APPM 4600 HW2

Ghulam Yahya Rajabi

Due Feb 7 2024

1

a)

$$(1+x)^n = 1 + nx + o(x) \text{ as } x \to 0$$

$$(1+x)^n = \frac{1}{0!}x^0 + \frac{n}{1!}x^1 + \frac{n(n-1)}{2!}x^2 + \ldots + nx^{n-1} + x^n$$

let $g(x) = \frac{n(n-1)}{2!}x^2 + \ldots + nx^{n-1} + x^n$

showing that $f(x)$ grows slower than $g(x)$ near $x = 0$

let $h(x) = nx$

$$\lim_{x\to 0}\left|\frac{g(x)}{h(x)}\right|$$

$$\lim_{x\to 0}\left|\frac{\frac{n(n-1)}{2!}x^2 + \frac{n(n-1)(n-2)}{3!}x^3 + \ldots + nx^{n-1} + x^n}{nx}\right|$$

$$\lim_{x\to 0}\left|\frac{\frac{n(n-1)}{2!}x + \frac{n(n-1)(n-2)}{3!}x^2 + \ldots + nx^{n-2} + x^{n-1}}{n}\right| = 0$$

Therefore, $(1+x)^n = 1 + nx + 0(x)$ as $x \to 0$

b)

$$xsin(\sqrt{x}) = O(x^{\frac{3}{2}}) \text{ as } x \to 0$$

$$\sin(x) = \sum_{n=1}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

$$\sin(\sqrt{x}) = \sum_{n=0}^{\infty}(-1)^n \frac{x^{n+\frac{1}{2}}}{(2n+1)!} \text{ for x} \geq 0$$

$$xsin(\sqrt{x}) = x^{\frac{3}{2}} \sum_{n=0}^{\infty}(-1)^n \frac{x^n}{(2n+1)!} \text{ for x} \geq 0$$

let f(x) = $xsin(\sqrt{x}) = x^{\frac{3}{2}} \sum_{n=0}^{\infty}(-1)^n \frac{x^n}{(2n+1)!}$ let g(x) = $x^{\frac{3}{2}}$ f(x) = O(g(x)) if:

$$\lim_{x\to 0^+} \left| \frac{x^{\frac{3}{2}} \sum_{n=0}^{\infty}(-1)^n \frac{x^n}{(2n+1)!}}{x^{\frac{3}{2}}} \right| \text{ k , k} > 0\text{, x}\geq 0$$

$$\lim_{x\to 0^+} \left| \sum_{n=0}^{\infty} \frac{(-1)^n(\sqrt{x})^{2n+1}}{(2n+1)!} . \frac{1}{\sqrt{x}} \right|$$

let u = $\sqrt{x}$

$$\lim_{x\to 0^+} \frac{\sin(u)}{u} = 1$$

Therefore, $x\sin(\sqrt{x}) = O(x^{\frac{3}{2}})$

c)

$$e^{-t} = o(\frac{1}{t^2}) \text{ as } t \to \infty$$

let $f(t) = e^{-t}$, $g(t) = \frac{1}{t^2}$

$$\lim_{t\to\infty} \left|\frac{f(t)}{g(t)}\right| = 0$$

$$\lim_{t\to\infty} \left|\frac{t^2}{e^t}\right| = \frac{\infty}{\infty}$$

L'hop:

$$\lim_{t\to\infty} \left|\frac{2}{e^t}\right| = 0$$

therefore, $e^{-t} = o(\frac{1}{t^2})$ as $t \to \infty$

d)

$$\int_0^\epsilon e^{-x^2}\,dx = O(\epsilon) \quad \text{as } \epsilon \to 0$$

let $f(\epsilon) = \int_0^\epsilon e^{-x^2}\,dx$
let $g(\epsilon) = \epsilon$

$$\lim_{\epsilon\to 0}\left|\frac{\int_0^\epsilon e^{-x^2}\,dx}{\epsilon}\right| = 0$$

$$e^{-x^2} = \frac{1}{0!} - \frac{x^2}{1!} + \frac{x^4}{2!}...$$

$$\int_0^\epsilon e^{-x^2}\,dx = \sum_{n=0}^{\infty}(-1)^n\frac{\epsilon^{2n+1}}{(2n+1)n!}$$

$$\lim_{\epsilon\to 0}\left|\frac{\epsilon - \frac{\epsilon^3}{3} + \frac{\epsilon^5}{10} - ...}{\epsilon}\right| = 1$$

Therefore, $\int_0^\epsilon e^{-x^2}\,dx = O(\epsilon) \quad \text{as } \epsilon \to 0$

2

a)

$$A\vec{x} = \vec{b}$$
$$\vec{x} = A^{-1}\vec{b}$$
$$A\hat{x} = \vec{b} + \triangle\vec{b}$$
$$\hat{x} = A^{-1}\vec{b} + A^{-1}\triangle\vec{b}$$
$$\triangle\vec{x} = A^{-1}\triangle\vec{b}$$

b)

$$K(A) = ||A||_2 \left|\left|A^{-1}\right|\right|_2 = \frac{\sigma_1}{\sigma_n}$$

K(A) = 19999973849.225224 (see appendix for code)

c)

relative error < 1414211.780405796 (see code in appendix)

The relative error increases with the magnitude of the perturbation and the condition number. A higher condition number means a higher sensitivity to perturbations, leading to larger relative errors. If the perturbations are the same, the relative error may increase if the condition number is high.

3

a)

$$K(f(x)) = \frac{|f'(c)|\,|x|}{|f(x)|}$$

$$f'(x) = \frac{\delta f}{\delta x}$$

$$K(f(x)) = \frac{|e^c|\,|x|}{|e^x - 1|}$$

for values of x far from 0, the function becomes ill-conditioned, meaning it is highly sensitive to small changes in x

b)

let's check the limit as it goes to ∞

$$\lim_{x\to\infty} \frac{e^c\,|x|}{|e^x - 1|}$$

$$e^c \lim_{x\to\infty} \frac{x}{e^x - 1}$$

L'hop

$$e^c \lim_{x\to\infty} \frac{1}{e^x} = e^c$$

let's check the limit as it goes to 0

$$\lim_{x\to 0} \frac{e^c\,|x|}{|e^x - 1|}$$

$$e^c \lim_{x\to 0^+} \frac{1}{e^x} = e^c$$

$$e^c \lim_{x\to 0^-} \frac{-1}{-e^x} = e^c$$

we can conclude that the algorithm is stable around x=0. For c near x=0, e^c is close to 1 and this low condition number means high stability for us. It is not stable for larger numbers of x as the limit approaching ∞ is again e^c.

c)

answer = 1.00000008274037e-0.9 (see code in appendix)
expected = 1.10^-9

16 correct digits as expected since f(x) is stable close to 0.

d)

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$

$$f(x) = e^x - 1 = x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$

let $g_n(x) = x + \frac{x^2}{2!} + ... + \frac{x^n}{n!}$

$$\frac{|f(x) - g_n(x)|}{f(x)} \leq 1.10^{-16}$$

$x_1 = 9.99...50.10^{-10}$

$$\frac{|(e^{x_1} - 1) - g_n(x_1)|}{e^{x_1} - 1} \leq 1.10^{-16}$$

only 2 terms needed (see code in appendix)
$g_2(x) = x + \frac{x^2}{2!}$

e) verified in code in appendix

4

a)
sum = -20.686852364346837

b)

4.1

section

4.2

section

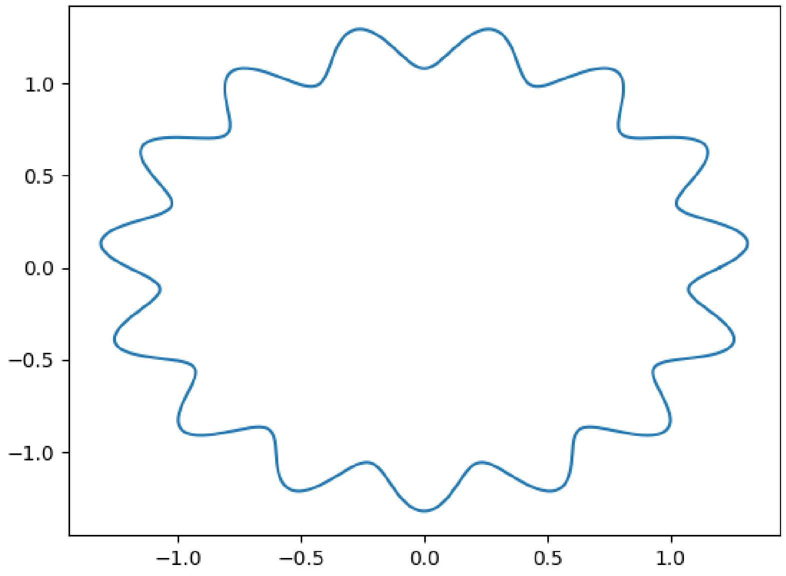

Figure 1: part 1

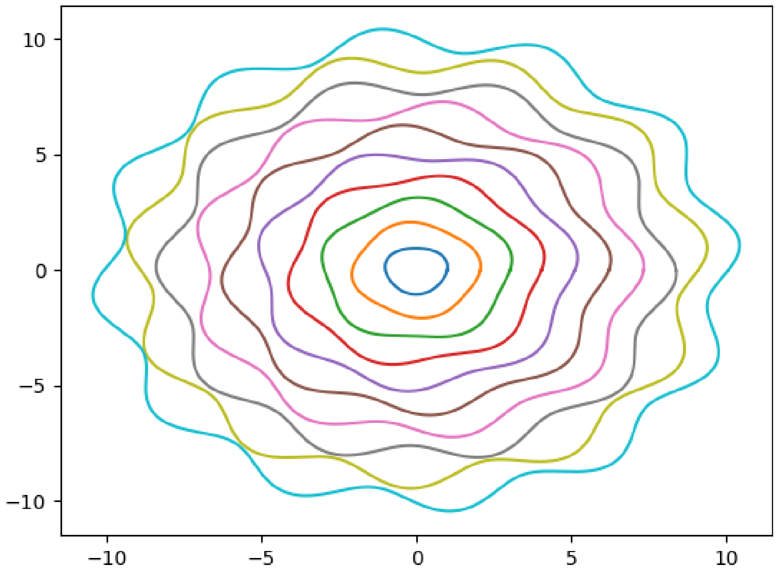

Figure 2: part 2

5 Appendix (code also available in github repo)

```
import numpy as np
import matplotlib.pyplot as plt
import random
import math

# Calculate condition number
matrix_A = 0.5 * np.array([[1, 1], [1 + 10 ** -10, 1 - 10 ** -10]])
poi_matrix = np.array([[1 - 10 ** 10, 10 ** 10], [1 + 10 ** 10, -1 * 10 ** 10]])
max_singular_val = np.linalg.svd(matrix_A)[1][0]
min_singular_val = np.linalg.svd(matrix_A)[1][1]
condition_num = max_singular_val / min_singular_val
print("Condition number: " + str(condition_num))

# Calculate norm and relative error
delta_B = np.array([5 * 10 ** -5, 5 * 10 ** -5])
delta_X = np.matmul(poi_matrix, delta_B)
norm_X = np.linalg.norm(delta_X)
print("Norm: " + str(norm_X))
relative_error = condition_num * norm_X
print("Relative error: " + str(relative_error))

# Define a function
def exponential_function(x):
    y = (math.e) ** x
    return y - 1

# Test the function
input_value = np.longdouble(9.999999995000000 * 10 ** -10)
print("Function output: " + str(exponential_function(input_value)))

# Calculate relative error iteratively
def calculate_relative_error():
    # Initialize variables
    relative_error_val = np.longdouble(10 ** 10)
    actual_value = np.longdouble(10 ** -9)
    accuracy_tolerance = np.longdouble(10 ** -16)
    g_n = np.longdouble(0)
    j = 0
    while (relative_error_val > accuracy_tolerance):
        j += 1
        g_n += np.longdouble((input_value ** j)) / np.longdouble((math.factorial
        relative_error_val = np.absolute(actual_value - g_n) / actual_value
    print("Number of terms: " + str(j))
    return relative_error_val

relative_error_input_value = calculate_relative_error()

# Define a function
def g_2_function(x):
    return x + ((x ** 2) / 2)

# Print the function output and relative error
print("g_2 function output: " + "{0:.40f}".format(g_2_function(np.longdouble(inp
print("Relative error: " + "{0:.40f}".format(relative_error_input_value))

# Calculate the sum
start_val = 0
end_val = np.pi
interval_points = np.linspace(start_val, end_val, 31)
cos_values = np.cos(interval_points)
sum_result = 0
for point, cos_val in zip(interval_points, cos_values):
    sum_result += point * cos_val
print("Sum result: " + str(sum_result))

# Plot curves
theta_values = np.linspace(0, 2 * np.pi, 1000)
amplitude = 1.2
delta_radius = 0.1
frequency = 15
phase = 0
plt.figure()
plt.plot()

plt.figure()
delta_radius = 0.05
for i in range(1, 11):
    amplitude_val = i
    frequency_val = 2 + i
    phase_val = random.uniform(0, 2)
    x_curve = amplitude_val * (1 + delta_radius * np.sin(frequency_val * theta_v
    y_curve = amplitude_val * (1 + delta_radius * np.sin(frequency_val * theta_v
    plt.plot(x_curve, y_curve)
plt.show()
```