Differentiable Simulation: Making Physics Learnable

Krishna Kumar

University of Texas at Austin

krishnak@utexas.edu

Overview

1. Introduction: From Neural Networks to Physics
2. JAX: The Scientific Computing Framework
3. The 1D Wave Equation: Our Physics Model
4. Full Waveform Inversion: The Inverse Problem
5. Advanced Optimization: Newton's Method and BFGS
6. Applications and Future Directions
7. Summary and Future Directions

Outline

1. Introduction: From Neural Networks to Physics
2. JAX: The Scientific Computing Framework
3. The 1D Wave Equation: Our Physics Model
4. Full Waveform Inversion: The Inverse Problem
5. Advanced Optimization: Newton's Method and BFGS
6. Applications and Future Directions
7. Summary and Future Directions

Learning Objectives

- Understand the paradigm shift from traditional simulation to differentiable simulation
- Learn how automatic differentiation enables gradient-based optimization in physics
- Master the JAX framework for high-performance scientific computing
- Implement a differentiable wave equation solver
- Apply differentiable simulation to Full Waveform Inversion (FWI)

Open Notebook: Differentiable Simulation

What is Differentiable Simulation?

Traditional Simulation:

- Given parameters $\rightarrow$ compute forward simulation
- Trial-and-error parameter search for inverse problems
- Expensive, often requires domain expertise

Differentiable Simulation Revolution

If we can compute $\frac{\partial \text{simulation}}{\partial \text{parameters}}$, we can use gradient descent to find optimal parameters!

The Paradigm Shift:

- **Forward Problem**: Given parameters $\rightarrow$ predict observations
- **Inverse Problem**: Given observations $\rightarrow$ optimize parameters using gradients

The Power of Automatic Differentiation

PyTorch Example:

- Tracks all operations on tensors
- `.backward()` computes gradients
- Reverse-mode AD (backpropagation)

Simple loss function:

$$L = (ax + b - \text{target})^2$$

JAX Advantages:

- Functional programming approach
- JIT compilation for speed
- Composable transformations
- `grad`, `jit`, `vmap`

Same computation in JAX:

$$\text{grad\_fn} = \text{grad}(\text{loss\_fn})$$

Outline

1. Introduction: From Neural Networks to Physics
2. JAX: The Scientific Computing Framework
3. The 1D Wave Equation: Our Physics Model
4. Full Waveform Inversion: The Inverse Problem
5. Advanced Optimization: Newton's Method and BFGS
6. Applications and Future Directions
7. Summary and Future Directions

JAX Transformations: The Power of Composition

Key transformations work together:

Core Transformations

- **grad**: Automatic differentiation
- **jit**: Just-In-Time compilation
- **vmap**: Vectorization (automatic batching)

Example: Polynomial with transformations

$$f(x) = x^4 - 3x^3 + 2x^2 - x + 1$$

- `fast_f = jit(f)` - Compiled version
- `df = grad(f)` - Gradient function
- `vectorized = vmap(f)` - Works on arrays
- `fast_grad = jit(grad(f))` - Compiled gradient

Performance gain: $3\times$ speedup with JIT compilation!

JAX vs PyTorch for Physics

PyTorch:

- Object-oriented design
- Mutable tensors
- Dynamic computation graphs
- Great for deep learning

Best for:

- Neural network training
- Rapid prototyping
- Dynamic models

JAX:

- Functional programming
- Immutable arrays
- Composable transformations
- NumPy-compatible API

Best for:

- Scientific computing
- High-performance simulation
- Mathematical optimization

For physics simulations: JAX's functional approach and JIT compilation provide significant advantages.

Outline

1. Introduction: From Neural Networks to Physics
2. JAX: The Scientific Computing Framework
3. The 1D Wave Equation: Our Physics Model
4. Full Waveform Inversion: The Inverse Problem
5. Advanced Optimization: Newton's Method and BFGS
6. Applications and Future Directions
7. Summary and Future Directions

The 1D Acoustic Wave Equation

Governing equation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

Where:

- $u(x, t)$ is the wavefield (pressure/displacement)
- $c(x)$ is the wave speed (what we want to estimate)

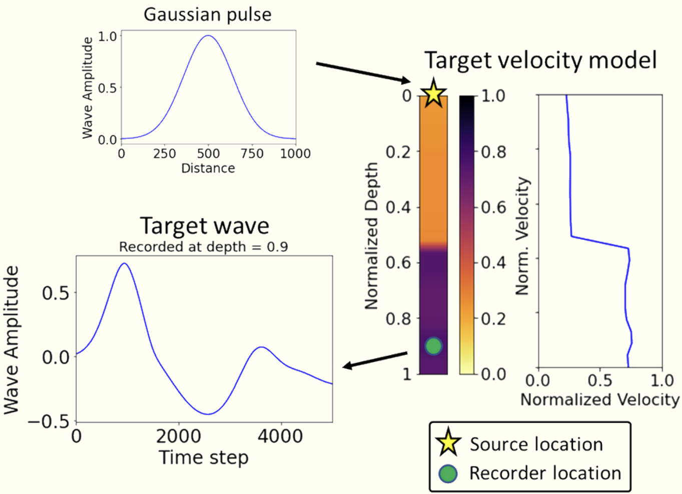
Finite difference discretization:

$$u_i^{n+1} = 2u_i^n - u_i^{n-1} + \frac{c_i^2 \Delta t^2}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

Initial conditions: Gaussian source at $x = 0.5$

$$u_0 = \exp(-5(x - 0.5)^2)$$

The Forward Problem: Wave Propagation



1D Full Waveform Inversion setup showing wave propagation.

JAX Implementation: JIT-Compiled Solver

Key implementation features:

JIT Compilation

`@jit` decorator compiles the entire time-stepping loop for high performance

Functional Programming

- Pure functions with no side effects
- Immutable arrays (no in-place operations)
- `lax.fori_loop` for compiled loops

Central difference scheme:

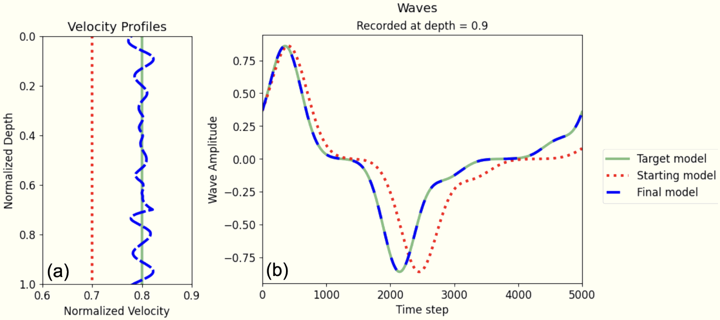
$$u_2 = 2u_1 - u_0 + C^2(u_{1,j-1} - 2u_{1,j} + u_{1,j+1})$$

where $C^2 = c^2 \Delta t^2 / \Delta x^2$

Boundary handling: `jnp.roll` with explicit boundary setting

Outline

1. Introduction: From Neural Networks to Physics
2. JAX: The Scientific Computing Framework
3. The 1D Wave Equation: Our Physics Model
4. Full Waveform Inversion: The Inverse Problem
5. Advanced Optimization: Newton's Method and BFGS
6. Applications and Future Directions
7. Summary and Future Directions



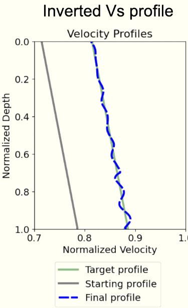
Constant velocity recovery using gradient descent.

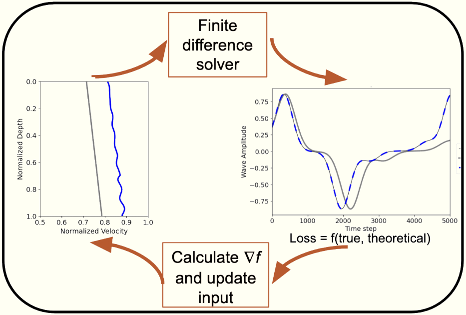
Problem Setup:

- **Target**: $c = 1.0$ (constant velocity)
- **Initial guess**: $c = 0.8$
- **Optimization**: Adam optimizer with learning rate 10^{-3}

Results:

Example 2: Linear Velocity Profile





Full waveform inversion animation showing optimization progress.

Problem Setup:

- **Target**: $c(x) = 0.9 + 0.1x$ (linear increase)
- **Parameters**: 1000 spatial points to optimize
- **Challenge**: High-dimensional optimization landscape

Algorithm Comparison:

Outline

1. Introduction: From Neural Networks to Physics
2. JAX: The Scientific Computing Framework
3. The 1D Wave Equation: Our Physics Model
4. Full Waveform Inversion: The Inverse Problem
5. Advanced Optimization: Newton's Method and BFGS
6. Applications and Future Directions
7. Summary and Future Directions

First-Order vs Second-Order Methods

First-Order (Adam):

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

Characteristics:

- Uses only gradient information
- Robust to noise
- Memory: O(1) per parameter
- Good for large, noisy problems

Second-Order (L-BFGS):

$$x_{k+1} = x_k - H_k^{-1} \nabla f(x_k)$$

Characteristics:

- Uses curvature information
- Superlinear convergence
- Memory: O(m) history vectors
- Best for smooth functions

L-BFGS Key Innovation

Approximates inverse Hessian using gradient differences from recent iterations, avoiding expensive Hessian computation.

BFGS Update Formula

BFGS approximates the inverse Hessian iteratively:

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

Where:

- $s_k = x_{k+1} - x_k$ (step difference)
- $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ (gradient difference)
- $\rho_k = \frac{1}{y_k^T s_k}$

L-BFGS (Limited-memory BFGS):

- Stores only recent m vector pairs (s_k, y_k)
- Suitable for large-scale problems
- Excellent for smooth optimization landscapes

Our results: L-BFGS achieved 20% better accuracy than Adam for the smooth wave equation optimization.

Outline

1. Introduction: From Neural Networks to Physics
2. JAX: The Scientific Computing Framework
3. The 1D Wave Equation: Our Physics Model
4. Full Waveform Inversion: The Inverse Problem
5. Advanced Optimization: Newton's Method and BFGS
6. Applications and Future Directions
7. Summary and Future Directions

The Revolution: Physics as Learnable Components

What we've demonstrated:

Technical Achievements

- **Physics-Based Model**: Realistic wave equation with finite differences
- **Automatic Differentiation**: Exact gradients through entire simulation
- **Scalable Optimization**: 1 to 1000 parameters seamlessly
- **Algorithm Comparison**: Practical insights on optimizer choice

Key Results

- **Constant velocity**: Perfect recovery with gradient descent
- **Linear profile**: High-fidelity reconstruction
- **L-BFGS advantage**: Superior convergence for smooth landscapes

The paradigm shift: Physics simulations become **differentiable building blocks** for gradient-based optimization.

Real-World Applications

Geophysics:

- Subsurface imaging
- Earthquake location
- Earth structure inversion
- Oil and gas exploration

Medical Imaging:

- Ultrasound tomography
- Photoacoustic imaging
- Tissue characterization

Engineering:

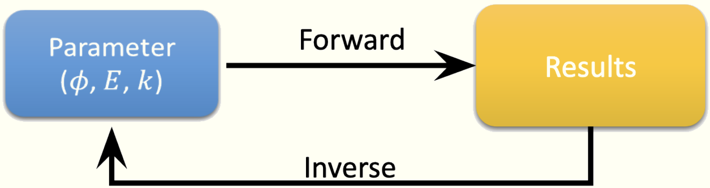
- Structural health monitoring
- Non-destructive testing
- Design optimization
- Material characterization

Climate Science:

- Atmospheric parameter estimation
- Ocean circulation models
- Weather prediction

Common theme: Transform expensive inverse problems into efficient gradient-based optimization.

Forward vs Inverse: The Complete Picture



Forward and inverse problems in scientific computing.

Traditional Approach:

- Forward problem: Well-posed, deterministic
- Inverse problem: Ill-posed, requires regularization
- Manual parameter tuning and domain expertise

Differentiable Simulation:

- Automatic gradient computation through physics
- Principled optimization with advanced algorithms
- End-to-end learning from data to physics parameters

Outline

1 Introduction: From Neural Networks to Physics

2 JAX: The Scientific Computing Framework

3 The 1D Wave Equation: Our Physics Model

4 Full Waveform Inversion: The Inverse Problem

5 Advanced Optimization: Newton's Method and BFGS

6 Applications and Future Directions

7 Summary and Future Directions

Key Takeaways

1. Paradigm Shift

- Physics simulations become differentiable components
- Gradient-based optimization replaces trial-and-error
- JAX enables high-performance scientific computing

2. Technical Implementation

- JIT compilation for performance
- Functional programming for correctness
- Composable transformations: `grad`, `jit`, `vmap`

3. Optimization Insights

- L-BFGS excels for smooth physics problems
- Adam provides robustness for noisy/large-scale problems
- Second-order methods worth the complexity for precision

Future Directions

Immediate Extensions:

- Multi-dimensional PDEs (2D/3D wave equations)
- Coupled multi-physics problems
- Stochastic differential equations
- Real-time optimization and control

Advanced Topics:

- Physics-Informed Neural Networks (PINNs) integration
- Uncertainty quantification in inverse problems
- Multi-objective optimization with physical constraints
- Hybrid neural-physics models

Next Steps:

- Explore the interactive projectile demo
- Try different velocity models and PDE types

The Bigger Picture: Scientific Machine Learning

Differentiable simulation bridges physics and machine learning

Traditional Scientific Computing:

- Model-driven approach
- Physics-based equations
- Limited by computational cost
- Parameter sensitivity analysis

Machine Learning:

- Data-driven approach
- Pattern recognition
- Scalable optimization
- Automatic differentiation

The Synthesis

Differentiable simulation combines the best of both worlds:

- Physical constraints and interpretability
- Efficient gradient-based optimization

Questions?

Thank you!

Contact:
Krishna Kumar
krishnak@utexas.edu
University of Texas at Austin

Interactive Demo:

Differentiable Simulation Notebook