**CST-305: Benchmark – Project 6 – Numeric Computations with Taylor Polynomials**

**Documentation**

**Responsibilities: We both worked on the code and documentation, while checking with each other for verifying the hand work**

**System Performance Context: The efficiency of a computer system is influenced by numerous elements. These elements comprise the velocity of the processor and memory, storage capacity, data volume, and the nature of tasks executed on the system.**

**Specific Problem Solved: The challenge at hand involves employing numerical techniques to solve a differential equation (DE) for simulating computer performance. Specifically, we're utilizing Taylor expansion for this purpose. Furthermore, we aim to introduce a performance model for the system. Our objective entails establishing a relationship that optimizes performance while concurrently minimizing costs to the greatest extent feasible.**

**Mathematical Approach:**

**We will be using the Taylor Polynomial formula, which consists of a reasonable function, f, and an input value, x0, and a positive integer n where the input x we will have**

**For some c in between x0 and x**

**Implementation in Code:**

**To implement this, we first need to import NumPy, and matplotlib. Next we define the Taylor Series function which is 1-x-1/3(x)^3-1/12(x)^4. It takes in all the x values as the parameter. Then we initialized the x and y values. Since we are evaluating at x = 3.5. Then we called Taylor series function to get the y for the Taylor series. Then we plot the values.**

**For part 1B, we used the same imports. This time we define the Taylor polynomial function which is 6 + x-3-11/2\*(x-3)^2. It takes in all the x values as the parameter. Then we initialized the x and y values. Then we called Taylor series function to get the y for the Taylor series. Then we plot the values.**

**For part 2 we used the same import and added in the SciPy odeint import. We first define our system of ODEs which takes in u and x. Then we setup our initial conditions. Next we find our X valus range. Then we solve the ODE. After solving the ODE, we plot it.**

**Part 1:**

1. Solve the differential equation and manually construct the Taylor expansion of . The initial conditions are . Find the value of at the point 3.5. Calculate up to . Write a Python program, define and discuss the convergence constraints and visualize the Taylor series and its convergence.

The analysis is presented in a separate document, where we utilize the Taylor series up to its first four terms. Subsequently, we compute the first, second, third, and fourth derivatives of the function, evaluating them at the point 3.5, resulting in a value of -29.2969. The convergence criteria indicate that the series converges for |x| < R, where R represents the radius of convergence. This radius is determined by the behavior of the coefficients, calculated using the formula an = (-an-2)/(n(n-1) – 2n+1). By taking the limit as n approaches infinity, we ensure that | (-an-1(n-1)(n-2)+2(n-1))/((n+1)n) | < 1, ultimately leading to the conclusion that the ratio tends to zero, indicating convergence for all values of x.

**Screenshots:**

**A graph with a blue line

Description automatically generated**

1. Find the second order Taylor polynomial near of with the initial value problem . Write a Python program, define and discuss the convergence constraints and visualize the Taylor polynomial, and its convergence.

**The analysis is presented in a separate document where we examine the first two terms of the Taylor series. Subsequently, we calculate the first two derivatives of the function and arrive at the solution: 6 + x-3 – (-11(x-3)^2)/2. The convergence of the function occurs either at x = 3, where y(3)’’ is finite, or at infinity if it is unbounded.**

**Screenshot:**

**A graph of a function

Description automatically generated**

**Part 2:**

Determine whether the given value of x is an ordinary point of the given differential equation

If x is an ordinary point, find the recurrence formula and then the general solution of the given differential equation. Solve the question manually for and then write a Python program to solve the differential equation numerically.

We found that x is an ordinary point. So in a separate file we showed the scratch work of finding the recurrence formula and then the general solution of the given differential equation. Then we computes a python program that solves the problem as well.

Screenshot:

A graph with a line

Description automatically generated

**Part 3:**

The relationship between temperature and power consumption in a computer is often associated with the efficiency and performance of the computer's components, particularly the central processing unit (CPU) and the graphics processing unit (GPU). This relationship is also influenced by factors such as the efficiency of the components, the cooling solutions in place (like fans or liquid cooling), and the workload the computer is handling.

The specific problem that was solved with the ODE was the temperature of the computer as the power consumption increase. This is important for because monitoring the temperature of a computer is crucial because it helps in maintaining the system's stability, performance, and overall health. As power consumption increases, components such as the CPU and GPU generate more heat.

The mathematical approach for solving this was using the basic heat transfer equation for conduction. The formula would be Q= h \* A \* ΔT. This equation is commonly used to describe the rate of heat transfer through a material when there is a temperature gradient across it.

* *Q* is the heat transfer (in watts or joules).
* *h* is the heat transfer coefficient, which represents the ability of the material to conduct heat (in watts per square meter per degree Celsius).
* *A* is the surface area through which heat is transferred (in square meters).
* Δ*T* is the temperature difference between the two sides of the material (in degrees Celsius).

Since we are looking for the change the temperature as the power consumptions we would have to adjust the equation.

Now, differentiate both sides with respect to time (*t*) to find the rate of change of temperature (*y*) with respect to time:

**dQ/dt=dh⋅A⋅ΔT/ dt**

**Since *Q* is the heat dissipated (temperature change), and *P* is the power consumption, we can rewrite the equation as:**

**dydt= 1/ h⋅A**

This is the differential equation representing the rate of change of temperature with respect to time, assuming a constant thermal resistance (h⋅A*)*. It implies that the temperature change is inversely proportional to the thermal resistance. This is a simplified model, and real-world scenarios may involve more complex dynamics and factors.

Screenshot:

A screen shot of a graph

Description automatically generated

Flowchart:

A diagram of a function

Description automatically generated with medium confidence

GitHub link:

<https://github.com/angel-vlzqz/Modeling-and-Simulation/tree/main/projects/project%206%20CLC>