Ejercicios SENOL new

Agustin Huczok

30/9/2021

#a

##Norma

```
norma <- function(y, metodo){
  if (metodo==2){
    return(sqrt(sum(y^2)))
  }
  if (metodo==Inf){
    return(max(abs(y)))
  }
  return("El metodo debe ser 2 o Inf")
}
```

##Sistema Ec. No lineal Newton

```
Sist_Ec_NoLineal_Newton <- function(n,x,TOL,N){
  #Paso 1
  k <- 1
  #Paso 2
  while(k<=N){
    #Paso 3
    fx <- Fx(x)
    J <- Jacobiano(x[1],x[2])
    #Paso 4
    y = solve(J)%*%-fx
    #Paso 5
    x <- x + t(y)
    #Paso 6
    if (norma(y,2) < TOL){
      return(x)
    }
    #Paso 7
    k <- k+1
  }
  #Paso 8
  return(paste('Numero max de iteraciones excedido'))
}
```

##Calculo derivadas dos variables

```r
fa=function(x1,x2){
  3*x1^2-x2^2
}

fae=expression(3*x1^2-x2^2)
D(fae,"x1")
```

```
## 3 * (2 * x1)
```

```r
D(fae,"x2")
```

```
## -(2 * x2)
```

```r
dfa1=function(x1,x2){3 * (2 * x1)}
dfa2=function(x1,x2){-(2 * x2)}


fb=function(x1,x2){
  3*x1*x2^2-x1^3
}
fbe=expression(3*x1*x2^2-x1^3)
D(fbe,"x1")
```

```
## 3 * x2^2 - 3 * x1^2
```

```r
D(fbe,"x2")
```

```
## 3 * x1 * (2 * x2)
```

```r
dfb1=function(x1,x2,x3){3 * x2^2 - 3 * x1^2}
dfb2=function(x1,x2,x3){3 * x1 * (2 * x2)}
```

##Matriz jacobiana

```r
Jacobiano <- function(x1,x2){
  col1 <-
    c(dfa1(x1,x2),dfa2(x1,x2))

  col2 <-
    c(dfb1(x1,x2),dfb2(x1,x2))


  J <- rbind(col1,col2) #con esta ultima armamos la matrix ampliada
  return(J)
}
```

##Definino Fx

```
Fx <- function(x){
  Fx <- rbind(fa(x[1],x[2]), fb(x[1],x[2]))
  return(Fx)
} #sera una matriz ampliada con las funciones definadas antes
```

Evaluo fn y el Jacobiano

```
x <- c(0.1,0.1)
n=2
Resultado=Sist_Ec_NoLineal_Newton(n, x, 10^-10, 100)
Resultado
```

```
##              [,1]         [,2]
## [1,] 9.658469e-11 1.672896e-10
```

Corroboro

```
##Asigno los rdos del algoritmo a las variables x1,x2
x1 <- Sist_Ec_NoLineal_Newton(n,x, 10^-5, 100)[1] #posicion, osea mult por posicion 1
x2 <- Sist_Ec_NoLineal_Newton(n,x, 10^-5, 100)[2]
```

Resultados

```
fa(x1, x2)
```

```
## [1] 1.039654e-16
```

```
fb(x1, x2)
```

```
## [1] 4.461497e-15
```

#b—- ##Norma

```
norma <- function(y, metodo){
  if (metodo==2){
    return(sqrt(sum(y^2)))
  }
  if (metodo==Inf){
    return(max(abs(y)))
  }
  return("El metodo debe ser 2 o Inf")
}
```

##Sistema Ec. No lineal Newton

```
Sist_Ec_NoLineal_Newton <- function(n,x,TOL,N){
  #Paso 1
  k <- 1
  #Paso 2
  while(k<=N){
    #Paso 3
    fx <- Fx(x)
    J <- Jacobiano(x[1],x[2],x[3])
    #Paso 4
    y = solve(J)%*%-fx
    #Paso 5
    x <- x + t(y)
    #Paso 6
    if (norma(y,2) < TOL){
      return(x)
    }
    #Paso 7
    k <- k+1
  }
  #Paso 8
  return(paste('Numero max de iteraciones excedido'))
}
```

##Calculo derivadas tres variables

```
fa=function(x1,x2,x3){
  3*x1-cos(x2*x3)-0.5
}
fae=expression(3*x1-cos(x2*x3)-0.5)
D(fae,"x1")
```

```
## [1] 3
```

```
D(fae,"x2")
```

```
## sin(x2 * x3) * x3
```

```
D(fae,"x3")
```

```
## sin(x2 * x3) * x2
```

```
dfa1=function(x1,x2,x3){3}
dfa2=function(x1,x2,x3){sin(x2 * x3) * x3}
dfa3=function(x1,x2,x3){sin(x2 * x3) * x2}

fb=function(x1,x2,x3){
  4*x1^2-625*x2^2+2*x2-1
}


fbe=expression(4*x1^2-625*x2^2+2*x2-1)
D(fbe,"x1")
```

```
## 4 * (2 * x1)
```

```
D(fbe,"x2")
```

```
## 2 - 625 * (2 * x2)
```

```
D(fbe,"x3")
```

```
## [1] 0
```

```
dfb1=function(x1,x2,x3){4 * (2 * x1)}
dfb2=function(x1,x2,x3){2 - 625 * (2 * x2)}
dfb3=function(x1,x2,x3){0}



fc=function(x1,x2,x3){
  20*x3+((10*pi-3)/3)+exp(-x1*x2)
}


fce=expression(20*x3+((10*pi-3)/3)+exp(-x1*x2))
D(fce,"x1")
```

```
## -(exp(-x1 * x2) * x2)
```

```
D(fce,"x2")
```

```
## -(exp(-x1 * x2) * x1)
```

```
D(fce,"x3")
```

```
## [1] 20
```

```
dfc1=function(x1,x2,x3){-(exp(-x1 * x2) * x2)}
dfc2=function(x1,x2,x3){-(exp(-x1 * x2) * x1)}
dfc3=function(x1,x2,x3){20}
```

##Matriz Jacobiana

```
Jacobiano <- function(x1,x2,x3){
  col1 <-
  c(dfa1(x1,x2,x3),dfa2(x1,x2,x3),dfa3(x1,x2,x3))

  col2 <-
  c(dfb1(x1,x2,x3),dfb2(x1,x2,x3),dfb3(x1,x2,x3))

  col3 <-
  c(dfc1(x1,x2,x3),dfc2(x1,x2,x3),dfc3(x1,x2,x3))

  J <- rbind(col1,col2,col3) #con esta ultima armamos la matrix ampliada
  return(J)
}
```

##Defino Fx

```
Fx <- function(x){
  Fx <- rbind(fa(x[1],x[2],x[3]), fb(x[1],x[2],x[3]), fc(x[1],x[2],x[3]))
  return(Fx)
} #sera una matriz ampliada con las funciones definadas antes
```

##Defino los puntos

```
x <- c(0.1, 0.1, -0.1)
n=3
Resultado <- Sist_Ec_NoLineal_Newton(n, x, 10^-16, 1000)
Resultado
```

```
##            [,1]        [,2]       [,3]
## [1,] 0.4999995 0.003199065 -0.5235189
```

##Corroboro

```
x1 <- Sist_Ec_NoLineal_Newton(n,x, 10^-1, 100)[1]
x2 <- Sist_Ec_NoLineal_Newton(n,x, 10^-1, 100)[2]
x3 <- Sist_Ec_NoLineal_Newton(n,x, 10^-1, 100)[3]
```

##Imprimo los resultados

```
fa(x1, x2, x3)
```

```
## [1] 6.434626e-05
```

```
fb(x1, x2, x3)
```

```
## [1] -0.3018302
```

```
fc(x1, x2, x3)
```

```
## [1] 6.205418e-05
```

#c ##Norma

```
norma <- function(y, metodo){
  if (metodo==2){
    return(sqrt(sum(y^2)))
  }
  if (metodo==Inf){
    return(max(abs(y)))
  }
  return("El metodo debe ser 2 o Inf")
}
```

##Sistema Ec. No lineal Newton

```
Sist_Ec_NoLineal_Newton <- function(n,x,TOL,N){
  #Paso 1
  k <- 1
  #Paso 2
  while(k<=N){
    #Paso 3
    fx <- Fx(x)
    J <- Jacobiano(x[1],x[2],x[3])
    #Paso 4
    y = solve(J)%*%-fx
    #Paso 5
    x <- x + t(y)
    #Paso 6
    if (norma(y,2) < TOL){
      return(x)
    }
    #Paso 7
    k <- k+1
  }
  #Paso 8
  return(paste('Numero max de iteraciones excedido'))
}
```

##Calculo derivadas tres variables

```
fa=function(x1,x2,x3){
  x1^3+x1^2*x2-x1*x3+6
}
fae=expression(x1^3+x1^2*x2-x1*x3+6)
D(fae,"x1")
```

```
## 3 * x1^2 + 2 * x1 * x2 - x3
```

```
D(fae,"x2")
```

```
## x1^2
```

```
D(fae,"x3")
```

```
## -x1
```

```
dfa1=function(x1,x2,x3){3 * x1^2 + 2 * x1 * x2 - x3}
dfa2=function(x1,x2,x3){x1^2}
dfa3=function(x1,x2,x3){-x1}

fb=function(x1,x2,x3){
  exp(x1)+exp(x2)-x3
}


fbe=expression(exp(x1)+exp(x2)-x3)
D(fbe,"x1")
```

```
## exp(x1)
```

```
D(fbe,"x2")
```

```
## exp(x2)
```

```
D(fbe,"x3")
```

```
## -1
```

```
dfb1=function(x1,x2,x3){exp(x1)}
dfb2=function(x1,x2,x3){exp(x2)}
dfb3=function(x1,x2,x3){-1}



fc=function(x1,x2,x3){
  x2^2-2*x1*x3
}


fce=expression(x2^2-2*x1*x3)
D(fce,"x1")
```

```
## -(2 * x3)
```

```
D(fce,"x2")
```

```
## 2 * x2
```

```
D(fce,"x3")
```

```
## -(2 * x1)
```

```
dfc1=function(x1,x2,x3){-(2 * x3)}
dfc2=function(x1,x2,x3){2 * x2}
dfc3=function(x1,x2,x3){-(2 * x1)}
```

##Matriz Jacobiana

```
Jacobiano <- function(x1,x2,x3){
  col1 <- c(dfa1(x1,x2,x3),dfa2(x1,x2,x3),dfa3(x1,x2,x3))
  col2 <- c(dfb1(x1,x2,x3),dfb2(x1,x2,x3),dfb3(x1,x2,x3))
  col3 <- c(dfc1(x1,x2,x3),dfc2(x1,x2,x3),dfc3(x1,x2,x3))
  J <- rbind(col1,col2, col3)
  return(J)
}
```

##Defino Fx

```
Fx <- function(x){
  Fx <- rbind(fa(x[1],x[2],x[3]), fb(x[1],x[2],x[3]), fc(x[1],x[2],x[3]))
  return(Fx)
} #sera una matriz ampliada con las funciones definadas antes
```

##Defino los puntos

```
x <- c(0.1, 0.1, -0.1)
n=3
Resultado <- Sist_Ec_NoLineal_Newton(n, x, 10^-10, 1000)
Resultado
```

```
##           [,1]      [,2]     [,3]
## [1,] 1.167123 -2.765193 3.275701
```

##Corroboro

```
x1 <- Sist_Ec_NoLineal_Newton(n,x, 10^-6, 100)[1] #posicion, osea mult por posicion 1
x2 <- Sist_Ec_NoLineal_Newton(n,x, 10^-6, 100)[2]
x3 <- Sist_Ec_NoLineal_Newton(n,x, 10^-6, 100)[3]
```

##Imprimo los resultados

```
fa(x1, x2, x3)
```

```
## [1] 0
```

```
fb(x1, x2, x3)
```

```
## [1] 4.440892e-16
```

```
  fc(x1, x2, x3)
```

```
## [1] 0
```

#d

##Norma

```
norma <- function(y, metodo){
  if (metodo==2){
    return(sqrt(sum(y^2)))
  }
  if (metodo==Inf){
    return(max(abs(y)))
  }
  return("El metodo debe ser 2 o Inf")
}
```

##Sistema Ec. No lineal Newton

```
Sist_Ec_NoLineal_Newton <- function(n,x,TOL,N){
  #Paso 1
  k <- 1
  #Paso 2
  while(k<=N){
    #Paso 3
    fx <- Fx(x)
    J <- Jacobiano(x[1],x[2])
    #Paso 4
    y = solve(J)%*%-fx
    #Paso 5
    x <- x + t(y)
    #Paso 6
    if (norma(y,2) < TOL){
      return(x)
    }
    #Paso 7
    k <- k+1
  }
  #Paso 8
  return(paste('Numero max de iteraciones excedido'))
}
```

##Calculo derivadas dos variables

```
fa=function(x1,x2){
  5*x1^2-x2^2
}

fae=expression(5*x1^2-x2^2)
D(fae,"x1")
```

```
## 5 * (2 * x1)
```

```
D(fae,"x2")
```

```
## -(2 * x2)
```

```
dfa1=function(x1,x2){5 * (2 * x1)}
dfa2=function(x1,x2){-(2 * x2)}


fb=function(x1,x2){
  x2-0.25*(sin(x1)+cos(x2))
}
fbe=expression(x2-0.25*(sin(x1)+cos(x2)))
D(fbe,"x1")
```

```
## -(0.25 * cos(x1))
```

```r
D(fbe,"x2")
```

```
## 1 + 0.25 * sin(x2)
```

```r
dfb1=function(x1,x2,x3){-(0.25 * cos(x1))}
dfb2=function(x1,x2,x3){1 + 0.25 * sin(x2)}
```

##Matriz jacobiana

```r
Jacobiano <- function(x1,x2){
  col1 <-
    c(dfa1(x1,x2),dfa2(x1,x2))

  col2 <-
    c(dfb1(x1,x2),dfb2(x1,x2))


  J <- rbind(col1,col2) #armo la matriz ampliada
  return(J)
}
```

##Definino Fx

```r
Fx <- function(x){
  Fx <- rbind(fa(x[1],x[2]), fb(x[1],x[2]))
  return(Fx)
} #sera una matriz ampliada con las funciones definadas antes
```

Evaluo fn y el Jacobiano

```r
x <- c(0.1,0.1)
n=2
Sist_Ec_NoLineal_Newton(n, x, 10^-6, 100)
```

```
##           [,1]      [,2]
## [1,] 0.1212419 0.2711052
```

Corroboro

```r
#Asigno los rdos del algoritmo a las variables x1,x2
x1 <- Sist_Ec_NoLineal_Newton(n,x, 10^-5, 100)[1] #posicion, osea mult por posicion 1
x2 <- Sist_Ec_NoLineal_Newton(n,x, 10^-5, 100)[2]
```

##Resultados

```r
fa(x1, x2)
```

```
## [1] 9.145462e-15
```

```r
fb(x1, x2)
```

```
## [1] 5.551115e-17
```

#e ##Norma

```r
norma <- function(y, metodo){
  if (metodo==2){
    return(sqrt(sum(y^2)))
  }
  if (metodo==Inf){
    return(max(abs(y)))
  }
  return("El metodo debe ser 2 o Inf")
}
```

##Sistema Ec. No lineal Newton

```r
Sist_Ec_NoLineal_Newton <- function(n,x,TOL,N){
  #Paso 1
  k <- 1
  #Paso 2
  while(k<=N){
    #Paso 3
    fx <- Fx(x)
    J <- Jacobiano(x[1],x[2])
    #Paso 4
    y = solve(J)%*%-fx
    #Paso 5
    x <- x + t(y)
    #Paso 6
    if (norma(y,2) < TOL){
      return(x)
    }
    #Paso 7
    k <- k+1
  }
  #Paso 8
  return(paste('Numero max de iteraciones excedido'))
}
```

##Calculo derivadas dos variables

```r
fa=function(x1,x2){
  log(x1^2+x2^2)-sin(x1*x2)-log(2)
}

fae=expression(log(x1^2+x2^2)-sin(x1*x2)-log(2))
D(fae,"x1")
```

```
## 2 * x1/(x1^2 + x2^2) - cos(x1 * x2) * x2
```

```
D(fae,"x2")
```

```
## 2 * x2/(x1^2 + x2^2) - cos(x1 * x2) * x1
```

```
dfa1=function(x1,x2){2 * x1/(x1^2 + x2^2) - cos(x1 * x2) * x2}
dfa2=function(x1,x2){2 * x2/(x1^2 + x2^2) - cos(x1 * x2) * x1}


fb=function(x1,x2){
  exp(x1-x2)+cos(x1*x2)
}
fbe=expression(exp(x1-x2)+cos(x1*x2))
D(fbe,"x1")
```

```
## exp(x1 - x2) - sin(x1 * x2) * x2
```

```
D(fbe,"x2")
```

```
## -(sin(x1 * x2) * x1 + exp(x1 - x2))
```

```
dfb1=function(x1,x2,x3){exp(x1 - x2) - sin(x1 * x2) * x2}
dfb2=function(x1,x2,x3){-(sin(x1 * x2) * x1 + exp(x1 - x2))}
```

##Matriz jacobiana

```
Jacobiano <- function(x1,x2){
  col1 <-
    c(dfa1(x1,x2),dfa2(x1,x2))

  col2 <-
    c(dfb1(x1,x2),dfb2(x1,x2))


  J <- rbind(col1,col2) #armo la matriz ampliada
  return(J)
}
```

##Definino Fx

```
Fx <- function(x){
  Fx <- rbind(fa(x[1],x[2]), fb(x[1],x[2]))
  return(Fx)
} #sera una matriz ampliada con las funciones definadas antes
```

Evaluo fn y el Jacobiano

```
x <- c(0.1,0.1)
n=2
Sist_Ec_NoLineal_Newton(n, x, 10^-6, 100)
```

```
##           [,1]       [,2]
## [1,] -2.093885 -0.8967253
```

Corroboro

```
#Asigno los rdos del algoritmo a las variables x1,x2
x1 <- Sist_Ec_NoLineal_Newton(n,x, 10^-5, 100)[1] #posicion, osea mult por posicion 1
x2 <- Sist_Ec_NoLineal_Newton(n,x, 10^-5, 100)[2]
```

##Resultados

```
fa(x1, x2)
```

```
## [1] -1.110223e-16
```

```
fb(x1, x2)
```

```
## [1] -5.551115e-17
```