Parametric curves and Surfaces

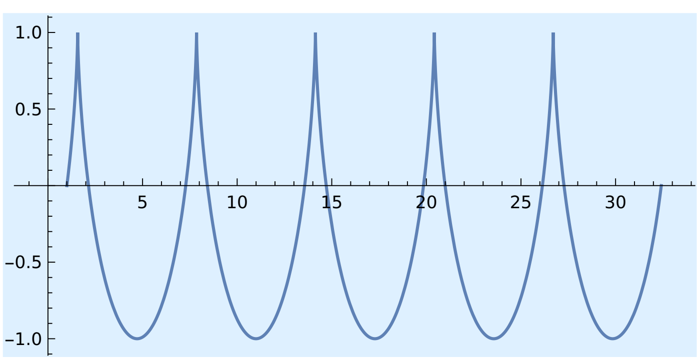
Parametric Curves in the plane : - If x and y are continuous functions of t
on an interval I , then the equations
x = x(t) and y = y(t)
are called parametric equations and t is called parameter .
The set of points (x , y) obtained as t varies over the interval I is called the graph of the parametric equations .
The graph of parametric equations is called a parametric curve or plane curve .

Vector is represented as a list in mathematica , a parametric curve can be defined as a List of two or more real - valued function .

In[25]:=
```
s[t_] := {Cos[t] + t, Sin[t]}
s[π / 4]
```

Out[26]= $\left\{\frac{1}{\sqrt{2}} + \frac{\pi}{4}, \frac{1}{\sqrt{2}}\right\}$

In[51]:=
```
ParametricPlot[s[t], {t, 0, 10 π}, AspectRatio → 1 / 2, Background → LightBlue]
```

Out[51]=


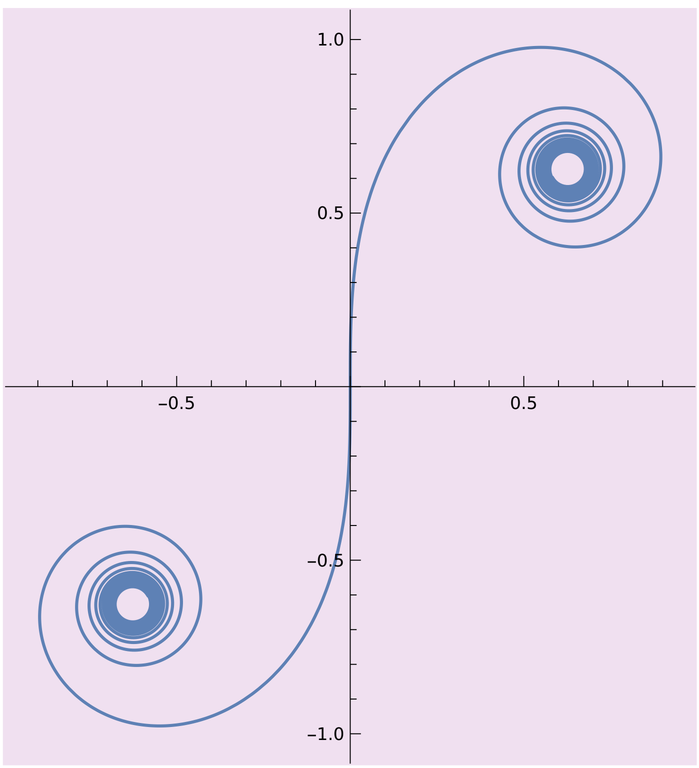
Here ,
AspectRatio - It is an option for Graphics and related functions that specifies

the ratio of height to
width for a plot .
ParametricPlot - It can express the x and y or x , y and z coordinates at each point on curves as a
function of one or more parameters .

Also , ParametricPlot works on well Piecewise functions and even on interesting curves like this .

In[49]:=
```
c[t_] = {Integrate[Sin[u^2], {u, 0, t}], Integrate[Cos[u^2], {u, 0, t}]};
ParametricPlot[c[t], {t, -10, 10}, Background → LightPurple]
```

Out[50]=

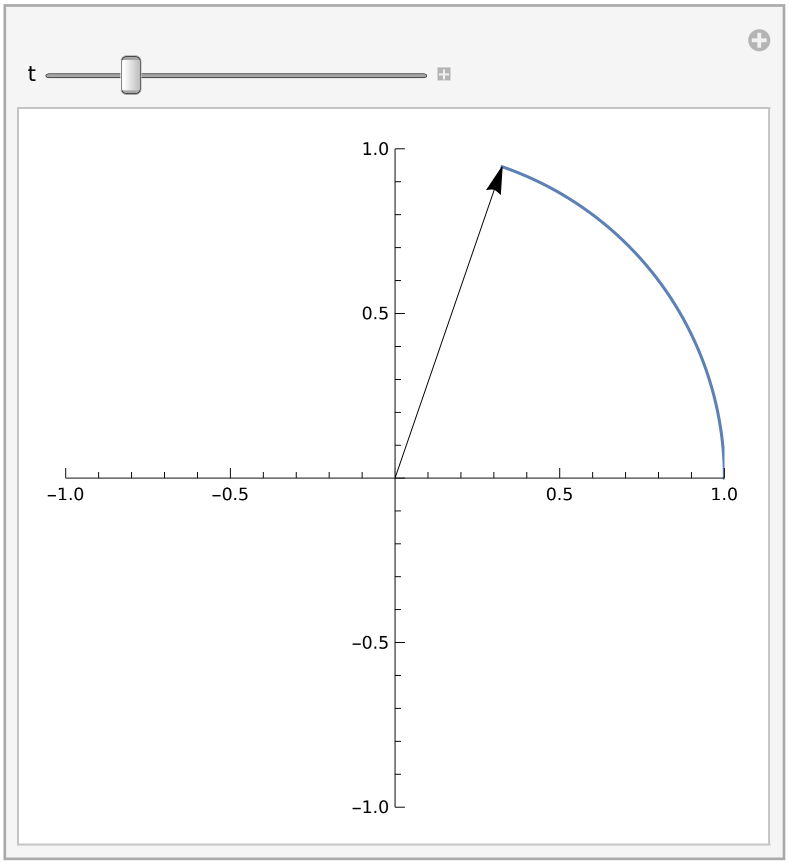


Manipulate can be harnessed to trace out a parametric curve , with a slider to control the independent variable.
Here , PlotRange setting (which keeps the plot range fixed as t varies) .

Here, for instance , is a standard parameterization of the unit circle .

In[64]:=
```
Manipulate[Show[ParametricPlot[{Cos[t0], Sin[t0]}, {t0, 0, t}, PlotRange → 1],
  Graphics[Arrow[{{0, 0}, {Cos[t], Sin[t]}}]]], {{t, 1.}, 0.01, 2 π}]
```

Out[64]=

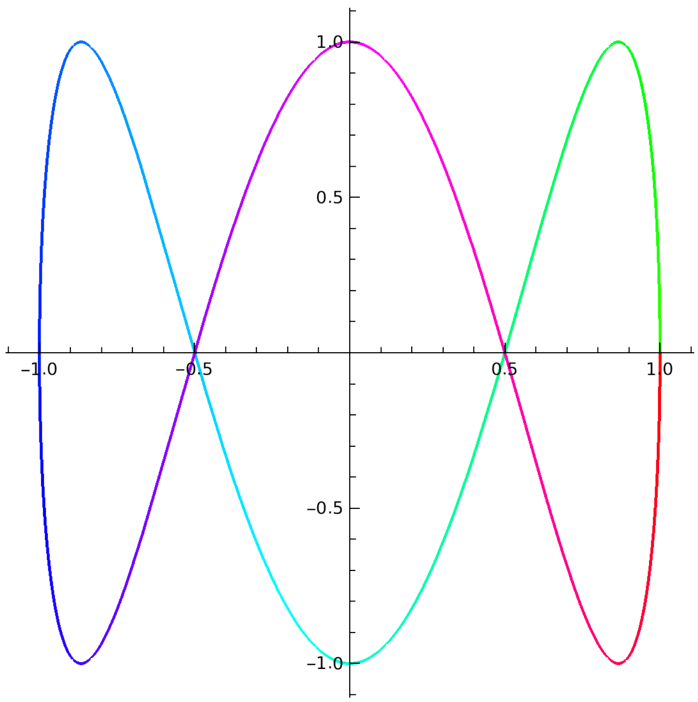


Manipulate - It is designed to work with the full range of possible types of output .
Graphics - It represents a two - dimensional graphical image .

Here , also define a custom parametric plotting command that will apply a color gradient to the curve , so that
it will start in green and gradually progress through the color gradient to end in red (Green for go , red for stop).

In[65]:=
```
ParametricPlot[{Cos[t], Sin[3 t]}, {t, 0, 2 π}, ColorFunction → (Hue[.7 #3 + .3] &)]
```

Out[65]=



Colorfunction - It is an option for graphics functions that specifies a function to apply to determine colors of
elements .
Hue - It corresponds to a cylindrical transformation pf RGB color , allowing for easier interpretation of color
parameters .

The ColorFunction accepts any or all of three arguments. The first two are the x and y coordinates of the parametric curve. The third (#3 used above) is the independent variable t. By default, the values of t will be scaled to run from 0 to 1 before being

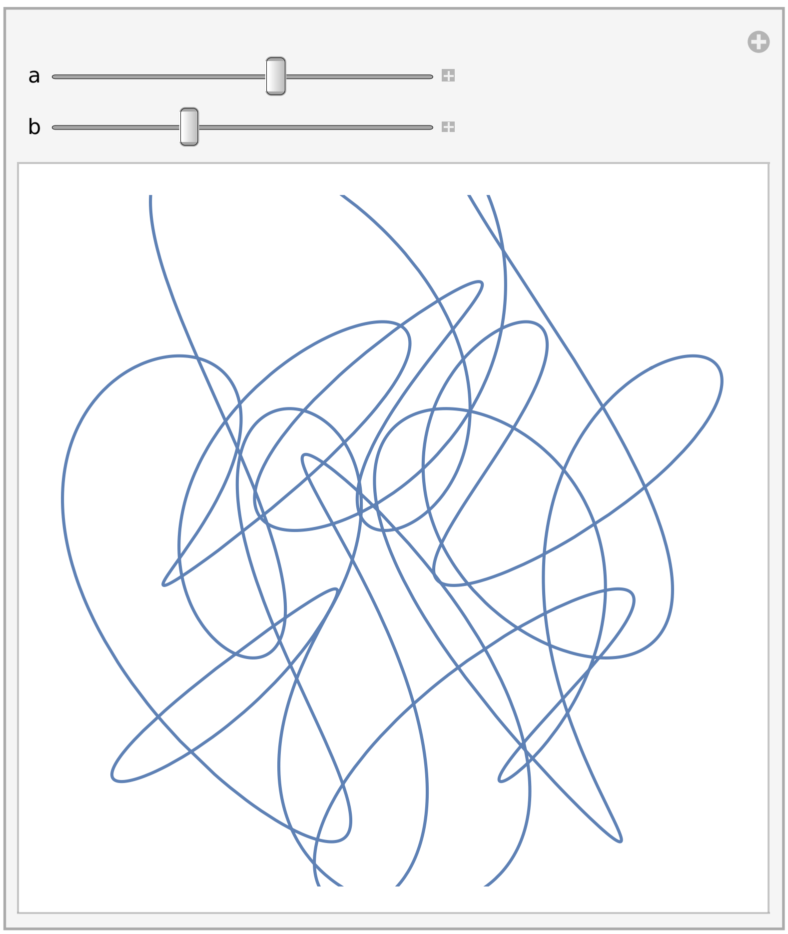
input to the ColorFunction, regardless of the domain you choose. Hence the option setting above produces a color gradient that starts at Hue[.3] (green) and ends at Hue[1.] (red) .

Then ,

We note that ParametricPlot accepts most of the options accepted by other two-dimensional plotting commands such as Plot. The PlotPoints option, for example, can be set to a numerical value (such as 100) if you see jagged segments where you suspect they should not be .

In[2]:=
```
Manipulate[
 ParametricPlot[{Cos[t] + 1 / 2 * Cos[7 t] + 1 / 2 * Sin[a * t], Sin[t] + 1 / 2 * Sin[7 t] + Cos[b * t]},
  {t, 0, 2 π}, Axes → False, PlotRange → 2], {{a, 17}, 5, 25}, {{b, 12}, 5, 25}]
```

Out[2]=



Further ,

The derivative of the parametric function [x(t) , y(t)] is [x'(t) , y'(t)] .

In[4]:=
```
s[t_] := {t + Cos[t], Sin[t]};
D[s[t], t]
```

Out[5]= {1 - Sin[t], Cos[t]}

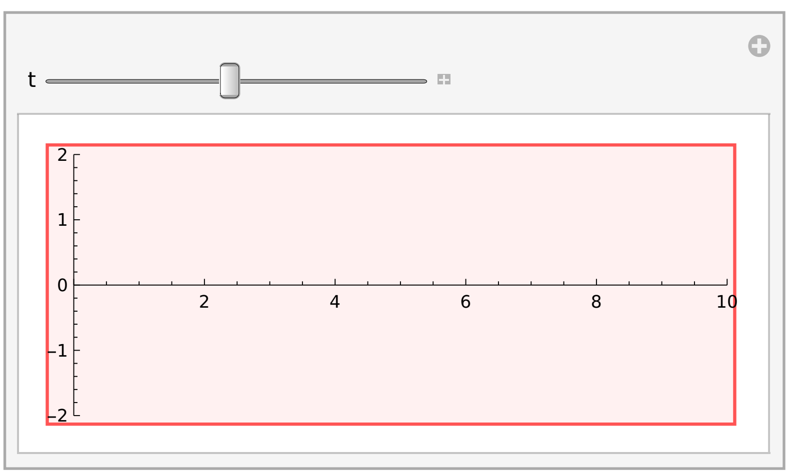
In[6]:=
```
s'[t]
```

Out[6]= {1 - Sin[t], Cos[t]}

Note that while the ParametricPlot of s(t) has sharp corners (it is the first plot shown at the beginning of this section), its derivative is defined everywhere. This can happen. The Manipulate below shows the derivative vector s' (t) with its tail at the point s(t), as t varies. When t is $\pi/2$ or $5\pi/2$, sin t = 1 and cos t = 0, and so the derivative is the zero vector. This happens at the top of each sharp corner in the plot.

In[11]:=
```
Module[{s}, s[t_] = {t + Cos[t], Sin[t]};
Manipulate[Show[ParametricPlot[s[t0], {t0, 0, 10}, PlotRange → {{0, 10}, {-2, 2}}],
Graphics[Arrow[{s[t], s[t] + s'[t]}]]], {{t, 4.8}, 0, 10}]]
```

Out[11]=



Module - It specifies that occurrence of the symbols x,y,... in expression should be treated as local .

If s (t) represents the position of a particle at time t, then its

velocity vector is s ' (t), and its speed is the magnitude of this vector. To compute speed, say at time t = 3 ,

Graphic commands represents the Two-dimensional graphical image.

In[35]:=
```
Norm[s '[3]] // N
```
Out[35]=
```
1.31063
```

Norm - It gives the norm of a number , vector , or matrix .

You can even get a formula for speed as a function of t. Here we produce and Simplify the formula, using the optional second argument for Simplify in order to specify that t is permitted to assume only real values (as opposed to complex values). We could have given the second argument as Element[t,Reals]

In[17]:=
```
Simplify[Norm[s '[t]], t ∈ Reals]
```
Out[17]= $\sqrt{2 - 2\,\mathrm{Sin}[t]}$

Note that this is consistent with the Manipulate above; speed is zero precisely when t is π / 2, 5 π / 2, etc.

 # Unit tangent vectors are constructed exactly as you would expect. The use of Simplify as above will generally serve you well.

In[22]:=
```
unitTangent[s_, t_] := Simplify[D[s, t] / Norm[D[s, t]], t ∈ Reals]
```
In[23]:=
```
s[t_] = {t + Cos[t], Sin[t]};
unitTangent[s[t], t]
```
Out[24]= $\left\{\frac{\sqrt{1 - \mathrm{Sin}[t]}}{\sqrt{2}}, \frac{\mathrm{Cos}[t]}{\sqrt{2 - 2\,\mathrm{Sin}[t]}}\right\}$

In[25]:=
```
unitTangent[s[t], t] /. t → 1.2
```
Out[25]=
```
{0.184338, 0.982863}
```

unitTangent -
unitNormal - Compute the unit normal of a surface .

Unit normal vectors can be formed in a similar way (by use of FullSimplify): -

In[28]:=
```
unitNormal[s_, t_] :=
 FullSimplify[D[unitTangent[s, t], t] / Norm[D[unitTangent[s, t], t]], t ∈ Reals]
```

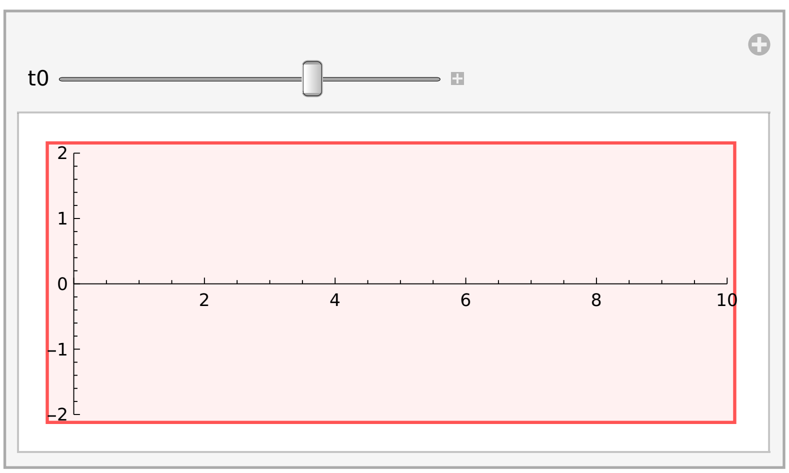
In[29]:=
```
unitNormal[s[t], t]
```

Out[29]=
$$\left\{-\frac{\text{Cos}[t]}{\sqrt{2-2\,\text{Sin}[t]}},\ \frac{\sqrt{1-\text{Sin}[t]}}{\sqrt{2}}\right\}$$

 # Even though neither the unit tangent nor the unit normal is defined when t =π/ 2 or t =5 π/ 2, the
following Manipulate works fine, as it is unlikely to sample these precise values. We define auxiliary commands ut and un so that unitTangent and unitNormal only need to be called once (they
are slow, after all, since they use Simplify and FullSimplify, respectively, each time they are called).
The auxiliary commands ut and un are defined using Set (=). So they use the formulas generated by
unitTangent and unitNormal, and simply replace the variable t by whatever argument x is specified.

In[30]:=
```
Module[{s, ut, un, t},
s[t_] = {t + Cos[t], Sin[t]};
ut[x_] = unitTangent[s[t], t] /. t → x;
un[x_] = unitNormal[s[t], t] /. t → x;
Manipulate[Show[ParametricPlot[s[t], {t, 0, 10}, PlotRange → {{0, 10}, {-2, 2}}],
Graphics[{Blue, Arrow[{s[t0], s[t0] + ut[t0]}]}],
Graphics[{Red, Arrow[{s[t0], s[t0] + un[t0]}]}]],
{{t0, 6.8}, 0, 10}]]
```

Out[30]=



In[32]:=
```
κ[s_, t_] := FullSimplify[Norm[D[unitTangent[s, t], t]] / Norm[D[s, t]], t ∈ Reals]
κ[s[t], t]
```

Out[33]=

$$\frac{1}{2\sqrt{2-2\,\mathrm{Sin}[t]}}$$

 # And so the radius of curvature is the reciprocal of this quantity, 2Sqrt [2-2Sint] . At the sharp corners
(when t is $\pi/2$, $5\pi/2$, etc.) the curvature is undefined, and the radius of curvature approaches zero.
The Manipulate below shows the osculating circle for any value of t. This illustrates that the radius
of curvature approaches zero very rapidly as t approaches $5\pi/2$.

In[34]:=
```
Module[{s, un, curv, t},
s[t_] = {t + Cos[t], Sin[t]};
un[x_] = unitNormal[s[t], t] /. t → x;
curv[x_] = κ[s[t], t] /. t → x;
Manipulate[Show[ParametricPlot[s[t], {t, 0, 10}, PlotRange → {{-4, 12}, {-2, 6}}],
Graphics[{Gray, Circle[s[t0] + un[t0] / curv[t0], 1 / curv[t0]]}],
Graphics[{Red, Arrow[{s[t0] + un[t0] / curv[t0], s[t0]}]}]],
{{t0, 6.5}, 0, 10}]]
```

Out[34]=