3.8 Curvas de perseguição

Nesta seção vamos estudar vários problemas clássicos de perseguição, que já foram investigados na época de Leonardo da Vinci. Trata-se de computar a trajetória de um perseguidor que persegue o seu alvo de tal maneira que seu vetor-velocidade, em cada momento, visa na posição atual do alvo. Um exemplo é um cachorro que corre para encontrar-se com o seu dono (ou que quer atacar um jogger ...). Queremos, evidentemente, visualizar os caminhos do cão e do seu dono numa gráfica animada.

A palavra *atacar*, que acabamos de usar, evoca outro exemplo moderno, a saber, o do míssil com uma "tête chercheuse", ou seja, equipado com uma cabeça de busca eletrônica, que persegue um alvo, supostamente mandado pelo inimigo, para produzir uma "reunião" espetacular.

Uma variação deste tema é o caso do nadador que quer atravessar um rio, sempre visando um ponto fixo no outro lado do rio. Veremos que a trajetória do nadador é uma espécie de parábola.

Resolvamos primeiro este problema do nadador.

3.8.1 Cruzando um rio

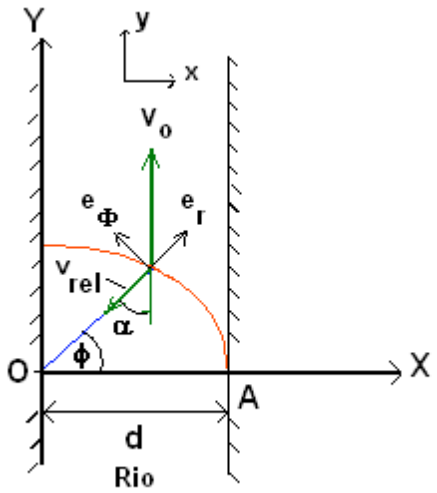

Fig.: 3.8-1

O nadador está no eixo-X no ponto A = (x_o,0). Ele cruza o rio com a velocidade $\mathbf{v}_{rel}$ e este vetor aponta a cada momento para o objetivo de sua busca, por exemplo uma amiga que o espera na origem O. $\mathbf{v}_{rel} = -c\,\mathbf{e}_r$ é a velocidade do nadador em relação ao rio; $\mathbf{e}_r$ é um vetor unitário radial. A distância OA é d e a velocidade absoluta do rio, medida em (X,Y), é $\mathbf{v}_o = v_o\,\mathbf{j}$. Na realidade, trata-se de um problema dos sistemas não-inerciais, enquanto nos problemas de perseguição usamos só um sistema inercial.

Qual a trajetória descrita pelo nadador? Em que ponto S = (0,y_{fin}) chegará ele ao lado oposto do rio?

Solução:

Suponhamos que o sistema não-inercial (x,y) está movendo-se com o rio, p.ex. fixado num barco, com velocidade absoluto $\mathbf{v}_o$, medida desde beira-rio. Na Eq. (4) do parágrafo 3.5.1, ou seja $\mathbf{v}_{abs} = \mathbf{v}_{rel} + \mathbf{v}_o + \boldsymbol{\omega} \times \mathbf{r}$, pomos $\boldsymbol{\omega} = 0$ e escrevemos, para simplificar, $\mathbf{v}_{abs} := \mathbf{v}$.

Temos, então,

$$\mathbf{v} = \mathbf{v}_{rel} + \mathbf{v}_o = -c\,\mathbf{e}_r + v_o\,\mathbf{j} \qquad (1)$$

Vamos decompor $\mathbf{v}$ em suas componentes radial e transversa $\mathbf{v}_r$ e $\mathbf{v}_\varphi$: $\mathbf{v} = \mathbf{v}_r + \mathbf{v}_\varphi$.

A descomposição de $\mathbf{v}_o$ da $\mathbf{v}_{o,r} = v_o \cos\alpha\,\mathbf{e}_r$ e $\mathbf{v}_{o,\varphi} = v_o \operatorname{sen}\alpha\,\mathbf{e}_\varphi$, sendo α o ângulo entre $\mathbf{v}_{rel}$ e ($-\mathbf{v}_o$).

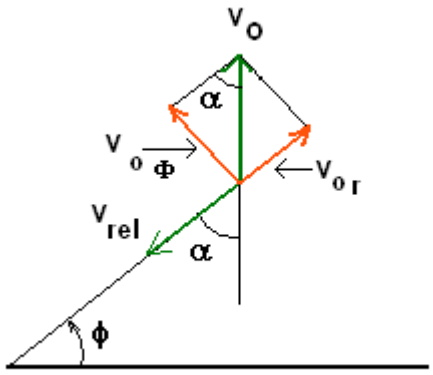

Fig.3.8-2

O vetor $\mathbf{v}_{rel}$ tem somente uma componente radial: $\mathbf{v}_{rel} = -c\,\mathbf{e}_r = \mathbf{v}_r$. Então

$\mathbf{v} = \mathbf{v}_r + \mathbf{v}_\varphi = (-c + v_o \cos\alpha)\,\mathbf{e}_r + v_o \operatorname{sen}\alpha \cdot \mathbf{e}_\varphi$. Assim temos, ver 3.4.9, Eq. (4):

$$\mathbf{v_r} = dr/dt \cdot \mathbf{e_r} = (- c + v_o \cos\alpha)\, \mathbf{e_r}$$
$$\mathbf{v_\varphi} = r \cdot d\varphi/dt \cdot \mathbf{e_\varphi} = v_o \operatorname{sen}\alpha \cdot \mathbf{e_\varphi}. \qquad (2)$$

Na segunda equação substituimos dφ/dt por -dα/dt, jα que α = 90°-φ e dα/dt = -dφ/dt. Dividindo a primeira equação pela segunda, obtemos $dr/r = (c-v_o\cos\alpha)/(v_o \operatorname{sen}\alpha) \cdot d\alpha$, integrando:

$$\ln(r) = (c/v_o) \cdot \ln[\operatorname{tg}(\alpha/2)] - \ln[\operatorname{sen}(\alpha)] + C \qquad (3)$$

A constante C de integração determinamos por meio das condições iniciais $\alpha = \alpha_o$ e r = d. Pondo r:= d, obtemos

$$r = d\frac{\operatorname{sen}\alpha_0}{\operatorname{sen}\alpha} \cdot \left(\frac{\operatorname{tg}\frac{\alpha}{2}}{\operatorname{tg}\frac{\alpha_o}{2}} \right)^{\frac{c}{v_0}} \qquad (4)$$

Suponhamos agora que $c = v_o$ e $\alpha_o = \pi/2$, então

$$r = d\frac{\operatorname{tg}\frac{\alpha}{2}}{\operatorname{sen}\alpha} = \frac{d}{1+\cos\alpha} = \frac{d}{1+\operatorname{sen}\varphi} \qquad (5)$$

Esta equação representa uma parábola. Isso se vê mais facilmente, introduzindo coordenadas cartesianas: $r = (x^2+y^2)^{1/2}$ e $\operatorname{sen}\varphi = y/r$.
Obtém-se $d-y = (x^2 + y^2)^{1/2}$, ou seja

$$y = d/2 - x^2/(2d) \qquad (6)$$

É isso a equação de uma parábola com vértice no ponto S = (0,d/2).

No seguinte Programa 1 temos colocada a equação (4) com as condições iniciais d = 100 m e $c = v_o = 5$ m/s. α_o = PI/2. O segundo programa trabalha com a equação (5).

Note em ambos os programas o uso da instrução **`plot::Polar.`**

Programa 1

```
c:=5:v:=5:a0:=PI/2:d:=100://nadador com c =
vplot(plot::Polar([d*(sin(a0)/sin(a))*
  (tan(a/2)/tan(a0/2))^(c/v),a],a=0..a0))
```

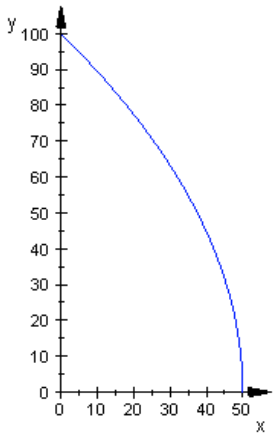

Fig.: 3.8-3

Programa 2

```
plot(plot::Polar([100/(1+sin(f)),f], f = 0..PI/2))
```

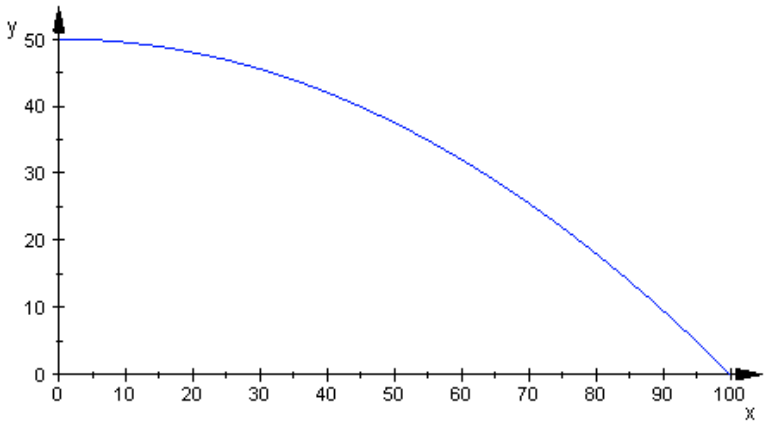

Fig.: 3.8-4

Suponhamos agora que o perfil da velocidade do rio tem forma parabólica:

$\mathbf{V_o}$ = V(y) **i** = (V - 4V(y-h/2)^2/h^2)·**i**. Nas bordas do rio, onde y = 0 e y = h, temos velocidade 0, no centro, com y = h/2, o rio tem sua velocidade máxima V.

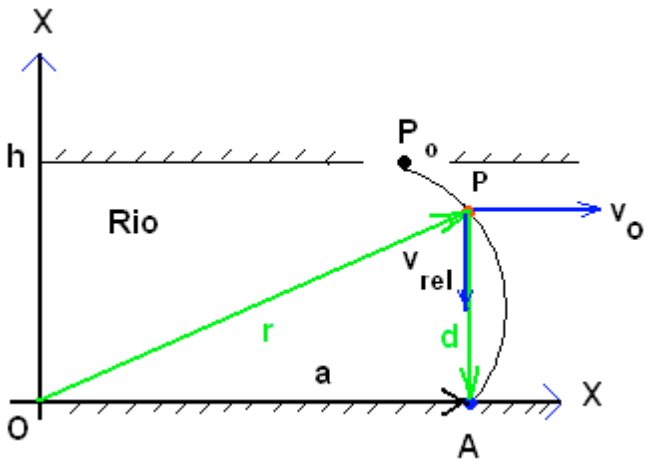

Fig.: 3.8-5

As seguintes relações vao conduzir-nos às equações do movimento.
$\mathbf{v} = \mathbf{v_{rel}} + \mathbf{V_o} = c\cdot\mathbf{c_o} + V(y)\cdot\mathbf{i}$, c = velocidade do nadador, $\mathbf{c_o}$ = vetor unitário na direção do vetor $\mathbf{v_{rel}}$. Da figura segue a relação **d** = **a** - **r**, onde |**d**| é a distância entre P e A. Para esta magnitude obtemos

$$d = |\mathbf{d}| = ((x_a - x)^2 + (y_a - y)^2)^{1/2}$$

O vetor unitário é dado por $\mathbf{c_o} = \mathbf{d}/|\mathbf{d}| = (\mathbf{a} - \mathbf{r})/d$, e suas componentes são

$$\mathbf{c_{o,x}} = (\mathbf{a}_x - \mathbf{r}_x)/d = (x_a - x)/d \cdot\mathbf{i} \quad \text{e} \quad \mathbf{c_{o,y}} = (\mathbf{a}_y - \mathbf{r}_y)/d = (y_a - y)/d \cdot\mathbf{j};\ y_a = 0$$

Para as componentes da velocidade absoluta **v** obtemos agora

$$\mathbf{v}_x = c\cdot\ \mathbf{c_{o,x}} + V(y)\cdot\mathbf{i} = c(x_a - x)/d\cdot\mathbf{i} + V(y)\cdot\mathbf{i} = dx/dt\ \cdot\mathbf{i}$$

$$\mathbf{v}_y = c\cdot\ \mathbf{c_{o,y}} = c(y_a - y)/d\cdot\mathbf{j} = dy/dt\ \cdot\mathbf{j}$$

Ou seja:

$$dx/dt = c(x_a - x)/d + V(y),$$
$$dy/dt = c(y_a - y)/d \text{ com } y_a = 0 \qquad (7)$$

São estes as equações do movimento do nadador. Ambas as equações são acopladas por meio de d e V(y), pois d contém x e y e V(y) contém y.

Para resolver o sistema (7) de duas equações diferenciais de primeiro grau, utilizamos **`numeric::ode2vectorfield`** e **`numeric::odesolve2`**, como já fizemos nas seções 2.5 e 3.2. Um outro método vamos conhecer no parágrafo 3.8.3

O seguinte programa tem a estrutura do programa em 2.5.1.

As condições iniciais são:
h = 15 m (largura do rio)
x_a = 15m, y_a = 0 (posição A do alvo)
x_o = 5m, y_o = h (ponto de partida do nadador)
c = 3 m/s (velocidade relativa do nadador, medida em relação ao rio)
V = 5 m/s (velocidade máxima do rio)

O nadador começa sua órbita no ponto P_o (x_o,y_o) e, a cada momento, o vetor $\mathbf{v_{rel}}$ do nadador aponta para o alvo que está em A(x_a,y_a).

Programa 3

```
reset();//perfilo pabólico; gráfico com pontos
h:=15://largura do rio
x0:=5:y0:=h://posição inicial do nadador
xa:=15:ya:=0://posição do alvo no eixo-x
c:=3:V:=5:// c = vel rel.do nadador,V = vel. máxima do rio
V(y):= V-4*V*(y-h/2)^2/h^2://vel.do rio em função de y
d:=sqrt((xa-x(t))^2+(ya-y(t))^2+0.0001):/*distância, se
acrescenta 0.0001, para evitar divisão por zero*/

IVP:={x'(t)=c*(xa-x(t))/d+V(y),y'(t)=c*(ya-y(t))/d,
x(0)=x0,y(0)=y0}:
fields:=[x(t),y(t)]:
ivp:=numeric::ode2vectorfield(IVP, fields):
Y := numeric::odesolve2(ivp): Y(5);//x(5s), y(5s)
```

```
 //Animation

dt:=0.1:imax:=100:
plot(
plot::Point2d(Y(t)[1],Y(t)[2], Color = RGB::Blue,
VisibleFromTo = t..t + 0.99*dt,
PointSize = 2*unit::mm)
$ t in [i*dt $ i = 0..imax],//nadador

plot::Point2d(xa,ya, Color = RGB::Green,
VisibleFromTo = t..t + 0.99*dt,PointSize = 2*unit::mm)
$ t in [i*dt $ i = 0..imax],//alvo

//pontos nao ligados

plot::Point2d(Y(t)[1], Y(t)[2], Color = RGB::Red,
VisibleAfter = t,PointSize = 1*unit::mm)
$ t in [i*dt $ i = 1..imax],
ViewingBox=[0..25,0..15])
```

```
[21.48824241, 2.873975916] // posição depois de 5 segundos
```

(Os pontos "Blue" e "Green" estão faltando no gráfico.)

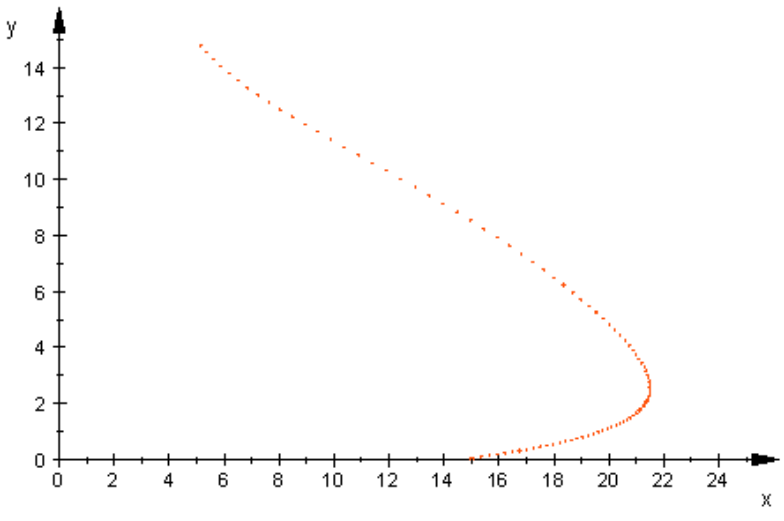

Fig.: 3.8-6

Os pontos da trajetória não foram conectados de propósito, para poder apreciar a variação da velocidade do nadador. Ele somente pode ganhar contra o rio quando a velocidade deste é o suficientemente pequena, o que sucede perto da beira inferior. Então, o nadador é capaz de nadar contra o rio e alcançar o alvo em x_a= 15 m sobre o eixo-X.

Se quisermos obter uma trajetória contínua, é somente necessário substituir as últimas quatro linhas pelas seguintes:

```
plot::Line2d([Y(t - dt)[1], Y(t - dt)[2]],
[Y(t)[1], Y(t)[2]], Color = RGB::Red,
VisibleAfter = t)
$ t in [i*dt $ i = 1..imax],
ViewingBox=[0..25,0..15])
```

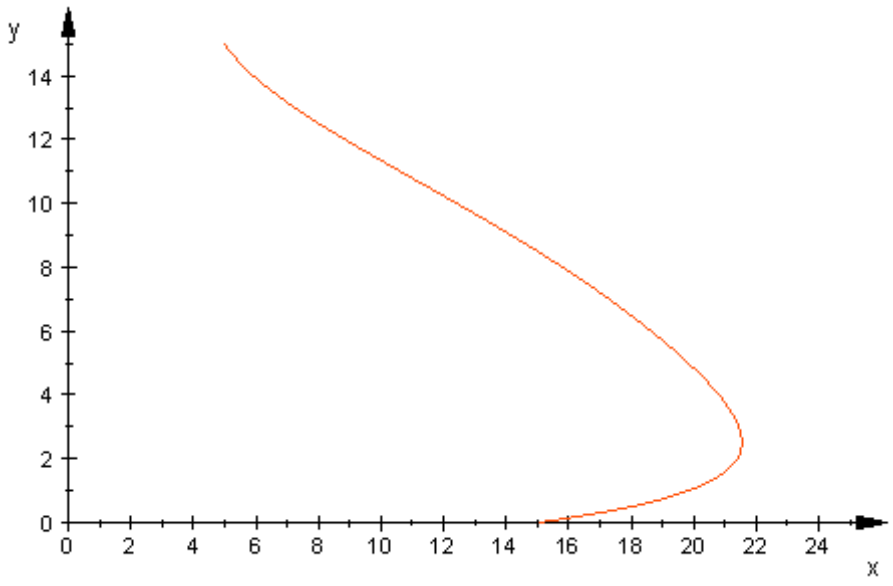

Fig.: 3.8-7

Ataquemos, agora, o problema de perseguição própria: V é zero e o alvo se move com velocidade constante de c_2 = 1 m/s ao longo do eixo-X. O nadador substituimos por um cão, que começa a corrida da perseguição no ponto P_o = (15, 100) com velocidade c_1 = 3 m/s. No programa anterior temos que substituir x_a por x = c_2 t.

Programa 4:

- `reset();//dono e cão são ambos visible`

```
x0:=15:y0:=100:
c1:=3:c2:=1://c1= vel. do cão
d:=sqrt((c2*t-x(t))^2+y(t)^2+0.0001):

IVP:={x'(t)=c1*(c2*t-x(t))/d,
y'(t)=-c1*y(t)/d,x(0)=x0,y(0)=y0}:
fields:=[x(t),y(t)]:
ivp:=numeric::ode2vectorfield(IVP, fields):
Y := numeric::odesolve2(ivp): Y(37.5);
//posição do cão depois de 37.5 segundos

//Animation

dt:=0.5:imax:=100:
plot(

plot::Point2d(Y(t)[1],Y(t)[2], Color = RGB::Blue,
VisibleFromTo = t..t + 0.99*dt,PointSize = 2*unit::mm)
$ t in [i*dt $ i = 0..imax],//cão

plot::Point2d(c2*t,0, Color = RGB::Green,
VisibleFromTo = t..t + 0.99*dt,PointSize = 2*unit::mm)
$ t in [i*dt $ i = 0..imax],//dono

plot::Line2d([Y(t - dt)[1], Y(t - dt)[2]],
[Y(t)[1], Y(t)[2]], Color = RGB::Red,

VisibleAfter = t)

$ t in [i*dt $ i = 1..imax],//cão
plot::Line2d([c2*(t - dt),0],
[c2*t, 0], Color = RGB::Blue,VisibleAfter = t)
$ t in [i*dt $ i = 1..imax],//dono
ViewingBox=[0..50,0..120])
```

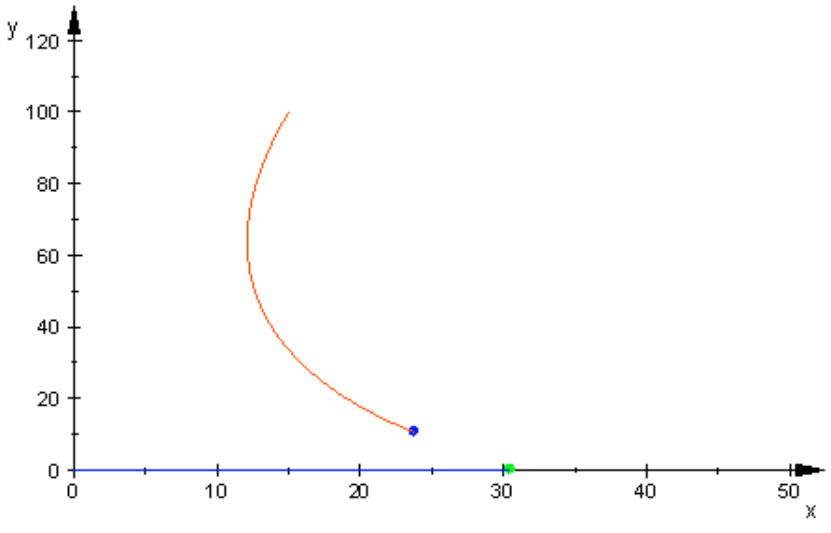

Fig.: 3.8-8

3.8.3 O Método de EULER

MuPAD nos permite integrar uma equação diferencial, e até mesmo um sistema de tais equações, com métodos muito complexos e sofisticados. Os resultados numéricos são sempre bem exatos.

Agora, resulta que em muitos casos não precisamos de resultados hiperexatos pois, estamos mais interessados numa olhada rápida sobre o fenómeno sob estudo.

Para tal análise "de olhada rápida", substituimos uma derivada dx/dt pelo quociente $\Delta x/\Delta t = (x(t + \Delta t)-x(t))/\Delta t$, o que vamos escrever como $(x_{n+1} - x_n)/\Delta t$.

x_{n+1} é a coordenada-x de um corpo no instante $t+\Delta t$ e x_n é a coordenada-x no instante anterior t. O quociente $(x_{n+1} - x_n)/\Delta t$ ı a velocidade média sobre o intervalo Δt.

A velocidade média nada nos diz sobre a velocidade do corpo em um dado instante. Mas, se calcularmos a velocidade média em intervalos de tempo Δt cada vez menores, podemos acercarmos ao valor instantáneo cada vez mais.

A primeira das equações (7) podemos, então, expressar, aproximadamente, por

$(x_{n+1} - x_n)/\Delta t = c\,(x_a - x_n)/d + V(y_n)$, tomando Δt o suficientemente pequeno e sendo a distância dada por

$$d = ((x_a - x_n)^2 + (h/2 - y_n)^2)^{1/2}$$

(Às vezes é mais cómodo escrever a relação anterior da seguinte maneira: $(x_n - x_{n-1})/\Delta t = c\,(x_a - x_{n-1})/d + V(y_{n-1})$. Nas Programas a seguir, vamos utilizar esta escrita.)

Para calcular a trajetória do corpo, começamos o cálculo no ponto (x_0,y_0). Por meio da expressão

$$x_{n+1} = x_n + c\,(x_a - x_n)\,\Delta t/d + V(y_n)\,\Delta t \qquad (8)$$

podemos calcular x_1, já que x_0 é conhecido. Sendo x_1 determinado, podemos calcular $x_2 = x_1 + c\,(x_a - x_1)\,\Delta t/d + V(y_1)\,\Delta t$ e depois x_3 etc. Chama-se este procedimento **iteração** e a equação (8) é uma fórmula de iteração.

Com a coordenada-y procedemos da mesma forma: partindo de y_0, calcula-se y_1, pois y_2 etc., usando como relação de iteração: $y_{n+1} = y_n + c\,(y_a - y_n)\,\Delta t/d$.

Este método para obter a solução aproximativa de uma equação diferencial é chamado de método de EULER. Os métodos numéricos utilizados por MuPAD são, indiscutivelmente, muito mais exatos e complicados. Também são, em certo modo, obscuros, pois não podemos ver o que está sucedendo, enquanto podemos ver ao vivo o que está fazendo o método de EULER, sendo ele bastante transparente. Mais adiante vamos ocupar-nos também de estes métodos "finos" e vamos poder apreciar a simplicidade de nosso método atual.

Para demonstrar esta simplicidade, escrevemos agora um pequeno programa utilizando o exemplo do nadador de acima. Esta vez, colocamos o eixo-x no centro do rio, que tem largura h = 15 m. O nadador parte no ponto Po = (5m, -7.5m). O alvo fica em A(15m, 7.5m). O perfil da velocidade do rio é outra vez parabólico: $V(y) = V - 4V/h^2 \cdot y^2$.

Programa 5

```
reset():Médodo de EULER
x0:=5:xa:=15:v:=5:h:=15:c:=3:
final:=100://número de pontos
DIGITS:=4:
x[0]:=x0:y[0]:=-h/2:dt:=0.1:
coord:=array(1..final,1..3)://array das coordenadas
for n from 1 to final do

d:=((xa-x[n-1])^2+(h/2-y[n-1])^2)^0.5:
x[n]:=x[n-1]+(xa-x[n-1])*c*dt/d +(v-(4*v/h^2)*y[n-1]^2)*dt:
y[n]:=y[n-1]+(h/2-y[n-1])*c*dt/d:

coord[n,1]:= n://elementos do array
coord[n,2]:=x[n-1]:
coord[n,3]:=y[n-1]:

end_for:

for n from 1 to final do
   print(n,x[n],y[n])
end_for:

plot(plot::Point2d([x[n],y[n]])$ n=1..final,
ViewingBox =[0..25,-10..10])
```

```
0, 5, -15/2 // resultados: n, x(n), y(n)
1, 5.166, -7.25
2, 5.366, -7.001
3, 5.596, -6.751
4, 5.856, -6.5
5, 6.144, -6.249
6, 6.46, -5.997
7, 6.8, -5.744
8, 7.165, -5.489
9, 7.552, -5.232
10, 7.961, -4.973
```

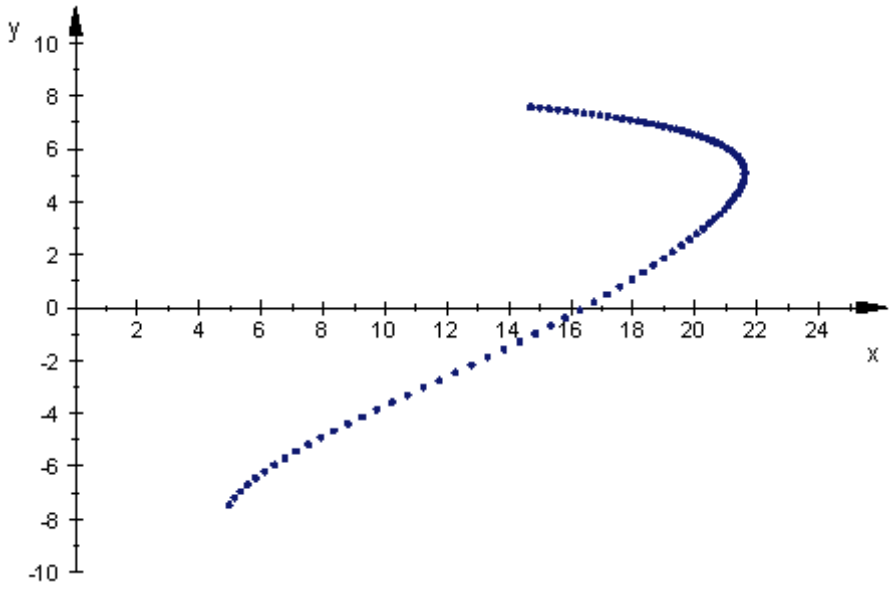

Fig.3.8-9

Esta trajetória deve-se comparar com a figura 3.8-6, que mostra o mesmo movimento, mas utiliza outro sistema de coordenadas.
Aliás, se você quiser, pode entrar os valores constantes interativamente:

```
input("x0=",x0,"xa=",xa,"v=",v,"h=",h,"c=",c,"final",final):
```

No seguinte programa foram introduzidos alguns estruturas de controle como

```
if (float(d) < 0.1) then final :=n;
```

e `if (n=0) or(modp(n,10)=0) then`

assim como os pontos da partida e da posição final.

O primeiro `if ...then` junto com `break` interrompe os cálculos quando a distância é menor do que 0.1.

O segundo comando condicional faz com que somente os valores iniciais e cada décimo resultado sejam impressos.

Programa 6

```
reset()://Método de EULER
xo:=5:xa:=15:v:=5:h:=15:c:=3:
final:=200://número de pontos
DIGITS:=4:

x[0]:=h:y[0]:=-h/2:dt:=0.1:

coord:=array(1..final,1..3)://array das coordenadas

for n from 1 to final do

  d:=((xa-x[n-1])^2+(h/2-y[n-1])^2)^0.5:
x[n]:=x[n-1]+(xa-x[n-1])*c*dt/d +(v-(4*v/h^2)*y[n-1]^2)*dt:
y[n]:=y[n-1]+(h/2-y[n-1])*c*dt/d:

coord[n,1]:= n:
coord[n,2]:=x[n-1]:
coord[n,3]:=y[n-1]:

if (float(d) < 0.1) then final :=n;

print("FINAL, n = ",n); break;

  end_if:
end_for:

for n from 0 to final do
if (n=0) or (modp(n,10)=0) then
print(n,x[n],y[n]);

  end_if://cada décima coordenada será impressa
end_for:


//gráfico

plot(
plot::Point2d([x[n],y[n]])$ n=0..final,
plot::Point2d([x[final],y[final]],
PointSize =2*unit::mm,Color=RGB::Green),
```

```
    plot::Point2d([x[0],y[0]],PointSize =2*unit::mm,
    Color=RGB::Red),
   plot::Text2d("Partida",[x[0],y[0]],
   HorizontalAlignment=Right),
   plot::Text2d("Alvo",[x[final],y[final]],
   HorizontalAlignment=Right),
    ViewingBox =[0..25,-10..10])
```

```
  "FINAL, n = ", 94

0, 5, -15/2
  10, 7.961, -4.973
  20, 12.83, -2.224    etc.
```

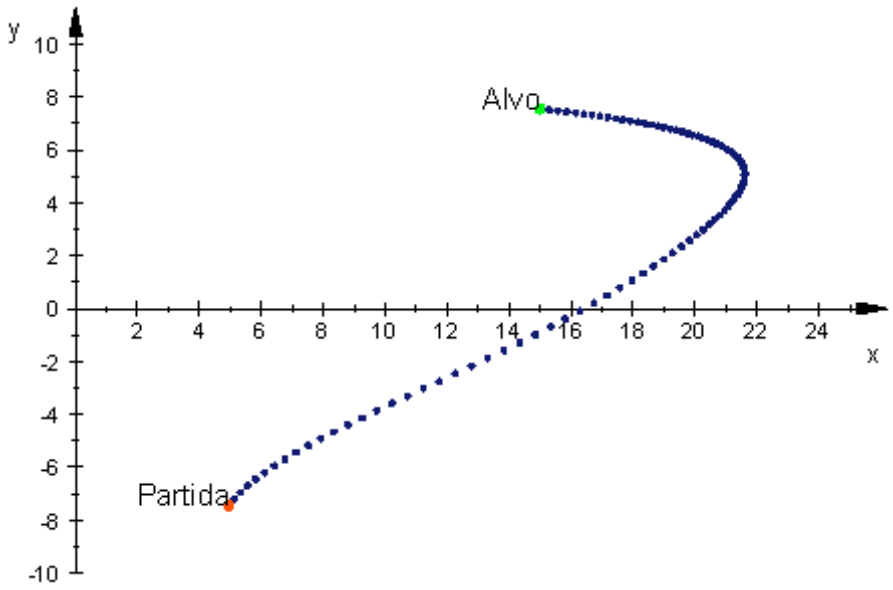

Fig.: 3.8-10

3.8.4 Com lápis e papel

Um barco sai do ponto A(0,0), para cruzar um rio que corre com velocidade constante e uniforme de V_o = 5 ft/s. A largura do rio é de 100 ft. O barco tem, em relação ao rio, uma velocidade de c = 10 ft/s. (A unidade ft/s nos vai proporcionar valores numéricos bem simples.)

Quanto tempo o barco vai gastar, para chegar ao ponto C(50,100) se utilizar uma trajetória linear entre A e C?

Solução:

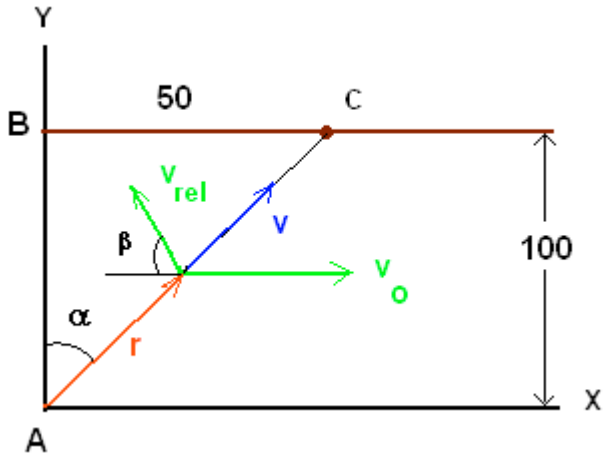

Fig.: 3.8-11

A velocidade absoluta **v** deve ter a direção do vetor **AC** que há um comprimento de 111,80 ft. A velocidade relativa do barco é $\mathbf{v_{rel}}$ = -10 cosβ **i** + 10 senβ **j.** A velocidade absoluta vem dada por

$$\mathbf{v} = v \operatorname{sen}\alpha\, \mathbf{i} + v \cos\alpha\, \mathbf{j} = -10 \cos\beta\, \mathbf{i} + 10 \operatorname{sen}\beta\, \mathbf{j} + 5\mathbf{i}$$

Colocando valores numéricos, chegamos às seguintes equações para v:

$v \cdot 0{,}447 = -\cos\beta + 5$ e $v \cdot 0{,}894 = 10 \cdot \operatorname{sen}\beta$.

Elevando ao quadrado e somando, obtemos a seguinte equação para v:

$$v^2 - 4{,}47 \cdot v + 25 = 100.$$

A solução desta equação é v = 11,18 ft/s. O tempo buscado é t = 111,80/11,18 = 10 s.