12.2.2. Case Study: Pursuit Problem with Event Location

Next we consider a pursuit problem [12, Chap. 5]. Suppose that a rabbit follows a predefined path $(r_1(t), r_2(t))$ in the plane, and that a fox chases the rabbit in such a way that (a) at each moment the tangent of the fox's path points towards the rabbit and (b) the speed of the fox is some constant k times the speed of the rabbit. Then the path $(y_1(t), y_2(t))$ of the fox is determined by the ODE

$$\frac{d}{dt}y_1(t) = s(t)\,(r_1(t) - y_1(t))\,,$$
$$\frac{d}{dt}y_2(t) = s(t)\,(r_2(t) - y_2(t))\,,$$

where

$$s(t) = \frac{k\sqrt{\left(\frac{d}{dt}r_1(t)\right)^2 + \left(\frac{d}{dt}r_2(t)\right)^2}}{\sqrt{(r_1(t) - y_1(t))^2 + (r_2(t) - y_2(t))^2}}.$$

Note that this ODE system becomes ill-defined if the fox approaches the rabbit. We let the rabbit follow an outward spiral,

$$\begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix} = \sqrt{1+t}\begin{bmatrix} \cos t \\ \sin t \end{bmatrix},$$

and start the fox at $y_1(0) = 3, y_2(0) = 0$. The function `fox1` implements the ODE, with k set to 0.75:

```
function yprime = fox1(t,y)
%FOX1   Fox-rabbit pursuit simulation.
%       YPRIME = FOX1(T,Y).

k = 0.75;
r = sqrt(1+t)*[cos(t); sin(t)];
r_p =(0.5/sqrt(1+t))*[cos(t)-2*(1+t)*sin(t);sin(t)+2*(1+t)*cos(t)];
dist = norm(r-y);
if dist > 1e-4
   factor = k*norm(r_p)/dist;
   yprime = factor*(r-y);
else
   error('ODE model ill-defined.')
end
```

The `error` function (see Section 14.1) has been used so that execution terminates with an error message if the denominator of $s(t)$ in the ODE becomes too small. The script below calls `fox1` to produce Figure 12.6. Initial conditions are denoted by circles and the dashed and solid lines show the phase plane paths of the rabbit and fox, respectively.

```
tspan = [0 10]; yzero = [3;0];
[tfox,yfox] = ode45(@fox1,tspan,yzero);
plot(yfox(:,1),yfox(:,2)), hold on
plot(sqrt(1+tfox).*cos(tfox),sqrt(1+tfox).*sin(tfox),'--')
plot([3 1],[0 0],'o');
axis equal, axis([-3.5 3.5 -2.5 3.1])
legend('Fox','Rabbit',0), hold off
```

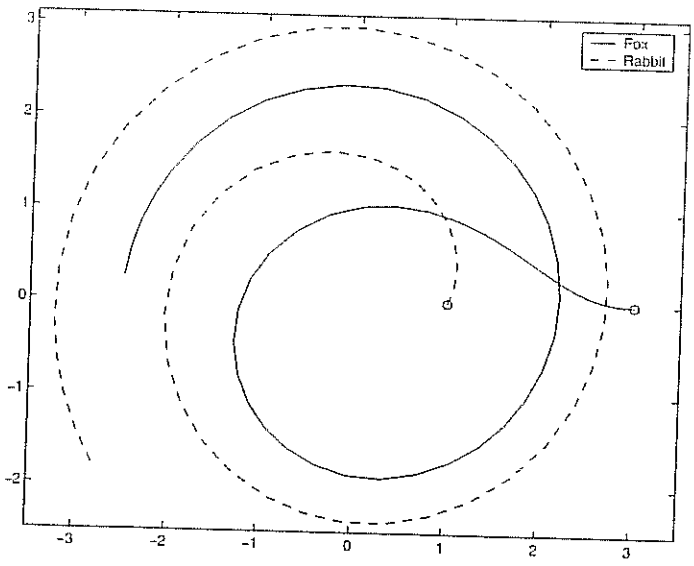

Figure 12.6. *Pursuit example.*

The implementation above is unsatisfactory for $k > 1$, that is, when the fox is faster than the rabbit. In this case, if the rabbit is caught within the specified time interval then no solution is displayed. It would be more natural to ask ode45 to return with the computed solution if the fox and rabbit become close. This can be done using the event location facility. The following script file uses the functions fox2 and events, which are given in Listing 12.1, to produce Figure 12.7. We have allowed k to be a parameter, and set k = 1.1 in the script file. The initial condition and the rabbit's path are as for Figure 12.6.

```
k = 1.1;
tspan = [0;10]; yzero = [3;0];
options = odeset('RelTol',1e-6,'AbsTol',1e-6,'Events',@events);
[tfox,yfox,te,ye,ie] = ode45(@fox2,tspan,yzero,options,k);
plot(yfox(:,1),yfox(:,2)), hold on
plot(sqrt(1+tfox).*cos(tfox),sqrt(1+tfox).*sin(tfox),'--')
plot([3 1],[0 0],'o'), plot(yfox(end,1),yfox(end,2),'*')
axis equal, axis([-3.5 3.5 -2.5 3.1])
legend('Fox','Rabbit',0), hold off
```

Here, we use odeset to set the event location property to the handle of the function events in Listing 12.1. This function has the three output arguments value, isterminal, and direction. It is the responsibility of ode45 to use events to check whether any component passes through zero by monitoring the quantity returned in value. In our example value is a scalar, corresponding to the distance between the rabbit and fox, minus a threshold of 10^{-4}. Hence, ode45 checks if the fox has approached within distance 10^{-4} of the rabbit. We set direction = -1, which signifies that value must be decreasing through zero in order for the event to be considered. The alternative choice direction = 1 tells MATLAB to consider only crossings where value is increasing, and direction = 0 allows for any type of

Listing 12.1. *Functions* fox2 *and* events.

```
function yprime = fox2(t,y,k)
%FOX2   Fox-rabbit pursuit simulation with relative speed parameter.
%       YPRIME = FOX2(T,Y,K).

r = sqrt(1+t)*[cos(t); sin(t)];
r_p = (0.5/sqrt(1+t)) * [cos(t)-2*(1+t)*sin(t); sin(t)+2*(1+t)*cos(t)];
dist = max(norm(r-y),1e-6);
factor = k*norm(r_p)/dist;
yprime = factor*(r-y);
```

```
function [value,isterminal,direction] = events(t,y,k)
%EVENTS    Events function for FOX2.
%          Locate when fox is close to rabbit.

r = sqrt(1+t)*[cos(t); sin(t)];
value = norm(r-y) - 1e-4;      % Fox close to rabbit.
isterminal = 1;                % Stop integration.
direction = -1;                % Value must be decreasing through zero.
```

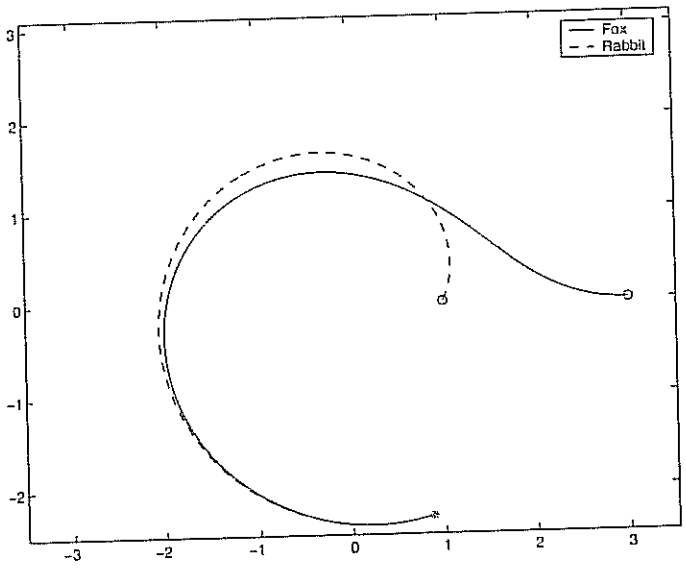

Figure 12.7. *Pursuit example, with capture.*

zero. Since we set `isterminal = 1`, integration will cease when a suitable zero crossing is detected. With the other option, `isterminal = 0`, the event is recorded and the integration continues. Note that the function `events` must accept the additional parameter `k` passed to `ode45`, even though it does not need it in this example.

The output arguments from `ode45` are `[tfox,yfox,te,ye,ie]`. Here, `tfox` and `yfox` are the usual solution approximations, so `yfox(i)` approximates $y(t)$ at time $t =$ `tfox(i)`. The arguments `te` and `ye` record those t and y values at which the event(s) were recorded and, for vector valued events, `ie` specifies which component of the event occurred each time. (If no events are detected then `te`, `ye` and `ie` are returned as empty matrices.) In our example, we have

```
>> te, ye
te =
    5.0710
ye =
    0.8646   -2.3073
```

showing that the rabbit was captured after 5.07 time units at the point $(0.86, -2.31)$.

12.2.3. Stiff Problems and the Choice of Solver

The Robertson ODE system

$$\begin{aligned}
\frac{d}{dt}y_1(t) &= -0.04y_1(t) + 10^4 y_2(t)y_3(t),\\
\frac{d}{dt}y_2(t) &= 0.04y_1(t) - 10^4 y_2(t)y_3(t) - 3\times 10^7 y_2(t)^2,\\
\frac{d}{dt}y_3(t) &= 3\times 10^7 y_2(t)^2
\end{aligned}$$

models a reaction between three chemicals [25, p. 3], [69, p. 418]. We set the system up as the function `chem`:

```
function yprime = chem(t,y)
%CHEM    Robertson's chemical reaction model.
%        YPRIME = CHEM(T,Y).

yprime = [-0.04*y(1) + 1e4*y(2)*y(3);
           0.04*y(1) - 1e4*y(2)*y(3) - 3e7*y(2)^2;
           3e7*y(2)^2];
```

The script file below solves this ODE for $0 \le t \le 3$ with initial condition $[1; 0; 0]$, first using `ode45` and then using another solver, `ode15s`, which is based on implicit linear multistep methods. (Implicit means that a nonlinear equation must be solved at each step.) The results for $y_2(t)$ are plotted in Figure 12.8.

```
tspan = [0 3]; yzero = [1;0;0];
[ta,ya] = ode45(@chem,tspan,yzero);
subplot(121), plot(ta,ya(:,2),'-*')
ax = axis; ax(1) = -0.2; axis(ax) % Make initial transient clearer.
xlabel('t'), ylabel('y_2(t)'), title('ode45','FontSize',14)
[tb,yb] = ode15s(@chem,tspan,yzero);
subplot(122), plot(tb,yb(:,2),'-*'), axis(ax)
xlabel('t'), ylabel('y_2(t)'), title('ode15s','FontSize',14)
```