Project 1: A simple projectile motion (no air resistance)

by Alex

February 6, 2010

Introduction.

A motion of a projectile without air resistance is a common theme for introductory physics courses. The only force on the projectile is the gravitational force.

Theory.

The projectile motion without air resistance is well described by the following four equations (2D case)

$$x = x_i + (v_0 \cos\theta_0)t$$
$$v_x = v_0 \cos\theta_0$$
$$y = y_i + (v_0 \sin\theta_0)t - \frac{gt^2}{2}$$
$$v_y = v_0 \sin\theta_0 - gt$$

where we use very common notations for any introductory physics book.

For the project we need to find the initial shooting angle for the given initial and finial positions, and the magnitude of the initial velocity. Eliminating time from the first and third equations gives a non-linear equation for the initial angle, namely

$$y_i - y_f + v_0 \sin\theta_0 \frac{x_f - x_i}{v_0 \cos\theta_0} - \frac{g}{2}\left(\frac{x_f - x_i}{v_0 \cos\theta_0}\right)^2 = 0$$

The effect of the moving ship during shooting can easily be incorporated into the equation by modifying the horizontal velocity as

$$v_x = v_0 \cos\theta_0 + v_{ship}$$

When the initial and final vertical positions are the same, the non-linear equation has a simple analytic solution

$$\theta = \frac{1}{2}\arcsin\left(\frac{g(x_f - x_i)}{v_0^2}\right)$$

This solution can be used for the verification of the program.

Method.

The brute force method based on the bisectional scheme has been used for solving the non-linear equation. There are following reason to use this approach: 1) from the physics of the problem we expect up two possible solutions between 0 and 90 degrees for the shooting angle, 2) using the bisectional scheme we do not count the same root twice, as it may happen using Newton's scheme.

Sample report

Verification of a program.

For the initial conditions: x_i=0.0 m, y_i=0.0 m, x_f=1952.0 m, y_f=0.0 m, v_0=200.0 m/s the program gives two following solutions: 14.301 and 75.699 degrees. The analytic equation gives the same 14.301 and 75.699 degrees.

Results.

For 200 m/s initial speed.

```
Question 1: Solutions for the moving ship
 root  angle(deg)    t(sec)   cannonball
  1      15.796      10.089   hits the wall
  2      75.921      39.290   clears the wall
Question 2: Solutions without ship motion
 root  angle(deg)    miss(m)
  1      15.872      10.440
  2      75.595      40.366
```

Analysis.

There are two solutions (angles) for the given conditions. However, only the second solution clears the wall. If the motion of the ship is not taken into account, then for the second (accepted) angle, the overshoot will actually be much more (40 meters) than the size of the armory (8 meters).

From the presented results we can see that a small variation in the angle (change in less than 0.5 degrees) results in shoot well away from the target. However, for old Navy ship the accuracy in the shooting angle was not as good, to say nothing about rough see conditions. Therefore, a chance to hit the armory from one short would be very slim.

Let's evaluate the effect of the air resistance. For fast moving projectiles the aerodynamic drag force can be evaluated as $F_{drag}=-0.5C\rho_0 Av^2$, where ρ_0 stands for air density (ρ_0=1.25 kg/m^3 at sea level), and A is the cross section. The drag coefficient C depends on an object shape and for many objects it can be approximated by a value within 0.05 - 0.5. Assume that the horizontal speed stays the same, we may evaluate what fraction of kinetic energy will be spend to overcome air resistance. Assume that C=0.2, and the diameter of the cannonball is 0.2 m with mass of about 10 kg. Then the initial kinetic energy of the cannonball is 200KJ. The energy required to overcome air resistance maintaining the same speed would be more than 200KJ! Thus, the effect or the air resistance can not be ignored, or even considered as a small correction.

Some advice to the captain

1 I am pretty sure that in 17th century sailors didn't calculate trajectories using algebra based physics, but rather used their experience. Most cannons had attached tables for shooting (factory calibrations), where the effect of the air resistance was incorporated.
(Thus, the advice "don't use algebra based physics" would not have much value.)

2 Shooting multiple projectiles (or simultaneous fire off from a few cannons) would increase chances to hit the target.

3 Selecting a position for shooting or the initial projectile speed in a way to have less sensitivity to the initial condition would also increase chances for success.

Program:

This is Fortran.90 program for the project.

```
module physics
implicit none
double precision, parameter:: g=9.81, pi=3.1415926, rad=57.2958
double precision:: xi, xf, yi, yf, xw, yw, v0, vs
end module physics

 program main
!=====================================================================
! Project 1. Simple projectile motion (no air resistance)
! "A British navy ship ..."
!=====================================================================
use physics
implicit none
integer, parameter:: n=100
double precision f,x1,x2,eps
double precision roots(n)
double precision t, x, y, tflight
integer key, nroots, i
character(16) message
character(16), dimension(2), parameter:: wall=&
(/" hits the wall  "," clears the wall"/)
external f
!
! initial data
!
xi = 0.0
yi = 0.0
xw = 1852.0
yw = 50.0 + 60.0
xf = xw + 100.0
yf = 50.0
v0 = 200.0
vs = 2.0*(1.852*1000.0/3600.0)
! print initial data
write(*,*) '            Initial Data'
write(*,*) '   xf       yf       xw       yw       v0        vs'
write(*,100) xf, yf, xw, yw, v0, vs
100 format(6f9.2,/)

! Part 1: find angles between 0 and 90 deg to hit the target at (xf,yf)

key = 1
x1 = 0.0
x2 = pi/2.0
```

```
eps = 1.0e-7

  call BForce(f,x1,x2,eps,roots,key,n,nroots)

!
! Part 2: check condition: cannon ball should NOT hit the wall
!
write(*,*) ' Question 1: Solutions for the moving ship'
write(*,*) '  root  angle(deg)    t(sec)   cannonball'
do i=1,nroots
  t = (xw-xi)/(v0*cos(roots(i))+vs)
  y = yi+v0*sin(roots(i))*t-0.5*g*t**2
  if (y.le.yw) then
    message = wall(1)
    else
    message = wall(2)
  end if
  tflight = (xf-xi)/(v0*cos(roots(i))+vs)
  write (*,101) i, roots(i)*rad, tflight, message
end do

!
! Part 3: The effect of the moving ship
!
vs = 0.0

  call BForce(f,x1,x2,eps,roots,key,n,nroots)

write(*,*) '  '
write(*,*) ' Question 2: Solutions without ship motion'
write(*,*) '  root  angle(deg)    miss(m)'
do i=1,nroots
  tflight = (xf-xi)/(v0*cos(roots(i))+vs)
  x = 2.0*(1.852*1000.0/3600.0)*tflight
  write(*,102) i, roots(i)*rad, x
end do
101 format(i5,2f12.3,a18)
102 format(i5,2f12.3)
end program main

  Function f(x)
!=============================================================
! the equations of the project
!=============================================================
use physics
implicit none
double precision f, x, t
t = (xf - xi)/(v0*cos(x)+vs)
```

```
f = yi - yf +  v0*sin(x)*t - 0.5*g*t**2
end function f

   Subroutine BForce(f,x1,x2,eps,Roots,key,n,nroots)
!============================================================
! Multiple roots of equation f(x)=0 on [x1,x2] interval
! Method: Brute force with one of closed domain methods
! Close domain methods: bisectional or false position
! Alex G. January 2010
!------------------------------------------------------------
! input ...
! f   - function - evaluates f(x) for any x in [x1,x2]
! x1  - left endpoint of initial interval
! x2  - right endpoint of initial interval
! eps - desired uncertainity of the root as |b-a|<eps
! key - select a method
!       1 - bisectional method
!       2 - false position method
! n   - number of subintervals for [x1,x2]
! output ...
! Root(n) - roots of the equation f(x)=0 on [x1,x2]
! nroots  - number of roots (nroots<=n)
!
! Comments:
! The program divide [x1,x2] into n subintervals
! Max number of iterations for every subinterval - 200
!============================================================
implicit none
integer n, nroots
double precision f, x1, x2, eps, Roots(n)
double precision a, b, c, dx, root
integer i, j, key
integer, parameter:: iter=200

! initialization
dx = (x2-x1)/real(n)
nroots = 0

! loop over subintervals
do j=1,n
  a = x1  + real(j-1)*dx
  b = a + dx
! check the closed domain condition f(a)*f(b)<0
  if(f(a)*f(b)>0) cycle

! Iterative refining the solution
  do i=1,iter
    if(key == 1) then
```

Sample report

```
      c=(b+a)/2.0
      else
      c = b - f(b)*(b-a)/(f(b)-f(a))
    end if
    if(f(a)*f(c).le.0.0) then
      b = c
    else
      a=c
    end if
!   condition(s) to stop iterations)
    if(abs(b-a)<= eps) exit
  end do
! check if it is a root or singularity
  root = (b+a)/2.0
  if (abs(f(root)) < 1.0) then
    nroots = nroots+1
    Roots(nroots)=root
  end if
end do
end subroutine BForce
```

Sample output

```
             Initial Data
    xf       yf       xw       yw       v0        vs
  1952.00    50.00  1852.00   110.00   200.00     1.03

  Question 1: Solutions for the moving ship
   root  angle(deg)    t(sec)   cannonball
    1      15.796      10.089   hits the wall
    2      75.921      39.290   clears the wall

  Question 2: Solutions without ship motion
   root  angle(deg)    miss(m)
    1      15.872      10.440
    2      75.595      40.366
```