hw1

Daniel Sun

4/16/2018

1 LS 30B Homework 1 Spring 2018 Conley

by Daniel Sun

Homework 1—Due Thu, Apr 12:

- Re-read* section 4.1 in the textbook. You do not need to redo the in-text exercises in that section.
- Read section 4.2 in the textbook, and do in-text exercises 3**, 4, 5, and 6 in that section.
- Read section 4.3 in the textbook, and do in-text exercises 1–3 in that section.
- Do Further Exercises 1, 4, and 6 at the end of section 4.1.
- Do Further Exercises 1, 2, and 5 at the end of section 4.2.

* You should have read section 4.1 and done some/all of the in-text exercises in it at the end of your LS 30A class, if you took LS 30A in the Fall or Winter. But if not, and you're seeing this material for the first time, then be sure to read this section thoroughly and do the in-text exercises, even though you don't have to turn them in.

** You'll need to simulate this model in CoCalc to verify this.

You can view this homework as a webpage by going to https://cocalc.com/share/00174c99-9486-4fe3-ad4f-5092f8eb433a/TA%20Sandbox/Daniel%20Sun/hw/hw1.sagews?viewer=share

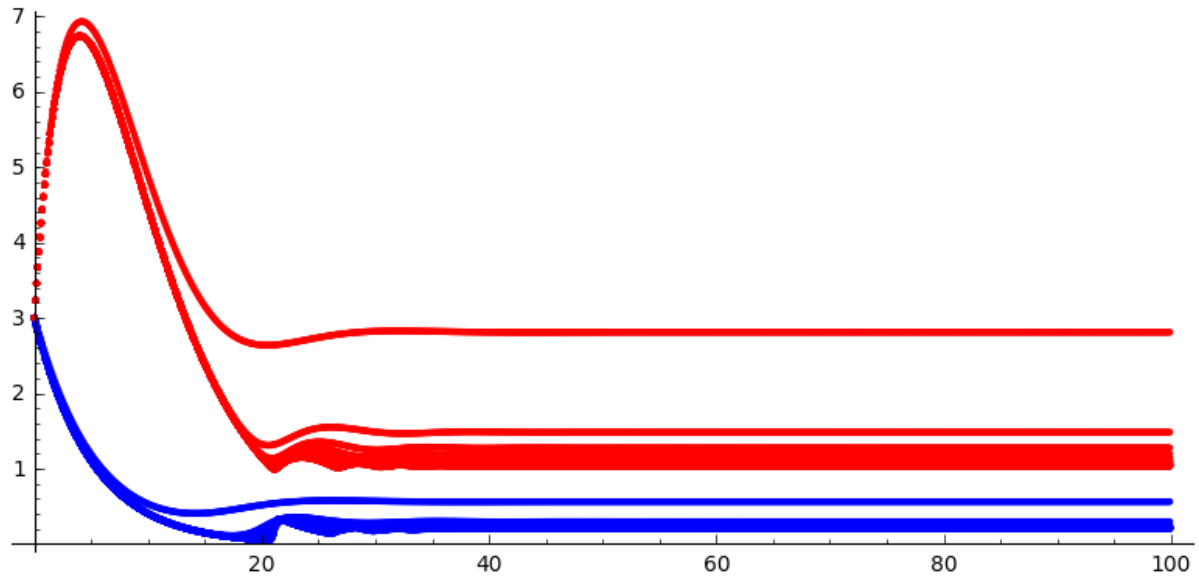
4.2.3

Let's try n at various levels from 2 all the way up to 100 and see if the system oscillates.

```
var('H','G')
k3 = k1 = .2

ic = [3,3]
t = srange(0,100,.1)
p = []
for n in srange(2,100,5): # try n at various levels from 2 to 100
    hprime(G,H) = 1/(1+G^n) - k1*H
    gprime(G,H) = H - k3*G
```

```
    sol = desolve_odeint([hprime, gprime], dvars=[H,G], ics=ic, \
    times=t)
    p.append( list_plot(zip(t,sol[:,0])) + list_plot(zip(t,sol[:,1]))\
    , color="red"))
show(sum(p), svg=False)

#Looks like it won't oscillate.
```

(H, G)



4.2.4

- Y(t-2) = value of Y 2 time units ago.
- Y(t) = value of Y right now at time t

4.2.5

Vmax controls the maximum ventilation rate, which is the horizontal asymptote.

4.2.6

This problem can be solved using Euler's method, keeping in mind that it is a delay differential equation. Be careful about what you are plugging in for X, $X(t-.2)$ and $X(t-.5)$. The procedure for this problem is exactly the same as a normal Euler's method integration, except for the slightly added complexity of needing to refer to what values occured in the past. We know that:

$$X(t) = 0.5 \ \ \forall t \leq 0$$

$$X_{next} = X_{old} + X'_{old} \cdot (stepsize)$$

$$X'(t) = 6 - \frac{16 \cdot X(t-0.2)^5}{1 + X(t-0.5)^5} \cdot X$$

Initial Step: $t = 0$

$$X(0) = 0.5 \qquad (t \leq 0)$$

$$X'(0) = 6 - \frac{16 \cdot (x(0-0.2)^5}{1 + (x(0-0.2))^5} \cdot x(0)$$

$$X'(0) = 6 - \frac{16 \cdot (0.5)^5}{1 + (0.5)^5} \cdot 0.5 \approx 5.758$$

$$X(0.1) = X(0) + X'(0) \cdot 0.1 = 0.5 + 5.758 \cdot 0.1 = 1.076$$

Second Step: $t = 0.1$

$$X'(0.1) = 6 - \frac{16 \cdot (x(0.1-0.2)^5}{1 + (x(0.1-0.2))^5} \cdot x(0.1)$$

$$X'(0.1) = 6 - \frac{16 \cdot (0.5)^5}{1 + (0.5)^5} \cdot \frac{71}{66} \approx 5.478$$

$$X(0.2) = X(0.1) + X'(0.1) \cdot 0.1 = 1.624$$

Third Step: $t = 0.2$

$$X'(0.2) = 6 - \frac{16 \cdot (x(0.2-0.2)^5}{1 + (x(0.2-0.2))^5} \cdot x(0.2) \approx 5.213$$

$$X(0.3) = X(0.2) + X'(0.2) \cdot 0.1 = 1.624 + 5.213 \cdot 0.1 = \boxed{2.145}$$

4.3.1

- V_0: natural rate of generation of F6P
- cSP^2: rate of conversion of S into P
- $-kP$: rate of decay of ADP into the environment

4.3.2

h is the half-saturation density, the value of N that maxes the sigmoid get to half of the maximum.

$$f(N = h) = \frac{C_{max} h}{h + h} = \frac{C_{max}}{2}$$

Thus, biologically, h is the number of prey at which an individual predator can consume half of its maximum consumption rate

4.3.3

For this problem, we want to find all of the equilibrium points. We will do this with the method of nullclines - finding functions that will make either P' or N' zero and then finding where those functions intersect.

$$N' = r_1 N(1 - \frac{N}{k}) - \frac{wN}{d+N}P$$

$$P' = r_2 P(1 - \frac{jP}{N})$$

Start by setting P' to zero:

$$0 = r_2 P(1 - \frac{jP}{N})$$

The above function can be factored into two parts. If either of the parts are equal to zero, the entire expression will be equal to zero.

1. $r_2 P = 0 \rightarrow P = 0$
2. $(1 - \frac{jP}{N}) = 0 \rightarrow P = \frac{N}{j}$

Coordinates (N,P) in any of these sets of points will make $P' = 0$.

Now, let's do the same thing for N'. We can factor out an N from everything, again leaving us with two functions multiplied together.

$$0 = N(r_1(1 - \frac{N}{k}) - \frac{w}{d+N}P)$$

1. $N = 0$
2. $r_1(1 - \frac{N}{k}) - \frac{w}{d+N}P = 0$

With a little bit of algebra, we can simplify (4) further:

1. $P = \frac{r_1}{k\omega}N^2 + (1 - \frac{d}{k})N + \frac{r_1 d}{\omega}$

(5) has the form of a quadratic equation.

Plugging in all of the variables, $(r1 = 1, r2 = .1, k = 7, d = 1, j = 1, w = .3)$ we end up with 4 nullclines, two from P' and two from N':

P': 1. $\boxed{P = 0}$ 2. $\boxed{P = N}$

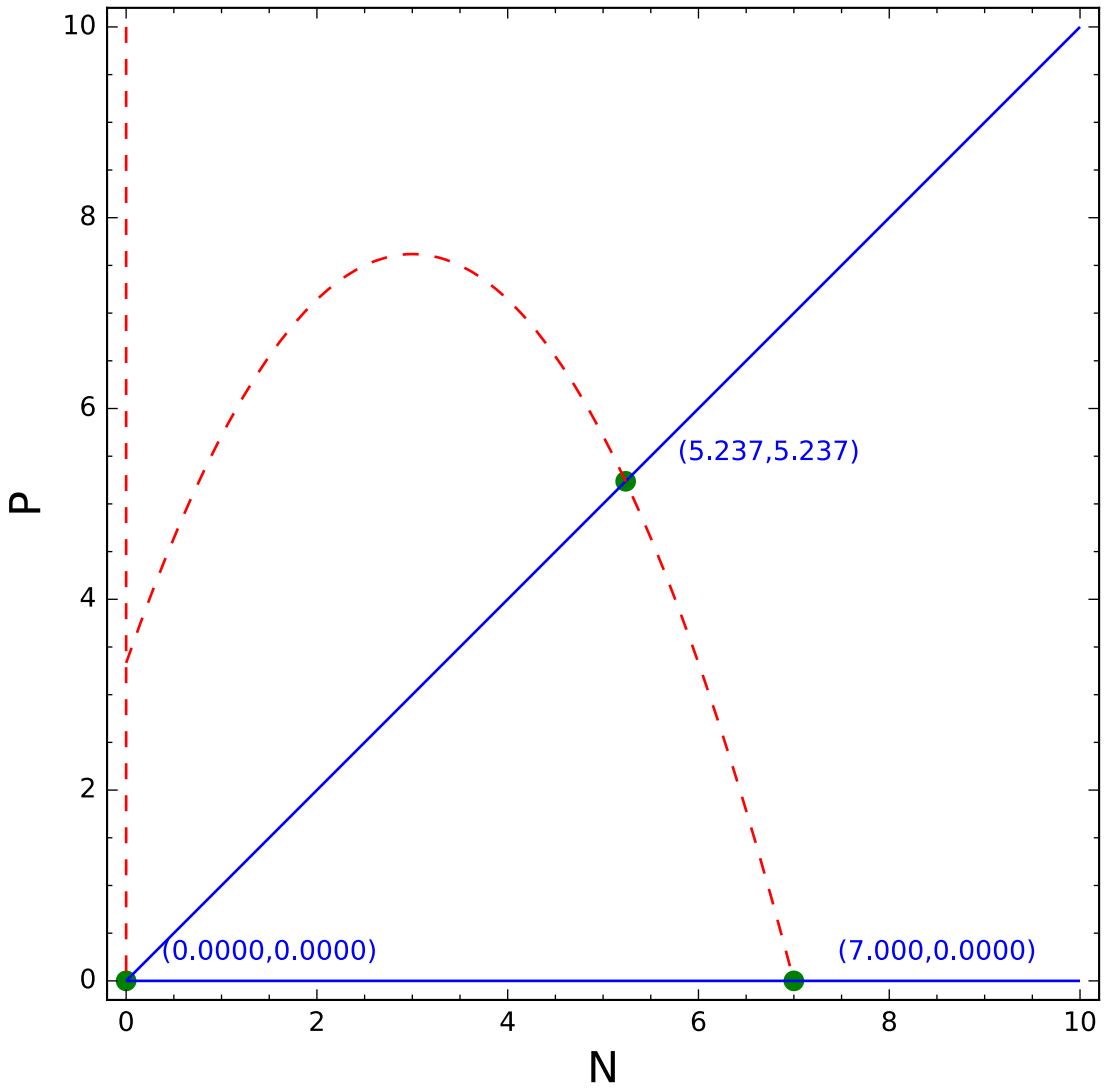
N': 1. $\boxed{N = 0}$ 2. $\boxed{P = \frac{10}{21}N^2 + \frac{20}{7}N + \frac{10}{3}}$

We'll plot them below:

```
p = list_plot([])
p += implicit_plot(P==0, (N, 0,10), (P, 0,10))
p += implicit_plot(P==N, (N, 0,10), (P, 0,10))
```

```
p += implicit_plot(P== -10/21*N^2 +20/7*N + 10/3, (N, 0,10), (P, \
    0,10), color="red", linestyle='dashed')
p += implicit_plot(N==0, (N,0,10), (P,0,10), color="red", linestyle=\
    'dashed', axes_labels=['N','P'], legend_label='sdf')

# this is a way of doing the above algebra with sagemath.
var('P','N')
r1 = 1
r2 = .1
k = 7
d = 1
j = 1
w = .3

assume(N >= 0)
assume(P >= 0)

pprime = r2*P*(1- j*P/N)
nprime = r1*N*(1-N/k) - w*N/(d+N)*P

# plot equilibrium points
sol = solve([pprime, nprime], [N,P], solution_dict=True)
for s in sol:
    if s[N] >= 0 and s[P] >= 0: # only add points if they're greater\
    than 0
        print "(N,P)  = ({},{})".format(s[N].n(digits=4), s[P].n(\
    digits=4))
        p += point([s[N], s[P]], size=60, color="green")
        p += text( "({},{})".format(s[N].n(digits=4), s[P].n(digits\
    =4)), [s[N]+1.5, s[P]+.3])
show(p, figsize=8)
```

```
(P, N)
(N,P)  = (0.0000,0.0000)
(N,P)  = (7.000,0.0000)
(N,P)  = (5.237,5.237)
```



The intersection of the blue and red nullclines are the equilbrium points. These occur at (0,0), (7,0) and (5.23,5.23).

4.1 FE4

Due to having a circadian rhythm, your body decides to go to sleep at regular times. When you experience jet lag, your body wants to go to sleep at different times that are + or - a certain amount from the regular time.

4.1 FE6

Stable eq. point outside a 2D limit cycle. A trajectory starting inside this limit cycle can't reach this point. That would mean that the trajectories would have to cross at some point, which is prohibited due to the FTEUSODE.

4.2 FE 1

a) this is negative feedback

b) find a way to reduce the time delay or sensitive feedback. Maybe commit to slowly increasing the number of times that you exercise or and weighing yourself often.

4.2 FE2

To prevent oscillations, put the thermostat and the A/C right next to each other.

4.2 FE5

Garibaldi population
Since we want to model this with only adults, this is a *delay differential equation*.
Let G = adult population of Garibaldi fishes

- larval garibaldis float as plankton, number of individuals joining a population is related to the ones that hatched 6 years earlier: $r * Eggs(t-6)$
- eggs laid: $bG(t-6)$
- fraction of eggs that hatch is declining sigmoid: $\frac{h^n}{h^n+G(t-6)^n}$
- Garibaldis' death rate: $-dG(t)$

So then, we have G' = eggs laid 6 years ago that survived - death rate of adults - eggs laid 6 years ago that survived = $rbG(t-6)\frac{h^n}{h^n+G(t-6)^n}$
and our final delay differential equation is

$$G'(t) = rbG(t-6)\frac{h^n}{h^n+G(t-6)^n} - dG(t)$$