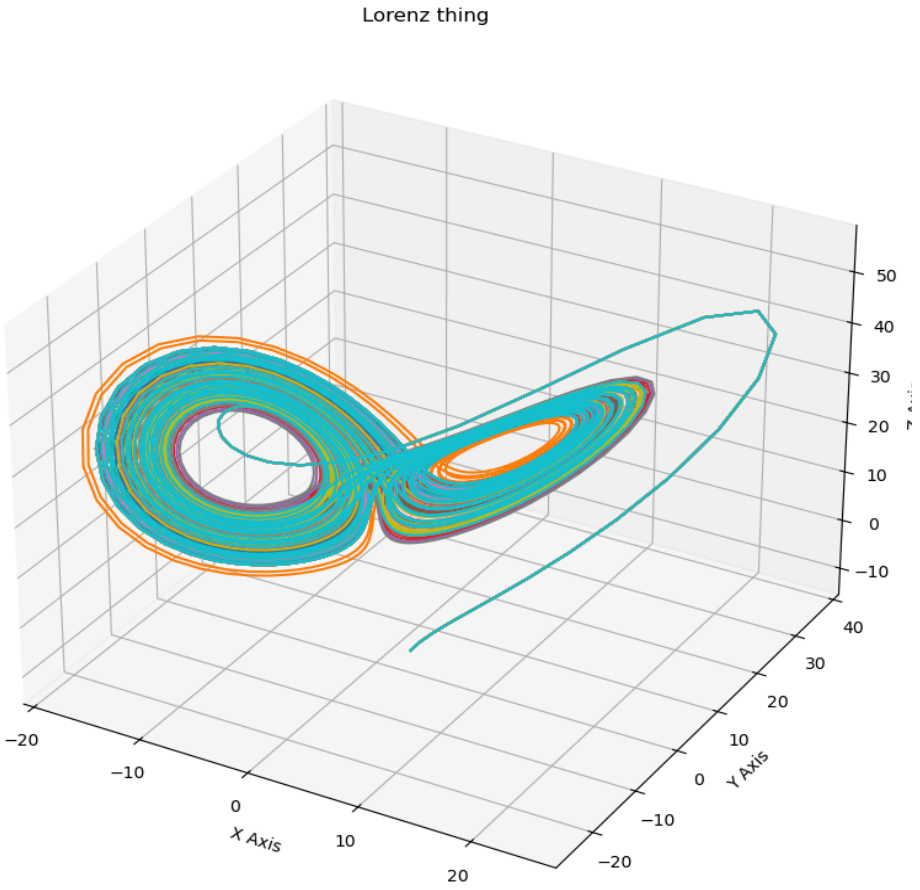
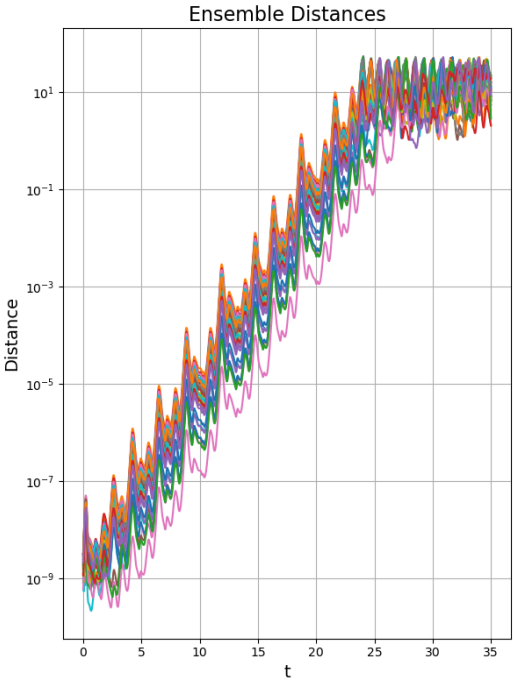
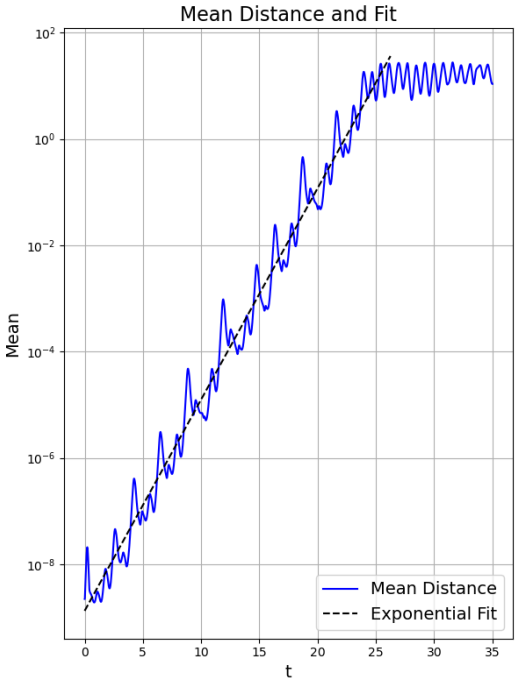
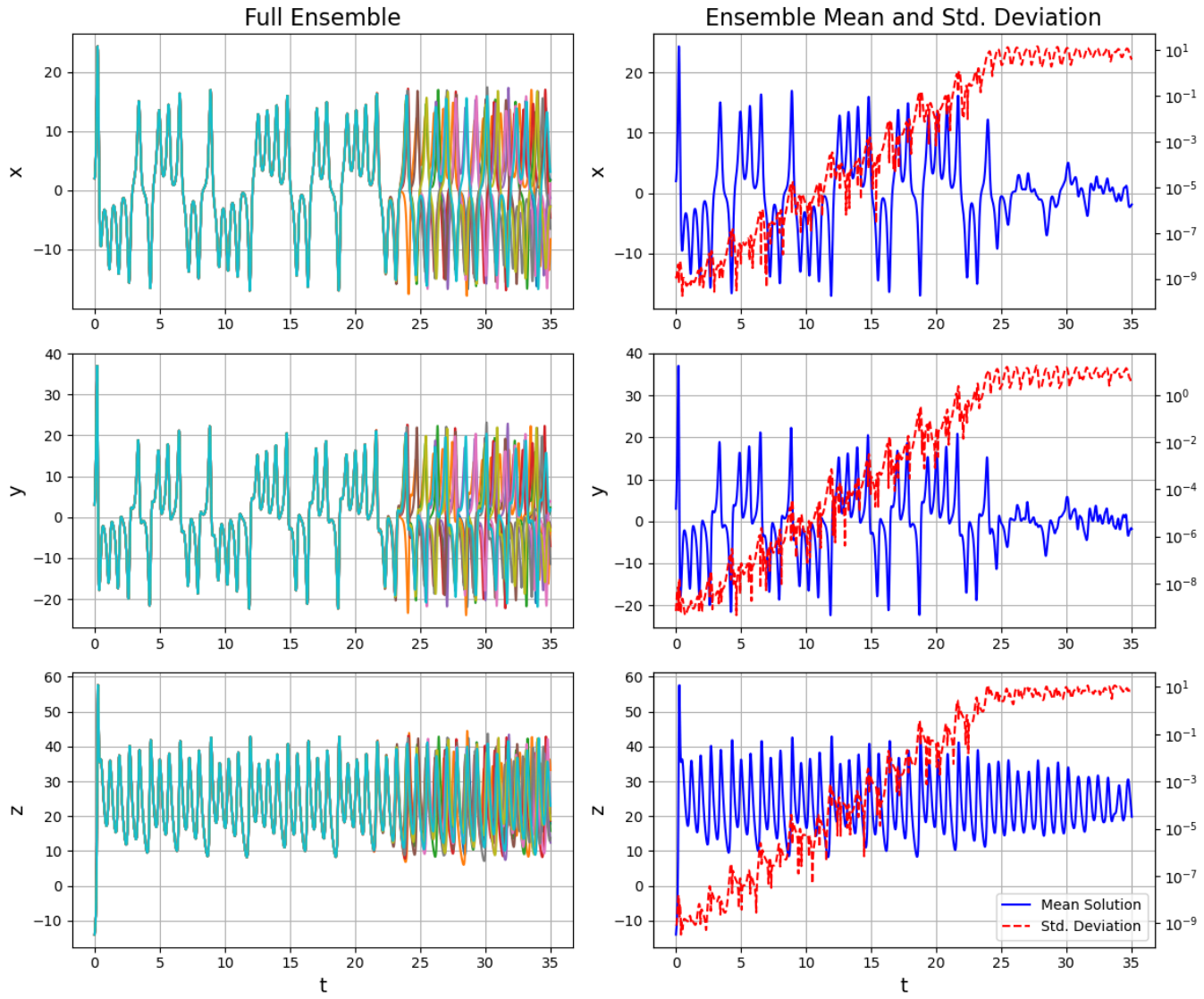
HW4 Lorenz System

1) We defined an empty array sol_list in our solveEnsemble function and filled it with matrices of size 3x2000, where the rows are our spatial values x, y, z, and the columns represent uniform time steps for our initial conditions. A matrix is appended to sol_list an nEnsemble number of times, which we defined as 10 at the very bottom, so sol_list is basically a list of lists.
2) ensembleStats calculates the average of the 3x2000 lists in sol_list and uses that to calculate the standard deviation of those elements in sol_list. We see the plots of these values in the first image we produce. ensembleStats is used in the plotDistances function to plot the average distance alongside the distances themselves to allow for easy comparison. Using this average we also compute an exponential fit to approximate the maximal Lyapunov exponent.. It is also used in the plotEnsemble function for the sake of comparing the change in our chaotic system's initial conditions individually to their average.
3) To calculate the distance between two solutions, I googled something to get an idea of how to compute distance in python and found code about a euclidean norm. Using numpy arrays we can use a function called np.linalg.norm which automatically computes a magnitude for vectors/matrices (in this case for our columns, using axis = 0), but I wanted to compute it a little more manually because that's how my brain works. Using the diff variable (final element - initial element), I squared each instance in a loop, added them together with np.sum and took the square root of the whole thing in np.sqrt, basically translating $\sqrt{\sum(x_f - x_0)^2}$, my arbitrary solution by hand into code even though it probably could've been done much cleaner.
4) The Lyapunov exponent my code predicts is 0.919073. According to the site for our homework this value is off by -0.015673. Immediately I think discrepancies arise from the fact that we use random seeds in numPy to create our initial conditions. In office hours we noticed that our approximation is consistently slightly greater than the known value, whose reason I'm not entirely sure of. If random seeds were the sole reason I'd believe we would also see undershoots.
5) For my plot, I looped over the rows of each element in sol_list (so each x,y, and z value of our initial conditions) and plotted them in a 3D space. It uses the argument sol_list defined in solveEnsemble and matplotlib which we imported at the very start. I called it in my last line of code.


Lorenz thing
X Axis
Y Axis
−20
−10
0
10
20
30
40
50


Ensemble Distances
Distance
t


Mean Distance and Fit
Mean
t
Mean Distance
Exponential Fit


Full Ensemble
Ensemble Mean and Std. Deviation
x
y
z
t
Mean Solution
Std. Deviation