

MATH971 – Autumn 2010

Tutorial Sheet – Week 6

This tutorial sheet covers chapter 3.

If you need a quick overview of Maple then print out the following document <http://www.uow.edu.au/content/groups/public/@web/@inf/@math/documents/doc/uow046825.pdf>. There is a link to this document on the MATH971 web-page under prerequisites.

1. A particular form of the *Holling-Tanner* model is given by

$$\begin{aligned}\dot{x} &= f(x, y) = x \left(1 - \frac{x}{7}\right) - \frac{6xy}{7 + 7x}, \\ \dot{y} &= g(x, y) = 0.2y \left(1 - \frac{Ny}{x}\right).\end{aligned}$$

In this assignment you investigate how the behaviour of this model depends upon the value of N by integrating the governing equations numerically using the initial condition $(x, y) = (7, 1)$. You can download code to integrate this system of equations from the course web-page.

www.uow.edu.au/~mnelson/teaching.dir/math971.dir/code.dir/doubleode.html.

The code is also given at the end of this assignment.

As outputs you should generate figures showing $x(t)$ as a function of time, $y(t)$ as a function of time and $y(t)$ against $x(t)$.

You might also like to try plotting the solution $(x(t), y(t), t)$ as a three-dimensional figure.

- (a) Take $N = 0.21$ and integrate the equations to $t = 1000$. Comment upon your finding.
- (b) Take $N = 0.3$, $N = 0.6$ and $N = 0.7$. Integrate the equations for a ‘long’ time (at least 200 time units). Only use the last data from the last 100 time units. On the same graph plot $y(t)$ against $x(t)$ for each value of N . What do you notice about your three solutions?

Make a conjecture about what is happening. Take a fourth value of N , higher than 0.7, and add the equivalent data to your plot.

Maple code

```
# doubleode.maple Maple code to integrate the Holling-Tanner model
# 13.04.10
#
# endemic.maple Maple code to integrate a simple SIR model for an
# 14.10.03 endemic disease with vaccination.
#
with(linalg):
with(plots):

N := 0.2; # Primary bifurcation parameter.

tstart := 0; # the initial value for time.
tend := 10; # the final value for time.

# define the differential equations
de1 := diff(x(t),t) = x(t)*(1-x(t)/7.0) -6*x(t)*y(t)/(7.0+7.0*x(t));
de2 := diff(y(t),t) = 0.2*y(t)*(1-N*y(t)/x(t));
```

```

interval := 5.0:      # number of points calculated per unit of time.
xinitial := 7.0:     # initial value for x
yinitial := 1.0:     # initial value for y

ic1 := x(0) = xinitial; # initial condition for x
ic2 := y(0) = yinitial; # initial condition for y

deqs := {de1,de2}:   # define the system of DEs.
ics := {ic1,ic2}:    # define the initial conditions.
vars := {x(t),y(t)}: # define the dependent variables.

# define the values of time at which the output is calculated.
ans := array([seq(i/interval,i=tstart..(interval*tend))]);
# integrate the system of differential equations
#
#
solnum := dsolve(deqs union ics,vars, \
                 type=numeric, output=ans, maxfun=300000):

# Print the values of the independent and dependent variables at the final
# integration point (tend). NOTE that the output is NOT necessarily in the
# format x(t), y(t):
#
solmatrix := eval(solnum[2,1]):
rowsize := rowdim(solmatrix):
eval(solnum[1,1]); # this gives us the output order
row(solmatrix,rowsize); # the output at the final integration point

# Plot some figures. You WILL need to play around with these
# commands to get the best possible figures for your report.
#

# Figure One. Plot the solution in the x-y plane.
odeplot(solnum,[x(t),y(t)],labels=["x(t)","y(t)"]);

# Figure Two. Plot the x-time and y-time data on the
# same figure. Make sure you know which one is which.
p1 := odeplot(solnum,[t,x(t)],colour=BLUE):
p2 := odeplot(solnum,[t,y(t)],colour=RED):
display({p1,p2},title="Population of x and y",\
        labels=["time","population"]);

# Figure Three. Show how to use the view function Plot the value of x.
#
#
odeplot(solnum,[t,x(t)],colour=RED,labels=["time","x(t)],\
        title="Population x(t)",\
        view=[tstart+5..tend,1..5]);
# Figure Four. Plot the y-time data.
odeplot(solnum,[t,y(t)],labels=["t","y(t)"],title="Population y(t)");

N := 'N':
deqs := 'deqs':
de1 := 'de1':
de2 := 'de2':

```

```
ics      := 'ics':  
ic1      := 'ic1':  
ic2      := 'ic2':  
interval := 'interval':  
p1       := 'p1':  
p2       := 'p2':  
rowsize  := 'rowsize':  
solmatrix := 'solmatrix':  
solnum   := 'solnum':  
tend     := 'tend':  
tstart   := 'tstart':  
vars     := 'vars':  
xinitial := 'xinitial':  
yinitial := 'yinitial':
```