Project 1 Analysis

jackj21

February 12, 2021

Analysis

1. The total run-time and the average time per time step for $m_x = 9$, $m_y = 7$, $T = 5$ and $n = 11, 26, 51, 101, 201, 301$:

n	dx	dt	nt	Total Run-Time	Avg Time Per Time Step
11	0.100000	0.070711	71.0	0.004480	0.000517
26	0.040000	0.028284	177.0	0.027940	0.003215
51	0.020000	0.014142	354.0	0.105547	0.013073
101	0.010000	0.007071	707.0	0.419767	0.052193
201	0.005000	0.003536	1414.0	1.736602	0.218724
301	0.003333	0.002357	2121.0	3.784307	0.477018

Figure 1: Table of values for dx, dt, nt, and run times for each value of n.

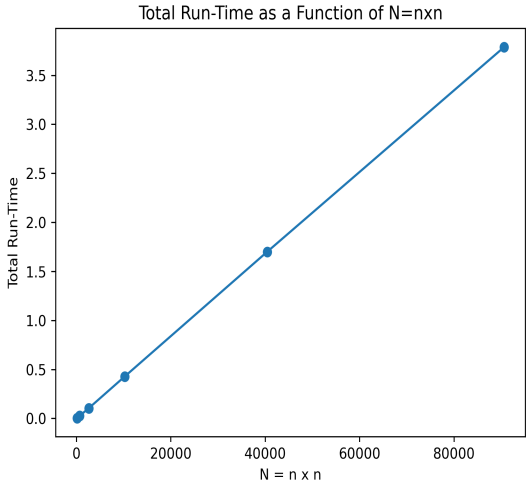

Figure 2: Plot for total run-time as a function of $N = nxn$.

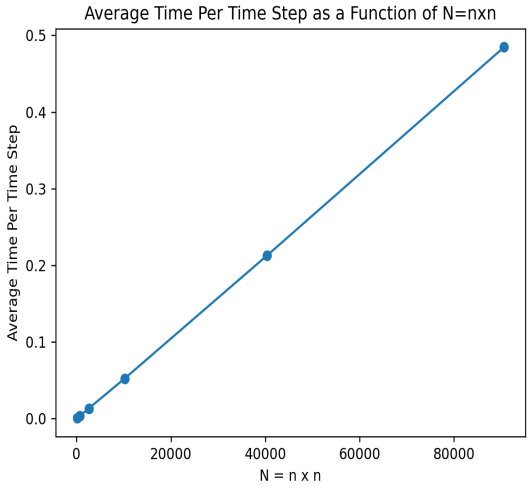

Figure 3: Plot for average time per time step as a function of $N = nxn$.

2. If $n = 100,000$ and $T = 5$, setting $m_x = 12$ and $m_y = 10$, the number of time steps required for a stable solution such that $\alpha = 1$ can be calculated based off a model created from data retrieved in the previous task. It can be observed from figure 1 that there appears to be a linear trend between n and the number of time steps/iterations, n_t. I created a model in Python using numpy's polyfit function to obtain a linear model for the 2 variables. The code is shown below:

```
# Task 2
fig3 = plt.figure()
x = n
y = n_t
plt.scatter(x, y)
model = np.polyfit(n, n_t, 1)
a = model[0]
b = model[1]
predict = np.poly1d(model)
x_lin_reg = range(302)
y_lin_reg = predict(x_lin_reg)
plt.plot(x_lin_reg, y_lin_reg)
predict(100000)
```

Using my model, I was able to enter in $n = 100000$ as a value and my model gave me the number of time steps/iterations for n. The model returned a value of 706861.568099968, and this seems reasonable because n and n_x follow a linear trend so when $n = 100000$, n_x will be a much larger number.
Performing the calculations to find the percentage 100 iterations would solve out of the whole simulation, I simply divided $\frac{100}{706861}$ and got around 0.01% which is essentially none of it.

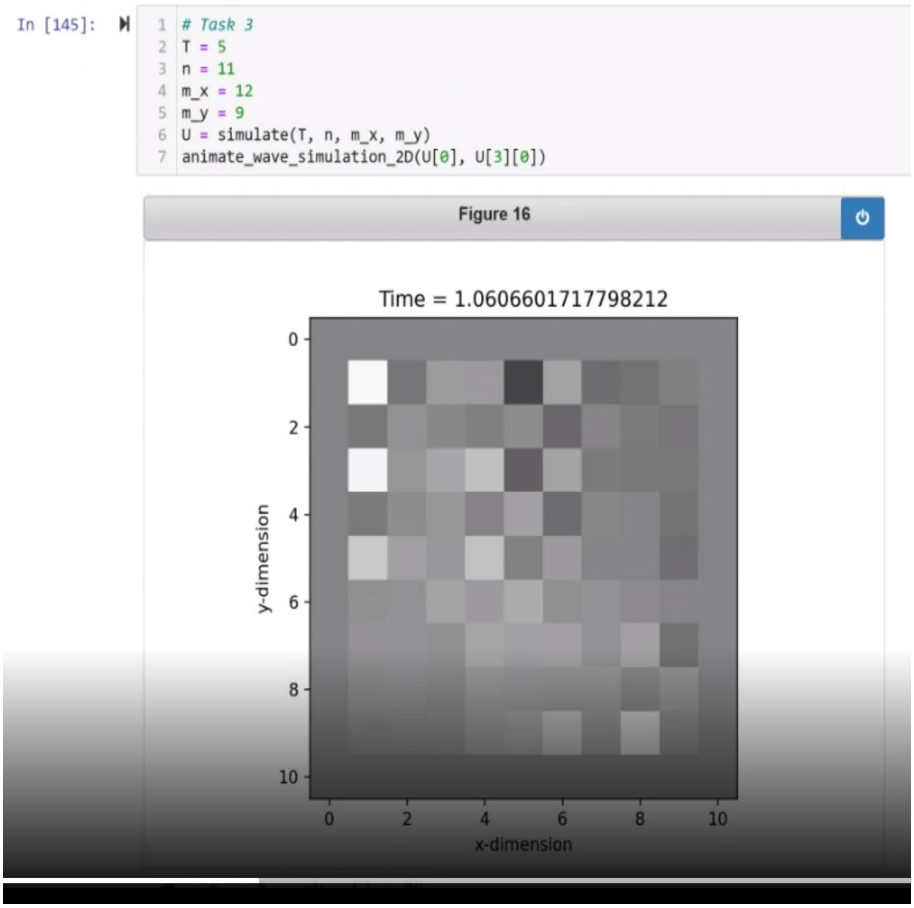

Figure 4: 2D Graph of Simulation for $T = 5 and n = 11$ at $t = 1$

3.

```
In [145]:
# Task 3
T = 5
n = 11
m_x = 12
m_y = 9
U = simulate(T, n, m_x, m_y)
animate_wave_simulation_2D(U[0], U[3][0])
```

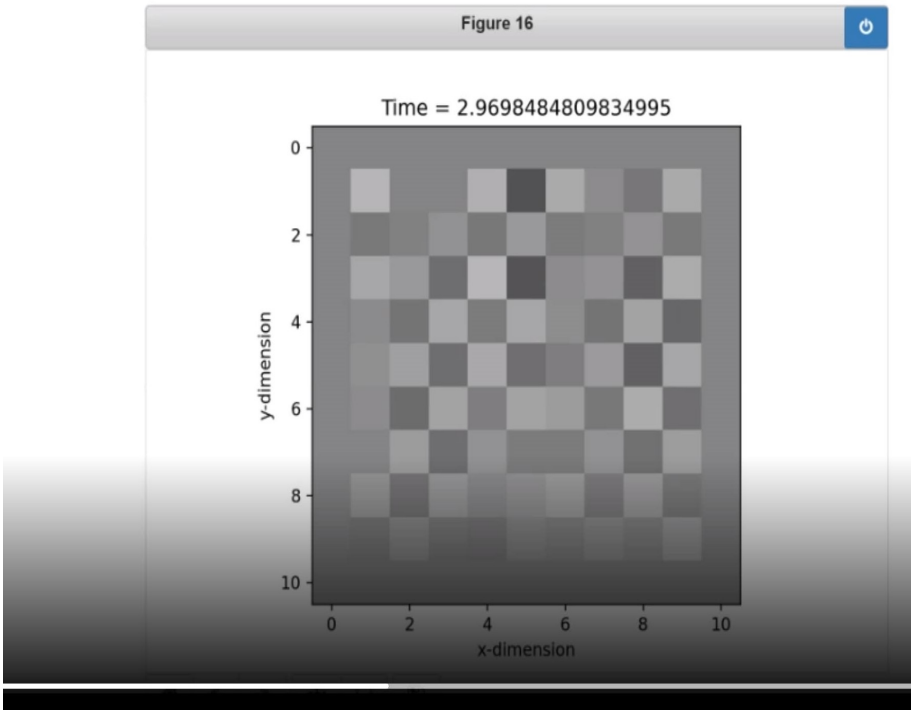

Figure 5: 2D Graph of Simulation for $T = 5 and n = 11$ at $t = 3$

These photos show the images of the solution near $t = 1, 3, 5$ when $T = 5$ and $n = 11, 101, 201$. The n-value equal to 201 takes very long for my laptop to process and compute. I believe something is wrong with my algorithm here that is messing with the computations and causing it to take too much time. Therefore, figures will no longer be added to this report and only to the gitlab repository.

4. Letting values $T = 5, n = 101, m_x = 2$, and $m_y = 3$, the simulation was run for α values 1.0 and 1.001.

   a)The latter solution of $\alpha = 1.001$ is expected to blow up. When running the simulation, it first starts to appear to blow up around time $t = 3$ seconds. Then, as the simulation time keeps going, it just gets worse and worse.
   b) Videos posted on Canvas.

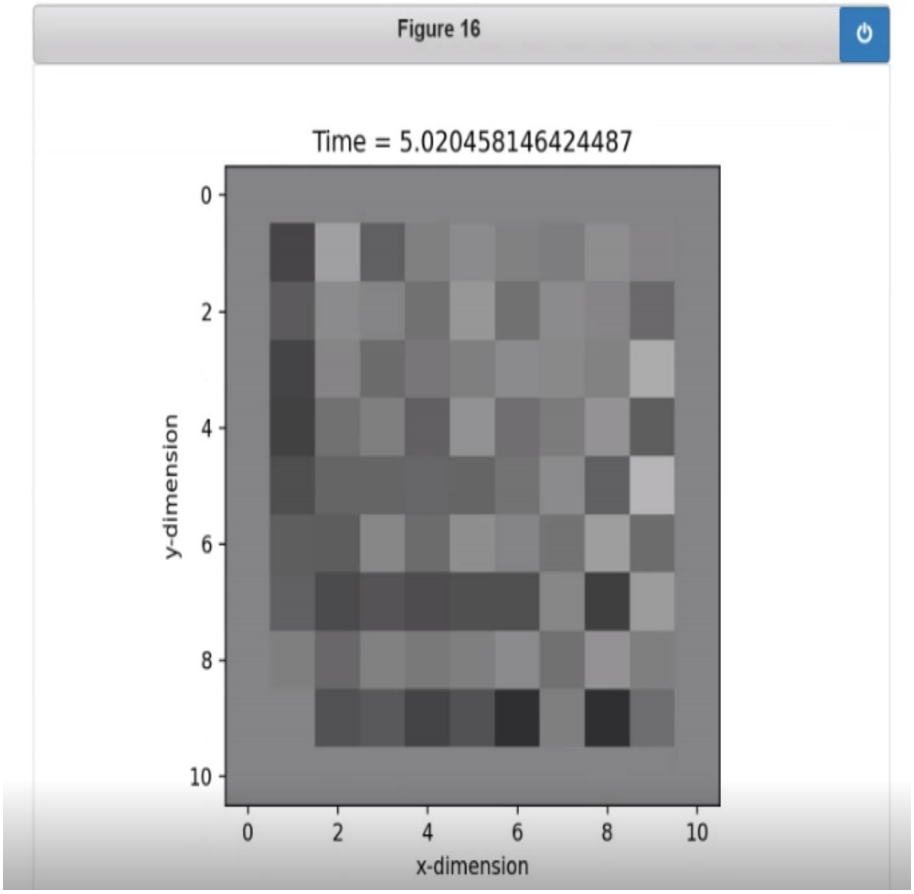

Figure 6: 2D Graph of Simulation for $T = 5 and n = 11$ at $t = 5$

```
In [145]:
# Task 3
T = 5
n = 11
m_x = 12
m_y = 9
U = simulate(T, n, m_x, m_y)
animate_wave_simulation_2D(U[0], U[3][0])
```

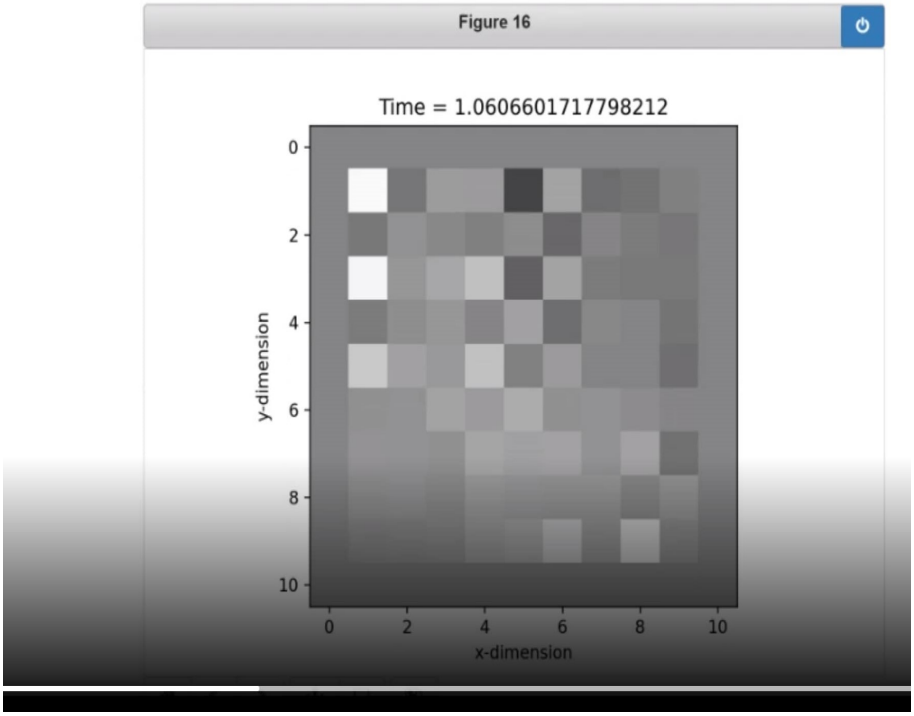

Figure 7: 3D Graph of Simulation for $T = 5 and n = 11$ at $t = 1$

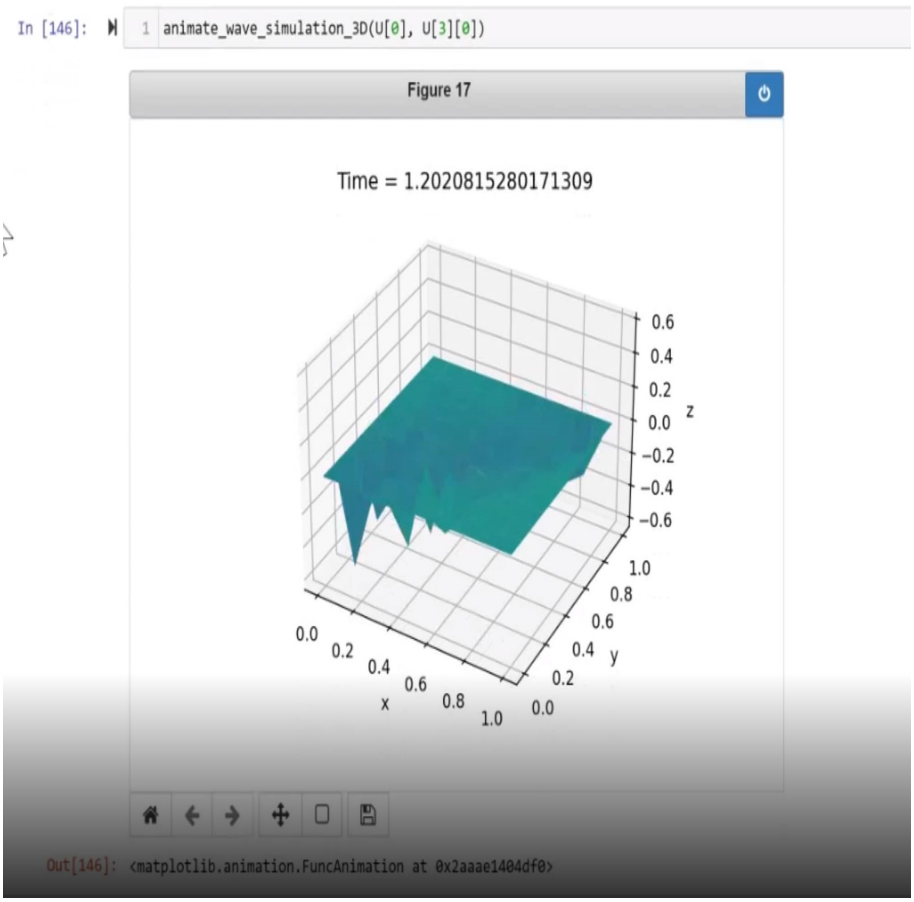

Figure 8: 3D Graph of Simulation for $T = 5$ and $n = 11$ at $t = 1$

In [145]:

```
# Task 3
T = 5
n = 11
m_x = 12
m_y = 9
U = simulate(T, n, m_x, m_y)
animate_wave_simulation_2D(U[0], U[3][0])
```

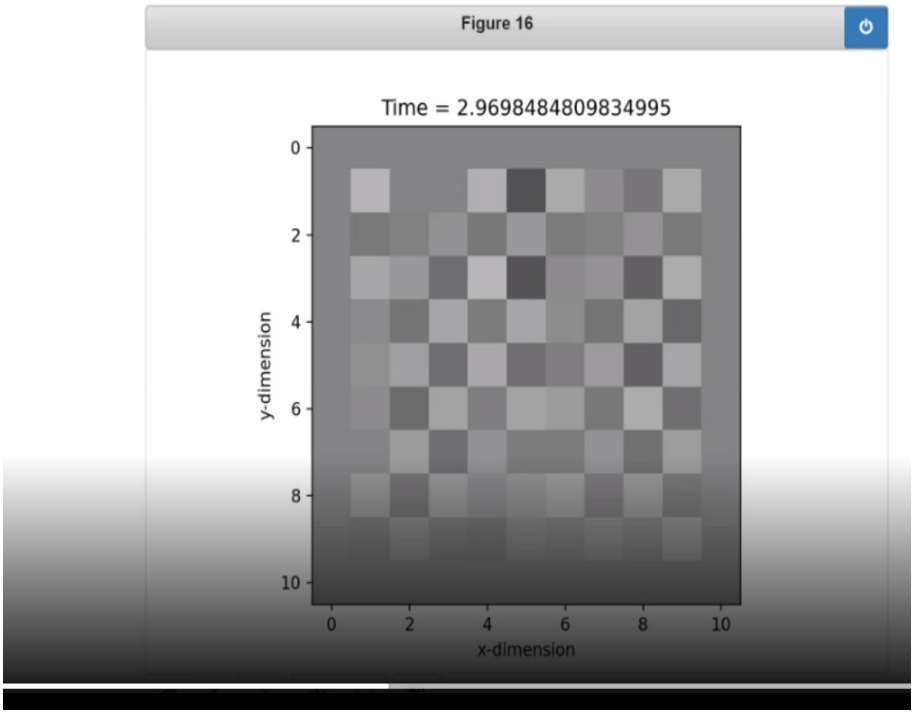

Figure 9: 3D Graph of Simulation for $T = 5$ and $n = 11$ at $t = 3$

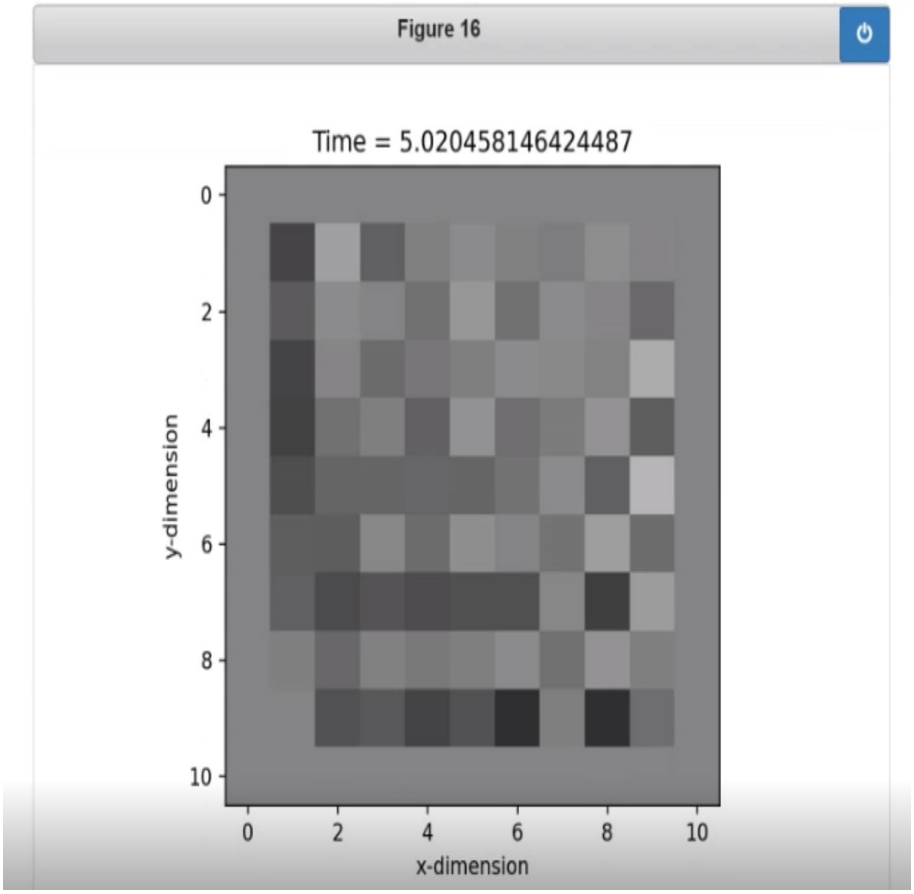

Figure 10: 3D Graph of Simulation for $T = 5$ and $n = 11$ at $t = 5$

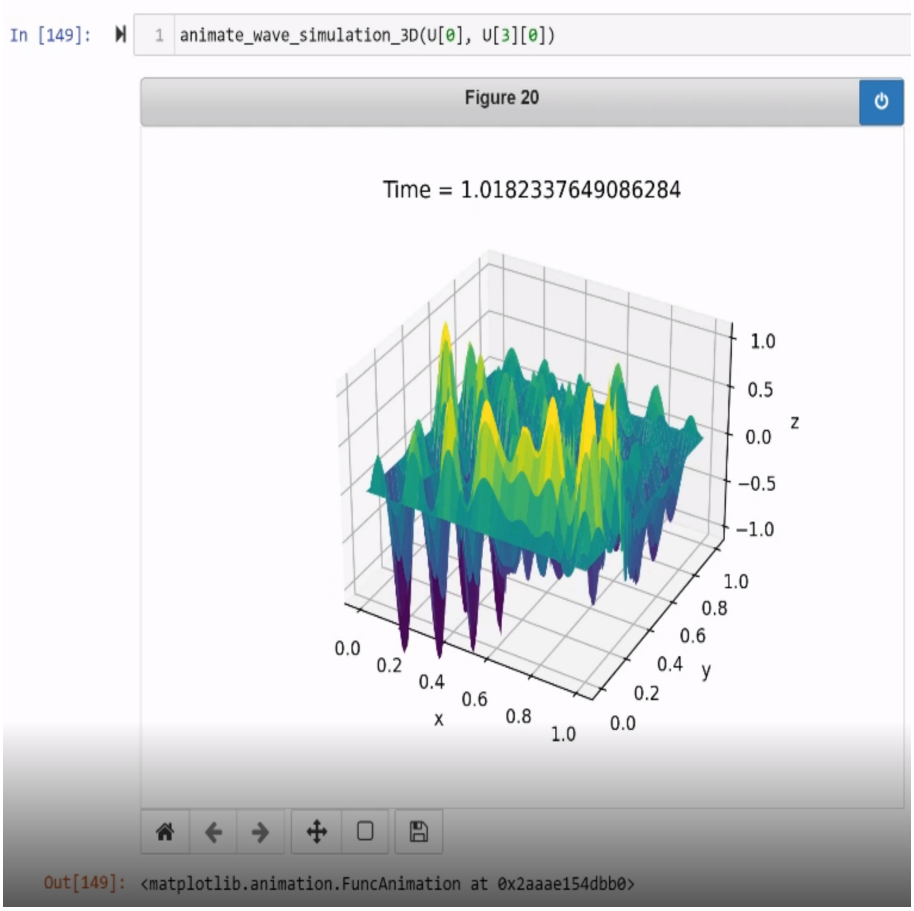

Figure 11: 3D Graph of Simulation for $T = 5$ and $n = 101$ at $t = 1$

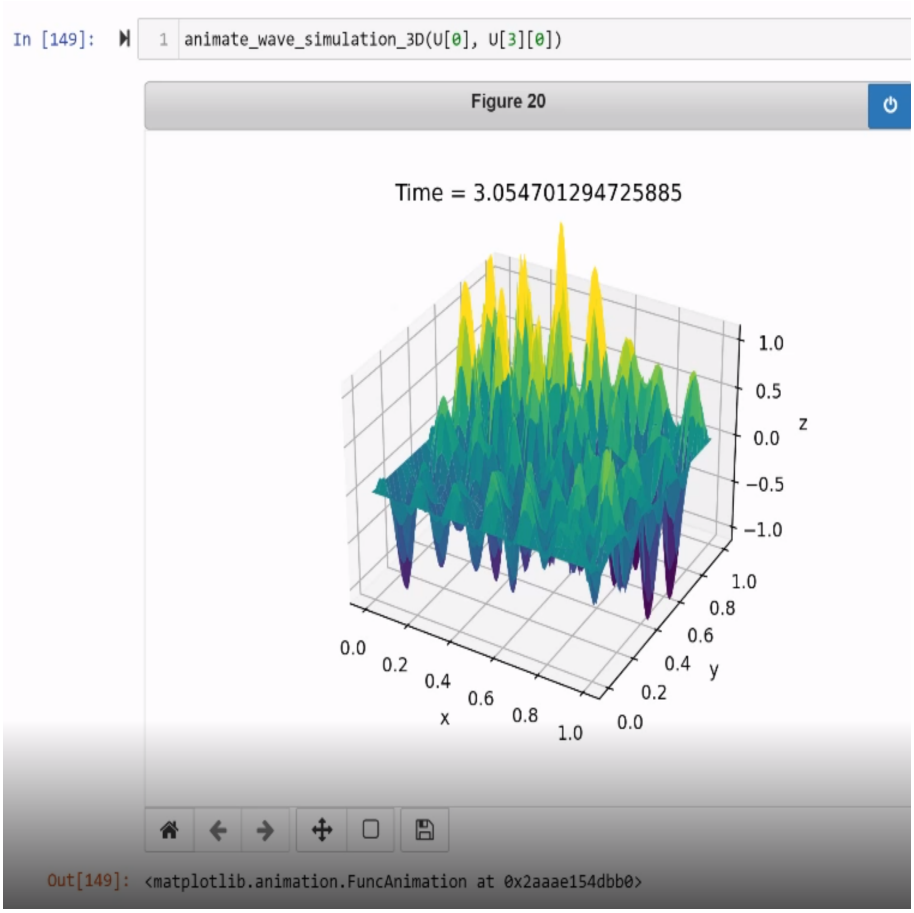

Figure 12: 3D Graph of Simulation for $T = 5$ and $n = 101$ at $t = 3$

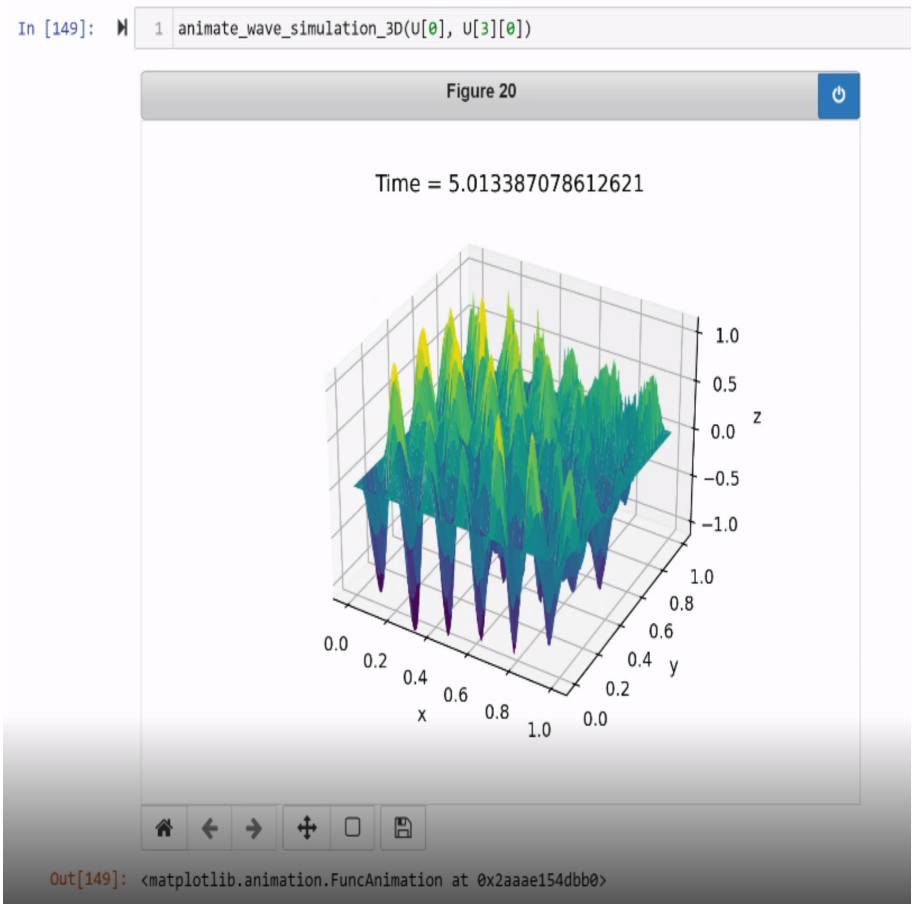

Figure 13: 3D Graph of Simulation for $T = 5$ and $n = 101$ at $t = 1$