Using MATLAB for Solving Differential Equations

Introduction to Mathematical Modelling
V63.0051 – Spring 1999

April 19, 1999

Many differential equations with which you will be confronted in applications are not amenable to exact solution techniques which you learn in an ODE class. One tool in the analysis of such equations is the use of numerical differential equation solving packages. Others involve advanced mathematical analysis which seeks to describe the qualitative behavior and obtain estimates for the solution of a differential equation, even though an explicit formula for the solution is not available. We will just show here how MATLAB can be used as a tool to solve and plot approximate solutions to differential equations.

1 Scalar Differential Equations

First, let's study the case of a single differential equation with one state variable, which has the general form:

$$\begin{aligned} \frac{dz}{dt} &= F(t, z), \\ z(t_0) &= z_0. \end{aligned}$$

where we think of the variable t as time and z as the variable describing the state of the system. In class, we have only discussed "autonomous" differentail equations in which the right hand side is only a function of the state variable $F(t, z) = F(z)$, but it's just as easy for MATLAB to deal with the more general case in which the right hand side may depend explicitly on time and/or the state of the system.

Let's show how MATLAB could be used to solve the Verhulst model:

$$\begin{aligned} \frac{dN}{dt} &= rN\left(1 - \frac{N}{K}\right), \\ N(t_0) &= N_0. \end{aligned} \tag{1.1}$$

The state variable is written as N instead of z, but this doesn't matter for MATLAB of

course. The parameters r and K are specified constants representing the unhindered growth rate and carrying capacity, respectively. Suppose we forgot about how to solve the Verhulst model exactly by separation of variables. Here's how MATLAB could plot an approximate solution without knowing about the explicit formula for the exact solution.

First we write a function M-file which computes F, the right hand side of the differential equation. For the case of the Verhulst model,

$$F = F(t, N) = rN\left(1 - \frac{N}{K}\right).$$

Here's the function M-file I wrote to do this, which I saved as `verhulst.m`:

```
function F=verhulst(t,N)
% Function which returns the population rate of change for
% Verhulst model.
% User should set parameters in USER PARAMETERS section
%
% dN/dt = F(N) = r (1 - N/K) N
%
% Input variables:
%       t = time
%       N = current population size
%
% Input parameters (external):
%       r = unhindered per capita growth rate of population
%       K = carrying capacity of population
%
% Output variables:
%       F = rate of change of population (dN/dt)

% USER PARAMETERS
r = 0.55;
K = 665;

% MAIN COMPUTATION
F = r * ( 1 - N/K) * N;

return
```

Notice that even though F doesn't really depend on t, I included an input argument for t in the first line. You have to do this for the ODE solver to work properly. The first argument of the function should be the time variable t, and the second argument of the function should be the state variable, which in this case is called N.

MATLAB of course requires specific numbers for all input parameters, so values of r and K must be provided. It's convenient to collect all constant input parameters in one place (such as I did under the "% USER PARAMETERS" section), so that the user can change them easily.

Now suppose we want to solve the Verhulst model (1.1) over the interval of time $0 \leq t \leq 18$ hours. (You should always keep some consistent set of units, such as "hours" for t, but MATLAB of course doesn't care what system of units you use. But it's important to keep track of units to make sure your computations are consistent and so that you can intepret them in terms of real world quantities). Suppose that at time $t_0 = 0$ hours, the population size is $N_0 = 9.6$ (as measured with respect to some biomass unit system). Here's how MATLAB can do this for you:

```
>> tspan = [0 18];
>> No = 9.6;
>> [t,N] = ode45('verhulst',tspan,No);
>> plot(t,N)
>> title('MATLAB Solution of Verhulst Model')
>> xlabel('t (hours)')
>> ylabel('N (biomass)')
```

The variables `t` and `N` are actually column vectors which store the values of t and N at each time step which MATLAB used to solve the ordinary differential equation. But you don't need to concern yourself too much with this point if it's confusing. The resulting plot is shown in Figure 1. Notice that the numerical solution indeed has the same shape as the exact solution. If we were to plot the explicit formula for the solution, it would match very closely.

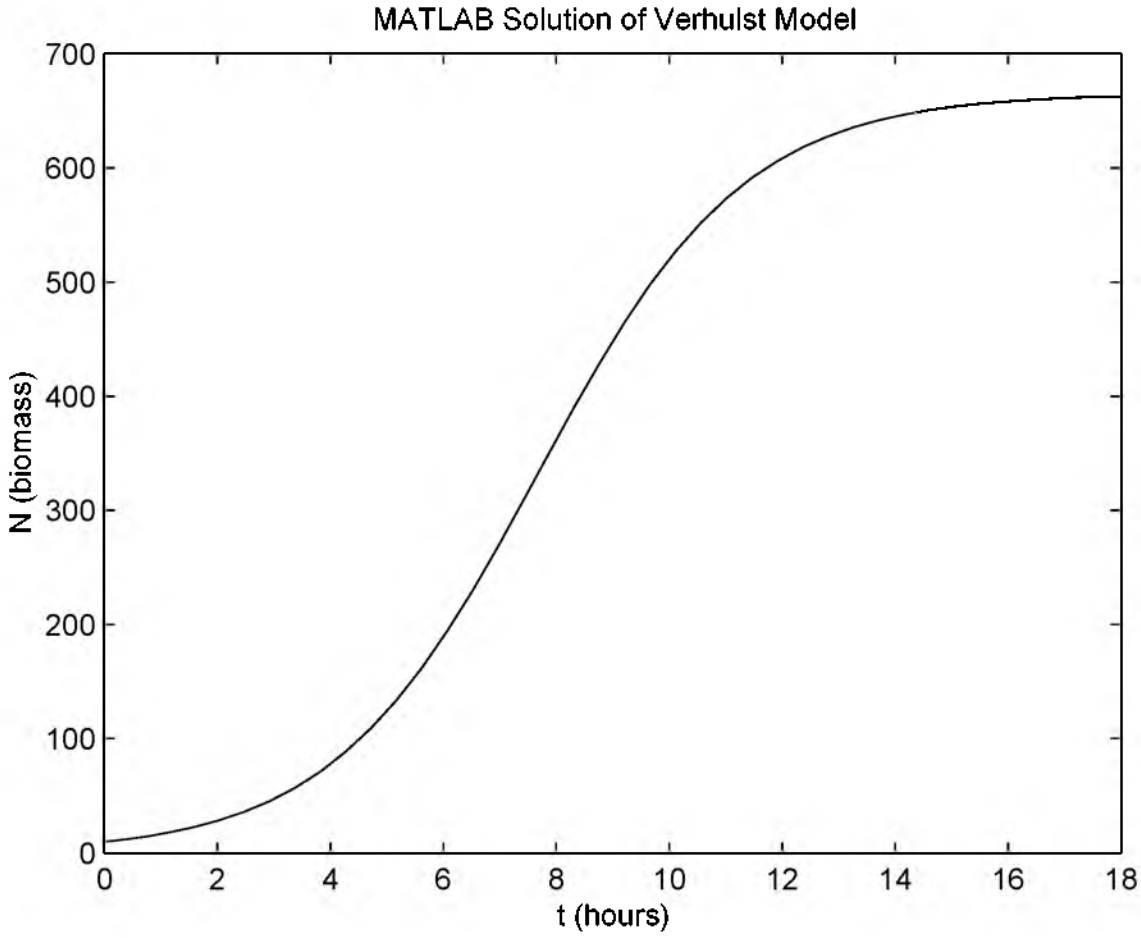

Figure 1: Solution of Verhulst model using MATLAB differential equation solver `ode45`. Parameters: $r = 0.55$/hour and $K = 665$. Initial condition: $N(0) = 9.6$.

2 Systems of Differential Equations

Often times one is considering the evolution of a state which requires two or more variables to describe. If the rate of change of the system can be expressed as an instantaneous function of time and the current state of the system, then one can mathematically model such a system as a differential equation for a vector:

$$\begin{aligned} \frac{d\mathbf{z}}{dt} &= \mathbf{F}(t, \mathbf{z}), \\ \mathbf{z}(t_0) &= \mathbf{z}^{(0)}. \end{aligned} \tag{2.2}$$

where $\mathbf{z}$ is a vector describing the state and $\mathbf{F}$ is a vector-valued function of time. This can be written out in component form as follows:

$$\begin{aligned} \frac{dz_1}{dt} &= F_1(t, z_1, z_2, \ldots, z_n), \\ \frac{dz_2}{dt} &= F_2(t, z_1, z_2, \ldots, z_n), \\ &\vdots = \vdots, \\ \frac{dz_n}{dt} &= F_n(t, z_1, z_2, \ldots, z_n), \end{aligned}$$

with initial conditions:

$$\begin{aligned} z_1(t_0) &= z_1^{(0)}, \\ z_2(t_0) &= z_2^{(0)}, \\ &\vdots = \vdots, \\ z_n(t_0) &= z_n^{(0)}. \end{aligned}$$

We assume here that the state requires n variables $z_1, z_2, \ldots, z_n$ to describe.

As an example, consider the model described in class for two competing species. (See also Giordano and Wehr, Section 10.2 and Mesterton-Gibbons Section 1.5). The state of the system is specified by two variables x and y, which represent the populations of each of the species. The system of differential equations describing this model is:

$$\begin{aligned} \frac{dx}{dt} &= (a - by)x, \\ \frac{dy}{dt} &= (m - nx)y, \end{aligned} \tag{2.3a}$$

where a, b, m, and n are constant parameters which need to be specified externally

to the model, and we must supply some initial conditions of the form:

$$\begin{aligned} x(t_0) &= x_0, \\ y(t_0) &= y_0. \end{aligned} \tag{2.3b}$$

We don't have an explicit solution available for this system of differential equations. But we can ask MATLAB to give us an approximate solution with the ODE package `ode45`. Conceptually, it's much the same as for the case of a differential equation for a single state variable. But you have to be careful with the notation. The MATLAB ODE solver expects you to describe the state of the system with a *vector*. Therefore, to solve (2.3) with MATLAB, we should first define a state vector with components equal to the various state variables:

$$\begin{aligned} \mathbf{z} &= \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \\ z_1 &= x, \\ z_2 &= y. \end{aligned}$$

Re-expressing the system of differential equations (2.3) in terms of this state vector, we have:

$$\begin{aligned} \frac{d\mathbf{z}}{dt} = \mathbf{F}(t, \mathbf{z}) &= \begin{bmatrix} (a - bz_2)z_1, \\ (m - nz_1)z_2 \end{bmatrix}, \\ \mathbf{z}(t_0) &= \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \end{aligned} \tag{2.4}$$

which is in the form of (2.2).

Now we can write a function M-file which computes the right hand sides of the system of differential equations in the form of a vector $\mathbf{F}(t, \mathbf{z})$.

```
function F=compspec(t,z)
% Evaluates rates of change of populations of simple
% competing species model
% User should set parameters in USER PARAMETERS section
%
% dx/dt = a x - b x y
% dy/dt = m y - n x y
%
% Input variables:
%       t = time
%       z = state vector with components z(1)=x, z(2)=y,
%               with x and y representing populations of species
%
% Input parameters (external):
%       a, m = isolated per capita growth rates of each species
%       b, n = interaction parameters
%
% Output variables:
%       F = column vector with F(1) = dx/dt and F(2) = dy/dt

% USER PARAMETERS
a=3;
b=5/2;
m=2;
n=1;

% TRANSLATE STATE VECTOR TO MORE CONVENIENT VARIABLES
x=z(1);
y=z(2);

% SET UP F AS A COLUMN VECTOR WITH TWO ENTRIES
F = zeros(2,1);

% MAIN COMPUTATION
F(1)=a*x-b*x*y;
F(2)=m*y-n*x*y;

return
```

Now suppose we want to solve for the evolution of the competing species populations over the time interval $0 \leq t \leq 2.5$, starting with initial population sizes $x_0 = 0.2$ and $y_0 = 0.18$ (imagining that population size is measured in units of thousands, say). Then here is how MATLAB can be instructed to do this:

```
>> tspan=[0 2.5];
>> zo=[0.2 0.18];
>> [t,z] = ode45('compspec',tspan,zo);
```

The variable t is again a column vector storing the values of t which MATLAB used to discretize time. The variable z is now a matrix. The first column refers to the values of z_1 at the various times, and the second column refers to the values of z_2 at the various, and so forth.

There are a number of plots you can make using this information. Perhaps, we would like to plot x (which is the same as z_1) versus t (see Figure 2):

```
>> plot(t,z(:,1))
>> title('Population of First Species versus Time')
>> xlabel('t')
>> ylabel('x')
```

Or maybe we'd like to plot y (which is the same as z_2) versus t (see Figure 3):

```
>> plot(t,z(:,2))
>> title('Population of Second Species versus Time')
>> xlabel('t')
>> ylabel('y')
```

And we can plot all state variables simultaneously as a function of time (Figure 4):

```
>> plot(t,z)
>> title('Populations versus Time')
>> xlabel('t')
>> ylabel('Population size')
```

Finally, particularly for systems of two variables, it is often interesting to make a state space (or phase space) plot of one state variable versus another (Figure 5):

```
>> plot(z(:,1),z(:,2))
>> title('State Space Plot')
>> xlabel('x')
>> ylabel('y')
```

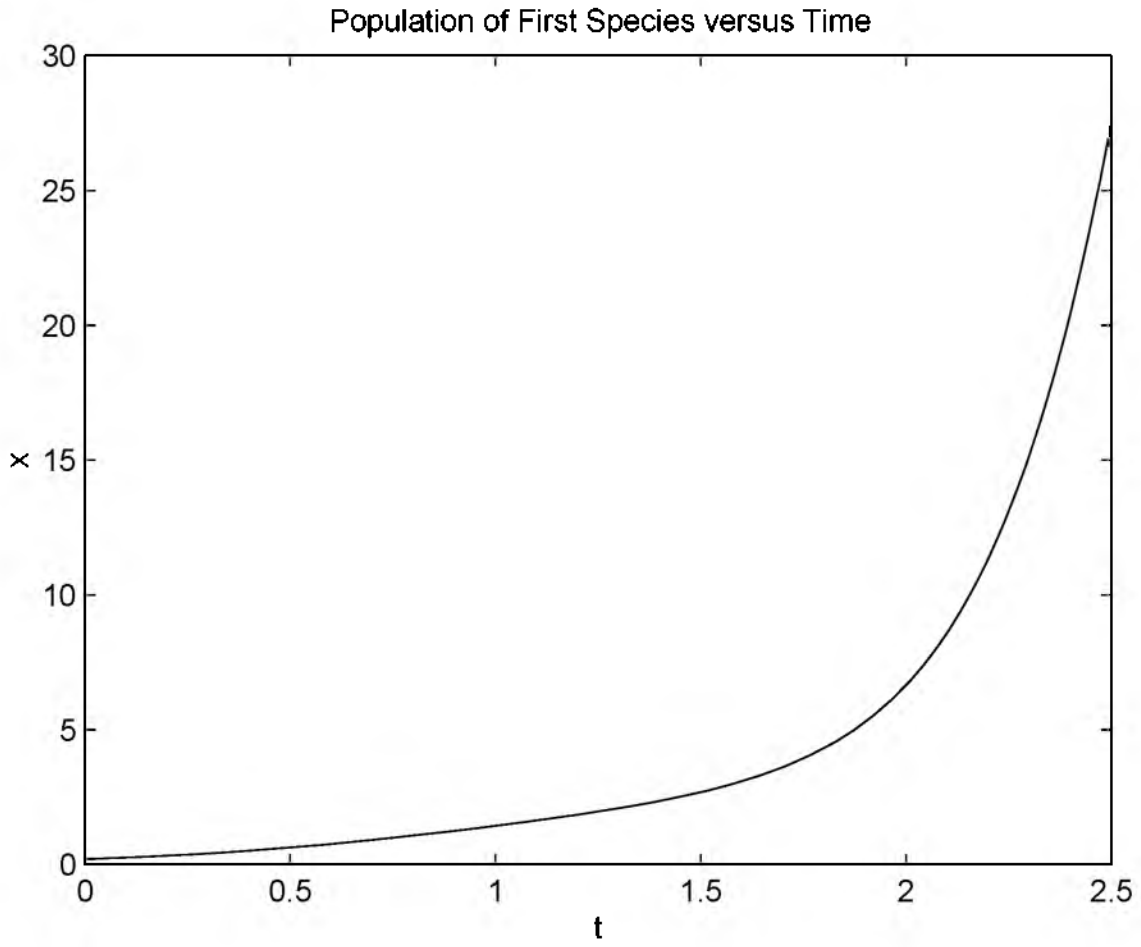

Figure 2: Solution of competing species model (2.3). Parameters: $a = 3$, $b = 5/2$, $m = 2$, $n = 1$. Initial condition: $x(0) = 0.2$, $y(0) = 0.18$.

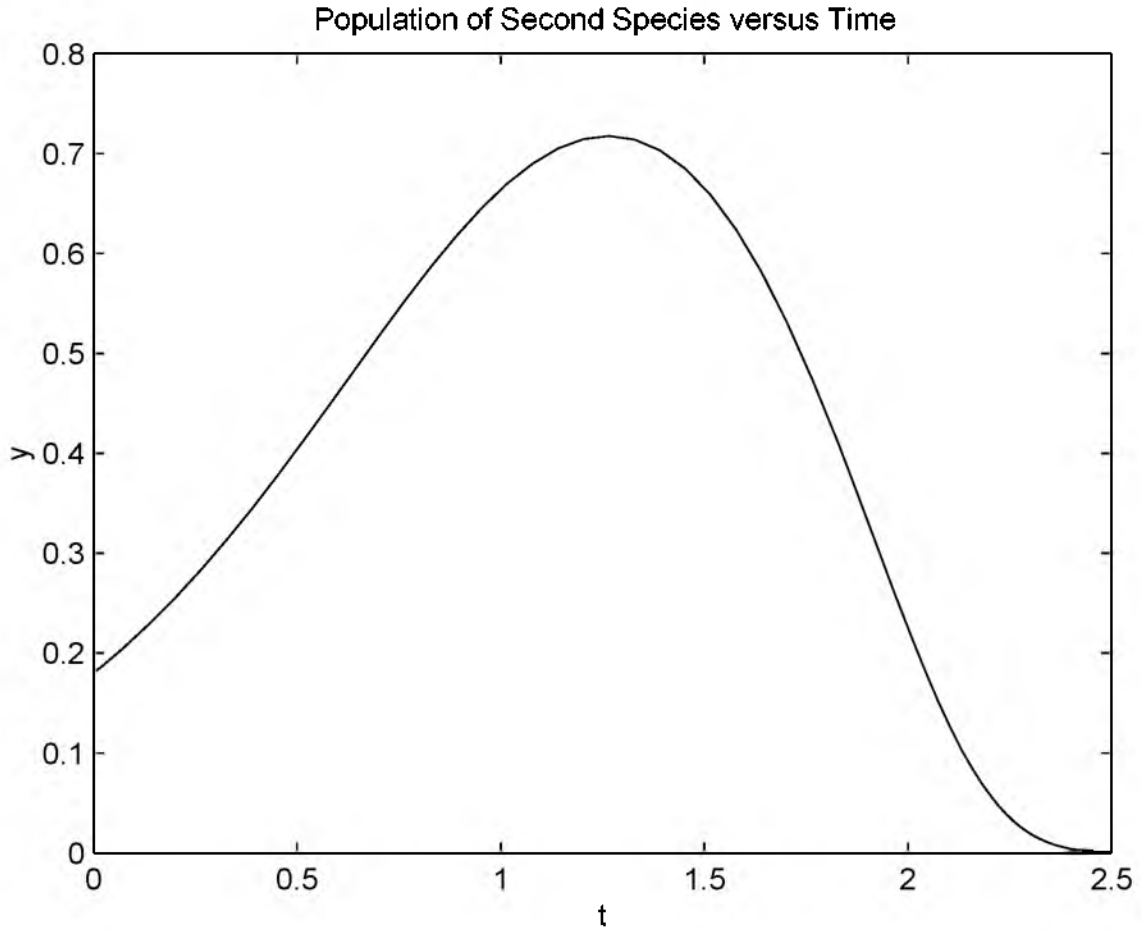

Figure 3: Solution of competing species model (2.3). Parameters: $a = 3$, $b = 5/2$, $m = 2$, $n = 1$. Initial condition: $x(0) = 0.2$, $y(0) = 0.18$.

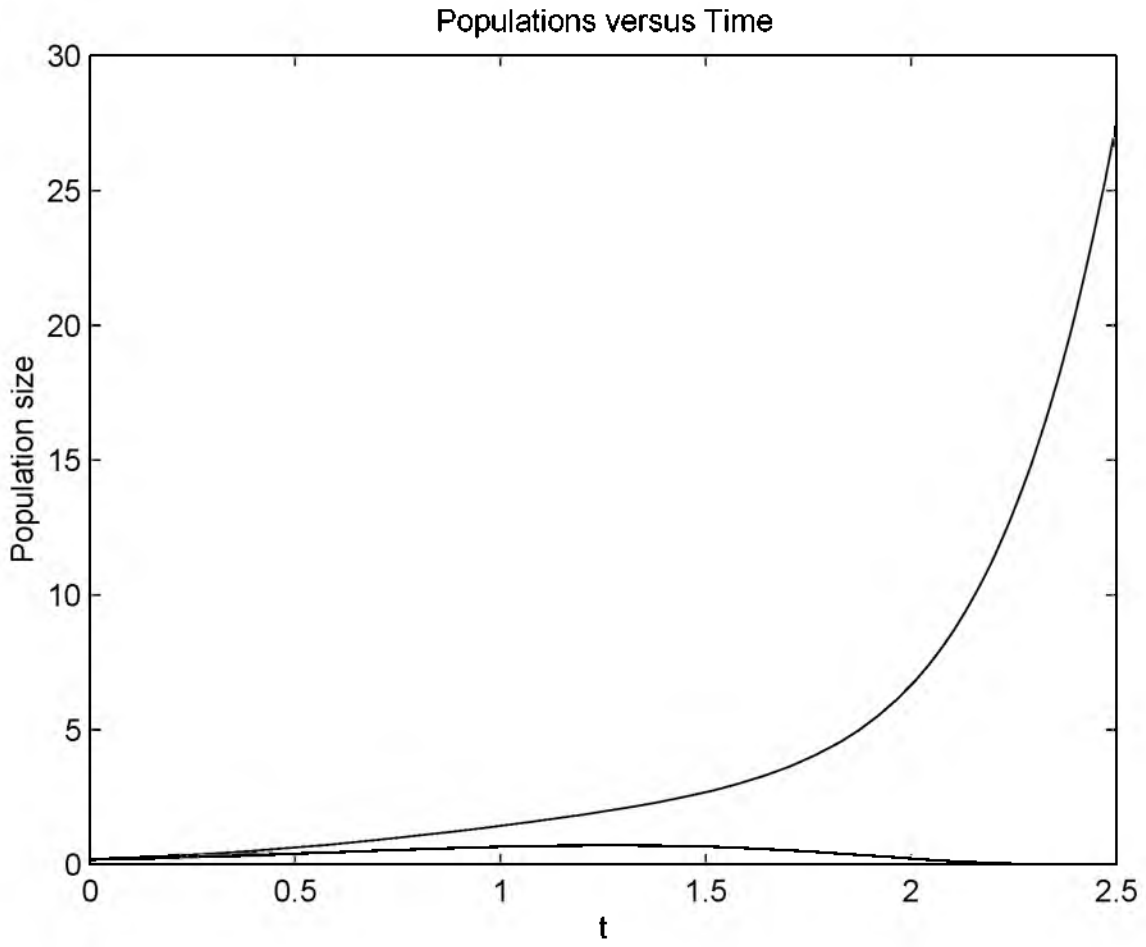

Figure 4: Solution of competing species model (2.3). Parameters: $a = 3$, $b = 5/2$, $m = 2$, $n = 1$. Initial condition: $x(0) = 0.2$, $y(0) = 0.18$.

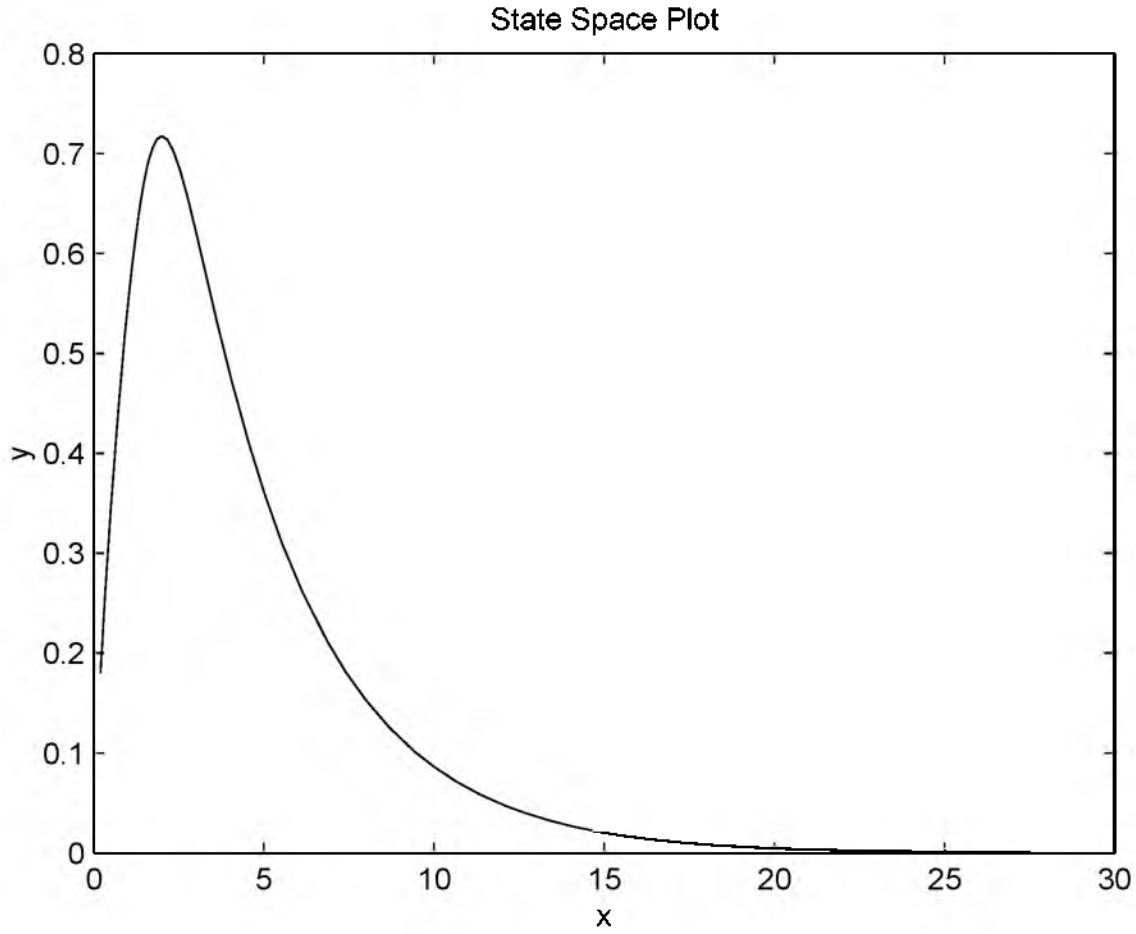

Figure 5: Solution of competing species model (2.3). Parameters: $a = 3$, $b = 5/2$, $m = 2$, $n = 1$. Initial condition: $x(0) = 0.2$, $y(0) = 0.18$.