Umeå Universitet
Institutionen för matematik och matematisk statistik 28-09-2018

Analyzing microbes lifecycles and survival in different initial conditions

Viktor Vigren Näslund (vina0010@student.umu.se)

Fysikens Mattematiska Metoder, Del 1
Lärare: Eric Libby

1 Assignments

1.1 Assignment 1

```
clear all;
clc;
load('randwebs.mat');
load('randinits.mat');

intmat = 1;

options = odeset('NonNegative',1);

A = randwebscell{intmat};
tend = 9000;
tspan = [0,tend];
y0 = randinitscell{intmat};

f = @(t,x) x.*A*x;

[t,x] = ode45(f,tspan,y0,options);
```

1.2 Assignment 2

```
clear all;
clc;
load('randwebs.mat');
load('randinits.mat');

intmat = 1;

options = odeset('NonNegative',1);

A = randwebscell{intmat};
tend = 2000;
tspan = [0,tend];
y0 = randinitscell{intmat};

f = @(t,x) x.*A*x

[t,x] = ode45(f,tspan,y0,options);

aliveThreshold = exp(-5)
xalive = zeros(1,length(x(1,:)));

start = length(x(:,1)) - floor(length(x(:,1))*0.1);
for i = start:length(x(:,1))
    xalive = xalive + ( x(i,:) > aliveThreshold );
end
xalive = (xalive > 0);

display(xalive);
sum(xalive)
```

1.3 Assignment 3

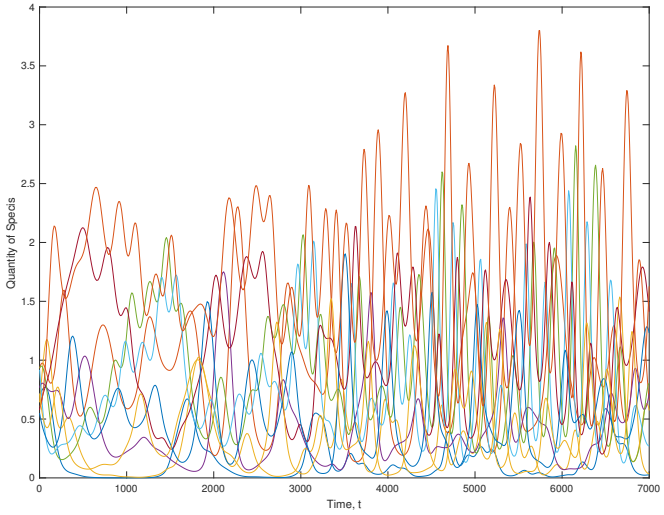
```
%assignment3and4 and optional plot for assignment 5
clear all;
clc;
load('randwebs.mat');
load('randinits.mat');

options = odeset('NonNegative',1);

tend = 9000;

interactionset_alive = zeros(1000,1);
species_alive = zeros(11,1);

hasPlotted = 0;
for k = 1:1000

    intmat = k;

    A = randwebscell{intmat};
    tspan = [0,tend];
    y0 = randinitscell{intmat};

    f = @(t,x) x.*A*x;

    [t,x] = ode45(f,tspan,y0,options);

    %change for different threshhold values
    aliveThreshold = exp(-5);
    xalive = zeros(1,length(x(1,:)));

    start = length(x(:,1)) - floor(length(x(:,1))*0.1);
    for i = start:length(x(:,1))
        xalive = xalive + ( x(i,:) > aliveThreshold );
    end
    xalive = (xalive > 0);

    alivesum = sum(xalive);
    interactionset_alive(k) = alivesum;
    species_alive(alivesum+1) = species_alive(alivesum+1) + 1;

    %uncommnent to se plot of all species alive, assignment 5
    %{
    if (alivesum==10) && (hasPlotted == 0)
        plot(t,x)
        hold on
        xlabel('Time, t')
        hold on
        ylabel('Quantity of Specis')
        hasPlotted = 1;
    end
    %}
end
% displayes a matrix over how many species that survived for each ...
    initial condition
disp(interactionset_alive)
% displayes how many times it occured that a specif count of species
% survived
disp(species_alive)
```

1.4 Assignment 4

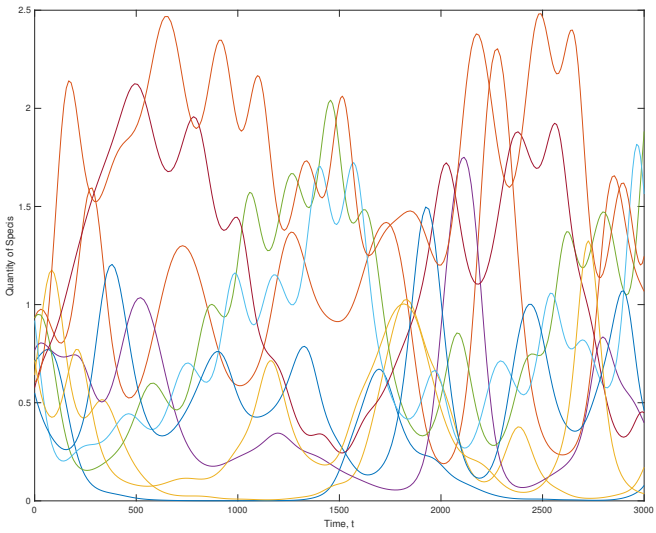
Assignment 4 uses the same code as in assignment 3. The answer is the second output of the code in assignment 3.

1.5 Assignment 5

By uncommenting the code from line 42 to line 51 a plot of the first solution where all species survives will be shown. This plot is shown in Fig. 1 and Fig. 2 for timescales of [0,7000] and [0,3000]. The timescale to solve and plot for can be changed by changing the value of the variable **tend**



Figur 1: Solution of a initial-condition were all species survives, timescale is [0, 7000]



Figur 2: The same solution as in Fig. 1 but for a smaller timescale [0, 3000]

1.6 Assignment 6

The number of times a 0,1,...,10 species survives in a solution is shown in the table below.

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	325	184	184	164	82	51	10

Tabell 1: Table that shows how many times a specific number of species survives.

From Table. 1 we can conclude that it's more likely that 5 species survives than 10 ($325 > 10$).

1.7 Assignment 7

Using the same method and table (Table . 1) We conclude that it is more likely that 5 species survives than 0. ($325 > 0$)

1.8 Assignment 8

```
clear all;
clc;
load('randwebs.mat');
load('randinits.mat');

options = odeset('NonNegative',1);

tend = 9000;

interactionset_alive = zeros(1000,1);
aliveForDifferentSolutionTimes = zeros(1000,2);
species_alive = zeros(11,1);

hasPlotted = 0;

for j = 1:2
    tend = 8000*j;
    for k = 1:1000

        intmat = k;

        A = randwebscell{intmat};
        tspan = [0,tend];
        y0 = randinitscell{intmat};

        f = @(t,x) x.*A*x;

        [t,x] = ode45(f,tspan,y0,options);

        %change for different threshhold values
        aliveThreshold = exp(-5);
        xalive = zeros(1,length(x(1,:)));

        for i = (length(x(:,1))-2):length(x(:,1))
            xalive = xalive + ( x(i,:) > aliveThreshold );
        end
        xalive = (xalive > 0);

        alivesum = sum(xalive);
        interactionset_alive(k) = alivesum;
        species_alive(alivesum+1) = species_alive(alivesum+1) + 1;

        %uncommnent to se plot of all species alive, assignment 5
        %{
        if (s==10) && (hasPlotted == 0)
            plot(t,x)
            hold on
            xlabel('Time, t')
            hold on
            ylabel('Quantity of Specis')
            hasPlotted = 1;
        end
        %}

    end
    aliveForDifferentSolutionTimes(:,j) = interactionset_alive;

end
disp(sum(aliveForDifferentSolutionTimes(:,1) & ...
    aliveForDifferentSolutionTimes(:,2)) )
```

1.9 Assignment 9

```
clear all;
clc;
load('randwebs.mat');
load('randinits.mat');

%format LONGE;

options = odeset('NonNegative',1);

%changes for how long we solves the equations
tend = 9000;
tspan = [0,tend];

%change to check which microbes are oscilating in corresponding
%initialconditions and to plot the solution
initialConditionsNum = 1000;

for k = 1:1000

    intmat = k;

    A = randwebscell{intmat};
    y0 = randinitscell{intmat};

    f = @(t,x) x.*A*x;

    [t,x] = ode45(f,tspan,y0,options);

    smallThreshold = exp(-15);
    start = length(x(:,1)) - floor(length(x(:,1))*0.15);
    checkDerivativeSignArray = x(start:end,:);
    signChanges = zeros(1,10);
    Y = diff(checkDerivativeSignArray);
    for i=1:(length(Y(:,1))-1)
       temp = ( (Y(i,:).*Y(i+1,:)) < 0);
       temp2 = (Y(i,:) > smallThreshold); % don't consider signchange ...
           for small values since we can have +-0
       signChanges = signChanges + (temp.*temp2);
    end

    if k == initialConditionsNum
        plot(t,x)
        hold on
        xlabel('Time, t')
        hold on
        ylabel('Quantity of Specis')
        disp(signChanges); %shows how many times the derivative makes ...
            a signchange
        disp(signChanges > 0); %shows which microbes that oscillates ...
            in the end of the solution
        disp(sum(signChanges > 0)); %shows how many microbes that ...
            oscillates in the end of the solution
    end

end
```

1.10 Assignment 10

To decide if a species has survived at the end of a solution we simply check if all the values at the end of a solution is lower than some threshold. Why we check a few values in the end and not simply one is to be sure that we accidentely picked a local maximum if the solution oscillated or if some numerical error made some sulutions higher than other. This might not even be problems if matlab has good enough numerical accuracy but it doesn't hurt to check a few more values. However what we call might be a bit ambigous. A bad threshold here might say that we consider some species that are alive dead and another bad threshold might consider to many solutions alive if there still are numbers left in the solution due to numerical error.

In assignment 6 and 7 we say that it is more likely that 5 species survives than 0 or 10. But this conclusion is strictly working for our initial-conditions. We might get a different answer if we were to use other initial-conditions.

To determine if we have solved the differential equations for long enough i used the following method. If we use the code where we find the number of species that survives in each setting of initialconditions. The number of species that survives in each initialcondition should be the same if we solves for a long period of time. The Matlab code in assignment 8 shows how the diff-equations are solved with two different timescales. The two solutions sum of species for each initial condition are then compares using the == operator, this gives a vector with the same length as the number of initial-conditions that exist. If two solutions with the same initial condition but different timescales have the same sum of species the == operator will give a one in that position in the vector, otherwise it will give a zero there. If we sum this vector we should get the number of times the answer to how many survived were the same.

If this summation were to give the number 1000. We would know that we got the same amount of alive species in each condition for the smaller timescale as we did in the larger timescale. This would mean that the lower timescale is a good enough solution time. And that no changes to the state of species has been made after the smaller timespan.

In addition to this test i also looked at some plots of solutions for different timescales and concluded that the end of the solutions looked the same as the end of solutions with smaller timescales. This whould mean that the solutions for the larger timescales don't change significantly from the solutions of the smaller timescales thus the smaller timescales are sufficient. This method might be deceiving, my visual test showed me that a solution time of maybe 4000-5000 should be enough but my code-test showed that it need atleast something like 8000 or higher. One drawback of the code-method i used is it's dependence on the code that decides if a species has survived or not. This means that my code method in assignment 8 is subject to the same issues as the code that decides wheter or not a species has survived or not.

Question 9 imposes some interesting questions. My biggest issue was with the definition of oscillation. I chose to call an oscillating solution a solution that has a changing derivative and that the change from one step to another is big enough that we can be certain the change isn't due to numerical error. Therefore i check first if the derivative changes sign anytime during the last 15 percent of the solution. And after that if the changes magnitude is big enough. If a species fullfill both of these conditions are i will call them oscillating. This means that the solutions that goes to a steady state solution (those that increases in the beggining and seems to stay around a fixed value), even if the change in derivative sign. They will probably not be called oscillating since the changes will be to small for my oscillating requirements. Worth noting though is that those that reached a steady state solution both those

that fixated around a value and those that died seem to always ha the same sign on the derivative in the last 15 percent of the solution.