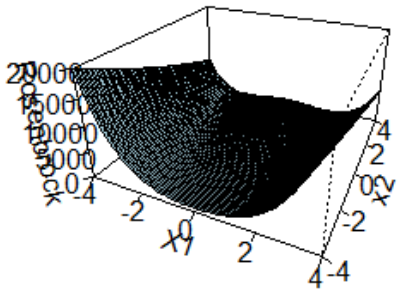
Deber function Rosenbrock

Fausto Fabian Crespo Fernandez

Gráfico de la function



Codigo:

```
x1=seq(-4, 4, 0.1)
x2=seq(-4, 4, 0.1)
z <- outer(x1, x2, function(x1,x2) (1/2)*(100*(x2-(x1)^2)^2 +(1-x1)^2))
persp(x1, x2, z, theta = 25, phi = 27,expand = 0.5, col = "lightblue",
        ltheta = 120, shade = 0.75, ticktype = "detailed",
        xlab = "X1", ylab = "x2", zlab = "Rosenbrock") -> res
round(res, 3)
```

Aplicación método CG a la función combinado con el método de las secantes

```
> xi
[1] -0.3509050910 -0.0001506973
```

Codigo:

```
rosenbrockFunction=function(x1,x2) (1/2)*(100*(x2-(x1)^2)^2 +(1-x1)^2)
gradientRosenbrock=function(z) {
  x=z[1]
```

```
   y=z[2]
   x1=200*x*y+200*x^3-1
   x2=100*(y-x^2)
   return(c(x1,x2))



}
alphafunction=function(fun,xi,alpha,di){
 g=function(alpha) {return(fun(x1+alpha*di))}
 return(g)
}
alpha<-function(fun,grad,xi,di,minError=0.01, method=1)
{
 alphafun=alphafunction(fun=fun,xi=xi,alpha,di=di)
 #biseccion
 if (method==1){
  a=0
  b=1
  alph=bisectionMethod(fun=fun,xi=xi,di=di,alphamin=a,alphamax=b,minError=minError)
  return(alph)
 }
 #secantes
 else if (method==2){
  alphaant=0
  alphaact=1
  alph=SecantMethod(fun=fun,xi=xi,di=di, alphaant=alphaant,
alphaact=alphaact,minError=minError)
  return(alph)
```

```
  }

}
bisectionMethod<-function(fun, xi,di ,maxiter=100,alphamin, alphamax, minError){
  error=abs(alphamax-alphamin)/2

  falphamax=fun(xi[1]+alphamax*di[1],xi[2]+alphamax*di[2])
  falphamin=fun(xi[1]+alphamin*di[1],xi[2]+alphamin*di[2])

  i=0
  if (falphamax*falphamin<0){
    while (!is.na(error) & (error>minError) & (i < maxiter)){


      alphamiddle=(alphamax+alphamin)/2

      falphamiddle=fun(xi[1]+falphamiddle*di[1],xi[2]+falphamiddle*di[2])
      if(falphamiddle < minError ){
        break()
      }
      if (falphamiddle*falphamin<0){
        alphamax=alphamiddle
        falphamax=falphamiddle
      }
      else if(falphamiddle*falphamax<0){
        alphamin=alphamiddle
        falphamin=falphamiddle
      }
      error=abs(alphamax-alphamin)
```

```
    i=i+1

  }
  return(alphamiddle)
 }
 else{
   if(abs(falphamax)>abs(falphamin)){

     return(alphamin)
   }
   else{
     return(alphamax)
   }
 }

}
SecantMethod=function(fun, xi,di ,maxiter=100,alphaant, alphaact, minError){
  error=abs(alphaact-alphaant)

  i=0
  while (!is.na(error) & (error>minError) & (i < maxiter)){

    yant=fun(xi[1]+alphaant*di[1],xi[2]+alphaant*di[2])
    yact=fun(xi[1]+alphaact*di[1],xi[2]+alphaact*di[2])

      temp=alphaact-yact*(alphaact-alphaant)/(yact-yant)
      alphaant=alphaact
      alphaact=temp
```

```
      error=abs(alphaact-alphaant)
      i=i+1

   }
   return(alphaact)


}

beta<-function(ri, rinew)
{
  n=t(rinew)%*%(rinew-ri)
  d=t(ri)%*%ri
  max(n/d,0)
}
ri=c(0,0)
gradient.conjugate <- function(fun,grad,x,max.iterations,minError)
{
  xi=x
  cont=0
  for (iteration in 1:max.iterations) {


    ri=-grad(xi)
    di=ri
    alphainic=1
    al=alpha(fun=fun,grad=grad,xi=xi,di=di, minError=0.0001,method=2)

    oldxi=xi
```

```
   xi[1]=xi[1]+al*di[1]
   xi[2]=xi[2]+al*di[2]

   error =xi -oldxi
   normerror=t(error)%*%error

   if (!is.na(normerror) & (normerror < minError))
   {break()}
   temp=-grad(xi)
   oldri=ri

   ri[1]=ri[1]-al*temp[1]
   ri[2]=ri[2]-al*temp[2]
   bet=beta(ri=oldri, rinew=ri)
   di[1]=ri[1]+bet*di[1]
   di[2]=ri[2]+bet*di[2]
   cont=cont+1

  }

  return(xi)
}

x=c(0,0)
xi=gradient.conjugate(fun=rosenbrockFunction,
grad=gradientRosenbrock,x=x,max.iterations=50,minError=0.001)
xi
```