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 FD contraria a la direccion de su movimiento y es dad por la ecuacion

donde u es la magnitud del vector velocidad el objeto, CD es el coeficiente de arrastre (adimensional), A es el area transversal presentada por el objeto. por ejemplo, para una esfera el area transversal es , y el coeficiende de arrastre es CD = 0.47

Se pide agregar el efecto de resistencia del aire al objeto lanzada en tiro parabolico, El aobjeto ahora experimenta una fuerza de arrastre en la direccion del movimiento o bien produciendo una aceleracion 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 elavorar primeramenta 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 seran reales, cuales seran integrales y cuales utilizaras para realizar las operaciones dentro del DO. Y se escriben al inicio del programa y al inicio de las subrutinas.

```
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,
ttotal, 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)
```

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

```
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
```

3.- Subrutinas.

```
subroutine Tiro_sfriccion(v0, dt_r, v0x, v0y, Xs, Ts, Ys, ttotal, xsf, ysf)
use Cte
implicit none
real, intent(in) :: v0x, v0y, v0, dt_r
real, intent(inout) :: Xs, Ts, Ys, ttotal, 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)

ttotal =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
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
```