

Making Your Models Fly

Computational Tools for Neuroeconomic Modeling

```
import numpy as np, jax.numpy as jnp
x = np.linspace(-1, 1, 1000000)
y = jnp.sin(x) + x**2
```

Gilles de Hollander

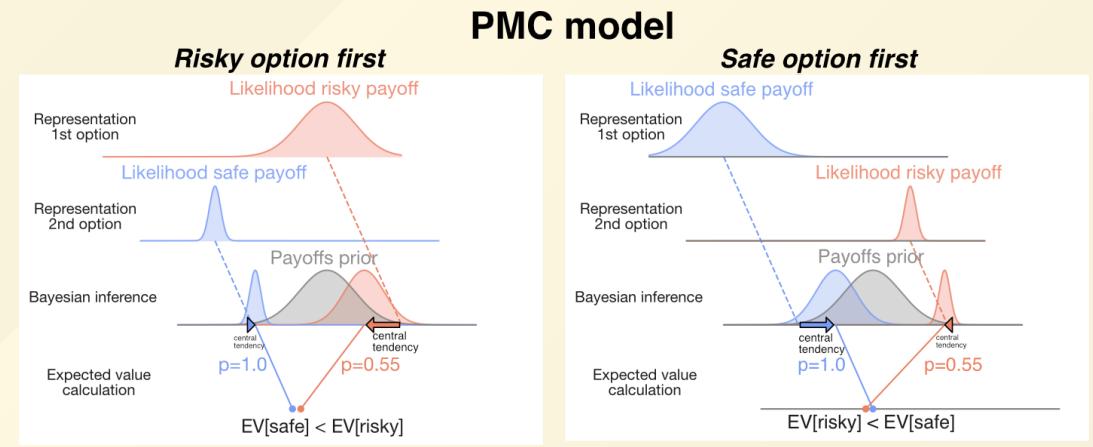
Today

- Why bother?
- A bit of computer science
- Computational graphs
- Optimizing code
- Example 1: Encoding models (e.g., PRF models)
- Example 2: Efficient coding models

Why bother?

Neuroeconomics is full of complex models

- Cognitive models
- Neural models

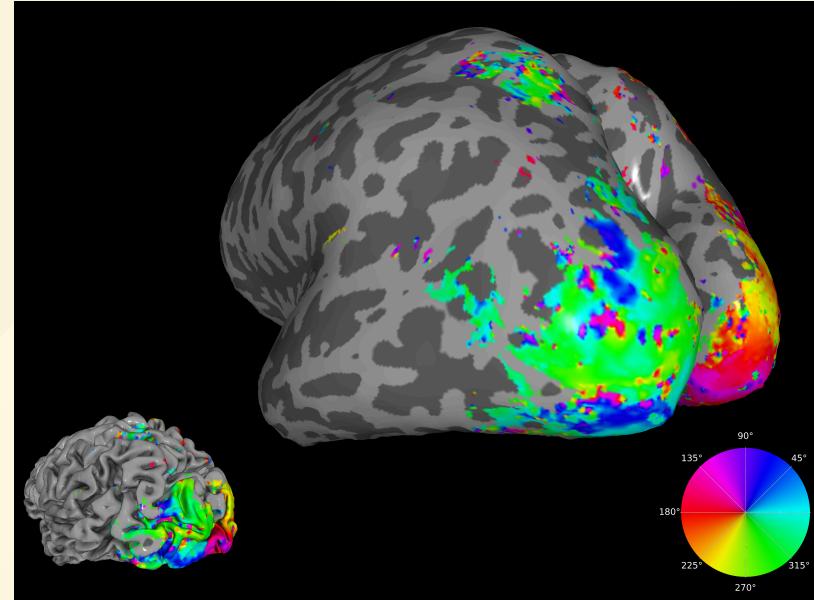


Bayesian perceptual model of risky choice
de Hollander et al. (2025)

Why bother?

Neuroeconomics is full of complex models

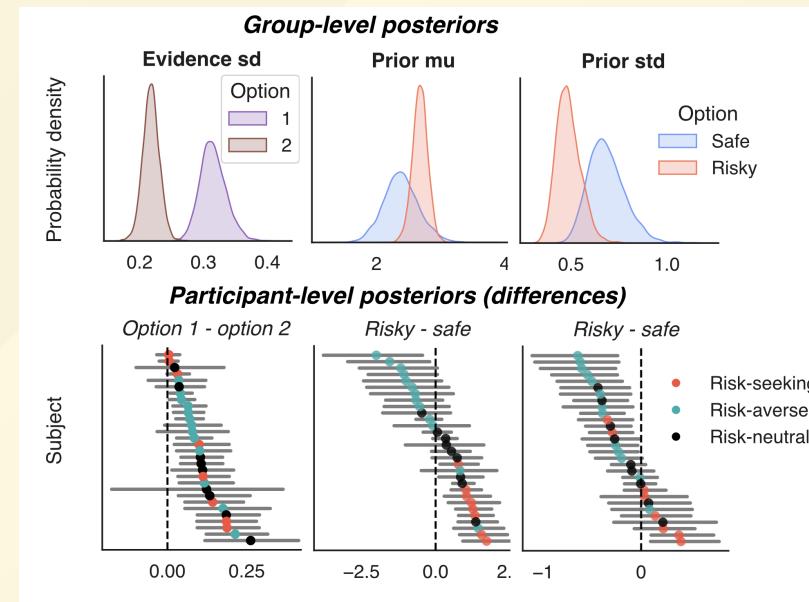
- Cognitive models
- Neural models



Polar angle map of retinotopic organization
de Hollander et al. (in prep)

Why bother?

**Bayesian estimation versus
point estimates**



**Posterior estimates at group-
and participant-level**
de Hollander et al., (2025)

Why bother?

Job market

Matlab jobs in Switzerland

Currently you can find 66 jobs that match your search for developer vacancies with Matlab in Switzerland.

[DEVELOPER SIGNUP](#) [POST A JOB](#)

Browse developer jobs

Matlab X Switzerland X Seniority Language Job Title...

Newest

Software Test Engineer (m/w/d)
Rocken AG Baden, Switzerland Continuous Integration LabView Matlab +1 DETAILS

Software Developer (Off-Car Systems) Graduate
Sautin Group Hinwil, Switzerland Data analysis C++ Python +1 DETAILS

System Engineer 80 - 100 % (f/m/d)
360-Talents Smarter Recruiting Bonaduz, Switzerland Data analysis Matlab Scripting (Bash/Python/Go/Ruby) DETAILS

Embedded Software Engineer (m/w/d)
Rocken AG Switzerland Embedded C Python +1 DETAILS

R&D Modelling Specialist - Mechanistic and Data Driven
Nestlé Lausanne, Switzerland Data analysis Python Matlab DETAILS

Discover your next career opportunity with BWI GmbH
BWI Check it out now

Experience the joy of a headless CMS—free of charge.

PyTorch jobs in Switzerland

Currently you can find 84 jobs that match your search for developer vacancies with PyTorch in Switzerland.

[DEVELOPER SIGNUP](#) [POST A JOB](#)

Browse developer jobs

PyTorch X Switzerland X Seniority Language Job Title...

Newest

Machine Learning Scientist (AI for Code)
Sonarsource SA Geneva, Switzerland Java Amazon Web Services (AWS) Python +3 DETAILS

Machine learning and computer vision developer
Spectrocode Geneva, Switzerland JavaScript Computer Vision C++ +5 DETAILS

Computer Vision Engineer
Daedalean AG Zürich, Switzerland Computer Vision C++ Python +2 DETAILS

Artificial Intelligence Engineer
Snappy Chavannes-de-Bogis, Switzerland API Computer Vision Python +5 DETAILS

Postdoctoral Researcher in Computer Vision for Motor...
Eidmannen Schwei... Zürich, Switzerland Computer Vision Python TensorFlow +1 DETAILS

Discover your next career opportunity with Dennermeyer Group
Dennermeyer Group Check it out now

Register Discover the latest capabilities in Redis REDIS

The Deep learning Revolution

Chat bots

I want this chat to be part of my computational technique sproject...

 That sounds like a great idea, Gilles! Your workshop on computational techniques for neuroeconomic modeling is coming up soon (September 18, 2025), and this chat could serve as a valuable resource for your students or as a reference for your own preparation.

How would you like to integrate this chat into your project?

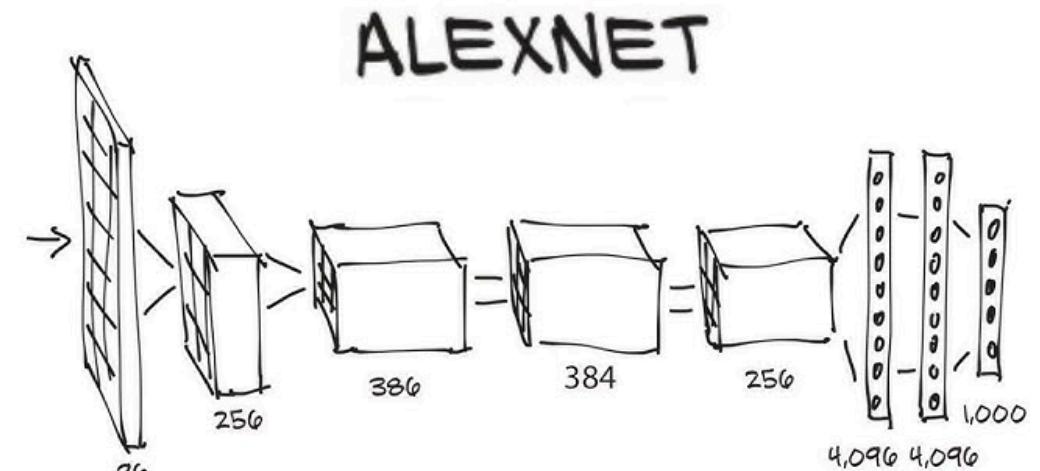
- As a **supplemental resource** for students (e.g., FAQ, troubleshooting, or conceptual explanations)?
- As a **personal reference** for your presentation or hands-on demo (e.g., code snippets, explanations of computational graph techniques, or optimization strategies)?
- As part of your **slide deck** (e.g., screenshots, summaries, or interactive Q&A examples)?
- Or something else? Let me know your vision!

Generative AI

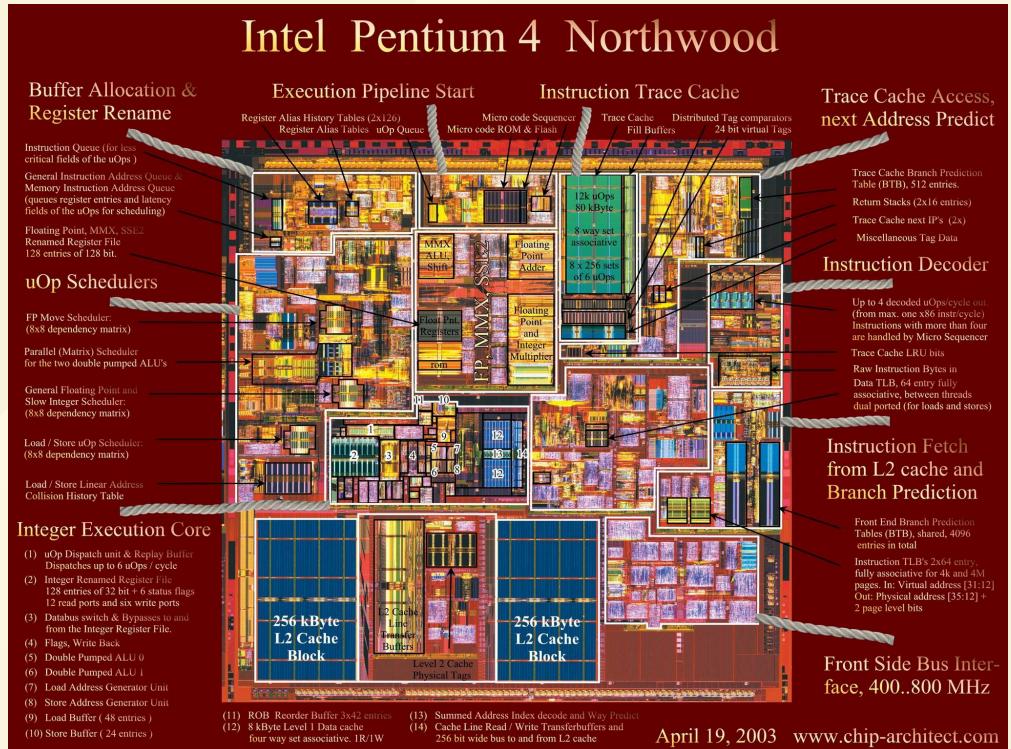


The Deep learning Revolution

- Started with *Alexnet* (2012)
- Key innovations
 - **Very large data set** (1.2 million images)
 - Training on **GPU**
 - Use of **autodiff**



A little bit of Computer Science



A little bit of Computer Science

- Processing units **cannot** "read" programming languages!
- They only execute **machine code** (binary instructions).
- Machine code depends on *platform/instruction set*
 - x86 (Your dad's PC)
 - ARM (New Macs/iPhones)
 - GPU (NVidia, deep learning)

Example: x86 Machine Code (Assembly)

```
; Simple function: add two numbers
section .text
global _start
_start:
    mov eax, 5          ; Load 5 into register EAX
    add eax, 3          ; Add 3 to EAX (result = 8)
    ret                ; Return from function
```

Binary representation:

```
`10110000 00000101` (mov eax, 5)
`00000000 00000011` (add eax, 3)
`11000011` (ret)
```

x86 Architecture

History:

- 1978: Intel 8086
- 1985: 32-bit (80386)
- 2003: 64-bit (AMD64)

Strengths:

- Single-core performance
- Legacy compatibility
- Desktop/server dominance



ARM Architecture

History:

- 1985: Acorn RISC
- 2007: iPhone
- 2020: Apple M1

Strengths:

- Energy efficiency
- Parallel processing
- System-on-a-chip



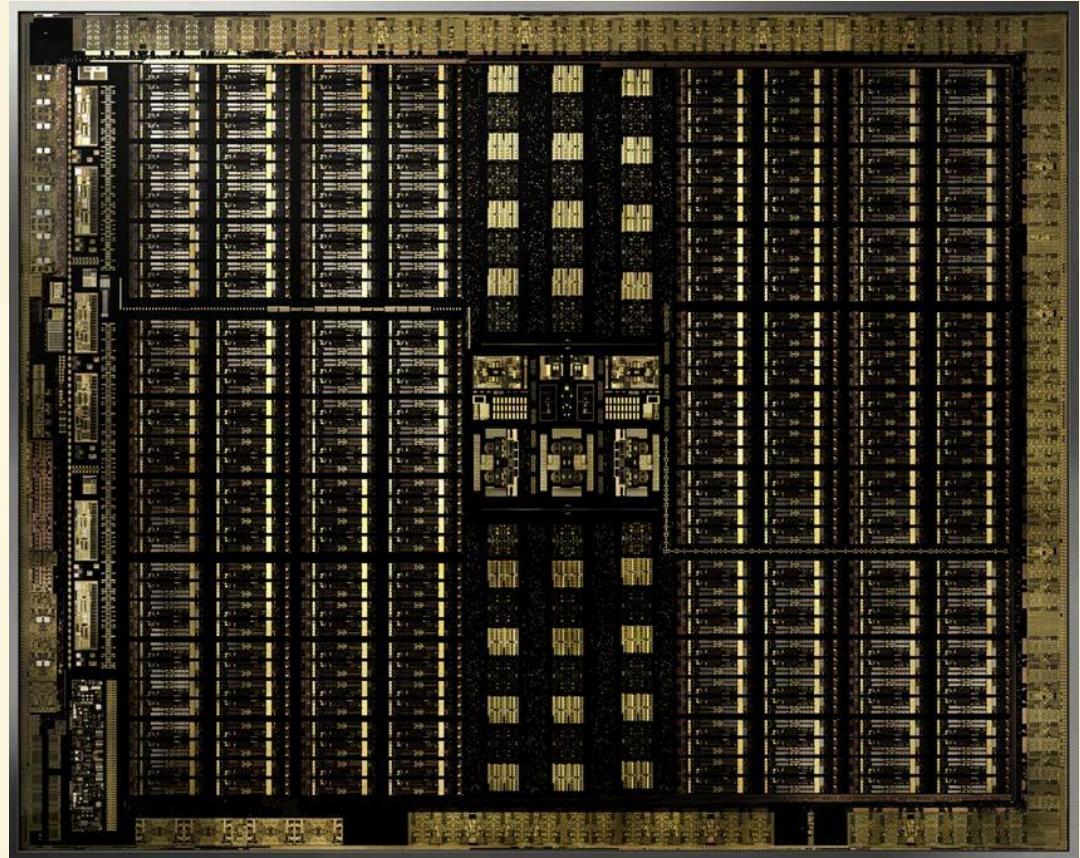
GPU Architecture

History:

- 1999: GeForce 256
- 2006: CUDA
- 2012: Deep learning

Strengths:

- Massive parallelism
- Tensor operations
- AI acceleration



A little bit of Computer Science

- Your code (Python/Matlab/C++) needs to be '*translated to platform-specific* byte-code.

Two main possibilities:

1. *Compiled languages* (C++, Rust)
2. *Interpreted languages* (Matlab/Python/R)

A little bit of Computer Science

Most of us use *hybrid* approaches, where the compiled executables of *Matlab*, *R*, *Python* (*numpy*) are linked to very fast functions compiled in C (e.g, matrix multiplication).

How It Works:

1. You write **high-level code** (Python/MATLAB/R).
2. Heavy computations **delegate to compiled libraries** (C/Fortran).
3. *Example:*

```
# Python (interpreted)
result = np.dot(A, B) # Calls BLAS (compiled C/Fortran)
```

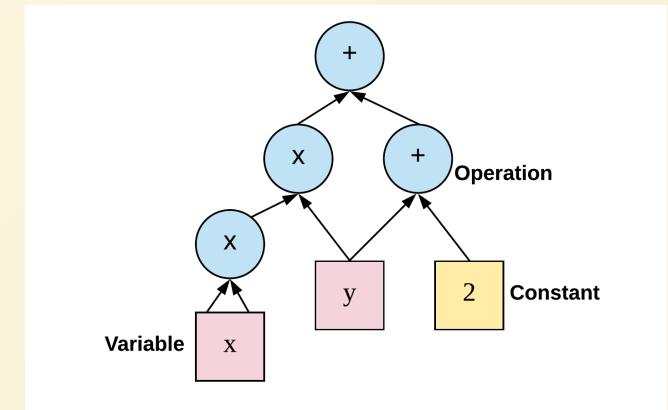
The Future: Computational Graph Libraries

1. Key Idea

Represent your computational problem as a *graph*.

2. Key features

- **Declarative programming:** Define *what* to compute, not *how*.
- **Automatic differentiation (autodiff):** Gradients computed symbolically.
- **Optimizations:** Graph-level fusion, parallelism, and hardware targeting.



Exercise 1: DDM simulation in Tensorflow

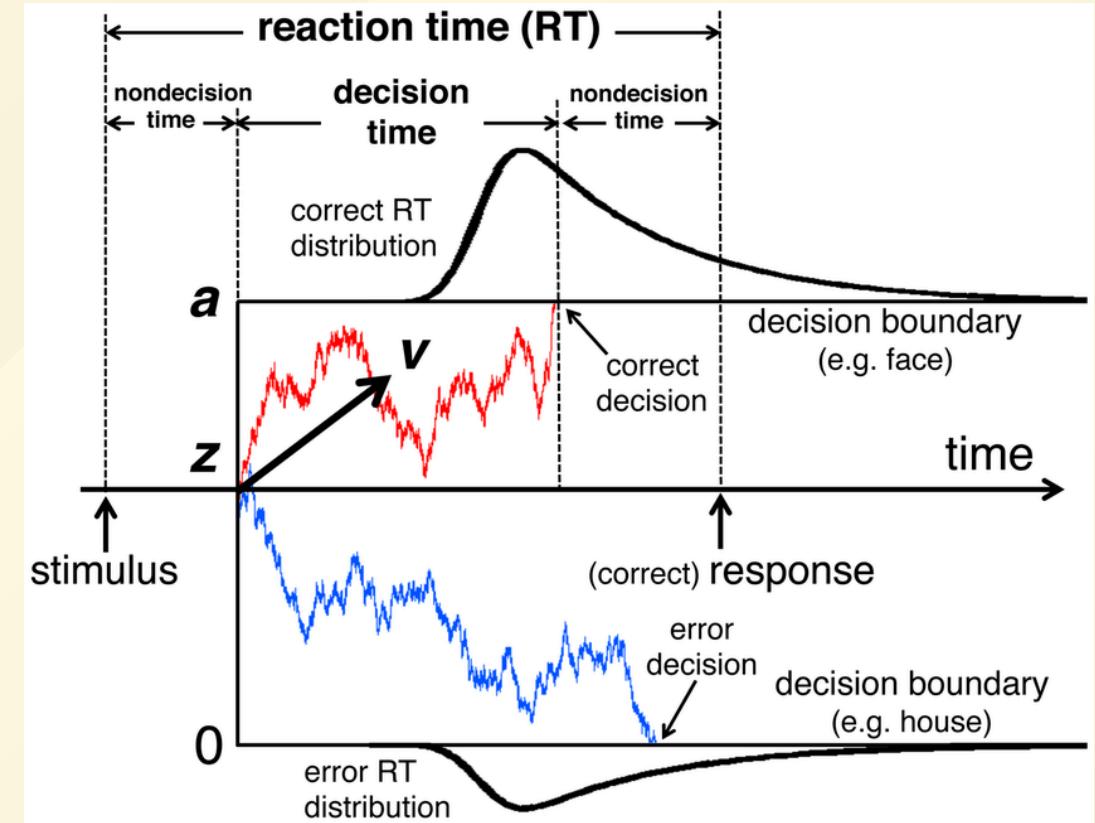
First things first

- Visual Studio Code
- Conda environments:

```
conda env list
  * soglio
  (* soglio_metal)
  (* soglio_gpu)
```

Exercise 1: DDM simulation in Tensorflow

How would you guys implement this?



DDM simulation in Tensorflow

```
@tf.function
def ddm_tensorflow(n_trials=1000, max_t=10.0, dt=0.001, drift=0.1, noise=0.1, bound=1.0):
    n_steps = int(max_t / dt)

    # Generate all noise terms at once
    noise_terms = noise * tf.sqrt(dt) * tf.random.normal([n_trials, n_steps])

    # Create evidence trajectory (n_trials x n_steps)
    drift_terms = tf.ones([n_trials, n_steps]) * drift * dt
    evidence = tf.cumsum(drift_terms + noise_terms, axis=1)

    # Add initial zeros
    evidence = tf.pad(evidence, [[0, 0], [1, 0]], constant_values=0.0)

    # Find crossing times
    crossed_up = evidence >= bound
    crossed_down = evidence <= -bound
    crossed = crossed_up | crossed_down

    # Get first crossing time
    rts = tf.where(
        tf.reduce_any(crossed, axis=1),
        dt * tf.cast(tf.argmax(tf.cast(crossed, tf.int32), axis=1), tf.float32),
        tf.constant(max_t, dtype=tf.float32)
    )

    # Determine responses
    responses = tf.where(tf.reduce_any(crossed_up, axis=1),
                         1,
                         tf.where(tf.reduce_any(crossed_down, axis=1),
                                  0,
                                  tf.random.uniform([n_trials], 0, 2, tf.int32)))

return responses, rts
```

Graph demo

