Please provide complete and well-written solutions to the following exercises.

Due November 17, 1159PM PST, to be uploaded as a single PDF document to blackboard (under the Assignments tab).

Homework 10

Exercise 1. This exercise investigates the numerical solutions of the following systems of equations which appears in mathematical ecology as a predator prey system:

$$\frac{dr}{dt} = 2r - \alpha r f$$
$$\frac{df}{dt} = -f + \alpha r f$$

where $r(0) = r_0$, $f(0) = f_0$, t is time, $r(t)$ is the number of rabbits, $f(t)$ is the number of foxes, and α is a positive constant. This system has no known analytical solution, but it is known that the solutions for r and f are periodic, with the same period.

Use the following code if you wish.

```
function [t,y]=predpreymod(alpha,rstart,fstart,tstop,tolpick)
%Volterrra-Lotka predator prey model
%alpha is parameter in the eqn
%
% the eqn is of the form
% [r';f']=A*[r;f]
%
% alpha= parameter in the differential equations
% rstart= starting population of rabbits
% fstart= starting population of foxes
% tstop= time at which to stop the calculation
% tolpick= relative tolerance chosen to give the calculator
%
% e.g. predpreymod(1,10,5,10,1.e-6)

y0=[rstart;fstart];                 %initial cond
tspan=[0,tstop];                    %span of time
opts=odeset('reltol',tolpick,'outputfcn',@odephas2,'events',@events);
F=@(t,y) [2*y(1)-alpha*y(2)*y(1); alpha*y(1)*y(2)-y(2)]; %defining eqn

[t,y,te,ye,ie]=ode23(F,tspan,[rstart; fstart],opts);    %solve the eqn

%add title to phase plot created by matlab
```

```
usetitle=strcat('Original LV Phase Plane Plot alpha=',num2str(alpha), ...
     '--period~',num2str(mode(diff(te))), ...
     '--r_0:',num2str(rstart), ...
     '--f_0:',num2str(fstart));
title(usetitle);
xlabel('Rabbit Population');
ylabel('Fox Population');

%post processing
figure;
plot(t,y(:,1),'r-',t,y(:,2),'b-');
legend('Rabbit Population','Fox Population','Location','Northwest');
xlabel('Time');
ylabel('Population');
usetitle=strcat('Original LV Pred Prey Model alpha=',num2str(alpha), ...
     '--period~',num2str(mode(diff(te))), ...
     '--r_0:',num2str(rstart), ...
     '--f_0:',num2str(fstart));
title(usetitle);

    function [value,isterminal,direction]=events(t,y)
    %determine the period by tracking the min/max points of either of the
    %curves

        ydot=F(t,y);
        value=ydot(1);  %track the change in the rabbit population
        isterminal=0;   %do not stop at critical values
        direction=1;    %track local minima
    end
end
```

- Compute some solutions for different values of r_0, f_0 and α. For example, once you save the function file as `predpreymod.m` in the current directory, you could call this function with the command `predpreymod(1,1,1,10,10^-5)`
- You might observe some periodicity behavior in these plots. To examine this behavior, consider a particular substitution. Note that $(r, f) = (\frac{1}{\alpha}, \frac{2}{\alpha})$ is a stable equilibrium point (i.e. when r, f take these values at one time, they take these values at all future times, since $r'(t) = f'(t) = 0$ in this case). With this in mind, we adjust our functions by this value as follows. For any $t \in \mathbf{R}$, define

$$u(t) := r(t) - \frac{1}{\alpha}$$

$$v(t) := f(t) - \frac{2}{\alpha}$$

  Then we have $u' = r'$, $v' = f'$ and ignoring the uv terms in our equations, derive the approximations

$$u' \approx -v, \qquad v' \approx 2u.$$

Taken together these equations yield

$$v'' \approx 2u' \approx -2v.$$

The equation $v'' = -2v$ is exactly the equation for a harmonic oscillator (notice if we solve for v we necessarily solve for u). For instance, an analytic solution is

$$v = A\cos(\sqrt{2}t + \phi)$$

where v has period $\sqrt{2}\pi \approx 4.44288$. In fact, all solutions will have such a period, and indeed, this is almost exactly the period observed for our third example above. Therefore, these manipulations give a valid analysis of the original system, where we observe oscillatory phenomena. Moreover, from this formal analysis, we can trust the solutions produced by the computer, and not have to worry that they are an artifact.

- Now investigate solutions to a system which is a modification of our previous equations

$$\frac{dr}{dt} = 2\left(1 - \frac{r}{R}\right) - \alpha r f$$
$$\frac{df}{dt} = -f + \alpha r f$$

where we are in the same situation as before except now R is a constant, and R essentially represents the maximum allowable population of rabbits. Compare the solutions of this equation to solutions of our previous equation under similar conditions.

Exercise 2. We now investigate solutions to equations which describe the flight of a projectile with wind. In this case, we model a cannonball shot from the origin in the direction of the positive x-axis. We also treat the x-axis as the ground, so we consider the flight of the projectile done when the projectile hits the ground. The equations are as follows

$$x'(t) = v\cos(\theta), \qquad y'(t) = v\sin(\theta)$$
$$\theta'(t) = -\frac{g}{v}\cos(\theta), \qquad v'(t) = -\frac{D}{m} - g\sin(\theta), \qquad \forall\, t \geq 0,$$

where $\theta = \theta(t)$ is the angle that the velocity vector makes with the x-axis, $x(t)$ and $y(t)$ are the usual spacial coordinates, $v(t)$ the speed and $D(t)$ is a function of time representing the drag with

$$D(t) = \frac{c\rho s}{2}\left((x'(t) - w(t))^2 + (y'(t))^2\right)$$

where $w(t)$ is also a function of time representing the wind, $c = .2$ is the drag coefficient, $\rho = 1.29$ kg/m^3 is the density of the air, and $s = .25$m^2 is the projectile's cross-sectional area. Also, in the above equations we have $v_0 = 50$ m/s the initial speed, $m = 15$ kg the weight of the cannonball and $g = 9.81$m/s^2 acceleration due to gravity.

- Plot a range of different initial angles for trajectories on one plot, where one plot corresponds to one wind function. Then, report the attributes of the trajectory of maximal distance, with its angle in degrees noted. When making these plots, consider the following four wind functions
  (1) $w(t) = 0$.
  (2) $w(t) = -10$.
  (3) $w(t) = 10$ if $\lfloor t \rfloor$ is even, and zero otherwise.

(4) w(t) is a Gaussian random variable with mean 0, standard deviation 10.
In Matlab, item (3) can be written as `10*(~mod(floor(t),2))*(t >= 0)`, and item (4) is `10*randn`

- In each plot, report (in the following order) information for the maximal projectile: the function w, the initial angle of flight in degrees, the flight time, the distance of the projectile, the impact speed (keeping in mind an initial speed of 50m/s), and the number of steps taken by the solver in making the calculation. Choose a relative tolerance of $1.e-6$ in all of our trials so that we are able to compare the performance of the calculations under each of the wind functions.
- Which wind function requires the most computation time?

Exercise 3. In a previous exercise, we used the command `orbitode` to investigate the motion of a small body subject to two gravitational forces. The code below solves this equation similar to `orbitode`. In the code below, the motion of the body is governed by its interaction with a nearby planet and a sun that is essentially infinitely far away (on the negative x-axis).

Run the code below, and try different values of `tspan` such as `tspan=[0 100]`. Also, try different Matlab ODE solvers such as `ode45` and stiff solvers such as `ode23s` and `ode15s`. Which solver seems to do the best? For example, does the solution for one solver have some type of qualitative behavior that suggests the solution is not at all accurate?

```
function planetaryode
tspan = [0 30];
y0 = [.994 0 0 -2.00158510637908252240537862224];
[t,y] = ode23(@(t,y) myode(t,y), tspan, y0);
plot(y(:,1),y(:,2));
end

function dydt = myode(t,y)
    mu=.012277471;
    d(1) = ((y(1) + mu).^2 + (y(2)).^2 ).^(3/2);
    d(2) = ((y(1) -(1-mu)).^2 + (y(2)).^2 ).^(3/2);

    dydt = zeros(4,1);
    dydt(1) = y(3);
    dydt(2) = y(4);
    dydt(3)= y(1)+2*y(4)- (1-mu)* (y(1)+mu)./d(1)  -mu *(y(1)- (1-mu))./ d(2);
    dydt(4)= y(2)-2*y(3)- (1-mu)* (y(2))./d(1)  -mu *(y(2))./ d(2);
end
```

Exercise 4. We investigate solutions of the differential equation

$$y'(t) = -1000(y(t) - \sin(t)) + \cos(t), \quad \forall\, t > 0, \quad y(0) = 1.$$

Using Matlab's equation solver find the exact solution. For example, the syntax for solving $y'(t) = ty(t)$, $y(0) = 2$ would be

```
    syms y(t)
```

```
dsolve(diff(y(t),t)==t*y(t) , y(0)==2)
```

Compare the performance of the textbook function `ode23tx` to Matlab's stiff solver `ode23s`. How does each method perform? That is, which one uses less time steps, say when you solve the equation on the region $0 \leq t \leq 1$? If you want, you can plot both solutions next to the exact solution you found with `dsolve`.

Denote $g(t) := \sin(t)$, $\lambda := 1000$, and consider the following rewritten version of the differential equation (without the initial condition):

$$y'(t) = -\lambda(y(t) - g(t)) + g'(t)$$

If we introduce the variable $z(t) = y(t) - g(t)$ we then have

$$\lambda z(t) = z'(t)$$

and if we look at our exact solution when $y(0) = 1$, we see that z kind of measures the distance between some general solution and the "attracting" solution $y(t) = \sin(t)$. Therefore, since we have $\lambda = 1000$ in this case, we see that this distance to the attracting solution changes very rapidly. This is exactly why our problem is stiff. Solutions move towards $\sin(t)$ at roughly 1000 times their distance from it. This is much faster than the "stable" solution $y(t) = \sin(t)$ varies (all its derivatives are bounded by 1). Therefore, we have an analytical explanation of exactly why the problem is stiff.

Exercise 5. Consider the following system of equations

$$y_1'(t) = -1000y_1(t) + y_2(t), \qquad y_2'(t) = 999y_1(t) - 2y_2(t).$$
$$y_1(0) = 1, \qquad y_2(0) = 0.$$

(a) Using Matlab, solve this equation exactly. Describe the behavior of y_1, y_2 and y_1/y_2 as $t \to \infty$.

(b) Solve the system using Euler's and backwards Euler's methods up to the time $T = 10$. In both cases find (approximately) the largest h which produces an absolute error E_T at time T where $E_T < 10^{-5}$ (this absolute error is the maximum of the errors of the two variables $y_1(T)$ and $y_2(T)$).

(For an equation of the form $y'(t) = f(y(t))$ where in this example both y and f take values in $\mathbf{R}^2$, Euler's method is the recursion $y(t + h) = y(t) + hf(y(t))$, and backwards Euler's is the implicit equation $y(t+h) = y(t)+hf(y(t+h))$. In order to use backwards Euler's you should either solve the equation $y(t+h) = y(t)+hf(y(t+h))$ for $y(t + h)$, or you can use a built-in backwards Euler solver.)

(c) Repeat part (b) for Euler's method, changing your error tolerance to $E_T < 10^{-6}$.

(d) Plot $\log(E_T)$ as a function of n, for Euler's method. Try to explain the origin of any interesting behavior of this plot.

Exercise 6. This exercise investigates the double pendulum. We have m_1, m_2 as the masses of our two bobs, θ_1, θ_2 the angle each rod makes with the $y - axis$ (0 indicates the rod is hanging straight down, an angle with a small and positive value indicates the rod is pointing into the lower right quadrant, etc.) and ℓ_1, ℓ_2 are the lengths of the rods. The positions of the bobs (x_1, y_1) and (x_2, y_2), where

$$x_1 = \ell_1 \sin(\theta_1), \qquad y_1 = -\ell_1 \cos(\theta_1)$$

$$x_2 = \ell_1 \sin(\theta_1) + \ell_2 \sin(\theta_2), \qquad y_2 = -\ell_1 \cos(\theta_1) - \ell_2 \cos(\theta_2)$$

and physics leads to a pair of second-order, nonlinear ODEs that describe the motion of the pendulum. Perform your investigation with the `swinger` program in the NCM package

(a) When the initial angle of the double pendulum is small, note that it acts like a normal pendulum. How large can you take the initial angle while maintaining this behavior?
(b) The initial orbit of the `swinger` function is interesting because it is periodic. It keeps repeating the same thing over and over, rather than being chaotic like most all other initial conditions. Try to observe some other initial conditions that lead to periodic behavior. (To do this, click on different points on the plot.)
(c) If you run `swinger` for a while, click `stop` and then type in the command line `get(gcf,'userdata')` the initial angles θ_1, θ_2 (the last ones that were chosen) are returned.
(d) Now, attempt to change the way the initial conditions of the system are chosen, by specifying the initial angles θ_1 and θ_2, instead of specifying the initial values of x_2, y_2. In the case where θ_1 and θ_2 differ, one sees by drawing a parallelogram that if the position of x_2 is specified, then the initial value of x_1 has two possible values. From this parallelogram, we see that the angles θ_1 and θ_2 are simply swapped between each case, corresponding to the two choices of x_1. Therefore, in the Matlab code that you write, you can choose either of these two orderings of the angles θ_1, θ_2.
(e) Now, change the function to handle different masses. That is, extend the main function definition to accept m_1,m_2 as the respective weights for the masses of the bobs.
(f) Modify the `swinger` function so that lengths other than $\ell_1 = \ell_2 = 1$ are possible
(g) Modify the function so that we can change g, the acceleration due to gravity.
(h) Alter the program so that it solves its differential equation use the textbook function `odetx`.
(i) Is the ODE solved by `swinger` stiff? (The book defines stiffness as a "wide disparity in the time scales of the components of the vector solution.")

(If it helps, check out some more information on this problem here on page 50:
https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/odes.pdf)