Ángulo 1515 Ángulo 3026 Ángulo 4536 Ángulo 6047 Ángulo 7557

Método de Runge-Kutta

Carlos Muñoz

November 2018

1 Introducción

En análisis numérico existen existen un grupo de métodos iterativos explícitos e implícitospara resolver ecuaciones diferenciales llamado Runge-Kutta//

Definido un problema de valor inicial:

$$y' = f(x, y), y(t_0) = y_0 \tag{1}$$

Donde hay una función y desconocida con respecto a t la cual nos gustaría aproximar

Ahora tomando un tamaño de paso $h > 0$ y definimos

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2}$$

$$t_{n+1} = t_n + h \tag{3}$$

Para $n = 1, 2, 3, ...$,usando

$$k_1 = hf(t_n, y_n) \tag{4}$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \tag{5}$$

$$k_3 = hf(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \tag{6}$$

$$k_4 = hf(t_n + h, y_n + k_{3)} \tag{7}$$

Donde:

K_1 es el incremento basado en la pendiente al inicio del intervalo usando y (Método de Euler)

k_2 es es el incremento basado en la pendiente en el punto medio del intervalo usando y y k_1

k_3 es es el incremento basado en la pendiente en el punto medio del intervalo usando y y k_2

k_4 es el incremento basado en la pendiente al final del intervalo usando y y k_3

2 Código

```
    program rk
  implicit none

real :: h, f                            ! Ancho de paso y función
real :: x0, y0, u0, x       ! Condiciones iniciales
real :: xi, yi, ui                  ! x sub i+1, y sub i+1 e u sub i + 1 del método
real, dimension(4) :: k, m  ! Arreglos con los valores de k1,...,k4 y m1,...,m3
integer :: i, n             ! Contador, cantidad de pasos a realizar para aproximar y(x)
integer :: nerr         ! Núm. de error en caso fallos inesperados
character(50) :: errmsg     ! Mensaje conrrespondiente al fallo
integer :: unit0


write(*,*) "Ingresa el valor del ancho de paso"

read*, h


write(*,*) "Ingresa la posicion inicial x0"

read*, x0

write(*,*) "Ingresa la posicion inicial y0"

read*, y0

write(*,*) "Ingresa la posicion inicial u0"

read*,u0

write(*,*) "Ingresa la angulo inicial x"

read*, x



n = (x - x0) / h

 open(unit = unit0, file = "trayectoria", status = "unknown")

    write(unit0, *) x0, y0

    yi = 0
    ui = 0
```

```
    xi = x0
    do i = 1, n

        ! Aplicación de la parte principal del método
        ! Nota: en base al algoritmo del método, debe ser yi y ui en donde van y0 y u0 en la
        ! m(1) hasta k(4), sin embargo, se hizo de esa manera para ahorrar algunas líneas de

        m(1) = u0

    k(1) = f(xi, y0, u0)

        m(2) = u0 + h*k(1)/2.

    k(2) = f(xi + h/2., y0 + h*m(1)/2., u0 + h*k(1)/2.)

        m(3) = u0 + h*k(2)/2.

    k(3) = f(xi + h/2., y0 + h*m(2)/2., u0 + h*k(2)/2.)

        m(4) =  u0 + h*k(3)

    k(4) = f(xi + h, y0 + h*m(3), u0 + h*k(3))

        ! u(sub-(i + 1)) = u(sub-n) + (h/6)(k1 + 2k2 + 2k3 + k4)

        ui = u0 + h/6.*( k(1) + 2*( k(2) + k(3) ) + k(4) )

        u0 = ui

        ! y(sub-(i + 1)) = y(sub-n) + (h/6)(k1 + 2k2 + 2k3 + k4)

     yi = y0 + h/6.*( m(1) + 2*( m(2) + m(3) ) + m(4) )

     y0 = yi

     xi = x0 + i*h

     write(unit0, *), xi, yi

  end do

close(unit0)

end program rk
```

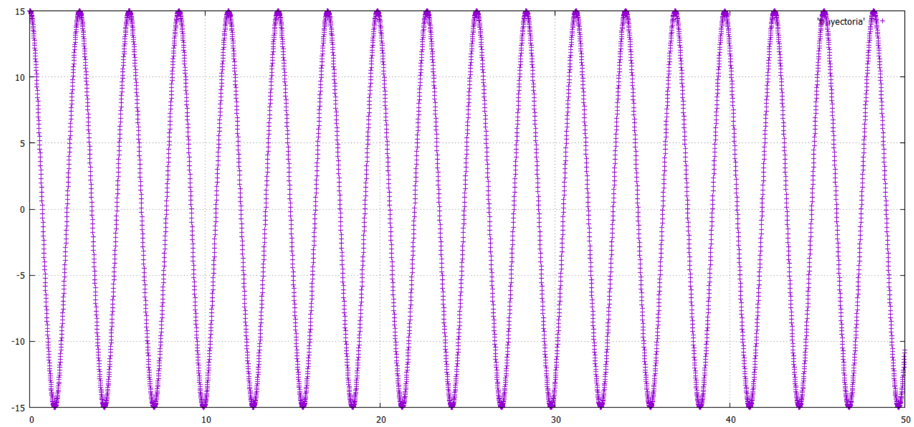

Figure 1: Ángulo 15

```
real function f(x, y, u)
 implicit none
 real :: x, y, u

! f = -4*y - 0.5*u
 f = -(9.81/2.0)*sin(y)

end function f
```

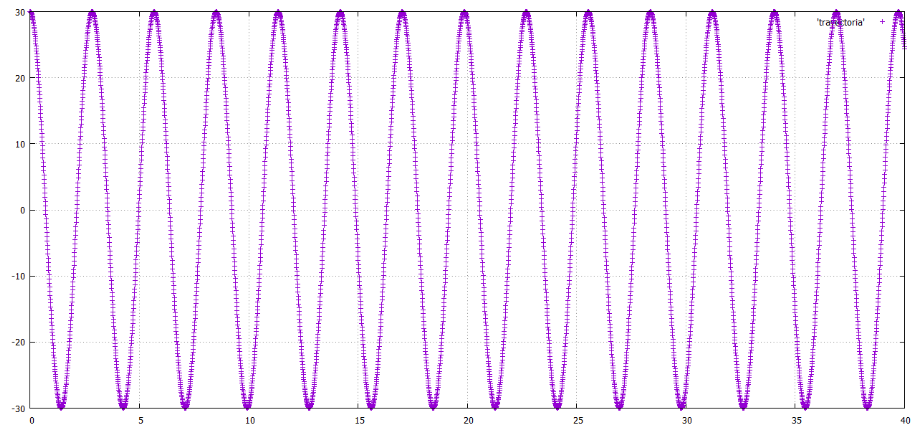

Figure 2: Ángulo 30

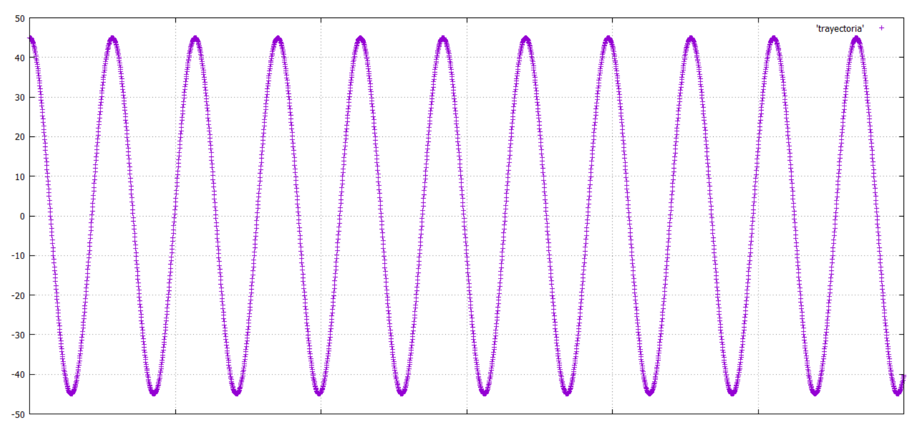

Figure 3: Ángulo 45

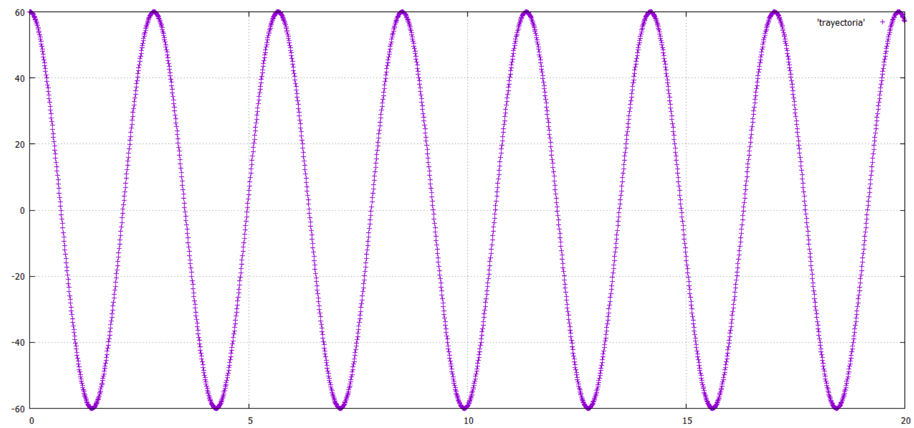

Figure 4: Ángulo 60

Figure 5: Ángulo 75