PHAS0102: Techniques of High-Performance Computing

Assignment 1

- On Moodle and mscroggs.co.uk/phas0102
- Reminder: 20% of the assessment for the course
- Deadline: Thursday 20 October 5pm

Column vectors vs row vectors

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}^{\mathrm{T}} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

```
import numpy as np

a = np.array([1, 2, 3])
b = a.transpose()

b = np.array([[1], [2], [3]])
```

Numpy will interpret this as either a row vector or a column vector depending on the situation.

timeit

```
t = timeit(matvec(A, v), repeat=10)
```

"Here's a vector, how long does it take?"

```
def f():
    matvec(A, v)
t = timeit(f, repeat=10)
```

"Here's a function, how long does it take?"

Do the same thing

```
t = timeit(lambda: matvec(A, v), repeat=10)
```

Numba

- Just-in-time compilation
  - Converts Python functions into fast compiled code when the function is first called

[live Numba demo]

What does Numba do?

- Detects information about your CPU then makes code that will run fast on your computer.
  - SIMD
  - Parallel for loops (with automatically detected number of processes)
- Numba can be configured if you don’t want to use the auto settings.

What can Numba *not* do?

- Many things Python can do:
  - Pandas
  - Lists with different types inside

- If you want to use Numba on something it can't do, you can use `@jit(nopython=False)` to make it only partially compile a function.

Numexpr

- Numexpr can be used to do fast operations on Numpy arrays

[live Numexpr demo]

O

- A (mathematical) function is $O(n^k)$ if (for very large n) the function is less than an^k.
- An algorithm is $O(n^k)$ if the number of operations is needs to be completed is $O(n^k)$.

O

```
result = 0
for i in range(n):
    for j in range(n):
        for k in range(n):
            result += A[i, j, k]
```

"This function is $O(n^3)$ because there are 3 for loops." ✓

O

```
result = 0
for i in range(n):
    result += A[i]
for j in range(n):
    result += B[j]
```

“This function is $O(n^2)$ because there are 2 for loops.” ✗

O

```
result = 0
for i in range(n):
    for j in range(n):
        for k in range(2):
            result += A[i, j, k]
```

"This function is $O(n^3)$ because there are 3 for loops." ✗

O

```
result = 0
for i in range(n):
    for j in range(n):
        result += f(i, j)
```

“This function is $O(n^2)$ because there are 2 for loops.” ?

Example: matrix-matrix multiplication

$$\begin{pmatrix} * & \cdots & * \\ \vdots & \ddots & \vdots \\ * & \cdots & * \end{pmatrix} \begin{pmatrix} * & \cdots & * \\ \vdots & \ddots & \vdots \\ * & \cdots & * \end{pmatrix}$$

Memory: n^2 numbers in result $\longrightarrow O(n^2)$

Number of operations:

Each entry of the result needs n multiplications and n-1 additions

There are n^2 entries

So overall, $n^2(2n-1)$ operations $\longrightarrow O(n^3)$

Example: matrix-matrix multiplication

- There are algorithms for matrix-matrix multiplication that are faster than $O(n^3)$.
  - In 1969, an $O(n^{2.8074})$ algorithm was found
  - In 2020, an $O(n^{2.3728596})$ algorithms was found
  - It is unknown what the optimal possible complexity is, but it is know that it's between $O(n^2)$ and $O(n^{2.3728596})$

Memory-bound & compute-bound

- An algorithm is memory-bound if it is limited by how much memory it needs.
- An algorithm is compute-bound if it is limited by how many operations is needs to do.

- The status of an algorithm can depend on the hardware of a computer.