/rspa.1950.0052
[3] Y. Zhou, Rayleigh–Taylor and Richtmyer–Meshkov instability induced flow, turbulence, and mixing. I, Phys. Rep. 720–722 (2017) 1–136. https://doi.org/10.1016/j.physrep.2017.07.005
[4] Y. Zhou, Rayleigh–Taylor and Richtmyer–Meshkov instability induced flow, turbulence, and mixing. II, Phys. Rep. 723–725 (2017) 1–160. https://doi.org/10.1016/j.physrep.2017.07.008
[5] Y. Zhou, R.J.R. Williams, P. Ramaprabhu, M. Groom, B. Thornber, A. Hillier, W. Mostert, B. Rollin, S. Balachandar, P.D. Powell, A. Mahalov, N. Attal, et al., Rayleigh–Taylor and Richtmyer–Meshkov instabilities: a journey through scales, Phys. D 423 (2021) 132838. https://doi.org/10.1016/j.physd.2020.132838
[6] G. Birkhoff, D.P. MacDougall, E.M. Pugh, S.G. Taylor, et al., Explosives with lined cavities, J. Appl. Phys. 19 (6) (1948) 563–582. https://doi.org/10.1063/1.1698173

[7] Y. Zhou, T.T. Clark, D.S. Clark, S. Gail Glendinning, M. Aaron Skinner, C.M. Huntington, O.A. Hurricane, A.M. Dimits, B.A. Remington, Turbulent mixing and transition criteria of flows induced by hydrodynamic instabilities, Phys. Plasma. 26 (8) (2019) 080901. https://doi.org/10.1063/1.5088745
[8] K.O. Mikaelian, Extended model for Richtmyer–Meshkov mix, Phys. D 240 (11) (2011) 935–942. https://doi.org/10.1016/j.physd.2011.01.008
[9] B.A. Remington, H.-S. Park, D.T. Casey, R.M. Cavallo, D.S. Clark, C.M. Huntington, D.H. Kalantar, C.C. Kuranz, A.R. Miles, S.R. Nagel, K.S. Raman, C.E. Wehrenberg, V.A. Smalyuk, Rayleigh–Taylor instabilities in high-energy density settings on the national ignition facility, Proceed. Natl. Acad. Sci. 116 (37) (2019) 18233–18238. https://doi.org/10.1073/pnas.1717236115
[10] D.M. Sterbentz, C.F. Jekel, D.A. White, S. Aubry, H.E. Lorenzana, J.L. Belof, et al., Design optimization for Richtmyer–Meshkov instability suppression at shock-compressed material interfaces, Phys. Fluid. 34 (8) (2022) 082109. https://doi.org/10.1063/5.0100100
[11] D.J. Kline, M.P. Hennessey, D.K. Amondson, S. Lin, M.D. Grapes, M. Ferrucci, P. Li, H.K. Springer, R.V. Reeves, K.T. Sullivan, J.L. Belof, et al., Reducing Richtmyer–Meshkov instability jet velocity via inverse design, J. Appl. Phys. 135 (7) (2024) 074902. https://doi.org/10.1063/5.0180712
[12] D.M. Sterbentz, D.J. Kline, D.A. White, C.F. Jekel, M.P. Hennessey, D.K. Amondson, A.J. Wilson, M.J. Sevcik, M.F.L. Villena, S.S. Lin, M.D. Grapes, K.T. Sullivan, J.L. Belof, et al., Explosively driven Richtmyer–Meshkov instability jet suppression and enhancement via coupling machine learning and additive manufacturing, J. Appl. Phys. 136 (3) (2024) 035102. https://doi.org/10.1063/5.0213123
[13] C.F. Jekel, D.M. Sterbentz, T.M. Stitt, P. Mocz, R.N. Rieben, D.A. White, J.L. Belof, et al., Machine learning visualization tool for exploring parameterized hydrodynamics*, Mach. Learn.: Sci. Technol. 5 (4) (2024) 045048. https://doi.org/10.1088/2632-2153/ad8daa
[14] D.M. Sterbentz, C.F. Jekel, D.A. White, R.N. Rieben, J.L. Belof, et al., Linear shaped-charge jet optimization using machine learning methods, J. Appl. Phys. 134 (4) (2023) 045102. https://doi.org/10.1063/5.0156373
[15] W.J. Schill, M.R. Armstrong, J.H. Nguyen, D.M. Sterbentz, D.A. White, L.X. Benedict, R.N. Rieben, A. Hoff, H.E. Lorenzana, J.L. Belof, B.M. La Lone, M.D. Staska, Suppression of Richtmyer-Meshkov instability via special pairs of shocks and phase transitions, Phys. Rev. Lett. 132 (2) (2024) 024001. https://doi.org/10.1103/PhysRevLett.132.024001
[16] W.J. Schill, R.A. Austin, K.L. Schimdt, J.L. Brown, N.R. Barton, et al., Simultaneous inference of the compressibility and inelastic response of tantalum under extreme loading, J. Appl. Phys. 130 (5) (2021) 055901. https://doi.org/10.1063/5.0056437
[17] W.J. Schill, K.L. Schmidt, R.A. Austin, W.W. Anderson, J.L. Belof, J.L. Brown, N.R. Barton, Inference of strength and phase transition kinetics in dynamically-compressed tin, J. Appl. Phys. 133 (24) (2023) 245903. https://doi.org/10.1063/5.0150749
[18] M.G. Gorman, C.J. Wu, R.F. Smith, L.X. Benedict, C.J. Prisbrey, W. Schill, S.A. Bonev, Z.C. Long, P. S. öderlind, D. Braun, D.C. Swift, R. Briggs, T.J. Volz, E.F. O'Bannon, P.M. Celliers, D.E. Fratanduono, J.H. Eggert, S.J. Ali, J.M. McNaney, et al., Ramp compression of tantalum to multiterapascal pressures: constraints of the thermal equation of state to 2.3 TPa and 5000 K, Phys. Rev. B 107 (1) (2023) 014109. https://doi.org/10.1103/PhysRevB.107.014109
[19] R. Hottois, A. Châtel, G. Coussement, T. Debruyn, T. Verstraete, Comparing gradient-free and gradient-based multi-objective optimization methodologies on the VKI-LS89 turbine vane test case, J. Turbomach. 145 (3) (2022) 031001. https://doi.org/10.1115/1.4055577
[20] A. Akerson, B. Bourdin, K. Bhattacharya, et al., Optimal design of responsive structures, Struct. Multidiscip. Optim. 65 (4) (2022) 111. https://doi.org/10.1007/s00158-022-03200-5
[21] A. Akerson, Optimal structures for failure resistance under impact, J. Mech. Phys. Solid. 172 (2023) 105172. https://doi.org/10.1016/j.jmps.2022.105172
[22] M.P. Bendsøe, O. Sigmund, Topology Optimization, Springer, Berlin, Heidelberg, 2004. https://doi.org/10.1007/978-3-662-05086-6
[23] R.E. Plessix, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications, Geophys. J. Int. 167 (2) (2006) 495–503. https://doi.org/10.1111/j.1365-246X.2006.02978.x
[24] H. Zhang, A. Sandu, FATODE: A library for forward, adjoint, and tangent linear integration of ODEs, SIAM J. Sci. Comput. 36 (5) (2014) C504–C523. https://doi.org/10.1137/130912335
[25] H. Zhang, S. Abhyankar, E. Constantinescu, M. Anitescu, et al., Discrete adjoint sensitivity analysis of hybrid dynamical systems with switching, IEEE Trans. Circuits Syst. I Regul. Pap. 64 (5) (2017) 1247–1259. https://doi.org/10.1109/TCSI.2017.2651683
[26] H. Zhang, E.M. Constantinescu, B.F. Smith, et al., PETSc TSAdjoint: A discrete adjoint ODE solver for first-order and second-order sensitivity analysis, SIAM J. Sci. Comput. 44 (1) (2022) C1–C24. https://doi.org/10.1137/21M140078X
[27] B. Talamini, M. Tupek, OptimiSM v.0.0.1, 2021, https://doi.org/10.11578/dc.20221005.6
[28] S. Carli, L. Hascoët, W. Dekeyser, M. Blommaert, et al., Algorithmic differentiation for adjoint sensitivity calculation in plasma edge codes, J. Comput. Phys. 491 (2023) 112403. https://doi.org/10.1016/j.jcp.2023.112403
[29] J.S. Jensen, P.B. Nakshatrala, D.A. Tortorelli, et al., On the consistency of adjoint sensitivity analysis for structural optimization of linear dynamic problems, Struct. Multidiscip. Optim. 49 (5) (2014) 831–837. https://doi.org/10.1007/s00158-013-1024-4
[30] A. Sandu, On the properties of Runge-Kutta discrete adjoints, in: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 550–557.
[31] A. Sandu, On Consistency Properties of Discrete Adjoint Linear Multistep Methods, Technical Report TR-07-40, 2007.
[32] J.M. Sanz-Serna, Symplectic Runge–Kutta schemes for adjoint equations, automatic differentiation, optimal control, and more, SIAM Rev. 58 (1) (2016) 3–33. https://doi.org/10.1137/151002769
[33] B.K. Tran, B.S. Southworth, M. Leok, On properties of adjoint systems for evolutionary PDEs 34 (5) 95. https://doi.org/10.1007/s00332-024-10071-1
[34] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch (2017).
[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. https://www.tensorflow.org/.
[36] 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.
[37] W.S. Moses, V. Churavy, Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020, pp. 12472–12485.
[38] W.S. Moses, V. Churavy, L. Paehler, J. Hückelheim, S.H.K. Narayanan, M. Schanen, J. Doerfert, et al., Reverse-mode automatic differentiation and optimization of GPU kernels via enzyme, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 1–16. https://doi.org/10.1145/3458817.3476165
[39] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, S. Zampini, et al., MFEM: A modular finite element methods library, Comput. Math. Applic. 81 (2021) 42–74. https://doi.org/10.1016/j.camwa.2020.06.009
[40] J. Andrej, N. Atallah, J.-P. Bäcker, J.-S. Camier, D. Copeland, V. Dobrev, Y. Dudouit, T. Duswald, B. Keith, D. Kim, T. Kolev, B. Lazarov, K. Mittal, W. Pazner, S. Petrides, S. Shiraiwa, M. Stowell, V. Tomov, et al., High-performance finite elements with MFEM, Int. J. High Perform. Comput. Appl. 38 (5) (2024) 447–467. https://doi.org/10.1177/10943420241261981
[41] A. Griewank, A. Walther, Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation, ACM Trans. Math. Softw. 26 (1) (2000) 19–45. https://doi.org/10.1145/347837.347846
[42] Q. Wang, P. Moin, G. Iaccarino, et al., Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation, SIAM J. Sci. Comput. 31 (4) (2009) 2549–2567. https://doi.org/10.1137/080727890
[43] J. Herrmann, G.P. (Aupy), H-Revolve: a framework for adjoint computation on synchronous hierarchical platforms, ACM Trans. Math. Softw. 46 (2) (2020) 12:1–12:25. https://doi.org/10.1145/3378672
[44] O. Sigmund, On the design of compliant mechanisms using topology optimization* 25 (4) 493–524. Publisher: Taylor & Francis, https://doi.org/10.1080/08905459708945415
[45] B. Bourdin, Filters in topology optimization, Int. J. Numer. Method. Eng. 50 (9) (2001) 2143–2158. https://doi.org/10.1002/nme.116

[46] O. Sigmund, Morphology-based black and white filters for topology optimization 33 (4) 401–424. https://doi.org/10.1007/s00158-006-0087-x
[47] M. Zhou, B.S. Lazarov, F. Wang, O. Sigmund, Minimum length scale in topology optimization by geometric constraints 293 266–282. https://www.sciencedirect.com/science/article/pii/S0045782515001693. https://doi.org/10.1016/j.cma.2015.05.003
[48] V.A. Dobrev, T.V. Kolev, R.N. Rieben, et al., High-order curvilinear finite element methods for Lagrangian hydrodynamics, SIAM J. Sci. Comput. 34 (LLNL-JRNL-516394) (2012). https://doi.org/10.1137/120864672
[49] F.H. Harlow, A.A. Amsden, Fluid Dynamics. A Lasl Monograph, Technical Report LA4700, Los alamos Scientific Lab., N. Mex., 1971.
[50] A. Lew, R. Radovitzky, M. Ortiz, et al., An artificial-viscosity method for the Lagrangian analysis of shocks in solids with strength on unstructured, arbitrary-order tetrahedral meshes, J. Comput.-Aid. Mater. Des. 8 (2) (2001) 213–231. https://doi.org/10.1023/A:1020064403005
[51] J.C. Campbell, M.J. Shashkov, A tensor artificial viscosity using a mimetic finite difference algorithm, J. Comput. Phys. 172 (2) (2001) 739–765. https://doi.org/10.1006/jcph.2001.6856
[52] M.L. Wilkins, Use of artificial viscosity in multidimensional fluid dynamic calculations, J. Comput. Phys. 36 (3) (1980) 281–303. https://doi.org/10.1016/0021-9991(80)90161-8
[53] P.D. Bello-Maldonado, T.V. Kolev, R.N. Rieben, V.Z. Tomov, et al., A matrix-free hyperviscosity formulation for high-order ALE hydrodynamics, Comput. Fluid. 205 (2020) 104577. https://doi.org/10.1016/j.compfluid.2020.104577
[54] R.W. Anderson, V.A. Dobrev, T.V. Kolev, R.N. Rieben, V.Z. Tomov, et al., High-order multi-material ALE hydrodynamics, SIAM J. Sci. Comput. 40 (1) (2018) B32–B58. https://doi.org/10.1137/17M1116453
[55] W.M. Czarnecki, S. Osindero, M. Jaderberg, G. Świrszcz, R. Pascanu, et al., Sobolev Training for Neural Networks, 2017, https://doi.org/10.48550/arXiv.1706.04859
[56] M.D. Homan, A. Gelman, The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo, J. Mach. Learn. Res. 15 (1) (2014) 1593–1623.