Chapter 1. The Tractrix and Similar Curves

W. Gander, S. Bartoň, and J. Hřebíček

1.1 Introduction

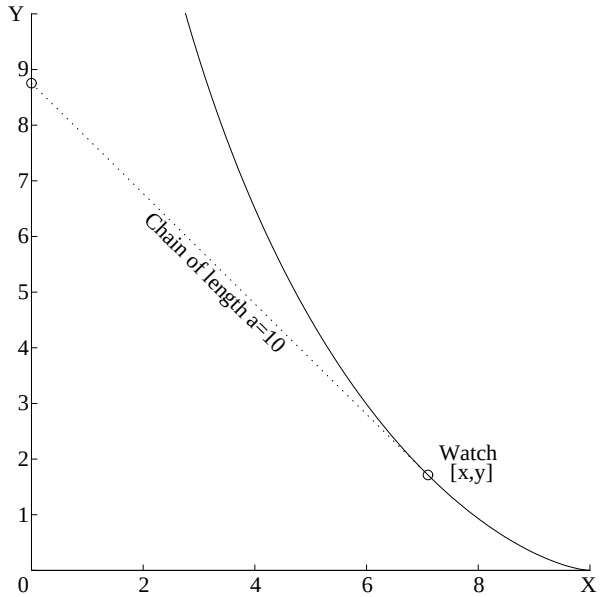
In this section we will use MATLAB to solve two similar systems of differential equations. First we generalize the classical tractrix problem to compute the orbit of a toy pulled by a child, and then we compute the orbit of a dog which attacks a jogger. We also show how the motions may be visualized with MATLAB.

1.2 The Classical Tractrix

In the 17th century Gottfried Wilhelm Leibniz discussed the following problem, see [2, 1]. *Given a watch attached to a chain, what is the orbit in the plane described by the watch as the endpoint of the chain is pulled along a straight line?*

Let a be the length of the chain. The problem is easily solved if we assume that the point-like watch is initially on the x-axis at the point $(a, 0)$, and that starting at the origin we pull in the direction of the positive y-axis, [2], (cf. Figure 1.1).

FIGURE 1.1. *Classical Tractrix.*



From Figure 1.1 we immediately obtain the following differential equation for the unknown function $y(x)$:

$$y' = -\frac{\sqrt{a^2 - x^2}}{x}. \tag{1.1}$$

To solve Equation (1.1) we only need to integrate:

```
> y:= -Int(sqrt(a^2-x^2)/x, x) + c;
```

$$y := -\int \frac{\sqrt{a^2 - x^2}}{x} dx + c$$

```
> y:=simplify(value(y), symbolic);
```

$$y := -\sqrt{a^2 - x^2} + a\ln(2) + a\ln(a) + a\ln(a + \sqrt{a - x}\,\sqrt{a + x}) - a\ln(x) + c \tag{1.2}$$

MAPLE does not include the constant when performing indefinite integration. So we added an integration constant c. We can determine its value by using the initial condition $y(a) = 0$:

```
> c:= solve(subs(x=a, y), c);
```

$$c := -a\ln(2) - a\ln(a)$$

Therefore the solution to our problem is

```
> y:= combine(y,ln, symbolic);
```

$$y := -\sqrt{a^2 - x^2} + a\ln\left(\frac{a + \sqrt{a^2 - x^2}}{x}\right).$$

Let us now assume that the object to be pulled is initially on the y-axis at the point $(0, a)$ and that we start pulling again at the origin, but this time in the direction of the positive x-axis.

Consider the point $(x, y(x))$ on the orbit of the object. The endpoint of the chain on the x-axis is at the point $(x - y(x)/y'(x), 0)$, that is where the tangent intersects the x-axis (this is the same point which would be obtained for one step of Newton's iteration!). Therefore, the condition that the chain has the constant length a, leads to the differential equation

$$\frac{y(x)^2}{y'(x)^2} + y(x)^2 = a^2, \tag{1.3}$$

which can no longer be solved directly by quadrature. Therefore we need to call the differential equation solver `dsolve`,

```
> unassign('y');
> eq := (y(x)/diff(y(x), x))^2 + y(x)^2 = a^2;
```

$$eq := \frac{\mathrm{y}(x)^2}{(\frac{\partial}{\partial x}\,\mathrm{y}(x))^2} + \mathrm{y}(x)^2 = a^2$$

```
> p := [dsolve(eq, y(x))]:
> p1:=simplify(p,symbolic);
```

$$
\begin{aligned}
p1 := \Big[& x - \sqrt{-\mathrm{y}(x)^2 + a^2} + a\ln(2) + a\ln(a) \\
& + a\ln\Big(a + \sqrt{a - \mathrm{y}(x)}\,\sqrt{a + \mathrm{y}(x)}\Big) - a\ln(\mathrm{y}(x))\Big) - \_C1 = 0, \\
& x + \sqrt{-\mathrm{y}(x)^2 + a^2} - a\ln(2) - a\ln(a) \\
& - a\ln\Big(a + \sqrt{a - \mathrm{y}(x)}\,\sqrt{a + \mathrm{y}(x)}\Big) + a\ln(\mathrm{y}(x))\Big) - \_C1 = 0\Big]
\end{aligned}
$$

and we obtain two solutions. Because of the initial condition

```
> p2:=subs({x=0,y(x)=a},p1);
```

$$p2 := [\, a\ln(2) + a\ln(a) - \_C1 = 0,\ -a\ln(2) - a\ln(a) - \_C1 = 0\,]$$

we obtain two equations which we solve for the two constants:

```
> p3:=map(u->solve(u,_C1),p2);
```

$$p3 := [\, a\ln(2) + a\ln(a),\ -a\ln(2) - a\ln(a)\,]\,.$$

The correct answer to our problem is the function with $y(x) > 0$ and $y'(x) < 0$. This is the one with the same integration constant c as before. We choose this solution by comparing the two lists (solutions and constants) with the `zip` function:

```
> zip((u,v)->`if`(u=c,subs(_C1=c,v),NULL),p3,p1)[];
```

$$x - \sqrt{-\mathrm{y}(x)^2 + a^2} + a\ln\Big(a + \sqrt{a - \mathrm{y}(x)}\,\sqrt{a + \mathrm{y}(x)}\Big) - a\ln(\mathrm{y}(x))\Big) = 0$$

and obtain an equation for the solution $y(x)$.

We could, of course, have obtained this equation also by interchanging the variables x and y in Equation (1.2). Note that it would be difficult to solve Equation (1.3) numerically, since for $x = 0$ we have the singularity $y'(0) = \infty$.

1.3 The Child and the Toy

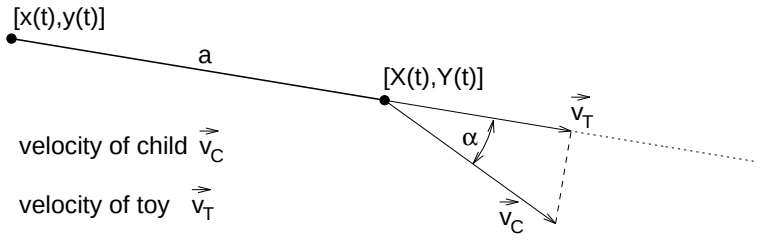
Let us now solve a more general problem and suppose that a child is walking on the plane along a curve given by the two functions of time $X(t)$ and $Y(t)$.

Suppose now that the child is pulling or pushing some toy, by means of a rigid bar of length a. We are interested in computing the orbit of the toy when the child is walking around. Let $(x(t), y(t))$ be the position of the toy. From Figure 1.2 the following equations are obtained:

1. The distance between the points $(X(t), Y(t))$ and $(x(t), y(t))$ is always the length of the bar. Therefore

$$(X - x)^2 + (Y - y)^2 = a^2. \tag{1.4}$$

FIGURE 1.2. *Velocities* $\mathbf{v}_C$ *and* $\mathbf{v}_T$.



2. The toy is always moving in the direction of the bar. Therefore the difference vector of the two positions is a multiple of the velocity vector of the toy, $\mathbf{v}_T = (\dot{x}, \dot{y})^T$:

$$\begin{pmatrix} X - x \\ Y - y \end{pmatrix} = \lambda \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \quad \text{with} \quad \lambda > 0. \tag{1.5}$$

3. The speed of the toy depends on the direction of the velocity vector $\mathbf{v_C}$ of the child. Assume, e.g., that the child is walking on a circle of radius a (length of the bar). In this special case the toy will stay at the center of the circle and will not move at all (this is the final state of the first numerical example, see Figure 1.3).

   From Figure 1.2 we see that *the modulus of the velocity* $\mathbf{v_T}$ *of the toy is given by the modulus of the projection of the velocity* $\mathbf{v_C}$ *of the child onto the bar.*

Inserting Equation (1.5) into Equation (1.4), we obtain

$$a^2 = \lambda^2(\dot{x}^2 + \dot{y}^2) \quad \longrightarrow \quad \lambda = \frac{a}{\sqrt{\dot{x}^2 + \dot{y}^2}}.$$

Therefore

$$\frac{a}{\sqrt{\dot{x}^2 + \dot{y}^2}} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} X - x \\ Y - y \end{pmatrix}. \tag{1.6}$$

We would like to solve Equation (1.6) for $\dot{x}$ and $\dot{y}$. Since we know the modulus of the velocity vector of the toy $|\mathbf{v}_T| = |\mathbf{v}_C| \cos\alpha$, see Figure 1.2, this can be done by the following steps:

- Normalize the difference vector $(X - x, Y - y)^T$ and obtain a vector $\mathbf{w}$ of unit length.
- Determine the projection of $\mathbf{v}_C = (\dot{X}, \dot{Y})^T$ onto the subspace generated by $\mathbf{w}$. This is simply the scalar product $\mathbf{v}_C^T\mathbf{w}$, since $\mathbf{v}_C^T\mathbf{w} = |\mathbf{v}_C||\mathbf{w}| \cos\alpha$ and $|\mathbf{w}| = 1$.
- $\mathbf{v}_T = (\dot{x}, \dot{y})^T = (\mathbf{v}_C^T\mathbf{w})\mathbf{w}$.

Now we can write the function to evaluate the system of differential equations in MATLAB (see Algorithm 1.1).

ALGORITHM 1.1. *Function f.*

```
function zs = f(t,z)
%
[X Xs Y Ys] = child(t);
v =[Xs; Ys];
w =[X-z(1); Y-z(2)];
w = w/norm(w);
zs = (v'*w)*w;
```

The function `f` calls the function `child` which returns the position $(X(t), Y(t))$ and velocity of the child $(Xs(t), Ys(t))$ for a given time `t`. As an example consider a child walking on the circle $X(t) = 5\cos t; Y(t) = 5\sin t$. The corresponding function `child` for this case is:

ALGORITHM 1.2. *Function Child.*

```
function [X, Xs, Y, Ys] = child(t);
%
X  =  5*cos(t);  Y  =  5*sin(t);
Xs = -5*sin(t);  Ys =  5*cos(t);
```

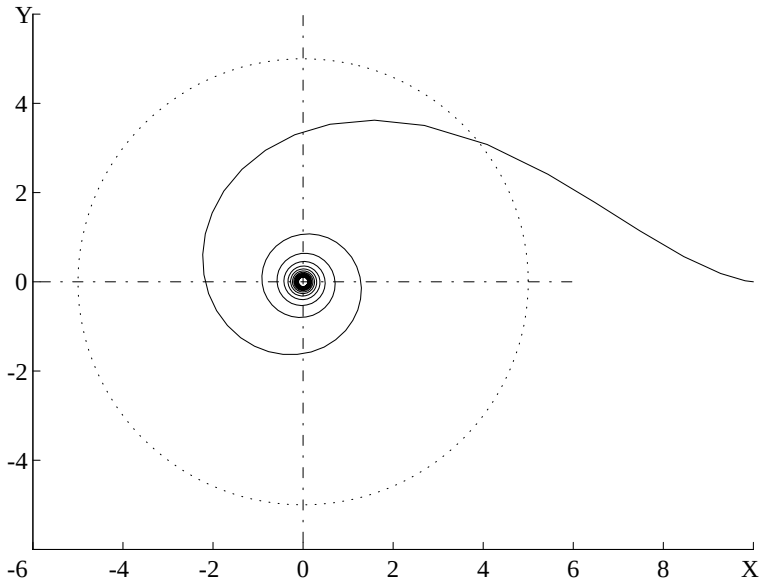
MATLAB offers two M-files `ode23` and `ode45` to integrate differential equations. In the following main program we will call one of these functions and also define the initial conditions (Note that for $t = 0$ the child is at the point $(5, 0)$ and the toy at $(10, 0)$):

```
>> % main1.m
>> y0 = [10 0]';
>> [t y] = ode45('f',[0 100],y0);
>> clf; hold on;
>> axis([-6 10 -6 10]);
>> axis('square');
>> plot(y(:,1),y(:,2));
```

If we plot the two columns of y we obtain the orbit of the toy (cf. Figure 1.3). Furthermore we add the curve of the child in the same plot with the statements:

```
>> t = 0:0.05:6.3
>> [X, Xs, Y, Ys] = child(t);
>> plot(X,Y,':')
>> hold off;
```

FIGURE 1.3. *Child Walks on the Circle.*



Note that the length of the bar a does not appear explicitly in the programs; *it is defined implicitly by the position of the toy, (initial condition), and the position of the child (function* `child`*) for* $t = 0$.

We conclude this section with some more examples. Let the child be walking along the graph of a sine function: $X(t) = t$ and $Y(t) = 5\sin t$. The child's curve is again plotted with a dotted line. With the initial conditions $x(0) = 0$ and $y(0) = 10$ we obtain Figure 1.4.

In the next example, the child is again walking on the circle $X(t) = 5\cos t$, $Y(t) = 5\sin t$. With the initial condition $x(0) = 0$ and $y(0) = 10$, we obtain a nice flower-like orbit of the toy (cf. Figure 1.5).

1.4 The Jogger and the Dog

We consider the following problem: a jogger is running along his favorite trail on the plane in order to get his daily exercise. Suddenly, he is being attacked by a dog. The dog is running with constant speed w towards the jogger. Compute the orbit of the dog.

The orbit of the dog has the property that the velocity vector of the dog points at every time to its goal, the jogger. We assume that the jogger is running on some trail and that his motion is described by the two functions $X(t)$ and $Y(t)$.
Let us assume that for $t = 0$ the dog is at the point (x_0, y_0), and that at time t his position will be $(x(t), y(t))$. The following equations hold:

1. $\dot{x}^2 + \dot{y}^2 = w^2$: The dog is running with constant speed.

2. The velocity vector of the dog is parallel to the difference vector between

FIGURE 1.4. *Example 2.*

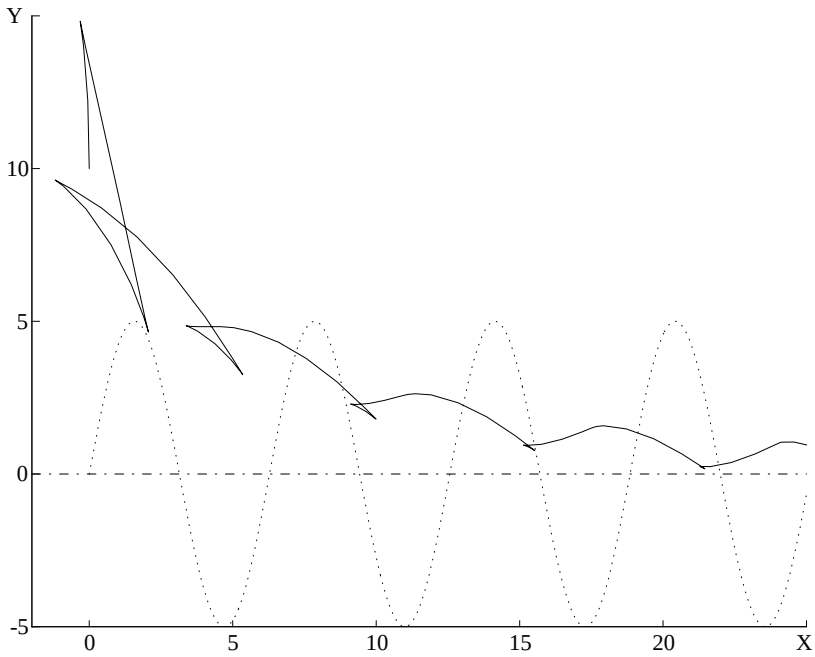
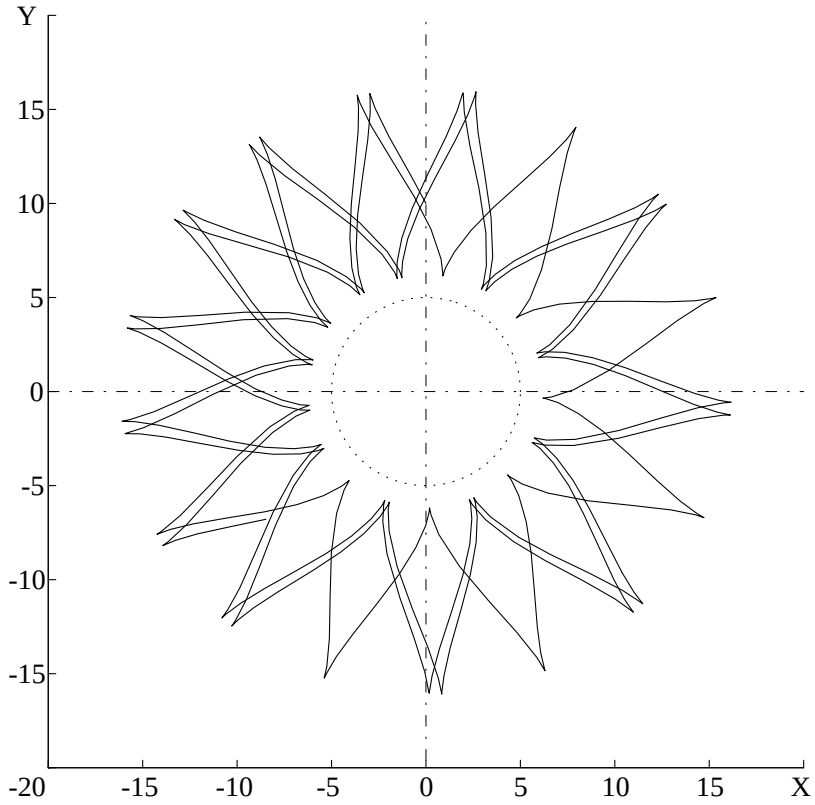


FIGURE 1.5. *Flower Orbit.*



the position of the jogger and the dog:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \lambda \begin{pmatrix}X-x\\ Y-y\end{pmatrix} \quad \text{with } \lambda > 0.$$

If we substitute this in the first equation we obtain

$$w^2 = \dot{x}^2 + \dot{y}^2 = \lambda^2 \left\| \begin{pmatrix}X-x\\ Y-y\end{pmatrix} \right\|^2 .$$

This equation can be solved for λ:

$$\lambda = \frac{w}{\left\| \begin{pmatrix}X-x\\ Y-y\end{pmatrix} \right\|} > 0.$$

Finally, substitution of this expression for λ in the second equation yields the differential equation of the orbit of the dog:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \frac{w}{\left\| \begin{pmatrix}X-x\\ Y-y\end{pmatrix} \right\|} \begin{pmatrix}X-x\\ Y-y\end{pmatrix}. \tag{1.7}$$

Again we will make use of one of the M-files `ode23.m` or `ode45.m` to integrate the system of differential equations. We notice that the system (1.7) has a singularity when the dog reaches the jogger. In this case the norm of the difference vector becomes zero and we have to stop the integration. The above mentioned MATLAB functions for integrating differential equations require as input an interval of the independent variable. MATLAB provides also the possibility to define another termination criterion for the integration, different from a given upper bound for the independent variable. It is possible to terminate the integration by checking zero crossings of a function. In our example one would like to terminate integration when the dog reaches the jogger, i.e. when $\left\|(X-x, Y-y)^T\right\|$ becomes small. In order to do so we have to add a third input and two more output parameters to the M-function `dog.m` (see Algorithm 1.3). The integrator `ode23` or `ode45` calls the function in two ways: The first one consists of dropping the third parameter. The function then returns only the parameter `zs`: the speed of the dog. In the second way the keyword `'events'` is assigned to the parameter `flag`. This keyword tells the function to return the zero-crossing function in the first output `zs`. The second output `isterminal` is a logical vector that tells the integrator, which components of the first output force the procedure to stop when they become zero. Every component with this property is marked with a nonzero entry in `isterminal`. The third output parameter `direction` is also a vector that indicates for each component of `zs` if zero crossings shall only be regarded for increasing values (`direction = 1`), decreasing values (`direction = -1`) or in both cases (`direction = 0`). The condition for zero crossings is checked in the integrator. The speed w of the dog must be declared global in `dog` and in the main program. The orbit of the jogger is given by the M-function `jogger.m`.

ALGORITHM 1.3. *Function Dog.*

```
function [zs,isterminal,direction] = dog(t,z,flag);
%
global w  % w = speed of the dog
X= jogger(t);
h= X-z;
nh= norm(h);
if nargin < 3 | isempty(flag) % normal output
   zs= (w/nh)*h;
else
   switch(flag)
   case 'events'  % at norm(h)=0 there is a singularity
      zs= nh-1e-3;   % zero crossing at pos_dog=pos_jogger
      isterminal= 1; % this is a stopping event
      direction= 0;  % don't care if decrease or increase
   otherwise
      error(['Unknown flag: ' flag]);
   end
end
```

The main program `main2.m` defines the initial conditions and calls `ode23` for the integration. We have to provide an upper bound of the time t for the integration.

```
>> % main2.m
>> global w
>> y0 = [60;70];  % initial conditions, starting point of the dog
>> w = 10;        % w  speed of the dog
>> options= odeset('RelTol',1e-5,'Events','on');
>> [t,Y] = ode23('dog',[0,20],y0,options);
>> clf; hold on;
>> axis([-10,100,-10,70]);
>> plot(Y(:,1),Y(:,2));
>> J=[];

>> for h= 1: length(t),
>>    w  = jogger(t(h));
>>    J = [J; w'];
>> end;
>> plot(J(:,1), J(:,2),':','Color','red');
```

The integration will stop either if the upper bound for the time t is reached or if the dog catches up with the jogger. For the latter case we set the flag `Events` of the ODE options to `'on'`. This tells the integrator to check for zero crossings of the function `dog` called with `flag = 'events'`. After the call to `ode23` the variable `Y` contains a table with the values of the two functions $x(t)$ and $y(t)$. We plot the orbit of the dog simply by the statement `plot(Y(:,1),Y(:,2))`.

In order to show also the orbit of the jogger we have to recompute it using the vector t and the function `jogger`.

Let us now compute a few examples. First we let the jogger run along the x-axis:

ALGORITHM 1.4. *First Jogger Example.*

```
function s = jogger(t);
s   = [8*t; 0];
```

In the above main program we chose the speed of the dog as $w = 10$, and since here we have $X(t) = 8t$ the jogger is slower. As we can see in Figure 1.6 the dog is catching the poor jogger.

If we wish to indicate the position of the jogger's troubles, *(perhaps to build a small memorial)*, we can make use of the following file `cross.m`

ALGORITHM 1.5. *Drawing a Cross.*

```
function cross(Cx,Cy,v)
% draws at position  Cx,Cy  a cross of height 2.5v
% and width 2*v
Kx = [Cx Cx Cx Cx-v Cx+v];
Ky = [Cy Cy+2.5*v Cy+1.5*v Cy+1.5*v Cy+1.5*v];
plot(Kx,Ky);
plot(Cx,Cy,'o');
```

The cross in the plot was generated by appending the statements to the main program.

```
>> p = max(size(Y));
>> cross(Y(p,1),Y(p,2),2)
>> hold off;
```

The next example shows the situation where the jogger turns around and tries to run back home:

ALGORITHM 1.6. *Second Jogger Example.*

```
function s = jogger1(t);
%
if t<6, s = [8*t; 0];
else    s = [8*(12-t) ;0];
end
```

However, using the same main program as before the dog catches up with the jogger at time $t = 9.3$ (cf. Figure 1.7).

FIGURE 1.6. *Jogger Running on the Line $y = 0$.*

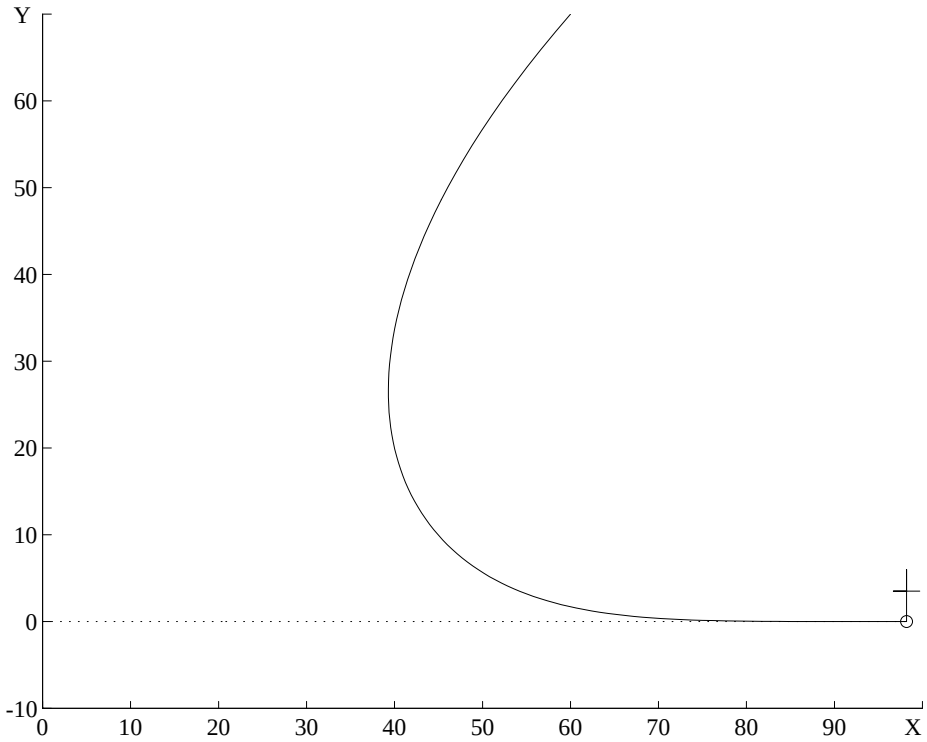
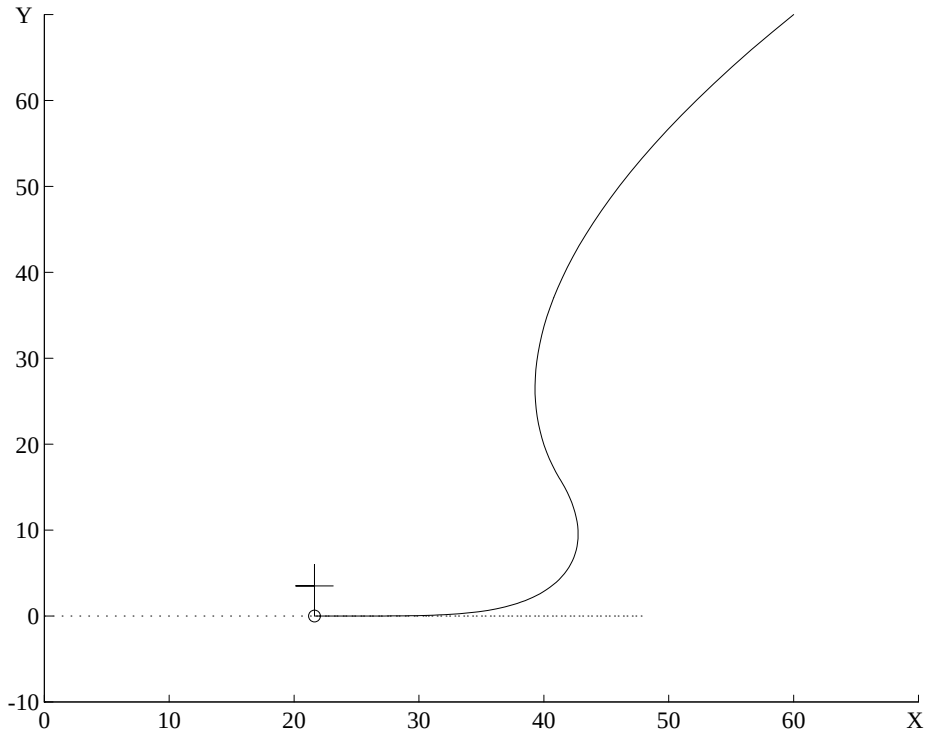


FIGURE 1.7. *Jogger Returning Back.*



Let us now consider a faster jogger running on an ellipse

ALGORITHM 1.7. *Third Jogger Example.*

```
function s = jogger2(t);
s   = [  10+20*cos(t)
       20 + 15*sin(t)];
```

If the dog also runs fast ($w = 19$), he manages to reach the jogger at time $t = 8.97$ (cf. Figure 1.8). We finally consider an old, slow dog ($w = 10$). He tries to catch a jogger running on a elliptic track. However, instead of waiting for the jogger somewhere on the ellipse, he runs (too slow) after his target, and we can see a steady state developing where the dog is running on a closed orbit inside the ellipse (cf. Figure 1.9).

1.5 Showing the Motions with MATLAB

It would be nice to show simultaneously the motions of the child and the toy or the dog and the jogger instead of just plotting statically their orbits. This is possible using the *handle graphics* commands in MATLAB. The main program for the child and its toy now looks as follows:

```
>> % main3.m
>> y0 = [0 20]';
>> options= odeset('RelTol',1e-10);
>> [t y] = ode45 ('f', [0 40], y0, options);
>> [X, Xs, Y, Ys] = child (t);

>> xmin = min (min (X), min (y (:, 1)));
>> xmax = max (max (X), max (y (:, 1)));
>> ymin = min (min (Y), min (y (:, 2)));
>> ymax = max (max (Y), max (y (:, 2)));

>> clf; hold on;
>> axis ([xmin xmax ymin ymax]);
>> % axis('equal');
>> title ('The Child and the Toy.');
>> stickhandle = line ('Color', 'yellow', 'EraseMode', 'xor', ...
>>                'LineStyle', '-', 'XData', [], 'YData', []);

>> for k = 1:length(t)-1,
>>   plot ([X(k), X(k+1)], [Y(k), Y(k+1)], '-', ...
>>         'Color', 'yellow', 'EraseMode', 'none');
>>   plot ([y(k,1), y(k+1,1)], [y(k,2), y(k+1,2)], '-', ...
>>         'Color', 'green', 'EraseMode', 'none');
```

FIGURE 1.8. *Jogger on an Ellipse.*

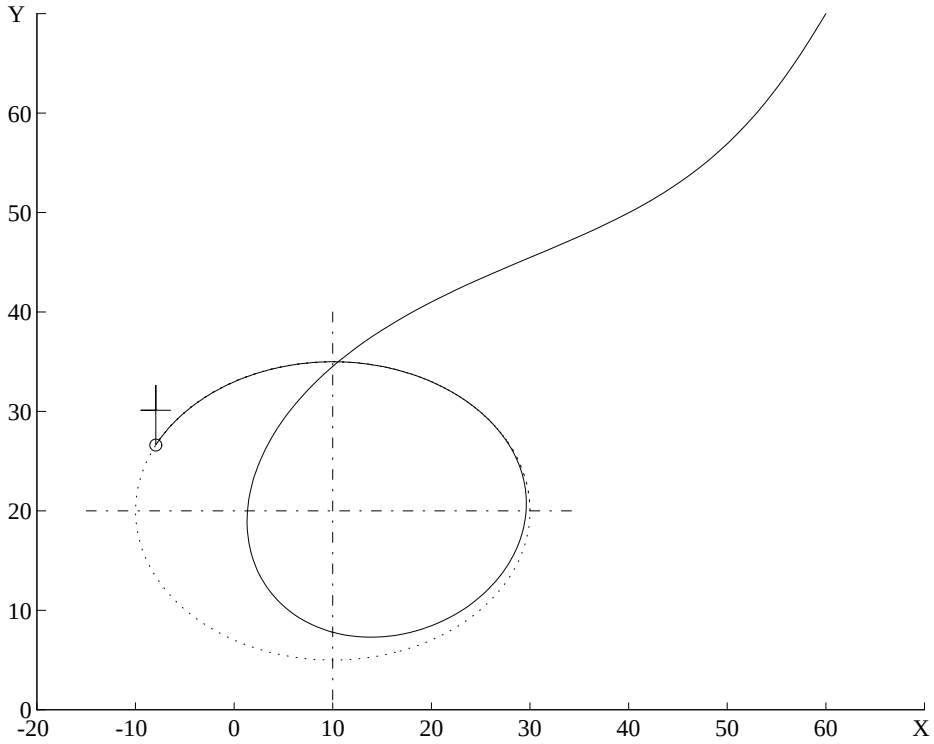
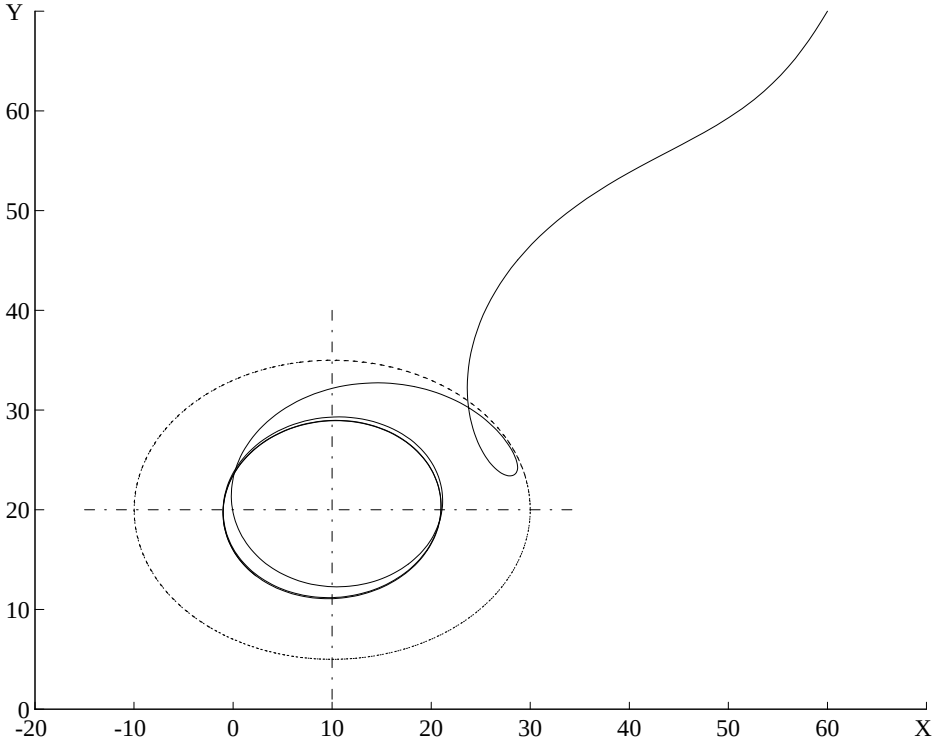


FIGURE 1.9. *Slow Dog.*



```
>>   set (stickhandle, 'XData', [X(k+1), y(k+1,1)], ...
>>        'YData', [Y(k+1), y(k+1,2)]);
>>   drawnow;
>> end;
>> hold off;
```

We define the variable `stickhandle` as a handle to a graphical object of type *line* associated with the stick. In the loop, we draw new segments of the child and toy orbits and move the position of the stick. The `drawnow` command forces these objects to be plotted instantaneously. Therefore, we can watch the two orbits and the stick being plotted simultaneously.

In the case of the jogger and the dog we do not even have to define a handle. All we have to do is to draw the segments of the two orbits in the proper sequence:

```
>> % main4.m
>> global w;
>> y0 = [60; 70];  % initial conditions, starting point of the dog
>> w = 10;         % w  speed of the dog
>> options= odeset('RelTol',1e-5,'Events','on');
>> [t,Y] = ode23 ('dog', [0 20], y0, options);

>> J=[];
>> for h= 1:length(t),
>>   w = jogger(t(h));
>>   J = [J; w'];
>> end

>> xmin = min (min (Y (:, 1)), min (J (:, 1)));
>> xmax = max (max (Y (:, 1)), max (J (:, 1)));
>> ymin = min (min (Y (:, 2)), min (J (:, 2)));
>> ymax = max (max (Y (:, 2)), max (J (:, 2)));
>> clf; hold on;
>> axis ([xmin xmax ymin ymax]);
>> % axis ('equal');
>> title ('The Jogger and the Dog.');

>> for h=1:length(t)-1,
>>   plot ([Y(h,1), Y(h+1,1)] , [Y(h,2), Y(h+1,2)], '-', ...
>>         'Color', 'yellow', 'EraseMode','none');
>>   plot ([J(h,1), J(h+1,1)] , [J(h,2), J(h+1,2)], ':', ...
>>        'Color', 'green', 'EraseMode','none');
>>   drawnow;
>>   pause(1);
>> end
>> hold off;
```

1.6 Jogger with Constant Velocity

We continue in this section with the elliptical orbit of the jogger. If we describe the ellipse as in Algorithm 1.7 and consider t as the time variable the velocity of the jogger will not be constant. Let $s(t)$ be the parameter for the description of the ellipse with center in (m_1, m_2) and main semi-axes a and b:

$$X(s) = m_1 + a\cos(s), \quad Y(s) = m_2 + b\sin(s)$$

We want to determine s as a monotonic increasing function of the time t such that for equidistant times t_i the points on the ellipse $X(s(t_i)), Y(s(t_i))$ are also equidistant on the border of the ellipse. An equivalent condition is that the velocity of the jogger $V(t)$ is constant:

$$V = \sqrt{\left(\frac{d\,\mathrm{X}(s(t))}{d\,t}\right)^2 + \left(\frac{d\,\mathrm{Y}(s(t))}{d\,t}\right)^2} = \text{const.} \tag{1.8}$$

Now $s(t)$ can be computed by first solving Equation (1.8) for ds/dt:

```
> restart;
> Xe := m1 + a*cos(s(t)): Ye := m2 + b*sin(s(t)):
> Ve2 := diff(Xe, t)^2 +diff(Ye, t)^2 =V^2;
```

$$Ve2 := a^2\left(\sin\left(s\left(t\right)\right)\right)^2\left(\frac{d}{dt}s\left(t\right)\right)^2 + b^2\left(\cos\left(s\left(t\right)\right)\right)^2\left(\frac{d}{dt}s\left(t\right)\right)^2 = V^2$$

```
> diff(s(t),t) = solve(Ve2, diff(s(t), t));
```

$$\frac{d}{dt}s\left(t\right) = \left\{\frac{V}{\%1}, -\frac{V}{\%1}\right\}, \ \%1 = \sqrt{\left(\sin\left(s\left(t\right)\right)\right)^2 a^2 - b^2\left(\sin\left(s\left(t\right)\right)\right)^2 + b^2}$$

We have to select the first expresion, because the the jogger moves counter-clockwise. The differential equation has no analytical solution so we will solve it numerically with MATLAB. Together with the two differential equations (1.7) we obtain a system of three coupled differential equations:

$$\begin{aligned}
X(t) &= m_1 + a\cos(s(t)) \\
Y(t) &= m_2 + b\sin(s(t)) \\
\dot{x}(t) &= c(X(t) - x(t)) \\
\dot{y}(t) &= c(Y(t) - y(t)) \\
\dot{s}(t) &= \frac{V}{\sqrt{a^2 sin(s(t))^2 + b^2\cos(s(t))^2}} \\
c &= \frac{w}{\left|\left|\binom{X-x}{Y-y}\right|\right|}
\end{aligned}$$

This system is implemented as function `fkt` (Algorithm 1.8). The corresponding main program is given as Algorithm 1.9.

ALGORITHM 1.8.
Function `fkt` *for the Jogger with Constant Velocity.*

```
function [ydot,isterminal,direction] = fkt(t,y,flag)
% system of differential equations
% for the jogger-dog problem
% where the jogger runs with constant
% velocity on an ellipse
global a b m c w
A =  cos(y(3)); B =  sin(y(3));
X = m(1) + a*A; Y = m(2) + b*B;
h = [X;Y] -y(1:2); nh = norm(h);
zs =  (w/nh)*h;

if nargin < 3 | isempty(flag) % normal output
    ydot = [zs;c/sqrt((a*B)^2+(b*A)^2)];
else
    switch(flag)
    case 'events' % at norm(h)=0 there is a singularity
        ydot= nh-(1e-3); % zero crossing at pos_dog=pos_jogger
        isterminal= 1;   % this is a stopping event
        direction= 0;    % don't care if decrease or increase
    otherwise
        error(['Unknown flag ''' flag '''.']);
    end
end
```

To compare the results with the previous computations we choose the constant in Equation (1.8) as the average jogger velocity of the example in Section 1.4:

$$V \equiv \overline{\mathrm{V}(t)} = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} \mathrm{V}(t)\,dt \equiv \frac{L}{T}\ .$$

```
> T := 2*Pi:
> L := evalf(4*int(sqrt(20^2*sin(f)^2 + 15^2*cos(f)^2),
>      f = 0..Pi/2));
> V := evalf(L/T);
```

$$L := 110.5174608$$

$$\mathrm{V} := 17.58940018$$

If we execute Algorithm 1.9 we notice that the dog catches the jogger at time $t = 8.22834$. This is a little earlier than in Section 1.4.

1.7 Using a Moving Coordinate System

In this section we will use a Cartesian coordinate system to describe the position of the child respectively the jogger. The position of the toy respectively the dog

ALGORITHM 1.9.
Main Program for the Jogger with Constant Velocity.

```
>> % main5.m
>> global a b m c w
>> a = 20; b = 15; % semi-axes
>> m = [10;20];
>> c = 17.58940018; % constant velocity of jogger
>> w = 19;  % velocity of dog
>> y0 = [60, 70, 0]';  % ini. cond., starting point of the dog

>> options= odeset('AbsTol',1e-5,'Events','on');
>> [t,Y]= ode23 ('fkt', [0 20], y0, options);
>> clf; hold on;
>> axis ([-10 70 -10 70]);
>> % axis ('equal');
>> title ('The Jogger Runs with Constant Velocity.');
>> p = length(t)-1;
>> for h=1:p
>>   plot ([Y(h,1), Y(h+1,1)] , [Y(h,2), Y(h+1,2)], '-', ...
>>        'Color', 'yellow', 'EraseMode','none');
>>   s1 = Y(h,3); s2 = Y(h+1,3);
>>   X1 = m(1) + a*cos(s1); Y1 = m(2) + b*sin(s1);
>>   X2 = m(1) + a*cos(s2); Y2 = m(2) + b*sin(s2);
>>   plot ([X1, X2] , [Y1, Y2], ':', ...
>>        'Color', 'green', 'EraseMode','none');
>>   drawnow;
>> end;
>> cross(Y(p,1),Y(p,2),2);
>> hold off;
```

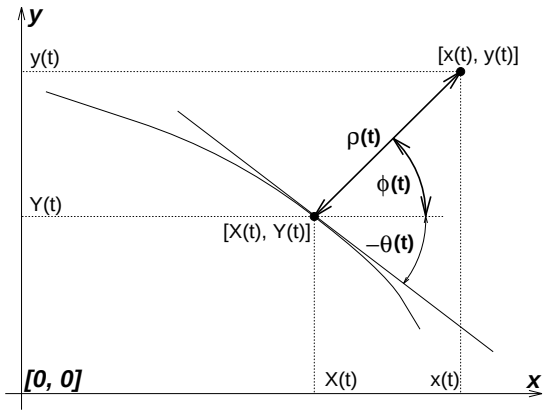
will be described in a moving polar coordinate system, see Figure 1.10. The origin of the moving system is the current position of the child respectively jogger. The current positions of the toy or the dog are thus given by the distance $\rho(t)$ and the polar angle $\phi(t)$.

If $[X(t), Y(t)]$ describe the current child/toy position then the position of the jogger/dog in Cartesian coordinates $[x(t), y(t)]$ is

$$x(t) = X(t) + \rho(t)\cos(\phi(t))\,, \quad y(t) = Y(t) + \rho(t)\sin(\phi(t))\,. \tag{1.9}$$

We want to express the system of differential equations (1.6) respectively (1.7) in the new variables $\rho(t)$ and $\phi(t)$. By doing so we will obtain a transformed system of differential equations for the functions $\rho(t)$ and $\phi(t)$.

FIGURE 1.10. *The moving Coordinate System,* $[\rho(t),\ \phi(t)]$



It is interesting to note that both systems are special cases of

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \frac{\sqrt{\dot{x}^2+\dot{y}^2}}{\left\|\begin{pmatrix}X-x\\ Y-y\end{pmatrix}\right\|}\begin{pmatrix}X-x\\ Y-y\end{pmatrix}. \tag{1.10}$$

If we keep the velocity constant $\sqrt{\dot{x}^2+\dot{y}^2} = w = const.$ then the system (1.10) describes the jogger/dog problem. We obtain on the other hand the equations for the child/toy problem if the distance is constant:

$$\rho(t) = \left\|\begin{pmatrix}X-x\\ Y-y\end{pmatrix}\right\| = a = const.$$

Since for the child/toy problem $\rho(t) = const.$, we expect that in polar coordinates the system will simplify to only one differential equation for $\phi(t)$. We will make this transformation with MAPLE.

1.7.1 Transformation for Jogger/Dog

We begin by defining the system (1.7):

```
> restart;
> S(t) := sqrt((X(t) - x(t))^2 + (Y(t) - y(t))^2):
> rx := diff(x(t), t) = W*(X(t) - x(t))/S(t);
> ry := diff(y(t), t) = W*(Y(t) - y(t))/S(t);
```

$$rx := \frac{d}{dt}x(t) = \frac{W(X(t)-x(t))}{\sqrt{(X(t)-x(t))^2+(Y(t)-y(t))^2}}$$

$$ry := \frac{d}{dt}y(t) = \frac{W(Y(t)-y(t))}{\sqrt{(X(t)-x(t))^2+(Y(t)-y(t))^2}}$$

Now we introduce the transformation of the functions and their derivatives:

```
> Rx := x(t) = X(t) + rho(t)*cos(phi(t)):
> Ry := y(t) = Y(t) + rho(t)*sin(phi(t)):
> Vx := diff(Rx, t);  Vy := diff(Ry, t);
```

$$Vx := \frac{d}{dt}x(t) = \frac{d}{dt}X(t) + \left(\frac{d}{dt}\rho(t)\right)\cos(\phi(t)) - \rho(t)\sin(\phi(t))\frac{d}{dt}\phi(t)$$

$$Vy := \frac{d}{dt}y(t) = \frac{d}{dt}Y(t) + \left(\frac{d}{dt}\rho(t)\right)\sin(\phi(t)) + \rho(t)\cos(\phi(t))\frac{d}{dt}\phi(t)$$

We substitute and simplify the result:

```
> qx := subs(Vx, rx):  qy := subs(Vy, ry):
> qx := simplify(subs(Rx, Ry, qx), symbolic);
> qy := simplify(subs(Rx, Ry, qy), symbolic);
```

$$qx := \frac{d}{dt}X(t) + \left(\frac{d}{dt}\rho(t)\right)\cos(\phi(t)) - \rho(t)\sin(\phi(t))\frac{d}{dt}\phi(t) = -W\cos(\phi(t))$$

$$qy := \frac{d}{dt}Y(t) + \left(\frac{d}{dt}\rho(t)\right)\sin(\phi(t)) + \rho(t)\cos(\phi(t))\frac{d}{dt}\phi(t) = -W\sin(\phi(t))$$

Finally we solve for the derivatives of $\rho(t)$ and $\phi(t)$:

```
> Dsys := solve({qx, qy}, {diff(rho(t), t), diff(phi(t), t)});
```

$$Dsys := \left\{ \begin{array}{l} \frac{d}{dt}\rho(t) = -\cos(\phi(t))\,\frac{d}{dt}X(t) - W - \left(\frac{d}{dt}Y(t)\right)\sin(\phi(t)), \\ \frac{d}{dt}\phi(t) = \dfrac{-\cos(\phi(t))\,\frac{d}{dt}Y(t) + \left(\frac{d}{dt}X(t)\right)\sin(\phi(t))}{\rho(t)} \end{array} \right\}$$

```
> Dradial:=select(has,Dsys,diff(rho(t),t))[];
> Daxial:=select(has,Dsys,diff(phi(t),t))[];
```

$$Dradial := \frac{d}{dt}\rho(t) = -\cos(\phi(t))\frac{d}{dt}X(t) - W - \left(\frac{d}{dt}Y(t)\right)\sin(\phi(t)) \quad (1.11)$$

$$Daxial := \frac{d}{dt}\phi(t) = \frac{-\cos(\phi(t))\,\frac{d}{dt}Y(t) + \left(\frac{d}{dt}X(t)\right)\sin(\phi(t))}{\rho(t)} \quad (1.12)$$

The resulting system of differential equations is somewhat simpler but cannot be solved analytically. We will therefore not continue the discussion.

1.7.2 Transformation for Child/Toy

As before we define the system of differential equations:

```
> restart;
> Rx := x(t) = X(t) + a*cos(phi(t)):
> Ry := y(t) = Y(t) + a*sin(phi(t)):
> Vx := diff(Rx, t): Vy := diff(Ry, t):
```

We will introduce the following substitution `RW` for the velocity of the toy:

```
> RW := W(t) = subs(Vx, Vy, sqrt(diff(x(t), t)^2
>       + diff(y(t), t)^2));
```

$$RW := W(t) = \sqrt{\left(\frac{d}{dt}X(t) - a\sin(\phi(t))\frac{d}{dt}\phi(t)\right)^2 + \left(\frac{d}{dt}Y(t) + a\cos(\phi(t))\frac{d}{dt}\phi(t)\right)^2}$$

The following statements generate the system (1.6)

```
> qx := diff(x(t), t) = W(t)/a*(X(t) - x(t));
> qy := diff(y(t), t) = W(t)/a*(Y(t) - y(t));
```

$$qx := \frac{d}{dt}x(t) = \frac{W(t)(X(t) - x(t))}{a}$$

$$qy := \frac{d}{dt}y(t) = \frac{W(t)(Y(t) - y(t))}{a}$$

We introduce the new variables and the derivatives by eliminating $x(t)$ and $y(t)$. Furthermore we squared both equations to get rid of the square root.

```
> qx := subs(RW, Vx, Rx, map(u -> u^2, qx));
> qy := subs(RW, Vy, Ry, map(u -> u^2, qy));
```

$$qx := \left(\frac{d}{dt}X(t) - a\sin(\phi(t))\frac{d}{dt}\phi(t)\right)^2$$

$$= \left(\left(\frac{d}{dt}X(t) - a\sin(\phi(t))\frac{d}{dt}\phi(t)\right)^2 + \left(\frac{d}{dt}Y(t) + a\cos(\phi(t))\frac{d}{dt}\phi(t)\right)^2\right)$$

$$(\cos(\phi(t)))^2$$

$$qy := \left(\frac{d}{dt}Y(t) + a\cos(\phi(t))\frac{d}{dt}\phi(t)\right)^2$$

$$= \left(\left(\frac{d}{dt}X(t) - a\sin(\phi(t))\frac{d}{dt}\phi(t)\right)^2 + \left(\frac{d}{dt}Y(t) + a\cos(\phi(t))\frac{d}{dt}\phi(t)\right)^2\right)$$

$$(\sin(\phi(t)))^2$$

We want to show that both equations are the same, and that the system reduces to only one differential equation for $\phi(t)$. To do this we make the following substitution:

```
> Subst := [rhs(Vx) = A, rhs(Vy) = B, phi(t) = F];
```

$$Subst := \left[\begin{array}{r} \frac{d}{dt}X\left(t\right) - a\sin\left(\phi\left(t\right)\right)\frac{d}{dt}\phi\left(t\right) = A, \\ \frac{d}{dt}Y\left(t\right) + a\cos\left(\phi\left(t\right)\right)\frac{d}{dt}\phi\left(t\right) = B, \\ \phi\left(t\right) = F \end{array} \right]$$

```
> q1x := expand(subs(Subst, qx));
> q1y := expand(subs(Subst, qy));
```

$$q1x := A^2 = \cos(F)^2 A^2 + \cos(F)^2 B^2$$

$$q1y := B^2 = \sin(F)^2 A^2 + \sin(F)^2 B^2.$$

In order to see that indeed the equations are the same we simplify them by collecting A and B terms.

```
> [map(u->simplify(sqrt( u-cos(F)^2*A^2),    symbolic),q1x),
>  map(u->simplify(sqrt(-u+sin(F)^2*A^2+B^2),symbolic),q1y)];
```

$$\left[A\sqrt{1-\cos\left(F\right)^2} = \cos\left(F\right)\ B, \qquad A\sqrt{1-\cos\left(F\right)^2} = \cos\left(F\right)\ B \right].$$

Now we see that they are identical. We continue the computation with the first one and remove the squares and the substitution. By solving for the derivative of $\phi(t)$ we obtain the desired differential equation:

```
> Q1:=subs(sqrt(1-cos(F)^2)=sin(F),%[1]);
```

$$Q1 := \sin(F)\, A = \cos(F)\, B$$

```
> BackSubst := map(u -> rhs(u) = lhs(u), Subst);
```

$$BackSubst := \left[\begin{array}{l} A = \frac{d}{dt}X\left(t\right) - a\sin\left(\phi\left(t\right)\right)\frac{d}{dt}\phi\left(t\right), \\ B = \frac{d}{dt}Y\left(t\right) + a\cos\left(\phi\left(t\right)\right)\frac{d}{dt}\phi\left(t\right) \\ F = \phi\left(t\right) \end{array} \right]$$

```
> Q2 := subs(BackSubst, Q1);
```

$$\begin{aligned} Q2 &:= \sin\left(\phi\left(t\right)\right)\left(\frac{d}{dt}X\left(t\right) - a\sin\left(\phi\left(t\right)\right)\frac{d}{dt}\phi\left(t\right)\right) \\ &= \cos\left(\phi\left(t\right)\right)\left(\frac{d}{dt}Y\left(t\right) + a\cos\left(\phi\left(t\right)\right)\frac{d}{dt}\phi\left(t\right)\right) \end{aligned}$$

```
> Daxial := diff(phi(t), t) =
>             simplify(solve(Q2, diff(phi(t), t)));
```

$$Daxial := \frac{d}{dt}\phi\left(t\right) = \frac{\sin\left(\phi\left(t\right)\right)\frac{d}{dt}X\left(t\right) - \cos\left(\phi\left(t\right)\right)\frac{d}{dt}Y\left(t\right)}{a}$$

It is interesting to note that we would obtain the same equation if we would replace the function $\rho(t)$ by the constant a in the differential equation for $\phi(t)$ for the jogger/dog problem.

1.8 Examples

Since the system of differential equations has simplified into one equation we may hope to obtain analytical solution for certain cases. We begin with the first example where the child is walking on a circle.

```
> Child_Subst := X(t) = R*cos(omega*t), Y(t) = R*sin(omega*t);
```

$$Child\_Subst := \mathrm{X}(t) = R\cos(\omega\, t),\ \mathrm{Y}(t) = R\sin(\omega\, t)$$

```
> Das := subs(Child_Subst, Daxial);
```

$$Das := \frac{d}{dt}\phi\,(t) = \frac{-\sin\left(\phi\,(t)\right) R\sin\left(\omega\, t\right)\omega - \cos\left(\phi\,(t)\right) R\cos\left(\omega\, t\right)\omega}{a}$$

```
> Das := combine(Das, trig);
```

$$Das := \frac{d}{dt}\phi\,(t) = -\frac{R\omega\,\cos\left(-\phi\,(t) + \omega\, t\right)}{a}$$

Depending of comparison of the lengths of a and R we may receive solution of the the following forms:

```
> Sol1 := dsolve(Das, phi(t)) assuming a>R;
```

$$Sol1 := \phi(t) = \omega\, t + 2\arctan\left(\frac{\tan\left(\frac{\omega\sqrt{a^2-R^2}\,(\_C1 - t)}{2\,a}\right)}{a-R}\sqrt{a^2-R^2}\right) \tag{1.13}$$

```
> Sol2 := dsolve(Das, phi(t)) assuming R<a;
```

$$Sol2 := \phi(t) = \omega\, t - 2\arctan\left(\frac{\tan\left(\frac{\omega\sqrt{R^2-a^2}\,(\_C1 - t)}{2\,a}\right)}{a-R}\sqrt{R^2-a^2}\right) \tag{1.14}$$

The functions $\phi(t)$ (1.13), (1.14) are given by an explicit expresion. Therefore MAPLE would be able to solve the differential equation with given initial condition $\phi(0) = \alpha$. Solving the differential equation `Das` with an initial conditions returns instead of (1.13) a much longer expresion. We abstain from showing it here.

```
> SuC1 := _C1 = solve(subs(t = 0, phi(0) = alpha, Sol1),_C1);
> SuC2 := _C1 = solve(subs(t = 0, phi(0) = alpha, Sol2),_C1);
```

$$SuC1 := \_C1 = 2\,\arctan\left(\frac{\tan\left(\alpha/2\right)(a-R)}{\sqrt{a^2-R^2}}\right)\frac{a}{\omega\sqrt{a^2-R^2}}$$

$$SuC2 := -\_C1 = 2\,\mathrm{arctanh}\left(\frac{\tan\left(\alpha/2\right)(a-R)}{\sqrt{R^2-a^2}}\right)\frac{a}{\omega\sqrt{R^2-a^2}}$$

To reproduce the orbit given in Figure 1.3 we set $\alpha = 0$:

```
> Sol10 := eval(subs(alpha=0,Sol1));
> Sol20 := eval(subs(alpha=0,Sol2));
```

$$Sol10 := \phi(t) = \omega t + 2 \arctan\left(\tanh\left(\frac{\omega\sqrt{R^2-a^2}t}{2a}\right)\frac{\sqrt{R^2-a^2}}{a-R}\right) \quad (1.15)$$

$$Sol20 := \phi(t) = \omega t - 2 \arctan\left(\tan\left(\frac{\omega\sqrt{a^2-R^2}t}{2a}\right)\frac{\sqrt{a^2-R^2}}{a-R}\right) \quad (1.16)$$

Furthermore we want to set the length of the bar a equal to the radius of the circle R. This is not possible by a simple substitution—we rather need to compute the limit $a \to R$: In this case both results (1.15) and (1.16) are equal.

```
> limit(rhs(Sol10), a = R), limit(rhs(Sol20), a = R);
```

$$\omega t - 2\arctan(\omega t), \quad \omega t - 2\arctan(\omega t)$$

```
> Sol0 := phi(t)= %[1]:
> Toy := [x(t), y(t)]:
> Toy_Plot := subs(Rx, Ry, Child_Subst, Sol0, omega = 1,
>                  a = R, R = 5, [Toy[], t = 0..40]);
```

$$Toy\_Plot := [5\cos(t) + 5\cos(-2\arctan(t) + t),$$
$$5\sin(t) + 5\sin(-2\arctan(t) + t),\ t = 0..40]$$

```
> plot(Toy_Plot, scaling = constrained, color = black);
```

We obtain the same plot as in Figure 1.3.

Figure 1.5 is obtained with the following statements:

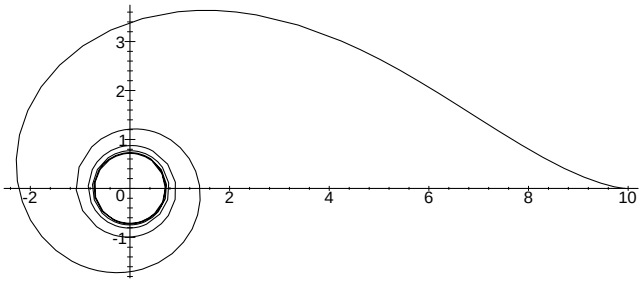
```
> Toy_Plot := subs(Rx, Ry, Child_Subst, Sol1, omega = 1,
>     alpha = arctan(-2)+Pi, a = sqrt(125), R = 5,
>     [Toy[], t  = 0..200]):
> plot(Toy_Plot, scaling = constrained, color = black);
```

Finally we compute the orbit of a variant of the first example. We will consider the case where the bar is shorter than the radius of the circle, see Figure 1.11.

```
> Toy_Plot := subs(Rx, Ry, Child_Subst, Sol2, omega = 1,
>     alpha = 0, a = 4.95, R = 5, [Toy[], t = 0..40]):
> plot(Toy_Plot, scaling = constrained, color = black);
```

Note that for the child walking on a straight line there is an analytical solution, the tractrix, as shown in the first section of this chapter. Let (X_0, Y_0) be the initial position and $(V_x,\ V_y)$ the constant velocity vector of the child:

FIGURE 1.11. *Toy Orbit for $a < R$*




```
> a := 'a': Vx := 'Vx': Vy := 'Vy':
> Child_Subst := X(t) = Xo + Vx*t, Y(t) = Yo + Vy*t:
> Das := eval(subs(Child_Subst, Daxial));
```

$$Das := \frac{d}{dt}\phi\left(t\right) = \frac{\sin\left(\phi\left(t\right)\right) Vx - \cos\left(\phi\left(t\right)\right) Vy}{a}$$

```
> Sol := dsolve(Das, phi(t));
```

$$Sol := \phi\left(t\right) = -2\arctan\left(\left(Vx + \tanh\left(\frac{\left(t + \_C1\right)\sqrt{Vy^2 + Vx^2}}{2\,a}\right)\frac{\sqrt{Vy^2 + Vx^2}}{Vy}\right)\right)$$

Repeating the same process as before for the case when the child is walking on a circle, we obtain again an implicit function for $\phi(t)$.

In general it will not be possible to obtain an analytic solution for the orbit of the toy. If we consider the child walking on the sine curve, we have to compute the orbit of the toy numerically. We will do this using MAPLE.

```
> X := t -> t:  Y := t -> 5*sin(t):  a := 10:
> Daxial;
```

$$Daxial := \frac{d}{dt}\phi\left(t\right) = 1/10\,\sin\left(\phi\left(t\right)\right) - 1/2\,\cos\left(\phi\left(t\right)\right)\cos\left(t\right)$$

```
> F := dsolve({Daxial, phi(0) = Pi/2}, phi(t), numeric);
```

$$F := \mathbf{proc}(rkf45\_x) \ldots \mathbf{end}$$

Because the plot is the same as in Figure 1.4 we will not print it here. We will, however, add a few commands to show the movements dynamically on the screen. Note that it is not possible to use the `animate` command for the same purpose.

We would like to see the movement of the bar, and the trajectory of both ends of the bar. One end has to move along the sine curve, the second will describe the computed orbit. We would like to see a movie describing the process during T seconds, as a sequence of $N + 1$ partial plots.

```
> N := 200: L := a:
> T:=evalf(6*Pi/N*[$0..N]):
> SF := [seq(rhs(F(t)[2]), t = T)]:
> plot(zip((u,v)->[X(u) + L*cos(v), Y(u) + L*sin(v)],T,SF));
```

The last MAPLE commands will display the orbit. For the movie we have to prepare and store the sequences of plots with increasing number of points.

```
> with(plots):
> TS := display(seq(plot(
>   zip((u,v)->[X(u)+L*cos(v), Y(u)+L*sin(v)],T[1..j],SF[1..j]),
>   color=blue,thickness=2), j = 1..N+1),insequence=true):
> BS := display(
>   zip((u,v)->plot([[X(u),Y(u)], [X(u)+L*cos(v),Y(u)+L*sin(v)]],
>   color=red),T,SF),insequence=true):
> PS := display(seq(plot([seq([X(u),Y(u)],u=T[1..j])],color=black),
>    j=1..N+1),insequence=true):
```

The animation TS contains information about the orbit. It stores $N+1$ partial plots containing 0 to N points. The animation BS describes the bar position for each frame. Finally PS describes the *sine* trajectory. Its structure is similar to TS.

The whole process can now be viewed with the command

```
> display({TS,BS,PS},view=[-2..20,-5..15]);
```

References

[1] E. HAIRER, S.P. NØRSETT and G. WANNER, *Solving Ordinary Differential Equations I,* Springer-Verlag Berlin Heidelberg, 1987.

[2] H. HEUSER, *Gewöhnliche Differentialgleichungen*, B. G. Teubner, Stuttgart, 1989.