Projeto 4

Juliana Naomi Yamauti Costa, nºUSP 10260434

20 Maio 2020

As equações clássicas do movimento, provindas das Leis de Newton, governam a dinâmica de sistemas muito grandes até aqueles muito pequenos (dentro de seus limites, quando flutuações quânticas podem ser relevadas).

Soluções analíticas exatas existem apenas em sistemas simples, e então foram estudadas técnicas de resolução de **equações diferenciais** para o estudo de sistemas dinâmicos clássicos, incluindo os caóticos.

Umas das formas de resolver equações diferenciais é através do algorítmo **Runge-Kutta** (RK), o qual pode ser utilizado para um considerável conjunto de equações diferenciais. RG de segunda ordem corresponde a regra do ponto médio utilizado em integração numérica

1 Subprojeto A

Aqui testaremos o método em uma dimensão. As primeiras linhas destacadas referem-se as condições iniciais aplicadas, e as restantes à aplicação da equação 5 (indicada nas instruções do projeto).

Como os subprojetos seguintes exigiriam a criação de vários arquivos, neste código eles já são gerados automaticamente e iniciam-se no número 10. Para indicar o que cada arquivo contém, foi incluído um *print* (linha 20) o qual indica as condições da simulação de cada um.

tarefaA-10260434.f90

```
   Program Subprojeto_A
2
   IMPLICIT REAL*8 (a-h,o-z) !dupla precisão
4  dimension at(1:3), v(1:2) !vetores das variáveis
   character(len=70) fn
6
   at = (/0.01,0.001,0.0001/)        !avanço temporal
8  v = (/0.d0,10.d0/)                        !velocidade inicial
   ac = -10.d0                                        !aceleração
10
   ifilenum = 10
12
   do i=1, 3
14         do j=1, 2
                   tv = 0                        !tempo inicial
16                 ty = 0
                   ypos = 100.d0         !posição inicial
18                 vel = v(j)                    !velocidade inicial
                   ifilenum = ifilenum + 1         !numeração dos arquivos
20                 write(fn,fmt='(i0,a)') ifilenum,'.dat'
```

```
                print*, ifilenum, 'v=', v(j), 'at=', at(i)
                open(unit=ifilenum,file=fn)
                do while (ypos .ge. 0.d0)        !para quando corpo chega
                 ↪  ao solo
                        write(ifilenum,*) ty, ypos, tv, vel
                        ty = ty + at(i)
                        tv = ty - at(i)/2
                        if (tv .eq. (at(i)/2.0)) then        !ajuste da
                         ↪  primeira iteração
                                vel = v(j) + (at(i)/2.d0)*ac
                        else
                                vel = vel + at(i)*ac
                        end if
                        ypos = ypos + at(i)*vel
                end do
                close(ifilenum)
        end do
end do
End Program Subprojeto_A
```

1.1 Resultados

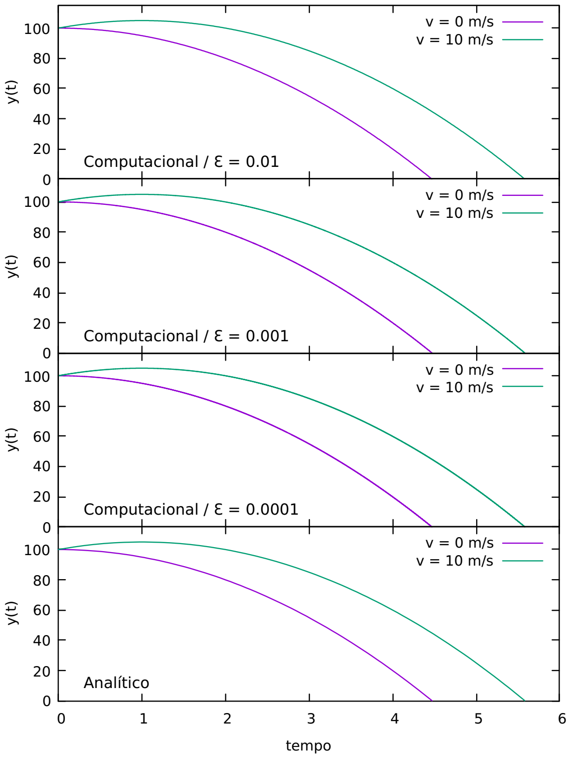
Os gráficos abaixo comparam as condições aplicadas e o comportamento das curvas, tanto da velocidade quanto da posição no eixo y.

Como se trata de um sistema simples, para obtermos a curva analítica basta utilizarmos as Leis de Newton correspondentes ao Movimento Uniformemente Acelerado (MUV).

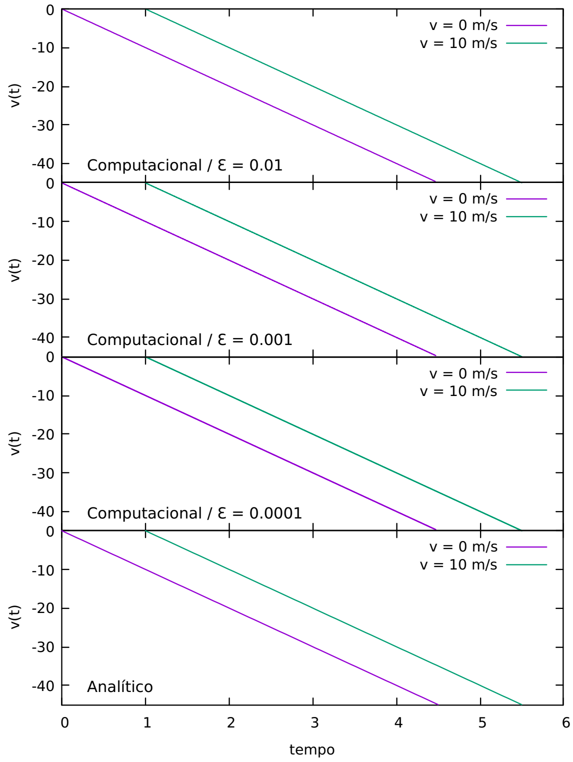
$$y = y_0 + v_0 \cdot t + \frac{a \cdot t^2}{2} \tag{1}$$

$$v = v_0 + a \cdot t \tag{2}$$

Portanto, deveríamos esperar uma curva parabólica (posição) e outra linear (velocidade).



(a) Comparação entre as curvas de 'y(t) *vs* t' obtidas numericamente com a curva analítica.



(b) Comparação entre as curvas de 'v(t) *vs* t' obtidas numericamente com a curva analítica.

Figura 1: Curvas da posição e velocidade com variação temporal.

Podemos verificar que ambas as curvas (posição e velocidade) correspondem à curva analítica, e que a mudança do incremento temporal não altera de forma significativa o comportamento das mesmas (afinal, a física é mesma).

No entanto, podemos comparar os valores do tempo obtidos em cada simulação com o resultado analítico e verificar como se comporta a precisão em cada caso:

v (m/s)		ε		EXATO
	0.01	0.001	0.0001	
0	4.46999990	4.47200021	4.47209988	4.47213595
10	5.57999987	5.58200026	5.58249985	5.58257569

Como esperado, a precisão aumenta conforme a discretização do tempo aumenta.

2 Subprojeto B

Agora incluímos a resistência do ar, portanto, a aceleração é variável. Visto isso, algumas alterações foram feitas no código a fim de computar a influência da aceleração nas variáveis de posição e velocidade.

tarefaB-10260434.f90

```fortran
Program Subprojeto_B

IMPLICIT REAL*8 (a-h,o-z) !dupla precisão
dimension at(1:3), v(1:2), res(1:2)        !vetores das variáveis
character(len=70) fn

at = (/0.01,0.001,0.0001/)        !avanço temporal
v = (/0.d0,10.d0/)                       !velocidade inicial
res = (/0.1,0.01/)                       !resistência do ar

ifilenum = 10

do i=1, 3
        do j=1, 2
                do k = 1, 2
                        tv = 0                          !tempo inicial
                        ty = 0
                        ypos = 100.d0         !posição inicial
                        acel = -10.d0         !aceleração         inicial
                        vel = v(j)                 !velocidade inicial
                        ifilenum = ifilenum + 1         !numeração dos
                        ↪ arquivos
                        print*,  'v=', v(j), 'res=', res(k),'at=', at(i),
                        ↪ ifilenum
                        write(fn,fmt='(i0,a)') ifilenum,'.dat'
                        open(unit=ifilenum,file=fn)
                        do while (ypos .ge. 0.d0)
                                write(ifilenum,*) ty, ypos, tv, vel
                                ty = ty + at(i)
                                tv = ty - at(i)/2.d0
                                ac = acel - res(k)*vel         !aceleração
                                ↪ variável devido a resistência do ar
                                if (tv .eq. (at(i)/2.d0))
                                ↪ then          !ajuste da primeira
                                ↪ iteração
                                vel = v(j) + (at(i)/2.d0)*ac
                        else
                                vel = vel + at(i)*ac
                        end if
                                ypos = ypos + at(i)*vel
                        end do
                        close(ifilenum)
                end do
        end do
```

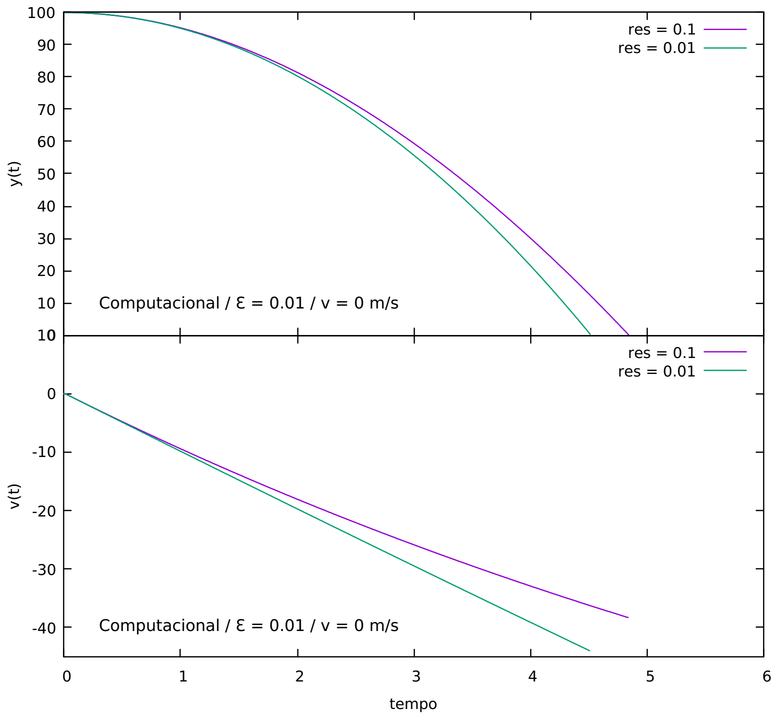
```
end do
End Program Subprojeto_B
```

2.1 Resultados

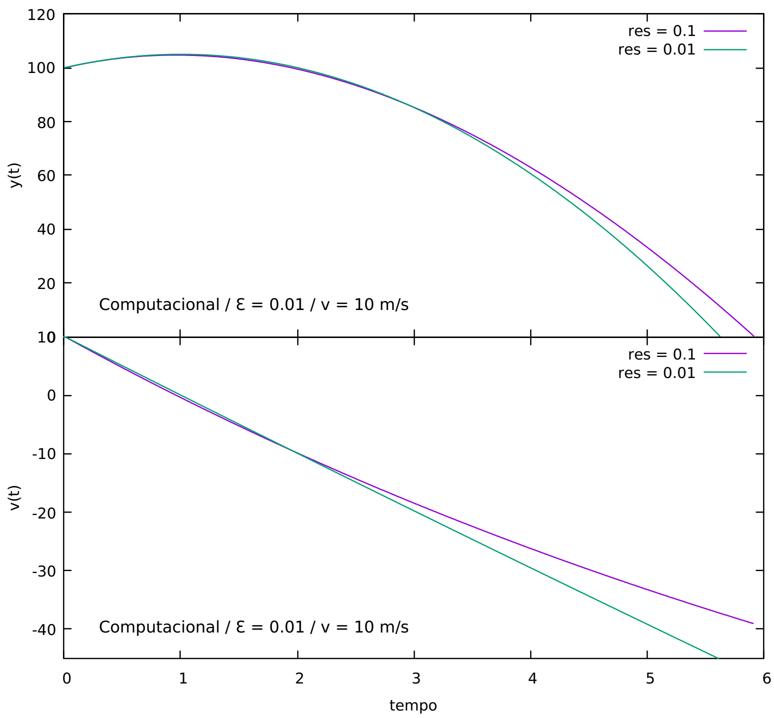
Ainda que a resistência do ar possa ser negligenciada quando muito pequena, ela produz efeitos significativos no movimento dos corpos.

Assim como no caso anterior, a aceleração gera a variação da velocidade, no entanto, neste caso a aceleração é variável, produzindo uma alteração não linear da velocidade. Também notamos que o tempo de queda aumenta significativamente, uma vez que a resistência do ar atua no corpo como uma força resistiva.

Podemos enxergar essas observações nos gráficos a seguir:



(a) Curvas da posição e velocidade com variação no tempo.



(b) Curvas da posição e velocidade com variação no tempo.

Figura 2: Resultados das simulações ve posição e velocidade com inclusão da resistência do ar no movimento 1D.

Podemos verificar que quanto maior a resistência do ar, mais visível é seu efeito, em que a velocidade apresenta um comportamento menos linear (curva em lilás).

Agora analisando a velocidade dos corpos, a expressão analítica para a velocidade estacionária é dada por:

$$v_{limite} = -\frac{g}{\gamma} \tag{3}$$

Para os casos fornecidos, com $\gamma = 0.1$, $v_{limite} = -100$ m/s e, com $\gamma = 0.01$, $v_{limite} = -1000$ m/s. Para os casos solicitados, vimos que os corpos não atingiram a velocidade limite.

Para atingi-la, podemos alterar a altura de lançamento, implicando em maior tempo de queda e aumento da velocidade/aceleração. A seguinte simulação foi feita para $h_0 = 2000$ m.

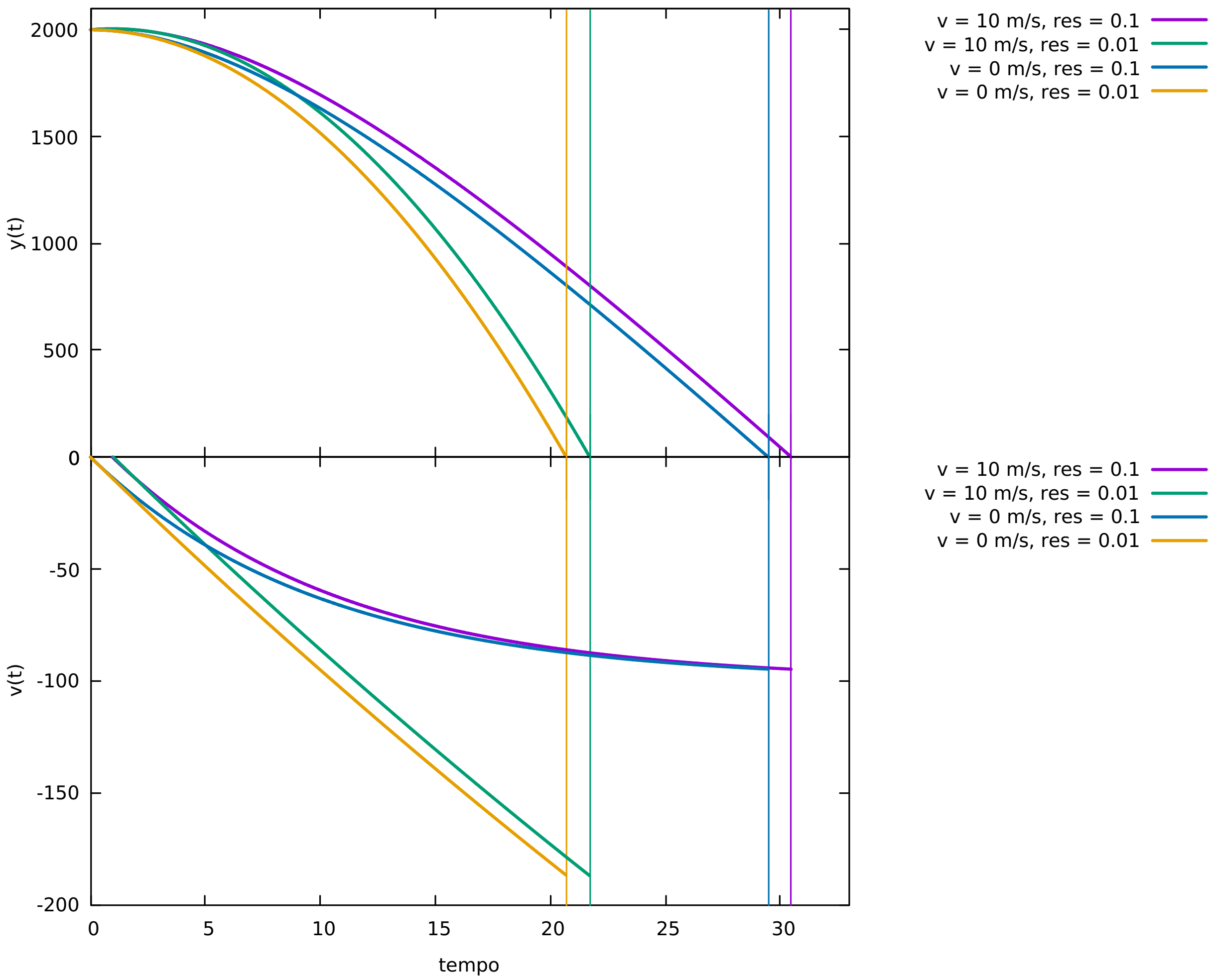


Figura 3: Verificação da velocidade estacionária.

Para os casos em que $res = 0.01$, obtivemos que a velocidade chega próxima ao seu limiar, com valor $vel = -94.780213402759060$ m/s. A curva da velocidade já apresenta estar se aproximando a um plato, assim como a posição parece se tornar linear - uma vez que que a aceleração se aproxima a zero e o movimento se torna uniforme.

3 Subprojeto C

Com a inclusão de um ângulo de lançamento do chiclete, o problema agora ocupa duas dimensões. Assim, foram íncluídos o armazenamento da variável 'x' e as componentes da velocidade. O avanço temporal foi fixado e foram analisadas as velocidades iniciais de 10 m/s e 20 m/s em cada ângulo.

tarefaC-10260434.f90

```
Program Subprojeto_C

IMPLICIT REAL*8 (a-h,o-z) !dupla precisão
dimension vel(1:2), alfa(1:3)        !vetores das variáveis
character(len=70) fn

pi = 4.d0*datan(1.d0)
vel = (/10.0,20.0/)          !velocidade inicial
alfa = (/pi/4.d0,0.d0,-pi/4.d0/)        !angulo inicial

at = 0.001                   !avanço temporal
y0 = 100.d0                   !y inicial
x0 = 0.d0                           !x inicial
ac = -10.d0                   !aceleração        inicial
ifilenum = 10          !numeração de arquivos

do i=1, 2
        do j=1, 3
                tv = 0                        !tempo inicial
                tp = 0
                ypos = y0         !posição inicial
                xpos = x0
                velx = vel(i)*dcos(alfa(j))        !componente x da velocidade
                vely = vel(i)*dsin(alfa(j))        !componente y da velocidade
                ifilenum = ifilenum + 1
                print*,  'v=', vel(i), 'alfa=', alfa(j), ifilenum
                write(fn,fmt='(i0,a)') ifilenum,'.dat'
                open(unit=ifilenum,file=fn)
                do while (ypos .ge. 0.d0)
                        write(ifilenum,*) tp, ypos, xpos, tv
                        tp = tp + at                              !tempo posição
                        tv = tp - at/2.d0                         !tempo velocidade
                        if (tv .eq. (at/2.d0)) then        !ajuste da primeira iteração
                                vely = vely + (at/2.d0)*ac
                        else
                                vely = vely + at*ac
                        end if
                        ypos = ypos + at*vely
```

```
                        xpos = xpos + at*velx        !eixo x velocidade constante
                end do
                close(ifilenum)
        end do
end do
End Program Subprojeto_C
```

3.1 Resultados

As curvas deveriam se manter parabólicas como nos subprojetos anteriores, mudando apenas alguns parâmetros como resultado do ângulo de lançamento, o qual altera o valor das velocidades iniciais em cada componente.

$$y = y_0 + v_0 \cdot sen(\alpha) \cdot t + \frac{a \cdot t^2}{2} \tag{4}$$

$$x(t) = x_0 + v_x \cdot t \tag{5}$$

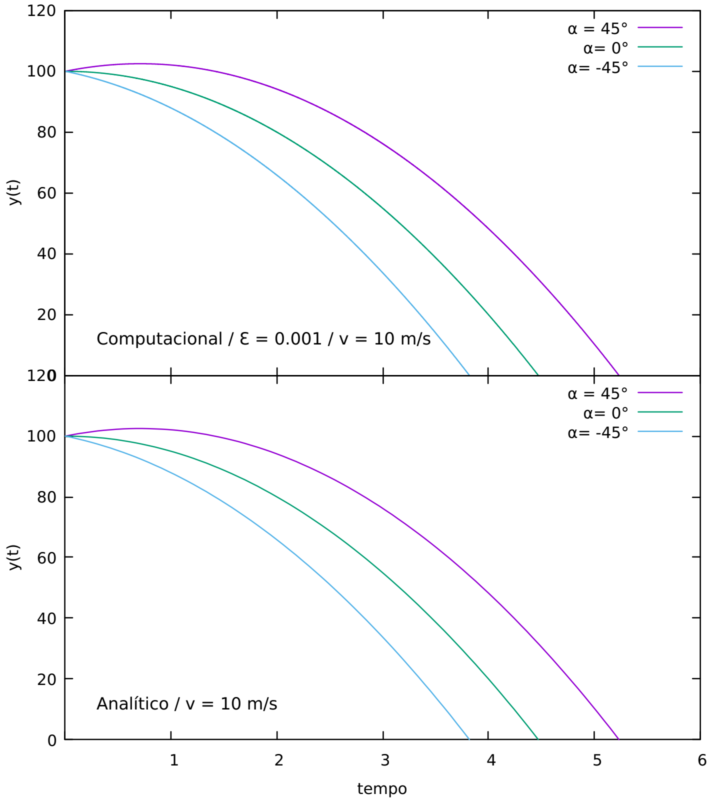
$$v_x = v_0 \cdot cos(\alpha) \tag{6}$$

$$v_y(t) = v_0 \cdot sen(\alpha) + a \cdot t \tag{7}$$

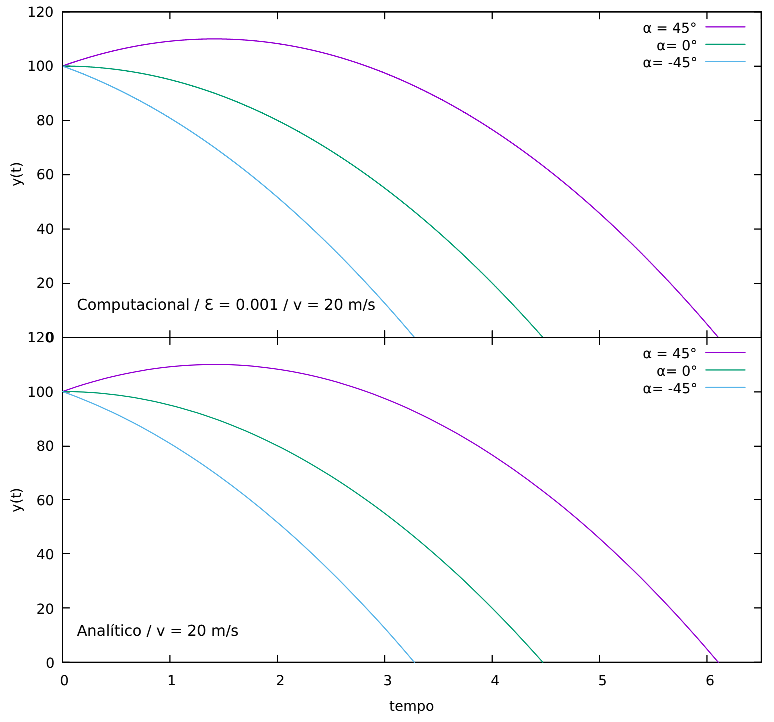
$$y(x) = y_0 + x \cdot tg(\alpha) + \frac{a}{2}\left(\frac{x}{v_0 \cdot cos(\alpha)}\right)^2 \tag{8}$$

As equações 4, 5 e 8 foram utilizadas para plotar as funções analíticas de cada simulação e compará-las aos resultados obtidos. Cada variável pode ser plotada individualmente pois o movimento horizontal e vertical são independentes entre si.

Todos corresponderam às funções analíticas.

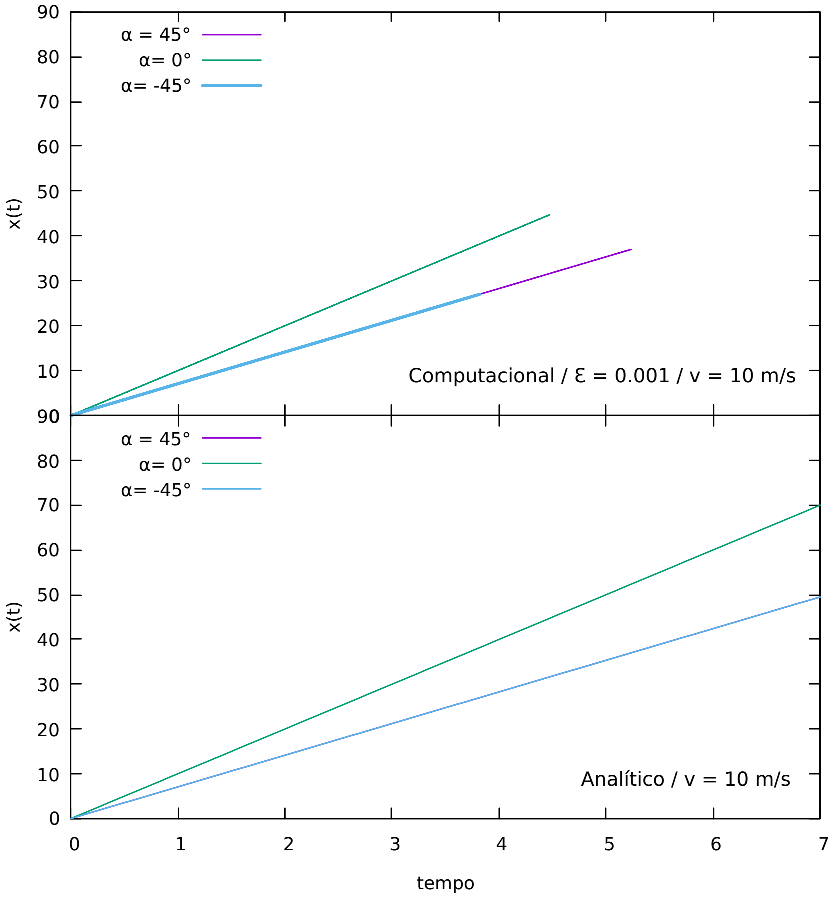


(a) Comparação entre as curvas de 'y(t) *vs* t' obtidas numericamente com a curva analítica $v = 10$ m/s.

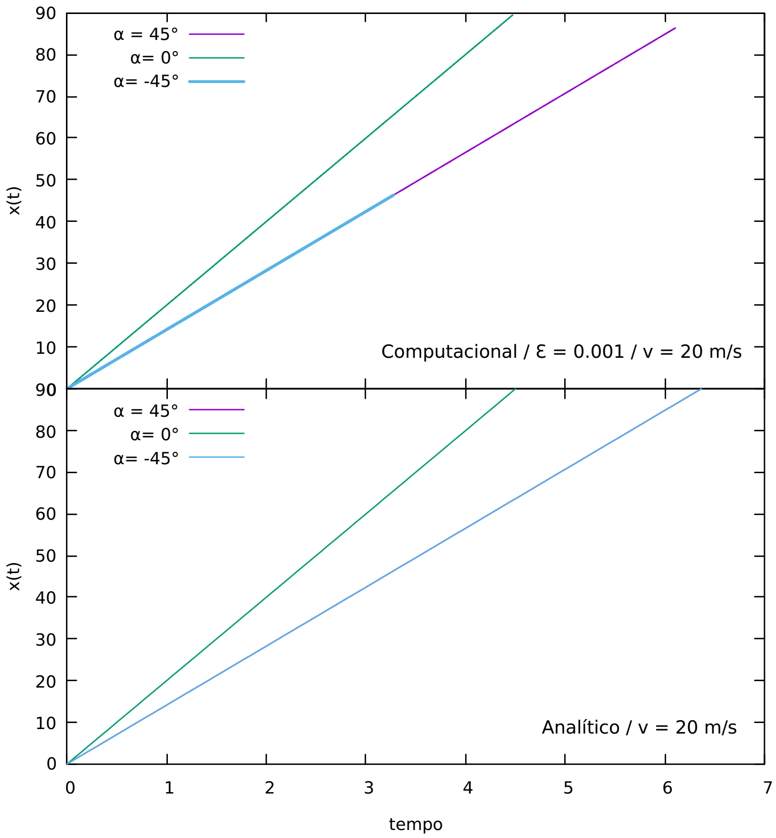


(b) Comparação entre as curvas de 'y(t) *vs* t' obtidas numericamente com a curva analítica para $v = 20$ m/s.

Figura 4: Curvas da posição vertical do corpo com variação temporal.

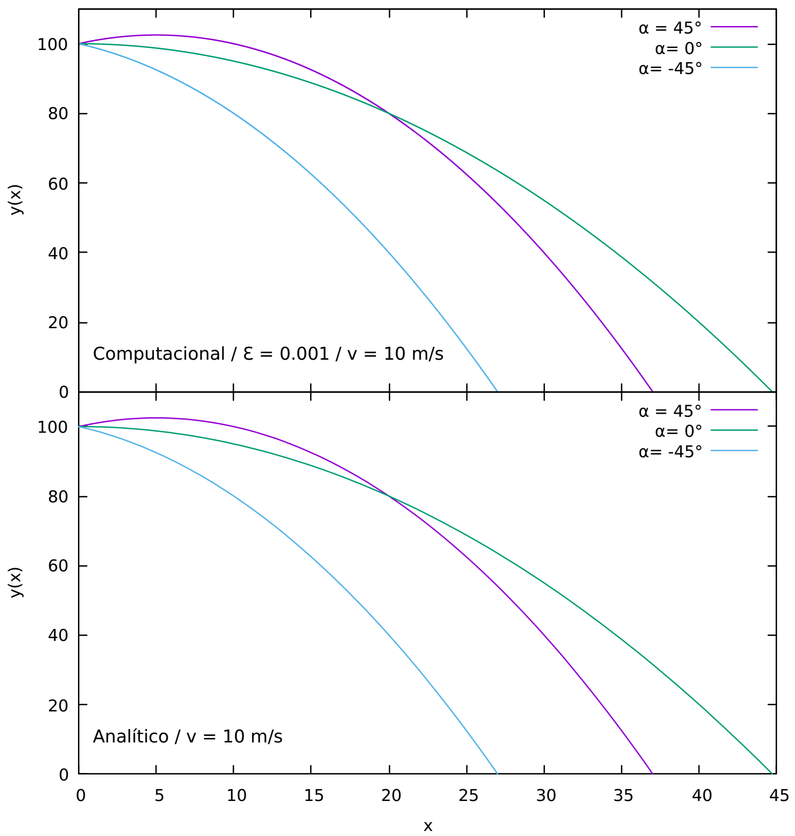


(a) Comparação entre as curvas de 'y(t) *vs* t' obtidas numericamente com a curva analítica $v = 10$ m/s.

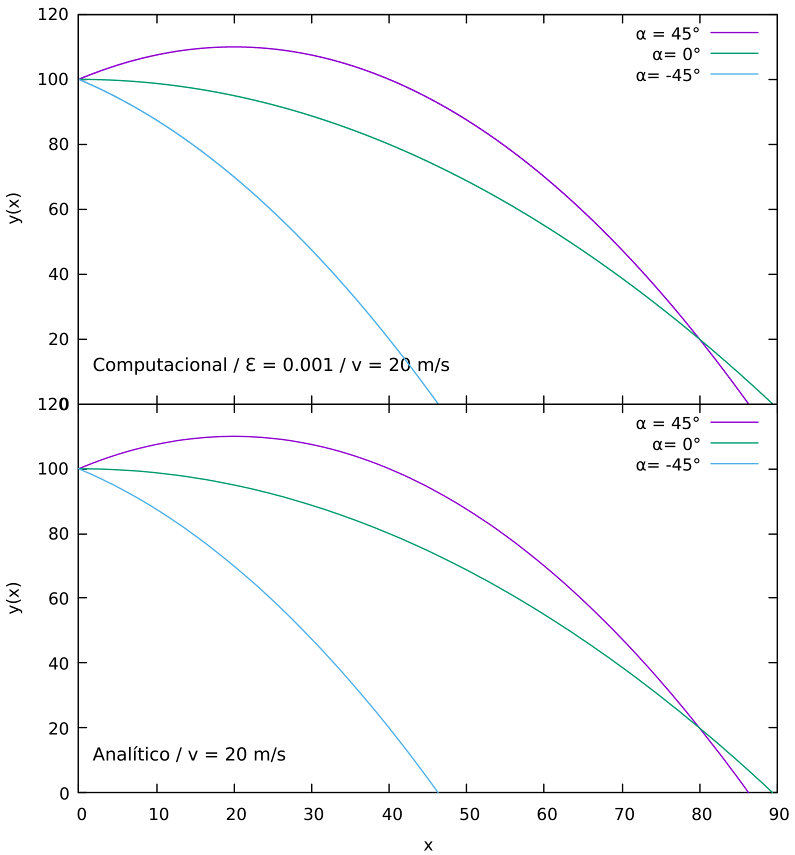


(b) Comparação entre as curvas de 'x(t) *vs* t' obtidas numericamente com a curva analítica para $v = 20$ m/s.

Figura 5: Curvas da posição horizontal do corpo com variação temporal.



(a) Curvas da posição do corpo no espaço para $v = 10$ m/s.



(b) Curvas da posição do corpo no espaço para $v = 20$ m/s.

Figura 6: Gráficos da posição do projétil.

4 Subprojeto D

Aqui temos o mesmo caso anterior com a adição da resistência do ar.

tarefaD-10260434.f90

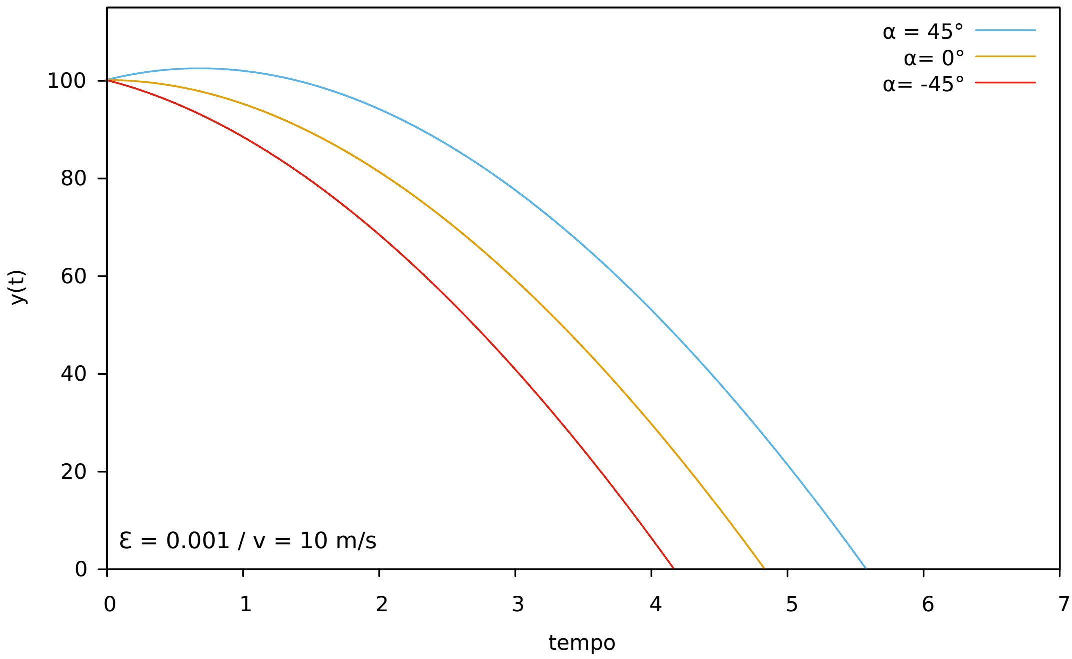
```fortran
Program Subprojeto_D

IMPLICIT REAL*8 (a-h,o-z) !dupla precisão
dimension vel(1:2), alfa(1:3) !vetores das variáveis
character(len=70) fn

pi = 4.0*datan(1.d0)
vel = (/10.0,20.0/)         !velocidade inicial
alfa = (/pi/4.d0,0.d0,-pi/4.d0/) !angulo inicial

at = 0.001                !avanço temporal
y0 = 100.0                !y inicial
x0 = 0                        !x inicial
acely = -10.0         !aceleração         inicial
res =  0.1                !resistência do ar
ifilenum = 10         !numeração de arquivos

do i=1, 2
        do j=1, 3
                tv = 0                        !tempo inicial
                tp = 0
                ypos = y0         !posição inicial
                xpos = x0
                velx = vel(i)*cos(alfa(j))        !componente x da velocidade
                vely = vel(i)*sin(alfa(j))        !componente y da velocidade
                ifilenum = ifilenum + 1
                print*, ifilenum, 'v=', vel(i), 'alfa=',alfa(j)
                write(fn,fmt='(i0,a)') ifilenum,'.dat'
                open(unit=ifilenum,file=fn)
                do while (ypos .ge. 0)
                        write(ifilenum,*) tp, ypos, xpos, tv
                        tp = tp + at                        !tempo posição
                        tv = tp - at/2.0                !tempo velocidade
                        acx = -res*velx                 !aceleração variável devido a resistência do ar
                        acy = acely - res*vely
                        if (tv .eq. (at/2.0)) then        !ajuste da primeira iteração
                                vely = vely + (at/2.0)*acy
                                velx = velx + (at/2.0)*acx
                        else
                                vely = vely + at*acy
                                velx = velx + at*acx
                        end if
                        ypos = ypos + at*vely         !eixo y
```

```
                        xpos = xpos + at*velx        !eixo x
                end do
                close(ifilenum)
        end do
end do
End Program Subprojeto_D
```

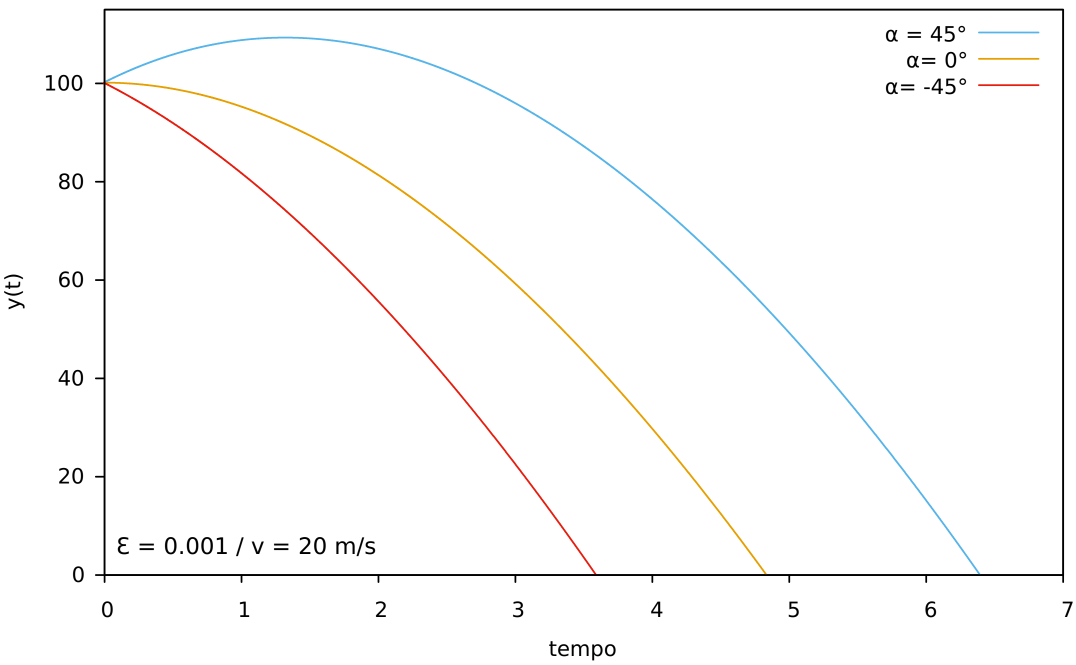
4.1 Resultados

Como nos casos anteriores em que há resistência do ar, a aceleração gera efeitos sobre as componentes da velocidade (tanto horizontal quanto vertical), portanto, ambas possuem comportamentos não lineares. Para melhor comparação entre cada caso, foram mantidas as mesmas escalas no gráficos correspondentes entre si.

Sendo o caso 2D, horizontalmente o movimento é mais lento, assim como no eixo y. Desta forma, vemos que o tempo de queda é maior quando comparado ao movimento sem resistência do ar.

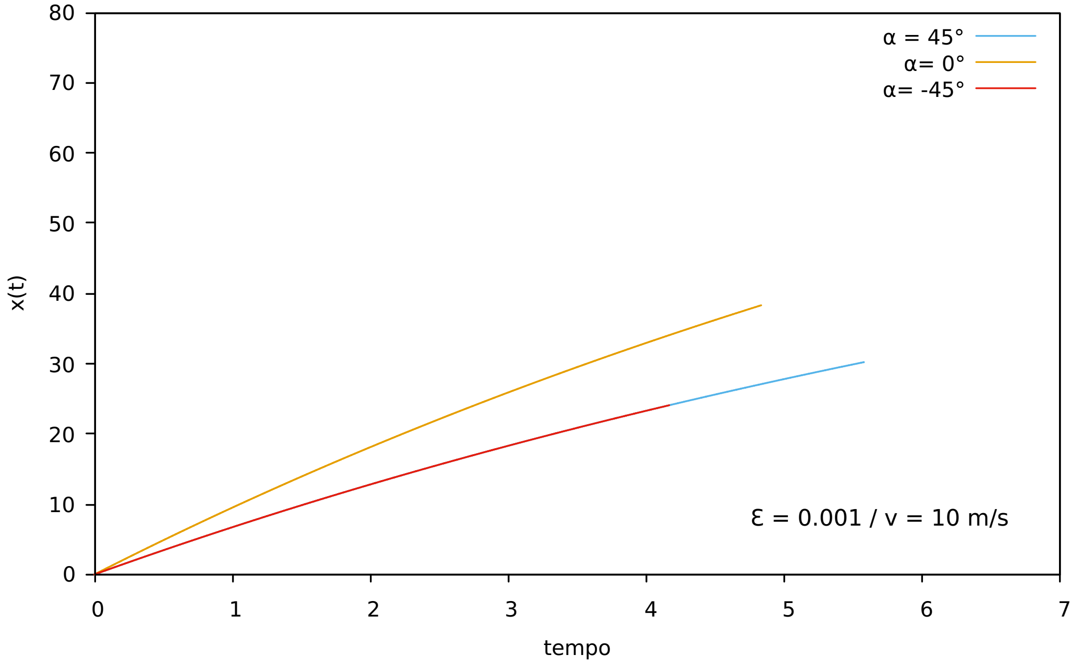


(a)

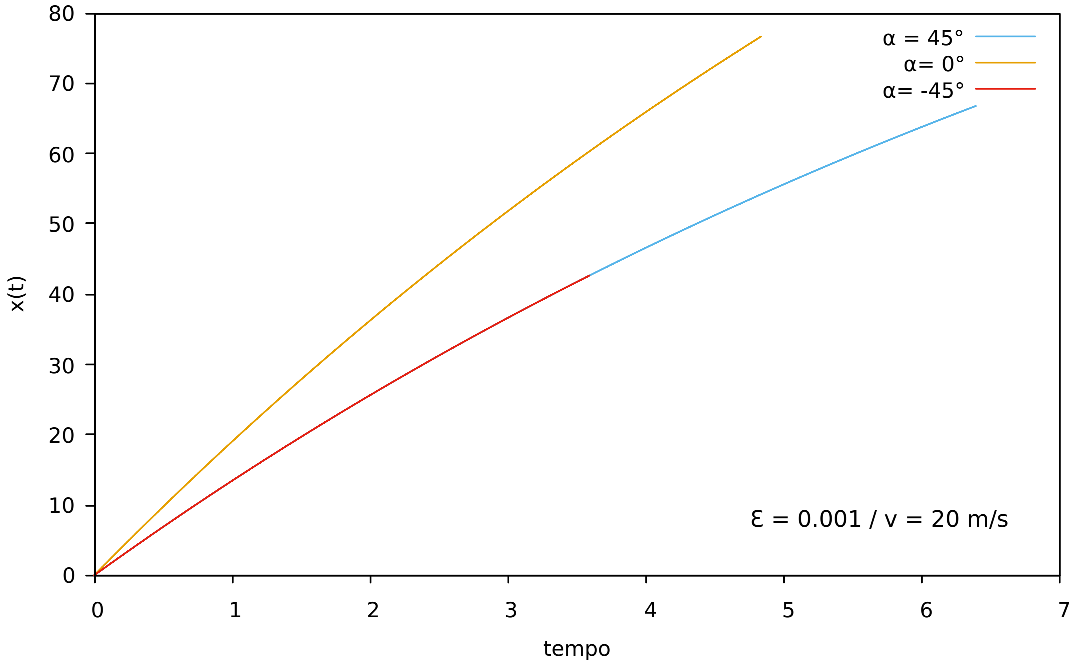


(b)

Figura 7: y vs t

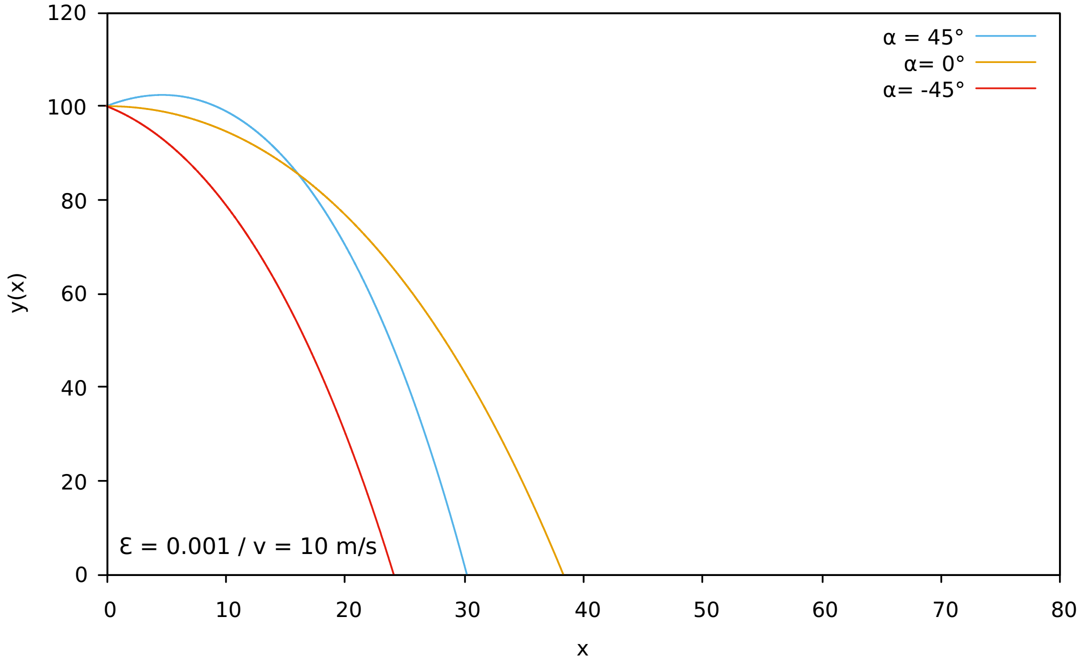


(a)

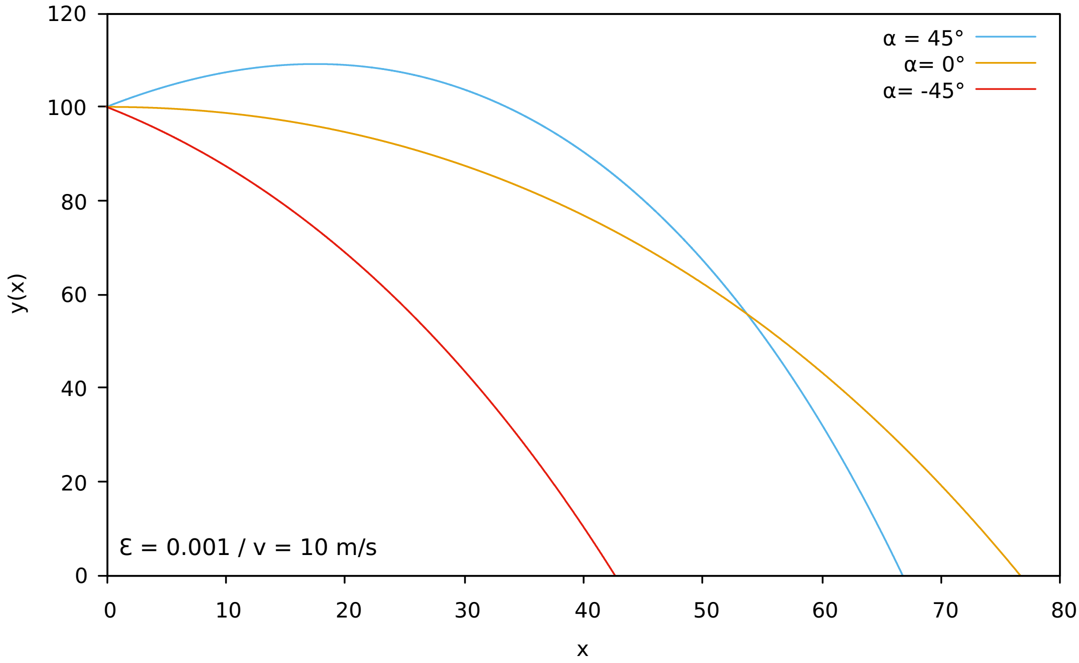


(b)

Figura 8: x(t) *vs* t



(a)



(b)

Figura 9: y(x) vs x

5 Subprojeto E

Aproximando o problema um pouco mais a realidade, quando o corpo chega próximo ao chão ele perde uma fração de energia e continua o movimento. Para representar isso computacionalmente, quando o corpo chega muito próximo à superfície sua velocidade vertical é invertida e o algorítmo continua a ser aplicado, enquanto a velocidade horizontal mantém seu sentido e a mesma fração de energia é retirada.

Foram testados dois valores, $dx = 0.3$ e $dx = 0.5$. Como parada, foi estabelecida uma altura máxima de proximidade do solo.

tarefaE-10260434.f90

```fortran
Program Subprojeto_E

dimension vel(1:2), alfa(1:3), dxy(1:2)
character(len=70) fn

pi = 4.0*atan(1.0)
vel = (/10.0,20.0/)        !velocidade inicial
alfa = (/pi/4.0,0.0,-pi/4.0/)
dxy = (/0.3,0.5/)

at = 0.001                 !avanço temporal
y0 = 100.0                 !y inicial
x0 = 0                         !x inicial
ac = -10.0                 !aceleração        inicial
ifilenum = 10         !numeração de arquivos
dif  = 0.001
hmax = 1

do i=1, 2
        do k=1, 2
                do j=1, 3
                        tv = 0                         !tempo inicial
                        tp = 0
                        ypos = y0         !posição inicial
                        xpos = x0
                        velx = vel(i)*cos(alfa(j))
                        vely_new = vel(i)*sin(alfa(j))
                        ifilenum = ifilenum + 1
                        print*, ifilenum, 'v=', vel(i), 'alfa=',alfa(j), 'dxy=', dxy(k)
                        write(fn,fmt='(i0,a)') ifilenum,'.dat'
                        open(unit=ifilenum,file=fn)
                        do while (ypos .ge. 0)
                                write(ifilenum,*) tp, ypos, xpos, tv, vely_old, velx
                                tp = tp + at             !tempo posição
                                tv = tv + at/2.0                          !tempo velocidade
```

```
                if (tv .eq. (at/2.0)) then
                    vely_old = vely_new + (at/2.0)*ac
                else
                    vely_old = vely_new + at*ac
                end if
                ypos = ypos + at*vely_old
                xpos = xpos + at*velx        !eixo x velocidade constante

                if (ypos .lt. 0) then
                    ypos = 0
                end if

                if (((vely_new*vely_old) .lt. 0) .and. (ypos .le. hmax)) then
                    exit !a altura é máxima quando velocidade muda de sinal
                end if

                if (ypos .lt. dif) then
                    vely_old = -vely_old + dxy(k)*vely_old
                    velx = velx - velx*dxy(k)
                    ypos =  ypos + at*vely_old
                    xpos = xpos + at*velx
                end if

                vely_new = vely_old
            end do
            close(ifilenum)
        end do
    end do
end do
End Program Subprojeto_E
```

5.1 Resultados

Analisando os gráficos da velocidade percebemos que há uma quebra de continuidade, correspondentes ao momento em que o corpo atinge o chão e perde uma fração de energia, evidente tanto em x quanto em y. A queda em x é linear pois o movimento horizontal é uniforme, já que não há resistência do ar.

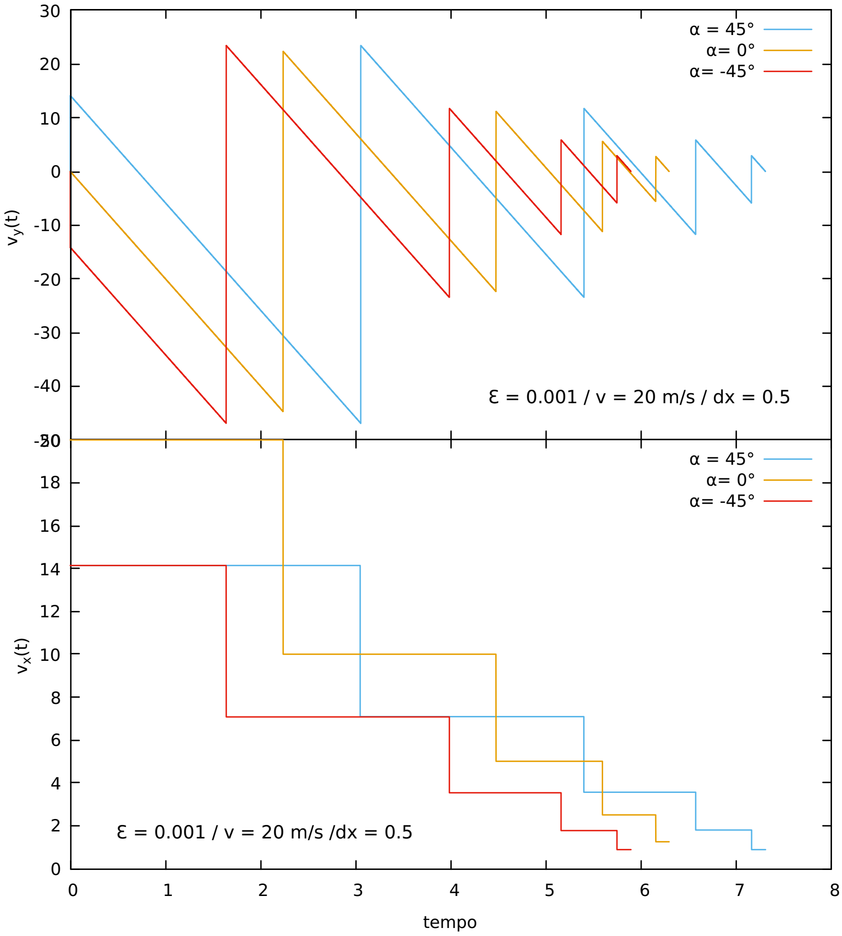
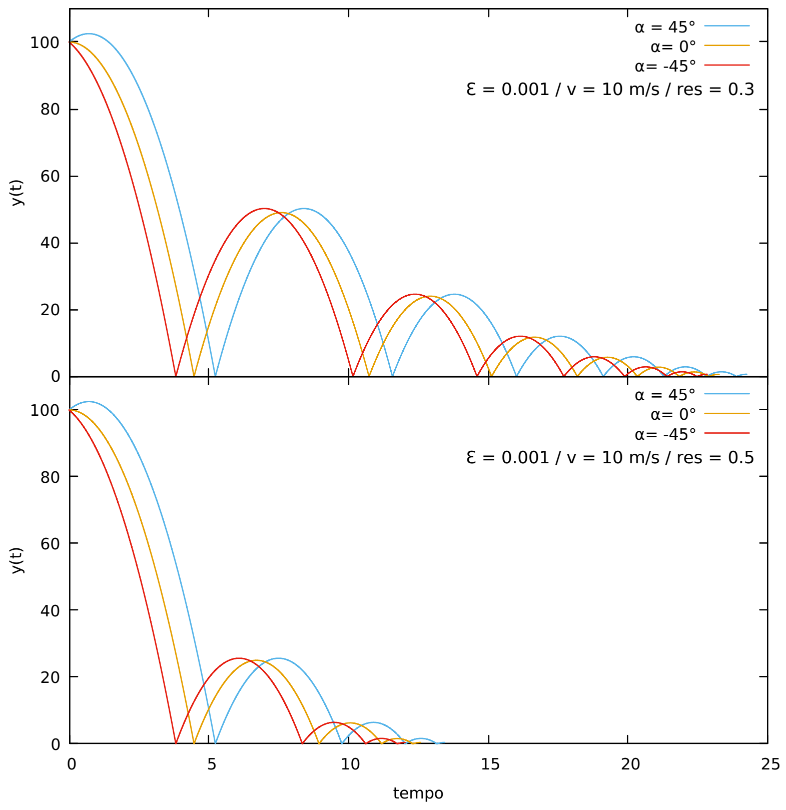


Figura 10: Comportamento das componentes da velocidade.

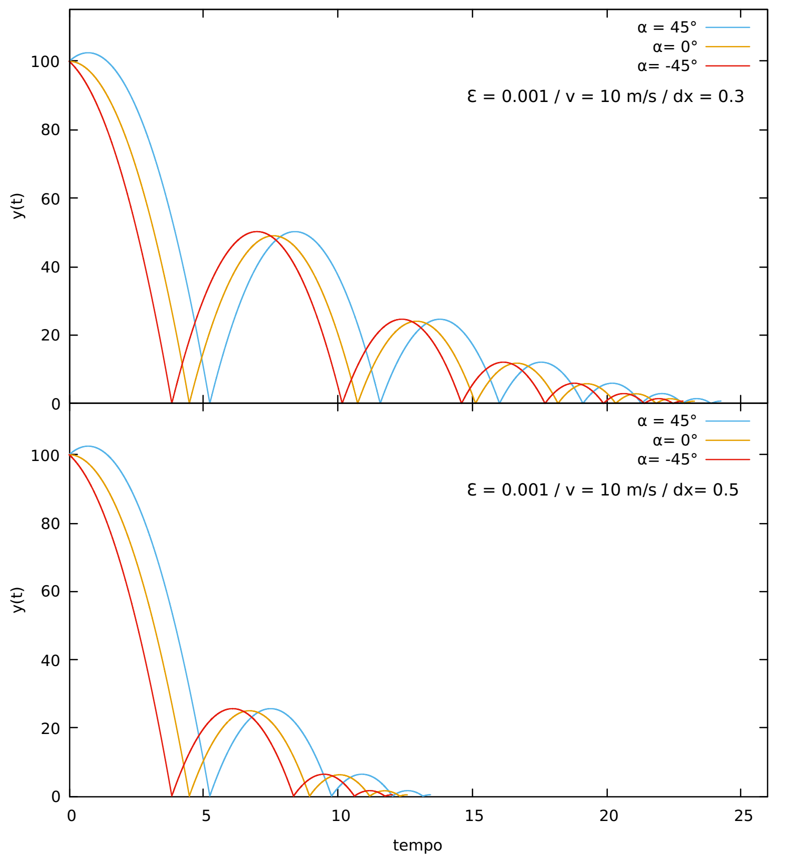
Também vemos que quanto maior a perda de energia, menor é o tempo de movimento, uma vez que mais velocidade é perdida a cada choque com o solo.

Quanto ao gráfico x(t) *vs* t, apesar do movimento ser linear horizontalmente, há o 'arredondamento' da curva devido a perda de velocidade, onde na realidade temos a junção de várias curvas lineares, diminuindo a inclinação de cada reta individual após cada choque.

Observando apenas as posições (gráfico y(x) *vs* x), vemos que o objeto se choca com o solo o mesmo número de vezes, apenas mudando o quão longe o corpo consegue ir devido as velocidades iniciais.

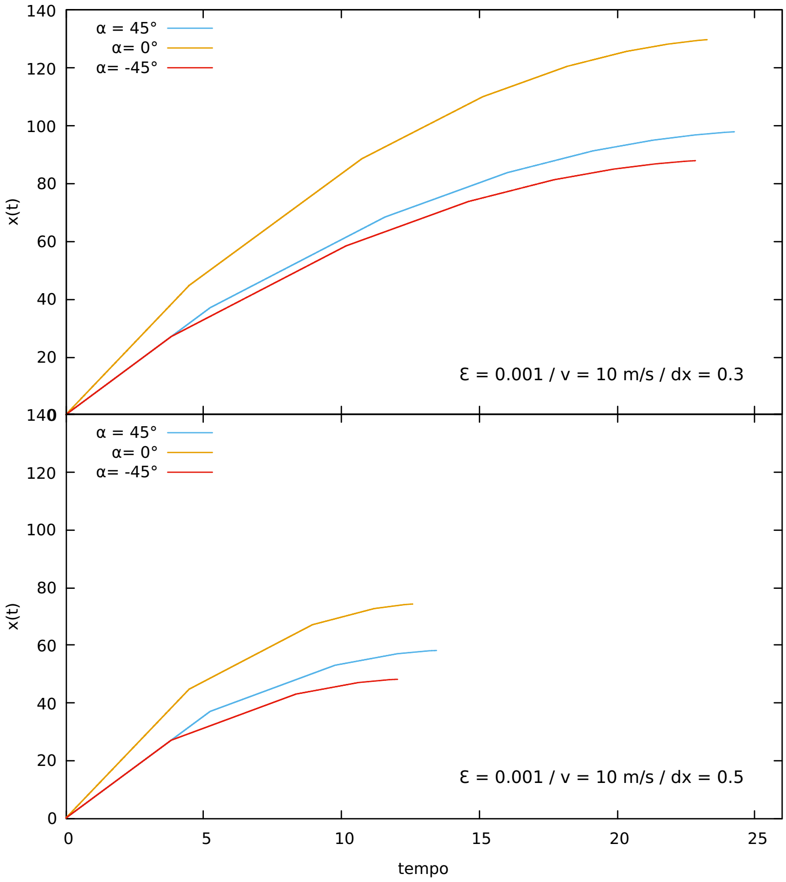


(a)

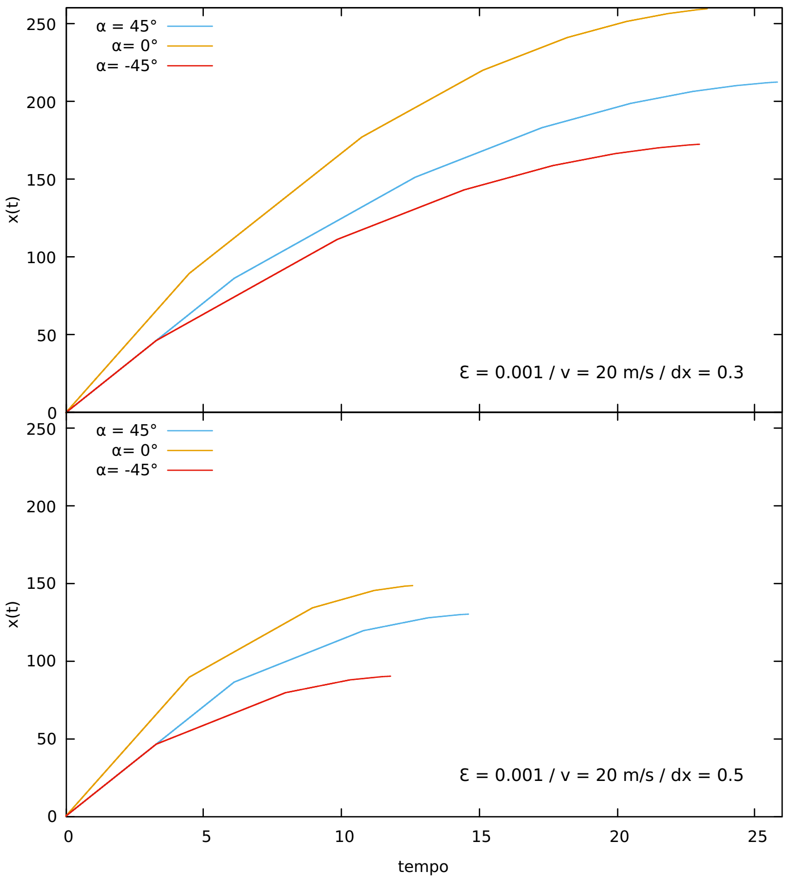


(b)

Figura 11: y(t) *vs* t

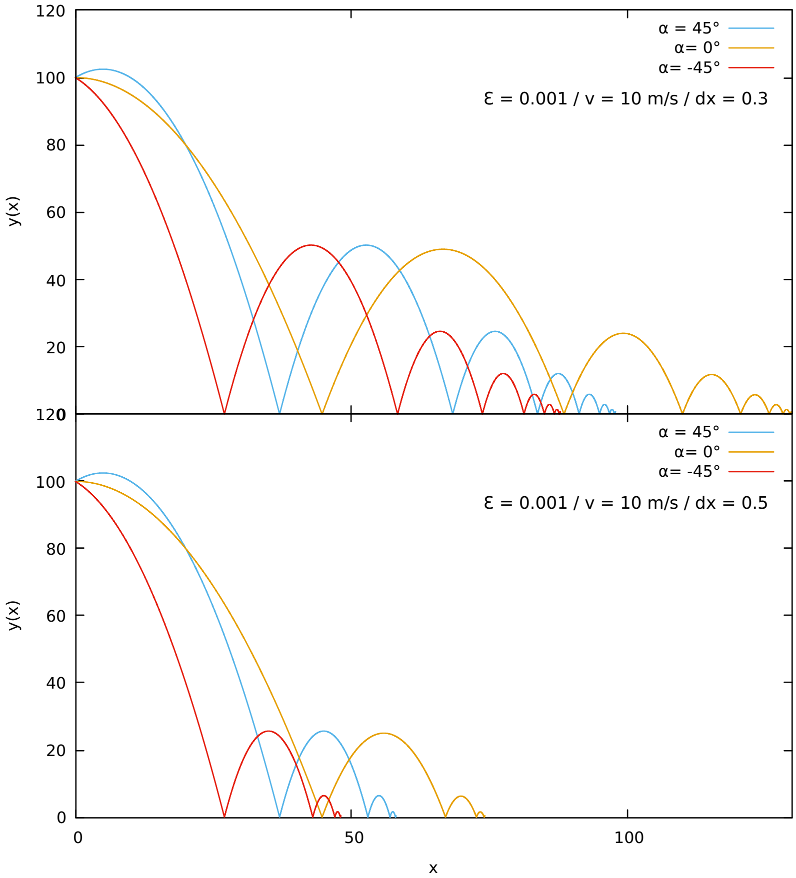


(a)

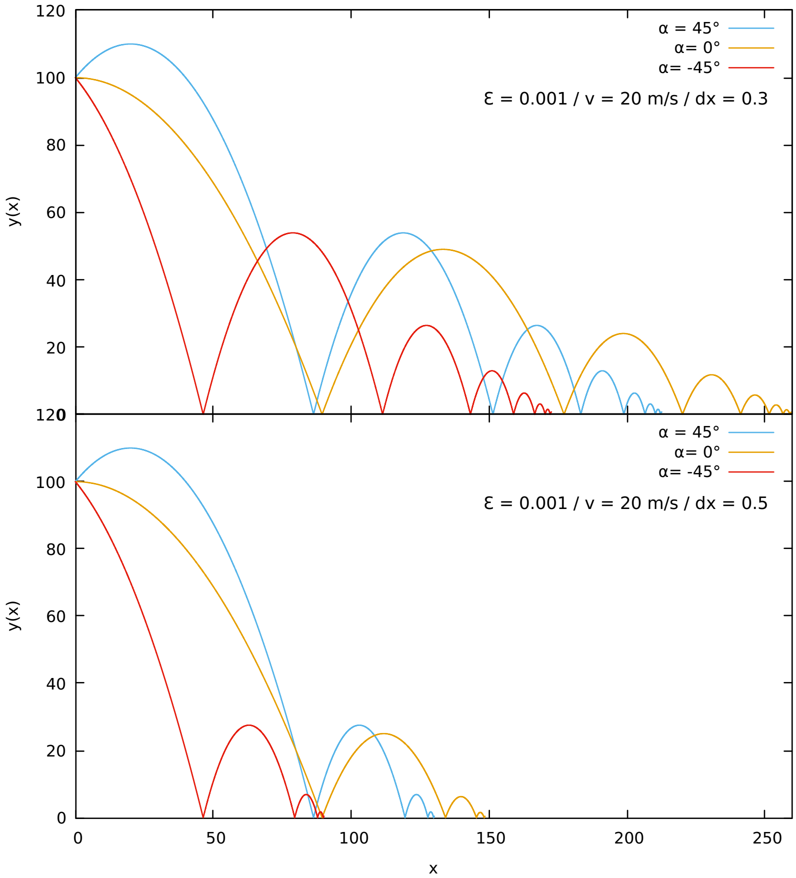


(b)

Figura 12: x(t) *vs* t



(a)



(b)

Figura 13: y(x) *vs* x

6 Subprojeto F

Código análogo ao anterior, porém com inclusão da resistência do ar. Foram testados dois valores de dx, $dx = 0$ e $dx = 0.5$.

tarefaF-10260434.f90

```fortran
Program Subprojeto_F

IMPLICIT REAL*8 (a-h,o-z) !dupla precisão
dimension vel(1:2), alfa(1:3), dxy(1:2)
character(len=70) fn

pi = 4.0*datan(1.d0)
vel = (/10.0,20.0/)          !velocidade inicial
alfa = (/pi/4.d0,0.d0,-pi/4.d0/) !angulo inicial
dxy = (/0.0,0.3/)

at = 0.001                !avanço temporal
y0 = 100.0                !y inicial
x0 = 0                        !x inicial
acely = -10.0                 !aceleração        inicial
acelx =  0.0
ifilenum = 10         !numeração de arquivos
dif  = 0.1
res = 0.1
hmax = 5

do i=1, 2
        do k=1, 2
                do j=1, 3
                        tv = 0               !tempo inicial
                        tp = 0
                        ypos = y0        !posição inicial
                        xpos = x0
                        velx = vel(i)*cos(alfa(j))
                        vely_new = vel(i)*sin(alfa(j))
                        ifilenum = ifilenum + 1
                        print*, ifilenum, 'v=', vel(i), 'alfa=',alfa(j),
                         ↪  'dxy=', dxy(k)
                        write(fn,fmt='(i0,a)') ifilenum,'.dat'
                        open(unit=ifilenum,file=fn)
                        do while (ypos .ge. 0)
                                write(ifilenum,*) tp, ypos, xpos, tv,
                                 ↪  vely_old, velx
                                tp = tp + at
                                 ↪           !tempo
                                 ↪  posição
                                tv = tv +
                                 ↪  at/2.0                     !tempo
                                 ↪  velocidade
```

```
                                acx = acelx -
                                ↪  res*velx                !aceeração
                                ↪  variável devido a res do ar
                                acy = acely - res*vely_new
                                if (tv .eq. (at/2.0)) then
                                        vely_old = vely_new + (at/2.0)*acy
                                        velx = velx+ (at/2.0)*acx
                                else        !variável p/ armazenamento da
                                ↪  velocidade anterior
                                        vely_old = vely_new + at*acy
                                        velx= velx + at*acx
                                end if
                                ypos = ypos + at*vely_old
                                xpos = xpos + at*velx         !eixo x
                                ↪  velocidade constante

                                if (ypos .lt. 0) then
                                        ypos = 0
                                end if

                                if (((vely_new*vely_old) .lt. 0) .and.
↪  (ypos .le. hmax)) then
                                        exit !a altura é máxima quando
↪  velocidade muda de sinal
                                                 !programa para dada uma
                                                 ↪  altura máxima
                                                 ↪  determinada
                                end if

                                !mudança do sentido da velocidade em y e
                                ↪  perda de velocidade em x
                                !devido a perda de energia
                                if (ypos .lt. dif) then
                                        vely_old = -vely_old +
↪  dxy(k)*vely_old
                                        velx = velx - velx*dxy(k)
                                        ypos =  ypos + at*vely_old
                                        xpos = xpos + at*velx
                                end if

                                vely_new = vely_old
                        end do
                        close(ifilenum)
                end do
        end do
end do
End Program Subprojeto_F
```

6.1 Resultados

Aqui temos que apesar do corpo não perder energia nos casos em que $dx = 0$, ainda assim o corpo perde altura com o avanço do tempo. Neste caso, mesmo que ele não esteja perdendo energia, a resistência do ar causa a perda de velocidade, pois é contrária ao movimento.

Também encontramos uma curvatura na velocidade horizontal devido a aceleração variável (o que não ocorre no caso anterior).

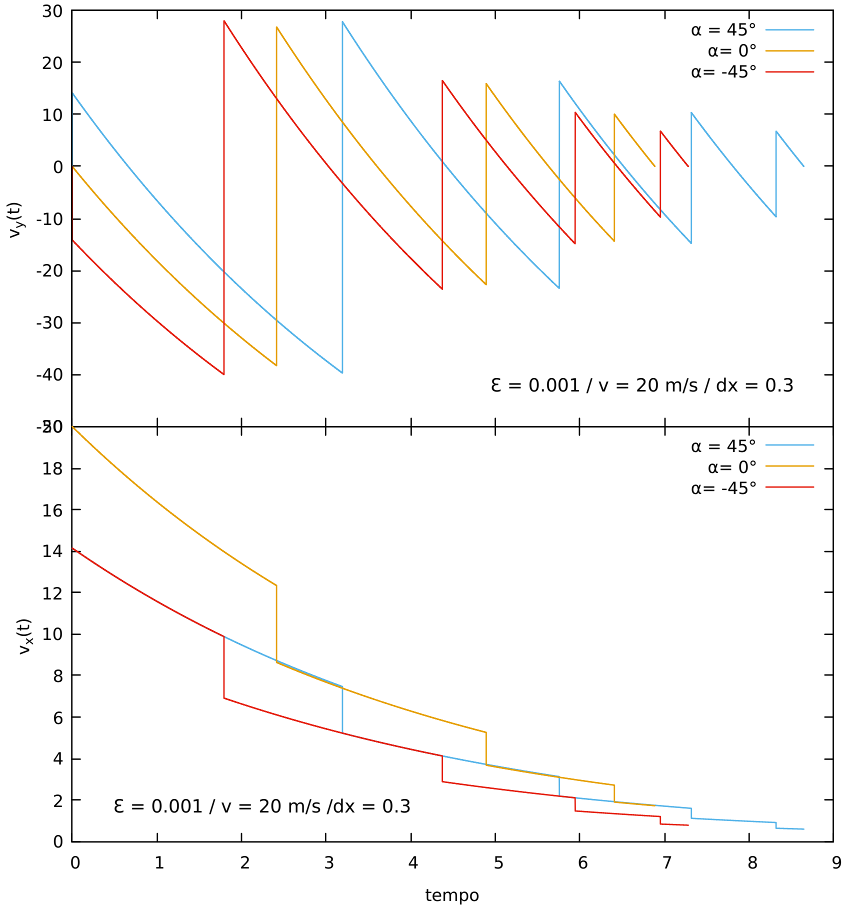
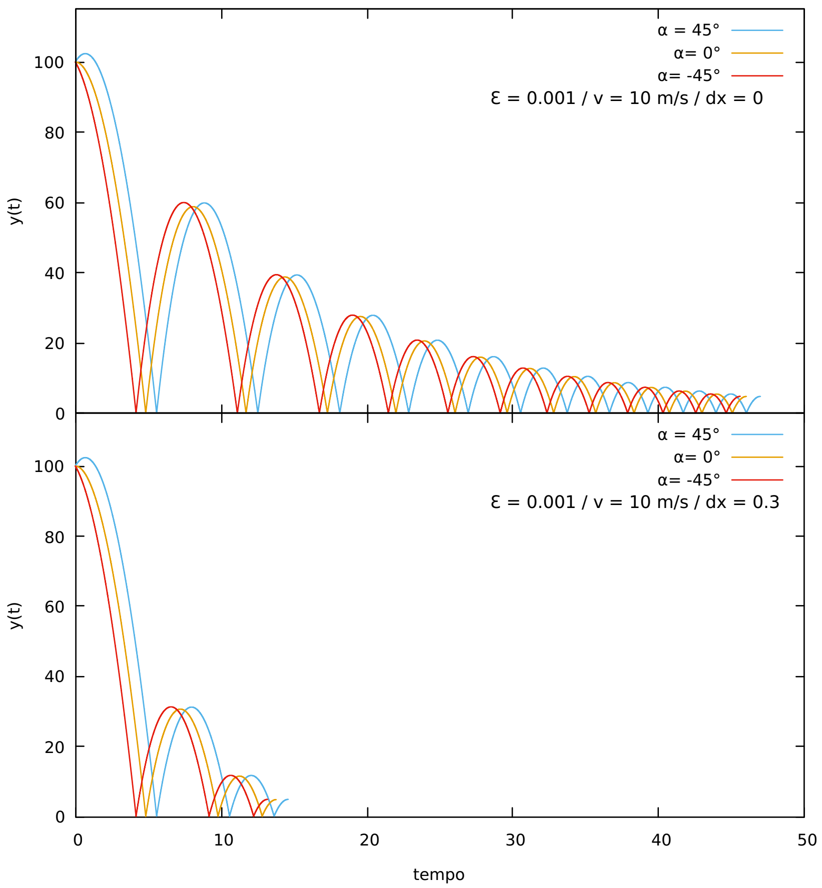
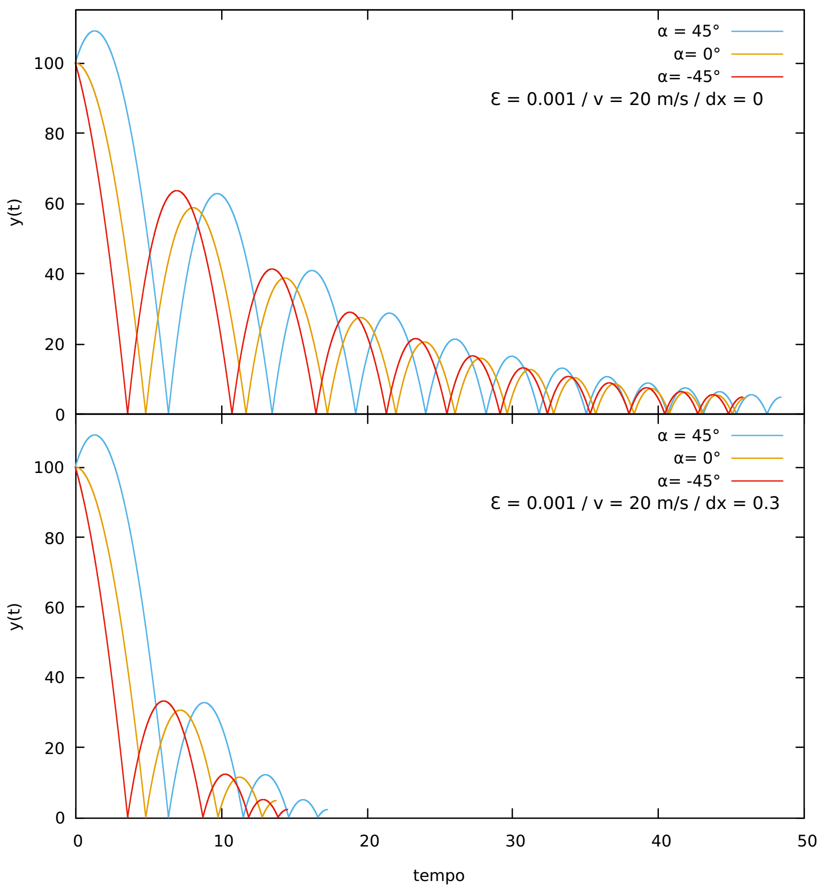


Figura 14: Comportamento das componentes da velocidade.

Devido a ação da resistência do ar, a velocidade horizontal decai mais rapidamento, fazendo com o gráfico de x(t) *vs* t apresente um plato para dx = 0, pois há estagnação na posição desse eixo uma vez que o corpo não tem mais velocidade para avançar.

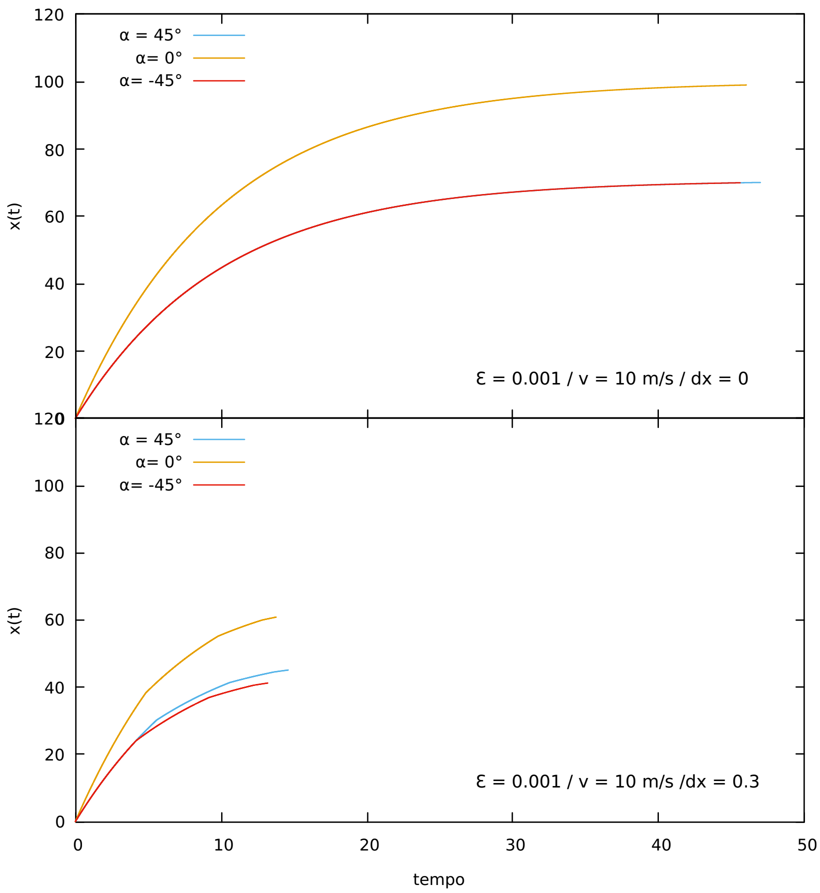


(a) D

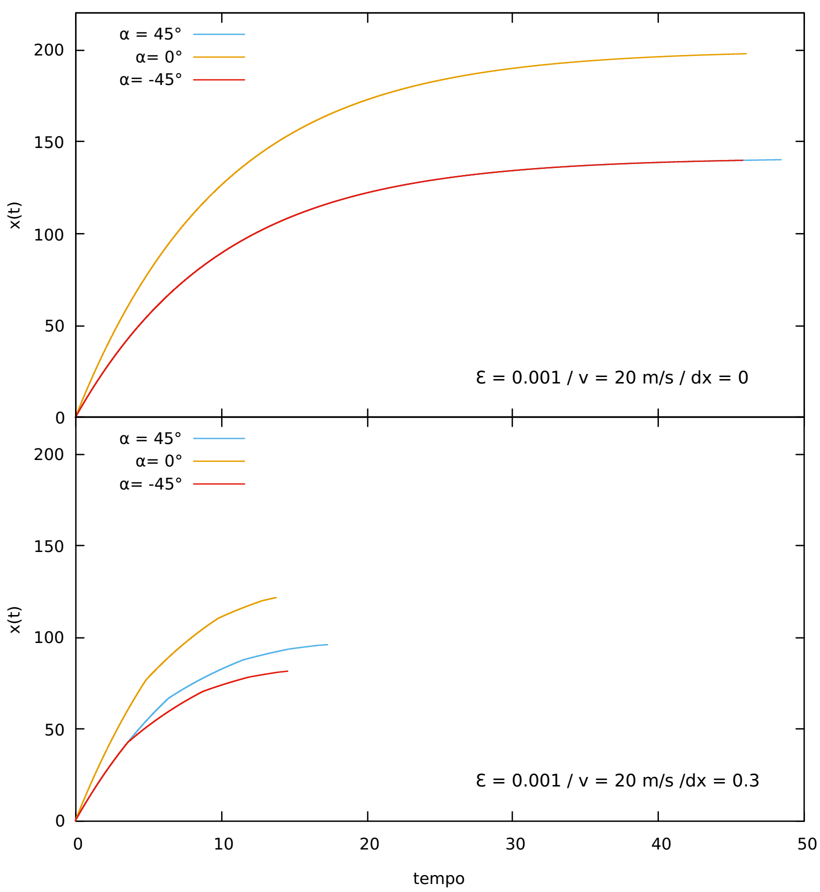


(b)

Figura 15: y vs t

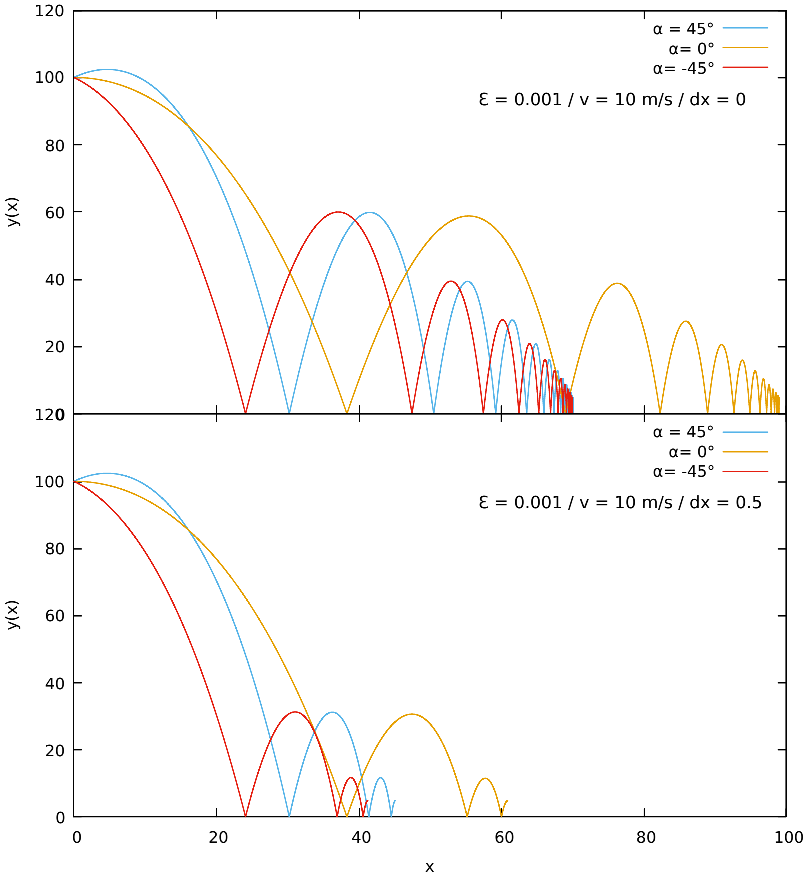


(a)

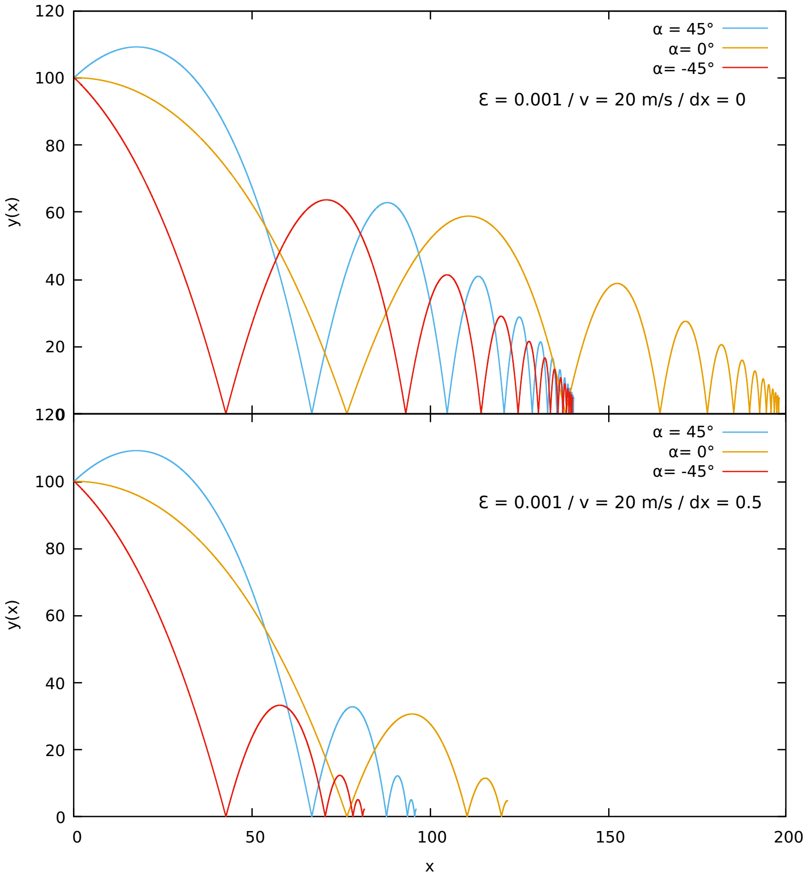


(b)

Figura 16: x vs t



(a)



(b)

Figura 17: y(x) *vs* x