Blake Van Dyken

Assignment 1

Methods Used

- Newton's Law of Cooling
  - $$\frac{dT_c}{dt} = -r(T_c - T_s)$$
  - I used this law to compute the temperature of the coffee cup at the given rate, cup temp, and room temp
- Forward Euler method
  - $$y_{n+1} = y_n + hf(t_n, y_n).$$
  - Euler's method is used for numerical integration and we were given an integral; thus, I used this equation above to compute the next temperature(y_n+1) of the coffee sup based on the stepsize (h) and where $f(t_n, y_n).$ is Newton's Law of cooling and y_n is our previous temperature.
- Trapezoidal Euler algorithm
  - This is another method that can solve the coffee cup problem and uses the previous method in its calculation. I followed the below image provided in class to help me come up with the algorithm:

EULER–TRAPEZOIDAL METHOD ALG.

1. START WITH n SET $= 0$
2. CALCULATE $y_{n+1}^{(1)} = y_n + h\, f(t_n, y_n)$
3. CALCULATE $f\left(t_{n+1}, y_{n+1}^{(1)}\right)$ WHERE $t_{n+1} = t_n + h$
4. FOR $K = 1, 2, \ldots$ CALCULATE:

$$y_{n+1}^{(k+1)} = y_n + \frac{h}{2}\left[f(t_n, y_n) + f\left(t_{n+1}, y_{n+1}^{(k)}\right)\right]$$

CONTINUE THE K LOOP UNTIL THE DIFFERENCE BETWEEN SUCCESSIVE VALUES OF y_{n+1} IS SMALL, FOR EXAMPLE $y_2 - y_1 < 0.05$, THEN YOU INCREMENT n BY 1 AND REPEAT STEPS 2, 3, & 4.

YOU CAN EXPERIMENT WITH DIFFERENT VALUES OF THE TOLERANCE, i.e. TRY 0.01, 0.005, etc., INSTEAD OF 0.05.

Results and Figures

Analytical Answer for 5 minutes:

$$\frac{dT_c}{dt} = -r(T_c - T_s)$$

$T_s = 19°C \quad T_c = 84°C \quad r = .025/sec$

$$\frac{dT}{(T_c - T_s)} = -r\,dt$$

$$\int \frac{dT}{(T_c - T_s)} = \int -r\,dt$$

$$\ln(T_c - T_s) = -r \cdot t + C$$

$$e^{\ln(T_c - T_s)} = e^{-rt + C}$$

$$T_c - T_s = Ce^{-rt}$$

$$T_c = T_s + C \cdot e^{-rt}$$

$$T_0 = T_s + c_1 \cdot e^{r \cdot 0}$$

$$c_1 = T_0 - T_s$$

For 5 minutes:

$T_0 = 84 \quad t_0 = 0$

$t_1 = 5$ min

$$T_1 = 84 + 5 \cdot -.025(84 - 19)$$

$$T_1 = 75.875$$

or $$Y_{N+1} = Y_N + h \cdot f(t_N, Y_N)$$

Forward Euler:

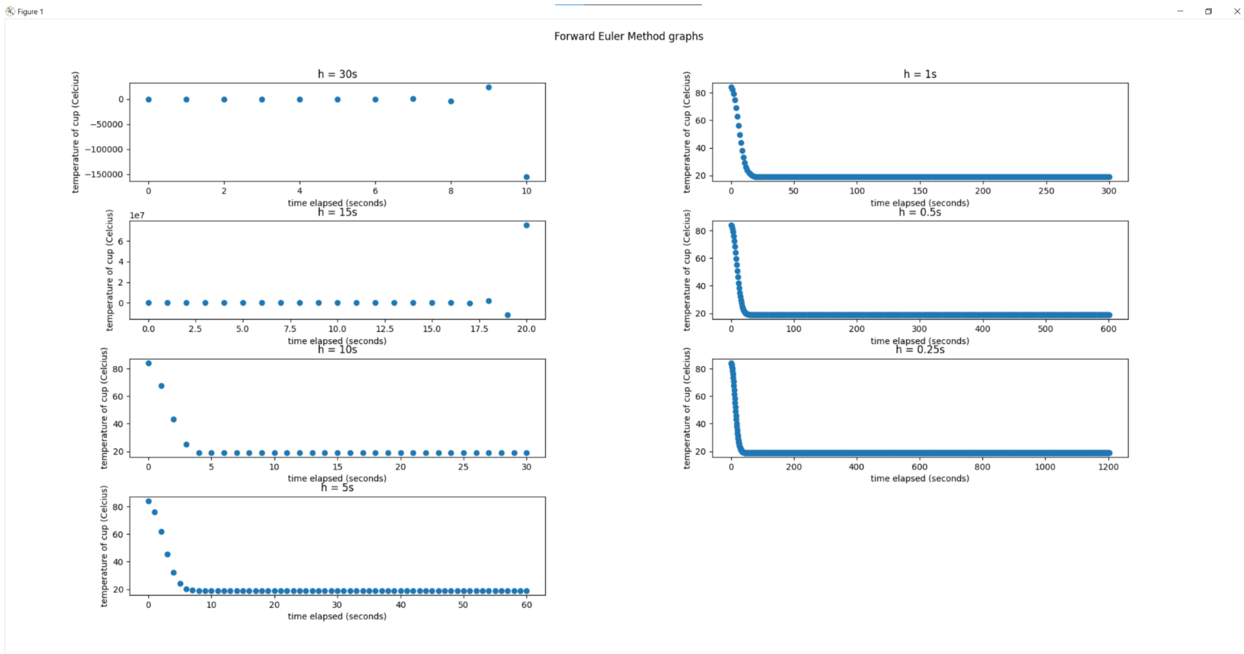
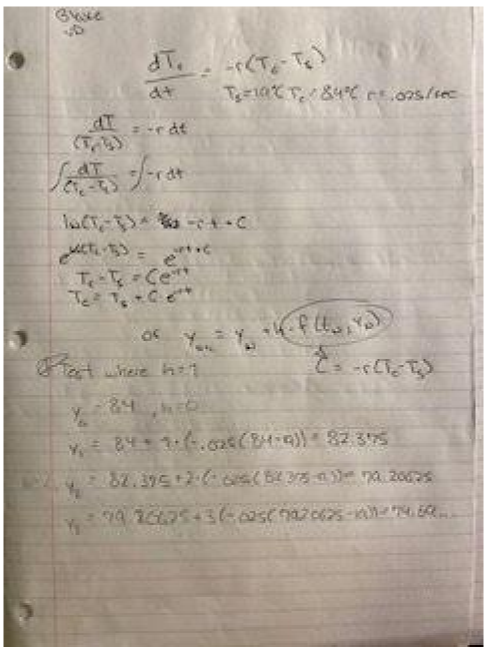


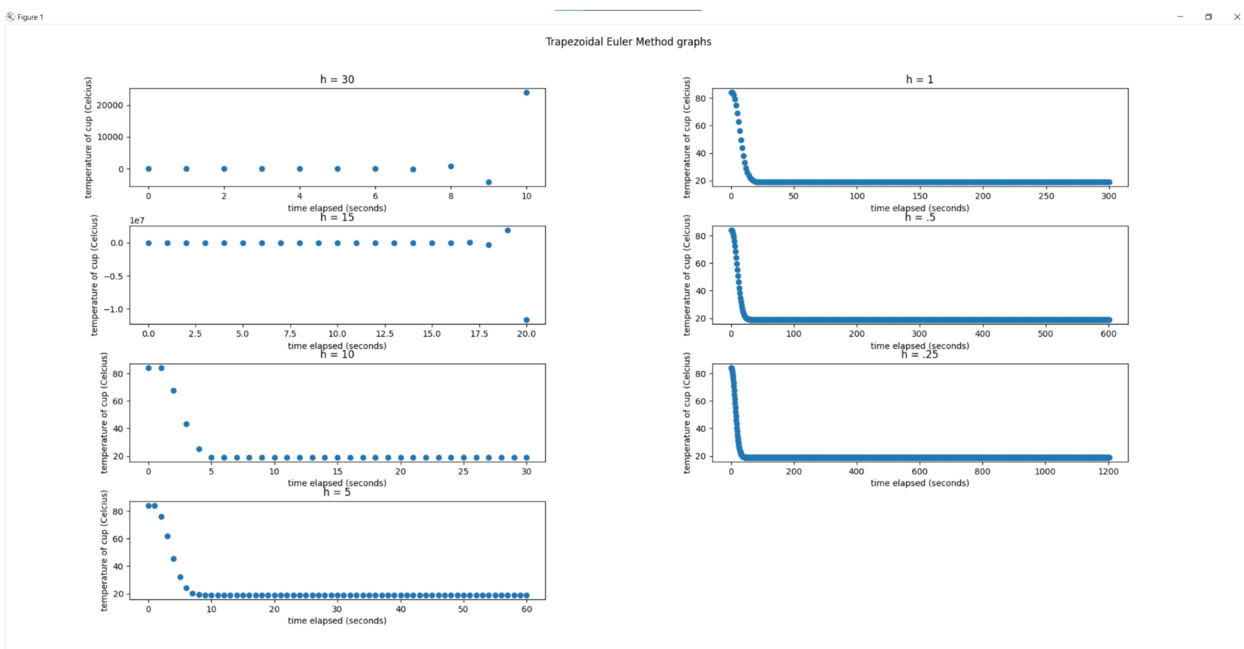
Figure 1
Forward Euler Method graphs
h = 30s
h = 15s
h = 10s
h = 5s
h = 1s
h = 0.5s
h = 0.25s
temperature of cup (Celcius)
time elapsed (seconds)
1e7

```
h = 1:
 [84.0, 82.375, 79.20625, 74.69078125, 69.121703125, 62.856490234374995,
.18429192881066, 20.990182455506663, 20.194109473303996, 19.686612947149
65556, 19.000472779179447, 19.00015365323332, 19.000046095969996, 19.000
000000000351, 19.000000000000174, 19.000000000000004, 19.0, 19.0, 19.0, 
9.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19
.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19
0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.
, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0
 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0,
19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 
9.0, 19.0, 19.0]
```



Looking at the forward Euler method, it is very straightforward and first, we should point out the first two graphs when **h is 30 seconds and 15 seconds**. There is a very big **error** in these graphs because the h value is so spread apart, that when we take measurements every 30 or15 seconds, we are dependent on our previous measurement, and thus when waiting too long to measure it messes the equation up and causes either very large numbers, negative and positive or just numbers close to zero. Also, since we are only measuring on However, once we put the step size to about 10 seconds, we can see we get a readable chart. As we decrease h or the time between when we take measurements, we get more accurate results as we see when h decreases we get a steeper downward curve that will more accurately reflect the temperature of the coffee cup. Looking at my analytical answer above for h = 1 second, comparing it to the screenshot of values outputted by my program(in black), we see that the first three analytical values I computed. Also, we get fewer data points as we are only measuring every 30 seconds on a five-minute interval.

Trapezoidal Method:



Trapezoidal

1) u set = 0

2) $Y_{u+1} = Y_u + h \cdot f(t_u, Y_u)$ = Forward Euler

3) calc $f(t_{u+1}, Y_{u+1})$; $t_{u+1} = t_u + h$

4) for k = 1, 2

$$Y_{k+1} = Y_k + \frac{h}{2}\left[f(t_u, Y_u) + f(t_{u+1}, Y_{u+1}^{(k)}) \right]$$

$= -r(T_t - T_s)$

$= f(t_0 + h, Y_0)$

so

$$Y_1 = Y_0 + \frac{h}{2} \cdot \underbrace{\left(-r(T_t - T_s) + \underbrace{f(t_1, Y_1)}_{\text{k loop}}\right)}_{\text{in i loop}}$$

Trapezoidal : test h = 1 sec

$Y_0 = 84$ · 0 · , h = 0

$Y_1 = 84 + 1 \cdot -0.025(84 - 19) = 82.375$

$-r(T_t - T_s)$

(K_0) = $f(t_{u+1}, Y_{u+1}) = -0.025(84 - 19) = -1.625$

for k = 1, 2, 3, ... i = 1

(K_1) $K_1 = -.025(82.375 - 19) = -1.584375$

then if $K_1 - K_0 < .05$

✓ $-1.584375 - -1.625 = 0.04...$

$Y_2 = 82.375 + \frac{1}{2}(-1.584375 + -1.625) =$

(white spot is 79.2)

```
[84.0, 84.0, 82.375, 79.20625,
8, 22.18429192881066, 20.990182
50797655558, 19.000472779179447
19.00000000000351, 19.000000000
.0, 19.0, 19.0, 19.0, 19.0, 19.
0, 19.0, 19.0, 19.0, 19.0, 19.0
, 19.0, 19.0, 19.0, 19.0, 19.0,
 19.0, 19.0, 19.0, 19.0, 19.0,
19.0, 19.0, 19.0, 19.0, 19.0, 1
9.0, 19.0, 19.0, 19.0, 19.0, 19
.0, 19.0, 19.0]
h =  1
```

When using the Trapezoidal Euler algorithm, we can see that we get the same type of **error** as I described in the forward Euler. Comparing it to the analytical answer when h is 1, I get the same values as the ones computing in the black image printed out in the console when running my program. Looking at the graphs we also get similar results as the trapezoidal method. The trapezoidal method needs fewer steps to be accurate whereas the forward method needs more steps to be accurate, thus this algorithm is better for computing the coffee temperature. When experimenting with different values of tolerance in this method, the larger the value the less accurate the results but the faster it takes to compute, and the smaller the threshold the more accurate our curve will be. Also, the smaller our h value, the more downwards steeper and more accurate our curve is.

Sources:

All my code is submitted with the project and on GitHub here:

https://github.com/iPupkin/Theory-3200