Curvas parametrizadas - trajetórias

Exemplo genérico 2D

```
> restart; with(plots) :
```

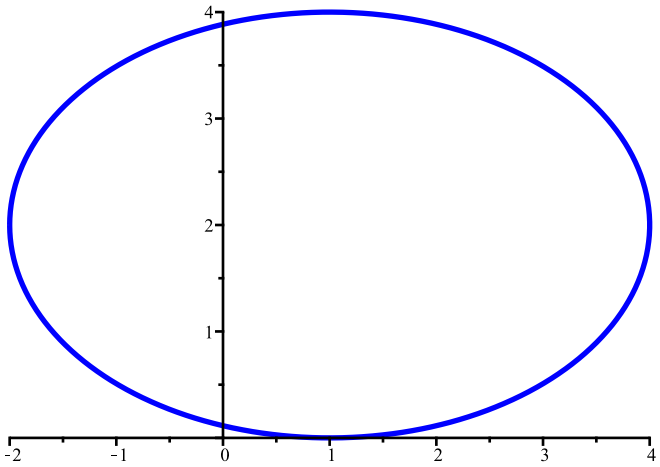
Digamos que temos a parametrização de uma curva no plano. Em coordenadas cartesianas, basta termos x(t) e y(t) funções reais a um parâmetro t .

A sintaxe para se desenhar uma curva parametrizada no plano é:

plot([x(t), y(t), t= a..b], opçoes);

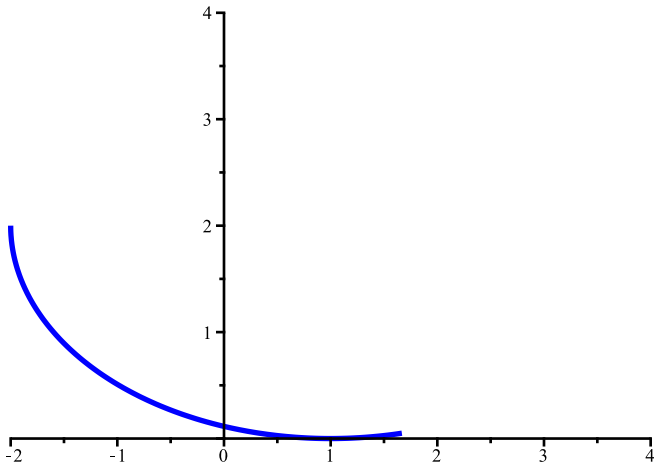
Vamos exemplificar com uma elipse:

```
> plot([1 + 3*cos(t), 2 + 2*sin(t), t = -Pi..Pi], color = blue, thickness = 2, scaling
      = constrained); # o desenho pronto e estático
```



criar uma sequência de quadros a serem mostrados a uma velocidade suficiente para dar a idéia de movimento:

```
> N := 7 : #(8=N+1 será o número de quadros)#
  for i from 0 to N do
  s := i/N;
  T := -Pi + (t + Pi) * s :
  quadro[i] := plot([1 + 3*cos(T), 2 + 2*sin(T), t = -Pi..Pi], color = blue, thickness = 2) :
  od:
> display([seq(quadro[i], i = 0..N)], insequence = true, scaling = constrained);
```



Uma ilustração da parametrização utilizada para Geometria Analítica

No caso da parametrização da elipse dada acima, o parâmetro t tem um significado geométrico, que pode ser visualizado através da seguinte animação:

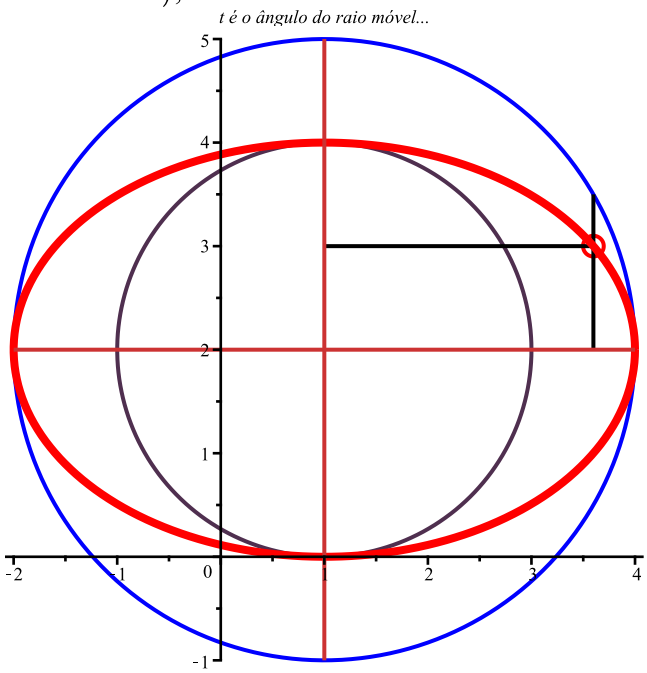
A programação da animação:

```
> restart; with(plots) : with(linalg) :
  with(plottools) : # somente do Release 4 em diante
```

```
> CE := circle( [1, 2], 3, color = blue) :
  CI := circle([1, 2], 2, color = violet) :
  elipse := plot( [1 + 3 * cos(t), 2 + 2 * sin(t), t = 0 ..2 * Pi], color = red, thickness = 3) :
  eixos := plot( {[[-2, 2], [4, 2]], [[1, -1], [1, 5]]}, color = orange) :
  theta := Pi / 6 : n := 20 :
  for i from 0 to n do
  raio := plot( [[1, 2], [1 + 3 * cos(theta), 2 + 3 * sin(the • ta)]], color = black, thickness = 2) :
  x0 := 1 + 3 * cos(theta) : y0 := 2 + 2 * sin(theta) :
  ponto := circle([x0, y0], .1, color = red) :
  rx := plot([[x0, 2 + 3 * sin(theta)], [x0, 2]], color = black) :
  ry := plot([[1 + 3 * cos(theta), y0], [1, y0]], color = black) :
  theta := theta + 2 * Pi / n :
  quadro[i] := display( {CE, CI, elipse, raio, rx, ry, eixos, ponto}, scaling = constrained) :
  od:
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
```

```
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
Warning, data could not be converted to float Matrix
```

```
> display( [seq(quadro[ j ], j = 0 ..n ) ], scaling = constrained, insequence = true, title
      = `t é o ângulo do raio móvel... `);
```



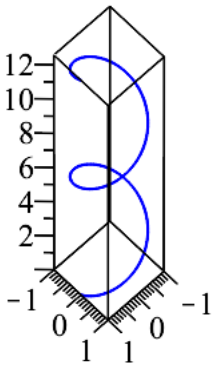
Exemplo genérico 3D -(curva parametrizada)

Neste caso, a sintaxe para plotar uma curva em 3D tem ligeira diferença em relação ao caso 2D:

spacecurve([x(t), y(t), z(t)], t=a ..b, opções);,

Note que a variação do parâmetro aqui está fora do colchete.

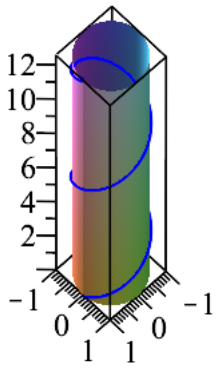
```
> restart; with( plots ) :
> alpha := t → [cos( t ), sin( t ), t ] : # a parametrização da curva (hélice)
> spacecurve( alpha( t ), t = 0 ..4 * Pi, scaling = constrained, color = blue, thickness = 2, orientation
      = [46, 26 ]); # o traço, estático
```



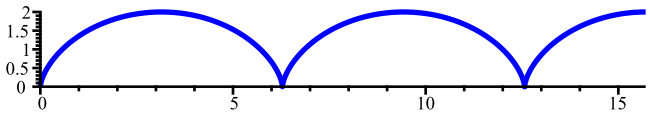
Animando a trajetória, com um fundo fixo:

```
> cilindro := plot3d([cos(u), sin(u), v], u = 0..2*Pi, v = 0..4*Pi, style = patchnogrid) :

  N := 10 :
  for i from 0 to N do
  s := i/N;
  quadro[i] := display({cilindro, spacecurve(alpha(t*s), t = 0..4*Pi, scaling = constrained,
      color = blue, thickness = 4, numpoints = 10*(i + 1), orientation = [46, 26])}) :
  od:
  display([seq(quadro[i], i = 0..N)], insequence = true);
```




```
> restart; with(plots) :
> plot( [t-sin(t), 1-cos(t), t=0..5*Pi], scaling=constrained, thickness=2, color=blue);
```

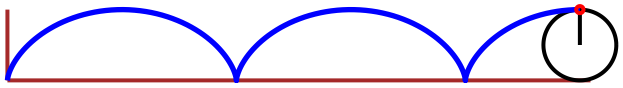


Vamos agora criar uma animação, para enxergar essa trajetória.

```
> restart; with(plots) : with(plottools) :
```

```
x := t → t - sin(t) :
y := t → 1 - cos(t) : # coordenadas do ponto
cx := t → t :
cy := t → 1 : # coordenadas do centro da circunferência
eixos := plot( {[[0, 0], [16, 0]], [[0, 0], [0, 2]]}, color = brown) :
N := 10 : # número de quadros da animação
t := 0 : passo := 5 * Pi / N :
for i from 0 to N do
ponto := circle( [x(t), y(t)], .1, color = red) :
curva := plot( [x(s), y(s), s = 0 ..t], color = blue, thickness = 2 ) :
roda := circle( [cx(t), cy(t)], 1, color = black) :
raio := plot( [[cx(t), cy(t)], [x(t), y(t)]], color = black) :
quadro[i] := display( [eixos, roda, raio, curva, ponto], scaling = constrained) :
t := t + passo :
od:
display( seq(quadro[i], i = 0 ..N ), insequence = true, title = `Ciclóide`, axes = none, scaling
    = constrained);
```

Ciclóide

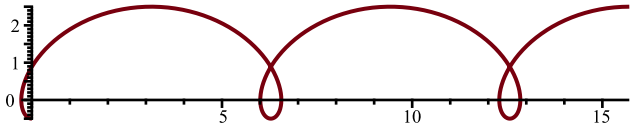


Voltar para ciclóide

Hiperciclóide

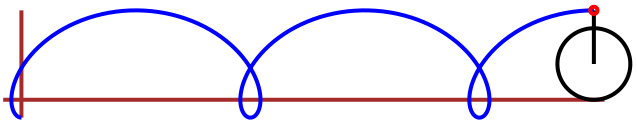
Vamos agora ver o que ocorre se o ponto estivesse na borda de uma roldana que se desloca sobre o eixo Ox, isto é, o raio ligando o ponto ao centro é maior que o raio interno da roldana.

```
> restart; with(plots) :
x := t→ t − 1.5 * sin(t) : y := t → 1 - 1.5 * cos(t) :
plot( [x(t), y(t), t = 0 ..5 * Pi], scaling = constrained);
```



Vamos animá-la, como anteriormente:

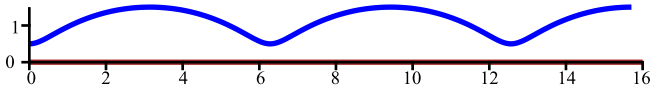
```
> restart; with(plots) : with(plottools) :
  x := t →t - 1.5 * sin(t) : y := t → 1 - 1.5 * cos(t) : # coordenadas do ponto
  cx := t → t : cy := t→ 1 : # coordenadas do centro da circunferência
 eixos := plot( {[[-.5, 0], [16, 0]], [[0,-.5], [0, 2.5]]}, color = brown) :
 N := 10 : # número de quadros da animação
  t := 0 : passo := 5 * Pi / N :
  for i from 0 to N do
 ponto := circle( [x(t), y(t)], .1, color = red) :
 curva := plot( [x(s), y(s), s = 0 ..t], color = blue) :
  roda := circle( [cx(t), cy(t)], 1, color = black) :
 raio := plot( [[cx(t), cy(t)], [x(t), y(t)]], color = black) :
  quadro[i] := display( [eixos, roda, raio, curva, ponto], scaling = constrained) :
  t := t + passo :
  od:
 display( seq(quadro[i], i = 0 ..N), insequence = true, title = `Hiperciclóide`, axes = none, scaling
      = constrained);
```



Hipociclóide

E se o ponto estivesse no interior da roda que se desloca sobre o eixo Ox?

```
> restart; with(plots) :
  x := t→ t − .5 * sin(t) : y := t → 1 − .5 * cos(t) :
  plot( [[x(t), y(t), t = 0 ..5 * Pi], [t, 0, t = 0 ..16]], scaling = constrained, color = [blue, brown],
        thickness = 2, tickmarks = [8, 3]);
```




```
> restart; with(plots) : with(plottools) :
  x := t → t − .5 * sin(t) :
```

```
y := t → 1 - .5 * cos(t) : # coordenadas do ponto
 cx := t → t :
cy := t → 1 : # coordenadas do centro da circunferência
eixos := plot( {[[0, 0], [16, 0]], [[0, 0], [0, 2]]}, color = brown) :
 N := 10 : # número de quadros da animação
 t := 0 : passo := 5 * Pi / N :
 for i from 0 to N do
 ponto := circle( [x(t), y(t)], .1, color = red) :
curva := plot( [x(s), y(s), s = 0 ..t], color = blue, thickness = 2 ) :
 roda := circle( [cx(t), cy(t)], 1, color = black) :
raio := plot( [[cx(t), cy(t)], [x(t), y(t)]], color = black) :
quadro[i] := display( [eixos, roda, raio, curva, ponto], scaling = constrained) :
t := t + passo :
od:
display( seq(quadro[i], i = 0 ..N), insequence = true, title = `Hipociclóide`, axes = none, scaling
      = constrained);
```

Hipociclóide