Samantha Sun

BIOEN 585

20190501

Lab 5: Modeling Stochastic Chemical System

# Question 1: Solve Artyomov model

1. Solved for system four times using the DSDEexact stochastic solver:

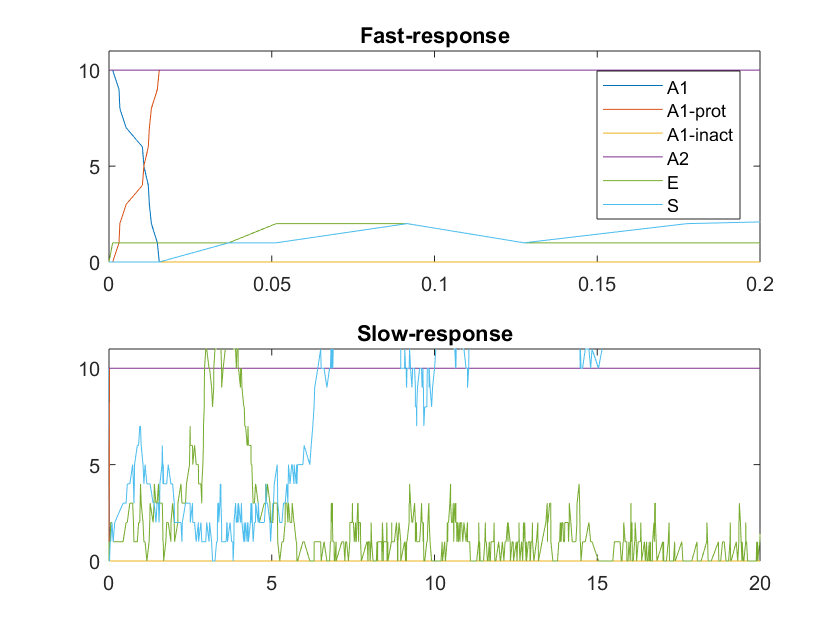
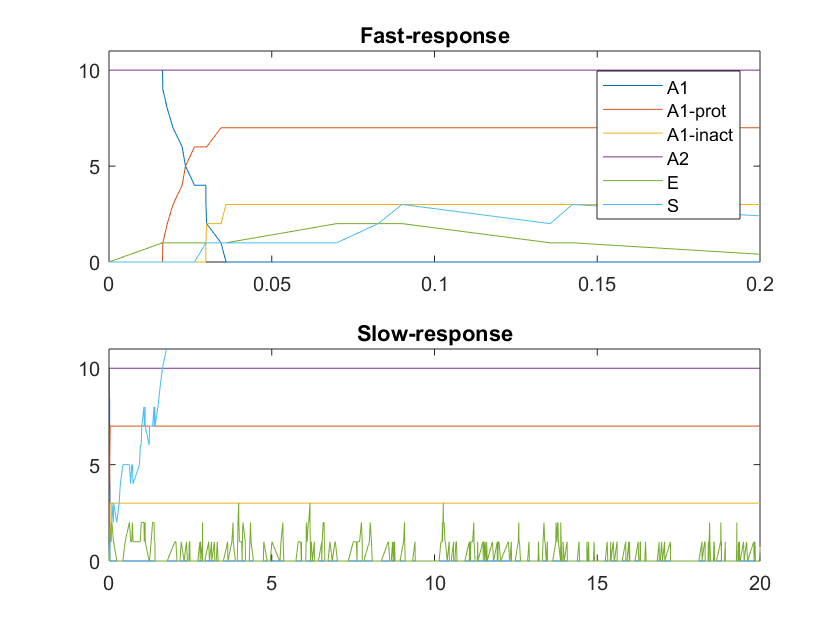
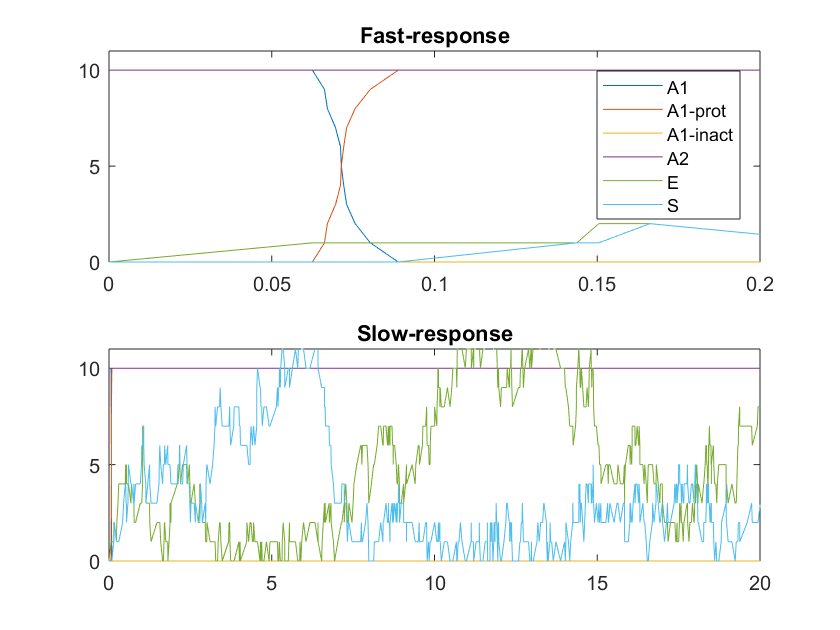
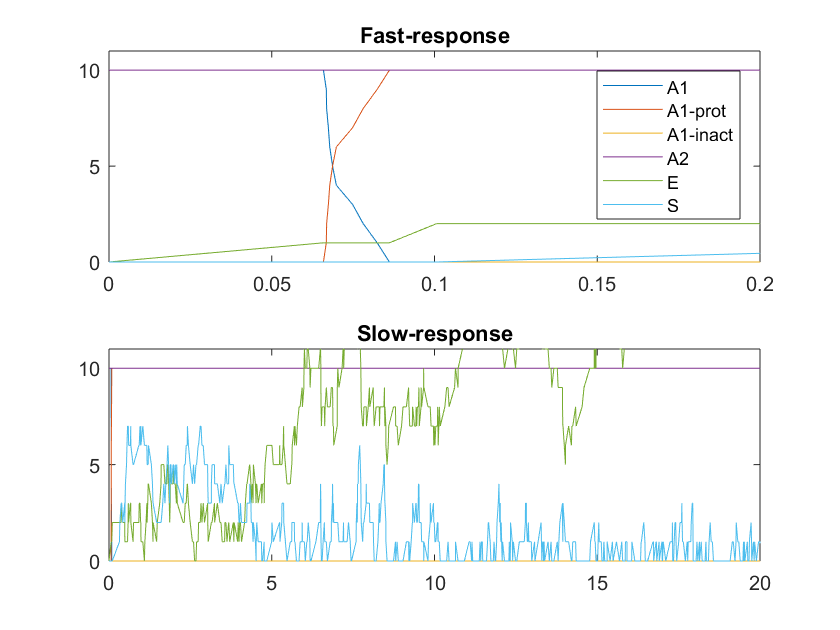


Figure: Plots of four independent runs of the exact stochastic solver given the same initial conditions. The top panel of each plot shows the fast-response (t = 0 to 0.2) and the bottom panel shows the slow-response (t = 0 to 20).

1. Deterministic solution

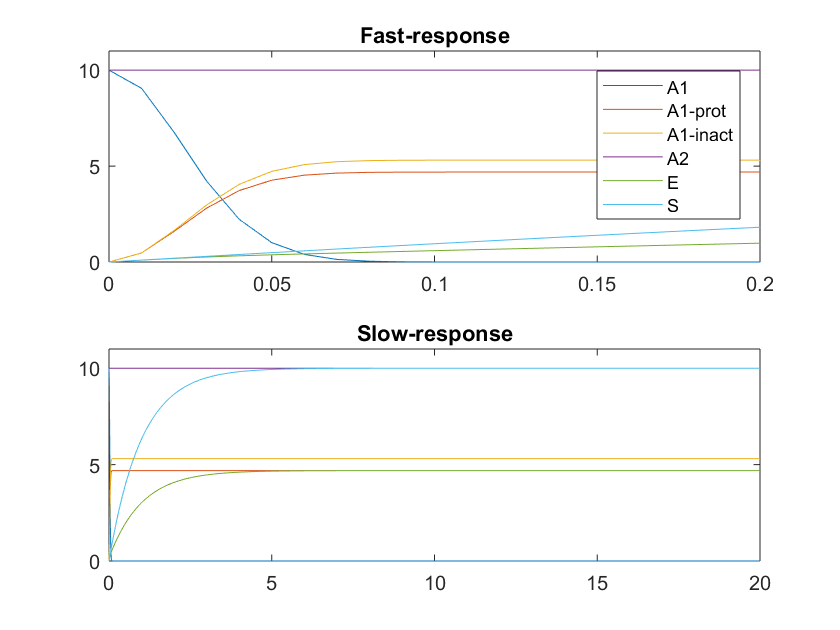


Figure: Shows the converging deterministic solution to the same system as above.

1. Relation to conclusions of Artyomov paper

In the Artymov paper, they demonstrate that introducing stochastic processes in a system can create behaviors that diverge to extremes, while the deterministic solution provides a solution that is in between the extremes. We can see that in the stochastic model, the concentrations of E and S wildly vary over time, with S approaching a concentration of 0 or very high over time, and the concentrations of A1\_prot, A1\_inact also vary in behavior, either 0 or 10. The deterministic solution show that the steady state behavior has a system concentration of A1\_prot and A1\_inact is between 0 and 10, much like how the Artymov paper shows that their deterministic solution is between the extreme values.

# Question 2: Comparing Efficiency and Accuracy of Solvers

1. Efficiency of tauleap (with relative tolerances of: 0.1, 0.01, 0.001) and exact stochastic solver in relation to initial values of A1, A2

\*note – simulations were run for a time span between [0,1]

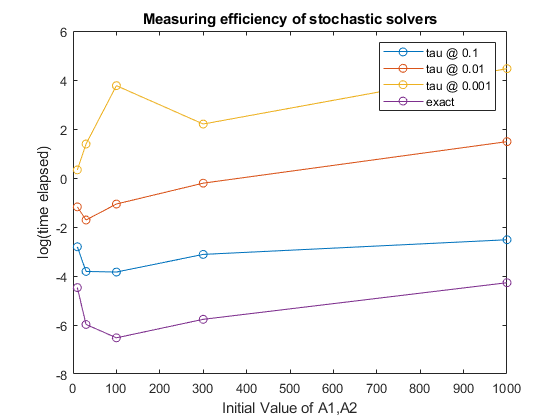


Figure: Plot of solver efficiency based on initial values of A1, A2. Solver time is plotted in a log scale. The exact solver was the most efficient, and the tau step solvers with the lower tolerances were more efficient. It seems like the exact solver and tau 0.1 solver may have equal efficiencies if we tested for larger initial values.

1. Accuracy of each solver – not the 0.001 tau solver because it took too long

Each solver was given up to 5 minutes to run as many times as it could, and the final value of A1\_prot was recorded. I initially noticed that all of the tau solvers would encounter NaN errors, which occurs when the E or S concentration jumps from 1 to -1. Since substrates cannot have a concentration of -1, and this was not something I could figure out how to prevent from happening, I saved the last instance of values that were before E or S turned into -1 and used that as my final output.

Since the exact solver and tau0.1 solver produced similar results, and the exact solver was more efficient, I would recommend the exact solver as being sufficiently accurate and efficient given this 5 minute limitation in order to draw a meaningful conclusion about the behavior of the model. However, it does seem like with higher initial condition values, the tau0.1 solver may be more efficient.

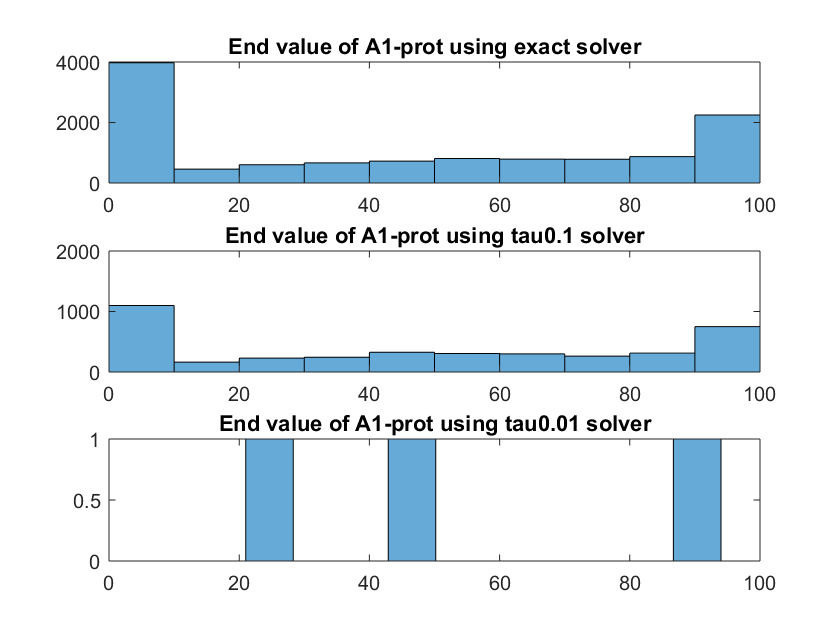


Figure: Histogram of the final value of A1-prot using the exact, tau0.1, and tau0.01 solver. The exact and tau0.1 solver had very similar end results, but during the 5 minute run, the tau0.01 solver only produced three values.

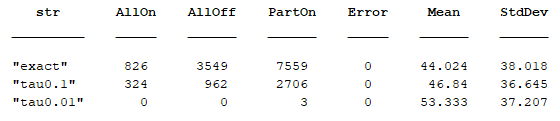


Table: Statistics for the 5 minute run for each solver. The mean and standard deviations were similar across different solvers. There were no errors (mentioned above). Most of the solutions for the exact and tau0.1 solvers produced a final output of A1-prot being partially on, but the top two modes were the two extremes, either 0 or 10.

# Question 3: Evaluating Initial Conditions vs. System Behavior

I used a combination of the exact solver for smaller initial conditions and the tau0.1 solver for larger initial conditions.

1. Histogram of A1\_prot for each initial condition: A1 = A2 = [10, 100, 1000, 10000, 100000]

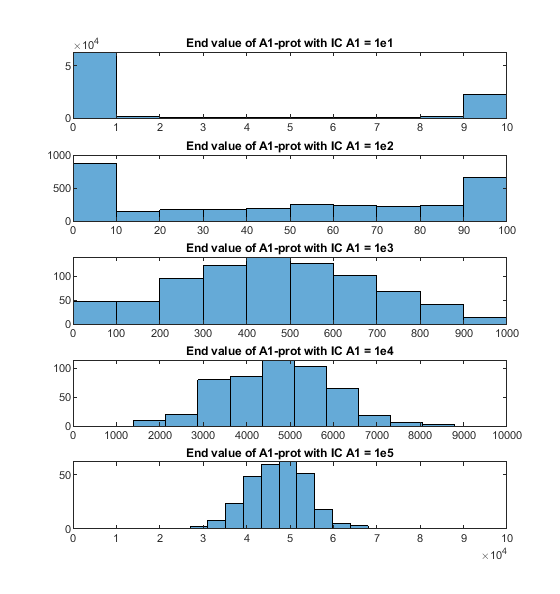


Figure: Histogram of end A1\_prot values with varying initial conditions. We can observe that when you increase the initial condition concentration, the system behaves less stochastically and more like what you expect the deterministic solution to resemble.

1. Average, Stdev of fraction protected (A1\_prot/A1(0)) against system size
2. Plot fraction simulations that end in “all on” (A1\_prot = A1(0)) against system size

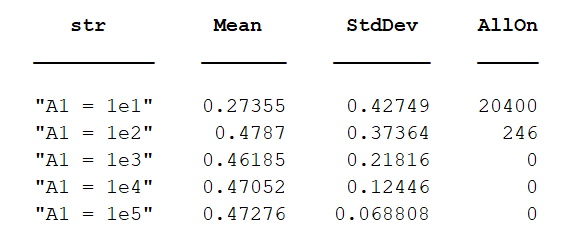


Table: Shows the fractional mean, fractional standard deviation, and number of “all-on” cases across the different initial conditions.

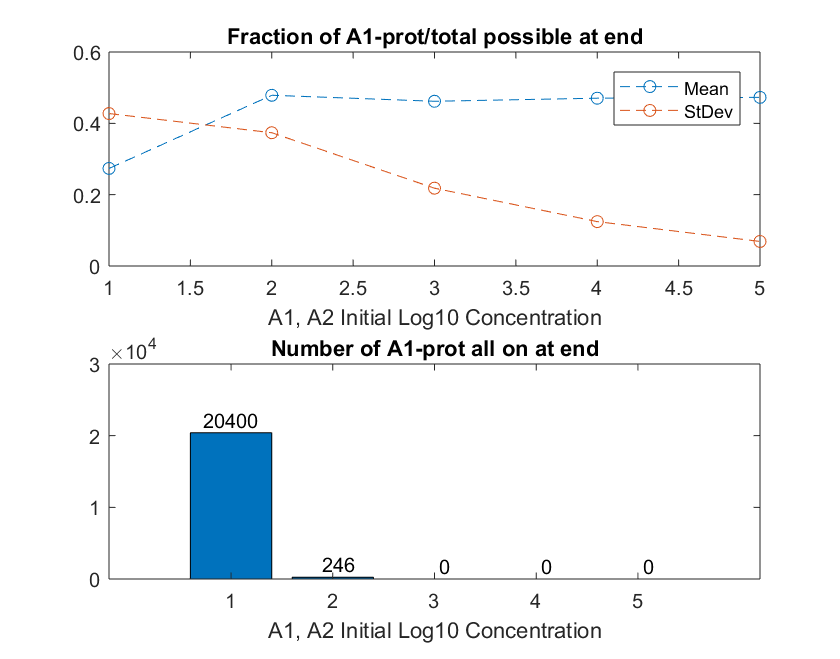


Figure: Displays the fractional mean, fractional standard deviation, and number of “all-on” cases across the different initial conditions/concentrations on a log10 scale. The mean is about the same across cases other than the first case, and the standard deviation decreases as the starting concentration increases. The number of all-on cases also decreases until there are no all-on cases.

1. How fraction protected changes with system size

From the initial histogram, we observe that as the system size increases (concentration of initial A1, A2 increases), the behavior of the system shifts from a “on-off extremes” behavior, where most of the A1\_prot values are at extreme concentrations, to a system that demonstrates more deterministic properties with an approximate binomial curve. With increasing system sizes, the stochastic effects and “switched on” or “switched off” effects are eliminated, and we are left with a fairy deterministic solution, with the mean fractional outcome reaching a constant value and the standard deviation of that value decreasing.

# Appendix: MATLAB Code Documentation

Files:

**MATLAB files**

* “lab5.m” – main code to run
* “DSDEexact.m” – Exact stochastic solver
* “DSDEtauleap.m” – Tau leap stochastic solver
* “Tcellode.m” – ode function for the t-cell, outputs next value
* “TCellRXN.m” – converts chemical reaction to SPK form

**Data files**

* “params.mat” – Parameters values for the system
* “t\_eff.mat” – Timing of different solvers (Q2a)
* “A1\_p\_end.mat” – Data of A1\_prot values for accuracy part (Q2b)
* “A1\_p\_end2.mat” – Data of A1\_prot values for analysis (Q3a)

Comments:

* To generate the figures, simply run the sections that are bolded below. The data has already been saved so there is no need to rerun the entire code unless for confirmation purposes (Q2,Q3 are known to take a long time)

## Code Outline:

set up system 1

*This part sets up the parameters needed for the system*

Q1: Solve Artymov paper model 1

*This part solves Q1 and* ***outputs figures for Q1***

Q2: Comparing solver efficiency, accuracy 2

*This part solves for Q2a*

**Figures for Q2a** 3

Accuracy 3

*This part solves for Q2b*

post processing - **Figures for Q2b** 5

Q3: Run @ different ICs to see how system changes 6

*This part solves for Q3*

post processing - **Figures for Q3** 7