

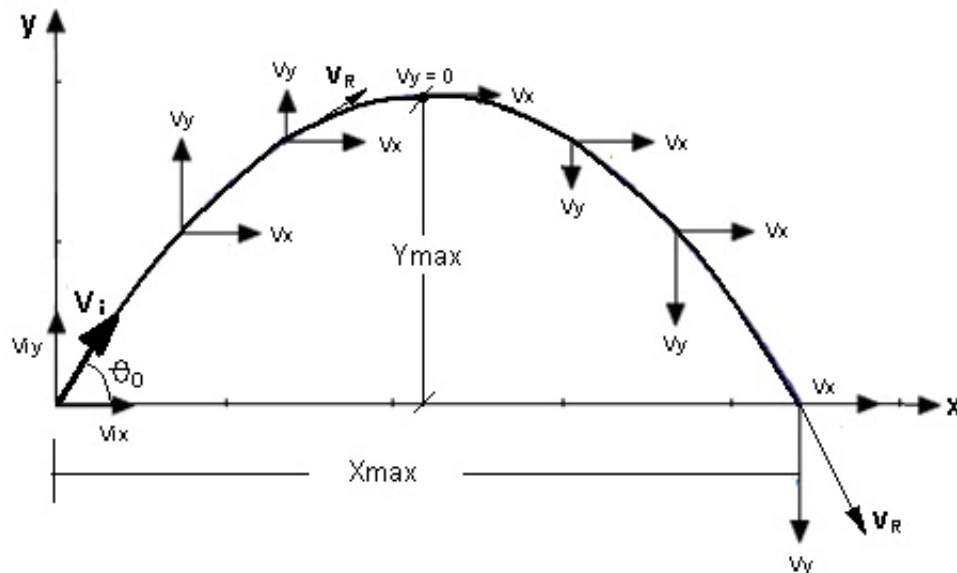
Producto 6 : Fuerza de Arrastre

Olga María Fimbres Morales

Todo objeto de masa m que se mueve a muy alta velocidad en un fluido de densidad P , experimenta una fuerza de arrastre F_D contraria a la dirección de su movimiento y es dada por la ecuación

donde u es la magnitud del vector velocidad del objeto, C_D es el coeficiente de arrastre (adimensional), A es el área transversal presentada por el objeto. por ejemplo, para una esfera el área transversal es $A = \pi r^2$, y el coeficiente de arrastre es $C_D = 0.47$

Se pide agregar el efecto de resistencia del aire al objeto lanzado en tiro parabólico, El objeto ahora experimenta una fuerza de arrastre en la dirección del movimiento o bien produciendo una aceleración variable .



Estructura del código.

1.- Sección de declaración de constantes.

Primeramente es necesario especificar las variables para un determinado escenario donde situaremos la simulación del tiro parabólico.

En un inicio se utilizaron valores de densidad, gravedad y coeficiente de fricción para elaborar primeramente un código que funcionara con estos valores, los cuales se posicionaron dentro de un modulo de constantes para que no fuera necesario especificarlos dentro de cada subrutina:

```
module Cte
implicit none
real, parameter :: g = 9.81, p =1.1644, pi = 4.0*atan(1.0), CD = 0.47
integer, parameter :: ntps=5000
end module Cte
```

2.- Sección de declaración de variables.

En esta sección se especifican todas aquellas variables con las que va a trabajar el programa y con las que identificará cada elemento necesario para hacerlo. Primeramente, es necesario separarlas por cuales serán reales, cuales serán integrales y cuales utilizaras para realizar las operaciones dentro del DO. Y se escriben al inicio del programa y al inicio de las subrutinas.

Algunas de las que se declaran como reales son definidas por operaciones realizadas con otros valores que se obtienen dentro del programa o bien dentro de las diferentes subrutinas y después son llamadas dentro de programa.

3.- Subrutinas.

Podemos realizar diferentes subrutinas dentro del programa que se encarguen de calcular diversas variables, las cuales podemos llamar dentro del programa para recoger sus valores y presentarlos como resultados.

En esta ocasión, se desarrollaron dos subrutinas, una que trabajara para calcular los resultados del tiro sin fricción y otra para el tiro con fricción.

Primeramente, la subrutina sin fricción, pide las siguientes variables:

```
subroutine Tiro_sfriccion(v0, dt_r, v0x, v0y, Xs, Ts, Ys, tttotal, xsf, ysf)
```

```
use Cte
```

las cuales corresponden a la velocidad inicial, los grados en radianes, el ángulo convertido a radianes, la posición final, el tiempo total de vuelo, la altura máxima alcanzada, y las variables que utilizaremos para realizar el LOOP, respectivamente.

En seguida corresponde ordenar las variables dependiendo de si serán solamente de entrada-intent(in)- o de entrada y salida -intent(inout)-. Tal como se muestra a continuación:

```
implicit none
real, intent(in) :: v0x, v0y, v0, dt_r
real, intent(inout) :: Xs, Ts, Ys, tttotal, xsf, ysf
real, dimension (0:ntps) :: xx, yy, tt
integer :: i
```

Después, iniciamos el LOOP iniciando por abrir el archivo del cual graficaremos, y especificando desde donde iniciará y donde terminará los cálculos:

```
open (1, file='tirosinfriccion.dat')
do i=0, ntps, 1
```

Y las fórmulas que ya conocemos para el tiro parabólico sin fricción deben ser modificadas solamente en sus variables para que trabajen con los nuevos intervalos de tiempo que se han especificado, y señalando las condiciones bajo las que deseamos que trabaje, en este caso, se desea que trabaje solamente para valores positivos de Y.

```
tt(i) = (float(i)*0.01)
xx(i) = v0x*tt(i)
yy(i) = v0y*tt(i) - 0.5*g*tt(i)*tt(i)
write (1,*) xx(i), yy(i)
if (yy(i)<0) exit
end do
close (1)
```

Finalmente, señalamos aquellos valores que llamaremos dentro del programa que serán nuestros resultados; para el tiempo total de vuelo y el alcance máximo, basta con señalar el último valor obtenido del LOOP, pero para obtener la altura máxima es necesario señalar que es ese el que deseamos con el comando Maxval, seguido de los arreglos a los que nos estamos refiriendo, el inicio de este y las condiciones bajo las que trabaja.

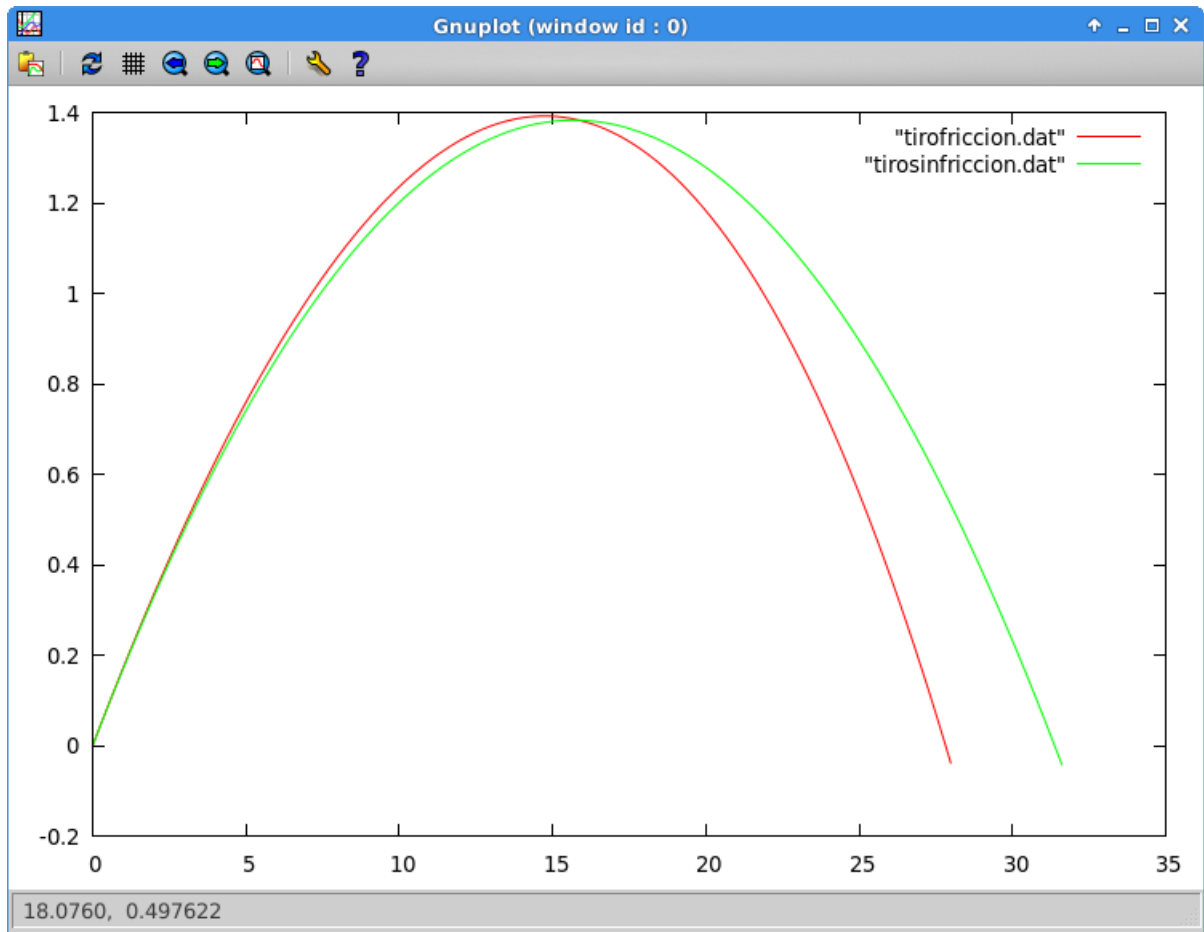
```
tttotal =tt(i)
xsf = xx(i)
ysf = maxval(yy, 1, (yy(i)<0))
```

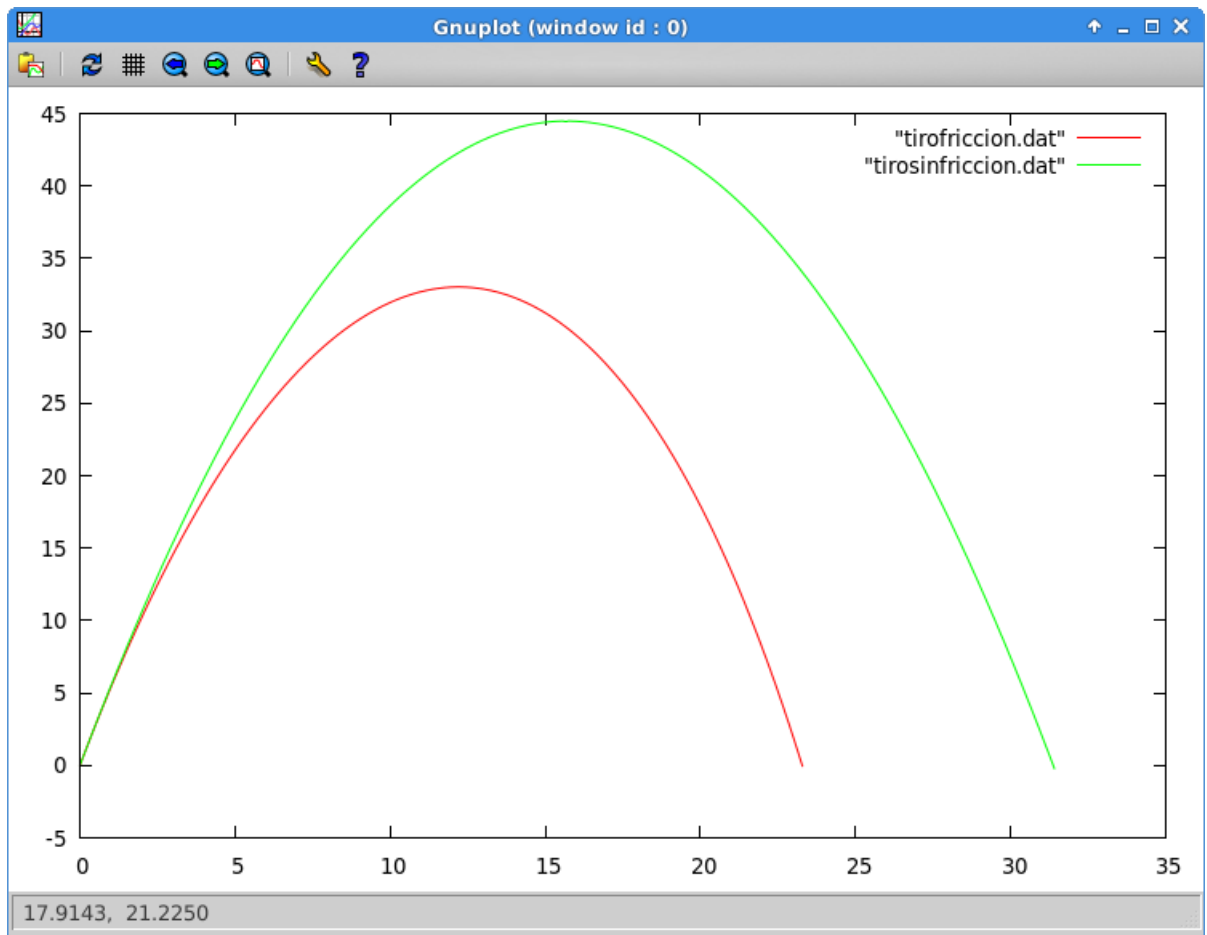
Análisis y evidencias.

Al comparar un tiro con una misma masa, velocidad inicial y ángulo, es posible darse cuenta de las claras diferencias entre un tiro parabólico en un modelo ideal y en un modelo real.

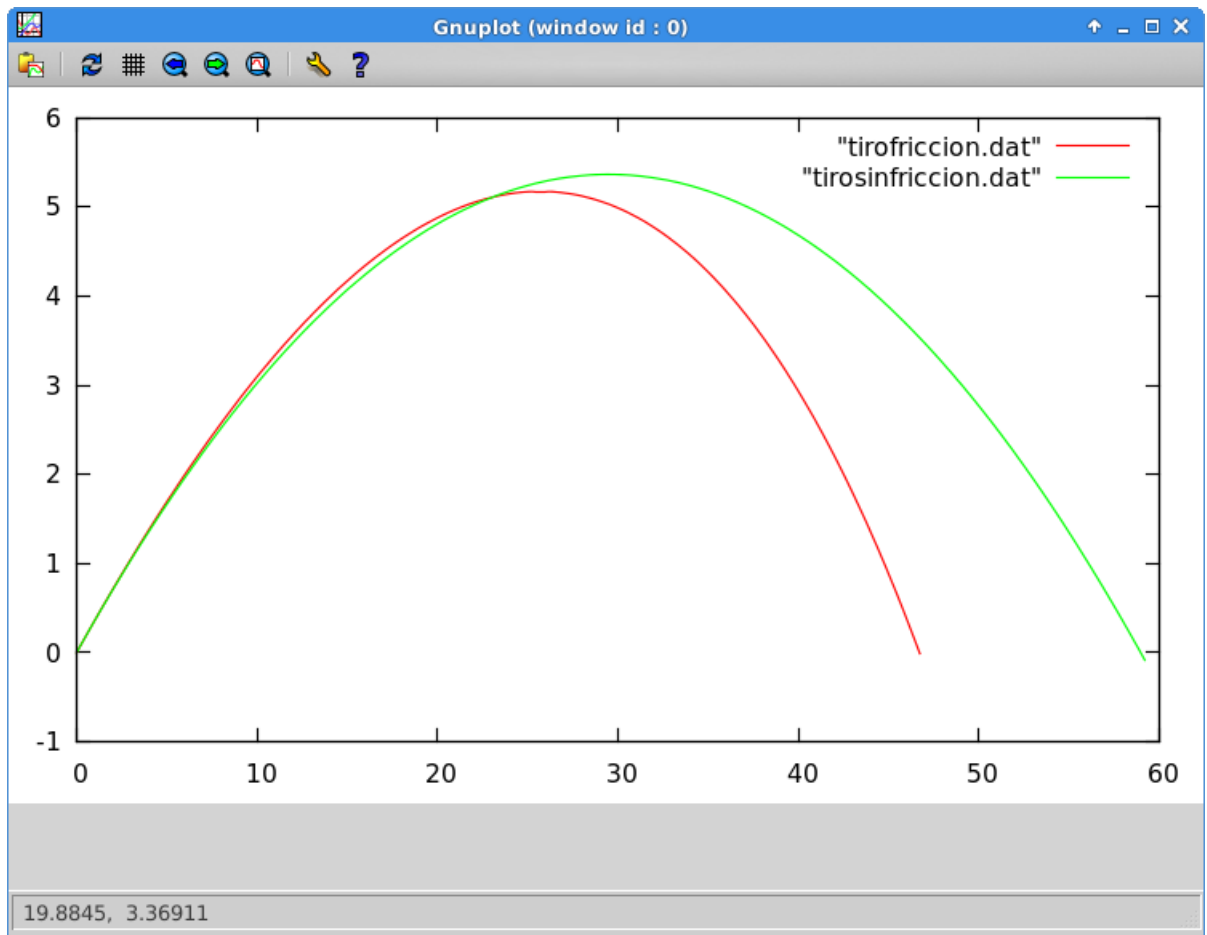
Como ejemplo de ello comparamos los tiros con y sin fricción de diversas parejas de ángulos complementarios.

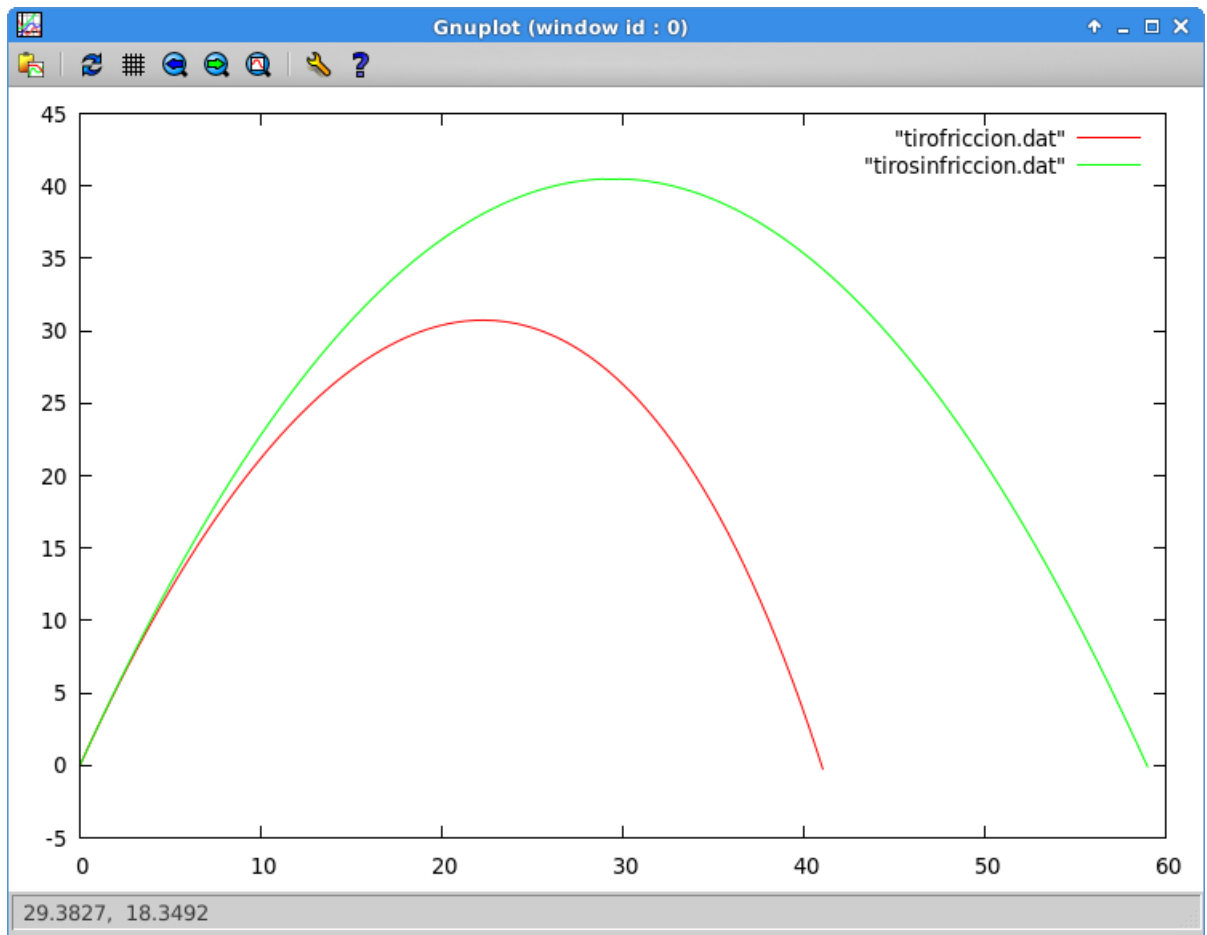
10 y 80 grados.



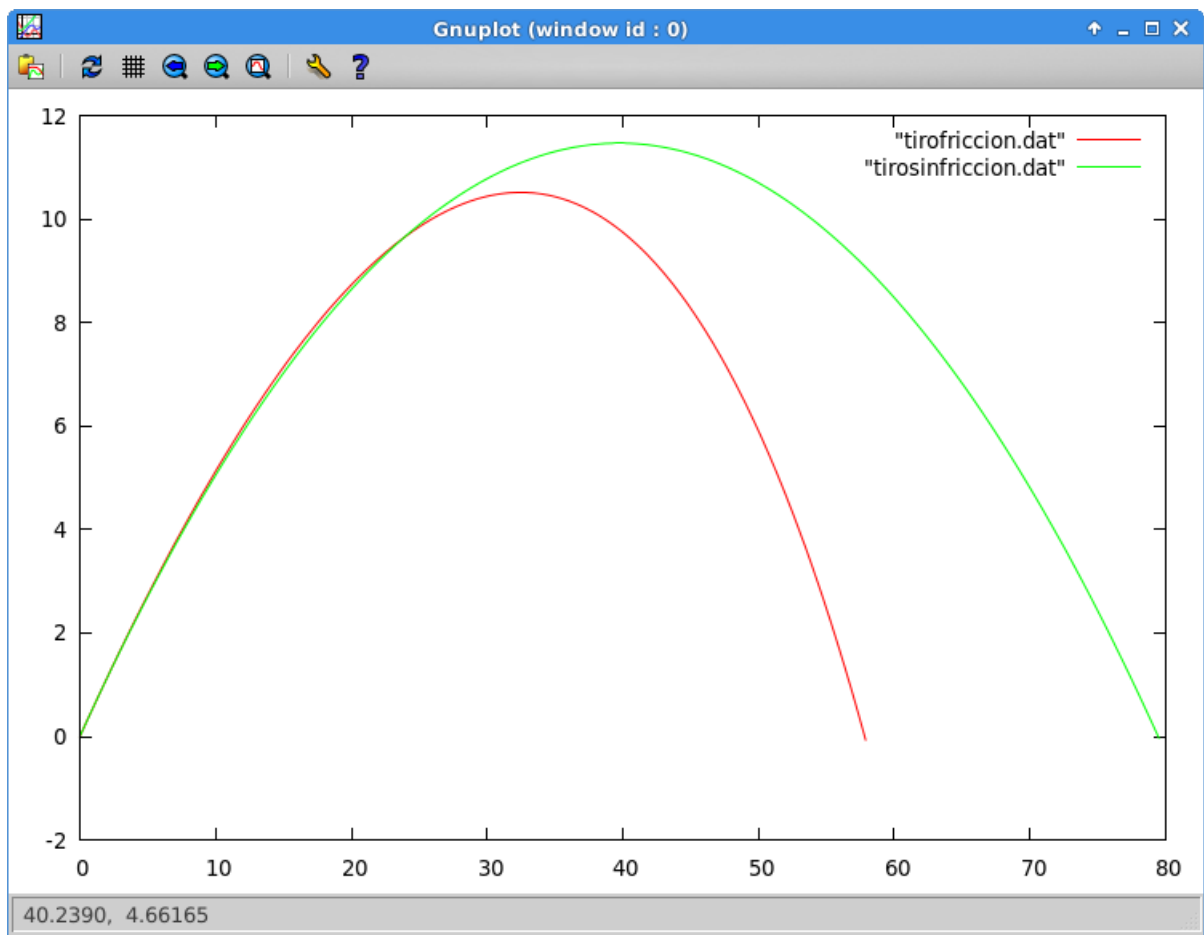


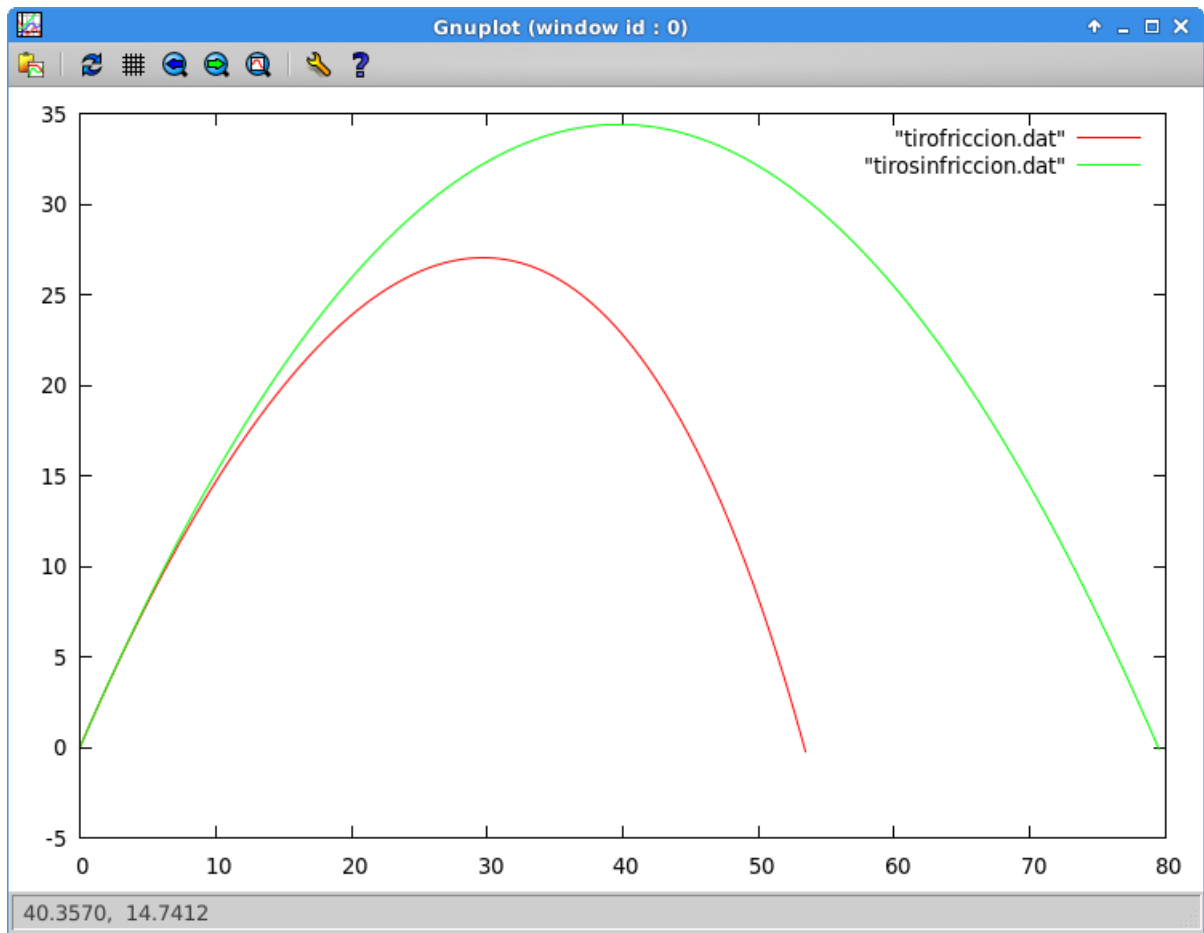
20 y 70 grados.





30 y 60 grados.





Finalmente, analizando estos resultados desde el punto de vista de la física, podemos darnos cuenta que el modelo real dista mucho del modelo ideal, ya que en el primero actúan más fuerzas que las que se consideran en el segundo; las cuales afectan los resultados del tiro.

Por ejemplo, al considerar la fricción del aire, podemos darnos cuenta que hay un punto en que al comparar las gráficas el tiro con fricción se encuentra sobre el tiro sin fricción, esto se debe a que en un mismo tiempo se encuentran en diferentes posiciones; en este caso, el tiro real se encuentra por detrás del tiro ideal debido a la fuerza que ejerce el aire en el objeto, empujándolo hacia atrás.

También podemos darnos cuenta que el tiempo que le toma subir al proyectil es mayor que el tiempo que le toma bajar, esto también debido a las fuerzas que actúan sobre él en cada etapa del tiro.

Código.

```
module Cte
implicit none
real, parameter :: g = 9.81, p = 1.1644, pi = 4.0*atan(1.0), CD = 0.47
! g = atiende al valor general que se le asigna a la gravedad terrestre, esta
puede variar dependiendo de la altura o del planeta donde deseemos posicionar
la simulación.
!p = es la densidad del aire a una temperatura de 30 grados centigrados,
la cual varia junto con la temperatura.
!CD = atiende al coeficiente de fricción de un cuerpo esférico, dependiendo
del la forma del proyectil esta varia.
integer, parameter :: ntps=5000 ! este número solo atiende a la cantidad de
puntos que se le desee poner como máximo para realizar las operaciones.
end module Cte

program Tiro_parabolico
use Cte
implicit none
real :: dt, x0, y0, v0, v0x, v0y, m, dt_r, D, Xs, Ts, Ys, Xf, Tf, Yf, A, r,
tttotal, xsf, ysf, tt
real :: vxf(0:ntps), vyf(0:ntps), ax(0:ntps), ay(0:ntps), xx(0:ntps),
yy(0:ntps), ft(0:ntps), Vo(0:ntps)

print * , 'Datos iniciales'
write (*,*) 'Ingrese la masa del proyectil'
read *, m
print * , '-----'
write (*,*) 'Identifique el radio del proyectil'
read *, r
print * , '-----'
write (*,*) 'Ingrese una velocidad inicial para el proyectil'
read *, v0
print * , '-----'
write (*,*) 'Ingrese un ángulo inicial para el proyectil'
read *, dt
print * , '-----'
```

```

write (*,*) 'Determine una posición inicial en el eje x'
read *, x0
print * , '-----'
write (*,*) 'Determine una posición inicial en el eje y'
read *, y0
print * , '-----'

dt_r=(dt*pi)/180 !fortran trabaja las funciones trigonométricas con radianes,
asi que debemos convertir los grados a radianes.
v0x = v0*cos(dt_r) !velocidad inicial en x.
v0y = v0*sin(dt_r) !velocidad inicial en y.
A = pi*r*r !area transversal .
D = p*CD*A*0.5 !constante de fricción.

print * , '-----'
print * , '-----'
print * , '-----'
print * , '-----'
print * , '-----'

call Tiro_sfriccion(v0, dt_r, v0x, v0y, Xs, Ts, Ys, tttotal, xsf, ysf)
!llamamos a la subrutina pertinente para que
nos muestre los valores que estamos pidiendo.
print * , 'Modelo Ideal'
print * , 'Tiempo de vuelo', tttotal, 'segundos'
print * , 'Alcance', xsf, 'metros'
print * , 'Altura máxima', ysf, 'metros'
print * , '-----'
print * , '-----'
print * , '-----'
call Tiro_friccion1(v0, v0x, v0y, ax, ay, Xf, Tf, Yf, x0, y0, vxf, vyf, ft, D,
m, Vo) !llamamos a la subrutina.
print * , 'Modelo real'
print * , 'Tiempo de vuelo',Tf , 'segundos'
print * , 'Alcance',Xf , 'metros'
print * , 'Altura máxima',Yf , 'metros'
end Program Tiro_parabolico

```

```

subroutine Tiro_sfriccion(v0, dt_r, v0x, v0y, Xs, Ts, Ys, tttotal, xsf, ysf)
use Cte
implicit none
real, intent(in) :: v0x, v0y, v0, dt_r
real, intent(inout) :: Xs, Ts, Ys, tttotal, xsf, ysf
real, dimension (0:ntps) :: xx, yy, tt
integer :: i

Ts = 2*v0*sin(dt_r)*(1/g)
Ys = v0*v0*sin(dt_r)*sin(dt_r)*(1/(2*g))
Xs = v0*v0*sin(2*dt_r)*(1/g)

open (1, file='tirosinfriccion.dat')
do i=0, ntps, 1

    tt(i) = (float(i)*0.01)
    xx(i) = v0x*tt(i)
    yy(i) = v0y*tt(i) - 0.5*g*tt(i)*tt(i)
write (1,*) xx(i), yy(i)
if (yy(i)<0) exit
end do
close (1)

tttotal =tt(i)
xsf = xx(i)
ysf = maxval(yy, 1, (yy(i)<0))

end subroutine Tiro_sfriccion

subroutine Tiro_friccion1(v0,v0x, v0y, ax, ay, Xf, Tf, Yf, x0, y0,
vxf, vyf, ft, D, m, Vo)
use cte
implicit none
real, intent(in) :: v0, v0x, v0y, x0, y0, D, m
real, intent(inout) :: Xf, Tf, Yf
real, dimension (0:ntps) :: fx, ft, fy, vxf, vyf, ax, ay, Vo
integer :: i

```

```

fy = 0
ft(0)=0
fx(0)=x0
fy(0)=y0
vxf(0)=v0x
vyf(0)=v0y
ax(0) = -(D/m)*(v0x)*(v0x)
ay(0) = -g - ((D/m)*(v0y)*(v0y))

open (2, file='tirofriccion.dat')
do i = 0, ntps, 1

ft(i+1) = (ft(i)*0.01) +0.01 !Intervalo de tiempo
vxf(i+1) = vxf(i) + (ax(i)*ft(i+1))!Velocidad X
vyf(i+1) = vyf(i) + (ay(i)*ft(i+1)) !Velocidad Y.
fx(i+1) = fx(i) + (vxf(i)*ft(i+1)) + ((1/2)*ax(i)*ft(i+1)*ft(i+1)) !Posición X.
fy(i+1) = fy(i) + (vyf(i)*ft(i+1)) + ((1/2)*ay(i)*ft(i+1)*ft(i+1)) !Posición Y.
ax(i+1) = -(D/m)*(vxf(i))*(vxf(i)) !Aceleracion X.
ay(i+1) = -g - ((D/m)*(vyf(i))*(vyf(i))) !Aceleración Y

write (2, 1001) fx(i), fy(i)
if (fy(i)<0) exit !Condición.
end do
1001 format (2f10.6)
close (2)
<<<<<< HEAD
velx= vxf(i)
vely= vyf(i)
acely= ay(i)
acelx= ax(i)
Tf = ft(i)
Xf = fx(i)

!Últimos valores encontrados y el máximo valor de Y.
Tf = ft(i) * 10.0
Xf = fx(i+1)
Yf = maxval(fy, 1, (fy(i)<0))
end subroutine Tiro_friccion1

```