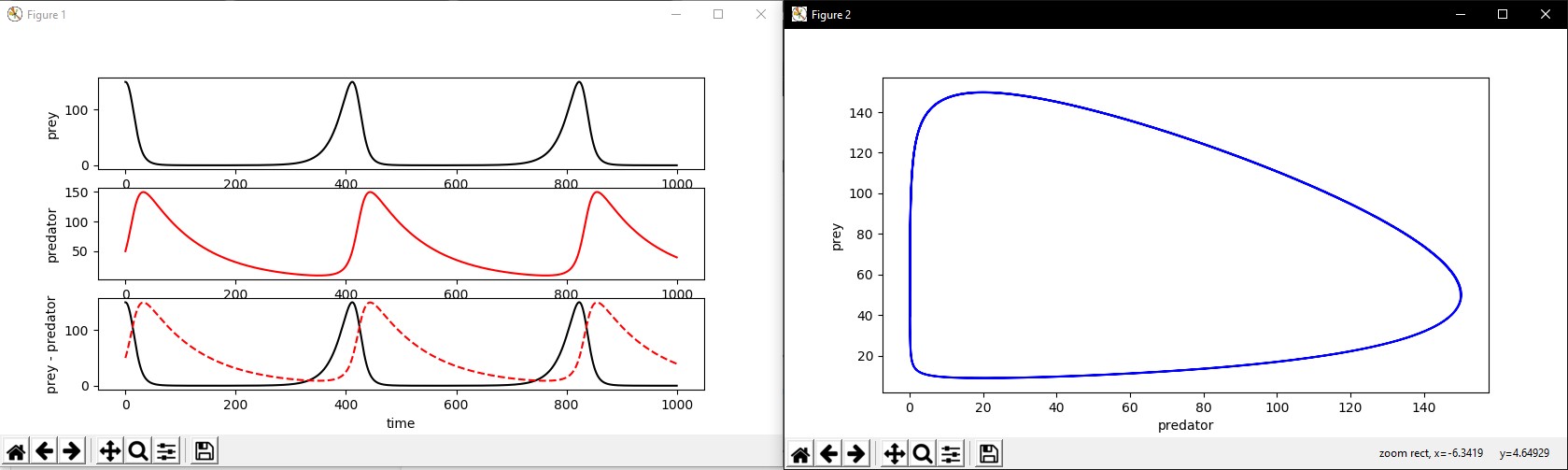
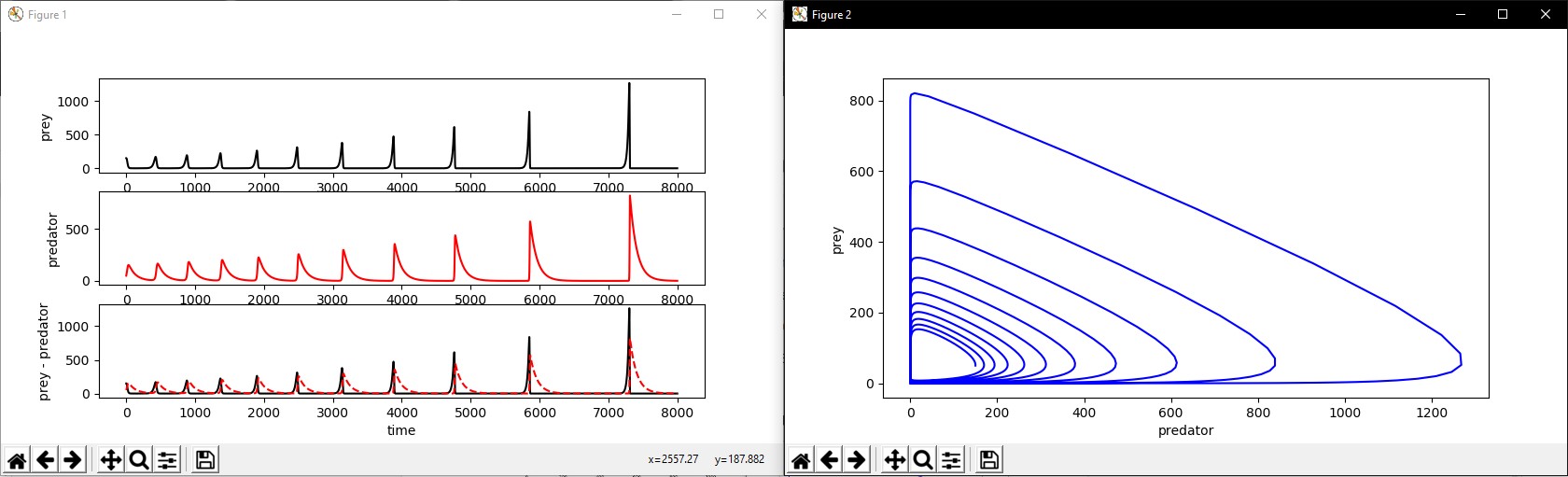
**System Simulations Project 3  
Daniel Walpole**

This project has us looking at the predator/prey simulation and having to try to understand what is going on with the simulation. In the first part of this simulation there are just two variables: the predators, and the prey. They live in a state of growth with each other. When the prey starts to run low on numbers the number of predators starts to decline at an almost equal pace. After this the prey has some breathing room again allowing them to grow more than the last time. When running the simulation with the values set to: Prey Birth: 0.05, Predator Birth: 0.0005, Prey Death: 0.001, Predator Death: 0.01, Initial Prey Size: 150, Initial Predator Size: 50, a step rate of 0.005 running for 1000 steps.



If we adjust the time to 8000 steps with a rate of 1 mainly for time sake, we can see that the population of both grow as the simulation runs with these variables.



The shape starts to change into a more triangular shape as the simulation runs. With each cycle of the prey dying and then the predators dying we can see that they each grow in turn. Also, with each cycle the drastic decrease in population each has occurs faster. When we look at the equation for each, we can see why this must be.

Prey: self.dq[0] = (self.preyInc \* self.q[0]) - (self.preyDec \* self.q[0] \* self.q[1])

Prey: dG/dt = (PreyBirth \* G(t)) - (PreyDeath \* G(t) \* I(t))

Predator: self.dq[1] = (self.predInc \* self.q[0] \* self.q[1]) - (self.predDec \* self.q[1])

Predator: dI/dt = (PredatorBirth \* I(t) \* G(t)) - (PredatorDeath \* I(t))

The predator values can only come into play if the prey values are great enough which then kill the prey and then kill the predators due to starvation.

Now when we introduce the pesticide value, we need to re-evaluate each function:

Prey: self.dq[0] = (self.preyInc \* self.q[0]) - (self.preyDec \* self.q[0] \* self.q[1]) -(0.1 \* self.dq[2])

Prey: dG/dt = (PreyBirth \* G(t)) - (PreyDeath \* G(t) \* I(t)) - (PesticidePreyKillRate \* P(t)’)

Predator: self.dq[1] = (self.predInc \* self.q[0] \* self.q[1]) - (self.predDec \* self.q[1]) - (0.3 \* self.dq[2])

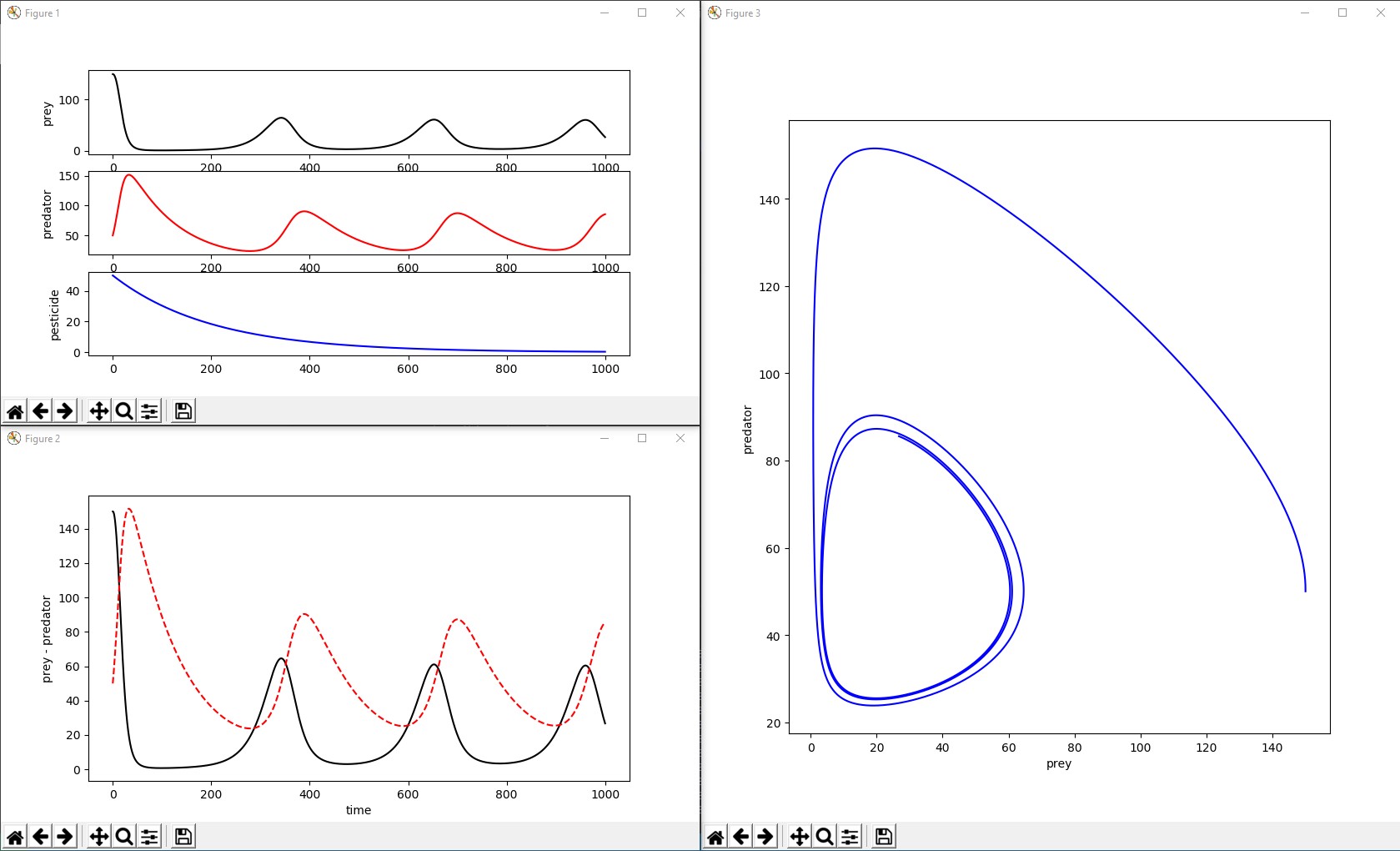
Predator: dI/dt = (PredatorBirth \* I(t) \* G(t)) - (PredatorDeath \* I(t)) - (PesticidePredatorKillRate \* P(t)’)

Pesticide: self.dq[2] = - (.005 \* self.q[2])

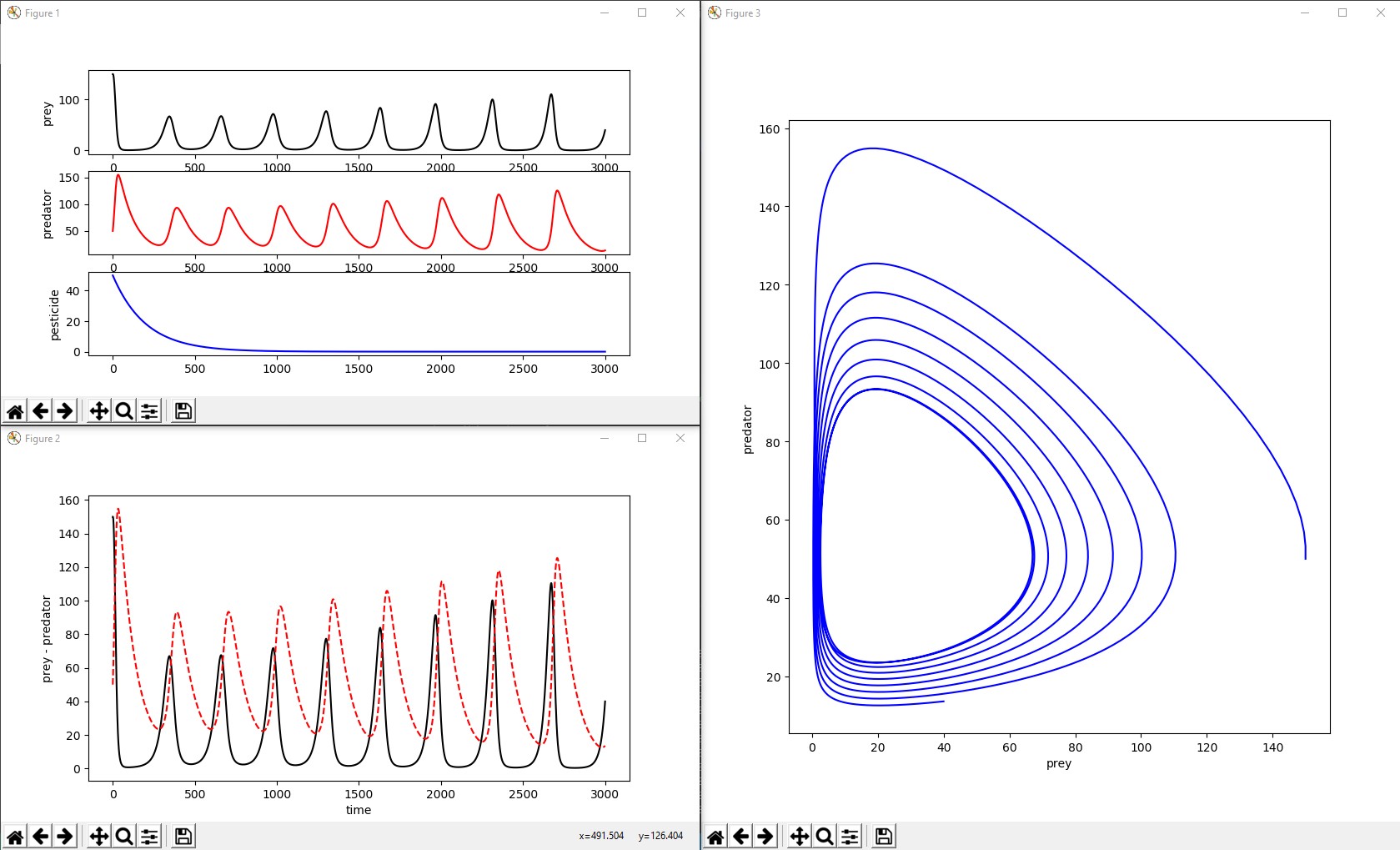
Pesticide: dP/dt = P(t) - (PestDecay \* P(t))

Note that the Prey and Predator functions now take the derivative of the pesticide function multiplied by a kill rate of the pesticide. We take the derivative of the pesticide function as that is the difference of how much pesticide has changed in an amount of time.

Now that we need to see how the pesticide works with our initial settings from before and with the Initial Pesticide: 100, Pesticide Kill Rate: 20%, a step rate of 0.005 running for 1000 steps.



So, unlike the initial testing we can see that the prey and predator both have gotten much smaller, due to the pesticide killing off a large portion of them. As the values are decreasing you can estimate that they are going to an extinction. However, this is not the case as with time the pesticide cannot hold the prey and predators down for long as they will inevitably bounce back. So, with a step rate of 1 for 3000 steps we get the following graph.



From the other simulation we can see that after an amount of time the values start to increase. So, with that said I think the pesticide did its job but should be reapplied if wanting to keep the prey and pest down to controlled levels.