Taller No. 2

Kevin Pelaez - Juan Diego Osorio

28 de octubre de 2018

##   
## Attaching package: 'PolynomF'

## The following object is masked from 'package:pracma':  
##   
## integral

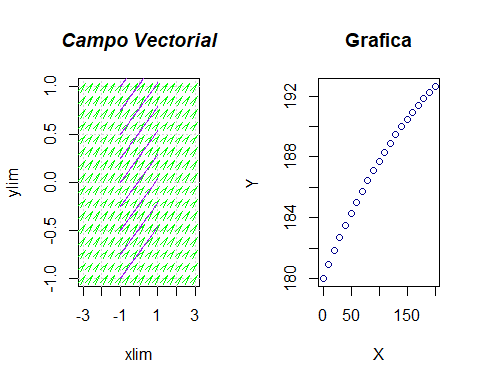
## PUNTO #1

1. Considere un cuerpo con temperatura interna **T**𝑇 el cual se encuentra en un ambiente con temperatura constante**Te**. Suponga que su masa **m**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:

Donde, **t** es tiempo y **ε** es la constante de Boltzmann , **γ** es la constante de “Emisividad” del cuerpo, **S** el área de superficie y **v** es la tasa de transferencia de calor. La tasa de variacion de la energía (C indica el calor específico del material que constituye el cuerpo). En consecuencia,

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.

## X Y  
## 21 200 192.6369

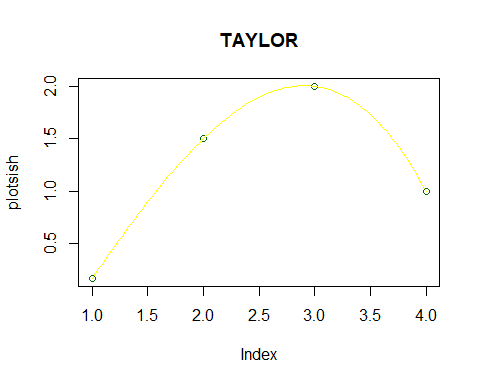


## PUNTO #2

1. 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

funcion4punto <- function(x){  
 exp(x) \* (x^2\* exp(-x) + x\* exp(-x) + 1)  
}  
  
num <- seq(1:4)  
plotsish <- c(taylor(funcion4punto, x0=0, n = 3))  
  
plot(plotsish, col = "darkgreen", main = "TAYLOR")  
  
poliajuste <- poly.calc(num,plotsish)  
curve(poliajuste,add = TRUE, col = "yellow")



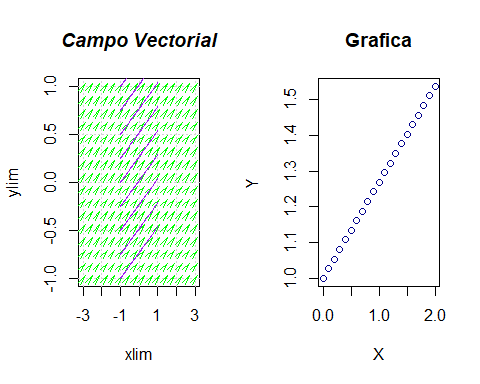
## PUNTO #3

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

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))  
}  
  
funcion3punto <- function(x,y) { exp(x) \* (x^2\* exp(-x) + x\* exp(-x) + 1)}  
  
e1 = metodoEuler(f, 0.1, 0, 1, 2)  
  
e1[nrow(e1),]

## X Y  
## 21 2 1.5376

par(mfrow = c(1,2))  
  
xx <- c(-3, 3); yy <- c(-1, 1)  
vectorfield(f, xx, yy, scale = 0.1)  
for (xs in seq(-1, 1, by = 0.25))   
{  
 sol <- rk4(f, -1, 1, xs, 100)  
 lines(sol$x, sol$y, col="purple")  
}  
title(main="Campo Vectorial", col.main="black", font.main=4)  
  
plot(e1, col = "darkblue", main = "Grafica")



## PUNTO #4

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

1. Defina h y la cantidad de puntos a calcular m
2. Defina f(x,y) y la condicion iniccial (x0,y0)
3. para i =12, …, m
4. K1= hf(xi, yi)
5. k2 = hf(xi + h, yi +h)
6. yi+1 = yi + 1/2 (k1 + k2)
7. xi+1 = xi + h
8. fin

funcion4punto <- function(x,y){  
 return (exp(x) \* (x^2\* exp(-x) + x\* exp(-x) + 1))  
}  
  
m <- 5  
   
h <- 0.1  
 x0 <- 1  
 y0 <- 0  
for (i in 1:m){  
 k1 <- h \* funcion4punto(x0, y0)  
 k2 <- h \* funcion4punto(x0 + h, y0+h)  
 y0 <- y0 + 0.5 \* (k1 + k2)  
 x0 <- x0 + h  
 }

## 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 + h/2 f((x\_i,y\_i) + f(x\_{i+1}, y\_{i+1}) )

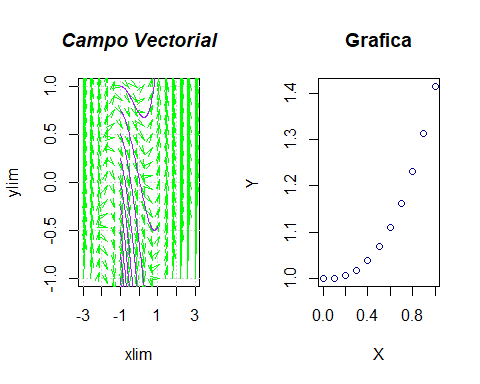
$$

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:

variacionMetodoEuler <- 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]))  
 i = i+1  
 }  
 return (data.frame(X = x, Y = y))  
}  
f <- function(x,y) {x+y-1+x^2}  
  
e1 = variacionMetodoEuler(f, 0.1, 0, 1, 1)  
  
e1[nrow(e1),]

## X Y  
## 11 1 1.414725

par(mfrow = c(1,2))  
  
xx <- c(-3, 3); yy <- c(-1, 1)  
vectorfield(f, xx, yy, scale = 0.1)  
for (xs in seq(-1, 1, by = 0.25))   
{  
 sol <- rk4(f, -1, 1, xs, 100)  
 lines(sol$x, sol$y, col="purple")  
}  
title(main="Campo Vectorial", col.main="black", font.main=4)  
  
plot(e1, col = "darkblue", main = "Grafica")

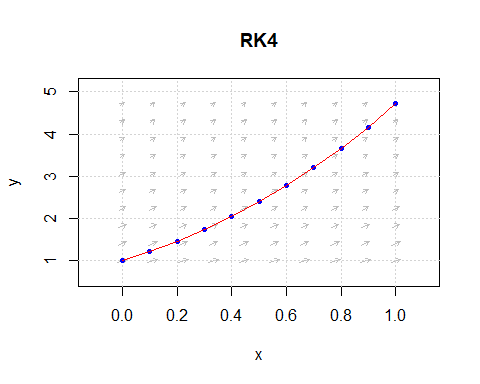


## PUNTO #7

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:

f<-function(fcn,x,y){  
 return(eval(fcn))  
}  
  
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="blue")  
 for (i in 2:length(x)){  
 segments(x[i-1], y[i-1], x[i], y[i], col="red")  
 }  
}  
  
Rrk4<-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, "RK4")  
 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")  
 }  
 if (graficar){  
 graficarCampoPendiente(min(t), max(t), min(y), max(y), dy, numpendientes, "RK3")  
 graficarSolucionNumerica(t, y)  
 }  
 rta<-list(w=y, t=t)  
}  
  
r<-Rrk4(expression(x+y+1-x^2), 0, 1, 1, 0.1)

## x |y |k1 |k2 |k3 |k4 |error absoluto  
## 0 | 1 | 0.2 | 0.21475 | 0.2154875 | 0.2305488 | 0   
## 0.1 | 1.215171 | 0.2305171 | 0.2457929 | 0.2465567 | 0.2621727 | 0.1048288   
## 0.2 | 1.461402 | 0.2621402 | 0.2779972 | 0.2787901 | 0.2950192 | 0.2185966   
## 0.3 | 1.739858 | 0.2949858 | 0.3114851 | 0.31231 | 0.3292168 | 0.3401402   
## 0.4 | 2.051823 | 0.3291823 | 0.3463914 | 0.3472519 | 0.3649075 | 0.4681739   
## 0.5 | 2.398719 | 0.3648719 | 0.3828655 | 0.3837652 | 0.4022485 | 0.6012768   
## 0.6 | 2.782116 | 0.4022116 | 0.4210722 | 0.4220152 | 0.4414132 | 0.7378787   
## 0.7 | 3.20375 | 0.441375 | 0.4611937 | 0.4621846 | 0.4825934 | 0.8762442   
## 0.8 | 3.665537 | 0.4825537 | 0.5034314 | 0.5044753 | 0.5260012 | 1.014455   
## 0.9 | 4.169599 | 0.5259599 | 0.5480078 | 0.5491102 | 0.5718709 | 1.150392



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

## x |y |k1 |k2 |k3 |error absoluto  
## 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

