Advanced Topics in Macro, 2025

Course based on Computational Methods for Macroeconomic Modeling by J. Stachurski

Joao B. Sousa

October 2025

About the course

- Modern Computing Tools: Parallelization, automatic differentiation, and JIT compilation in scientific computing.
- Applications: Using these tools for empirical and quantitative economic analysis.
- Slides, notebooks, code based on a course by John Stachurski (https://github.com/QuantEcon/bank_of_portugal_2025)
- Classroom: coding and exercises. Bring your laptop!
- Course Requirement: Submit a replication package of a published paper using the tools learned in the course.

Topics

1. Introduction
2. AI systems
3. ANNs and deep learning
4. Computational methods
5. Economic applications

The AI revolution

- generative AI (LLMs, image / music / video)
- image processing / computer vision
- speech recognition
- translation
- scientific knowledge discovery
- forecasting and prediction
- etc.

Example: AlphaEvolve

A coding agent for scientific and algorithmic discovery

Google Deepmind May 2025

Features

- Employs an evolutionary algorithm
- Asks an ensemble of LLMs and then iterates, tests, refines

Process

1. Proposed solutions evaluated
2. Promising solutions are selected and mutated by LLMs
3. “Survival of the fittest” progressively improves performance

Outcomes at Google:

- Enhanced efficiency in chip design (TPUs)
- Improved data center scheduling
- Discovered new matrix multiplication algorithms (surpassing Strassen's algorithm for 4x4 complex matrices)

Example: AlphaFold

- AI system by Google DeepMind
- Predicts 3D protein structures from amino acid sequences

2024 Nobel Prize in Chemistry awarded to

- Demis Hassabis
- John Jumper

(For development of AlphaFold)

Gemini 2.5 Pro

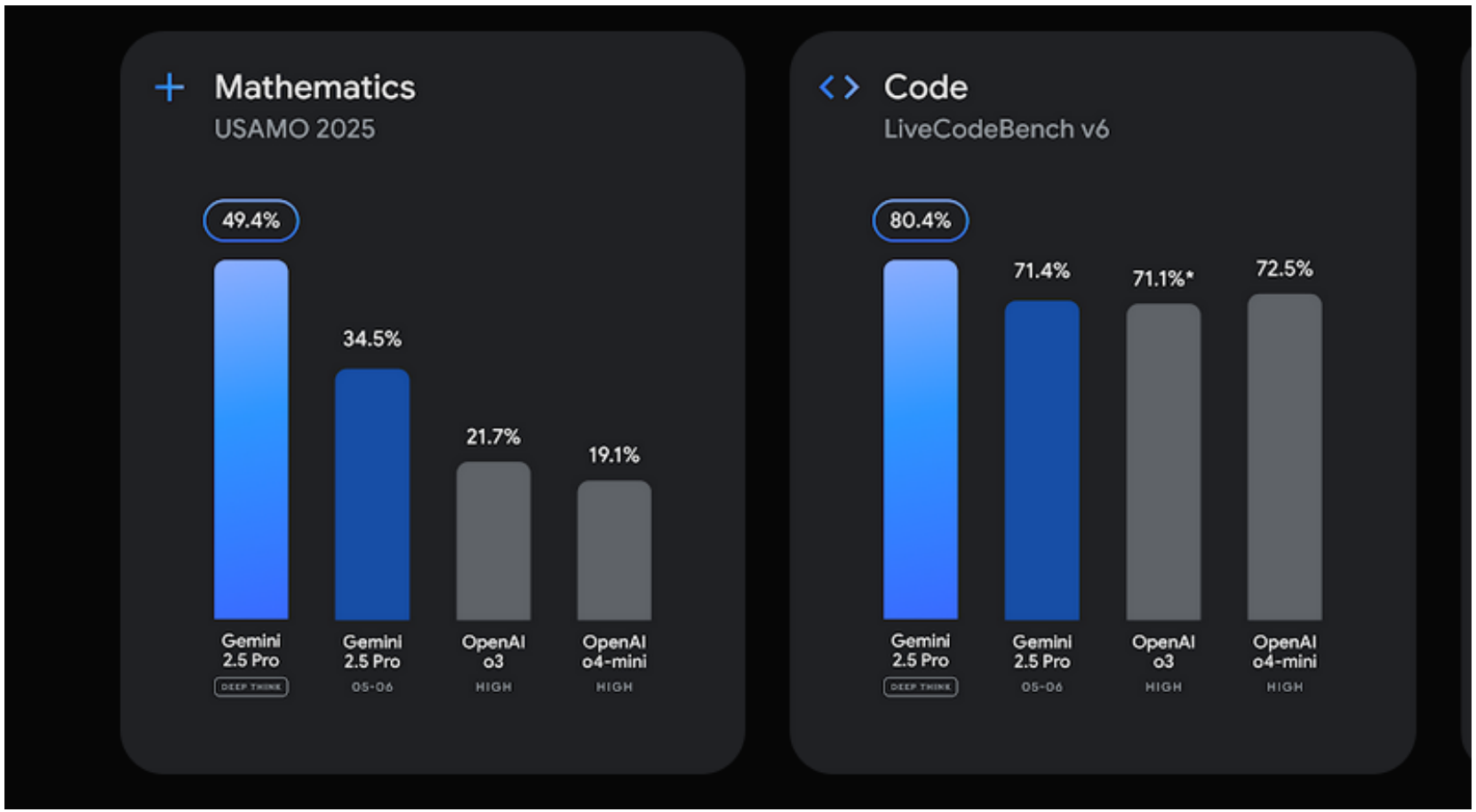
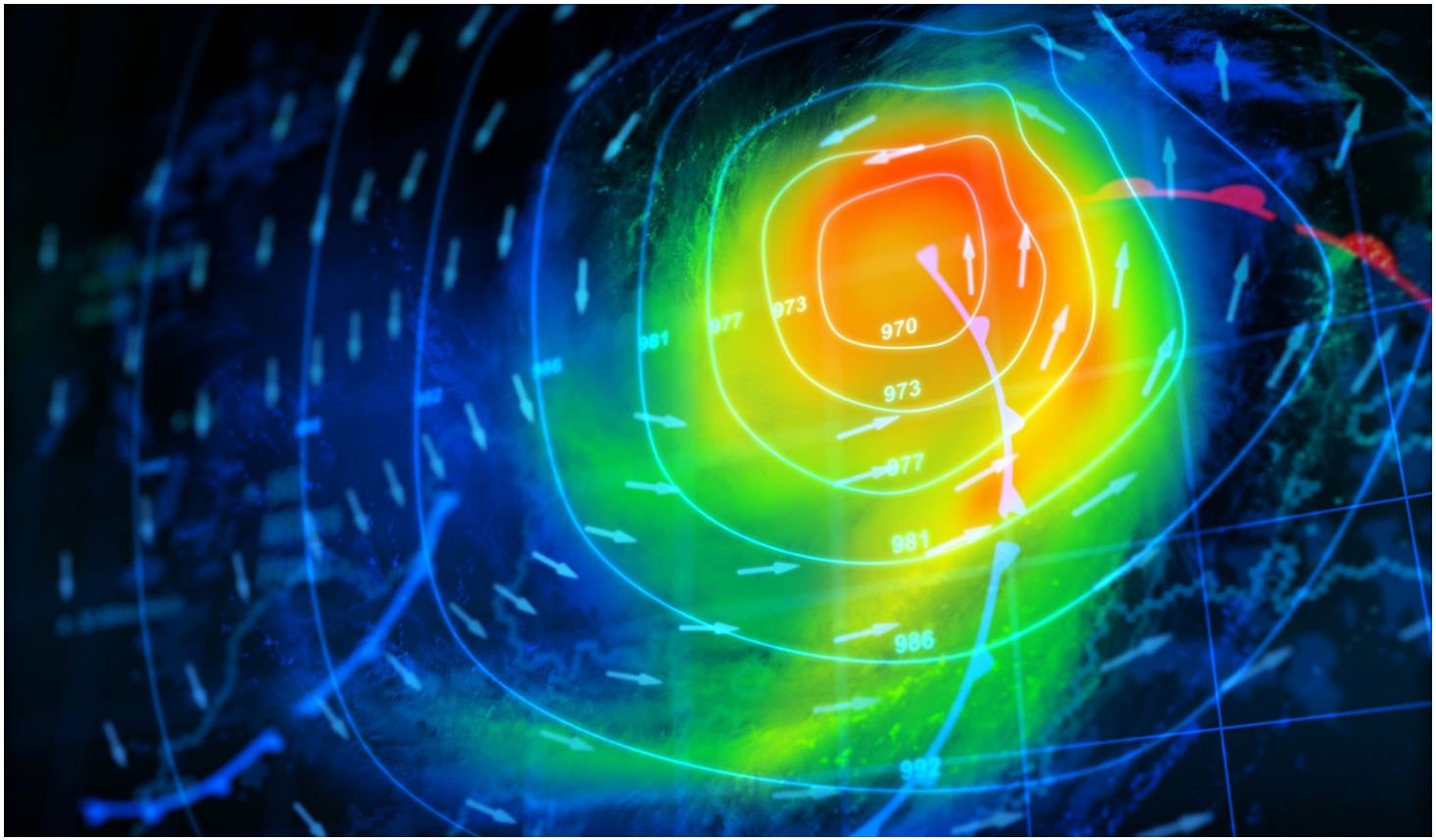


Image Generators

Video Generators

Weather forecasts



"ECMWF's weather forecasting model is considered the gold standard for medium-term weather forecasting...Google DeepMind claims to beat it 90% of the time..."

"Traditional forecasting models are big, complex computer algorithms based on atmospheric physics and take hours to run. AI models can create forecasts in just seconds."

Source: MIT Technology Review

Killer drones, Skynet, etc.

Investment

Private AI investment by US firms in 2024 = $109 billion USD

Estimate for 2025 = $615 billion

Massive investments in

- data centers
- server / GPU / TPU design and production
- software development

What kinds of problems are they trying to solve?

Statistical learning (induction)

We observe input-output pairs (x, y), where

- $x \in \mathbb{R}^k$
- $y \in \mathbb{R}$ (for example)

Examples.

- $x =$ cross section of returns today, $y =$ return on oil futures tomorrow
- $x =$ weather sensor data today, $y =$ max temp tomorrow

Problem:

- observe $(x_i, y_i)_{i=1}^n$ and seek f such that $y_{n+1} \approx f(x_{n+1})$

Deep Learning (DL)

Training:

1. Choose function class $\{f_\theta\}_{\theta \in \Theta}$
2. Minimize loss

$$\ell(\theta) := \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2 \qquad \text{s.t.} \qquad \theta \in \Theta$$

In the case of DL, elements of $\{f_\theta\}_{\theta \in \Theta}$ have a particular structure

- each f_θ is a neural net — we discuss more soon
- typically, $\theta \mapsto f_\theta(x)$ is smooth for all x
- MSE is a popular loss function but others are also used

Deep Learning (DL)

Training:

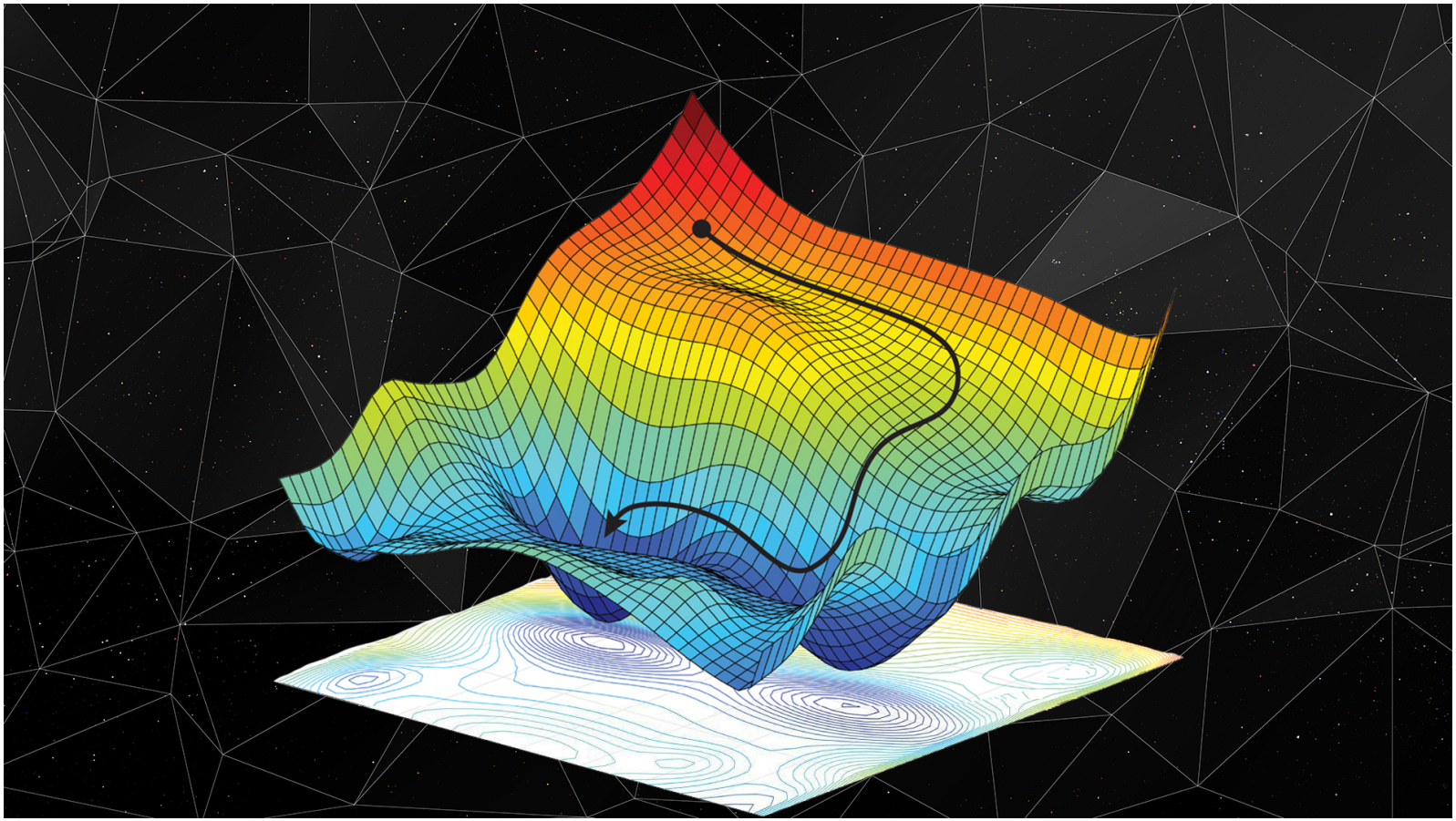
1. Choose function class $\{f_\theta\}_{\theta \in \Theta}$
2. Minimize loss

$$\ell(\theta) := \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2 \qquad \text{s.t.} \qquad \theta \in \Theta$$

In the case of DL, elements of $\{f_\theta\}_{\theta \in \Theta}$ have a particular structure

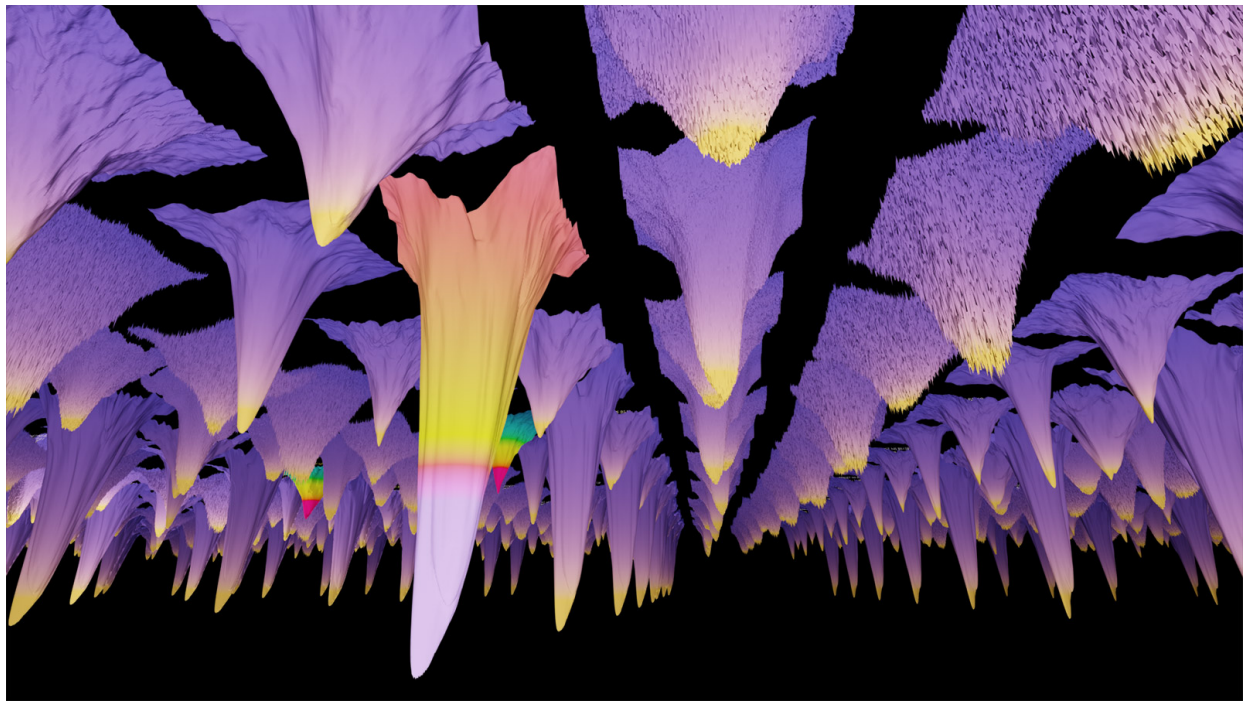
- each f_θ is a neural net — we discuss more soon
- typically, $\theta \mapsto f_\theta(x)$ is smooth for all x
- MSE is a popular loss function but others are also used

Minimizing a smooth loss functions – what algorithm?



Source: https://danielkhv.com/

Deep learning: $\theta \in \mathbb{R}^d$ where $d = ?$



Source: https://losslandscape.com/gallery/

How does it work?

How is it possible to minimize loss over such high dimensions??

Core elements

1. parallelization over powerful hardware (GPUs or TPUs)
2. automatic differentiation (for gradient descent)
3. Compilers / JIT-compilers for fast parallelized machine code

How does it work?

How is it possible to minimize loss over such high dimensions??

Core elements

1. parallelization over powerful hardware (GPUs or TPUs)
2. automatic differentiation (for gradient descent)
3. Compilers / JIT-compilers for fast parallelized machine code

Parallelization

- Vectorization on one accelerator

```
def function(data_point):
    # perform a calculation on one data point
    return output

vectorized_function = vmap(function)
# Perform the same action on the whole data set
outputs = vectorized_function(data_set)
```

- Data parallelism across multiple accelerators

```
parallel_function = pmap(vectorized_function)
all_outputs = parallel_function(data_sets)
```

Note: Full GPU-Python integration is on the way!

At GTC 2025, NVIDIA announced native support and full integration of Python in its CUDA toolkit

Over the last year, NVIDIA made CUDA Core – a "Pythonic reimagining of the CUDA runtime to be naturally and natively Python."

https://thenewstack.io/nvidia-finally-adds-native-python-support-to-cuda/

Automatic differentiation

```
from jax import grad

def f( , x):
    # add details here
    return prediction

def loss( , x, y):
  return jnp.sum((y - f( , x))**2)

loss_gradient = grad(loss)    # exact automatic
 ↪  differentiation
  =   -   * loss_gradient( , x_data, y_data)
```

Just-in-time compilers

```
@jax.jit
def f(x):
    return jnp.sin(x) - jnp.cos(x**2)
```

Advantages over AOT compilers:

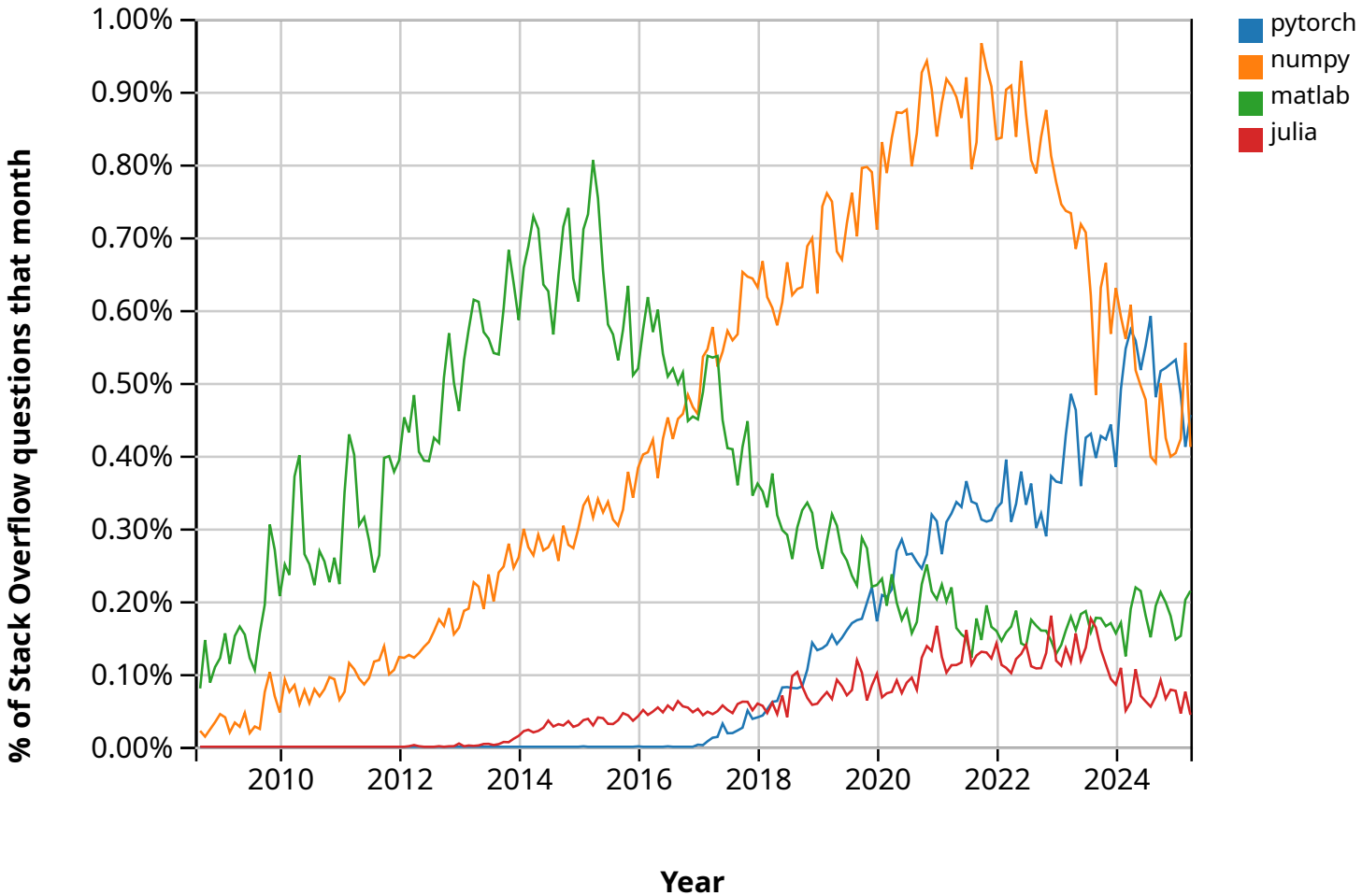
- more adaptable
- more portable
- detects and adapts to existing hardware
- automatic parallelization (same code for CPUs / GPUs)

Platforms

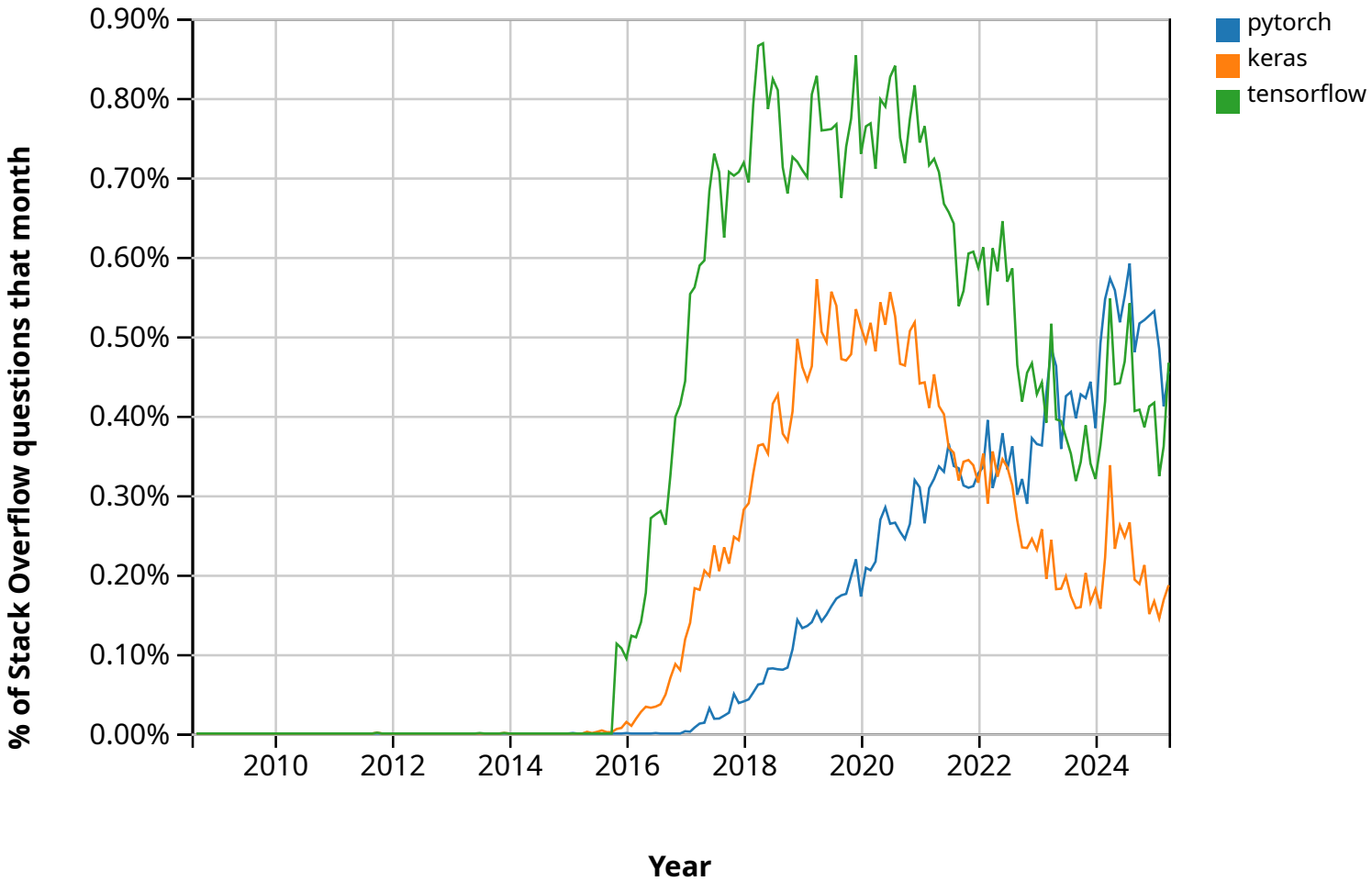
Platforms that support AI / deep learning:

- Tensorflow
- PyTorch (Llama, ChatGPT)
- Google JAX (Gemini, DeepMind)
- Keras (backends = JAX, PyTorch)
- Mojo (Modular (Python))
- MATLAB???

Popularity – languages and libraries



Popularity – DL / ML frameworks



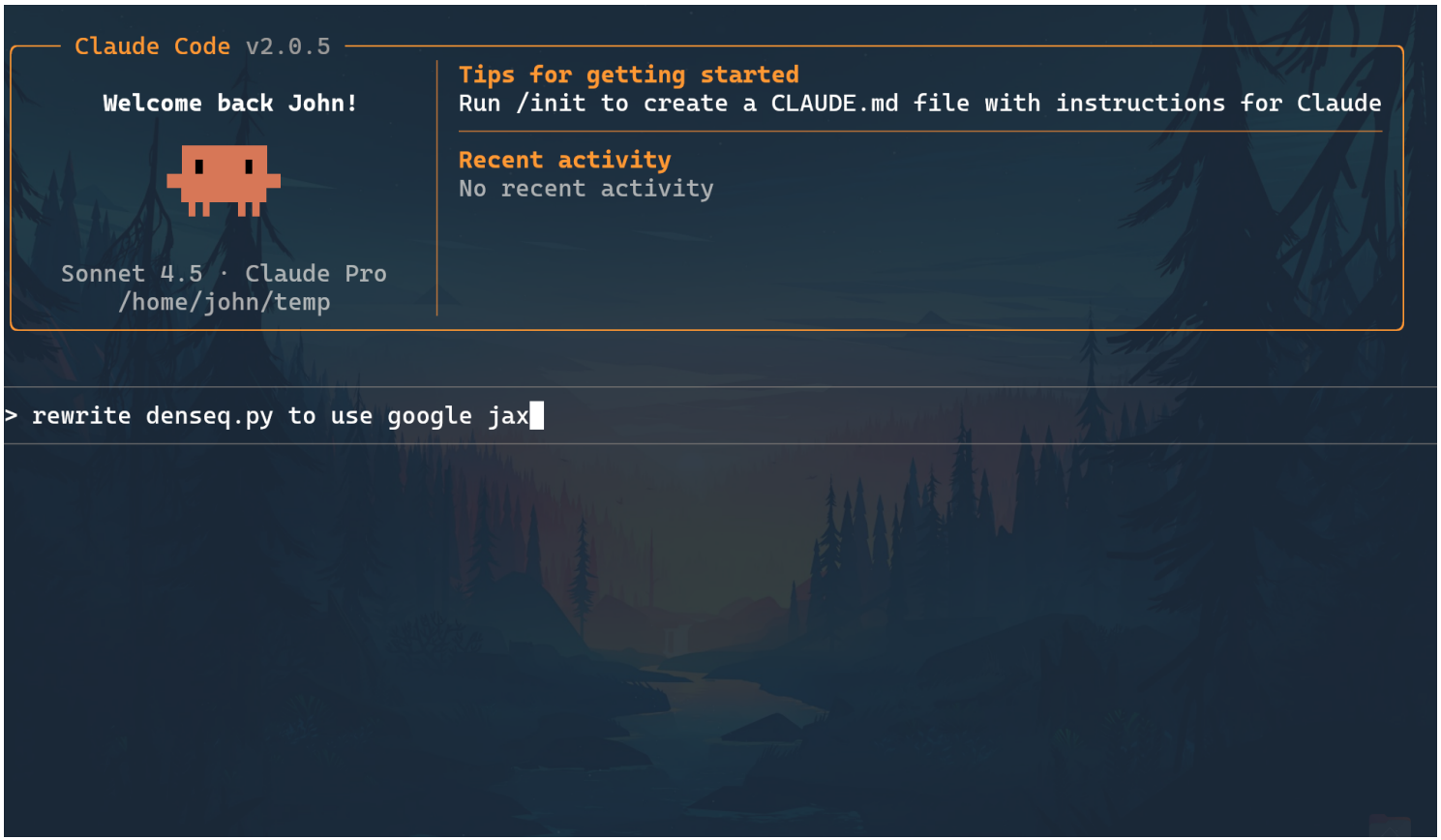
Coding with AI

J please write a function for gradient descent in 6 lines or less

Here's a gradient descent function in 6 lines:

python

```python
def gradient_descent(grad_func, init_params, lr=0.01, iters=100):
    params = init_params.copy()
    for _ in range(iters):
        gradient = grad_func(params)
        params = params - lr * gradient
        if np.linalg.norm(lr * gradient) < 1e-6: break
    return params
```


Claude Code v2.0.5
Welcome back John!
Sonnet 4.5 · Claude Pro
/home/john/temp
Tips for getting started
Run /init to create a CLAUDE.md file with instructions for Claude
Recent activity
No recent activity
> rewrite denseq.py to use google jax

AI code generation is great...but not perfect

Thoughts from pro developer `Lonely-Public2655`

- AI doesn't see the big picture
- Can ace small tasks but struggles to connect them
- You still need to be the architect
- Context is fragile: AI forgets
- Once things get weird, AI starts guessing
- Sometimes AI gets really weird

AI coding affects optimal language choice

Claude Sonnet 4:

"I'm definitely stronger with Python than MATLAB."

"My capabilities with Python are more comprehensive. I have deeper familiarity with Python's extensive ecosystem of libraries, frameworks, and modern development practices."

"I can more confidently help with advanced Python topics, debugging complex Python code, and implementing Python best practices."

"I'm definitely stronger with Python than Julia."

"Python is one of my most proficient languages - I have deep familiarity with its syntax, libraries, frameworks, and best practices across many domains including data science, web development, machine learning, and general-purpose programming."

"While I understand Julia's syntax and core concepts, my expertise with it isn't as comprehensive as with Python."

AI tools for economic modeling

Let's say that you want to do computational economics without deep learning

Can these new AI tools be applied?

Yes!

- fast matrix algebra
- fast solutions to linear systems
- fast nonlinear system solvers
- fast optimization, etc.

AI tools for economic modeling

Let's say that you want to do computational economics without deep learning

Can these new AI tools be applied?

Yes!

- fast matrix algebra
- fast solutions to linear systems
- fast nonlinear system solvers
- fast optimization, etc.

Case Study

The CBC uses the "overborrowing" model of Bianchi (2011)

- credit constraint loosens during booms
- bad shocks $\rightarrow$ sudden stops

CBC implementation in MATLAB

- runs on \$10,000 mainframe with 356 CPUs and 1TB RAM
- runtime = 12 hours

Rewrite in Python + Google JAX (J. Stachurski)

- runs on \$400 gaming GPU with 10GB RAM
- runtime = 7 seconds

Case Study

The CBC uses the "overborrowing" model of Bianchi (2011)

- credit constraint loosens during booms
- bad shocks $\rightarrow$ sudden stops

CBC implementation in MATLAB

- runs on \$10,000 mainframe with 356 CPUs and 1TB RAM
- runtime = 12 hours

Rewrite in Python + Google JAX (J. Stachurski)

- runs on \$400 gaming GPU with 10GB RAM
- runtime = 7 seconds

Summary

- We are at the start of a massive AI revolution
- This revolution will have a huge impact on science
- What impact on economics?

Aims

- Better understanding of core AI methods
- Better understanding of core tools (hardware / software)
- Apply knowledge to current economic modeling

Summary

- We are at the start of a massive AI revolution
- This revolution will have a huge impact on science
- What impact on economics?

Aims

- Better understanding of core AI methods
- Better understanding of core tools (hardware / software)
- Apply knowledge to current economic modeling