Week 2 Notes (5/31/18)

During our meeting:

- Any questions about chapter 1 of Wiggins?
- Review worksheet and answers
- Review/walkthrough code and figures
- Discuss nondimensionalization of gLV equations
- Discuss Lyapunov functions for dynamical systems

For next week:

- Read chapter 2 of Wiggins
- Show that 2D gLV equations can be nondimensionalized so that

$$\begin{aligned} \frac{\mathrm{d}x_a}{\mathrm{d}t} &= x_a(\mu_a - x_a + M_{ab}x_b) \\ \frac{\mathrm{d}x_b}{\mathrm{d}t} &= x_b(\mu_b + M_{ba}x_a - x_b), \end{aligned} \tag{1}$$

  and that you may further nondimensionalize time by setting $\mu_a \to 1$.

- Show that the Lyapunov function given by Tang, Yuan and Ma in Phys. Rev. E 2013 satisfies the Lyapunov conditions given in chapter 2 of Wiggins. In our notation, the Lyapunov function they provide is

$$\begin{aligned} V(x_a, x_b) = M_{ba}x_a^2/2 + M_{ab}x_b^2/2 \\ - M_{ba}\mu_a x_a - M_{ab}\mu_b x_b + M_{ab}M_{ba}x_a x_b \end{aligned} \tag{2}$$

  Some hints: 1) remember that $\hat{x}_a$ and $\hat{x}_b$ are "directions" (like $\hat{x}$ and $\hat{y}$) in a 2-dimensional space, 2) remember that $\dot{V} = \nabla V \cdot \dot{\mathbf{x}}$, where $\dot{\mathbf{x}}$ is the vector form of the dynamical system, and 3) assume $M_{ab} > 0$ and $M_{ba} > 0$ (this condition must be satisfied in order for there to be two stable steady states). Note that this equation satifies the Lyapunov conditions except for the fact that $V(\bar{x}) = 0$ for two different $\bar{x}$, corresponding to the two stable steady states. For this reason, this is called a *split Lyapunov function*.

- Generalize your code for solving the 2-dimensional gLV equations so that it can uses numpy arrays (`np.array`). Hint: consider the commands `np.dot(np.diag(mu), Y)` and `np.dot(np.diag(np.dot(M, Y)), Y)` do, and compare them to the 2-dimensional gLV equations.

- Modify the interaction matrix so that it is time dependent. Use Eq. (1) above, and set $\mu_a = \mu_b = 1$. Make your interaction matrix M time dependent, with $M_{aa} = M_{bb} = -1$ for all time. Then, choose M_{ab} and M_{ba} so that

$$\begin{cases} M_{ab} = M_{ba} = -.5, & \text{if } t < 5 \text{ or } t > 20 \\ M_{ab} = M_{ba} = -1.5, & \text{if } 5 <= t <= 20. \end{cases} \tag{3}$$

  To do this, make a function `def M(t):  some stuff; return np.array([[0, 1 + t], [2*t, 4]])`. Then simulate this system starting from the initial condition (.1, .9) and ensure it does what you would expect.