Reporte de Evaluación

Carlos Muñoz De La Toba

November 2018

1 Introducción

Para ésta evaluación fue requerido modificar un programa en Fortran (probablemente en una versión vieja) de un tiro parabólico. Lo primero que se hizo fue cambiar los comentarios a "!" pues aparecian como "*c*" ademas de traer una subrutina de Euler para que funcionara el programa, para esto tambien fue necesario cambiar un "*pause*" por un "*contains*".
Luego se pedía que leyera una serie de datos proporcionados los cuales fueron escritos tal como se pedían en la lista del programa en un archivo *.txt* y que los imprimiera para después poder graficarlos en GNUPLOT como mostraremos a continuación.
El rango en los diferentes ángulos se comportó de la siguiente manera $r15 < r75 < r30 < r60 < r45$ siendo así el rango de 45 grados el mayor y el de 15 grados el menor.

2 Código

```
    program EV

  implicit none
      Real*8 d1x, d2x, d1y, d2y, ti, tf
      Real*8 xi(2), xf(2), yi(2), yf(2)
      character output*20,input*20
      real*8 g, v0, angle, dt, C, rho, Rp, Mp, yrho, u
      real*8 rad, Cd0, energy, energy0, xc, yc, vxc, vyc
real*8 xfly(5000), yfly(5000), xrange
      integer*4 i, j, key, jmax
      integer iflag, iwork(5), ne
      real*8 y(4), relerr, abserr, work(27)
      parameter (rad=3.1415926/180.0, jmax=5000)
      parameter (relerr=1.0e-9, abserr=0.0)
      common/const/ Cd0, g, yrho

      write(*,*) "Escribe el nombre del archivo de entrada de datos"
```

```
      read 201,  input

      write(*,*) "Escribe el nombre del archivo de salida de datos"

      read 201, output

      open (unit=7,file=input)

      read (7,202) key
      read (7,203) g
      read (7,203) xi(1)
      read (7,203) yi(1)
      read (7,203) v0
      read (7,203) angle
      read (7,203) dt
      read (7,203) C
      read (7,203) rho
      read (7,203) Rp
      read (7,203) Mp
      read (7,204) yrho
      read (7,203) u




!c*** end reading and set initial time to 0.0
      ti = 0.0

!c*** end initial data
      xi(2) = v0*cos(angle*rad)
      yi(2) = v0*sin(angle*rad)

!c Cd0 is the air resistance coefficient /Mp projectile
      Cd0 = C*rho*3.141592*Rp**2/Mp

!c energy0 is the initial energy of the projectile
!c later energy is calculated that is printed as a fraction of energy0
!c if there is no frictional forces the energy must be conserved
      energy0= Mp*g*yi(1) + 0.5*Mp*(xi(2)**2+yi(2)**2)
      !Aquí se almacenarán los datos
      open(unit=10,file=output,status='unknown')

      !Imprimimos los valores iniciales
      write(10,211)  xi(1), yi(1)
```

```
!c*** loop over time till the projectile hits the ground
      j=0
!c rkf45 initial data and conditions for rkf45 and first call
!c       it is very important to call rkf45 for the first time with
!c       iflag = 1 (otherwise the code does not run)
      if(key.eq.2) then
   ne = 4
   iflag = 1
   y(1) = xi(1)
   y(2) = yi(1)
   y(3) = xi(2)
   y(4) = yi(2)
      end if

!c*** loop till the projectile hits the ground i.e. yf=y1

      do while (yf(1).gt.-0.01)
        j = j+1
        tf = ti + dt

        if(key.eq.0) call euler22m(ti,tf,xi,xf,yi,yf)
        !if(key.eq.1) call  rk4_d22(d1x,d2x,d1y,d2y,ti,tf,xi,xf,yi,yf)
        if(key.eq.2) then
!          call rkf45(cannon,ne,y,ti,tf,relerr,abserr,iflag,work,iwork)
!            xf(1)=y(1)
!     yf(1)=y(2)
!     xf(2)=y(3)
!     yf(2)=y(4)
  if(iflag.eq.7) iflag = 2
end if



        energy = Mp*g*yf(1) + 0.5*Mp*(xf(2)**2+yf(2)**2)
        energy = energy/energy0

        xfly(j) = xf(1)/u
        yfly(j) = yf(1)/u


       !Imprimimos los valores siguientes
         write(10, 211)  xf(1)/u, yf(1)/u
```

```
!c* TEST section
!c  good test for the code: no air resistance
!c  then one may compare with analytic solution
        xc = 0.0 + v0*cos(angle*rad)*tf
        yc = 0.0 + v0*sin(angle*rad)*tf-0.5*g*(tf)**2
        vxc= v0*cos(angle*rad)
        vyc= v0*sin(angle*rad)-g*(tf)
!c remove comment from the next line to print
      !write(8, 211) tf,xf(1)/xc,yf(1)/yc,xf(2)/vxc,yf(2)/vyc,energy

!       c preparation for the next step
         ti = tf
         do i=1,2
            xi(i) = xf(i)
            yi(i) = yf(i)

         end do
!c***  max number of time steps is 2000
if(j.ge.jmax) exit

end do

!c*** calculate max range (using linear interpolation on the last two points)
      xrange = xfly(j-1)
      xrange = xrange+(xfly(j)-xfly(j-1))*yfly(j-1)/(yfly(j-1)-yfly(j))
      !Imprimimos el rango
     ! write (10, 213) xrange

201   format (a12)
202   format (i5)
203   format (f10.4)
204   format (e10.2)
      !Lo modificamos a solo 2 coordenadas x,y
210   format(7x,'X',11x,'Y')
211   format (f8.2, 4f12.3,1pe12.3)
212   format (' Iflag from Rkf45 = ',i2,' -> increase time step')
213   format (/,' Range is =',f12.3,'km')
    contains

      subroutine cannon(t, y, yp)
!c-----------------------------------------------
!c first and second derivatives for rkf45
!c definition of the differential equations
!c y(1) = x    yp(1)=vx=y(3)
!c y(2) = y    yp(2)=vy=y(4)
!c y(3) = vx   yp(3)=d2x/dt2 = - Cd*v*vx
```

```
!c y(4) = vy   yp(4)=d2y/dt2 = -g - Cd*v*vy
!c----------------------------------------------
      implicit none
      Real*8 t, y(4), yp(4), Cd0, g, v, yrho
      common/const/ Cd0, g, yrho
      yp(1)  =  y(3)
      yp(2)  =  y(4)
!c equation of motion
      v = sqrt(y(3)**2+y(4)**2)
  yp(3) = (-1.0)*(Cd0*exp(-y(2)/yrho))*v*y(3)
yp(4) = (-1.0)*(g + (Cd0*exp(-y(2)/yrho))*v*y(4))
      return
    end subroutine cannon

      Subroutine euler22m(ti,tf,xi,xf,yi,yf)
!c==================================================
!c euler22m.f: Solution of the second-order 2D ODE
!c method:     modified Euler (predictor-corrector)
!c written by: Alex Godunov
!c last revision: 21 October 2006
!c--------------------------------------------------
!c input ...
!c d1x(t,x,y)- function dx/dt   (supplied by a user)
!c d2x(t,x,y)- function d2x/dt2 (supplied by a user)
!c d1y(t,x,y)- function dy/dt   (supplied by a user)
!c d2y(t,x,y)- function d2y/dt2 (supplied by a user)
!c    where x(2) and y(2) (x(1)-position, x(2)-speed, etc.)
!c ti  - initial time
!c tf  - time for a solution
!c xi(2) - initial position and speed for x component
!c yi(2) - initial position and speed for y component
!c
!c output ...
!c xf(2) - solutions (x position and speed) at point tf
!c yf(2) - solutions (y position and speed) at point tf
!c==================================================
      implicit none
      Real*8 d1x, d2x, d1y, d2y, ti, tf
      Real*8 xi(2), xf(2), yi(2), yf(2)
      Real*8 h,t, x1, x2, y1, y2
      Real*8 k1x(2),k2x(2),k3x(2),k4x(2),k1y(2),k2y(2),k3y(2),k4y(2)
      h = tf-ti
      t = ti
!c*** Euler
      xf(1) = xi(1) + h*d1x(t,xi,yi)
      xf(2) = xi(2) + h*d2x(t,xi,yi)
```

```
      yf(1) = yi(1) + h*d1y(t,xi,yi)
      yf(2) = yi(2) + h*d2y(t,xi,yi)
!c*** modified Euler
      xf(1) = xi(1) + (d1x(t,xi,yi)+d1x(t,xf,yf))*0.5*h
      xf(2) = xi(2) + (d2x(t,xi,yi)+d2x(t,xf,yf))*0.5*h
      yf(1) = yi(1) + (d1y(t,xi,yi)+d1y(t,xf,yf))*0.5*h
      yf(2) = yi(2) + (d2y(t,xi,yi)+d2y(t,xf,yf))*0.5*h
      Return
    End Subroutine euler22m

  end program EV



       Function d1x(t,x,y)
!c----------------------------------------
!c function dx/dt
!c----------------------------------------
      implicit none
      Real*8 d1x, t, x(2), y(2)
       d1x = x(2)
      return
    end Function d1x


      Function d1y(t,x,y)
!c----------------------------------------
!c function dy/dt
!c----------------------------------------
      implicit none
      Real*8 d1y, t, x(2), y(2)
       d1y = y(2)
      return
      end

      Function d2x(t,x,y)
!c----------------------------------------
!c function d2x/dt2
!c----------------------------------------
      implicit none
      Real*8 d2x, t, x(2), y(2), Cd0, g, v, yrho
      common/const/ Cd0, g, yrho
       v = sqrt(x(2)**2+y(2)**2)
       d2x = (-1.0)*(Cd0*exp(-y(1)/yrho))*v*x(2)
      return
    end Function d2x
```

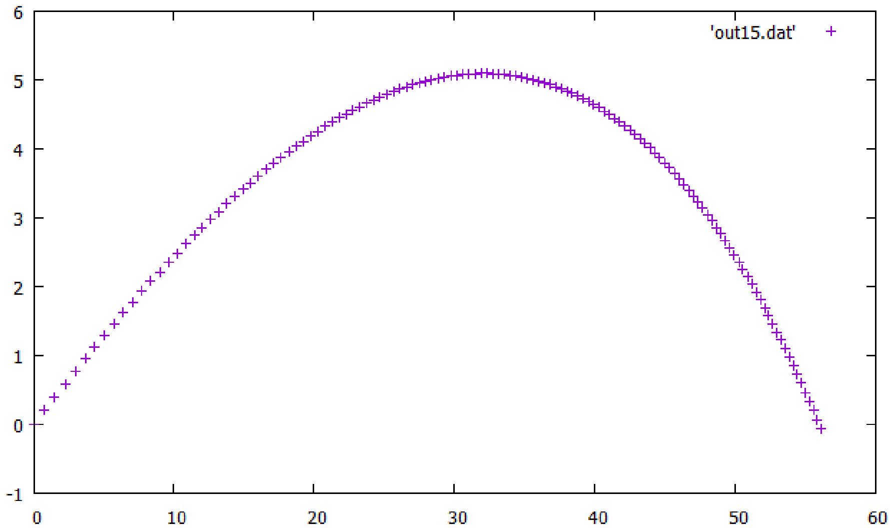

Figure 1: Sin fricción a 15

```
      Function d2y(t,x,y)
!c--------------------------------------
!c function d2y/dt2
!c--------------------------------------
      implicit none
      Real*8 d2y, t, x(2), y(2), Cd0, g, v, yrho
      common/const/ Cd0, g, yrho
       v = sqrt(x(2)**2+y(2)**2)
       d2y = (-1.0)*(g + (Cd0*exp(-y(1)/yrho))*v*y(2))
      return
    end Function d2y
```

.

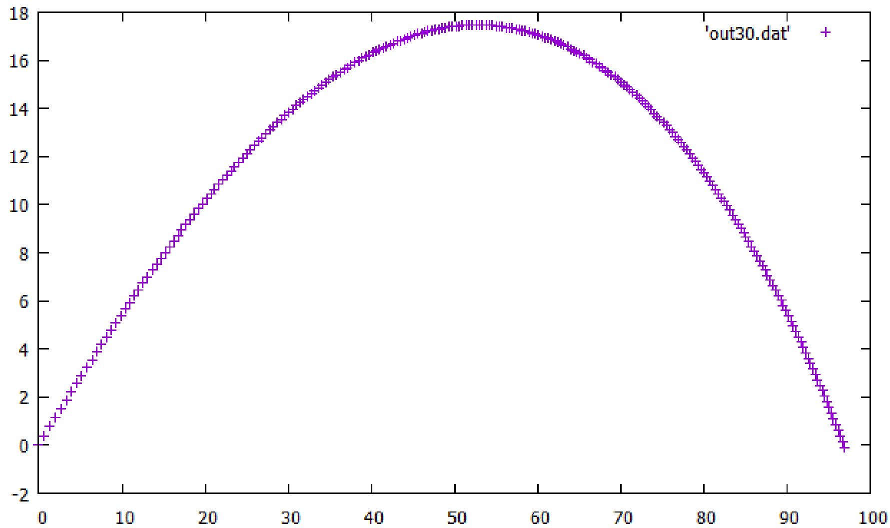

Figure 2: Sin fricción a 30

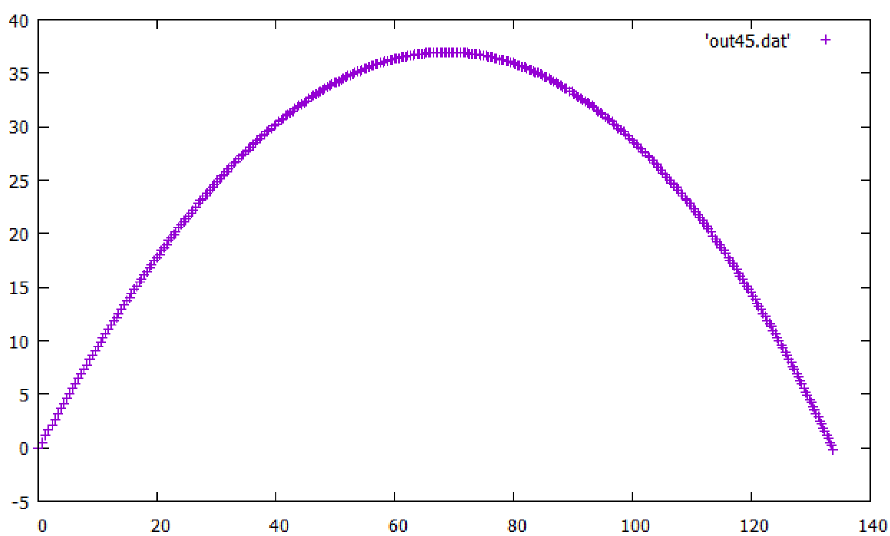

Figure 3: Sin fricción a 45

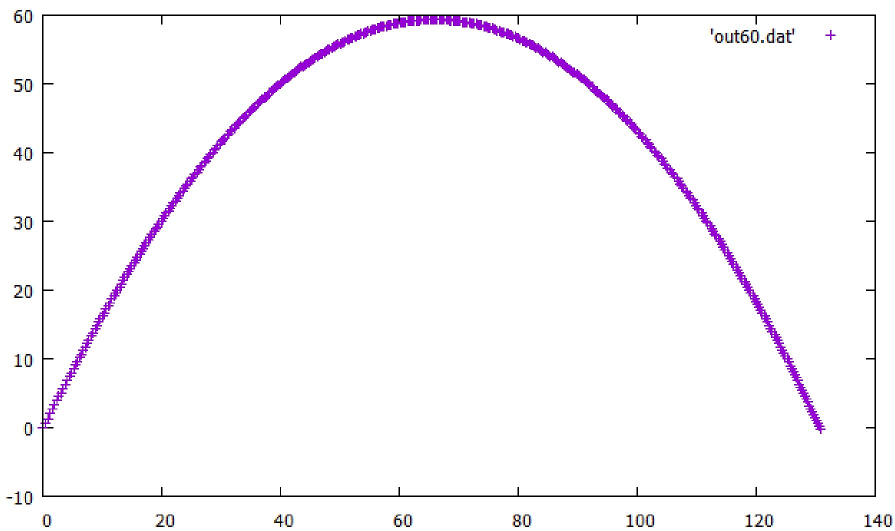

Figure 4: Sin fricción a 60

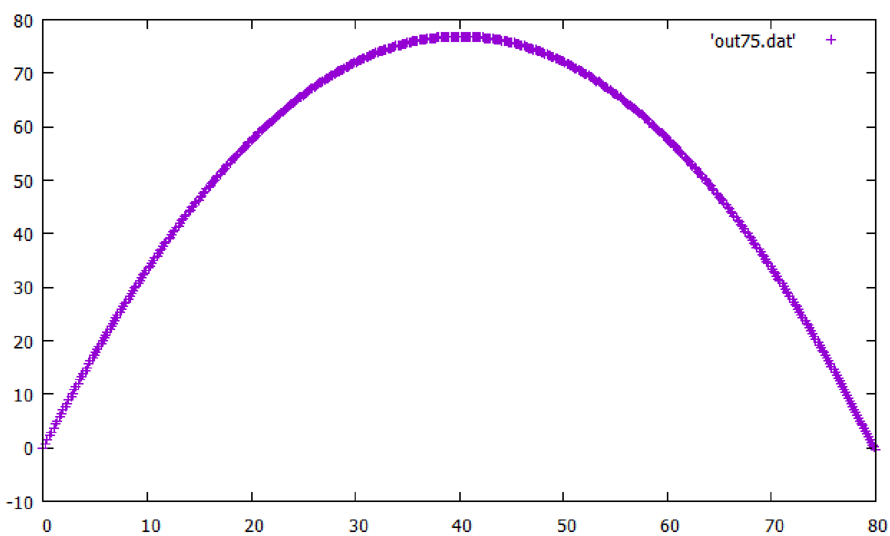

Figure 5: Sin fricción a 75

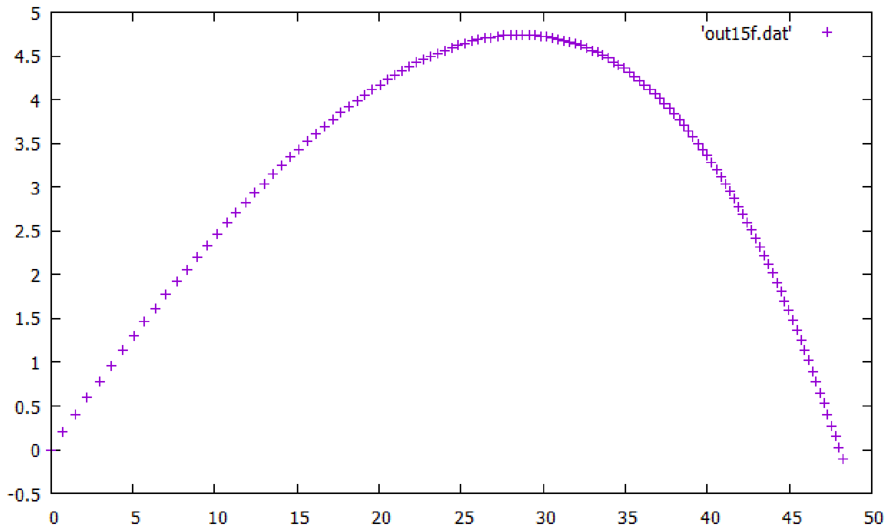

Figure 6: Con fricción a 15

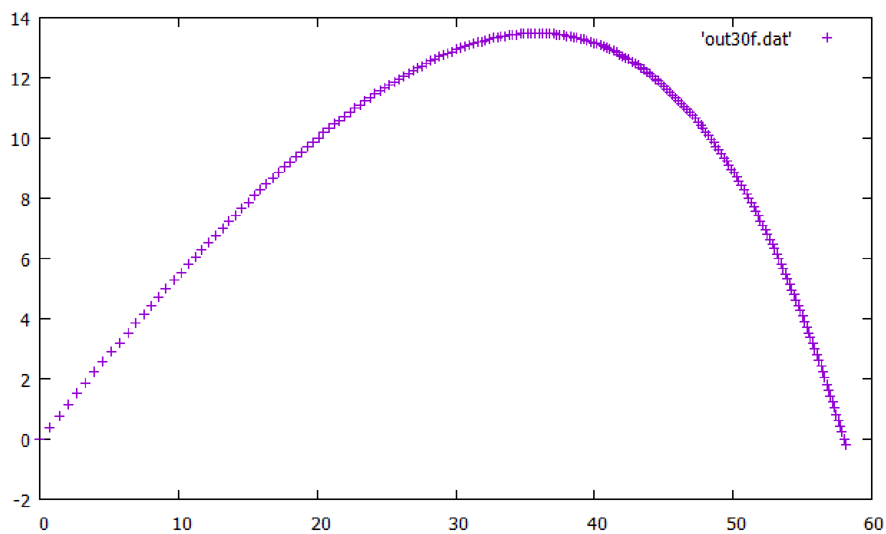

Figure 7: Con fricción a 30

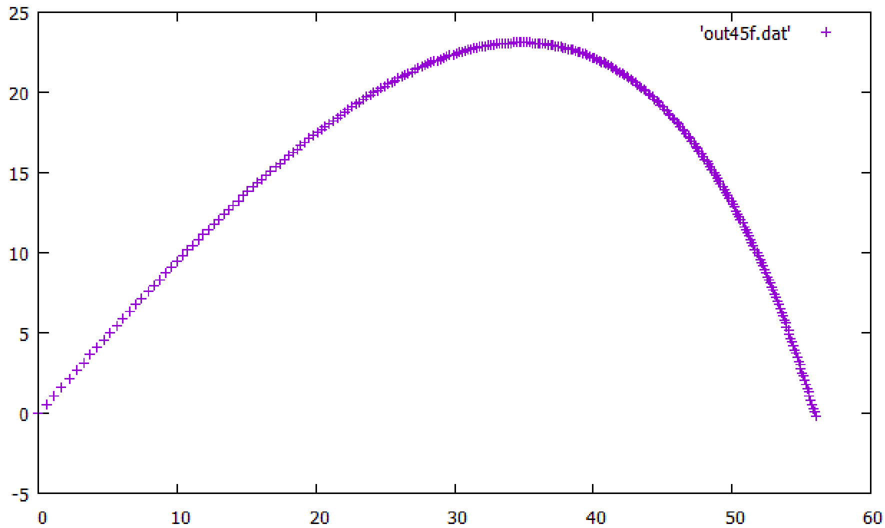

Figure 8: Con fricción a 45

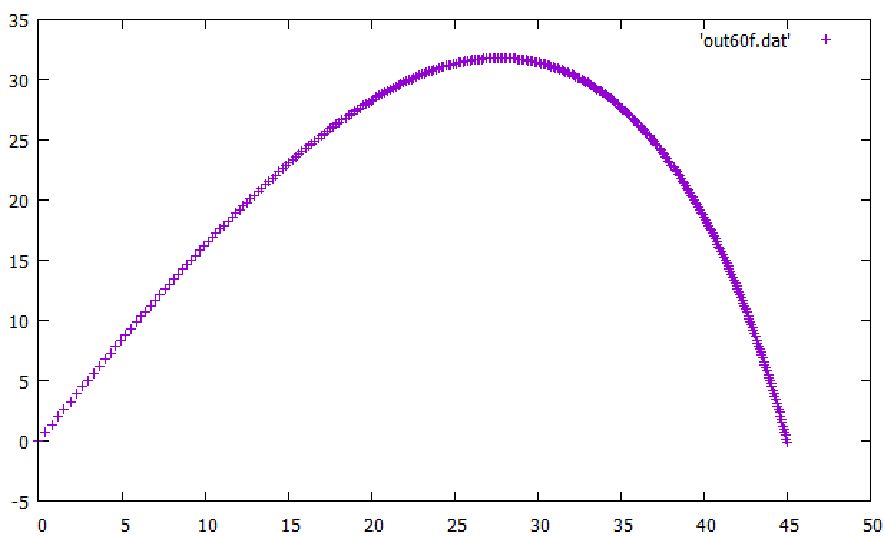

Figure 9: Con fricción a 60

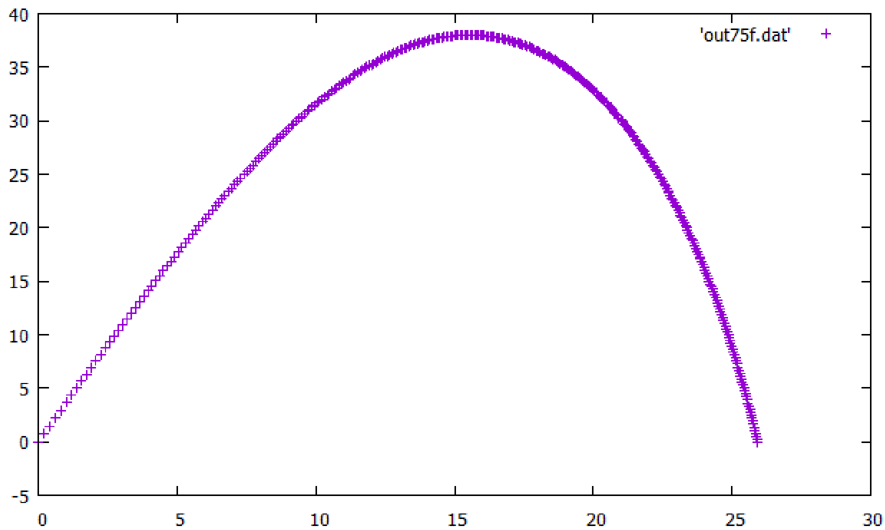

Figure 10: Con fricción a 75