idinn: A Python Package for Inventory-Dynamics Control with Neural Networks

Jiawei Li [1¶], **Thomas Asikis** [2], **Ioannis Fragkos** [3], and **Lucas Böttcher** [1,4]

1 Department of Computational Science and Philosophy, Frankfurt School of Finance and Management **2** Game Theory, University of Zurich **3** Department of Technology and Operations Management, Rotterdam School of Management, Erasmus University Rotterdam **4** Laboratory for Systems Medicine, Department of Medicine, University of Florida ¶ Corresponding author

DOI: 10.21105/joss.08508

Software

- Review
- Repository
- Archive

Editor: Evan Spotte-Smith

Reviewers:

- @arnauqb
- @oak97

Submitted: 17 April 2025
Published: 20 August 2025

Summary

Identifying optimal policies for replenishing inventory from multiple suppliers is a key problem in inventory management. Solving such optimization problems requires determining the quantities to order from each supplier based on the current inventory and outstanding orders, minimizing the expected ordering, holding, and out-of-stock costs. Despite over 60 years of extensive research on inventory management problems, even fundamental dual-sourcing problems—where orders from an expensive supplier arrive faster than orders from a low-cost supplier—remain analytically intractable (Barankin, 1961; Fukuda, 1964). Additionally, there is a growing interest in optimization algorithms that can handle real-world inventory problems with non-stationary demand (Song et al., 2020).

We provide a Python package, `idinn`, which implements inventory dynamics-informed neural networks designed to control both single-sourcing and dual-sourcing problems. In single-sourcing problems, a single supplier delivers an ordered quantity to a firm within a known lead time and at a known unit cost. In dual-sourcing problems, which are more complex, a company has two potential suppliers of a product, each with different known lead times and unit costs. The objective is to place orders that minimize the expected order, inventory, and out-of-stock costs over a finite or infinite horizon. `idinn` implements neural network controllers and inventory dynamics as customizable objects using PyTorch as the backend, allowing users to identify near-optimal ordering policies with reasonable computational resources.

The methods used in `idinn` take advantage of advances in automatic differentiation (Paszke et al., 2019, 2017) and the growing use of neural networks in dynamical system identification (Chen et al., 2018; Fronk & Petzold, 2023; Wang & Lin, 1998) and control (Asikis et al., 2022; Böttcher et al., 2022, 2025; Böttcher, 2023; Böttcher & Asikis, 2022; Mowlavi & Nabi, 2023).

Statement of need

Inventory management problems arise in many industries, including manufacturing, retail, warehousing, and energy. A fundamental but analytically intractable inventory management problem is dual sourcing (Barankin, 1961; Fukuda, 1964; Xin & Van Mieghem, 2023). Single sourcing, in contrast, is analytically tractable and is frequently employed as a baseline for testing and comparing policies used in more complex multi-sourcing setups. `idinn` is a Python package for controlling both dual-sourcing and single-sourcing problems using dynamics-informed neural networks. The sourcing problems we consider are usually formulated as infinite-horizon problems focusing on minimizing average cost while considering stationary stochastic demand. Using neural networks, we minimize costs over multiple demand trajectories. This approach allows us to address not only non-stationary demand, but also finite-horizon and infinite-horizon discounted problems. Unlike traditional reinforcement-learning approaches, our optimization approach takes into account how the system to be optimized behaves over time, leading to more efficient training and accurate solutions.

Training neural networks for inventory-dynamics control presents a unique challenge. The adjustment of neural network weights during training relies on propagating real-valued gradients, while the neural network outputs - representing replenishment orders - must be integers. To address this challenge in optimizing a discrete problem with real-valued gradient-descent learning algorithms, we apply a problem-tailored straight-through estimator (Asikis, 2023; Dyer et al., 2023; Yang et al., 2022). This approach enables us to obtain integer-valued neural network outputs while backpropagating real-valued gradients.

While general-purpose reinforcement learning libraries like `Stable Baselines3` (Raffin et al., 2021) support policy optimization, they do not offer inventory-specific modeling or enforce integer constraints required for replenishment decisions. A related, more specialized library, `ddop` (Philippi et al., 2021), focuses on machine learning for operations management, but does not address multi-period inventory dynamics. `Abmarl` (Rusu & Glatt, 2021) connects multi-agent reinforcement learning with agent-based simulations. In contrast, `idinn` is tailored to single- and dual-sourcing problems, combining domain-specific inventory dynamics with neural networks and integer-constrained outputs.

`idinn` has been developed for researchers, practitioners, and students working at the intersection of optimization, operations research, and machine learning. It has been made available to students in a machine learning course at the Frankfurt School of Finance & Management, as well as in a tutorial at the California State University, Northridge. In a previous publication (Böttcher et al., 2023), a proof-of-concept codebase was used to compute near-optimal solutions of dozens of dual-sourcing instances.

Example usage

As an example, we describe how to solve single-sourcing problems using `idinn`. Excess inventory incurs a holding cost h, while unmet demand results in an out-of-stock cost b. With `idinn`, we initialize the sourcing model and neural network controller, train the controller on cost data from the model, and use the trained controller to compute near-optimal, state-dependent order quantities.

Initialization

We use the `SingleSourcingModel` class to initialize a single-sourcing model.

```python
import torch
from idinn.sourcing_model import SingleSourcingModel
from idinn.single_controller import SingleSourcingNeuralController
from idinn.demand import UniformDemand

single_sourcing_model = SingleSourcingModel(
    lead_time=0,
    holding_cost=5,
    shortage_cost=495,
    batch_size=32,
    init_inventory=10,
    demand_generator=UniformDemand(low=0, high=4)
)
```

This single-sourcing model has a lead time of 0 (i.e., an order arrives immediately after it is placed) and an initial inventory of 10. The holding cost, h, and the out-of-stock cost, b, are 5 and 495, respectively. Demand is drawn from a discrete uniform distribution over the integers $\{0, 1, \dots, 4\}$. We use a batch size of 32 to train the neural network, i.e., the sourcing model generates 32 samples simultaneously.

To identify an ordering policy that minimizes total costs over a given time horizon, we initialize a neural network controller using the `SingleSourcingNeuralController` class. For illustration purposes, we use a simple neural network with 1 hidden layer and 2 neurons. The activation function is `torch.nn.CELU(alpha=1)`.

```
single_controller = SingleSourcingNeuralController(
    hidden_layers=[2],
    activation=torch.nn.CELU(alpha=1)
)
```

Training

We train the neural network controller using the `fit()` method. To monitor the training process, we specify the `tensorboard_writer` parameter:

```
from torch.utils.tensorboard import SummaryWriter

single_controller.fit(
    sourcing_model=single_sourcing_model,
    sourcing_periods=50,
    validation_sourcing_periods=1000,
    epochs=2000,
    tensorboard_writer=SummaryWriter(comment="_single_1"),
    seed=1
)
```

To evaluate the neural network controller, we compute the average cost over a specified number of periods for the previously defined sourcing model:

```
single_controller.get_average_cost(single_sourcing_model, sourcing_periods=1000)
```

For the selected single-sourcing parameters, the optimal average cost is 10.

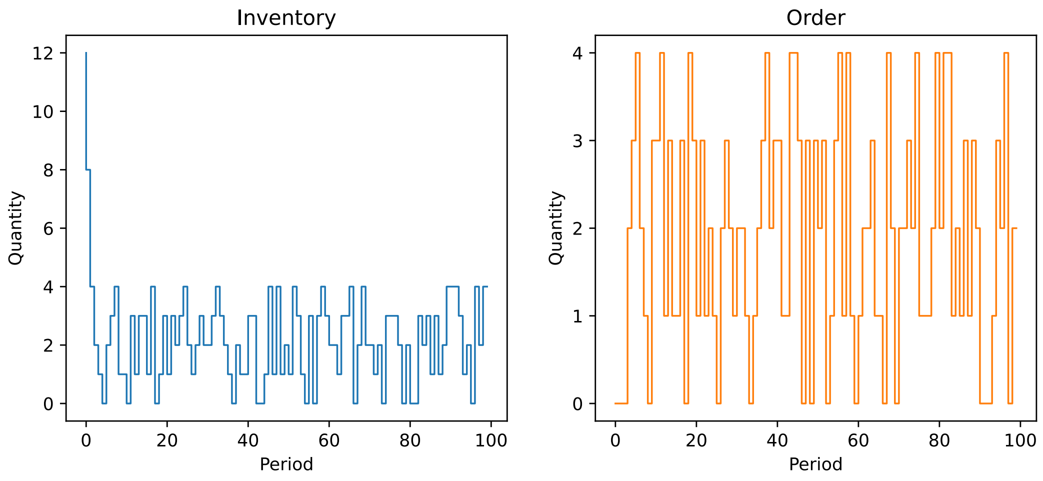

Figure 1: Evolution of inventory and orders for a neural network controller applied to single-sourcing dynamics.

To further evaluate a controller's performance in a given sourcing environment, users can visualize the inventory and order histories (see Figure 1):

```
single_controller.plot(sourcing_model=single_sourcing_model, sourcing_periods=100)
```

Order calculation

For a given inventory level and trained controller, we use the `predict` function to compute the corresponding orders:

```
single_controller.predict(current_inventory=10)
```

Additional control methods

In addition to the neural network control method, single-sourcing dynamics can also be managed using a traditional base-stock controller (Arrow et al., 1951; Scarf & Karlin, 1958), available in `idinn` as the `BaseStockController` class.

For dual-sourcing problems, `idinn` supports both neural and classical control approaches. Neural control is provided through the `DualSourcingModel` and `DualSourcingNeuralController` classes. Classical methods include the capped dual index controller (Sun & Van Mieghem, 2019) and a dynamic programming-based controller, implemented in the `CappedDualIndexController` and `DynamicProgrammingController` classes, respectively.

Acknowledgements

LB acknowledges financial support from hessian.AI and the Army Research Office (grant W911NF-23-1-0129). TA acknowledges financial support from the Schweizerischer Nationalfonds zur Förderung der Wissenschaftlichen Forschung through NCCR Automation (grant P2EZP2 191888).

References

Arrow, K. J., Harris, T., & Marschak, J. (1951). Optimal inventory policy. *Econometrica*, *19*(3), 250–272. https://doi.org/10.2307/1906813

Asikis, T. (2023). Towards recommendations for value sensitive sustainable consumption. *NeurIPS 2023 Workshop on Tackling Climate Change with Machine Learning: Blending New and Existing Knowledge Systems*. https://nips.cc/virtual/2023/76939

Asikis, T., Böttcher, L., & Antulov-Fantulin, N. (2022). Neural ordinary differential equation control of dynamics on graphs. *Physical Review Research*, *4*(1), 013221. https://doi.org/10.1103/PhysRevResearch.4.013221

Barankin, E. (1961). A delivery-lag inventory model with an emergency provision. *Naval Research Logistics Quarterly*, *8*, 285–311. https://doi.org/10.1002/nav.3800080310

Böttcher, L. (2023). Gradient-free training of neural ODEs for system identification and control using ensemble Kalman inversion. *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems, Honolulu, HI, USA, 2023*.

Böttcher, L., Antulov-Fantulin, N., & Asikis, T. (2022). AI Pontryagin or how artificial neural networks learn to control dynamical systems. *Nature Communications*, *13*(1), 1–9. https://doi.org/10.1038/s41467-021-27590-0

Böttcher, L., & Asikis, T. (2022). Near-optimal control of dynamical systems with neural ordinary differential equations. *Machine Learning: Science and Technology*, *3*(4), 045004. https://doi.org/10.1088/2632-2153/ac92c3

Böttcher, L., Asikis, T., & Fragkos, I. (2023). Control of dual-sourcing inventory systems using recurrent neural networks. *INFORMS Journal on Computing*, *35*(6), 1308–1328.

https://doi.org/10.1287/ijoc.2022.0136

Böttcher, L., Fonseca, L. L., & Laubenbacher, R. C. (2025). Control of medical digital twins with artificial neural networks. *Philosophical Transactions A*, *383*(2292), 20240228. https://doi.org/10.1098/rsta.2024.0228

Chen, T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (pp. 6572–6583). https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html

Dyer, J., Quera-Bofarull, A., Chopra, A., Farmer, J. D., Calinescu, A., & Wooldridge, M. J. (2023). Gradient-assisted calibration for financial agent-based models. *4th ACM International Conference on AI in Finance, ICAIF 2023, Brooklyn, NY, USA, November 27-29, 2023*, 288–296. https://doi.org/10.1145/3604237.3626857

Fronk, C., & Petzold, L. (2023). Interpretable polynomial neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *33*(4). https://doi.org/10.1063/5.0130803

Fukuda, Y. (1964). Optimal policies for the inventory problem with negotiable leadtime. *Management Science*, *10*(4), 690–708. https://doi.org/10.1287/mnsc.10.4.690

Mowlavi, S., & Nabi, S. (2023). Optimal control of PDEs using physics-informed neural networks. *Journal of Computational Physics*, *473*, 111731. https://doi.org/10.1016/j.jcp.2022.111731

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. *NIPS 2017 Autodiff Workshop*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (pp. 8024–8035). https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Philippi, A., Buttler, S., & Stein, N. (2021). Ddop: A Python package for data-driven operations management. *Journal of Open Source Software*, *6*(66), 3429. https://doi.org/10.21105/joss.03429

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1–8. http://jmlr.org/papers/v22/20-1364.html

Rusu, E., & Glatt, R. (2021). Abmarl: Connecting agent-based simulations with multi-agent reinforcement learning. *Journal of Open Source Software*, *6*(64), 3424. https://doi.org/10.21105/joss.03424

Scarf, H., & Karlin, S. (1958). Inventory models of the Arrow-Harris-Marschak type with time lag. In K. J. Arrow, S. Karlin, & H. E. Scarf (Eds.), *Studies in the mathematical theory of inventory and production*. Stanford University Press.

Song, J.-S., Van Houtum, G.-J., & Van Mieghem, J. A. (2020). Capacity and inventory management: Review, trends, and projections. *Manufacturing & Service Operations*

Management, *22*(1), 36–46. https://doi.org/10.1287/msom.2019.0798

Sun, J., & Van Mieghem, J. A. (2019). Robust dual sourcing inventory management: Optimality of capped dual index policies and smoothing. *Manufacturing & Service Operations Management*, *21*(4), 912–931. https://doi.org/10.1287/msom.2018.0731

Wang, Y.-J., & Lin, C.-T. (1998). Runge–Kutta neural network for identification of dynamical systems in high accuracy. *IEEE Transactions on Neural Networks*, *9*(2), 294–307. https://doi.org/10.1109/72.661124

Xin, L., & Van Mieghem, J. A. (2023). Dual-sourcing, dual-mode dynamic stochastic inventory models. In *Research Handbook on Inventory Management* (pp. 165–190). Edward Elgar Publishing. https://doi.org/10.4337/9781800377103.00015

Yang, Z., Lee, J., & Park, C. (2022). Injecting logical constraints into neural networks via straight-through estimators. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, & S. Sabato (Eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA* (Vol. 162, pp. 25096–25122). PMLR. https://proceedings.mlr.press/v162/yang22h.html