Dynamic Models in Biology
Final Exam
Jonathan Levine
Fall 2023

Problem 1

The model:

$$\frac{d}{dt}m(t)=\frac{v_s}{1+(p_{\mathrm{N}}(t)/K_{\mathrm{I}})^n}-\frac{v_m m(t)}{K_{m1}+m(t)}$$

$$\frac{d}{dt}p_0(t)=k_s m(t)-\frac{V_1 p_0(t)}{K_1+p_0(t)}+\frac{V_2 p_1(t)}{K_2+p_1(t)}$$

$$\frac{d}{dt}p_1(t)=\frac{V_1 p_0(t)}{K_1+p_0(t)}-\frac{V_2 p_1(t)}{K_2+p_1(t)}-\frac{V_3 p_1(t)}{K_3+p_1(t)}+\frac{V_4 p_2(t)}{K_4+p_2(t)}$$

$$\frac{d}{dt}p_2(t)=\frac{V_3 p_1(t)}{K_3+p_1(t)}-\frac{V_4 p_2(t)}{K_4+p_2(t)}-k_1 p_2(t)+k_2 p_{\mathrm{N}}(t)-\frac{v_{\mathrm{d}} p_2(t)}{K_{\mathrm{d}}+p_2(t)}$$

$$\frac{d}{dt}p_{\mathrm{N}}(t)=k_1 p_2(t)-k_2 p_{\mathrm{N}}(t).$$

(a)

The state variables:
m: concentration of *per* mRNA
p_0: concentration of PER protein
p_1: concentration of phosphorylated PER protein (PER-P)
p_2: concentration of twice phosphorylated PER protein (PER-P2) in the cytosol
p_N: concentration of PER protein in the nucleus

The first term in the m differential represents the transcription and export of the *per* gene into the PER protein. In the absence of p_N, the transcription and export process is constant with rate v_s. In the presence of p_N, transcription is repressed by p_N via a Hill function with Hill coefficient n. The hill function is a simple way to model the binding kinetics of the repressor, where K_1

represents the binding strength (ligand concentration producing half occupation) and n represents the steepness of the dynamics.
The second term in the m differential represents a Michaelis-Menten type degradation of the mRNA with v_m and K_{m1} being the maximal rate and the reaction constant respectively. There is no negative term for translation since mRNA is not "used up" when translated into protein.

The p_0 differential has a positive term for translation from the *per* mRNA (linear term). The next two terms are MM reaction terms for the phosphorylation reaction (negative) from p_0 to p_1 and the backwards dephosphorylation reaction from p_1 to p_0.

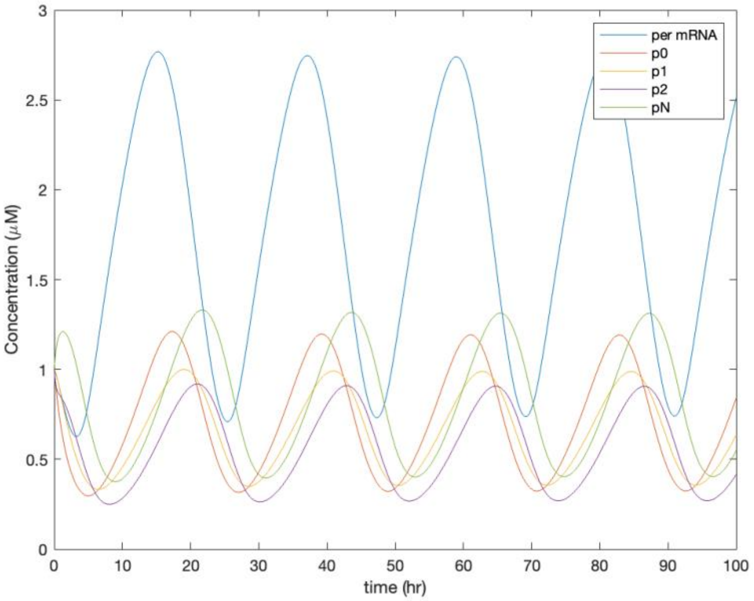
The p_1 differential has the same two terms for reversible p_0/p_1 phosphorylation reaction, but with opposite signs. Similarly, the next two terms are MM reaction terms for the phosphorylation reaction (negative) from p_1 to p_2 and the backwards dephosphorylation reaction from p_2 to p_1.

The p_2 differential has the same two terms for reversible p_2/p_1 phosphorylation reaction, but with opposite signs. p_2 also undergoes degradation with a MM term governed by v_d and K_d. There are two more terms related to the import and export of p_2 across the nuclear membrane. This is also mathematically represented as a "reaction" of sorts, where p_2 is reversibly converted to p_N. The positive term represents nucleus → cytosol and is linear with rate k_2. The negative term represents cytosol → nucleus and is linear with rate k_1.

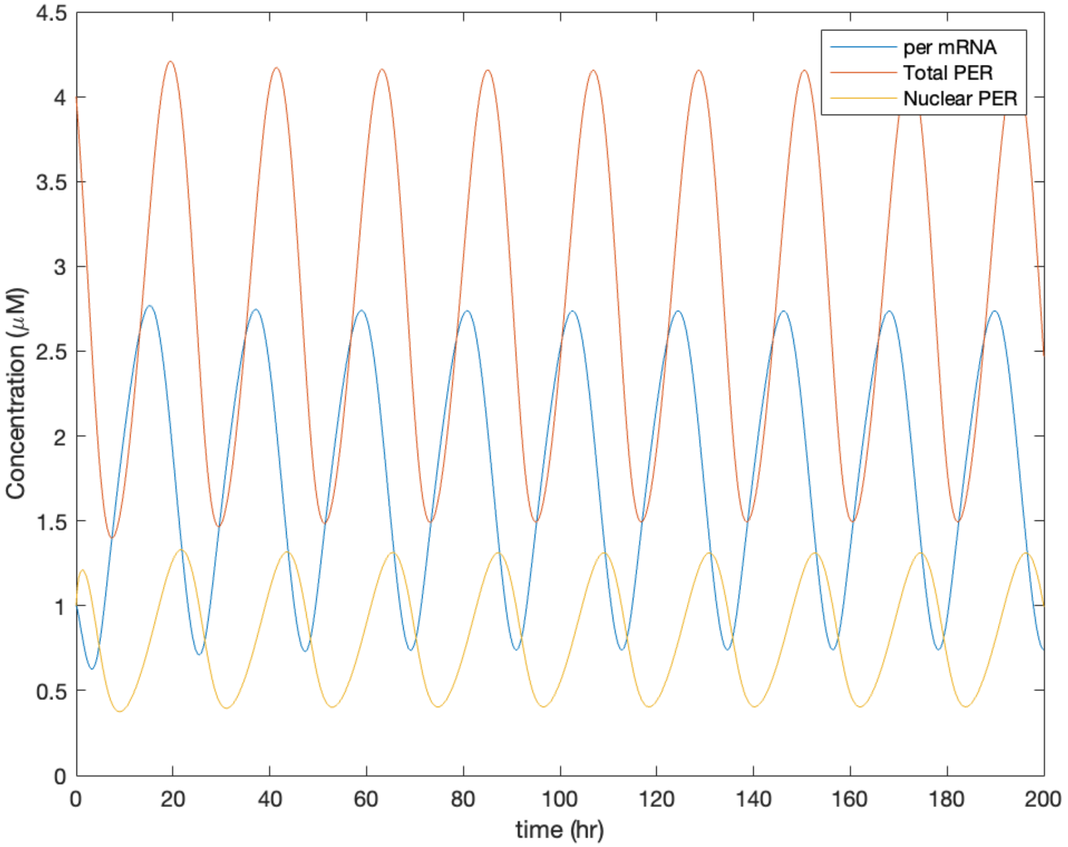
The p_N differential has the same two nuclear membrane transport terms, but with opposite signs. There is no degradation term since degradation of PER is assumed to occur only in the cytosol.

(b)

Simulating this model with the baseline parameters, we see stable oscillations in all 5 state variables:

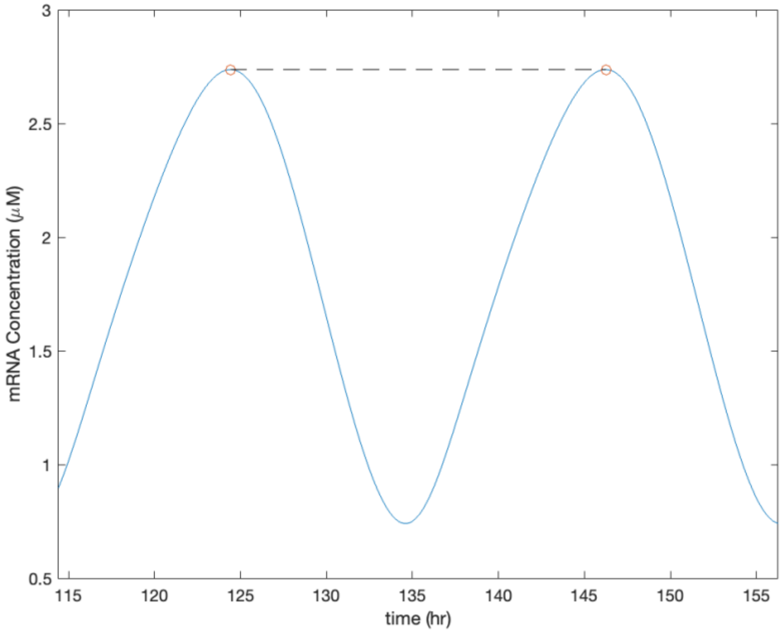


To look at the data in a slightly less cluttered way, we can combine all of the PER protein states together as "Total Per", and then look at the subset that is in the nucleus, along with the mRNA levels (matching Figure 2, left)

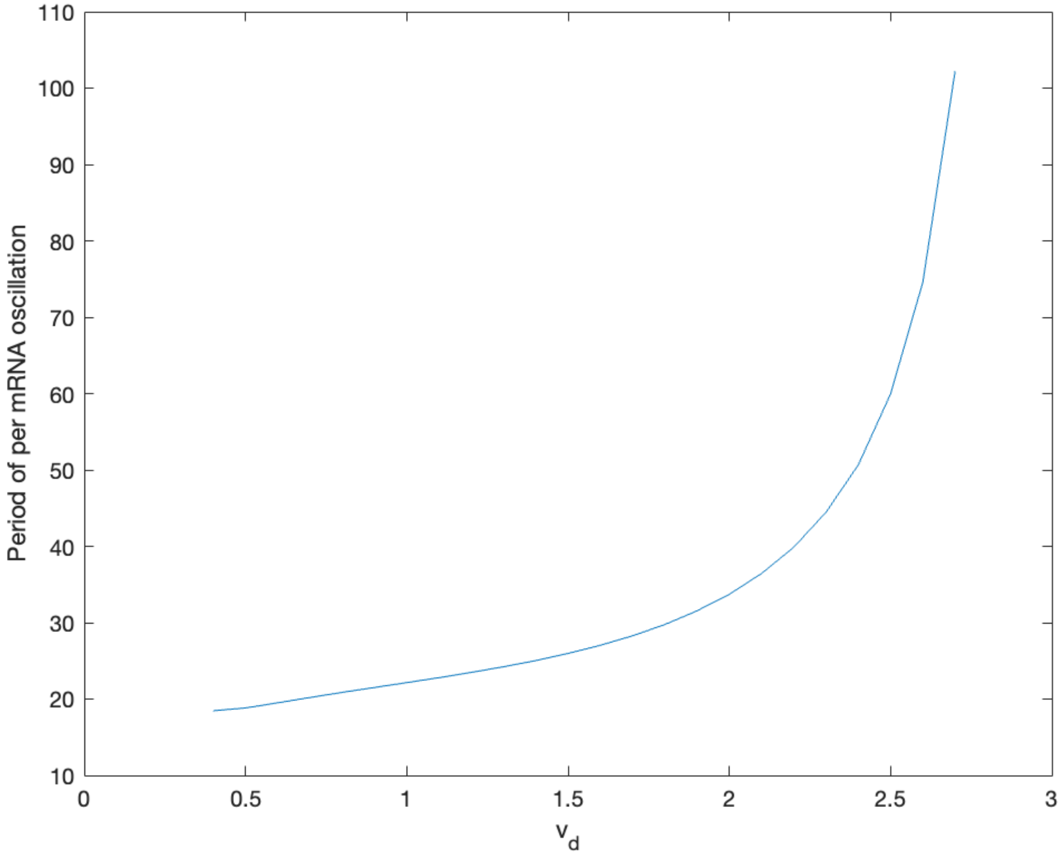


(c)

We can measure the period by looking at the oscillations of one of the state variables, such as *per* mRNA, and looking at the time between peaks:

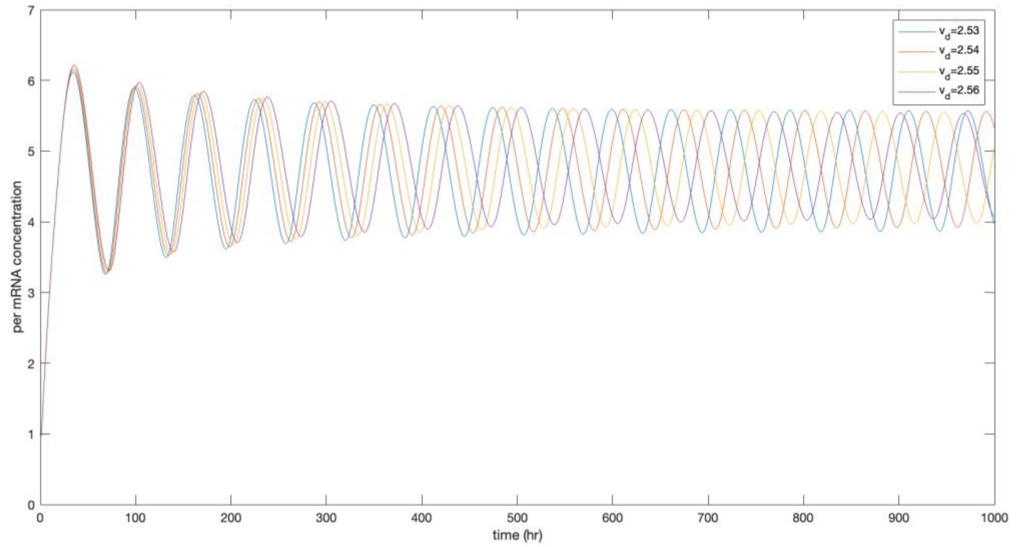


We then look at this period with varying v_d, and it matches Figure 2, right:

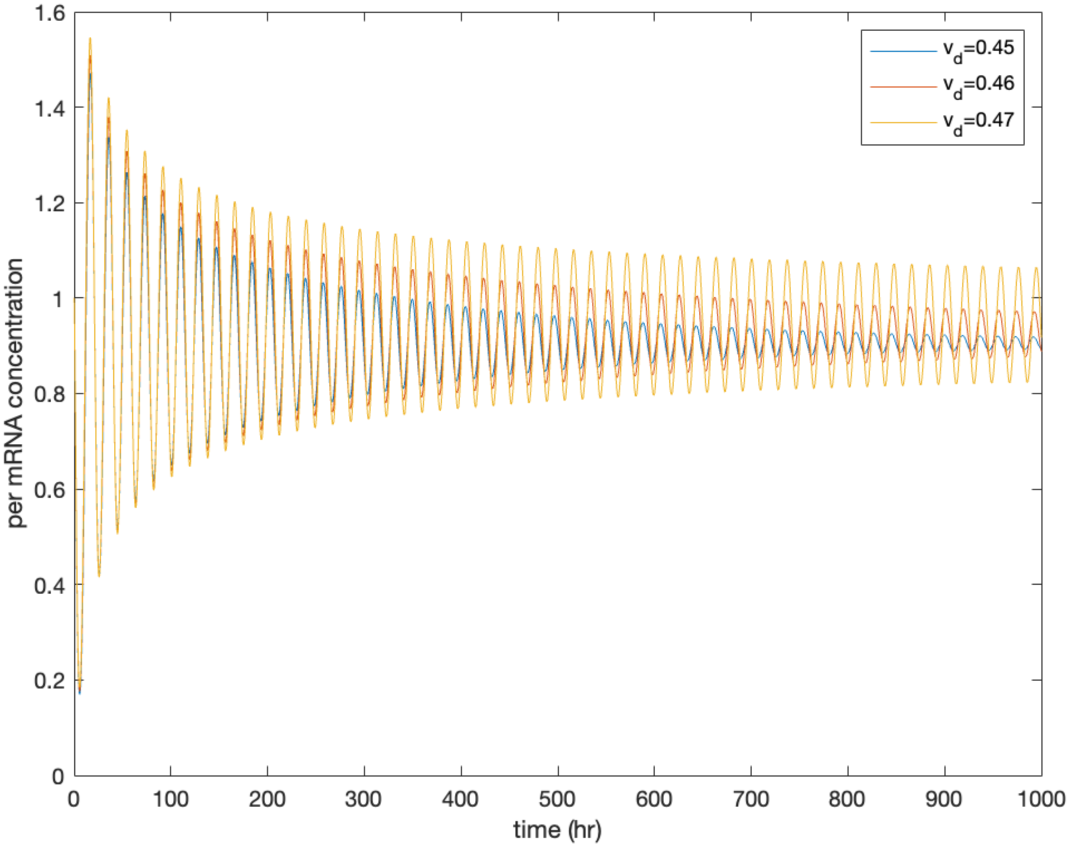


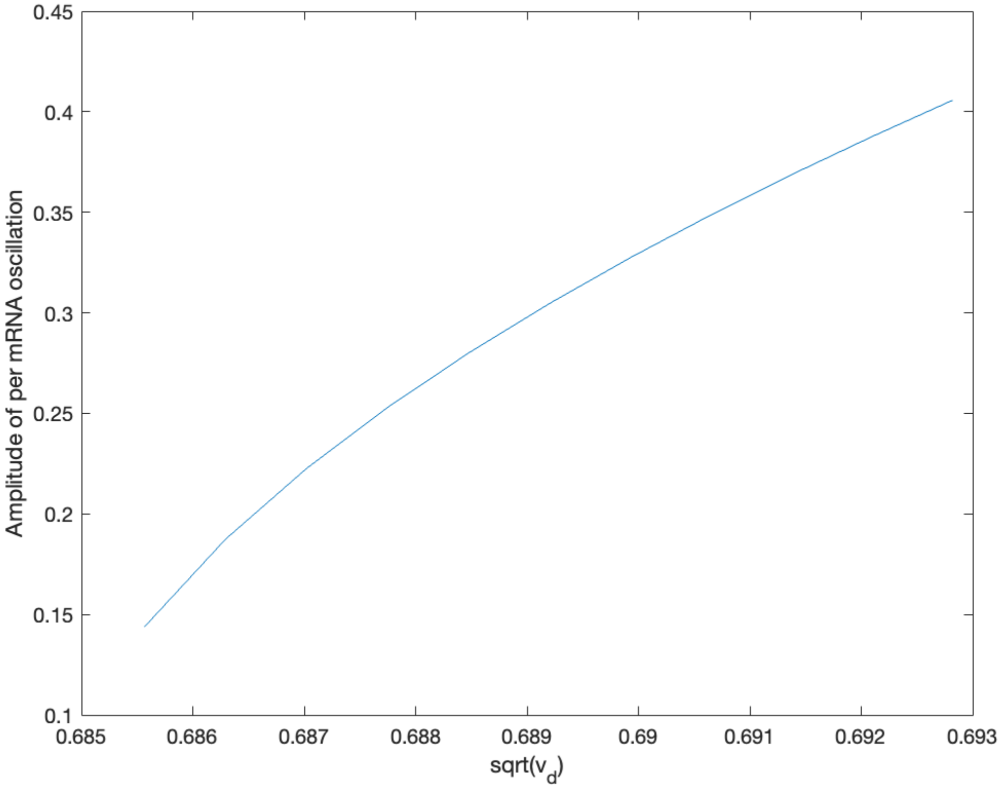
(d)

Around v_d=2.57 µM/hr the system undergoes a subcritical Hopf bifurcation (you can tell because the amplitude of the oscillation does not change much as v_d changes):



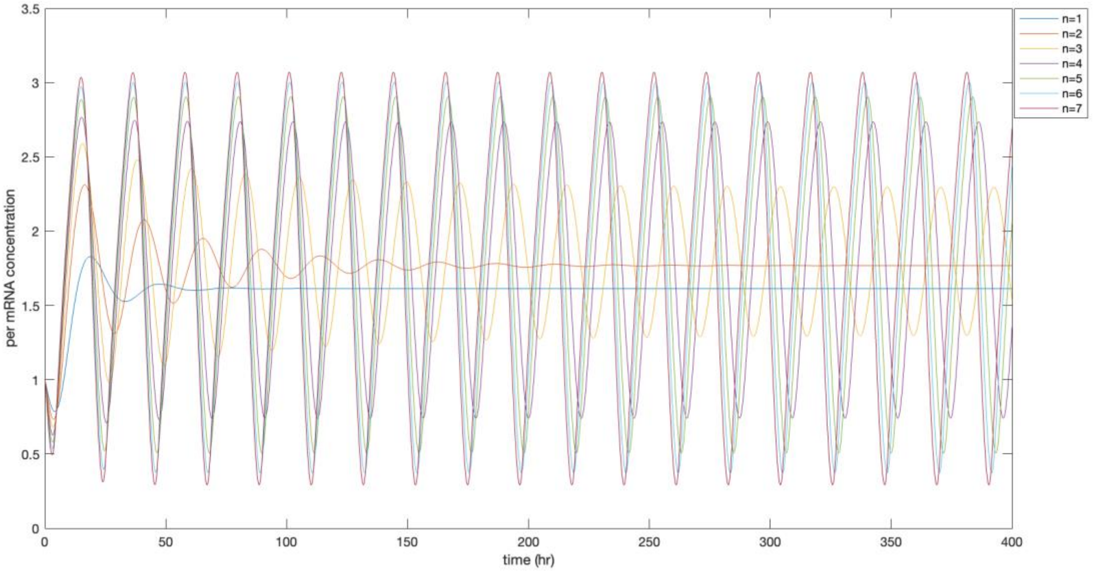
Around v_d=0.45 µM/hr the system undergoes a supercritical Hopf bifurcation. You can tell because the amplitude of the oscillation is small at first, and then grows like the sqrt of the parameter as you get further from the bifurcation point.



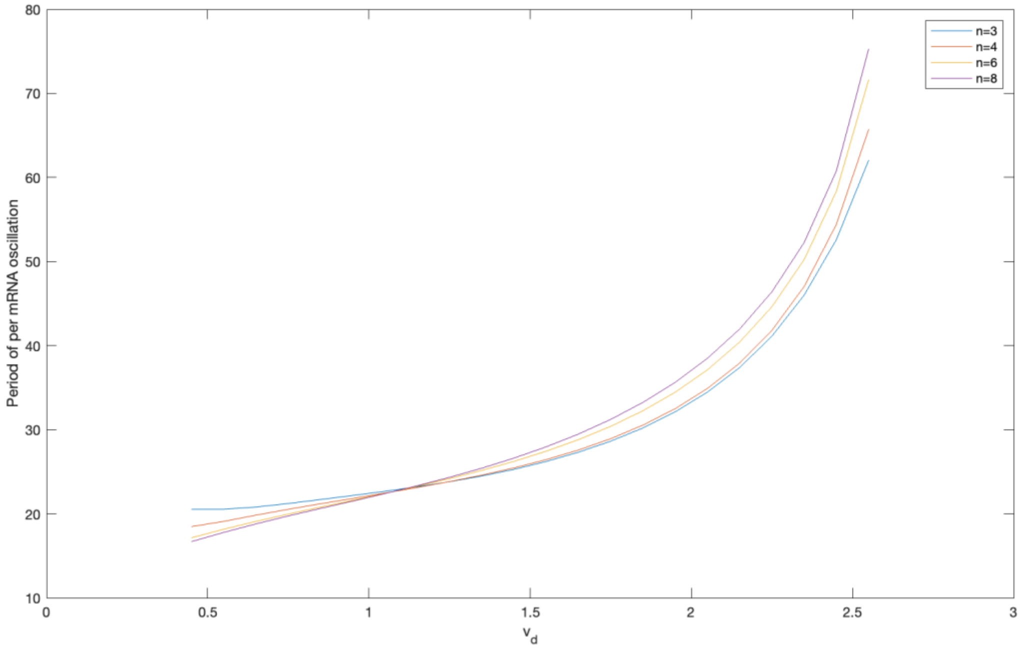


(e)

Varying the n value, we see that the oscillations change amplitude (and disappear for n=1,2) for v_d=0.95



We can then ask how the period vs. v_d curve shifts for the different values of n where oscillations still exist:



You can see that as n increases, the period vs. Vd curve becomes steeper (purple and yellow traces vs. red and blue traces). This is essentially showing that the model's sensitivity to V_d is greater with larger values of n. This is because *n* controls the steepness of the repression of the gene *per* by PER protein, which is the feedback loop that contributes to oscillations. When the exponent is larger, the feedback loop is sharper and more non-linear, which also means that the model is more sensitive to changes in the components leading up to the existence/degradation of the repressor protein (in this case the degradation of PER in the cytosol).

(f)

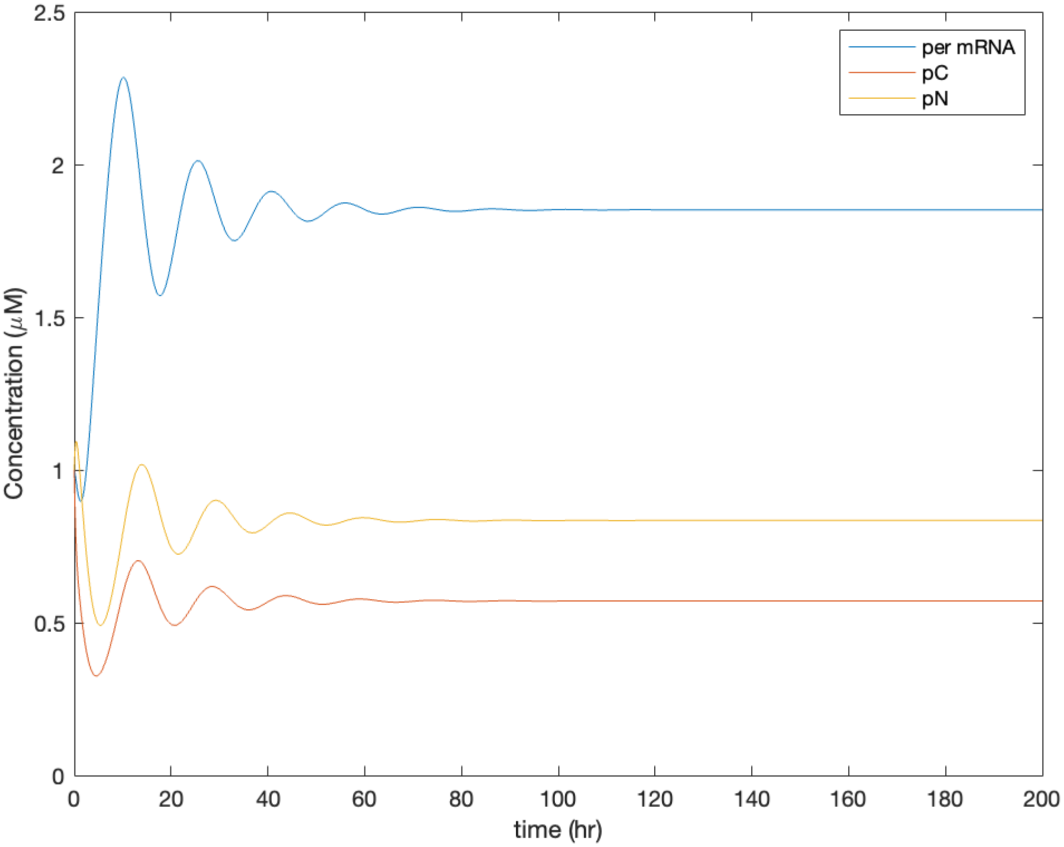
An alternative version of this model might lump all forms of the cytosolic PER protein together into one term (p_C). This would then be:

$$\frac{d}{dt}m = \frac{v_s}{1+\left(\frac{p_N}{K_I}\right)^n} - \frac{v_m m}{K_{m1}+m}$$

$$\frac{d}{dt}p_C = k_s m - \frac{v_d p_c}{K_d + p_c} - k_1 p_c + \ k_2 p_N$$

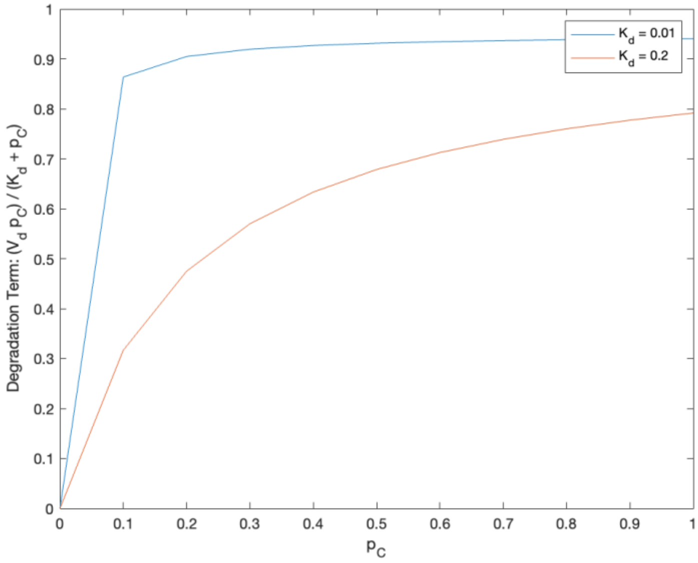
$$\frac{d}{dt}p_N = k_1 p_c - \ k_2 p_N$$

With the baseline parameters, the system comes to a stable fixed point without stable oscillations:

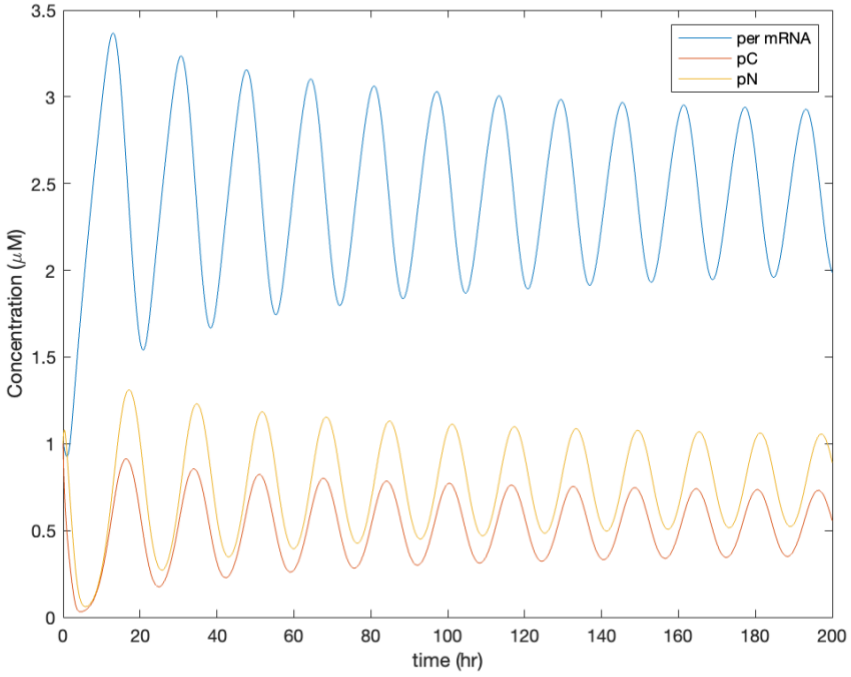


Because the system is no longer explicitly modelling the phosphorylation reactions, the turnover from mRNA → protein → repression happens too quickly to induce oscillations. The extra intermediate steps effectively delayed the total feedback loop from mRNA to protein to repression of transcription. Therefore, if we want to induce oscillations in the model without explicitly modelling those extra steps, we need to slow done the feedback loop enough to get the effective delay back.

If we decrease K_d enough, we can approximate the degradation as closer to a pseudo-zero order reaction instead, which will slow done the feedback loop since it will take longer for enough PER to accumulate and trigger the repression. For example, for $K_d = 0.01$ this degradation term is closer to zero-order for most concentrations of P_c and thus will "slow down" the reaction by increasing the degradation rate of PER protein.



This then induces oscillations in the model:



(g)

An alternative way to "slow down" the feedback loop is by explicitly modelling a time delay in the model, and using a delay differential equation instead:

The model would then be:

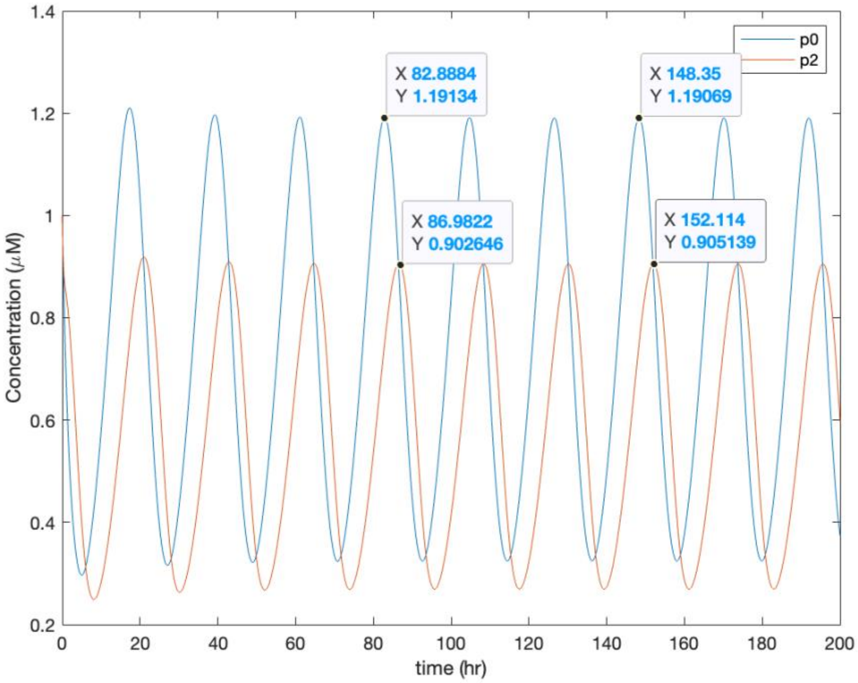
$$\frac{d}{dt}m(t) = \frac{v_s}{1+\left(\frac{p_N(t)}{K_I}\right)^n} - \frac{v_m m(t)}{K_{m1}+m(t)}$$

$$\frac{d}{dt}p_C(t) = k_s m - \frac{v_d p_c(t)}{K_d + p_c(t)} - k_1 p_c(t-\tau) + \ k_2 p_N(t)$$

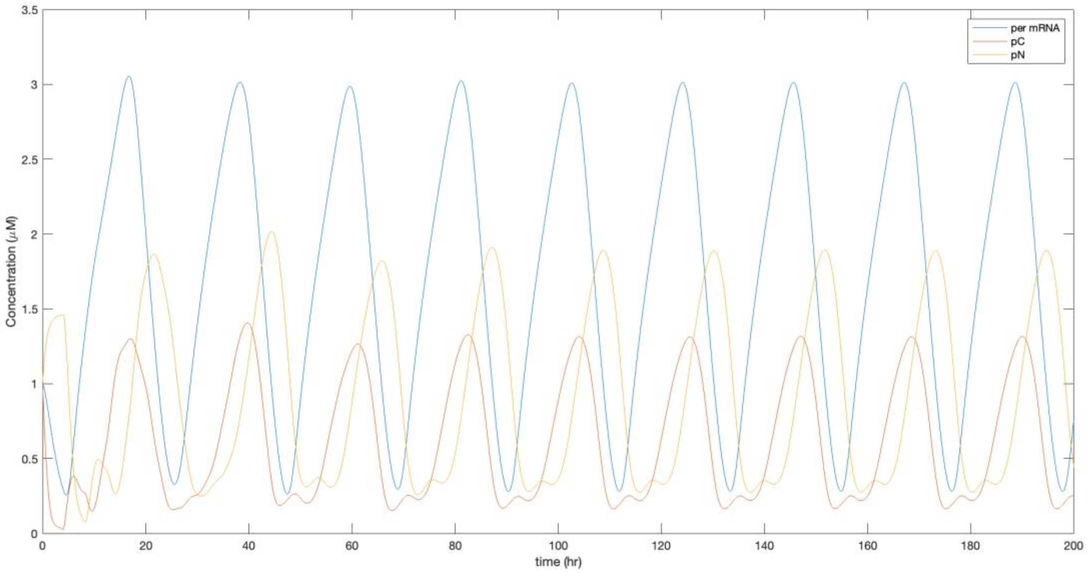
$$\frac{d}{dt}p_N = k_1 p_c(t-\tau) - \ k_2 p_N(t)$$

The difference here is that there is an assumed delay between the translation of PER and the import of PER into the nucleus. The nuclear import term is based on the quantities of cytosolic PER from τ hours ago rather than the current quantity. This should effectively model the intermediate reactions that needed to happen to go from translated PER into the double phosphorylated form that is able to be imported in the nucleus.

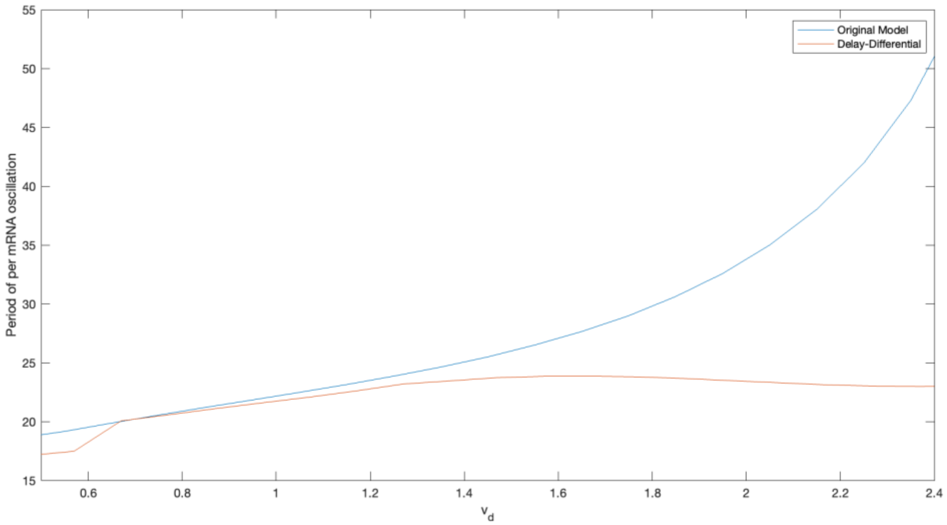
The time delay τ, represents the time it takes for PER to predominantly be in the p0 form until it is predominantly in the p2 form. Therefore, this delay should be set to the time delay between the p0 peak and the p2 peak in the original model, which is around 4 hours.



Setting $\tau = 4$ and simulating the delay-differential, we see oscillations return, though they look a little bit weirder in the waveform:

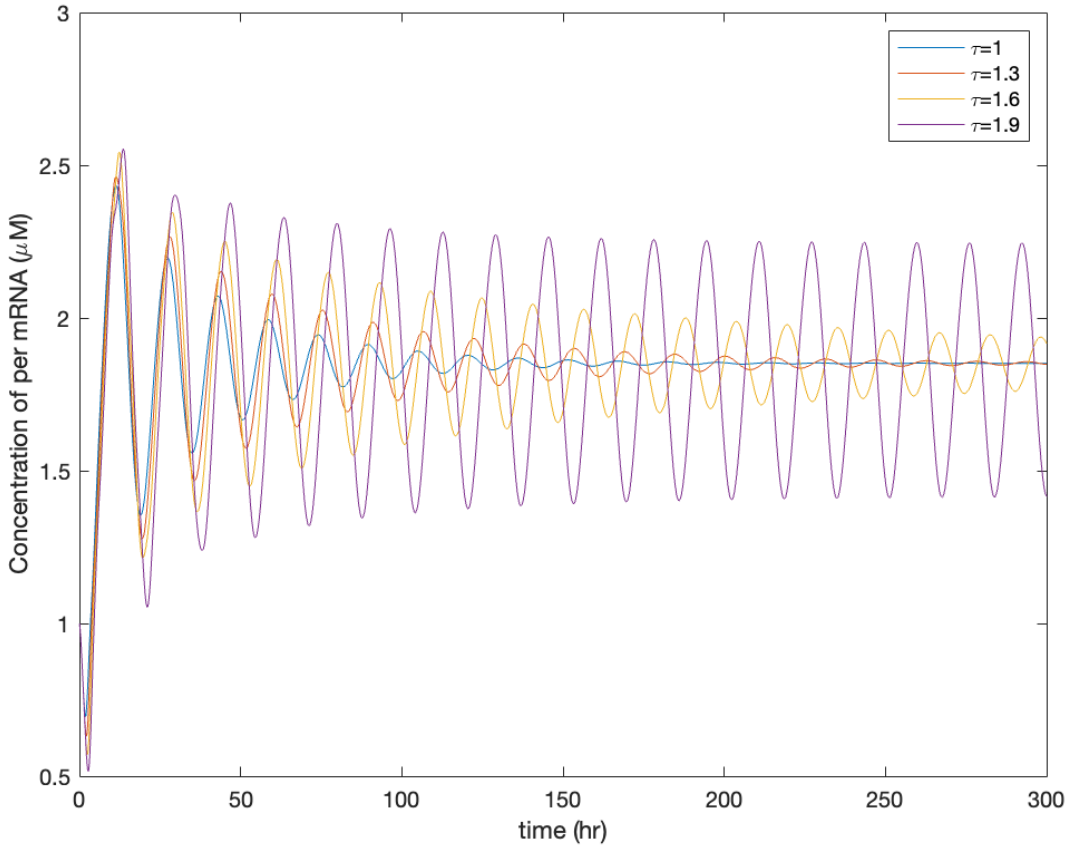


In this model however, the oscillations are less sensitive to the degradation term, since the delay is the main driver of oscillations. This is evident in looking at the period vs. V_d curve from before and comparing it to the same curve measured for the delay-differential:

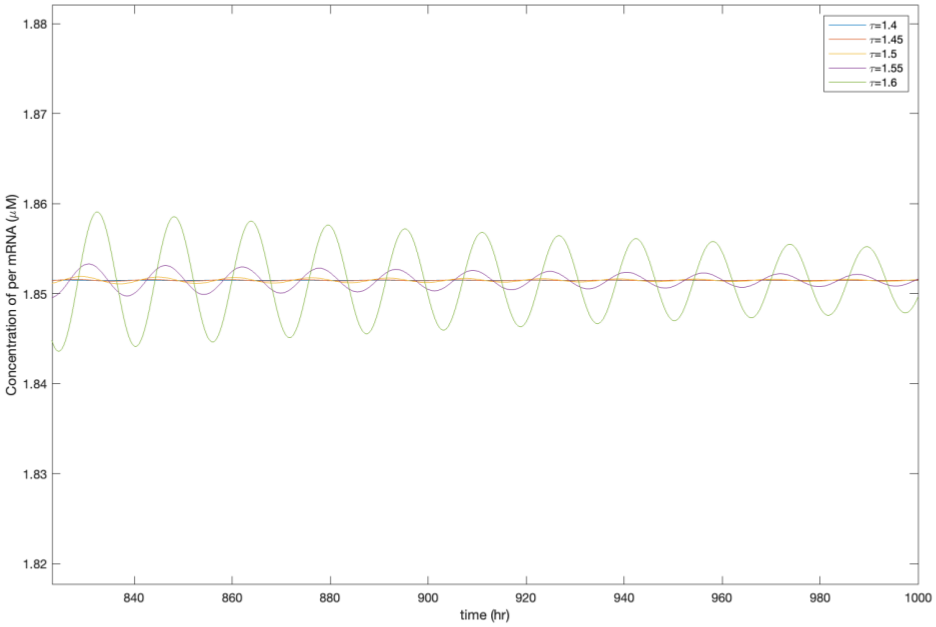


As you can see, the period of the oscillation is largely insensitive to the degradation rate since the oscillations are explicitly modelled as time delayed and the degradation is no longer critical to slow down or speed up the feedback loop.

If you reduce the time delay, the oscillations get smaller in amplitude, and eventually disappear altogether:



Zooming in on this range (and extending the simulation to ensure the oscillations are actually non-transient), we see that this minimum delay threshold is around $\tau = 1.5$:



(h)

If the model were to incorporate spatial diffusion without a time delay, I don't think that would be "slow" enough to induce oscillations. Even assuming a relatively large protein with a slow diffusion constant, the small distance from ribosome to nucleus means that the diffusion process would, on average, be on a fast time scale relative the delays needed to induce oscillations. Specifically, the diffusion constant is likely on the order of magnitude of $10\ \mu m^2/s$. The time to traverse a distance R with diffusion constant D is proportional to $\frac{R^2}{D}$, which in this case (distance is on the order of 1-10 μm) would mean that the time to diffuse would be an order of magnitude around 10 seconds. This is far less than the minimum delay found in part g above which was on the order of around 1.5 hours.

Problem 2

One of the clearest differences between a diffusion process and a Turing instability is whether the pattern repeats. If my friend was able to take measurements of this system at longer length scales, it would become obvious if the pattern repeated like a sinusoid or simply diffused close to zero like a pure diffusion process.

My friend could also wait for a longer period of time and re-measure the distribution. If the distribution is flatter and more uniform, that would support diffusion which eventually should lead to a fully uniform concentration distribution.

If my friend was only able to use this single experiment, I would advise trying to measure the slop very close to the supposed "point" sources. For a pure diffusion process, the slope at the point source should be negative and far from zero. This is because a diffusion process would give rise to the sum of two exponentials, whose derivative is also the sum of two exponentials (although one may have a negative sign in front). This indicates a specific hypothesis for what the data should look like: the derivative of the curve plotted there should be non-zero around the sources, and specifically negative around the first point source and positive around the second point source. For a Turing instability, the slopes should be close to zero, since that is more like a sinusoidal wave (in this case, a cosine or sum of cosines) whose derivative is zero at the crests.

It does appear from the data that the curve matches a sinusoid more than two exponentials, so without the ability to explicitly test this hypothesis with measurements along a longer length scale, I would be inclined to side with friend #2 based on the shape of the curve.