Introduction to HPC with Python

Workshop at Columbia University

John Stachurski

October 2023

Outline

1. Trends in scientific computing

2. Likely future directions

3. Hands-on computing with Python + NumPy + Numba

4. Hands-on computing with Python + JAX

A (very) short history of scientific computing

General purpose scientific computing environments:

1. Fortran & C / C++

2. MATLAB & (Python + NumPy)

3. Julia & (Python + Numba)

4. AI-driven scientific computing (e.g., Python + Google JAX)

Fortran & C — static types and AOT compilers

```c
#include <stdio.h>
int main() {
    int x = 1 + 1;
    printf("1 + 1 = %d\n", x);
    return 0;
}
```

```fortran
PROGRAM ONE_PLUS_ONE
INTEGER :: X = 1 + 1
PRINT *, '1 + 1 = ', X
END PROGRAM ONE_PLUS_ONE
```
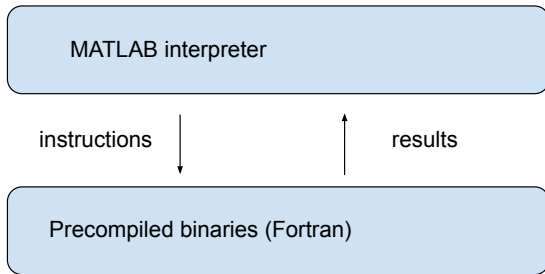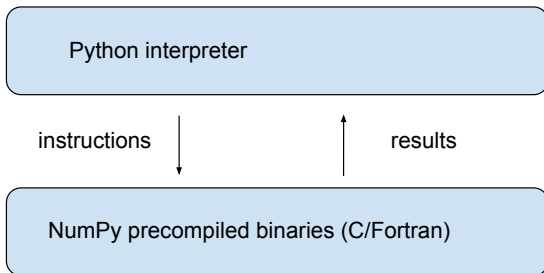
Pros

- fast — on a single thread

Cons

- tedious to write
- lack of portability
- hard to debug
- hard to parallelize
- low interactivity

Phase 2: MATLAB

Phase 2A: Python + NumPy

Python interpreter

instructions ↓   ↑ results

NumPy precompiled binaries (C/Fortran)

Phase 3: Julia — rise of the JIT compilers

```julia
function quad(x0, α, n)
    x = x0
    for i in 1:(n-1)
        x = α * x * (1 - x)
    end
    return x
end

quad(0.2, 4.0, 10_000_000)
```

Phase 3 continued: Python + Numba copy Julia

```python
from numba import jit

@jit
def quad(x0, α, n):
    x = x0
    for i in range(n-1):
        x = α * x * (1 - x)
    return x

quad(0.2, 4.0, 10_000_000)
```

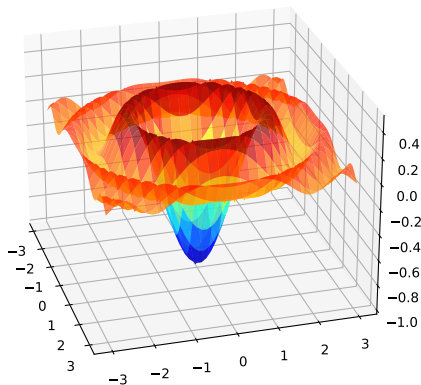# Phase 4: AI-driven scientific computing

Core elements

- JIT-compilers

- automatic differentiation

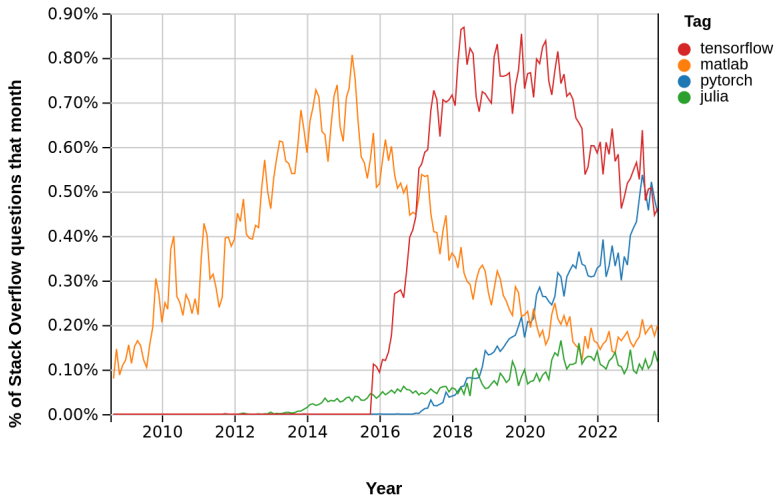- parallelization (CPUs / GPUs / TPUs)

Key players

- TensorFlow

- PyTorch

- Google JAX

AI / machine learning: minimizing differentiable loss functions

Stack Overflow Trends

But PyTorch / TensorFlow are not general scientific computing environments

- specialized to AI / neural nets

Instead we will focus on Google JAX

- Exposes low level tools
- MATLAB / NumPy -like interface
- automatic differentiation
- JIT compiler with XLA
- Automatically detects hardware accelerators
- Same code runs on CPU / GPU / TPU

But PyTorch / TensorFlow are not general scientific computing environments

- specialized to AI / neural nets

Instead we will focus on Google JAX

- Exposes low level tools
- MATLAB / NumPy -like interface
- automatic differentiation
- JIT compiler with XLA
- Automatically detects hardware accelerators
- Same code runs on CPU / GPU / TPU

Sample code

https://github.com/QuantEcon/columbia_2023/notebooks