Taller EDO Análisis Numérico

Laura Donado y Jhonny Parra

27 de octubre de 2018

Punto 1

Considere un cuerpo con temperatura interna T 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:

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

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 área de la superficie y v es la tasa de transferencia dle calor. La tasa de variación de la energía $\frac{dT}{dt} = -\frac{v(t)}{mC}$ (C indica el calor específico del material que constituye el cuerpo). En consecuencia,

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

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, g = 0.5 y C = 100J/(Kg/K). Hacer una representación gráfica del resultado.

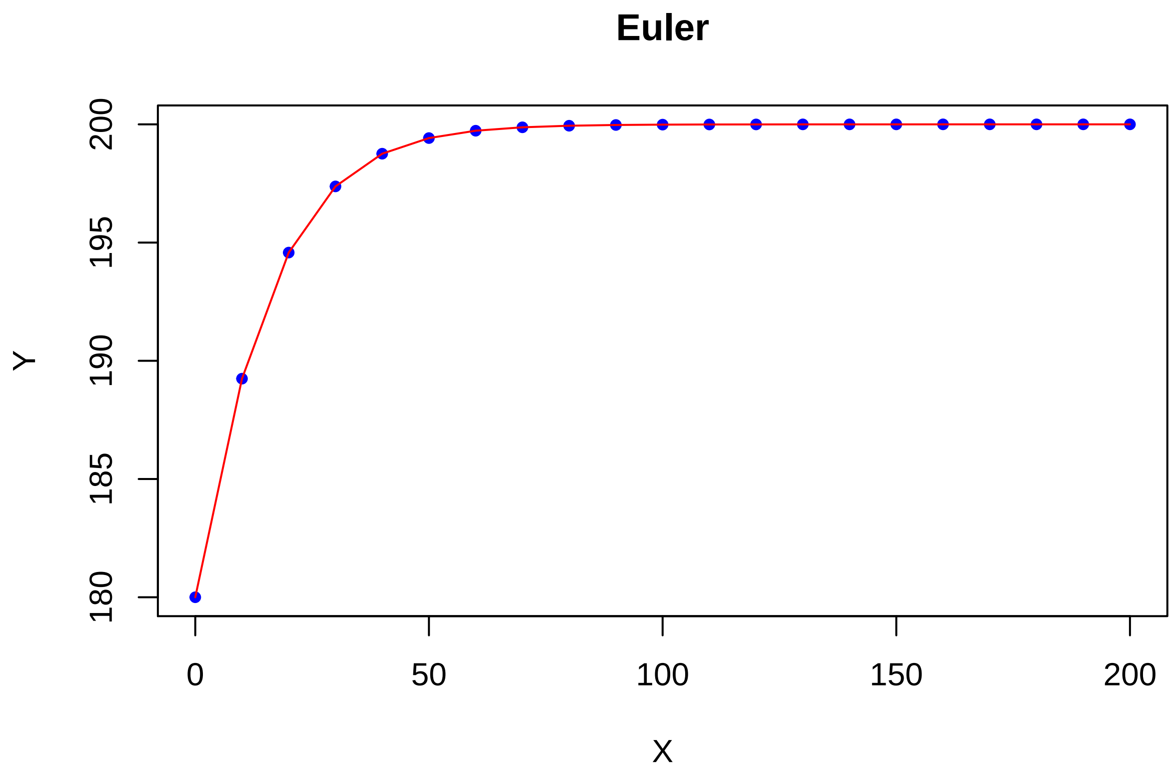
Solución:

Usando $\gamma = 0.5$

```
graficarSolucionNumerica<-function (x, y){
  plot(x,y, pch=20, col = "blue", xlab="X", ylab="Y", main="Euler")
  for (i in 2:length(x)){
    segments(x[i-1], y[i-1], x[i], y[i], col="red")
  }
}
euler <- function(f, x0, y0, h, n) {
  x <- x0
  y <- y0

  for(i in 1:n) {
    y0 <- y0 + h * f(x0, y0)
    x0 <- x0 + h
    x <- c(x, x0)
    y <- c(y, y0)
  }

  graficarSolucionNumerica(x, y)
  return(data.frame(x = x, y = y))
}
euler (function(t,T){-1.68*10^(-9) * T^4 + 2.6880 }, 0, 180, 10, 20)
```




```
##      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
```

Punto 2

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

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

Grafique su solución y compare con la solución exacta, cuál es el error de truncamiento en cada paso.

Primero, se sabe que el método de Taylor de orden n esta definido de la siguiente manera:

$$y_{i+1} = y_i + hT_n(t_i,\ y_i)\,, i\ =\ 0, 1, 2, ...N - 1$$

Donde:

$$T_1(t,\ y(t))\ = f(t, y(t))$$

$$T_n(t,\ y(t))\ =\ T_{n-1}(t,\ y(t)) + \frac{f^{(n-1)}(t, y(t))}{n!}h^{n-1}$$

De esta manera:

$$T_2(t_i,\ y_i) = f(t_i,\ y_i) + \frac{f'\ (t_i,\ y_i)}{2!}h$$

$$T_3(t_i,\ y_i) = T_2(t_i,\ y_i) + \frac{f\,''\ (t_i,\ y_i)}{3!}h^2$$

$$f(t,\ y(x)) = 1 - x^2 + x + y$$

$$f'(t,\ y(x)) = 2 - x + y - x^2$$

$$f''(t,\ y(x)) = y - x^2 - x$$

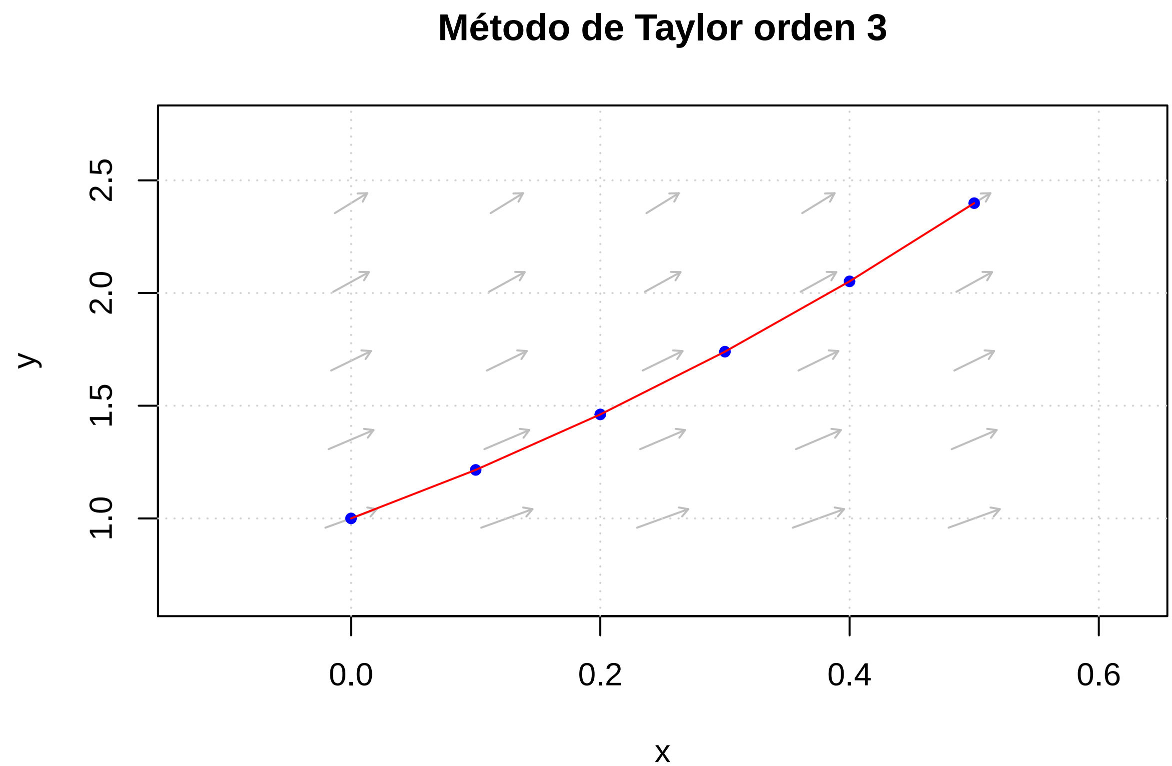
$$y_{i+1} = y_i + h\left(f(x_i,\ y_i) + \frac{h}{2}(f'(x_i,\ y_i)) + \frac{h^2}{6}(f''(x_i,\ y_i))\right)$$

Con lo anterior, se realizó el siguiente código

Solución:

```
options (digits=8)
library(phaseR)
f<-function(fcn,x,y){
  return(eval(fcn))
}
yprima<-function(x,y){
  return(1-x^2+x+y)
}
yprima1<-function(x,y){
```

```
  return(2-x+y-x^2)
}
yprima2<-function(x,y){
  return(y-x^2-x)
}
errorTruncamiento<-function(x,y){
  sol=x^2+x+exp(x)
  return(abs(y-sol))
}
taylor<-function(dy,x0, y0, h, n){
  x<-seq(from=x0, by=h, length.out = n+1)
  y<-c(y0)
  error<-c(errorTruncamiento(x0,y0))
  for(i in 2:length(x)){
    y<-c(y, y[i-1]+h*(yprima(x[i-1],y[i-1])+h/2*
                       (yprima1(x[i-1],y[i-1]))+
                       (h^2)/6*(yprima2(x[i-1],y[i-1]))))
    error<-c(error,errorTruncamiento(x[i-1],y[i-1]))
  }
  graficarCampoPendiente(min(x), max(x), min(y), max(y), dy, n, "Método de Taylor orden 3")
  graficarSolucionNumerica(x, y)
  return(data.frame(x=x, y=y, "Error de truncamiento"=error))
}
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="blue")
  for (i in 2:length(x)){
    segments(x[i-1], y[i-1], x[i], y[i], col="red")
  }
}
taylor(expression(1-x^2+(x+y)), 0, 1, 0.1, 5)
```




```
##     x         y Error.de.truncamiento
## 1 0.0 1.0000000         0.0000000e+00
## 2 0.1 1.2151667         0.0000000e+00
## 3 0.2 1.4613934         4.2514090e-06
## 4 0.3 1.7398432         9.3970491e-06
## 5 0.4 2.0518017         1.5577988e-05
## 6 0.5 2.3986896         2.2955075e-05
```

Punto 3

Obtenga 20 puntos de la solución de la ecuación, utilizando el método de Euler (los tres primeros términos) con h=0.1

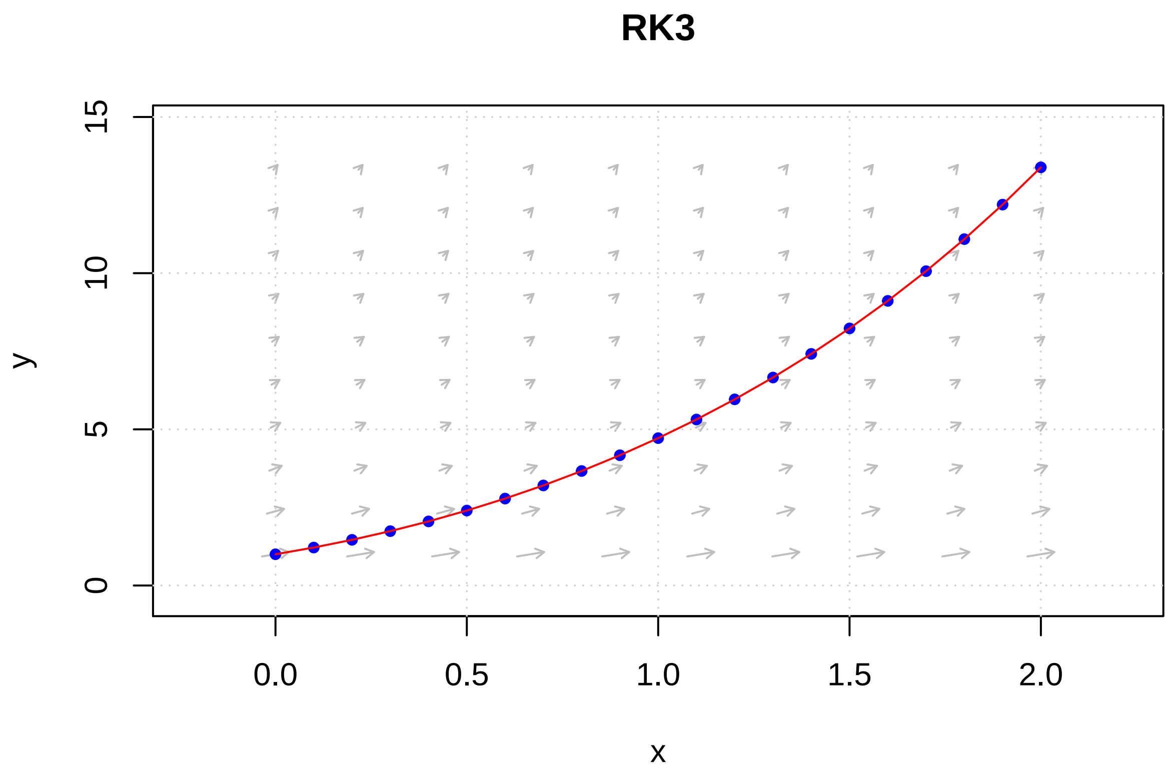
$$\frac{dy}{dx} - (x + y) = 1 - x^2;\ y\,(0)\ =\ 1$$

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

Solución:

```
options (digits=8)
library(phaseR)
f<-function(fcn,x,y){
```

```r
  return(eval(fcn))
}
errorTruncamiento<-function(x,y){
  sol=x^2+x+exp(x)
  return(abs(y-sol))
}
rk3<-function(dy, ti, tf, y0, h, graficar=TRUE, numpendientes=10){
  t<-seq(ti, tf, h)
  y<-c(y0)
  error<-c(errorTruncamiento(ti,y0))
  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))
    error<-c(error,errorTruncamiento(t[i-1],y[i-1]))
  }
  if (graficar){
    graficarCampoPendiente(min(t), max(t), min(y), max(y), dy, numpendientes, "RK3")
    graficarSolucionNumerica(t, y)
  }
  return(data.frame(x=t, y=y, "Error de truncamiento"=error))
}
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="blue")
  for (i in 2:length(x)){
    segments(x[i-1], y[i-1], x[i], y[i], col="red")
  }
}
rk3(expression(1-x^2+(x+y)), 0, 2, 1,0.1)
```




```
##       x          y Error.de.truncamiento
## 1  0.0  1.0000000          0.0000000e+00
## 2  0.1  1.2151583          0.0000000e+00
## 3  0.2  1.4613758          1.2584742e-05
## 4  0.3  1.7398155          2.6940105e-05
## 5  0.4  2.0517628          4.3299322e-05
## 6  0.5  2.3986382          6.1925102e-05
## 7  0.6  2.7820116          8.3113249e-05
## 8  0.7  3.2036182          1.0719671e-04
## 9  0.8  3.6653753          1.3455014e-04
## 10 0.9  4.1694023          1.6559495e-04
## 11 1.0  4.7180411          2.0080504e-04
## 12 1.1  5.3138801          2.4071315e-04
## 13 1.2  5.9597798          2.8591801e-04
## 14 1.3  6.6589017          3.3709232e-04
## 15 1.4  7.4147395          3.9499171e-04
## 16 1.5  8.2311546          4.6046468e-04
## 17 1.6  9.1124144          5.3446386e-04
## 18 1.7 10.0632349          6.1805847e-04
## 19 1.8 11.0888285          7.1244832e-04
## 20 1.9 12.1949553          8.1897946e-04
## 21 2.0 13.3879814          9.3916166e-04
```

Punto 4

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

1) Defina f(x,y) y la condición incial (x_0, y_0)
2) Defina h y la cantidad de puntos a calcular m
3) Para i =1, 2, ..., m
4) $K_1 = hf(x_i, y_i)$
5) $K_2 = hf(x_i + h, y_i + K_1)$
6) $y_{i+1} = y_i + \frac{1}{2}(K_1 + K_2)$.
7) $x_{i+1} = x_i + h$
8) fin

Solución:

```r
options (digits=8)
f<-function(x,y){1-x^2+(x+y)}
x<-c(0)
y<-c(1)
h<-0.1
m=20
for(i in 1:m){
  k1<-h*f(x[i],y[i])
  k2<-h*f(x[i]+h,y[i]+k1)
  y<-c(y,y[i]+(k1+k2)/2)
  x<-c(x,x[i]+h)
}
data.frame (x=x, y=y)
```

```
##      x          y
## 1  0.0  1.0000000
## 2  0.1  1.2145000
## 3  0.2  1.4599725
## 4  0.3  1.7375696
## 5  0.4  2.0485644
## 6  0.5  2.3943637
## 7  0.6  2.7765219
## 8  0.7  3.1967567
## 9  0.8  3.6569661
## 10 0.9  4.1592476
## 11 1.0  4.7059186
## 12 1.1  5.2995400
## 13 1.2  5.9429417
## 14 1.3  6.6392506
## 15 1.4  7.3919219
## 16 1.5  8.2047737
## 17 1.6  9.0820249
## 18 1.7 10.0283376
## 19 1.8 11.0488630
## 20 1.9 12.1492936
```

```
## 21 2.0 13.3359194
```

Punto 5

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

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

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:

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

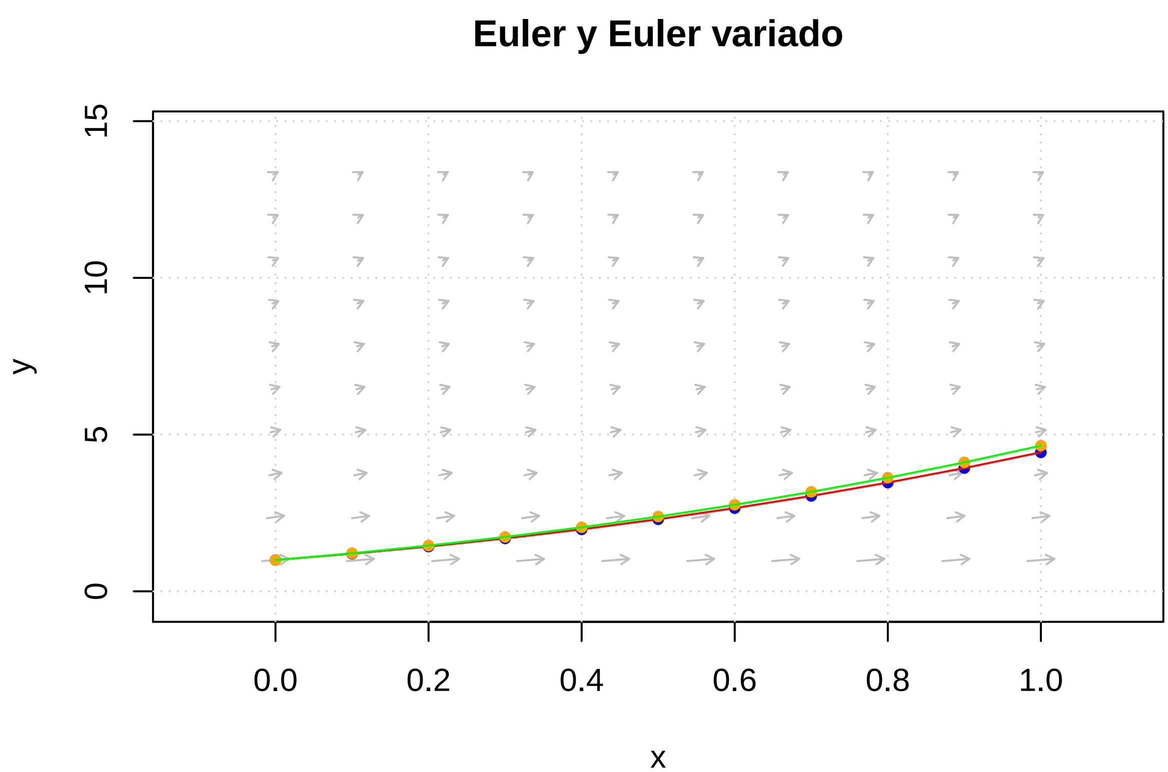
Solución:

```
options (digits=8)
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))
}
obtenerErrorAbsoluto<-function(x,y){
  solucion=exp(x)*(x^2*exp(-x) + x*exp(-x) + 1)
  return(abs(y-solucion))
}
graficarSolucionNumerica<-function (x, y, colorPuntos="blue", colorLineas="red"){
  points (x, y, pch=20, col=colorPuntos)
  for (i in 2:length(x)){
    segments(x[i-1], y[i-1], x[i], y[i], col=colorLineas)
  }
}
variacionEuler<-function(dy, ti, tf, y0, h, graficar=TRUE, numpendientes=10){
  t<-seq(ti, tf, h)
  y_euler<-c(y0)
  y_eulerv<-c(y0)
  error_euler<-c(0)
  error_eulerv<-c(0)
  for(i in 2:length(t)){
    y_euler<-c(y_euler, y_euler[i-1] + h * f(dy, t[i-1], y_euler[i-1]))

    y_eulerv<-c(y_eulerv, y_eulerv[i-1]+h*0.5*(f(dy, t[i-1],
                                                 y_eulerv[i-1]) +
                                                 f(dy, t[i], y_euler[i])))

    error_euler<-c(error_euler, obtenerErrorAbsoluto(t[i], y_euler[i]))
    error_eulerv<-c(error_eulerv, obtenerErrorAbsoluto(t[i], y_eulerv[i]))
```

```r
  }
  if (graficar){
    graficarCampoPendiente(min(t), max(t),
                           min(y), max(y), dy, numpendientes, "Euler y Euler variado")
    graficarSolucionNumerica(t, y_euler)
    graficarSolucionNumerica(t, y_eulerv, "orange", "green")
  }
  return (data.frame(x=t, y_euler=y_euler,
                     y_eulervariado=y_eulerv,
                     error_euler=error_euler,
                     error_eulervariado=error_eulerv))
}
variacionEuler(expression(x+y+1-x^2), 0, 1, 1, 0.1)
```




```
##       x   y_euler y_eulervariado error_euler error_eulervariado
## 1   0.0 1.0000000      1.0000000 0.000000000      0.00000000000
## 2   0.1 1.2000000      1.2145000 0.015170918      0.00067091808
## 3   0.2 1.4290000      1.4591750 0.032402758      0.00222775816
## 4   0.3 1.6879000      1.7350288 0.051958808      0.00483005758
## 5   0.4 1.9776900      2.0431647 0.074134698      0.00866001014
## 6   0.5 2.2994590      2.3847959 0.099262271      0.01392539883
## 7   0.6 2.6544049      2.7612559 0.127713900      0.02086288992
## 8   0.7 3.0438454      3.1740110 0.159907317      0.02974173198
## 9   0.8 3.4692299      3.6246730 0.196310999      0.04086790778
## 10  0.9 3.9321529      4.1150143 0.237450189      0.05458879331
## 11  1.0 4.4343682      4.6469834 0.283913614      0.07129838401
```

La variación al método de Euler produce una reducción en el error de truncamiento con respecto al método de Euler original.

Punto 7

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:

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

Solución:

```r
options (digits=7)
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))
}
obtenerErrorAbsoluto<-function(x,y){
  solucion=exp(x)*(x^2*exp(-x) + x*exp(-x) + 1)
  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, colorPuntos="blue", colorLineas="red"){
  points (x, y, pch=20, col=colorPuntos)
  for (i in 2:length(x)){
    segments(x[i-1], y[i-1], x[i], y[i], col=colorLineas)
  }
}
rk3_4yeuler<-function(dy, ti, tf, y0, h, graficar=TRUE, numpendientes=10){
  t<-seq(ti, tf, h)
  y_rk3<-c(y0)
  y_rk4<-c(y0)
  y_euler<-c(y0)

  error_euler<-c(0)
  error_rk3<-c(0)
  error_rk4<-c(0)
```

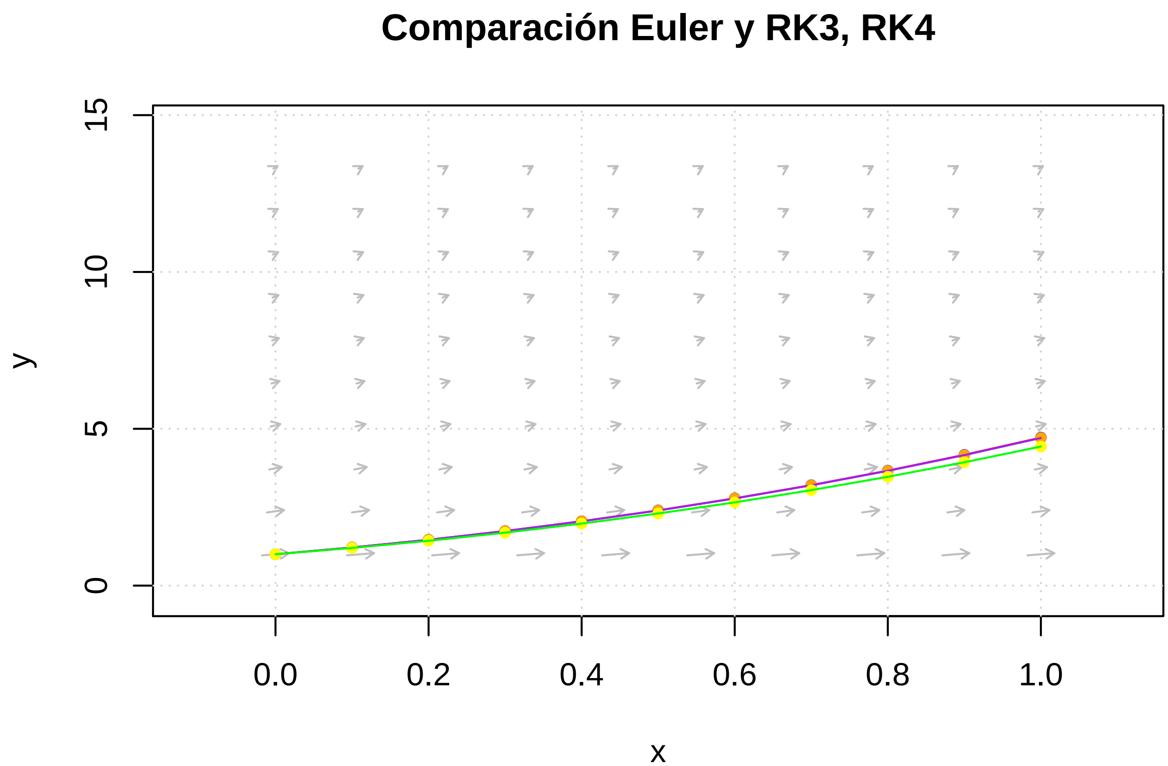
```r
  for(i in 2:length(t)){
    y_euler<-c(y_euler, y_euler[i-1] + h * f(dy, t[i-1], y_euler[i-1]))

    k1=h*f(dy, t[i-1], y_rk3[i-1])
    k2=h*f(dy, t[i-1]+h/2, y_rk3[i-1]+k1*(0.5))
    k3=h*f(dy, t[i-1]+h, y_rk3[i-1]-k1+2*k2)
    y_rk3<-c(y_rk3, y[i-1]+1/6*(k1+4*k2+k3))


    k1=h*f(dy, t[i-1], y_rk4[i-1])
    k2=h*f(dy, t[i-1]+h/2, y_rk4[i-1]+k1*(0.5))
    k3=h*f(dy, t[i-1]+h/2, y_rk4[i-1]+k2*(0.5))
    k4=h*f(dy, t[i-1]+h, y_rk4[i-1]+k3)
    y_rk4<-c(y_rk4, y_rk4[i-1]+1/6*(k1+2*k2+2*k3+k4))



    error_euler<-c(error_euler, obtenerErrorAbsoluto(t[i], y_euler[i]))
    error_rk3<-c(error_rk3, obtenerErrorAbsoluto(t[i], y_rk3[i]))
    error_rk4<-c(error_rk4, obtenerErrorAbsoluto(t[i], y_rk4[i]))
  }
  if (graficar){
    graficarCampoPendiente(min(t), max(t),
                           min(y), max(y), dy,
                           numpendientes, "Comparación Euler y RK3, RK4")
    graficarSolucionNumerica(t, y_rk4)
    graficarSolucionNumerica(t, y_rk3, "orange", "purple")
    graficarSolucionNumerica(t, y_euler, "yellow", "green")
  }
  return(data.frame(x=t, y_euler=y_euler,
                    y_RK3=y_rk3, y_RK4=y_rk4,
                    error_Euler=error_euler, error_RK3=error_rk3,
                    error_RK4=error_rk4))
}
rk3_4yeuler(expression(x+y+1-x^2), 0, 1, 1, 0.1)
```




```
##       x  y_euler     y_RK3     y_RK4 error_Euler    error_RK3    error_RK4
## 1  0.0 1.000000 1.000000 1.000000  0.00000000 0.000000e+00 0.000000e+00
## 2  0.1 1.200000 1.215158 1.215171  0.01517092 1.258474e-05 2.930756e-07
## 3  0.2 1.429000 1.460717 1.461402  0.03240276 6.852734e-04 6.258867e-07
## 4  0.3 1.687900 1.738343 1.739858  0.05195881 1.515852e-03 1.003550e-06
## 5  0.4 1.977690 2.049362 2.051823  0.07413470 2.462684e-03 1.431817e-06
## 6  0.5 2.299459 2.395187 2.398719  0.09926227 3.533944e-03 1.917157e-06
## 7  0.6 2.654405 2.777374 2.782116  0.12771390 4.744580e-03 2.466835e-06
## 8  0.7 3.043845 3.197641 3.203750  0.15990732 6.111979e-03 3.089018e-06
## 9  0.8 3.469230 3.657885 3.665537  0.19631100 7.655709e-03 3.792876e-06
## 10 0.9 3.932153 4.160205 4.169599  0.23745019 9.397729e-03 4.588707e-06
## 11 1.0 4.434368 4.706919 4.718276  0.28391361 1.136267e-02 5.488071e-06
```

Los errores se ordenan de mayor a menor, teniendo Euler el mayor error de truncamiento y RK4 el menor error de truncamiento con respecto a la solución analítica.