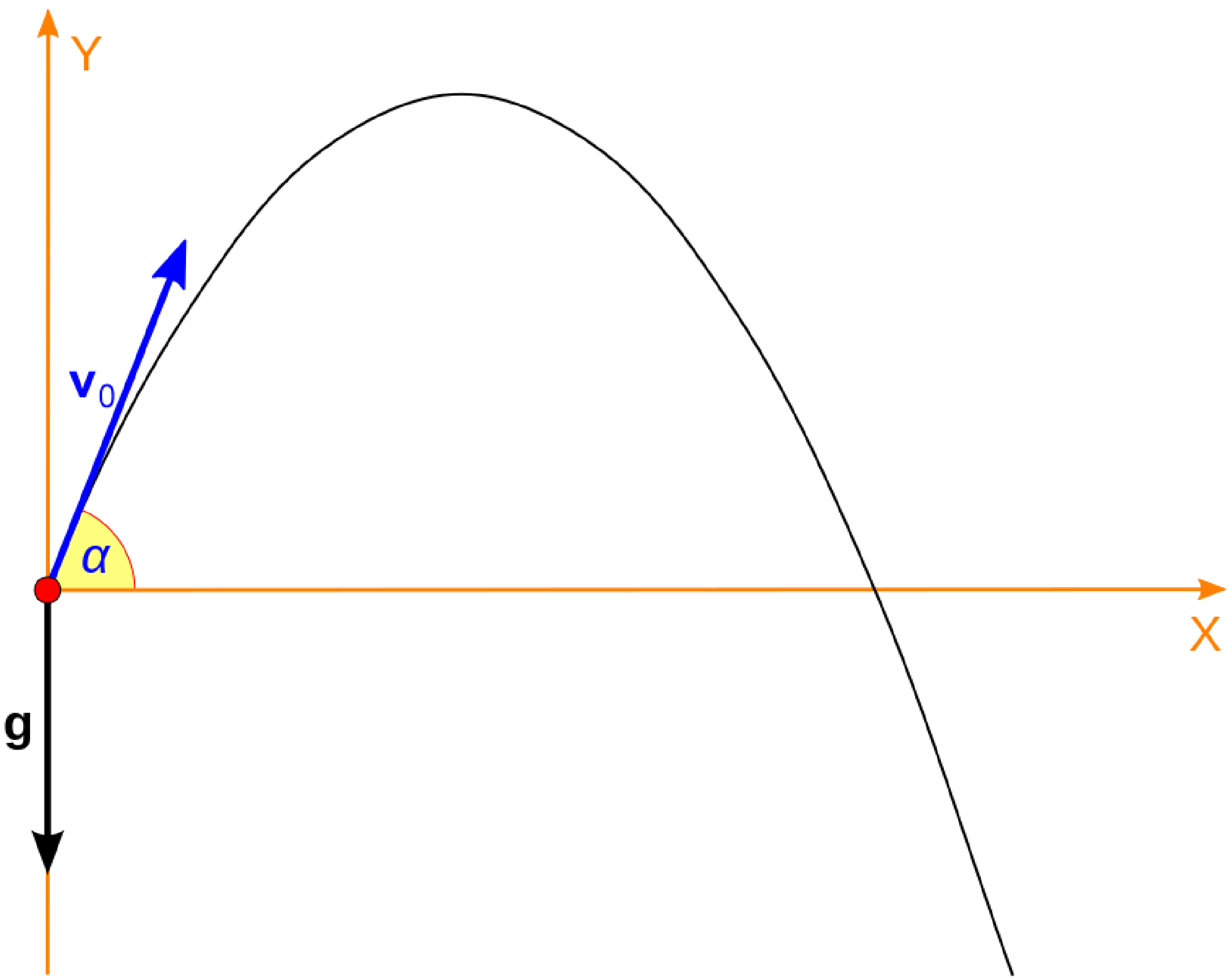
Project II

Path to source code: `/home/d/dx/dxj4360/Project2`

The projectile problem

This is a projectile problem.



Initial Conditons

- $m = 30$kg
- $v_0 = 100$m/s
- θ_0 is the angle to the horizontal, which is denoted α in the diagram.

- Air-friction force $F = -kv^2$, where $k = 5.0 \times 10^{-2}$ SI unit

Analysis

- Applied force
  - x-direction: $F_x = F\cos(\theta) = -kv^2\cos(\theta)$
  - y-direction: $F_y = -mg + F\sin(\theta) = -mg - kv^2\sin(\theta)$

- DE of motion
  - x-direction: $\frac{dv_x}{dt} = \frac{F_x}{m} = -\frac{k}{m}v^2\cos(\theta)$ & $\frac{dx}{dt} = v_x$
  - y-direction: $\frac{dv_y}{dt} = \frac{F_y}{m} = -g - \frac{k}{m}v^2\sin(\theta)$ & $\frac{dy}{dt} = v_y$

- Initial condition
  - $v_x(0) = v_0\cos(\theta_0)$ & $x(0) = 0$
  - $v_y(0) = v_0\sin(\theta_0)$ & $y(0) = 0$

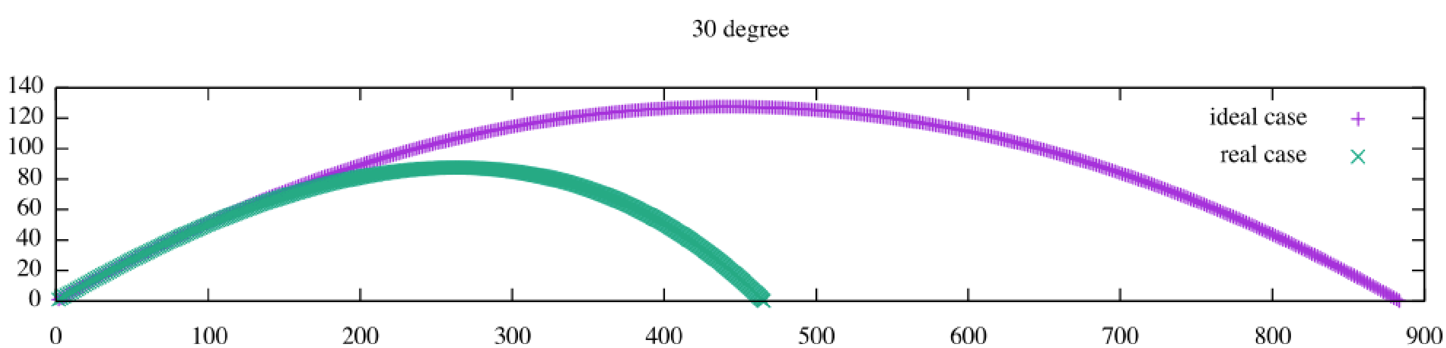
- Geometric relations
  - $v(t) = \sqrt{v_x^2(t) + v_y^2(t)}$
  - $\theta(t) = \arctan\frac{v_y(t)}{v_x(t)}$

Flow chart

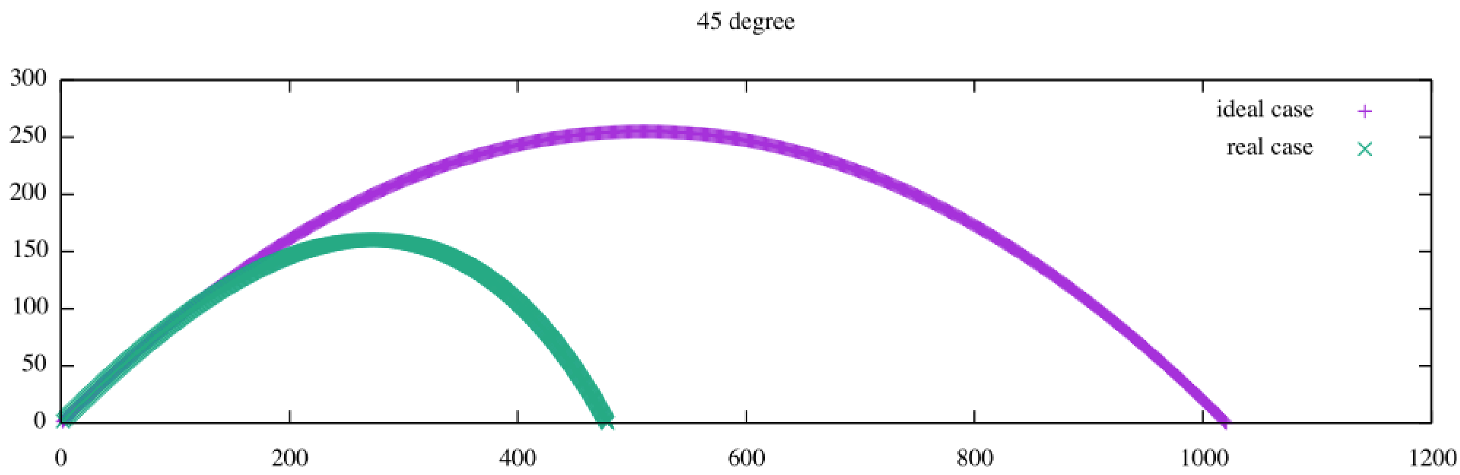
```fortran
! initialization
    x = 0
    v_x = v_x0
    y = 0
    v_y = v_y0
    v_t = v_t0
    theta = theta0
    t = 0

    dt = 0.02
    do while (y>0)
        ! update after one time-step
        t = t + dt
        call rk4(dt, f_x, x, v, theta) ! update x, v_x
        call rk4(dt, f_y, y, v, theta) ! update y, v_y
        v = sqrt(v_x**2 + v_y**2) ! update v
        theta = atan(v_y/v_x)   ! update theta

    ! estimate t(y=0) based on y(t)>0 & y(t+1)<0
    offset = y/(v_y) ! y<0, v_y<0
    t = t - offset
    ! offset everything
    x = x - offset * v_x
    y = y - offset * v_y
```

Result

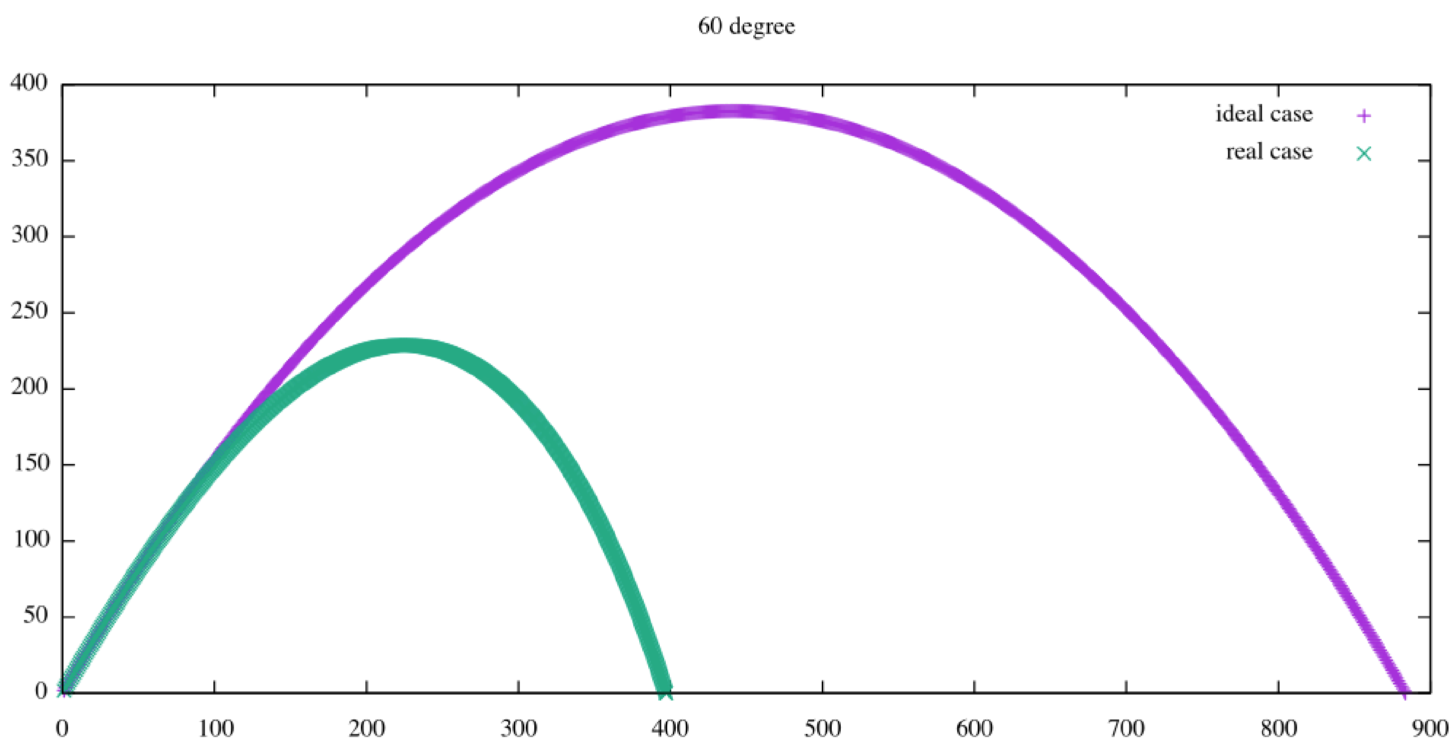
x-y Projectile trajectories



> [ideal case] flying time: 10.204s; distance: 883.702 meter
>
> [real case] flying time: 8.388 s; distance: 465.496 meter



> [ideal case] flying time: 14.431 s; distance: 1020.409 meter
>
> [real case] flying time: 11.359 s; distance: 478.513 meter



> [ideal case] flying time: 17.674 s; distance: 883.699 meter
>
> [real case] flying time: 13.623 s; distance: 397.141 meter

Best shooting angle in $h = 0.02s$ resolution

- ideal case: `45 degree`
- real case: `39 degree`

Appendix

```
Program proj2
    Implicit none
    real*8 :: t, dt, v, theta0, theta, pi, offset, alpha
    real*8, dimension(2) :: x, y
    real*8, external :: f_x, g_x, g_y
    integer :: i
    character (len=10) :: filename

    pi = 4.0d0*atan(1.0d0)

    ! read setting
    print *, "Initial angle in degree:"
    read *, theta0

    do i=1,2
        write(filename , '("res",i1,".dat")') i
        theta = theta0 /180.0d0 * pi
        alpha = 5.0d-2 / 30.0d0 * (i-1.d0) ! k/m

        ! initial conditions
        v = 100.0D0
        t = 0.0d0
        dt = 0.02D0

        x(1) = 0.0D0
        x(2) = v * cos(theta)

        y(1) = EPSILON(0.0d0)
        y(2) = v * sin(theta)

        open(10, file=filename)

        ! Runga-Kutta iteration
        do while( .true. )
            t = t + dt
            call rk4(dt, f_x, g_x, x, v, theta, alpha)
            call rk4(dt, f_x, g_y, y, v, theta, alpha)
            v = sqrt(x(2)**2 + y(2)**2)
            theta = atan(y(2)/x(2))
            if (y(1).gt.0) then
                write (10,*) t, x(1), y(1), x(2), y(2), v, theta / pi * 180
            else
```

```
                exit
            endif
        enddo

        offset = y(1) / y(2)
        t = t - offset
        x(1) = x(1) - offset * x(2)
        y(1) = y(1) - offset * y(2)
        write (10,*) t, x(1), y(1), x(2), y(2), v, theta / pi * 180
        close(10)
        print *, 'flying time:', t, 's; distance:', x(1), 'meter'
    enddo
End program proj2

! 4th-order Runge-Kutta subroutine
subroutine rk4(dt, df, dg, y, v, theta, alpha)
    implicit none
    real*8, external :: df, dg
    real*8, intent(in) :: dt, alpha
    real*8, intent(inout) :: v, theta
    real*8, intent(inout), dimension(2) :: y
    real*8 :: h, k0, k1, k2, k3, l0, l1, l2, l3

    h=dt/2.0D0

    k0 = dt * df(y(1),y(2))
    l0 = dt * dg(y(1),y(2),v,theta,alpha)
    k1 = dt * df(y(1)+h, y(2)+0.5d0*l0)
    l1 = dt * dg(y(1)+0.5d0*k0,y(2)+0.5d0*l0,v,theta,alpha)
    k2 = dt * df(y(1)+0.5d0*k1,y(2)+0.5d0*l1)
    l2 = dt * dg(y(1)+0.5d0*k1,y(2)+0.5d0*l1,v,theta,alpha)
    k3 = dt * df(y(1)+k2,y(2)+l2)
    l3 = dt * dg(y(1)+k2,y(2)+l2,v,theta,alpha)
    y(1) = y(1) + (k0+2*k1+2*k2+k3)/6.0d0
    y(2) = y(2) + (l0+2*l1+2*l2+l3)/6.0d0
    Return
End subroutine rk4

! function which returns the derivatives (RHS)
real*8 function f_x(a, b)
! dx/dt = v(t)
    Implicit none
    real*8 ,intent(in) :: a, b
    f_x = b
    Return
```

```fortran
End function f_x

real*8 function g_x(a, b, v, theta, alpha)
    implicit none
    real*8, intent(in) :: a, b, v, theta, alpha
    g_x = - alpha * v**2 * cos(theta)
!   g_x = 0
    Return
End function g_x

real*8 function g_y(a, b, v, theta, alpha)
    implicit none
    real*8, intent(in) :: a, b, v, theta, alpha
    real*8 :: g
    g = 9.8d0
    g_y = -g - alpha * v**2 * sin(theta)
!   g_y = -g
    Return
End function g_y
```

> input: initial angel `theta0` to horizontal
> output:
>
> > res1.dat: `t, x, y, v_x, v_y, v, theta` for ideal case
> > res2.dat: `t, x, y, v_x, v_y, v, theta` for real case