Método de Runge-Kutta de orden 4

Iván García bojorquez

Noviembre del 2018

1 Método de Runge-Kutta

Los métodos de Runge kutta tienen el error local de truncamiento del mismo orden que los métodos de Taylor, pero prescinden del cálculo y evaluación de las derivadas de la función f(t, y).

Se presenta de nuevo el problema de valor inicial cuya solución se intenta aproximar:

$$\frac{dy}{dt} = f(Y, t), a < y < b \tag{1}$$

$$y(a) = \alpha \tag{2}$$

Se determina primero la malla t0, t1, ... , tN de paso h, donde t0 = a y tN = b. En estos puntos es donde se va a obtener la aproximación de la solución.

En esencia, los métodos de Runge-Kutta son generalizaciones de la fórmula básica de Euler $y_i + 1 = y_i + hf(t_i, y_i)$ en los que el valor de la función f se reemplaza por un promedio ponderado de valores de f en el intervalo $t_i \leq t \leq t_i + 1$, es decir,

$$y_i + 1 = y_i + h(w_1 k_1 + w_2 k_2 + ... + w_m k_m) \tag{3}$$

En esta expresión las ponderaciones $w_i, i = 1, ..., m$ son constantes para las que en general se pide que su suma sea igual a 1, es decir, $w_1 + w_2 + ... + w_m = 1$, y cada k_j es la función f evaluada en un punto seleccionado (t, y) para el cual $t_i \leq t \leq t_i + 1$. Se mostrará que los k_j se definen en forma recursiva.

Se define como orden del método al número m, es decir, a la cantidad de términos que se usan en el promedio ponderado.

y como se llama el tema de este ensayo, hablaremos precisamente del metodo de cuarto orden.

Si ahora $m = 4$, se obtiene, con un desarrollo del tipo del anterior, la siguiente fórmula, para i desde 0 hasta $N - 1$:

$$k_1 = hf(t_1, y_1) \tag{4}$$

$$k_2 = hf(t_i + \frac{h}{2}, y_o + \frac{1}{2}k_1 \tag{5}$$

$$k_2 = hf(t_i + \frac{h}{2}, y_o + \frac{1}{2}k_2 \tag{6}$$

$$k_3 = hf(t_i + h_i y_i + k_3) \tag{7}$$

$$y_i + 1 = y_i + frac16(k_1 + 2k_2 + 2k_3 + k_4) \tag{8}$$

Si bien con facilidad se pueden deducir otras fórmulas, el algoritmo expresado en (16) se denomina método de Runge-Kutta de cuarto orden, o método clásico de Runge-Kutta, abreviado como RK4.Se corrio el programa con los siguientes valores iniciales: $l = 8, h = .1, a = 50$ donde a es el tiempo total, l longitu de la cuerda y h el ancho de paso. Se corrio para los angulos a 15,30,45,60 y 75 grados al final se muestran las graficas resultantes.

1.1 Programa

A continuacion se muestra el codigo para resolver la ecuacion de un pendulo simple de oscilaciones grandes utilizando el metodo RK4:

```
Program Pendulo

IMPLICIT NONE

Real :: l, a, h, m, Ang_0, grados
!Variables para rk4
Real :: k1, k2, k3, k4, l1, l2, l3, l4
Real :: aux, aux1, aux2, aux3, aux4
!Vectores sin dimension
Real, allocatable :: t(:), W(:), Teta(:), Ang(:)
Integer :: i, n
real, external :: func
character:: output1*12,output2*12

Print*, "longitud de la cuerda"
Read*, l

Print*, "Angulo inicial del pendulo"
read*,grados
!transformar a radianes
Ang_0= (3.1416*grados)/180

!solo pido un tiempo el final, suponiendo que
!siempre parte del reposo y mi ti=0
Print*, "Tiempo de oscilacion"
Read*, a

!pedir ancho de paso
Print*, "Ancho de paso(h<|)"
```

```
Read*, h

!Archivos de datos
print*,"nombre archivo de salida t vs radianes"
read*,output1
print*,"nombre archivo de salida t vs grados"
read*,output2

!SE calcula el numero de particiones
m=a/h
!se toma un numero entero de particiones
n=NINT(m)

!Allocate dimensiones
!angulo(radianes)
Allocate(Teta(n))
!angulo(grados)
Allocate(Ang(n))
!tiempo
Allocate(t(n))
!velocidad angular
Allocate(W(n))

Print*, "Gracias!"

!Ponemos valores iniciales en los arreglos

Teta(1)=Ang_0
Ang(1)=grados

t(1)=0
W(1)=0

Do i=2,n

!Primer pendiente
k1= h*W(i-1)
l1= h*func(Teta(i-1),l)

!Segunda pendiente
aux2= Teta(i-1)+(k1/2)
k2= h*(W(i-1)+(l1/2))
l2= h*func(aux2,l)

!tercer pendiente
aux3= Teta(i-1)+(k2/2)
```

```
k3= h*(W(i-1)+(l2/2))
l3= h*func(aux3,l)

!cuarta pendiente
aux4= Teta(i-1)+k3
k4= h*(W(i-1)+l3)
l4= h*func(aux4,l)

!hacemos una suma para teta
Aux= k1+(2*k2)+(2*k3)+k4
!hacemos una suma para la rapidez angular
Aux1= l1+(2*l2)+(2*l3)+l4

!Calculamos las nuevas rapidez y angulo
W(i)= W(i-1) + (aux1/6)
Teta(i)= Teta(i-1) + (aux/6)
Ang(i) = Teta(i)*(180/3.1416)
!aux/6 es el promedio de las pendientes

!Calcula paso del tiempo
t(i)=h*(i-1)

End do

!Escribe los datos
Open(1, file=output1)
Do i=1,n
Write(1,*) t(i), Teta(i)
End do
Close(1)

Open(3,file=output2)
Do i=1,n
Write(3,*)t(i), Ang(i)
End do
Close (3)

End program Pendulo

!funcion a resolver
Function func(Teta,l)
implicit none
Real :: Teta, func, l
func=(-9.81/l)*(Sin(Teta))
End function
```

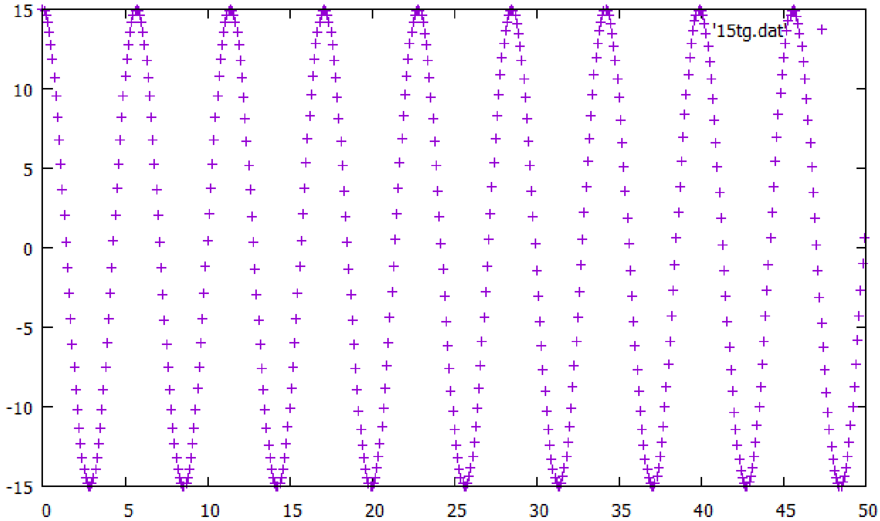

Figure 1: Tabla t x θ 15 grados

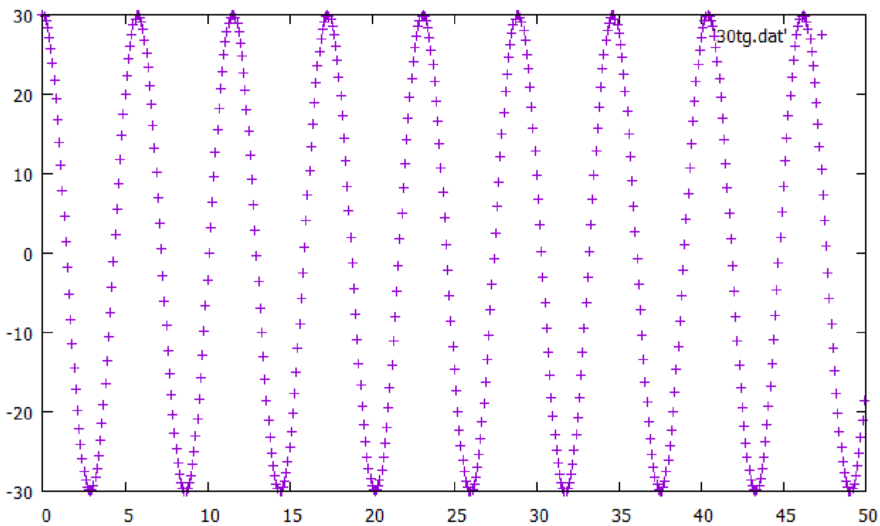

Figure 2: Tabla t x θ 30 grados

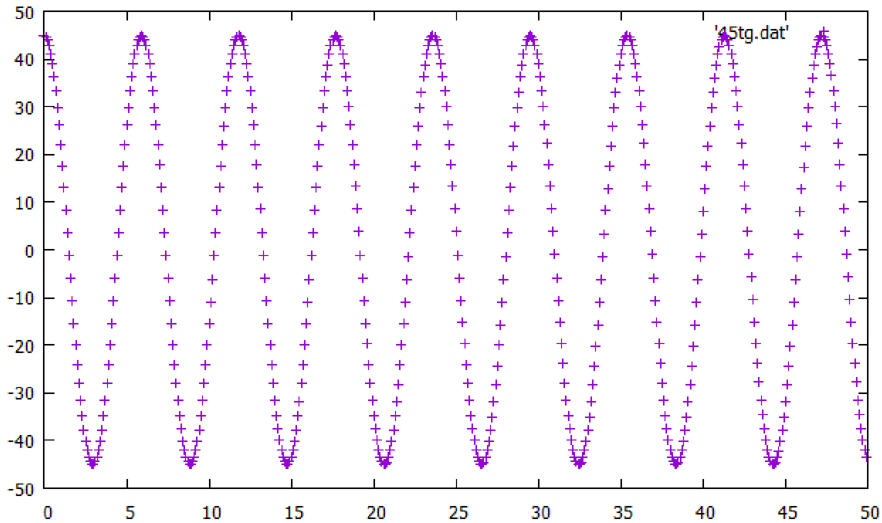

Figure 3: Tabla t x θ 45 grados

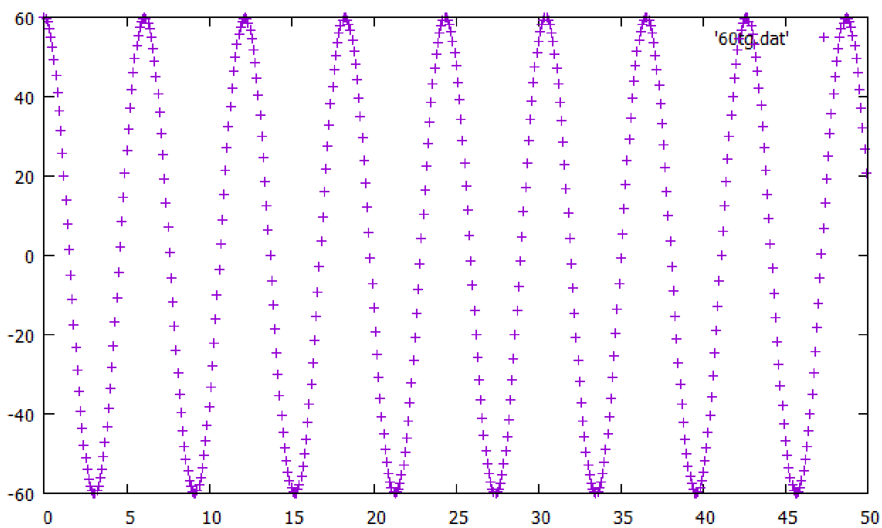

Figure 4: Tabla t x θ 60 grados

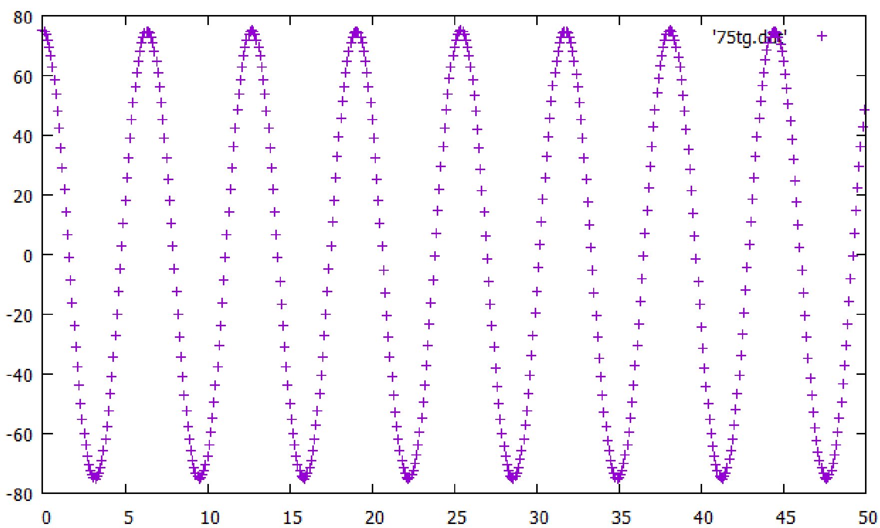

Figure 5: Tabla t x θ 75 grados