Abstract
Plasma supports collective modes and particle-wave interactions that leads to complex behavior in inertial fusion energy applications. While plasma can sometimes be modeled as a charged fluid, a kinetic description is useful towards the study of nonlinear effects in the higher dimensional momentum-position phase-space that describes the full complexity of plasma dynamics. We create a differentiable solver for the plasma kinetics 3D partial-differential-equation and introduce a domain-specific objective function. Using this framework, we perform gradient-based optimization of neural networks that provide forcing function parameters to the differentiable solver given a set of initial conditions. We apply this to an inertial-fusion relevant configuration and find that the optimization process exploits a novel physical effect that has previously remained undiscovered.

1. Introduction

1.1. Fusion, Charged Fluid Dynamics, and Kinetics
Thermonuclear fusion involves conditions where matter is best represented as a warm plasma where the ions and electrons form a charged particle soup. Such plasmas can often be described using a combination of Maxwell’s equations and the Navier-Stokes equations. However, many non-linear effects in plasmas cannot be modeled using the fluid description without further modifications.

These effects are termed kinetic effects. To model kinetic effects, a more complete description is required. This can be accomplished by either retaining the velocities of individual particles of the fluid or retaining a statistical representation of the particles. In either case, the objective becomes

to model the phase space dynamics rather than a version integrated in velocity.

Phase space dynamics for plasmas are given by formulations or realizations of the non-linear transport equation. A common simplification that is valid for fast time-scales, is to assume the ions, and their phase space, remains stationary. Then, the transport equation for electrons is given by

\[
\frac{\partial f}{\partial t} + v \frac{\partial f}{\partial x} - E \frac{\partial f}{\partial v} = \frac{\delta f}{\delta t} \tag{1}
\]

where \( E = \int f \, dv - 1 \), and \( f = f(t, x, v) \). Equation 1, along with Gauss’s Law, is often termed the Vlasov-Poisson-Fokker-Planck (VPFP) equation set. Solving the VPFP set is often analytically intractable, even in one spatial and one velocity dimension (1D-1V). This is because the left-hand-side has a stiff linear transport term, has a non-linear term in \( E \partial f / \partial v \), and can sustain wave propagation and other hyperbolic partial-differential-equation (PDE) behavior. Additionally, the right hand side is typically represented by a hyperbolic, advection-diffusion, partial-differential-equation. Making progress on kinetic plasma physics requires computational simulation tools.

Numerical solutions to the 1D-1V VPFP equation set have been applied in research on laser-plasma interactions in the context of inertial fusion plasma-based accelerators (Thomas, 2016), space physics (Chen et al., 2019), fundamental plasma physics (Pezzi et al., 2019), and inertial fusion (Strozzi et al., 2007; Fahlen et al., 2009; Banks et al., 2011)

In this work, we extend previous findings (Fahlen et al., 2009) in the context of inertial fusion using a gradient-based approach towards discovery of new physics.

1.2. Gradient-Based Optimization in Plasma Physics
There have been several recent applications of gradient-descent towards fusion plasma physics. Analytic approaches have resulted in the development of adjoint methods for shape derivatives of functions that depend on magnetohydrodynamics (MHD) equilibria (Antonsen et al., 2019; Paul et al., 2020). These methods have been used to perform optimization of stellarator design (Paul et al., 2021). In
2. Parameter and Function Learning using Differentiable Simulations

In this section, we provide a step-by-step description of how a traditional simulation-based computational physics workflow may be modified to perform closed-loop optimization as applied to parameter and function learning.

2.1. Open Loop - Manual Workflow

Figure 2a depicts a typical workflow of a computational scientist represented as a cyclic graph. The scientist defines the parametric inputs which create the state vector \( x \). This can contain any parameters that are used to define the simulation e.g. the grid size, the number of solver steps, etc. For didactic purposes, the physical parameters to the simulation may be separated into a different vector of inputs \( p_d \) e.g. the forcing function parameters, the viscosity coefficient etc.

In the work here, we train a neural network that provides control parameters to the PDE solver. We choose physical parameters as inputs and control parameters as outputs of the neural network. This enables the neural network to learn a function that describes the physical relationship between the plasma parameters and the forcing function parameters e.g. the resonance frequency.

We train the neural network in an unsupervised fashion using a cost function based on the maximum entropy principle. This enables us to create self-learning plasma physics simulations, where the optimization process provides a physically interpretable function that can enable physics discovery.

2.2. Closed Loop - Brute Force Parameter Scan

Figure 2b shows a more automated workflow. We replace the gray-box postprocessing step with the calculation of a scalar quantity \( S \) using a Cost Function \( C \) on the final state \( x_f \). This reduces the complexity of the interpretation of the postprocessing and enables a more rapid search in parameter space. The decrease in required human effort for completing other work, by using gradients obtained from analytic (Zhu et al., 2017) and automatic differentiation (AD) (McGreivy et al., 2021), the FOCUS and FOCUSADD codes optimize coil shape. These advances are founded on the concept of performing a sensitivity analysis towards device design.

This work applies AD towards learning physical relationships and discovering novel phenomena in the VPFP dynamical system. We do this by training neural networks through differentiable simulations that solve the 1D-1V version of eq. 1 and optimize for a custom objective function.

1.3. Applications of Differentiable Physical Simulations

Differentiable physics simulations have been used in a variety of contexts. For example, for learning parameters for molecular dynamics (Schoenholz & Cubuk, 2019), for learning differencing stencils in PDEs (Bar-Sinai et al., 2019; Zhuang et al., 2020; Kochkov et al., 2021), and for controlling PDEs (Holl et al., 2020).

In the work here, we train a neural network that provides control parameters to the PDE solver. We choose physical parameters as inputs and control parameters as outputs of the neural network. This enables the neural network to learn a function that describes the physical relationship between the plasma parameters and the forcing function parameters e.g. the resonance frequency.

We train the neural network in an unsupervised fashion using a cost function based on the maximum entropy principle. This enables us to create self-learning plasma physics simulations, where the optimization process provides a physically interpretable function that can enable physics discovery.
Figure 2. (a) A typical workflow of a computational scientist where the scientist provides the initial conditions and forcing function parameters to a partial differential-equation (PDE) solve. The output of the solve is stored as the final state $x_f$. The final state is analyzed for the resonant frequency, $\omega$, that optimizes for the scalar, $S$, we compute
\[
\frac{\partial S}{\partial \omega} = \frac{\partial S}{\partial \omega} \frac{\partial \omega}{\partial \omega} = \frac{\partial S}{\partial \omega} \cdot \frac{\partial \omega}{\partial \omega}.
\]
Assuming a well-behaved solution manifold, performing gradient-descent tends to reduce the number of iterations required to find the minimum in comparison to a evenly-spaced parameter scan, especially when the required resolution is unknown (Nocedal & Wright, 2006). Put another way, gradient-descent effectively provides an adaptive stepping mechanism in lieu of the pre-defined values that represent a parameter scan.

2.3. Gradient-Descent-Based Parameter Learning

Figure 2c includes two modifications. The scientist/parameter search graybox has been replaced with a gradient-descent-based optimization algorithm. This algorithm provides the updated parameters, e.g. $\omega_G$, a guess for the resonant frequency of the system, for the next iteration of the loop. The gradient-descent algorithm requires the calculation of an accurate gradient. By writing our PDE solver $\mathcal{V}$ and the cost function $\mathcal{C}$ using a numerical framework that supports automatic differentiation, we are able to calculate $\partial S / \partial \omega_G$.

Since
\[
S = \mathcal{C}(x_f) = \mathcal{C}(\mathcal{V}(x, p_d)),
\]
the gradient for the update-step is given by
\[
\frac{\partial S}{\partial p_d} = \frac{\partial \mathcal{C}(\mathcal{V}(x, p_d))}{\partial p_d} = \frac{\partial S}{\partial \mathcal{V}} \frac{\partial \mathcal{V}}{\partial p_d}.
\]
For example, if we wish to learn the resonant frequency, $\omega$, that optimizes for the scalar, $S$, we compute
\[
\frac{\partial S}{\partial \omega} = \frac{\partial S}{\partial \mathcal{V}} \frac{\partial \mathcal{V}}{\partial \omega}.
\]

2.4. Gradient-Descent-Based Function Learning

In the final step, shown in fig. 2d, we can replace the lookup-like capability of the parameter optimization and choose to learn a function that can provide the learned parameters. The advantage to learning a function, however, is the ability to interpolate and extrapolate with respect in the input space.

Here, we choose to use neural networks, with a parameter vector $\theta$, to parameterize the desired function. This allows us to extend the gradient-descent based methodology and...
leverage existing numerical software to implement this differentiable programming loop.

Now,
\[ S = C(x_f) = C(V(x, p_d)) = C(V(x, G(x, p_d; \theta)), \]
where \( G \) is a function that generates the desired forcing function parameter given a parameter vector \( \theta \). To extend the example from sec. 2.3, \( \omega \) is now a function given by \( \omega = G(x, p_d; \theta) \).

We compute the same gradient as in eq. 2 and add a correction factor that arises because the parameter (vector) is now \( \theta \), rather than \( \omega \). The necessary gradient for the gradient update is now given by
\[ \frac{\partial S}{\partial \theta} = \left[ \frac{\partial S}{\partial V} \right] \frac{\partial G(x, p_d; \theta)}{\partial \theta}. \] (3)

3. Physics Discovery as an Optimization Problem

We will start with a basic physical effect that is well understood, Electrostatic Wavepackets in Inertial Fusion Plasmas. Then we will ask the question of “what happens when you excite this system?”

3.1. Electrostatic Wavepackets in Inertial Fusion

When electrostatic waves are driven to large amplitude, electrons can become trapped in the large potential (O’Neil, 1965). Simulations of Stimulated Raman Scattering (SRS) in inertial confinement fusion (ICF) scenarios show that large-amplitude waves of finite extent are generated in the laser-plasma interaction, and that particle trapping is correlated with the transition to the high-reflectivity burst regime of SRS (Strozzi et al., 2007; Ellis et al., 2012).

Simulating wavepackets, similar to those generated in SRS, but in isolation, has illuminated kinetic dynamics where trapped electrons, with velocity \( v_{ph} \), transit the wavepacket which is moving at the slower group velocity \( v_g \). The transit of the trapped electrons from the back of the wavepacket to the front results in the resumption of Landau damping at the back and the wavepacket is then damped away (Fahlen et al., 2009).

Recent work modeled the interaction of multiple speckles with a magnetic field acting as a control parameter. Since the effect of the magnetic field is to rotate the distribution in velocity space, the field strength serves as a parameter by which the authors control scattered particle propagation. Using this, along with carefully placed laser speckles, they show that scattered light and particles can serve as the trigger for SRS (Winjum et al., 2019).

Here, we ask What happens when a non-linear electron plasma wavepacket is driven on top of another?

To answer this question, we reframe it as an optimization problem and ask, What is the best way to excite a wavepacket that interacts with a pre-existing wavepacket?

To solve this optimization problem, we turn to the framework established in the previous section and in fig. 2. To automate this process, we can use the workflow described in fig. 2c or fig. 2d. In both, the physicist is still required to provide their expertise in the form of 1/ the parameterization, and 2/ the objective function.

3.2. Physicist Input 1: Parameterizing the Forcing Function with a Neural Network

We start with a large-amplitude, finite-length electrostatic wavepacket driven by a forcing function with parameters given by
\[ p_0 = [x_0, \omega_0, t_0, k_0], \] (4)
where \( x_i \) is the location of excitation, \( \omega_i \) is the frequency, \( t_i \) is the time of excitation, and \( k_i \) is the wavenumber, of the \( i \)th wavepacket.

Since we seek to excite a second wavepacket that can interact with the detrapped electrons, we set the wavenumber \( k_1 = k_0 \). We reparameterize the resonant frequency, \( \omega_1 \), with a frequency shift, \( \Delta \omega_1 \) and the linear resonant frequency \( \omega_0 \) such that \( \omega_1 = \omega_0 + \Delta \omega_1 \).

We use the time of excitation of the second wavepacket, \( t_1 \), as an independent variable along with \( k_0 \). For each \( t_1 \) and \( k_0 \), we seek to learn functions that produce \( x_1 \) and \( \Delta \omega_1 \) i.e. we seek to learn \( x_1(t_1, k_0) \) and \( \Delta \omega_1(t_1, k_0) \). The entire
parameter vector for the second wavepacket is given by

\[ p_0 = [x_1(t_1, k_0), \Delta \omega_1(t_1, k_0), t_1, k_0]. \]

(5)

This framing is also illustrated in fig. 3 where given \( k_0 \) and \( t_1 \), we seek functions for \( \omega_1 \) and \( x_1 \).

### 3.3. Physicist Input 2: A Minimum Energy and Maximum Entropy Loss Function

We reparameterize \( \Delta \omega_1 \) and \( x_1 \) with a neural network with a parameter vector, \( \theta^* \), that maximizes the electrostatic energy (minimizes the free energy) and maximizes the kinetic entropy i.e.

\[ x_1 = x_1(t_1, k_0; \theta^*), \]

\[ \Delta \omega_1 = \Delta \omega_1(t_1, k_0; \theta^*). \]

(6)\hspace{1cm}(7)

where,

\[ \theta^* = \arg\min \left[ -E_{es}(p; \theta) - \Delta K E(p, \theta) \right], \]

(8)

and

\[ E_{es} = \sum_{t_i} \Delta t \sum_x E_x^2 \]

\[ \Delta K E = \sum_{t_i} \Delta t \sum_x \sum_v \Delta u \left( f \log(f) - f_{MX} \log(f_{MX}) \right), \]

(9)\hspace{1cm}(10)

are the electrostatic energy, and entropy, terms in the loss function, respectively. \( f_{MX} = f_{MX}(n, T), n = \int f dv, T = \int f v^2 dv \) where \( f_{MX} \) is the Maxwell-Boltzmann distribution.

### 4. Function Learning Results in the Exploitation of Novel Physics

In this section, we describe the outcome of posing "Physics Discovery as an Optimization Problem" using our differentiable simulations framework. We start at the highest level, describing the training dynamics, and work our way deeper into the results by attempting to interpret what the training process has learned.

#### 4.1. Training Dynamics suggest Sample Efficiency

Figure 4 shows that the loss value is reduced over the duration of the training process, roughly 30 hours, which runs for 60 epochs. The convergence in the loss metric suggests that we were able to train a overparameterized neural network with 35 samples of data in 60 epochs. We attribute this to the effect of having so-called 'physical' gradients from training through a PDE solver (Holl et al., 2021).

#### 4.2. Physical Interpretation of the Learning Process

Figure 4 implies that the training process did indeed minimize eq. 14. To understand how this quantity was minimized, it is important to diagnose the simulations that minimize this quantity. To better understand the simulation results, we look at the electric field with respect to time, and find that the wavepacket survives for a much longer duration without damping.

#### 4.2.1. Electric Field

In this section, we analyze the observed behavior of the electric field in the simulations that minimize the metric function.

Figure 5 shows the electric field profile for three different simulations. In fig. 5a, only the first wavepacket is excited, and in fig. 5b, only the second wavepacket is excited. In fig. 5c, both wavepackets are excited.

Early in time, \( t = 400 \omega_p^{-1} \) (green), when only the first wavepacket has been excited figs. 5a and 5c agree perfectly. The second wavepacket is excited at \( t = 500 \omega_p^{-1} \).

At \( t = 900 \omega_p^{-1} \), once some time has passed after the excitation of the second wavepacket, the first wavepacket has not fully damped away. It is visible as small bumps in figs. 5a and 5c. The second wavepacket is also present at this time and easily seen in fig. 5b. A larger amplitude wavepacket is seen in fig. 5c.

Late in time, the difference in amplitude between the second wavepacket in figs. 5b and 5c is obvious. The second wavepacket has nearly damped away in fig. 5b. In fig. 5c, the second wavepacket continues to persist, at nearly the same energy as it was at \( t = 900 \omega_p^{-1} \).

This superadditive behavior, where \( f(x) + f(y) \leq f(x+y) \),
Figure 5. Early in time (green), (c) is the same as (a). Later in time, \( t = 900 \omega_p^{-1} \) (blue), (a) and (c) show very similar magnitudes for the first wavepacket near \( x = 1500 \lambda_D \) but the second wavepacket excitation is larger in (c) than (b). At \( t = 1300 \omega_p^{-1} \) (red), it is clear that (c) is not a superposition of (a) and (b) because (b) has damped away, while (c) retains electrostatic energy suggesting the involvement of a superadditive process.

suggests that there is a constructive interference. We observe this effect for all wavenumbers we model.

4.2.2. Phase Space

To determine the mechanism behind this phenomenon, we turn to the phase-space dynamics. In fig. 6, (i) is a space-time plot of the electric field. The two dashed-dot red lines at the front and back of the wavepacket are parallel and indicate the velocity of the wavefront. In fig. 6a(i), the front of the wavepacket propagates at a seemingly faster rate than the rear. This is due to the etching effect (Fahlen et al., 2009). In fig. 6b, the wave survives for a much longer time, as was also illustrated in fig. 5.

Figure 6 (ii) and (iii) are phase-space plots with their center indicated by the intersection of the horizontal timestamp line, and the dashed-dot line at the rear wavepacket red line at the back of the wavepacket. (iv) and (v) correspond to the intersection with the dashed-dot line at the front. (ii) and (iv) show the phase space within a window in x, while (iii) and (v) are the spatially averaged distribution function. (iii) and (v) serve as a proxy for approximating the propensity of Landau damping in that region.

In fig. 6a(ii) and (iii), we see that the rear of the wavepacket is Maxwellian. As previously shown, this is why the rear of the wavepacket damps faster than the front as in fig. 6a(i) (Fahlen et al., 2009).

In the simulations described here, fig. 6b show that the distribution function at the back of the wavepacket has trapped particle activity (fig. 6b(ii)) and near zero slope at the phase velocity of the wave (fig. 6b(iii)). Both plots show that the slope is negligible because of the arrival of streaming detrapped particles from the first wavepacket. Due to this effect, the remergence of Landau damping that occurs due to the loss of trapped particles in isolated wavepackets no longer occurs here. This results in a reduction of the etching and the wavepacket propagates freely for some time while the particles from the first wavepacket propagate and arrive at the rear of the second wavepacket.

5. Conclusion

In this preprint, we show how one may be able to discover novel physics using differentiable simulations by posing a physical question as an optimization problem. This removes the physicist from the loop and allows the simulation process to use physical gradients in order to arrive to the solution. The job of the physicist is abstracted one level up and now consists of providing their domain expertise in 1/ determining which parametric or functional dependencies to learn (and whether they should be learned using neural networks or other parameterized functions) and 2/ determining the nature of the objective function that best fulfills the desired criteria.

In fig. 2, we show how one may adapt an existing computational science workflow to the autodidactic process described here. In the work performed here, this process enabled the discovery of parameters in a 4D search space with known bounds but an unknown resolution. We trained the model over a coarse grid in \( t_0 \) and \( t_1 \), and learned functions for \( x_1 \) and \( \omega_1 \). Using gradient descent here allows an escape from the curse of dimensionality and reduces the problem from a 4D search to a 2D search + 2D gradient descent.

This discovery process is not limited to differentiable simulations. While in fig. 2, \( V \) represents a PDE solve, it only needs to be an AD-enabled function that is a model for a physical system. For example, rather than a PDE solve, \( V \) could represent a pre-trained neural-network-based emula-
Figure 6. Left - Spacetime plot of the electrostatic energy shows the long-lived wavepacket in (b) where the field in (b) survives for a longer duration than in (a). The horizontal line indicates the timestamp of the snapshots in the middle and right. The diagonal dashed-dot lines indicate the spatial location of the snapshots in the middle and right.

Middle Phasespace plots at the back (top) and front (bottom) of the wavepacket. In (b), the phase space shows significant activity at the back of the wavepacket while in (a), the distribution function is nearly undisturbed.

Right The spatially averaged distribution function. This confirms the fact that the distribution function has returned to a Maxwell-Boltzmann at the back of the wavepacket in (a), while in (b), the distribution function remains flat at the phase velocity of the wave. This is the reason behind the loss of damping.

The authors thank the anonymous reviewers for valuable feedback towards generalizing the content for a non-plasma-physics audience.

References


Unsupervised Discovery of Inertial-Fusion-Physics using AD + a PDE Solver + a Maximum Entropy Loss Function


A. Simulation and Modeling details

A.1. Training and Simulation Details

We vary the independent variables such that

\[ k_0 \in [0.26, 0.27, \ldots, 0.32], \]
\[ t_1 \in [400, 500, \ldots, 800], \]

(11) (12)

giving an input space of 35 samples from which we seek to learn these functions.

A.1.1. Architecture Details

We use a neural network with 2 hidden linear layers with 8 nodes activated with a ReLU function. The final layer is activated with a tanh function. The output is normalized such that

\[ p_1 = p_{\text{norm}} \times p_{\text{out}} + p_{\text{shift}} \]

where \( p_{\text{out}} \) is the output from the tanh function. We normalize the inputs between 0 and 1, and outputs with reasonable windows. For \( x_1 \), we allow the entire domain, and \( \Delta \omega_1 \in [-0.06, 0.06] \).

We use the ADAM optimizer with a learning rate of 0.05.

A.1.2. Simulation Details

The training simulations are performed with \( N_x = 6656, N_v = 160, N_t = 1200, t_{\text{max}} = 1100\omega_p^{-1}, x_{\text{max}} = 6600\lambda_D \). The amplitude of both drivers is 0.05. The spatial width is 400\( \lambda_D \) and temporal width is 50\( \omega_p^{-1} \).

We solve the Vlasov equation using a sixth order integrator in time (Casas et al., 2017) with operator splitting. Both operators are computed with exponential integrators with spectral discretizations in phase space. We use a spectral solver for Gauss’s Law.

In order to more closely match realistic plasma conditions, we also implement a Fokker-Planck collision operator and use a non-zero collision frequency \( \nu_{ee} = 10^{-4} \) for these simulations.

We implement absorbing boundaries by increasing the collision frequency of the Krook operator in a small localized region at the boundary (Strozzi et al., 2007).

We reproduce the solvers and tests implemented in (Joglekar & Levy, 2020) using JAX (Bradbury et al., 2018) and Haiku (Hennigan et al., 2020) to allow the usage of automatic differentiation (AD). The validation tests cover Gauss’s Law, Landau damping, and the energy and momentum conservation of the Fokker-Planck implementations when applicable depending on the operator.

A.2. Validating the Differentiable Simulator by Recovering Known Physics

We describe an additional test here which involves recovering the linear, small-amplitude electrostatic resonance using the gradient-based implementation enabled by this AD-capable implementation. This ensures that the gradients given by the AD system are physically reasonable and are representative of physical phenomenon.

A.2.1. Electrostatic Waves

A fundamental wave in plasma physics is the electrostatic wave in unmagnetized plasmas. The dispersion relation is given in numerous textbooks e.g. (Bellan), and reproduced here as

\[ 1 + \frac{\omega_e^2}{k^2} \int dv \frac{dg(v)/dv}{\omega - kv} = 0, \]

(13)

where \( \omega_e \) is the plasma frequency, \( k \) is the wavenumber, \( \omega \) is the resonant frequency, and \( v \) is the independent variable representing the velocity in the integral. \( g(v) \) is the distribution function of the plasma particles. This equation has been solved numerically and a lookup table for \( \omega \) as a function of \( k \) is provided in (Canosa, 1973).

We test the capability of our differentiable simulator framework by reproducing those calculations. To do so, we implement
the functionality in fig. 2c. We choose a loss function that minimizes free energy and maximizes the entropy given by

$$C(x_f) = -\sum_x (E_x^2 + \sum_v (f \log(f) - f_{MX} \log(f_{MX}))) \, dv).$$  \hspace{1cm} (14)

Because collisions return the distribution to a Maxwell-Boltzmann distribution, it is not enough to simply maximize $f \log(f)$ if one is interested in retaining non-linear effects. Because of this, the entropy in term quantifies the deviation of the distribution function from a Maxwell-Boltzmann distribution.

In this test, we launch plasma waves using a ponderomotive driver in a box with periodic boundary conditions. For each optimization, the wavenumber $k$ and the box size $x_{max}$ change. We provide a lower bound of 0.5 and an upper bound of 1.5 to the optimization algorithm. We use the L-BFGS algorithm implemented in SciPy (SciPy 1.0 Contributors et al., 2020). We set $f_{tol} = r_{tol} = 10^{-12}$. We learn the $\omega$ that will minimize the value given by eq. 14 and ensure that it is within 2 decimal places of the previously computed numerical solution of eq. 13 (Canosa, 1973).