

# Computational Economics and the AI Revolution

John Stachurski

2025

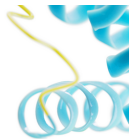
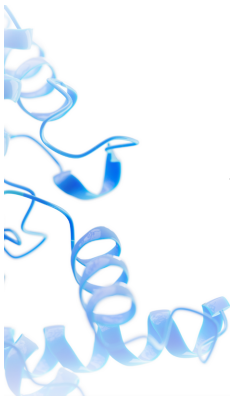
# Topics

- Introduction
- ANNs and deep learning
- Stochastic approximation
- Dynamic programming
- Reinforcement learning
- Economic applications

# AI-driven scientific computing

AI is changing the world

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



# AlphaFold

Accelerating breakthroughs in biology with AI


[Explore the AlphaFold Database >](#)

# AlphaFold

- AI system by DeepMind that predicts protein structures
- Open-sourced code and database for global access
- Accelerating drug discovery and design
- Enabling research on diseases like cancer and Alzheimer's
- Supporting enzyme engineering for sustainability

2024 Nobel Prize in Chemistry awarded to Demis Hassabis and John Jumper for development of AlphaFold

# LLMs

 is john stachurski smart?

Based on what I know, John Stachurski is considered quite accomplished in his field. He's a professor of economics known for his work in economic theory, computational economics, and machine learning applications in economics.

He's particularly recognized for developing QuantEcon, an open-source project that provides resources for quantitative economics, which demonstrates both technical expertise and a commitment to education and open access to knowledge.

His contributions to economic modeling, particularly dynamic economic models, suggest strong technical and mathematical abilities. His published research and textbooks are well-regarded in the academic community.

Would you like me to search for more recent information about John Stachurski's work and contributions?



   Retry 

Claude can make mistakes. Please double-check responses.

# Image Generators



# Translation Engines

English

↔

Japanese

John Stachurski is quite accomplished in his field

×

ジョン・スタチャー  
スキーは彼の分野で  
非常に優れた実績を  
持っている  
Jon sutachāsukī wa kare no bun'ya  
de hijō ni sugureta jisseki o motte iru

🔊

📄 🔊 🌐

[Open in Google Translate](#) • [Feedback](#)



Killer drones, Skynet, etc.



# Investment

Private AI investment in 2024:

- U.S. = \$109 billion
- China \$9.3 billion
- UK \$4.5 billion

Massive investments in

- data centers
- GPU design and production
- software development

What kinds of problems are they trying to solve?

# Deep learning in two slides

Aim: approximate an unknown functional relationship

$$y = f(x) \quad (x \in \mathbb{R}^k, y \in \mathbb{R})$$

## 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 is nonlinear regression:

1. Choose function class  $\{f_\theta\}_{\theta \in \Theta}$
2. Minimize loss over the parameter space, where

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

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

- We discuss this structure 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 is nonlinear regression:

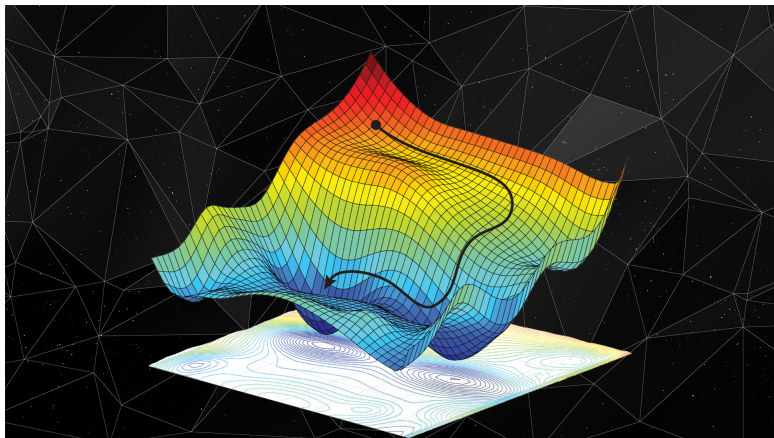
1. Choose function class  $\{f_\theta\}_{\theta \in \Theta}$
2. Minimize loss over the parameter space, where

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

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

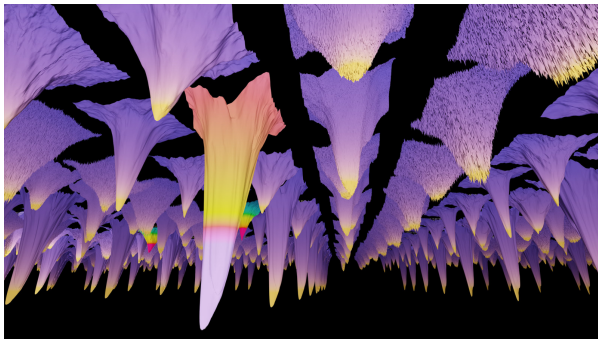
- We discuss this structure soon
- Typically,  $\theta \mapsto f_\theta(x)$  is smooth for all  $x$
- MSE is a popular loss function but others are also used

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?

Why is it possible to minimize over  $\theta \in \mathbb{R}^d$  when  $d = 10^{12}$  ?!?

## Core elements

- automatic differentiation (for gradient descent)
- parallelization (GPUs or TPUs)
- Compilers / JIT-compilers

# How does it work?

Why is it possible to minimize over  $\theta \in \mathbb{R}^d$  when  $d = 10^{12}$  ?!?

## Core elements

- automatic differentiation (for gradient descent)
- parallelization (GPUs or TPUs)
- Compilers / JIT-compilers

# Automatic differentiation

“Exact numerical” differentiation

```
def loss( $\theta$ , x, y):  
    return jnp.sum((y - f( $\theta$ , x))**2)  
  
loss_gradient = grad(loss)  
d $\theta$  = loss_gradient( $\theta$ , x_data, y_data)  
 $\theta$  =  $\theta$  -  $\lambda$  * d $\theta$ 
```

## Parallelization



```
outputs = pmap(f, data)
```

- multithreading over GPU cores (how many?)
- multiprocessing over accelerators in a GPU farm / supercomputing cluster (how many?)

# Just-in-time compilers

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

Advantages over AOT compilers:

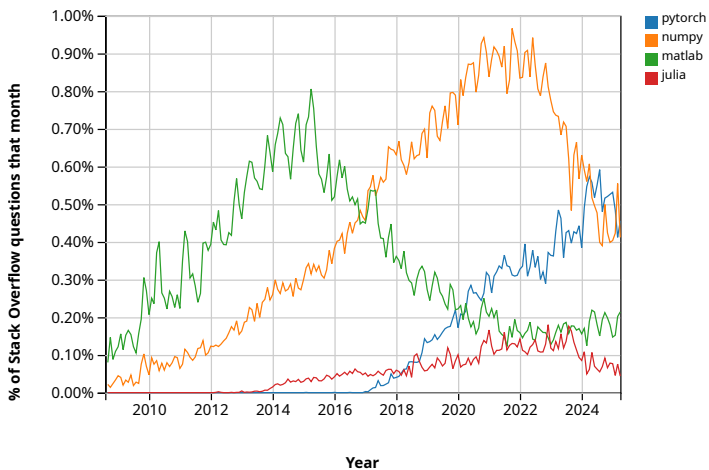
- cleaner code
- more portable
- automatic parallelization (same code for CPUs / GPUs)

# Platforms

Platforms that support AI / deep learning:

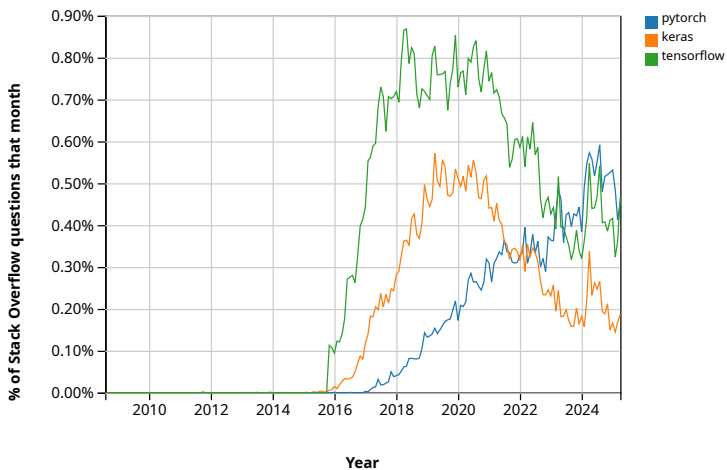
- Tensorflow
- PyTorch (Llama, ChatGPT)
- Google JAX (Gemini, DeepMind)
- Keras (backends = JAX, PyTorch)
- Mojo? (Modular (Python))
- MATLAB?

## Popularity – languages





## Popularity – deep learning frameworks



# AI for coding

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

Here's a gradient descent function in 6 lines:

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
```

## Affects language choice

“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.”

## Thoughts from professional developer Lonely-Public2655

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

# 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! Yes! 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! Yes! 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 → 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

- 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 → 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

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