CSCI / MATH 2072U - Assignment 3

Mariyam Alim

6th March 2022

1(a) Pseudo code

Input: integer N

1: $x \leftarrow []$
2: $sum \leftarrow 0$
3: $step \leftarrow 1/N$
4: $V \leftarrow$ empty $N+1$ x $N+1$ numpy array
5: **for** $i = 0, ..., N$ **do**
6:     **for** $j = 0, ..., N$ **do**
7:         x $\leftarrow sum$ $\triangleright$ Append sum to list x
8:         $sum \leftarrow sum + step$
9:         $V[i,j] \leftarrow x[i]^j$
10:     **end for**
11: **end for**
12: **return** V

Output: V, an $N+1$ x $N+1$ array of floats

Counting FLOPs

Line 3: 1 FLOP
Line 4: 2 FLOPs
Line 8: 1 x $(N+1)$ x $(N+1) = (N+1)^2$
Line 9: Maximum possible FLOP count = N

FLOP count: $(N+1)^2 + N + 3$

My algorithm has quadratic time complexity, $\mathcal{O}(N^2)$, i.e. its performance is directly proportional to N^2.

1(f) The numerically obtained solution is accurate up until $N = 18$. This is because the maximal relative error is larger than 1 for $N > 18$. The condition number of V for $N = 18$ is 1.468×10^{15} and the relative residual is 2.250×10^{-16}.

2(a) Pseudo code

Input: arrays of n floats **x, a, b** and **c**

1: $n \leftarrow$ size of vector x
2: $y \leftarrow n$ x 1 empty numpy array
3: $y[0] \leftarrow b[0] \times x[0] + c[0] \times x[1]$ ▷ (Setting first entry of y)
4: **for** $i = 1, ..., N - 2$ **do**
5: $\quad y[i] \leftarrow a[i] \times x[i-1] + b[i] \times x[i] + c[i] \times x[i+1]$
6: **end for**
7: $y[n-1] \leftarrow a[n-1] \times x[n-2] + b[n-1] \times x[n-1]$ ▷ (Setting last entry of y)
8: **return** y

Output: array of n floats y such that $\mathbf{y = Ax}$

2(b)

Counting FLOPs

Line 3: 3 FLOPs
Line 4: 1 FLOP
Line 5: 7 x $(n - 2) = 7n - 14$ FLOPs
Line 7: 8 FLOPs

FLOP count: $3 + 1 + 8 + 7n - 14 = 7n - 2$

My algorithm has linear time complexity, $\mathcal{O}(n)$. The algorithm's runtime grows directly in proportional to n.

2(e) The plot shows that my algorithm indeed does have linear time complexity, i.e. has a slope of 1.
The plot would show a quadratic time complexity if other matrix-vector product functions such as np.matmul and np.dot were used. This is because the matrix product of A and x would be determined using a nested for loop. The plot would still be a straight line but with a slope of 2.