Taller

Taller EDO

Realizado por: Camilo Hoyos y Catalina Morales

A continuación, se muestra la resolución del taller de Ecuaciones Diferenciales

Primer Punto

$$\frac{dT}{dt} = \frac{-EyS(T^4(t) - (T_e)^4)}{mC}$$

Usando el método de Euler (en R) y 20 intervalos iguales y t variando de 0 a 200 segundos, resuelva numéricamente la ecuación, si el cuerpo es un cubo de lados de longitud 1m y masa igual a 1Kg. Asuma, que T0 = 180K, Te = 200K, y = 0.5 y C = 100J/(Kg/K). Hacer una representación gráfica del resultado.

Solución

Primero hallamos S (área de la superficie), como sabemos que el cuerpo es un cubo decimos que:

$$S = 6 * longitud^2$$

$$S = 6 * (1)^2$$

$$\boldsymbol{S = 6m^2}$$

Reemplazamos los valores dados en la formula con el fin de simplificarlo y obtenemos:

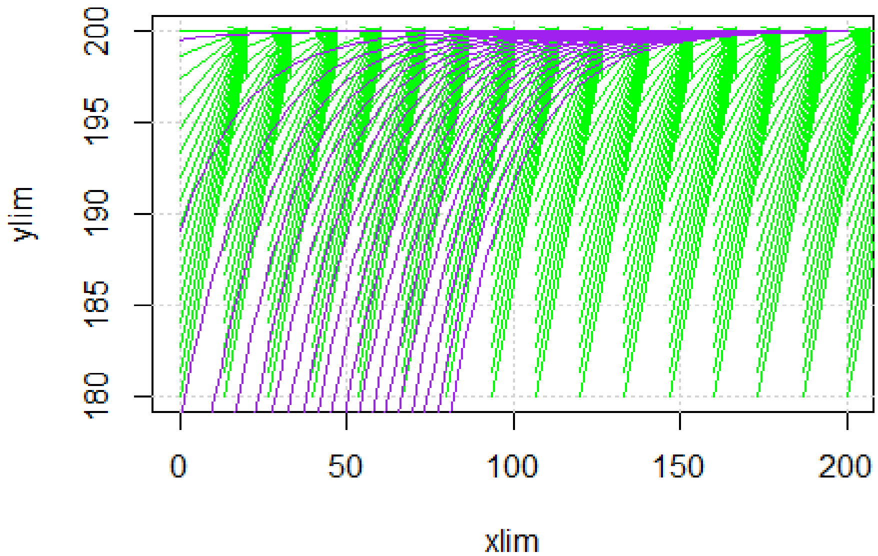
$$\frac{dT}{dt} = -1.68 * 10^{-9} * T(t)^4 + 2.6880$$

Luego introducimos la formula en el método de Euler y graficamos el resultado.

```
library(pracma)
metodoEuler <- function(f, h, xi, yi, xf)
{
  N = (xf - xi) / h
  x = y = numeric(N+1)
  x[1] = xi;
  y[1] = yi;
  i = 1
  while (i <= N)
  {
    x[i+1] = x[i]+h
    y[i+1] = y[i]+(h*f(x[i],y[i]))
```

```r
    i = i+1
  }
  return (data.frame(X = x, Y = y))
}
f <- function(t, v)
{
  a <- -1.68e-9*v^4+2.688
  return(a)
}
e1 = metodoEuler(f ,10, 0, 180, 200)
e1[nrow(e1),]

xx <- c(0, 200)
yy <- c( 180, 200)
vectorfield(f, xx, yy, scale = 20)
for (xs in seq(0, 200, by = 10.5))
{
  sol <- rk4(f, 0, 200, xs, 100)
  lines(sol$x, sol$y, col="purple")
}
```

i	X	Y
1	0	180.0000
2	10	189.2440
3	20	194.5765
4	30	197.3757
5	40	198.7590
6	50	199.4200
7	60	199.7304
8	70	199.8751
9	80	199.9422
10	90	199.9732
11	100	199.9876
12	110	199.9943
13	120	199.9974
14	130	199.9988
15	140	199.9994
16	150	199.9997
17	160	199.9999
18	170	199.9999
19	180	200.0000
20	190	200.0000
21	200	200.0000



Segundo Punto

Obtenga cinco puntos de la solución de la ecuación, utilizando el método de Taylor (los tres primeros términos) con h=0.1 implemente en R. Grafique su solución y compare con la solución exacta, cuál es el error de truncamiento en cada paso

$$\frac{dy}{dx} - (x+y) = 1 - x^2; y(0) = 1$$

Solución

Primero solucionamos la ecuación diferencial:

$$h = 0.1; \qquad y(0) = 1; \qquad y'(0) = 2; \qquad y''(0) = 3$$

$$y'(x) = 1 - x^2 + x + y$$

$$y''(x) = -2x + 1 + y'$$

$$\boldsymbol{y''(x) = -2x + 1 + (1 - x^2 + x + y)}$$

$$y = y(0) + y'(0)x + \frac{1}{2}x^2 f''(0)$$

$$\boldsymbol{y = 1 + 2x + \frac{3}{2}x^2}$$

Ahora implementamos en R la solución, graficamos y hallamos el error de truncamiento.

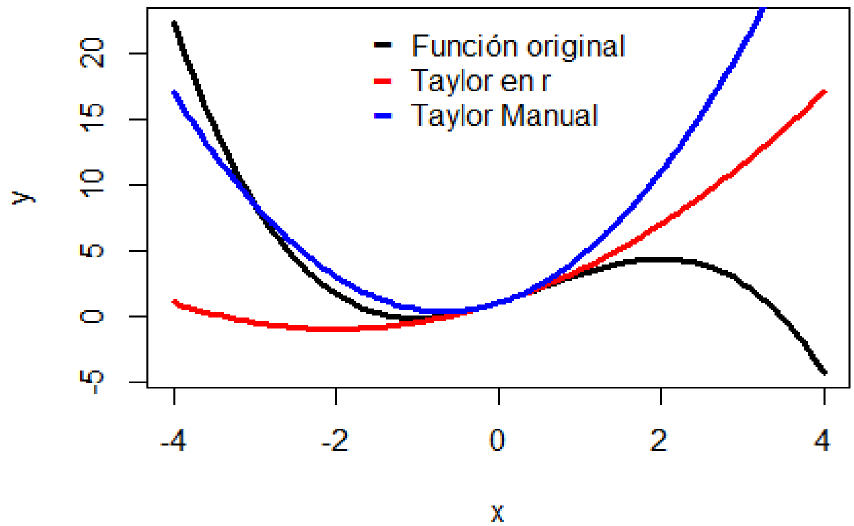
```
library(pracma)
#Función original
f <- function(x,y)
```

```r
{
  a <- 2*x
  b <- (x^3)/3
  c <- (x^2)/2
  d <- a-b+c+1
  return(d)
}
#Taylor manual
g <- function(x)
{
  a <- 1+2*x+(3/2)*x^2
  return(a)
}
#Taylor en R
p = taylor(f, 0, n = 2)
#Valores de Taylor

xi <- seq(0,1,0.1)
xii<-0
for (i in 1:5)
  xii[i]=xi[i]
print("Valores de Taylor: ")

print(data.frame(X = xii, Y = polyval(p,xii)))
```

```
[1] "Valores de Taylor: "

 i     X       Y
 1   0.0   1.000
 2   0.1   1.205
 3   0.2   1.420
 4   0.3   1.645
 5   0.4   1.880
```

```r
#Error de truncamiento
error<-0
tayr<-0
taym<-0
for (i in 1:5)
{
  tayr[i] = polyval(p,xii[i])
  taym[i] = g(xii[i])
  error[i] = abs(tayr[i]-taym[i])
}
#Gráficas
x <- seq(-4, 4, length.out=100)
y <- f(x)
yp <- polyval(p, x)
plot(x, y, type = "l", col = "black", lwd = 3)
lines(x, yp, col = "red", lwd = 3)
lines(x, g(x), col = "blue" ,lwd = 3)
```

```
legend("top",legend=c("Función original","Taylor en r","Taylor Manual"),pch="-",col=c("black","red","blue"),bty="n",pt.cex = 2,cex = 1)
```




```
print("-----------------")

print("Valores con paso de h = 0.1")

 [1] "-----------------"

 [1] "Valores con paso de h = 0.1"

approx.df <- data.frame(cbind(tayr,taym,error))
colnames(approx.df) <- c('Exacta', 'Manual', "Error")
approx.df
```

i	Exacta	Manual	Error
1	1.000	1.000	0.00
2	1.205	1.215	0.01
3	1.420	1.460	0.04
4	1.645	1.735	0.09
5	1.880	2.040	0.16

Tercer Punto

Obtenga 20 puntos de la solución de la ecuación, utilizando el método de Euler (los tres primeros términos) con h=0.1. Grafique su solución y compare con la solución exacta, cuál es el error de truncamiento en cada paso.

$$\frac{dy}{dx} - (x + y) = 1 - x^2; y(0) = 1$$

Solución

Basados en el punto anterior ya sabemos que la solución de la ecuación es:

$$h = 0.1; \quad y(0) = 1; \quad y'(0) = 2; \quad y''(0) = 3$$

$$\boldsymbol{y'(x) = 1 - x^2 + x + y}$$

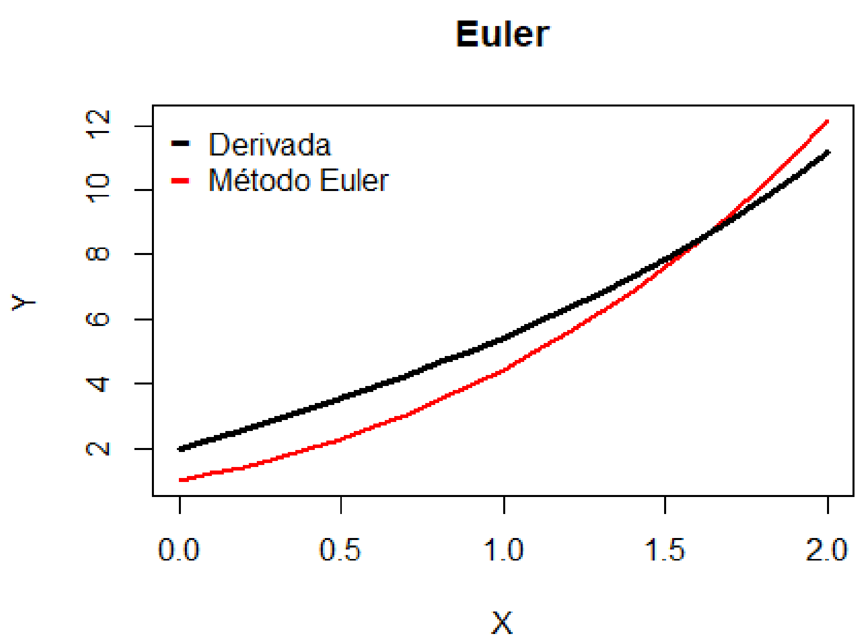
Ahora reemplazamos la ecuación en el método de Euler, graficamos y hallamos el error de truncamiento en cada paso.

```
library(pracma)
f <- function(x, y) {x+y+1-x^2}
#Error de truncamiento
truncamiento<-function(dy,y)
{
  error<-0
  fu<-dy
  fui<-y
  for (i in 1:length(y))
  {
    error[i] = abs(fu[i]-fui[i])
  }
  return (error)
}

#Método de Euler
metodoEuler <- function(f, h, xi, yi, xf)
{
  N = (xf - xi) / h
  x = y = numeric(N+1)
  x[1] = xi;
  y[1] = yi;
  i = 1
  while (i <= N)
  {
    x[i+1] = x[i]+h
    y[i+1] = y[i]+(h*f(x[i],y[i]))
    i = i+1
  }
  error<-truncamiento(f(x,y),y)
  return (data.frame(DY = f(x,y), X = x, Y = y, Error = error))
}
e1 = metodoEuler(f, 0.1, 0, 1, 2)
print(e1)
```

i	DY	X	Y	Error
1	2.000000	0.0	1.000000	1.00
2	2.290000	0.1	1.200000	1.09
3	2.589000	0.2	1.429000	1.16
4	2.897900	0.3	1.687900	1.21
5	3.217690	0.4	1.977690	1.24
6	3.549459	0.5	2.299459	1.25
7	3.894405	0.6	2.654405	1.24
8	4.253845	0.7	3.043845	1.21

```
 9      4.629230   0.8     3.469230    1.16
 10     5.022153   0.9     3.932153    1.09
 11     5.434368   1.0     4.434368    1.00
 12     5.867805   1.1     4.977805    0.89
 13     6.324586   1.2     5.564586    0.76
 14     6.807044   1.3     6.197044    0.61
 15     7.317749   1.4     6.877749    0.44
 16     7.859523   1.5     7.609523    0.25
 17     8.435476   1.6     8.395476    0.04
 18     9.049023   1.7     9.239023    0.19
 19     9.703926   1.8    10.143926    0.44
 20    10.404318   1.9    11.114318    0.71
 21    11.154750   2.0    12.154750    1.00
```

```
#Valores de y
plot(e1$X,e1$Y, type='l', lwd=2, col='red', main="Euler",xlab="X",
ylab="Y")
#Valores de las derivadas
lines(e1$X, e1$DY, col = "black", lwd = 3)
legend("topleft",legend=c("Derivada","Método Euler"),pch="-
",col=c("black","red"),bty="n",pt.cex = 2,cex = 1)
```



Cuarto Punto

Implemente en R el siguiente algoritmo y aplíquelo para resolver la ecuación anterior

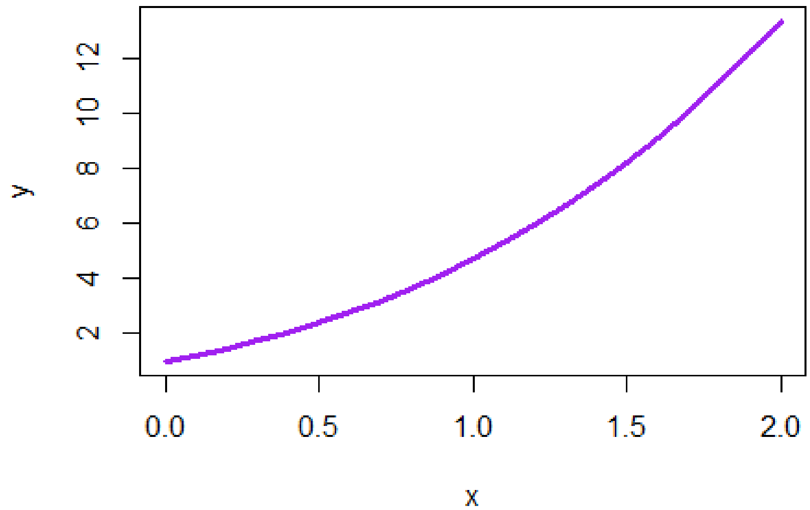
Solución

```
f <- function(x, y) {x+y+1-x^2}
x0 <- 0
y0 <- 1
h <- 0.1
```

```r
m <- 20
x <- c(x0)
y <- c(y0)
for(i in 1:m)
{
 k1 <- h*f(x0,y0)
 k2 <- h*f(x0+h, y0+k1)
 y0 <- y0+(1/2)*(k1+k2)
 x0 <- x0+h
 x <- c(x,x0)
 y <- c(y,y0)
}
print(data.frame(X = x, Y = y))
```

i	X	Y
1	0.0	1.000000
2	0.1	1.214500
3	0.2	1.459973
4	0.3	1.737570
5	0.4	2.048564
6	0.5	2.394364
7	0.6	2.776522
8	0.7	3.196757
9	0.8	3.656966
10	0.9	4.159248
11	1.0	4.705919
12	1.1	5.299540
13	1.2	5.942942
14	1.3	6.639251
15	1.4	7.391922
16	1.5	8.204774
17	1.6	9.082025
18	1.7	10.028338
19	1.8	11.048863
20	1.9	12.149294
21	2.0	13.335919

```r
plot(x, y, type = "l", col = "purple", lwd = 3)
```



Quinto Punto

Utilizar la siguiente variación en el método de Euler, para resolver una ecuación diferencial ordinaria de primer orden, la cual calcula el promedio de las pendientes en cada paso

$$y_{i+1} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, y_{i+1}))$$

Implemente un código en R, para este método y obtenga 10 puntos de la solución con h=0.1, grafíquela y compárela con el método de Euler:

$$\frac{dy}{dx} - x - y - 1 + x^2 = 0\,; \quad y(0) = 1$$

Solución

Primero hay que despejar la ecuación

$$y'(x) = 1 - x^2 + x + y$$

Después reemplazamos en el método de Euler modificado y en el método normal (usado anteriormente), graficamos y comparamos los resultados para 10 valores.

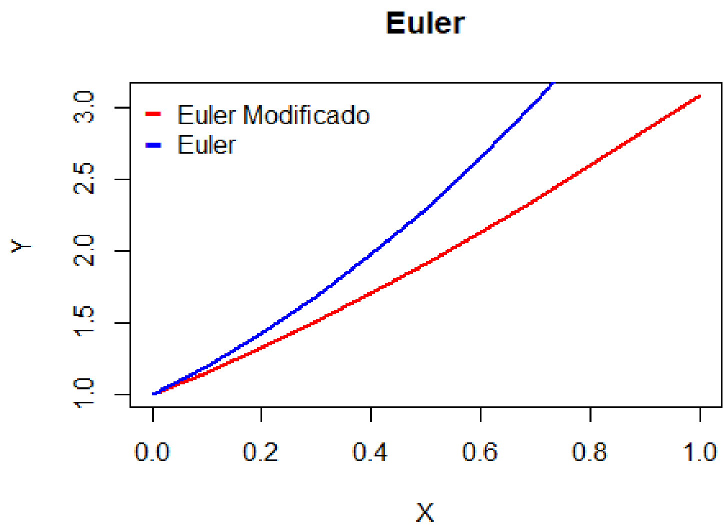
```
library(pracma)
#Ecuación
f <- function(x, y) {x+y+1-x^2}
#Método de Euler modificado
metodoEulerMod <- function(f, h, xi, yi, xf)
{
  N = (xf - xi) / h
  x = y = numeric(N+1)
  x[1] = xi;
  y[1] = yi;
  i = 1
```

```
  while (i <= N)
  {
    x[i+1] = x[i]+h
    y[i+1] = y[i]+(h/2*(f(x[i],y[i])+f(x[i+1],y[i+1])))
    i = i+1
  }
  return (data.frame(X = x, Y = y))
}
e2 = metodoEulerMod(f, 0.1 , 0, 1, 1)
e2[nrow(e2),]

#Método de Euler sin modificar
metodoEuler <- function(f, h, xi, yi, xf)
{
  N = (xf - xi) / h
  x = y = numeric(N+1)
  x[1] = xi;
  y[1] = yi;
  i = 1
  while (i <= N)
  {
    x[i+1] = x[i]+h
    y[i+1] = y[i]+(h*f(x[i],y[i]))
    i = i+1
  }
  return (data.frame(X = x, Y = y))
}
e1 = metodoEuler(f, 0.1, 0, 1, 1)
e1[nrow(e1),]

#Tabla de datos
approx.df <- data.frame(cbind(e2,e1))
colnames(approx.df) <- c('X','Euler Modificado','X' ,'Euler')
approx.df
```

i	X	Euler Modificado	Euler
1	0.0	1.000000	1.000000
2	0.1	1.154500	1.200000
3	0.2	1.324725	1.429000
4	0.3	1.509461	1.687900
5	0.4	1.707434	1.977690
6	0.5	1.917306	2.299459
7	0.6	2.137671	2.654405
8	0.7	2.367055	3.043845
9	0.8	2.603908	3.469230
10	0.9	2.846603	3.932153
11	1.0	3.093433	4.434368

```
#Gráfica de los dos métodos
plot(e2$X,e2$Y, type='l', lwd=2,main="Euler", col='red', xlab="X",
ylab="Y")
```

```
lines(e1$X,e1$Y, type='l', lwd=2, col='blue',xlab="X", ylab="Y")
legend("topleft",legend=c('Euler Modificado','Euler'),pch="-
",col=c("red","Blue"),bty="n",pt.cex = 2,cex = 1)
```



Séptimo Punto

Pruebe el siguiente código en R del método de Runge Kutta de tercer y cuarto orden y obtenga 10 puntos de la solución con h=0.1, grafíquela y compárela con el método de Euler:

$$\frac{dy}{dx} - x - y - 1 + x^2 = 0\,; \quad y(0) = 1$$

Solución

Ya sabemos que la ecuación diferencial es

$$y'(x) = 1 - x^2 + x + y$$

Luego ingresamos la ecuación el método de Runge Kutta de 3er orden, 4to orden y Euler, graficamos y comparamos los valores dados.

Cabe resaltar que como Runge Kutta de 3er orden, 4to orden da los mismos valores una línea se sobrepone a la otra en la gráfica por eso no se pueden apreciar las tres líneas en la gráfica.

```
list.of.packages <- c("phaseR")
new.packages <- list.of.packages[!(list.of.packages %in%
installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)
library(phaseR)
f<-function(fcn,x,y){
  return(eval(fcn))
}
# Solo para prueba con dy=x+y, y(0)=1
```

```
obtenerErrorAbsoluto<-function(x,y){
  solucion=exp(x)*((-x*exp(-x))-exp(-x)+2)
  return(abs(y-solucion))
}

graficarCampoPendiente<-function(x0, xn, y0, yn, fcn, numpendientes,
metodo){
  apma1 <- function(t, y, parameters){
    a <- parameters[1]
    dy <- a*(f(fcn, t, y))
    list(dy)
  }
  apma1.flowField <- flowField(apma1, x = c(x0, xn),
                               y   = c(y0, yn), parameters = c(1),
                               points = numpendientes, system =
"one.dim",
                               add = FALSE, xlab = "x", ylab = "y",
                               main = metodo)
  grid()
}

graficarSolucionNumerica<-function (x, y){
  points (x, y, pch=20, col="black")
  for (i in 2:length(x)){
    segments(x[i-1], y[i-1], x[i], y[i], col="red")
  }
}

rk4<-function(dy, ti, tf, y0, h, graficar=TRUE, numpendientes=10){
  t<-seq(ti, tf, h)
  y<-c(y0)
  cat("x     |y          |k1         |k2         |k3         |k4        |error
absoluto\n")
  for(i in 2:length(t)){
    k1=h*f(dy, t[i-1], y[i-1])
    k2=h*f(dy, t[i-1]+h/2, y[i-1]+k1*(0.5))
    k3=h*f(dy, t[i-1]+h/2, y[i-1]+k2*(0.5))
    k4=h*f(dy, t[i-1]+h, y[i-1]+k3)
    y<-c(y, y[i-1]+1/6*(k1+2*k2+2*k3+k4))
    cat(t[i-1]," | ", y[i-1]," | ",k1," | ",k2," | ",k3," | ",k4," |
",obtenerErrorAbsoluto(t[i-1],y[i-1]),"\n")
  }
  if (graficar){
    graficarCampoPendiente(min(t), max(t), min(y), max(y), dy,
numpendientes, "Runge Kutta 3 Vs Runge Kutta 4 Vs Euler")
    graficarSolucionNumerica(t, y)
  }
  rta<-list(w=y, t=t)
}
```

```
rk3<-function(dy, ti, tf, y0, h, graficar=TRUE, numpendientes=10){
  t<-seq(ti, tf, h)
  y<-c(y0)
  cat("x    |y          |k1          |k2         |k3        |error
absoluto\n")
  for(i in 2:length(t)){
    k1=h*f(dy, t[i-1], y[i-1])
    k2=h*f(dy, t[i-1]+h/2, y[i-1]+k1*(0.5))
    k3=h*f(dy, t[i-1]+h, y[i-1]-k1+2*k2)
    y<-c(y, y[i-1]+1/6*(k1+4*k2+k3))
    cat(t[i-1]," | ", y[i-1]," | ",k1," | ",k2," | ",k3," |
",obtenerErrorAbsoluto(t[i-1],y[i-1]),"\n")
  }
  rta<-list(w=y, t=t)
}

r<-rk4(expression(1-x^2+x+y), 0, 1, 1, 0.1)

x     |y          |k1           |k2           |k3           |k4
0     |  1        |  0.2        |  0.21475    |  0.2154875  |  0.2305487
0.1   |  1.215171 |  0.2305171  |  0.2457929  |  0.2465567  |  0.2621727
0.2   |  1.461402 |  0.2621402  |  0.2779972  |  0.2787901  |  0.295019
0.3   |  1.739858 |  0.2949858  |  0.3114851  |  0.31231    |  0.3292168
0.4   |  2.051823 |  0.3291823  |  0.3463914  |  0.3472519  |  0.3649075
0.5   |  2.398719 |  0.3648719  |  0.3828655  |  0.3837652  |  0.4022485
0.6   |  2.782116 |  0.4022116  |  0.4210722  |  0.4220152  |  0.4414132
0.7   |  3.20375  |  0.441375   |  0.4611937  |  0.4621846  |  0.4825934
0.8   |  3.665537 |  0.4825537  |  0.5034314  |  0.5044753  |  0.5260012
0.9   |  4.169599 |  0.5259599  |  0.5480078  |  0.5491102  |  0.5718709

r2<-rk3(expression(1-x^2+x+y), 0, 1, 1, 0.1)

x     |y          |k1           |k2           |k3           |error
0     |  1        |  0.2        |  0.21475    |  0.23195    |  0
0.1   |  1.215158 |  0.2305158  |  0.2457916  |  0.2636226  |  0.1048165
0.2   |  1.461376 |  0.2621376  |  0.2779945  |  0.2965227  |  0.2185703
0.3   |  1.739816 |  0.2949816  |  0.3114806  |  0.3307795  |  0.3400979
0.4   |  2.051763 |  0.3291763  |  0.3463851  |  0.3665357  |  0.4681134
0.5   |  2.398638 |  0.3648638  |  0.382857   |  0.4039488  |  0.6011956
0.6   |  2.782012 |  0.4022012  |  0.4210612  |  0.4431933  |  0.737774
0.7   |  3.203618 |  0.4413618  |  0.4611799  |  0.4844616  |  0.8761127
0.8   |  3.665375 |  0.4825375  |  0.5034144  |  0.5279667  |  1.014293
0.9   |  4.169402 |  0.5259402  |  0.5479872  |  0.5739437  |  1.150196

metodoEuler <- function(f, h, xi, yi, xf)
{
  N = (xf - xi) / h
  x = y = numeric(N+1)
  x[1] = xi;
  y[1] = yi;
  i = 1
```

```
  while (i <= N)
  {
    x[i+1] = x[i]+h
    y[i+1] = y[i]+(h*f(x[i],y[i]))
    i = i+1
  }
  return (data.frame(X = x, Y = y))
}
f <- function(x, y) {1-x^2+x+y}
e1 = metodoEuler(f, 0.1, 0, 1, 1)
print(e1)
```

i	X	Y
1	0.0	1.000000
2	0.1	1.200000
3	0.2	1.429000
4	0.3	1.687900
5	0.4	1.977690
6	0.5	2.299459
7	0.6	2.654405
8	0.7	3.043845
9	0.8	3.469230
10	0.9	3.932153
11	1.0	4.434368

```
lines(e1$X,e1$Y, type='l', lwd=2, col='blue', xlab="X", ylab="Y")
lines(r2$t,r2$w, type='l', lwd=2, col='yellow', xlab="X", ylab="Y")
legend("topleft",legend=c('Runge Kutta 3er orden','Runge Kutta 4to
orden', 'Euler'),pch="-",col=c("yellow","red","blue"),bty="n",pt.cex =
2,cex = 1)
```