CST-305: Project 1 – Visualize ODE With SciPy

Jack Utzerath

College of Technology, Grand Canyon University

CST-305: Principles of Modeling and Simulation

Ricardo Citro

9/14/23

Responsibilities and completed tasks by each team member

All task and programs completed by Jack Utzerath

System performance context description

One performance metric that uses ordinary differential equations to solve is the heat dissipation over time. Specifically, this project focuses on the heat dissipation of the computer. When talking about the heat dissipation, the central processing unit is the central idea. It is important for computer systems to maintain optimal operating temperatures. If a computer system overheats, damage to hardware could decrease performance in the future which may be costly. Efficient heat dissipation is vital to prevent this from happing.

Specific problem solved

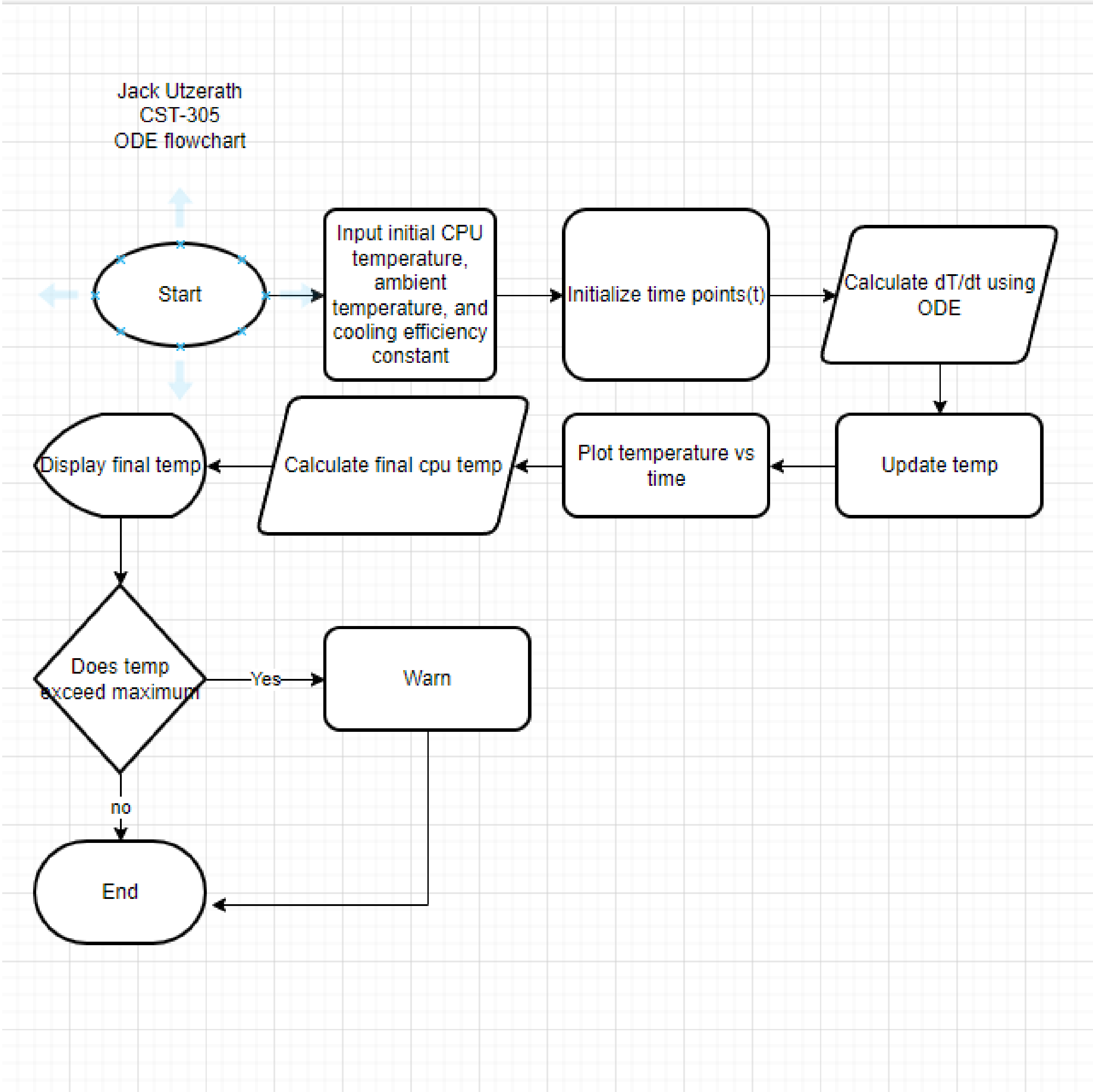
The problem that this project aims to address is controlling the CPU's thermal performance. In the real world, cooling efficiency and temperature of the CPU have to be closely monitored to ensure optimal performance.

The mathematical approach for solving it

The heat dissipation in a CPU can be modeled using Newton's Law of Cooling. This law has a couple of components to it. One being the temperature of the CPU (T). The $T_{ambient}$ is the room temperature and k is the cooling efficiency.

Equation: $$dT/dt = -k(T - T_{ambient})$$

The approach for implementation in code (e.g., algorithm, flowchart)



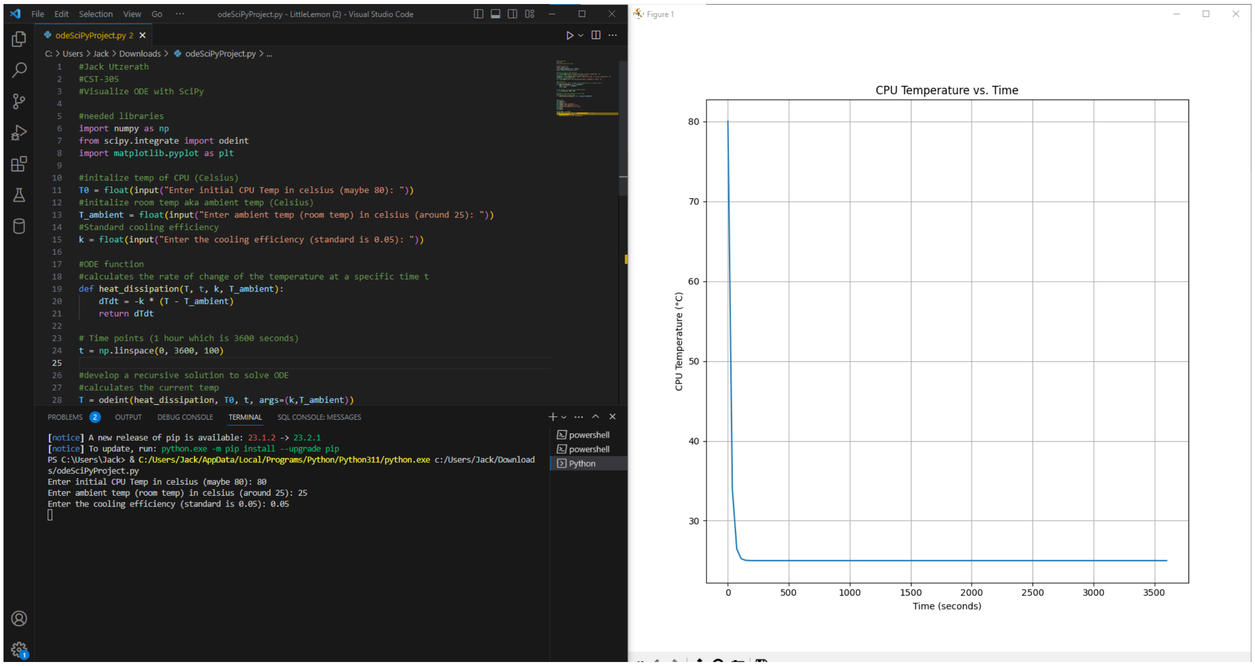
Screenshots depicting key phases in the program execution

odeSciPyProject.py 5 ●

C: > Users > Jack > Downloads > odeSciPyProject.py > ...

```python
#Jack Utzerath
#CST-305
#Visualize ODE with SciPy

#needed libraries
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

#initalize temp of CPU (Celsius)
T0 = float(input("Enter initial CPU Temp in celsius (maybe 80): "))
#initalize room temp aka ambient temp (Celsius)
T_ambient = float(input("Enter ambient temp (room temp) in celsius (around 25): "))
#Standard cooling efficiency
k = float(input("Enter the cooling efficiency (standard is 0.05): "))

#ODE function
#calculates the rate of change of the temperature at a specific time t
def heat_dissipation(T, t, k, T_ambient):
    dTdt = -k * (T - T_ambient)
    return dTdt

# Time points (1 hour which is 3600 seconds)
t = np.linspace(0, 3600, 100)

#develop a recursive solution to solve ODE
#calculates the current temp
T = odeint(heat_dissipation, T0, t, args=(k,T_ambient))


#make plot
plt.figure()
plt.plot(t, T)
plt.xlabel('Time (seconds)')
plt.ylabel('CPU Temperature (°C)')
plt.title('CPU Temperature vs. Time')
plt.grid()
plt.show()

final_temp = T[-1][0]
print(f"The final CPU temp is {final_temperature:.2f} °C.")
if final_temperature > 85:
    print("Warning: The CPU overheated")

```

Program Execution



References for theory and code sources

"Newton's Law of Cooling." *Carolina Knowledge Center*, 8 Mar. 2023, knowledge.carolina.com/discipline/physical-science/physics/newtons-law-of-cooling/.

"Pynotes in Agriscience." *Newton's Law of Cooling - Pynotes in Agriscience*, soilwater.github.io/pynotes-agriscience/notebooks/newton_law_cooling.html. Accessed 14 Sept. 2023.

README document written in Markdown, detailing how to install and run the program

This python script file can be executed using linux or other IDEs like visual code. Necessary libraries- numpy, scipy, and matplotlib

This project is uploaded on my GitHub

Github- utzerath