Taller Ecuaciones Diferenciales

Juan Sebastian Triana Perez, Miguel Angel Beltran Rodriguez

26 de octubre de 2018

Punto 1.

Considere un cuerpo con temperatura interna T el cual se encuentra en un ambiente con temperatura constante T_e. Suponga que su masa m concentrada en un solo punto. Entonces la transferencia de calor entre el cuerpo y el entorno externo puede ser descrita con la ley de Stefan-Boltzmann.

$$v\left(t\right) = \epsilon\gamma S\left(T^4\left(t\right) - T_e^4\right)$$

Donde, t es tiempo y ϵ es la constante de Boltzmann ($\epsilon = 5.6x10^{-8}J/m^2K^2s$), γ es la constante de "emisividad" del cuerpo, S el area de la superficie y v es la tasa de transferencia del calor. La tasa de variacion de la energia $\frac{dT}{dt} = \frac{-v(t)}{mC}$ (C indica el calor especifico del material que constituye el cuerpo). En consecuencia,

$$\frac{d\left(T\right)}{d\left(t\right)} = \frac{-\epsilon\gamma S\left(T^4\left(t\right) - T_e^4\right)}{mC}$$

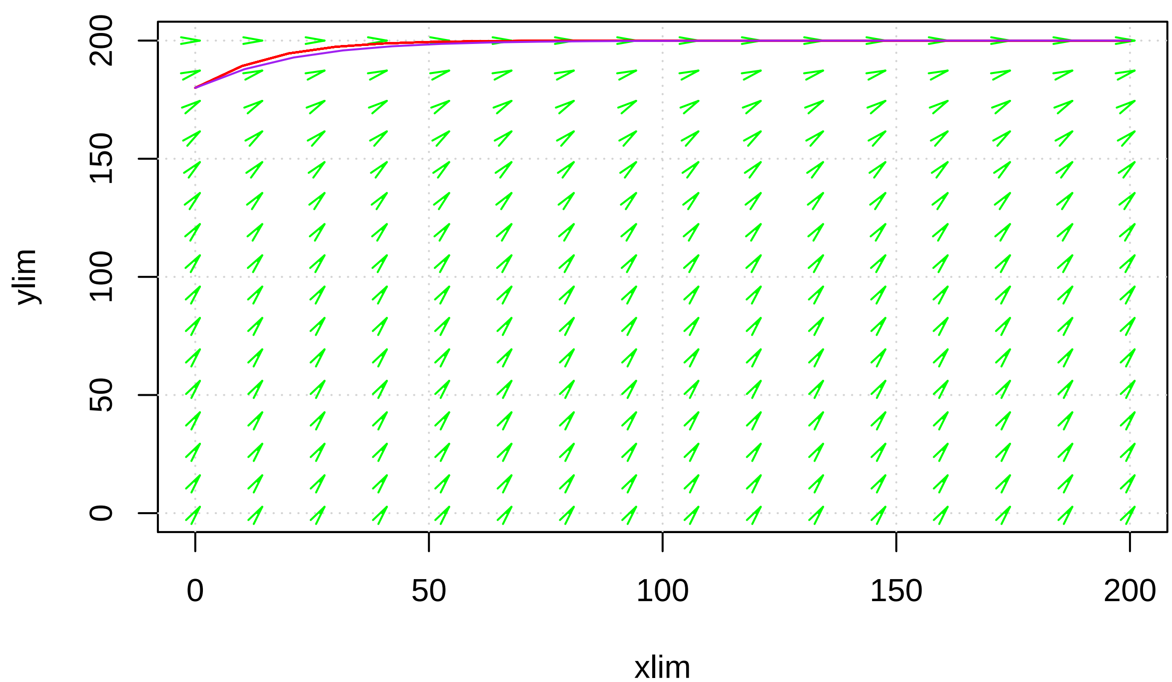
Usando el m??todo de Euler (en R) y 20 intervalos iguales y t variando de 0 a 200 segundos, resuelva numericamente la ecuacion, si el cuerpo es un cubo de lados de longitud 1m y masa igual a 1Kg. Asuma, que $T_0 = 180$K, $T_e = 200$K, $g = 0.5$ y $C = 100$J/(Kg/K). Hacer una representaci??n grafica del resultado.

```r
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]))
    i = i+1
  }
  return (data.frame(X = x, Y = y))
}

imprimir<-function(e1, col="red"){
  i=1
  while(i<=nrow(e1))
  {
    lines(e1$X, e1$Y, col=col)
    i = i+1
  }
}

f <- function(x, y) return((-5.6e-8*6*0.5*(y^4-200^4))/(1*100))
```

```
e1 = metodoEuler(f, 10, 0, 180, 200)

e1[nrow(e1),]
```

```
##       X   Y
## 21 200 200
```

```
xx <- c(0, 200); yy <- c(0, 200)
vectorfield(f, xx, yy, scale = 1)

imprimir(e1)

sol <- rk4(f, 0, 200,180, 19)
lines(sol$x, sol$y, col="purple")
```



Punto 2.

Obtenga cinco puntos de la solucion de la ecuacion, utilizando el metodo de Taylor (los tres primeros terminos)con h=0.1 implemente en R

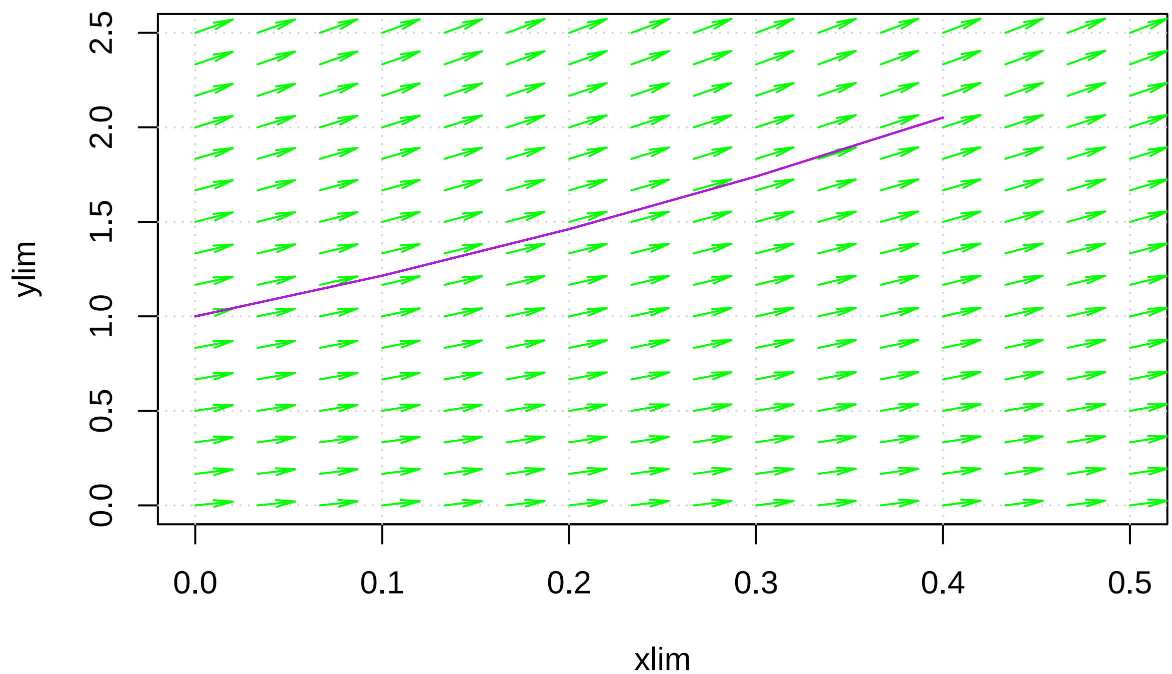
$$\frac{dy}{dx} - (x + y) = 1 - x^2;\ y\,(0) = 1$$

Grafique su solucion y compare con la solucion exacta, cual es el error de truncamiento en cada paso

```
#install.packages(Deriv)
require(Deriv) # derivadas parciales
```

```
## Loading required package: Deriv
```

```r
#--- Metodo de Taylor, orden 4
mtaylor4= function(f, t0, y0, h, n){
  #Datos igualmente espaciados iniciando en t0 = a, paso h. "n" datos
  t = seq(t0, t0 + (n-1)*h, by = h) # n datos
  y = rep(NA, times=n) # n datos
  y[1] = y0
  # Derivadas parciales con el paquete Deriv. Deriv(f)
  ft=Deriv(f,"t",nderiv = 1); fy=Deriv(f,"y",nderiv = 1)
  f1 = function(t,y)
    ft(t,y)+fy(t,y)*f(t,y)
  f1t=Deriv(f1,"t"); f1y=Deriv(f1,"y")
  f2= function(t,y) f1t(t,y)+f1y(t,y)*f(t,y)
  f2t=Deriv(f2,"t"); f2y=Deriv(f2,"y")
  f3= function(t,y) f2t(t,y)+f2y(t,y)*f(t,y) # orden m = 4
  for(i in 2:n ){
    f0i = f(t[i-1], y[i-1])
    f1i = f1(t[i-1], y[i-1])
    f2i = f2(t[i-1], y[i-1])
    f3i = f2(t[i-1], y[i-1])
    y[i] = y[i-1] + h*(f0i + h/2*f1i + h^2/6*f2i + h^3/24*f3i )
  }
  return(data.frame(X = t, Y = y))
}

imprimir<-function(e1, col="red"){
  i=1
  while(i<=nrow(e1))
  {
    lines(e1$X, e1$Y, col=col)
    i = i+1
  }
}

f = function(t,y) 1-t^2+(t+y)
t0 = 0; y0 = 1; h = 0.1; n=5
s = mtaylor4(f, t0, y0, h, n)

xx <- c(0, 0.5); yy <- c(0, 2.5)
vectorfield(f, xx, yy, scale = 0.02)
imprimir(s)
sol <- rk4(f, 0, 0.4,1, 4)

lines(sol$x, sol$y, col="purple")
```




```
errory = abs(sol$y[5]- s[5,2])

errory
```

```
## [1] 9.742566e-07
```

Punto 3.

Obtenga 20 puntos de la solucion de la ecuacion, utilizando el metodo de Euler (los tres primeros terminos)con h=0.1

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

Grafique su solucion y compare con la solucion exacta, cual es el error de truncamiento en cada paso

```
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
```

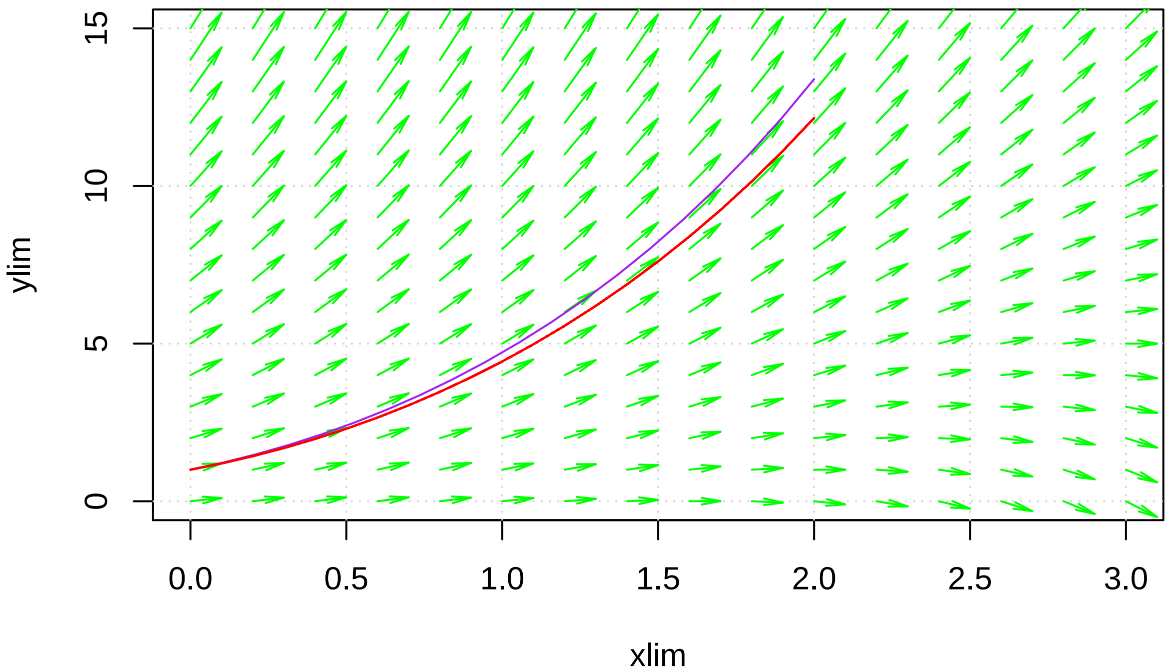
```r
    y[i+1] = y[i]+(h*f(x[i],y[i]))
    i = i+1
  }
  return (data.frame(X = x, Y = y))
}

imprimir<-function(e1, col="red"){
  i=1
  while(i<=nrow(e1))
  {
    lines(e1$X, e1$Y, col=col)
    i = i+1
  }
}

f = function(x,y) 1-x^2+(x+y)
t0 = 0; y0 = 1; h = 0.1

xx <- c(0, 3); yy <- c(0, 15)
vectorfield(f, xx, yy, scale = 0.1)
e1 = metodoEuler(f, h, t0, y0, 2)
e1
```

```
##      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
## 12 1.1  4.977805
## 13 1.2  5.564586
## 14 1.3  6.197044
## 15 1.4  6.877749
## 16 1.5  7.609523
## 17 1.6  8.395476
## 18 1.7  9.239023
## 19 1.8 10.143926
## 20 1.9 11.114318
## 21 2.0 12.154750
```

```r
e1[nrow(e1),]
```

```
##    X        Y
## 21 2 12.15475
```

```r
sol <- rk4(f, 0, 2, 1, 19)
lines(sol$x, sol$y, col="purple")
imprimir(e1)
```



Punto 4.

Implemente en R el siguiente algoritmo y apliquelo para resolver la ecuacion anterior

1. Defina f(x,y) y la condicion inicial (x0,y0)
2. Defina h y la cantidad de puntos a calcular m
3. Para i=1,2, . . . , m
4. K1 = hf(xi,yi)
5. K2 = hf(xi+h, yi+K1)
6. yi+1 = yi+ 1/2 (K1+K2)
7. xi+1 = xi + h
8. fin

```
f = function(x,y) 1-x^2+(x+y)
x0 = 0; y0 = 1; h = 0.1; m=20

g<-function(x0,y0,f ,h, m){
  i = 1
  x = numeric(m+1)
  y = numeric(m+1)
  x[1] = x0
  y[1] = y0
  while(i <= m){
    k1 = h*f(x[i], y[i])
    k2 = h*f(x[i]+h, y[i]+k1)

    y[i+1] = y[i] + ((1/2)*(k1+k2))
```

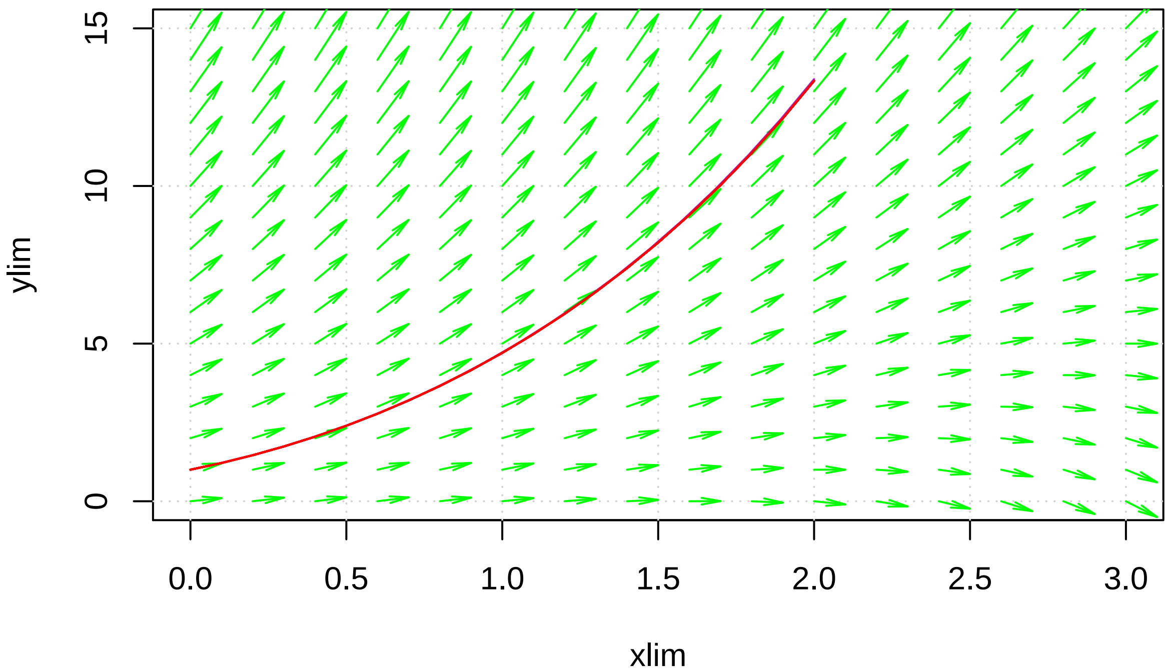
```r
    x[i+1] = x[i] + h
    i = i+1
  }
  return (data.frame(X = x, Y = y))
}

imprimir<-function(e1, col="red"){
  i=1
  while(i<=nrow(e1))
  {
    lines(e1$X, e1$Y, col=col)
    i = i+1
  }
}

e1 = g(x0,y0,f,h,m)
e1
```

```
##      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
e1[nrow(e1),]
```

```
##    X        Y
## 21 2 13.33592
```

```r
xx <- c(0, 3); yy <- c(0, 15)
vectorfield(f, xx, yy, scale = 0.1)
sol <- rk4(f, 0, 2, 1, 19)
lines(sol$x, sol$y, col="purple")
imprimir(e1)
```



Punto 5.

Utilizar la siguiente variacion en el metodo de Euler, para resolver una ecuacion diferencial ordinaria de primer orden, la cual calcula el promedio de las pendientes en cada paso

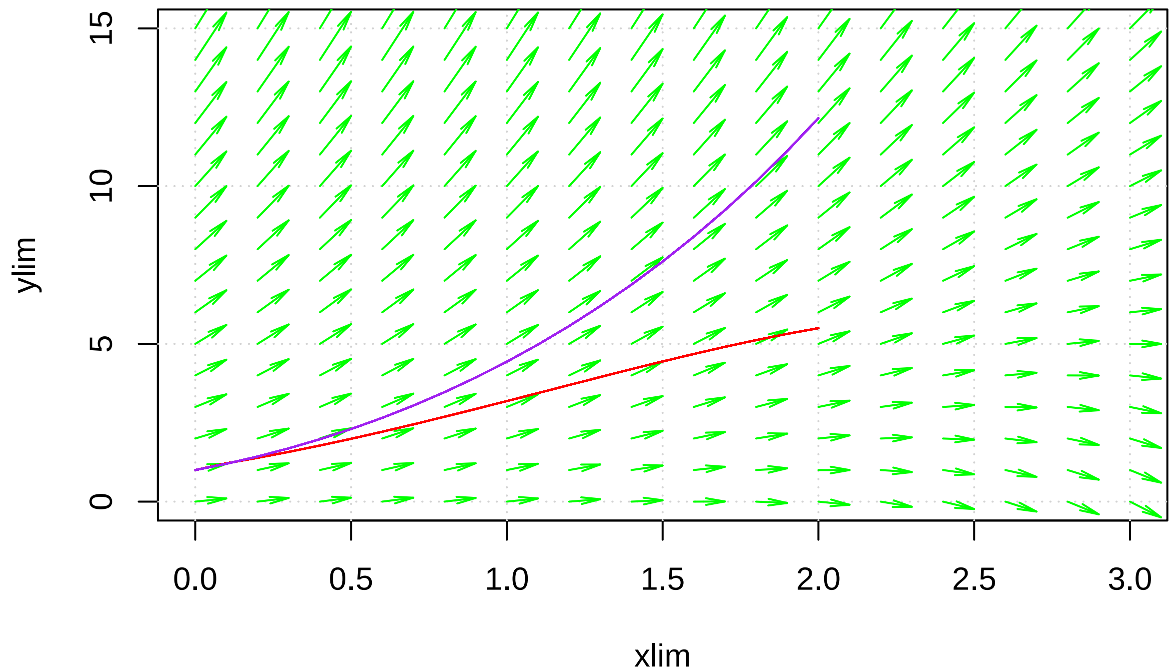
$$y_{i+1} \;=\; y_i \;+\; \frac{h}{2}\;\left(f\left(x_i\;,\;y_i\right) + f\left(x_{i+1},\;y_{i+1}\right)\right)$$

Implemente un codigo en R, para este metodo y obtenga 10 puntos de la solucion con h=0.1, grafiquela y comparela con el metodo de Euler:

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

```
library(pracma)

metodoEulerModificado <- function(f, h, xi, yi, xf)
{
  N = (xf - xi) / h
  x = y = numeric(N+1)
  x[1] = xi;
  y[1] = yi;
  y[2] = y[1]+(h*f(x[1],y[1]))
  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))
}

imprimir<-function(e1, col="red"){
  i=1
  while(i<=nrow(e1))
  {
    lines(e1$X, e1$Y, col=col)
    i = i+1
  }
}

f = function(x,y) 1-x^2+(x+y)
t0 = 0; y0 = 1; h = 0.1


e1 = metodoEulerModificado(f, h, t0, y0, 2)
xx <- c(0, 3); yy <- c(0, 15)
vectorfield(f, xx, yy, scale = 0.1)
e2 = metodoEuler(f, h, t0, y0, 2)

imprimir(e1) #eulermodificado
imprimir(e2, col="purple") #euler
```



Punto 7.

Pruebe el siguiente codigo en R del metodo de Runge Kutta de tercer y cuarto orden y obtenga 10 puntos de la solucion con h=0.1, grafiquela y comparela con el metodo de Euler:

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

```
rungekutta = function(f,t0,y0,h,n){
  t = seq(t0, t0+n*h, by=h)
  y = rep(NA, times=(n+1))

  y[1] = y0
  for(k in 2:(n+1)){
    k1=h/2*f(t[k-1],y[k-1])
    k2=h/2*f(t[k-1]+h/2, y[k-1]+k1)
    k3=h/2*f(t[k-1]+h/2, y[k-1]+k2)
    k4=h/2*f(t[k-1]+h, y[k-1]+2*k3)
    y[k] = y[k-1]+1/3*(k1+2*k2+2*k3+k4)
  }
  return(data.frame(X = t, Y = y))
}

imprimir<-function(e1, col="red"){
  i=1
  while(i<=nrow(e1))
```

```
  {
    lines(e1$X, e1$Y, col=col)
    i = i+1
  }
}


f = function(x,y) 1-x^2+(x+y)
t0 = 0; y0 = 1; h = 0.1


e1 = rungekutta(f, t0,y0,h,10)
xx <- c(0, 1.5); yy <- c(0, 6)
vectorfield(f, xx, yy, scale = 0.1)
e2 = metodoEuler(f, h, t0, y0, 1)

imprimir(e1) #rungekutta
imprimir(e2, col="purple") #euler
```