Case study of a differentiable heterogeneous multiphysics solver for a nuclear fusion application

Jack B. Coughlin
Pasteur Labs; Brooklyn, NY
jack.coughlin@simulation.science

Archis Joglekar
Ergodic LLC; Seattle, WA
Pasteur Labs; Brooklyn, NY

Jonathan Brodrick
Pasteur Labs; Brooklyn, NY

Alexander Lavin
Pasteur Labs & ISI; Brooklyn, NY

Abstract

This work presents a case study of a heterogeneous multiphysics solver from the nuclear fusion domain. At the macroscopic scale, an auto-differentiable ODE solver in JAX computes the evolution of the pulsed power circuit and bulk plasma parameters for a compressing Z Pinch. The ODE solver requires a closure for the impedance of the plasma load obtained via root-finding at every timestep, which we solve efficiently using gradient-based Newton iteration. However, incorporating non-differentiable production-grade plasma solvers like Gkeyll (a C/CUDA plasma simulation suite) into a gradient-based workflow is non-trivial. The "Tesseract" software addresses this challenge by providing a multi-physics differentiable abstraction layer made fully compatible with JAX (through the `tesseract_jax` adapter). This architecture ensures end-to-end differentiability while allowing seamless interchange between high-fidelity solvers (Gkeyll), neural surrogates, and analytical approximations for rapid, progressive prototyping.

1 Introduction

The deep learning revolution has borne many fruits, but for scientific computing one of the main benefits has been the development of software frameworks for writing automatically differentiable (AD) numerical programs. End-to-end differentiable solvers have been created for fluid dynamics [1, 6], finite element analysis [17], molecular dynamics [12], and plasma transport in tokamaks [3], to name just a few examples. Differentiable simulations unlock sensitivity analysis and gradient-based optimization workflows end-to-end through large PDE solutions.

However, many physical systems of interest in engineering applications are "multiphysics" in nature. By a multiphysics system we mean a system involving a coupling between two or more physical domains or scales, each governed by its own set of ordinary or partial differential equations. The coupling between equation is often constrained by an implicit relation that requires the solution of a nonlinear system. Moreover, spatial and temporal scales may vary widely, necessitating the use of implicit time stepping methods or convergence to a steady state solution on the fast timescale. Examples of multiphysics systems include fluid-structure interactions, combusting and reacting flows, and radiation hydrodynamics [10].

Applying AD to multiphysics problems presents a particular challenge because their complexity may make it infeasible to rewrite all physics solvers in an AD-native framework. In this work we present one such example. One component of our system is an ODE for which it is quite simple to write a new solver in JAX. However, this is coupled to a high-dimensional PDE for which an excellent, high-accuracy and efficient solver is available in C. To leverage this existing battle-tested code we use

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025

Tesseracts [8] to provide an abstraction layer at the boundary between solvers. Crucially, as Gkeyll itself is non-differentiable, Jacobian-vector products and vector-Jacobian products of the C solver are implemented using finite differences, and the resulting differentiable function is incorporated into the outer JAX program using `tesseract_jax`[1]. Furthermore, Tesseracts enable a modular software architecture, enabling us to swap out different implementations for the PDE component, such as neural surrogates or analytic approximations. End-to-end differentiability of the program is maintained regardless of the underlying solver.

2 Fusion Z Pinch Compression Modeling

2.1 Background

The Sheared Flow-Stabilized Z Pinch [13] is a promising approach to magnetically confined nuclear fusion (MCF). It has the benefits of a compact size, requires no external magnetic field coils, and relatively simple engineering compared to other MCF concepts. The Z Pinch functions on the basis of the plasma pinch principle: a plasma carrying a current with density vector $\boldsymbol{j}$ produces a magnetic field $\boldsymbol{B}$ and corresponding force vector $\boldsymbol{F} = \boldsymbol{j} \times \boldsymbol{B}$ which acts to compress the plasma in on itself. By ramping up the plasma current, one can increase $\boldsymbol{F}$ and thus the equal and opposite pressure gradient force $-\nabla p$, driving the plasma density and temperature up to fusion conditions.

2.2 Macroscale circuit equation

The pulsed power driver of the Z Pinch device can be modeled as a series RLC circuit as done in Ref. [5]. Let Q and I denote the time-dependent capacitor charge and total circuit current respectively. The voltage balance law around the circuit, including all three RLC circuit components and the plasma load, is

$$\left(L + L_p + \dot{L}_p \frac{I}{\dot{I}}\right)\dot{I} + RI + \frac{Q}{C} = -V_{R_p}, \tag{1}$$

where L is the circuit inductance, R the circuit resistance and C the capacitance. The plasma load impedance is characterized by the plasma inductance L_p and the resistive plasma voltage V_{R_p}. The plasma inductance is a geometric property of the pinch profile and is derived in Ref. [4]. The resistive plasma voltage V_{R_p} is the voltage required to drive a current I through the plasma at steady state, and is a function of the current I and the plasma temperature T and density n. Temperature T is related to current I, linear density N, and charge state Z by the well-known Bennett relation for pinches. The relation between T and n is given by the specific entropy $s = \ln(T/n^{\gamma-1})$, where $\gamma = 5/3$ is the adiabatic constant for hydrogenic plasmas. The complete ODE system for the plasma and circuit state is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} Q \\ I \\ s \end{pmatrix} = \begin{pmatrix} I \\ \frac{1}{L + \frac{L_z \mu_0}{2\pi(\gamma-1)} + L_p}\left[-\frac{Q}{C} - RI - V_{R_p}(I,T,n)\right] \\ \frac{\gamma-1}{(1+Z)T}(P_\eta - P_{Br}) \end{pmatrix}, \quad T = \frac{\mu_0 I^2}{8\pi(1+Z)N}, \quad n = \left(\frac{T}{e^s}\right)^{\frac{1}{\gamma-1}} \tag{2}$$

The terms P_η and P_{Br} are the contributions from Ohmic heating of the plasma and bremsstrahlung radiative cooling, a key energy loss mechanism in fusion plasmas. Expressions for them may be found in [7].

2.3 Microscale plasma physics closure

The key remaining challenge is to specify a closure for $V_{R_p}(I, T, n)$. While the circuit dynamics can be described by the lumped element model eq. (2), the plasma impedance depends closely on the distribution of current-carrying plasma particles near the electrodes. Plasma-electrode boundaries form a structure known as a plasma sheath, in which the mobility difference between light electrons and heavy, positively charged ions results in a nonzero charge density at the plasma edge. It can be shown [16] that in the presence of a bias voltage, the plasma current density limits to a value that scales as $n\sqrt{T}$. Moreover, computational investigations have shown [14, 15] that the current density

[1] https://github.com/pasteurlabs/tesseract-jax

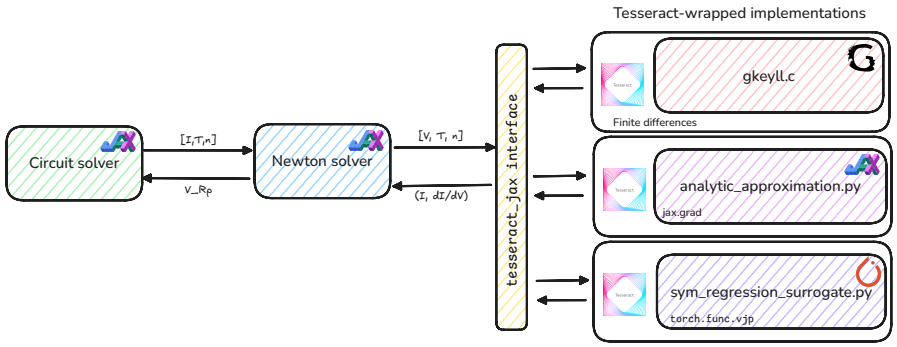

Figure 1: Dataflow diagram of the software components used in solving Equations (2) and (3). Different implementations of the $V \mapsto I$ map are packaged in Tesseracts. Implementation details are hidden behind a `tesseract_jax` call enabling rapid experimentation with different models.

is sensitive to details of the sheath structure and electrode boundary conditions. A physically accurate model of the sheath requires the solution of the Vlasov-Poisson-Fokker-Planck (VPFP) system of equations,

$$\partial_t f_s + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f_s + \frac{q_s}{m_s} \boldsymbol{E} \cdot \nabla_{\boldsymbol{v}} f_s = \sum_{s'} \nu_{ss'} C_{ss'} + \Gamma_s, \quad \nabla_{\boldsymbol{x}} \cdot \boldsymbol{E} = \frac{1}{\epsilon_0} \sum_s q_s \int_{\mathbb{R}^3} f_s \, \mathrm{d}\boldsymbol{v}, \tag{3}$$

where the species index s varies over the electrons and ions. In full generality, Eqn. (3) is a 6-dimensional time-dependent partial differential equation (PDE). Near the magnetic axis of the Z Pinch, we assume spatial symmetry in the azimuthal and radial directions, reducing (3) to a 2D PDE. The plasma-electrode boundary condition is a perfectly absorbing wall on both sides of the physical domain. The term Γ_s is a source term that serves to replenish particles and energy lost to the wall. Formally, the coupling between the VPFP system (3) and the circuit ODE (2) is given by

$$V_{R_p}(I, T, n) = V \text{ s.t. } \begin{cases} I = \frac{N}{n} \sum_s q_s \int_{\mathbb{R}^3} \boldsymbol{v} f_s^\infty \, \mathrm{d}\boldsymbol{v}, \\ f_s^\infty = f_s(t \to \infty), \quad \partial_t f_s^\infty = 0, \\ f_s|_{t=0} = \frac{m_s n}{(2\pi T)^{1/2}} \exp\left(-m_s |\boldsymbol{v}|^2 / (2T)\right), \\ \boldsymbol{E} = \nabla \phi, \quad \phi|_{x=L_z} - \phi|_{x=0} = V. \end{cases} \tag{4}$$

Equation (4) is solved via Newton iteration. A proposed solution V is applied as a Dirichlet boundary condition on the electrostatic potential ϕ, and Equation (3) is integrated in time to a steady-state solution f_s^∞, with initial conditions determined by the plasma temperature and density T, n. The residual is the difference between the target current I and the observed steady-state current. The Newton iteration requires gradients of the map $V \mapsto I$.

3 Software architecture

Our implementation uses JAX [2], a Python framework for writing AD programs. Equation (2) is discretized using Diffrax [9], and the Newton inversion of the $V \mapsto I$ map is implemented with the Optimistix library [11].

We have three implementations of the V→I map occupying different points on the accuracy-efficiency curve: (1) The C/CUDA suite Gkeyll provides high-fidelity solutions requiring 18 GPU-hours per trajectory, but gradients can only be approximated using finite differences; (2) a symbolic regression trained on the outputs of a purpose-built semi-Lagrangian VPFP solver, with AD provided by PyTorch; (3) a pure JAX closed-form approximation to the saturation current [14] to enable rapid prototyping The architecture is diagrammed in Figure 1. The Tesseract abstraction enables seamless interchange between these models without modifying the outer optimization loop (Listing 1) and ODE solve.

4 Results

To demonstrate the flexibility of the implementation, Figure 2 presents solution trajectories in density-temperature space for the three different plasma models. The solver is end-to-end differentiable

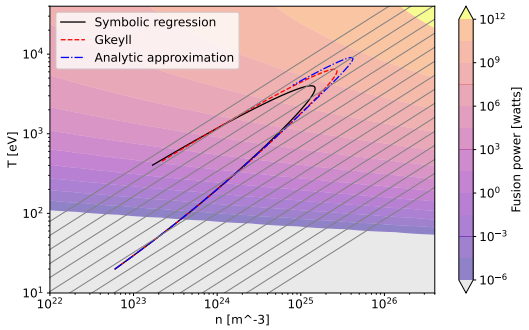

Figure 2: Density-temperature trajectories of three rollouts of the circuit ODE, each with a different plasma model. Straight lines are lines of constant specific entropy.

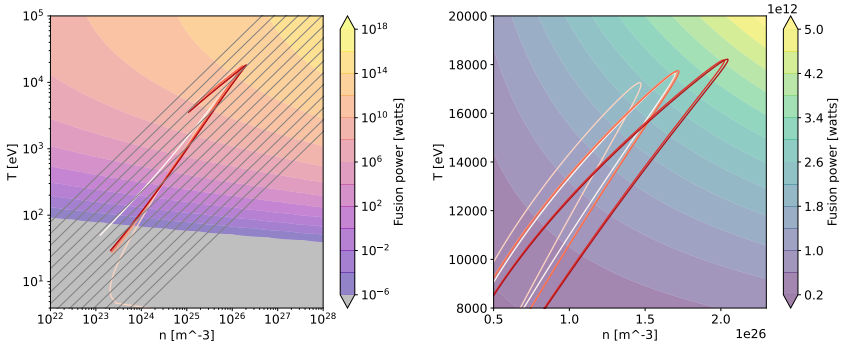

Figure 3: Density-temperature trajectories of successive optimization iterates. We optimize over initial temperature T and circuit capacitance C using the L-BFGS quasi-Newton method. The gradient-based optimizer quickly traverses the two-dimensional parameter space.

Listing 1: Pseudocode for JAX implementation of a Newton solve through an arbitrary Tesseract.

```
def newton_solve(I, T, n, plasma_tesseract):
    def residual(V_guess, args):
        return tesseract_jax.apply_tesseract(
            plasma_tesseract, {"V": V_guess, "T": T, "n": n}
        ) - I
    solver = optimistix.Newton(rtol=1e-4, atol=1e-4)
    return optimistix.root_find(residual, T, solver)
```

for any choice of plasma model Tesseract. Figure 3 presents the solution trajectories of successive optimization iterates for the closed form approximation. We use L-BFGS to optimize for Q, the ratio of fusion energy to capacitor energy, with respect to initial temperature $T|_{t=0}$ and capacitance C.

5 Conclusion

Multiphysics simulations in engineering applications often require coupling heterogeneous software components including legacy C/CUDA codes that cannot be easily rewritten in AD frameworks. We demonstrated that Tesseracts enable end-to-end differentiable multiphysics workflows that incorporate such solvers without dealing with low-level custom JAX primitives or code rewrites. The key benefit is modularity: our implementation seamlessly interchanges between high-fidelity (Gkeyll), surrogate (symbolic regression), and analytical closure models, enabling rapid design-space exploration with appropriate accuracy-efficiency tradeoffs. This approach is particularly valuable when computational budgets vary (e.g., hyperparameter sweeps vs. final validation) or when solver fidelity requirements are uncertain during development.

Acknowledgements and Disclosure of Funding

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award FES-ERCAP0026741. This work also received support from Pasteur Labs and Institute for Simulation Intelligence (ISI). The authors declare no competing interests.

References

[1] Deniz A. Bezgin, Aaron B. Buhendwa, and Nikolaus A. Adams. "JAX-Fluids 2.0: Towards HPC for Differentiable CFD of Compressible Two-Phase Flows". In: *Computer Physics Communications* 308 (Mar. 2025), p. 109433. ISSN: 00104655. DOI: 10.1016/j.cpc.2024.109433. (Visited on 02/20/2025).

[2] James Bradbury et al. *JAX: Composable Transformations of Python+NumPy Programs*. 2018.

[3] Jonathan Citrin et al. *TORAX: A Fast and Differentiable Tokamak Transport Simulator in JAX*. Dec. 2024. DOI: 10.48550/arXiv.2406.06718. arXiv: 2406.06718 [physics]. (Visited on 10/09/2025).

[4] Daniel W. Crews et al. "The Kadomtsev Pinch Revisited for Sheared-Flow-Stabilized Z-pinch Modeling". In: *IEEE Transactions on Plasma Science* 52.10 (Oct. 2024), pp. 4804–4816. ISSN: 0093-3813, 1939-9375. DOI: 10.1109/TPS.2024.3383312. arXiv: 2404.06636 [physics]. (Visited on 05/23/2025).

[5] I.A.M. Datta, E.T. Meier, and U. Shumlak. "Whole Device Modeling of the Fuze Sheared-Flow-Stabilized Z Pinch". In: *Nuclear Fusion* 64.6 (June 2024), p. 066016. ISSN: 0029-5515, 1741-4326. DOI: 10.1088/1741-4326/ad3fcb. (Visited on 06/03/2025).

[6] Aleksandra Franz et al. *PICT – A Differentiable, GPU-Accelerated Multi-Block PISO Solver for Simulation-Coupled Learning Tasks in Fluid Dynamics*. May 2025. DOI: 10.48550/arXiv.2505.16992. arXiv: 2505.16992 [cs]. (Visited on 10/09/2025).

[7] R. J. Goldston and P. H. Rutherford. *Introduction to Plasma Physics*. Bristol, UK ; Philadelphia: Institute of Physics Pub, 1995. ISBN: 978-0-7503-0325-5 978-0-7503-0183-1.

[8] Dion Häfner and Alexander Lavin. "Tesseract Core: Universal, Autodiff-Native Software Components for Simulation Intelligence". In: *Journal of Open Source Software* 10.111 (July 2025), p. 8385. ISSN: 2475-9066. DOI: 10.21105/joss.08385. (Visited on 10/09/2025).

[9] Patrick Kidger. "On Neural Differential Equations". PhD thesis. University of Oxford, 2021.

[10] Alexander Lavin et al. "Simulation Intelligence: Towards a New Generation of Scientific Methods". In: *ArXiv* abs/2112.03235 (2021). URL: https://api.semanticscholar.org/CorpusID:244909059.

[11] Jason Rader, Terry Lyons, and Patrick Kidger. "Optimistix: Modular Optimisation in JAX and Equinox". In: *arXiv:2402.09983* (2024). arXiv: 2402.09983.

[12] Samuel S. Schoenholz and Ekin D. Cubuk. *JAX, M.D.: A Framework for Differentiable Physics*. Dec. 2020. DOI: 10.48550/arXiv.1912.04232. arXiv: 1912.04232 [physics]. (Visited on 10/09/2025).

[13] U. Shumlak et al. "The Sheared-Flow Stabilized Z-Pinch". In: *Fusion Science and Technology* 61.1T (Jan. 2012), pp. 119–124. ISSN: 1536-1055, 1943-7641. DOI: 10.13182/FST12-A13407. (Visited on 09/24/2024).

[14] Chirag R. Skolar et al. "Continuum Kinetic Investigation of the Impact of Bias Potentials in the Current Saturation Regime on Sheath Formation". In: *Physics of Plasmas* 30.1 (Jan. 2023), p. 012504. ISSN: 1070-664X, 1089-7674. DOI: 10.1063/5.0134656. arXiv: 2211.06488 [physics]. (Visited on 05/23/2025).

[15] Chirag R. Skolar et al. *General Kinetic Ion Induced Electron Emission Model for Metallic Walls Applied to Biased Z-pinch Electrodes*. Feb. 2025. DOI: 10.48550/arXiv.2502.01802. arXiv: 2502.01802 [physics]. (Visited on 05/23/2025).

[16] P. C. Stangeby. *The Plasma Boundary of Magnetic Fusion Devices*. Plasma Physics Series. Bristol ; Philadelphia: Institute of Physics Pub, 2000. ISBN: 978-0-7503-0559-4.

[17] Tianju Xue et al. "JAX-FEM: A Differentiable GPU-accelerated 3D Finite Element Solver for Automatic Inverse Design and Mechanistic Data Science". In: *Computer Physics Communications* 291 (Oct. 2023), p. 108802. ISSN: 00104655. DOI: 10.1016/j.cpc.2023.108802. (Visited on 10/09/2025).

A Technical Appendices

A.1 Simulation parameters

Details of the numerical setup of the Vlasov-Poisson system match those used in [14], except for T and n which take their values from the ODE solver.

Table 1: Simulation parameters for rollouts contained in Figure 2

Parameter	Description	Size (μm)
C	Capacitance	$400\,\mu\mathrm{F}$
R	Circuit resistance	$1.5\,\mathrm{m}\Omega$
L	Circuit inductance	$200\,\mathrm{nH}$
V_c	Initial capacitor voltage	$40\,\mathrm{kV}$
$T\|_{t=0}$	Initial temperature	$20\,\mathrm{eV}$
$n\|_{t=0}$	Initial density	$6 \times 10^{22}\,\mathrm{m}^{-3}$
L_z	Axial length	$50\,\mathrm{cm}$
r_w	Outer electrode radius	$10\,\mathrm{cm}$
Z	Charge state	1
N	Linear density	$1 \times 10^{19}\,\mathrm{m}^{-1}$

Table 2: Simulation parameters for optimization problem contained in Figure 3

Parameter	Description	Size (μm)
C	Capacitance	$100\,\mu\mathrm{F} \leq C \leq 400\,\mu\mathrm{F}$
R	Circuit resistance	$1.5\,\mathrm{m}\Omega$
L	Circuit inductance	$286\,\mathrm{nH}$
V_c	Initial capacitor voltage	$0\,\mathrm{V} \leq V_c \leq 80\,\mathrm{kV}$
$T\|_{t=0}$	Initial temperature	$10\,\mathrm{eV} \leq T \leq 1\,\mathrm{keV}$
$n\|_{t=0}$	Initial density	$n = 1\,\mathrm{MPa}/T$
L_z	Axial length	$50\,\mathrm{cm}$
r_w	Outer electrode radius	$10\,\mathrm{cm}$
Z	Charge state	1
N	Linear density	$1 \times 10^{19}\,\mathrm{m}^{-1}$

A.2 Computing resources

The Gkeyll run included in Figure 2 required 18 hours on a single A100 GPU on the Perlmutter cluster. The other runs were done on a 2024 M4 Macbook Pro.