In Praise of Differentiable Economic Models: A Unified Framework for Optimization, Sensitivity, and Control in Macro-Financial Systems

Abstract

Traditional economic modeling—spanning Stock-Flow Consistent (SFC) models, Input-Output (IO) analysis, and Agent-Based Models (ABMs)—has historically relied on discrete-time simulations. While these "black-box" approaches ensure structural integrity, they are notoriously difficult to calibrate, optimize, and analyze for sensitivities. This paper advocates for a transition to **Differentiable Programming (DP)** as the standard for economic modeling. By representing economic identities as computational graphs in frameworks like PyTorch, JAX, or TensorFlow, researchers can leverage automatic differentiation to compute exact gradients across time. We demonstrate how this shift enables a "control knob" approach to policy design, instantaneous multi-parameter sensitivity analysis, and automated calibration against national accounts. We argue that "Differentiable Economics" provides the necessary mathematical infrastructure to manage systemic risks in an increasingly volatile global economy.

1. Introduction: The Simulation Bottleneck

The Status Quo

Economic models are constructed as sets of iterative equations. Whether implemented in System Dynamics platforms (Stella, Vensim), spreadsheets (Excel), or custom code (C++, Python), the underlying paradigm remains unchanged: **run the simulation forward, observe the outcome**.

The "Run-and-See" Problem

This approach creates a computational bottleneck:

- **Sensitivity Analysis:** Requires expensive Monte Carlo methods or manual parameter sweeping
- **Calibration:** Multi-dimensional optimization landscapes that are intractable without sampling
- **Policy Discovery:** Economists manually adjust "trigger values" until the system appears stable—a process akin to archaeology rather than engineering

The Control Challenge

In high-dimensional state spaces (which any realistic model occupies), finding stable policy rules is difficult. The problem is exacerbated by:

- Non-convexity (multiple local minima)
- Time-dependent effects (a policy that works in year 1 may destabilize the system in year 20)
- Feedback loops (where the impact of a policy changes as the economy evolves)

Thesis

Differentiability is not merely a computational convenience; it is a **fundamental expansion** of the economist's analytical toolkit. It allows us to move from *observing* economic systems to *optimizing* them via gradients—the same mathematical machinery that has driven breakthroughs in machine learning, physics simulation, and climate science.

2. The Differentiable Paradigm

Automatic Differentiation (AD) vs. Numerical Derivatives

Traditional sensitivity analysis uses numerical differentiation: perturb a parameter slightly, rerun the simulation, measure the change. This is expensive and imprecise.

Automatic differentiation (AD) computes exact symbolic derivatives by applying the chain rule throughout a computation. For a complex simulation with thousands of steps and millions of variables, AD computes the full gradient in **one backward pass**—a process known as **reverse-mode differentiation** or **backpropagation**.

Key Insight: The cost of computing a gradient is roughly the same as running the simulation once, regardless of the number of parameters being differentiated.

The Macroeconomy as a Computational Graph

Every economic model is, at its core, a directed acyclic graph (DAG):

- **Nodes** represent variables (GDP, consumption, investment, debt, etc.)
- **Edges** represent dependencies (behavioral rules, accounting identities)
- **Time** is unrolled as multiple copies of this graph, one for each period

By representing this graph explicitly in a framework like PyTorch or JAX, we make it **differentiable** by default. Each edge becomes a differentiable operation, and the entire system becomes a function from parameters → outcomes.

Backpropagation Through Time (BPTT)

Consider a macroeconomic shock in year 1 that triggers a debt spiral, culminating in financial collapse in year 40. How does this future collapse influence the optimal policy in year 1?

BPTT answers this question: Starting from the collapse at year 40, backpropagation flows the gradient backward through all 40 time steps, revealing how changes to year-1 policy parameters would have prevented the cascade.

Mathematically:

$$L_T\theta_1 = L_{TT} \cdot {}_{TT-1} \;{}_{21} \cdot {}_1\theta_1$$

This gradient is computed exactly, without approximation, through automatic differentiation.

3. Case Study 1: Stock-Flow Consistent (SFC) Models

Focus: Financial Accounting and Systemic Stability

SFC models ensure that every dollar flows from a source to a use, and every stock is accounted for. This is their greatest strength—and a bottleneck when implemented conventionally.

The Differentiable SIM/DISCE

We have implemented the SIM model (Service-Induced Macroeconomics) as a differentiable program in PyTorch. The innovation is straightforward: instead of running the model once and observing the outcome, we define a **loss function** representing economic "health":

$$L = \lambda_1 \sum_t (U_t - U^*)^2 + \lambda_2 \sum_t (\Delta Y_t)^2 + \lambda_3 \sum_t (\Omega_t)^2$$

Where:

- $(U_t - U^*)^2$ penalizes unemployment above the natural rate
- $(\Delta Y_t)^2$ penalizes GDP volatility
- $(\Omega_t)^2$ penalizes unsustainable debt-to-wealth ratios

The optimizer then adjusts policy parameters (tax rates, interest rate rules) via gradient descent to minimize L.

Benefit: Solving the "Oscillation Problem"

In discrete simulation environments like Stella, SFC models often exhibit damped or undamped oscillations depending on parameter choices. Economists respond by manually tuning "smoothing constants" or adding ad-hoc damping mechanisms.

With differentiable SFC models, the optimizer **automatically finds the policy rate that perfectly damps oscillations**. There is no guessing—the mathematics dictates the solution.

4. Case Study 2: Supply Chain Resilience & Bottleneck Analysis

Focus: Physical Production and Logistics

Modern supply chains are complex networks with thousands of interdependent nodes. A disruption at a single "critical node" can cascade into global shortages.

The Problem

Identifying which node is most critical requires expensive sensitivity analysis. In a traditional model, you would:

1. Reduce node capacity by 10%
2. Rerun the entire supply chain simulation
3. Measure the impact on total output
4. Repeat for all nodes

For a 10,000-node network, this requires 10,000 forward passes.

Benefit: Differentiable Supply Chains

With automatic differentiation, we compute:

$$\mathrm{olpopciy}$$

in **a single backward pass**. We instantly identify which nodes have the highest "elasticity" of total output—the most critical points for infrastructure investment or resilience retrofitting.

Moreover, we can define a loss function that balances output, carbon emissions, and supply chain redundancy, then use gradients to optimize the network architecture itself.

5. Case Study 3: DebtRank & Networked Financial Contagion

Focus: Interbank Lending and Systemic Risk

DebtRank is a standard algorithm for measuring contagion in financial networks: if Bank A defaults, how much impact does it have on Bank B, which affects Bank C, etc.? Traditional DebtRank is a **discrete algorithm**—each bank either defaults or doesn't—making it non-differentiable.

The Problem

Regulators want to know: what is the optimal capital buffer for each systemically important bank to minimize the probability of cascading collapse? This is a high-dimensional optimization problem.

Benefit: Differentiable Contagion

By relaxing the discrete "default/no-default" decision using smooth approximations (e.g., sigmoid functions), we make the contagion model differentiable. We can then:

1. Define a loss function: $L = (\mathrm{yiccollp}) + \lambda \cdot (\mathrm{cpil})^2$
2. Use gradients to find the **optimal capital buffer for each bank** that minimizes collapse risk while minimizing regulatory costs
3. Perform sensitivity analysis to understand which interbank relationships matter most

This moves financial regulation from static rules to **dynamic, gradient-informed policy**.

6. Case Study 4: Input-Output (IO) and Computable General Equilibrium (CGE) Models

Focus: Sectoral Interdependencies and Trade

Input-Output models describe how sectors depend on each other: to produce cars, you need steel, which requires iron mining, electricity, labor, etc. Traditionally, IO tables are static "snapshots" of the economy.

The Problem

- **Fixed Coefficients:** Technical coefficients (how much steel per car) don't update as technologies change
- **Limited Scenarios:** "What-if" analysis requires manually rerunning the model with different assumptions

Benefit: Inverse IO Modeling

By making IO tables differentiable, we can:

1. **Learn Technical Coefficients from Data:** Use historical trade data as a training set. The model "learns" how coefficients shift over time by minimizing the error between predicted and observed trade flows.
2. **Policy Optimization:** Define a loss function (e.g., minimize carbon emissions while maintaining GDP growth), then use gradients to find the sectoral structure that optimizes this objective.

3. **Dynamic Rebalancing:** In real-time, as new data arrives, the model updates its technical coefficients and suggests optimal sectoral reallocation.

This transforms static IO tables into **living, learning models** that continuously calibrate to reality.

7. Case Study 5: Agent-Based Models (ABMs)

Focus: Micro-Foundations and Emergent Behavior

Agent-Based Models explicitly model individual agents (households, firms, banks) with discrete behavioral rules ("if wealth > threshold, invest in capital"). While realistic, this explodes model complexity and breaks differentiability.

The Challenge

Discrete "if/then" logic is inherently non-differentiable. You cannot compute a gradient through a decision tree.

The Solution: Relaxing Agents

Using techniques from machine learning (Gumbel-Softmax, Sigmoid approximations), we can replace discrete decisions with smooth, differentiable approximations:

- Instead of "if wealth > threshold, invest," use $\mathrm{i} = \mathrm{ioi}(\mathrm{l} - \mathrm{ol})$
- The agent behavior is now smooth and differentiable, while remaining interpretable

Benefit: Training Agent Populations

Rather than manually calibrating each agent's behavior, we can:

1. Initialize a population of agents with random behavioral parameters
2. Define a social welfare objective (e.g., minimize poverty + inequality + volatility)
3. Use gradient descent to **train the agent population** toward this objective
4. The result: an ABM that automatically discovers emergent behaviors that optimize society-wide goals

This is conceptually similar to how neural networks are trained—but applied to agent-based economies.

8. The "Holy Trinity" of Advantages

1. Analytic Sensitivity: The Jacobian of the Economy

At any point in time, the **Jacobian matrix** gives the first-order sensitivity:

$$= \mathrm{pP}$$

This matrix answers questions like:

- "By how much does unemployment respond to a 1% increase in government spending?"
- "If central bank interest rates rise, which financial institutions are most exposed?"
- "Which supply chain nodes have the highest cascading risk?"

Computing the full Jacobian traditionally requires thousands of simulations. With AD, it costs one backward pass.

2. Optimal Control: Policy via Gradient Descent

Define a loss function $L(\mathrm{policy})$ representing economic "health" or "welfare." Then simply:

$$\mathrm{policy} = \mathrm{policy}_{\mathrm{ol}} - \eta \nabla_{\mathrm{policy}} L$$

This is exactly how neural networks are trained. Now economists can train policies to reach multi-objective goals automatically.

3. Automated Calibration: Learning from History

Treat historical economic data (national accounts, trade, financial flows) as a training set. Define a loss:

$$L = \sum_t \mathrm{ol}_t(\theta) - {}_t{}^2$$

Then optimize:

$$\theta^* = \mathrm{i}_{\theta} L$$

The model automatically "learns" behavioral parameters that match historical reality, with no manual guessing.

9. Technical Obstacles & Philosophical Critiques

Non-Convexity

Economic loss landscapes are highly non-convex. Gradient descent may find local minima rather than global optima. **Mitigation strategies:**

- Ensemble methods (train multiple times with different initializations)
- Simulated annealing or temperature scheduling (see Section 10)
- Hybrid approaches combining local search with differentiable optimization

Discrete Events

Economic reality includes discontinuities: bankruptcies, policy regime shifts, defaults. Pure differentiability can't handle these. **Solutions:**

- Smooth relaxations (sigmoid approximations)
- Piecewise differentiable functions
- Event detection layers that trigger smooth transitions

The Lucas Critique

Robert Lucas argued that econometric relationships break down when policy changes. Does differentiability make it *easier* for policy-makers to over-optimize and ignore structural change?

Counter-argument: Differentiability provides transparency. We see exactly how policies couple to outcomes via gradients. This can *enhance* awareness of structural risks, not diminish it.

10. Future Horizon: Thermodynamic Tensor Methods

The Maslov-Gibbs Einsum (MGE) Framework

Emerging work applies thermodynamic tensor contraction (TTC) methods to economic modeling. Rather than a deterministic optimization, we introduce a "temperature" parameter (inverse temperature) that controls the "fuzziness" of economic decision-making.

The Role of Variable

- At $=$ (infinite temperature): All decisions are equally likely (maximum entropy)
- At (zero temperature): Decisions are deterministic and rigid (mimics discrete logic)
- At intermediate : A tunable balance between flexibility and structure

Annealing and Stability

By gradually increasing during training, we can:

1. Start with a high-entropy (flexible) system that explores many policy configurations
2. Gradually "cool" toward a deterministic solution
3. Identify phase transitions (points where the economy suddenly becomes unstable)
4. Navigate non-convex policy spaces more effectively

This approach reveals not just the optimal policy, but the **stability landscape** around it.

11. Conclusion: A Call for Differentiable National Accounts

Toward Differentiable Models as Public Infrastructure

We propose that central banks and statistical agencies maintain "differentiable codebases" for national economic models. Rather than running discrete simulations, these systems would be continuously differentiable, enabling:

- Real-time policy analysis
- Automated multi-objective optimization
- Integration with live national accounts data
- Transparency about policy-outcome relationships via gradients

The "Policy Cockpit" Vision

Imagine a 21st-century economic control center where policy-makers see not just the current state of the economy, but its **gradient landscape**. They can visualize:

- Which policy levers have the highest impact on unemployment or stability
- The cost (in terms of other objectives) of pursuing each policy direction
- The risks of straying from the optimal policy path
- How the landscape changes as new data arrives

The Climate-Economic Manifold

As governments navigate the transition to net-zero emissions while maintaining stability, differentiable economic models become essential. The optimal policy path minimizes not just unemployment and volatility, but also carbon emissions and biodiversity loss. This multi-objective optimization is computationally intractable without differentiability.

Conclusion: Differentiable programming is not a niche optimization technique—it is the foundation for 21st-century macroeconomic governance.

References

Godley, W., & Lavoie, M. (2012). *Monetary Economics: An Integrated Approach to Credit, Money, Income, Production and Wealth* (2nd ed.). Palgrave Macmillan.

Lavoie, M. (2014). *Post-Keynesian Economics: A New Guide to Heterodox Macroeconomics*. Edward Elgar.

Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32.

Griewank, A., & Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (2nd ed.). SIAM.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 281-305.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tesfatsion, L., & Judd, K. L. (Eds.). (2006). *Handbook of Computational Economics: Agent-Based Computational Economics* (Vol. 2). Elsevier.

Jørgensen, S. E. (Ed.). (2009). *System Ecology: Thermodynamic Bases for Complex Systems*. Oxford University Press.

Maddison, A. (2010). *Historical Statistics of the World Economy: 1-2008 AD*. OECD Development Centre.