Método de Runge-Kutta de cuarto orden para un sistema de EDO

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

1. Explicación

Este programa se diseñó para resolver un conjunto de cualquier número de EDO de primer orden. En el subprograma `FUNCT` se definen el número de ecuaciones IM, asi como los IM valores de las condiciones iniciales. Para utilizar el programa con un nuevo problema, el usuario debera cambiar las ecuaciones en `FUNCT`, el valor de IM y las condiciones iniciales. La estructura del programa es esencialmente la misma del programa RK4, pero se calcula cada paso intermedio en un ciclo `DO` para el número IM de ecuaciones.

2. Variables

Listado de variables

Y(1) : y
Y(2) : z
Y(I) : I-ésima incógnita
YN(I): y_n para $I = 1$ y z_n para $I = 2$, etc.
YA(I) : $y_n + k_1/2$ ó $y_n + k_2/2$ ó $y_n + k_3/2$ para $I = 1$
$z_n + l_1/2$ ó $z_n + l_2/2$ ó $z_n + l_3/2$ para $I = 2$
K(J,1),J = 1,2,3,4 : k_1, k_2, k_3, k_4
K(J,2),J = 1,2,3,4 : l_1, l_2, l_3, l_4
K(J,M),J = 1,2,3,4 : similar al anterior para la M-ésima ecuación diferencial
IM : número de ecuaciones en el conjunto
NS : número de intervalos de tiempo en un intervalo de impresión, TD
XP : límite máximo de t
TD : intervalo de impresión para t

3. Código

A continuación se indica el código del programa.

```
DIMENSION YA(0:10), YN(0:10), EK(0:4,0:10), Y(0:10)
PRINT *
PRINT *, 'Esquema RK4 para un conjunto de ecuaciones'
!Numero de ecuaciones
IM=2
!Condicion inicial para y1 en t=0
Y(1) = 1
```

```
!Condicion inicial para y2 en t=0
Y(2) = 0
1 PRINT *
PRINT *, 'Intervalo de impresion de T?'
READ *, PI
PRINT *, '¿Numero de pasos en un intervalo de impresión de T?'
READ *, NS
PRINT *, 'T maximo para detener los calculos?'
READ *, XL
H = PI/NS
PRINT *, 'H = ' , H
XP = 0
HH = H/2
PRINT * !Inicializacion del numero de linea
LI = 0
PRINT *, 'Linea           T           Y(1)           Y(2), ... '
WRITE (*,98) LI, XP, (Y(I),I=1,IM)
28 LI = LI+1


DO N = 1
   XB=XP                    !Tiempo anterior
   XP=XP+H                  !Tiempo nuevo
   XM=XB+HH                 !Tiempo en el punto nuevo
   J=1                      !Esta parte calcula k1
   DO I= 1, IM
       YA(I)=Y(I)
   END DO

   XA=XB
   CALL FUNCT(EK,J,YA,H)
   J=1                   !Esta parte calcula k2
   DO I = 1, IM
       YA(I)=Y(I)+EK(1,I)/2
   END DO

   XA=XM
   CALL FUNCT(EK,J,YA,H)
   J=3                   !Esta parte calcula k3
   DO I=1, IM
       YA(I)=Y(I)+EK(2,I)/2
   END DO

   XA=XM
   CALL FUNCT(EK,J,YA,H)
   J=4                   !Esta parte calcula k4
   DO I = 1, IM
       YA(I) = Y(I) +EK(3,I)
   END DO
```

```
   XA=XP
   CALL FUNCT(EK,J,YA,H)
   DO I = 1, IM          !Esquema RK4
       Y(I) = Y(I) + (EK(1,I)+EK(2,I)*2+EK(3,I)*2+EK(4,I))/6
   END DO

END DO
WRITE (*,98) LI, XP, (Y(I), I=1,IM)
98 FORMAT (1X, I2, F10.6, 2X, 1P4E16.6/ (15X, 1P4E16.6))
IF (XP .LT. XL) GOTO 28
200 PRINT *
PRINT *, 'Oprime 1 para Continuar ó 0 para Terminar'
READ *, K
IF (K .EQ. 1) GOTO 1
PRINT *
END
!++++++++++++++++++++++++++++++++++
SUBROUTINE FUNCT(EK, J, YA, H)
DIMENSION EK(0:4, 0:10), YA(0:10)          !Define un conjunto de ecuaciones
EK(J,1) = YA(2) * H
EK(J,2) = -YA(1) * H
RETURN
END
```

Nótese que no está implementado en el código, el almacenamiento de los datos en un archivo, por lo que habrá que agregarlo y posteriormente trabajar con ese archivo.