Python and Julia for economic modeling: recent developments and trends

John Stachurski

October 2023

Topics

- Trends in scientific computing
- Likely future directions
- Python and Julia as MATLAB replacements

A (very) short history of scientific computing

1. Fortran / C / C++ — static type AOT compiled languages
2. MATLAB — interpreter + precompiled Fortran binaries
3. Python + NumPy + SciPy — MATLAB within Python
4. Julia — rise of the JIT compilers
5. Python + Numba — Julia style computing in Python
6. Python + JAX — Parallelization, JIT and autodiff

Fortran & C — static types and AOT compilers

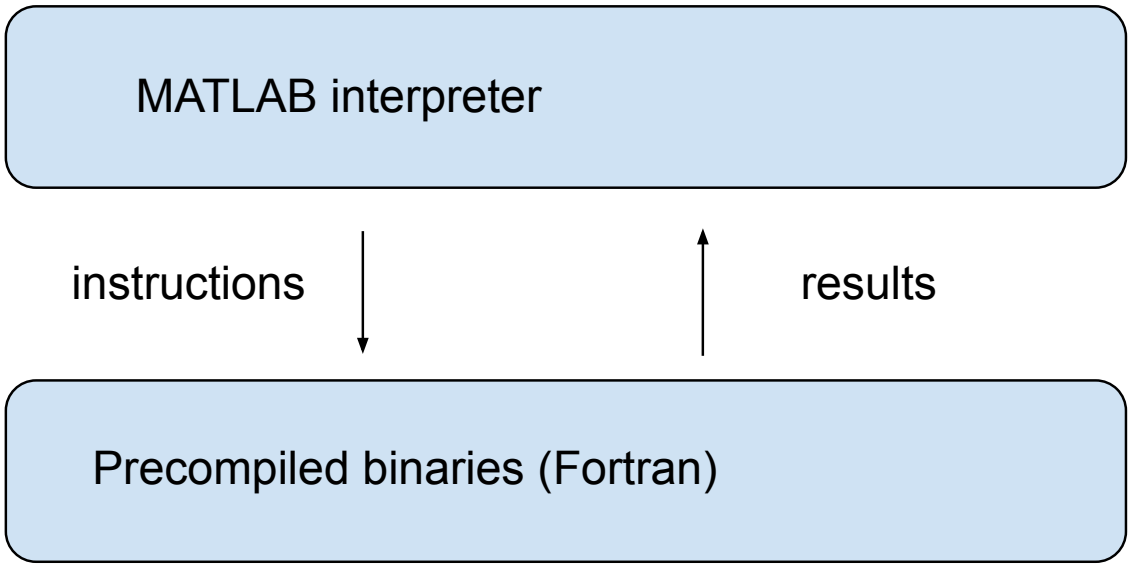
Example. $1 + 1$ in C

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

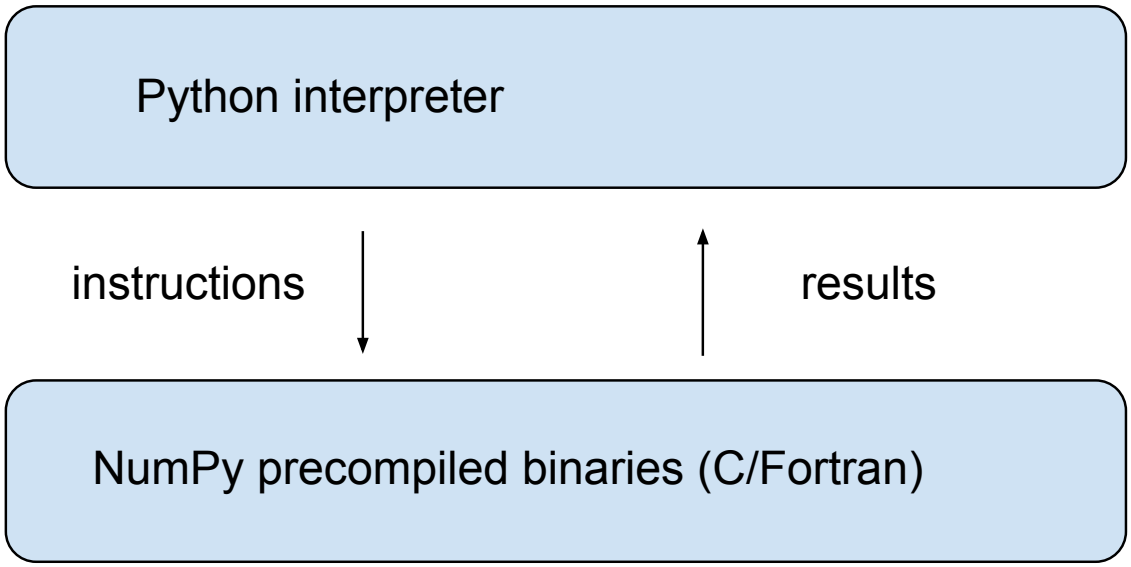
Example. $1 + 1$ in Fortran

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

Phase 2: MATLAB



Phase 2A: Python + NumPy



Phase 3: Julia — rise of the JIT compilers

```
const alpha = 4.0

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

quad(0.2, 10_000_000)
```

Phase 3 continued: Python + Numba copy Julia

```
const alpha = 4.0

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

quad(0.2, 10_000_000)
```

Phase 4: AI-driven scientific computing

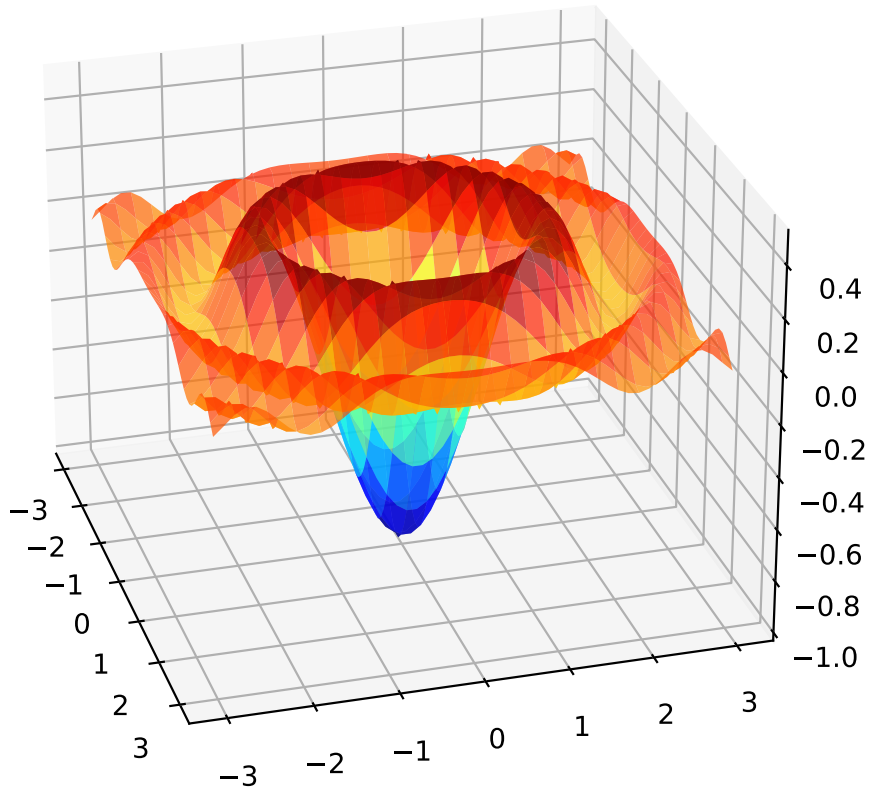
Core elements

- JIT-compilers
- autodiff
- parallelization (CPUs / GPUs / TPUs)

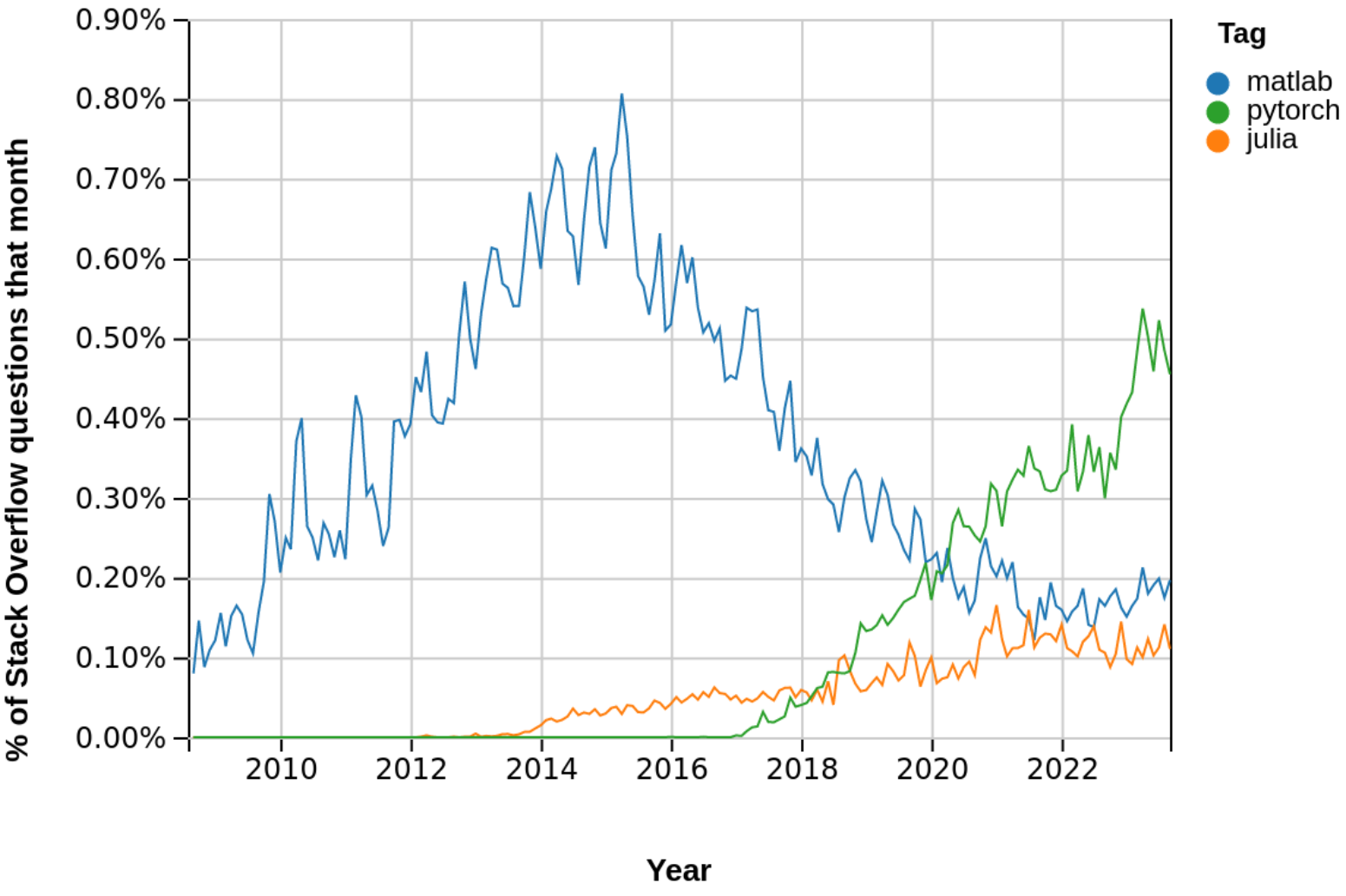
Key players

- PyTorch
- tensorflow
- Google JAX

AI / machine learning: minimizing differentiable loss functions



Popularity:


0.90%
0.80%
0.70%
0.60%
0.50%
0.40%
0.30%
0.20%
0.10%
0.00%
% of Stack Overflow questions that month
2010
2012
2014
2016
2018
2020
2022
Year
Tag
matlab
pytorch
julia