Assignment 3

Exercise :1

a) $\sin x = O(1)$ as $x \to \infty$

Since sin oscillate between 1 and -1 $\forall x$
$|\sin x| \leq 1 \cdot C \quad \forall x \geq t_0$
Comparing with Big O notation $C = 1$ and $t_0 = 0$.
True

b) $\sin x = O(1)$ as $x \to 0$

$|\sin x| \leq 1 \quad \forall x \in [0, \pi/2]$
$|\sin x| \leq C \cdot 1 \quad \forall x \leq t_0$
$C = 1$ and $t_0 = \pi/2$

True

c) $\ln x = O(x^{1/100})$ as $x \to \infty$
$|\ln x| \leq c \cdot x^{1/100}$.
$\frac{1}{C} |\ln x| \leq x^{1/100}$.

$|\ln x^{1/c}| \leq x^{1/100}$

$(|\ln k| \leq k \quad \forall k \geq 1)$
assume $C = 100$.
$(\ln x^{1/100}) \leq x^{1/100}$.
$|\ln x| \leq c \cdot x^{1/100} \quad \forall x \geq t_0$
$C = 100$ and $t_0 = 1$.

True.

d) $n! = O\left(\left(\frac{n}{e}\right)^n\right)$ as $n \to \infty$ (stirling's Approximation)

$n! \le c\left(\left(\frac{n}{e}\right)^n\right)$

stirling equation

$$e^{\frac{1}{12n+1}} \le \frac{n!}{\sqrt{2\pi n}\cdot\left(\frac{n}{e}\right)^n} \le e^{\frac{1}{12n}}$$

$$e^{\frac{1}{12n+1}}\cdot\sqrt{2\pi n}\cdot\left(\frac{n}{e}\right)^n \le n! \le e^{\frac{1}{12n}}\cdot\sqrt{2\pi n}\cdot(n/e)^n$$

$$n! \le \underbrace{e^{\frac{1}{12n}}\sqrt{2\pi n}}_{c}\left(\frac{n}{e}\right)^n$$

Here $c = e^{(1/12n)}\sqrt{2\pi n}$ dependent on n. So a constant value of c is not possible. as $(n\to\infty) \Rightarrow (c\to\infty)$

False.

e) $fl(\pi) - \pi = O(\epsilon m)$ as $\epsilon m \to 0$

$fl(\pi) - \pi \le c\, O(\epsilon m)$

Theorem (13.5) $fl(\pi) = \pi(1+\epsilon)$

$|\epsilon| \le \epsilon$machine.

$fl(\pi) - \pi = \pi(1+\epsilon) - \pi$

$= \pi\epsilon$

$|\pi\epsilon| \le \pi\, \epsilon$machine

$\pi\epsilon \le c(\epsilon m)$

$\boxed{c = \pi}$ $n_0 = 100$.

True.

Page No.
Date

f) $fl(n\pi) - n\pi = O(\epsilon_m)$ as $\epsilon_m \to 0$.

$fl(n\pi) - n\pi \leq C\, O(\epsilon_m)$.

$fl(n\pi) = (1+\epsilon)\, n\pi$.

$fl(n\pi) - n\pi = (1+\epsilon)\, n\pi - n\pi$

$= n\pi\epsilon$

$n\pi\epsilon \leq C\, O(\epsilon_m)$

$C = n\pi$.

Here $n\pi$ monotonically increases with n. (and c is dependent on the ~~vale~~ variable n) not a constant value

False.

Exercise : 2

a) $f(x) = 2x$ for $x \in \mathbb{C}$ $\tilde{f}(x) = x \oplus x$

$fl(x) = x(1+\epsilon_1)$

$\tilde{f}(x) = x \oplus x$.

$[x(1+\epsilon_1) + x(1+\epsilon_1)](1+\epsilon_2)$

$= 2x(1+\epsilon_1)(1+\epsilon_2) \Rightarrow 2x(1+\epsilon_3)$.

$= 2\tilde{x}$ $[\tilde{x} = x(1+\epsilon_3)]$

accuracy condition $= \dfrac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = O\,\epsilon_{machine}$.

$= \dfrac{\|2x(1+\epsilon_3) - 2x\|}{\|2x\|}$

$= \dfrac{\|2x\epsilon_3\|}{\|2x\|} = \|\epsilon_3\| = O\,\epsilon_{machine}$.

Algorithm is accurate.

Backward stability $\Rightarrow \tilde{f}(x) = f(\tilde{x})$

$\tilde{x} = \frac{\|\tilde{x} - x\|}{\|x\|} = O\,\epsilon_{machine}$.

$\tilde{f}(x) = 2\tilde{x}$

$f(\tilde{x}) = \tilde{x} + \tilde{x} = 2\tilde{x}$

$\tilde{f}(x) = f(\tilde{x})$.

Backward stable.

b) $f(x) = x^2$ for $x \in ¢$ $\tilde{f}(x) = x \otimes x$

$fl(x) = x(1+\epsilon_1)$

$\tilde{f}(x) = x \otimes x$.

$= fl(x) \otimes fl(x)$

$= x(1+\epsilon_1) \otimes x(1+\epsilon_1)$

$= x^2(1+\epsilon_1)^2 (1+\epsilon_2)$

$= x^2(1+2\epsilon_1+\epsilon_1^2)(1+\epsilon_2)$

$= x^2(1+2\epsilon_1+\epsilon_1^2+\epsilon_2+2\epsilon_1\epsilon_2+\epsilon_1^2\epsilon_2)$

$= x^2(1+\epsilon_3)$.

$[\epsilon_3 = 2\epsilon_1+\epsilon_1^2+\epsilon_2+2\epsilon_1\epsilon_2+\epsilon_1^2\epsilon_2]$

$= (\tilde{x})^2$

accuracy: $\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = O\,\epsilon_{machine}$.

$= \left\| \frac{\{x^2(1+\epsilon_1)^2 - x^2\}}{\|x^2\|} \right\|$

$= \frac{\|x^2(1+2\epsilon_1+\epsilon_1^2) - x^2\|}{\|x^2\|}$

$= \frac{\|x^2(2\epsilon_1+\epsilon_1^2)\|}{\|x\|} = O\,\epsilon_{machine}$.

acurate.

Page No.
Date

Backward stability $\Rightarrow \frac{\|\check{x}-x\|}{\|x\|} = O\ \epsilon_{machine}$.

$$\left.\begin{aligned} f(\check{x}) &= \check{f}(x) \\ \check{f}(x) &= (\check{x})^2 \\ f(\check{x}) &= (\check{x} \times \check{x}) = (\check{x}) \end{aligned}\right\} \quad f(\check{x}) = \check{f}(x)$$

Backward stable.

c) $f(x) = 1$ for $x \in \mathbb{C} \setminus \{0\}$ $\quad \hat{f}(x) = x \oslash x$.

$$fl(x) = x(1+\epsilon_1)$$
$$\check{f}(x) = x \oslash x$$
$$= fl(x) / fl(x)$$
$$= \frac{x(1+\epsilon_1)}{x(1+\epsilon_1)}(1+\epsilon_2)$$
$$= (1+\epsilon_2)$$

$$\text{accuracy} = \frac{\|\hat{f}(x) - f(x)\|}{\|f(x)\|} = O\ \epsilon_{machine}.$$
$$= \frac{\|(1+\epsilon_2) - 1\|}{\|1\|} = O\ \epsilon_{machine}$$
$$= \|\epsilon_2\| = O\ \epsilon_{machine}.$$

accurate.

Backward stability $\Rightarrow \frac{\|\check{x}-x\|}{\|x\|} = O\ \epsilon_{machine}$

$$f(\check{x}) = \check{f}(x)$$
$$\left.\begin{aligned} f(\check{x}) &= \check{x}/\check{x} = 1 \\ \check{f}(x) &= (1+\epsilon_2) \end{aligned}\right\} \quad f(\check{x}) \neq \hat{f}(x)$$

Not Backward stable

stability $\frac{\|\breve{f}(x) - f(\breve{x})\|}{\|f(\breve{x})\|} = O(\epsilon_{machine})$.

$$= \frac{\|(1+\epsilon_2) - 1\|}{\|1\|} = \|\epsilon_2\| = O(\epsilon_{machine})$$

stable

d) $f(x) = 0$ for $x \in \mathbb{C}$. $\breve{f}(x) = x \ominus x$

$fl(x) = x(1+\epsilon_1)$

$\breve{f}(x) = x \ominus x$

$= fl(x) \ominus fl(x)$

$= [x(1+\epsilon_1) - x(1+\epsilon_1)](1+\epsilon_2)$

$= 0$.

accuracy $= \frac{\|\breve{f}(x) - f(x)\|}{\|f(x)\|} = O(\epsilon_{machine})$.

$$= \frac{\|0 - (x-x)\|}{\|(x-x)\|} = \frac{0}{0} \neq O(\epsilon_{machine}).$$

not accurate.

Backward stability $\breve{f}(x) = f(\breve{x})$.

$$\frac{\|\breve{x} - x\|}{\|x\|} = O(\epsilon_{machine}).$$

$\left.\begin{array}{l} \breve{f}(x) = 0 \\ f(\breve{x}) = \breve{x} - \breve{x} = 0 \end{array}\right\}$ $f(\breve{x}) = \breve{f}(x)$

Backward stable

Exercise : 3

a) $A \in \mathbb{C}^{(m+1)\times m}$

$QR = A$.

In the 1^{st} column, we need to reduce $(m-1)$ elements to zero using row operation on $(n-1)$ non trivial row elements. But from the next column, we need to reduce $(m-2)$ element to zero.

Eg:

$$\begin{array}{cc} & n \\ A \,\{\, m+1 & \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \end{array} \rightarrow \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \rightarrow \cdots \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}$$

$$\downarrow \;\vdots$$

$$\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} = R.$$

Computing Total cost:

$$\sum_{k=1}^{n} (m-k)(n-k) = mn^2 - (m+n)\frac{n(n+1)}{2} + \frac{n(n+1)(2n+1)}{6}.$$

$$= \frac{n(m-1)}{2}\left(m - \frac{n+1}{3}\right)$$

$A \in \mathbb{C}^{(m+1)\times m}$

$$\text{Flops} = \frac{m(m-1)}{2}\left[(m+1) - \frac{(m+1)}{3}\right]$$

$$= \frac{m(m-1)}{2}\left[\frac{3m+3-m-1}{3}\right] = \frac{m(m-1)}{\cancel{2}}\left[\frac{\cancel{2}m+\cancel{2}}{3}\right]$$

$$= \frac{m(m-1)(m+1)}{3} = \frac{m(m^2-1)}{3}.$$

b) $A \in \mathbb{C}^{(m+1)\times m}$ = upper hessenberg matrix

Eg:

$$A = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & \cancel{x} \\ 0 & 0 & 0 & x \end{bmatrix}$$

needs to be zero

Total no. of operation $n_{op} = \frac{1}{2}(m)(m+1)\times 6$

$\sim O(m^2)$

$n_{op}(m) = C g \cdot m^2$

$g(m) = m^2$

```python
import numpy as np
from numpy.testing import assert_allclose


def givens_qr(H):
    H = H + 0.0
    dtype = np.complex128 if np.iscomplexobj(H) else np.float64
    m = np.shape(H)[1]
    i = 0
    R = np.copy(H)
    Q = np.zeros((m, 2)).astype(H.dtype)
    G = np.zeros((m, 2)).astype(H.dtype)

    def givens_rotation(a, b):
        r = np.sqrt(abs(a) ** 2 + abs(b) ** 2)
        c =  np.conj(a)/ r
        s = np.conj(b)/ r
        return c, s



    for k in range(m):
        a = R[k, k]
        b = R[k + 1, k]

        c, s = givens_rotation(a, b)

        if b != 0:
            G = np.identity(2).astype(c.dtype)
            G[0] = c, s
            G[1, 0] = -s.conjugate()
            G[1, 1] = c.conjugate()
```

```
            R[k:k + 2, k:] = G @ R[k:k + 2, k:]

            Q[i] = c, s
        else:
            Q[i] = 1, 0

        i += 1

    return Q, R

# Example usage with complex numbers:
H = np.array([[1,2,3],[4,5,6],[0,7,8],[0,0,9]])


print("Original Upper Hessenberg Matrix:")
print(H)
# Compute QR factorization using Givens rotations
R = givens_qr(H)
print("\nUpper Triangular Matrix R:")
print(R)

def form_q(G):
    m = G.shape[0]
    Q = np.identity(m + 1).astype(G.dtype)

    for j in range(m):

        K = np.identity(2).astype(G.dtype)
        K[:, 0] = G[j].conjugate()
        K[0, 1] = -G[j, 1]
        K[1, 1] = G[j, 0]


        Q[:, j:j + 2] = Q[:, j:j + 2] @ K

    return Q
```