Práctica 7

Búsqueda local

Introducción

En esta práctica se simula la búsqueda de los máximos o mínimos locales de una función dada con ayuda de dos modelos matemáticos.

Objetivos

1. Maximizar la función bidimensional g (x,y) con la técnica utilizada en el ejemplo uno para una función unidimensional.
2. Demostrar visualmente como se lleva a cabo el proceso de búsqueda del máximo para la función g(x,y).
3. Cambiar la regla de movimiento para la función f(x) utilizando el mecanismo de recocido simulado.

Simulación y Resultados

Para la tarea base se modificó el código para buscar el máximo de la función bidimensional g(x,y). Lo primero que se realizó fue programar una función llamada *BusquedaLocal* donde se incluyó el movimiento en dos dimensiones una Δx y Δy para evaluar a los cuatro vecinos más cercanos de nuestro punto inicial, (arriba, abajo, izquierda y derecha) la variable *best* donde se guarda el mejor resultado máximo encontrado y se actualiza en cada paso si es que se encuentra un máximo mejor para la función evaluada. Se realizó una simulación con esta función con un con diez réplicas y cuarenta pasos por réplica.

```
1. BusquedaLocal <- function() {
2.
3.   resul=data.frame()
4.   resuldatos=data.frame()
5.   currx <- runif(1, low, high)
6.   curry <- runif(1, low, high)
7.   best <- c(currx,curry)
8.
9.
10.
11.        for (pasos in 1:t) {
12.
13.          deltax <- runif(1, 0, step)
14.          deltay <- runif(1, 0, step)
15.          left <- currx - deltax
16.          right <- currx + deltax
17.          top <- curry + deltay
18.          bot <- curry - deltay
19.
20.          if (f(left,curry) > f(right,curry)) {
21.            bestx <- c(left,curry)
```

```
          } else {
            bestx <-c(right,curry)
          }
          if (f(currx,top) > f(currx,bot)) {
            besty <- c(currx,top)
          } else {
            besty <- c(currx,bot)
          }

          if (f(bestx[1],bestx[2])> f(besty[1],besty[2]))
          {  currx=bestx[1]
             curry=bestx[2]
          } else {
               currx=besty[1]
               curry=besty[2]
             }

          if (f(currx,curry) > f(best[1],best[2])) {
            best <- c(currx,curry)
          }
          resul=cbind(q,pasos,best[1],best[2],f(best[1],best[2]))
          resuldatos=rbind(resuldatos,resul)
        }

        return(resuldatos)

      }
```

En la figura 1 se pueden observar los resultados para esta simulación donde con una línea horizontal en color verde está marcado el valor máximo para la función calculado por Wolfram Alpha (0.666822), con líneas y puntos se observa cada una de las réplicas y como se alcanza el máximo en cada réplica aproximadamente en 20 pasos para casi todos las réplicas. En la figura 2 se aprecia con mayor detalle el comportamiento de los gráficos que se encuentran juntos en la primera figura. Con estas dos imágenes se demuestra que el código modificado realiza la búsqueda del máximo de la función sin importar cuantas réplicas se apliquen y sin importar el punto inicial de la simulación.

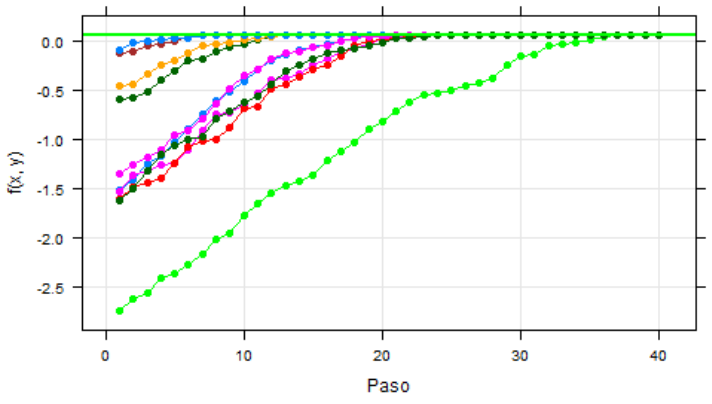


Figura 1. Diagrama de datos de búsqueda del máximo

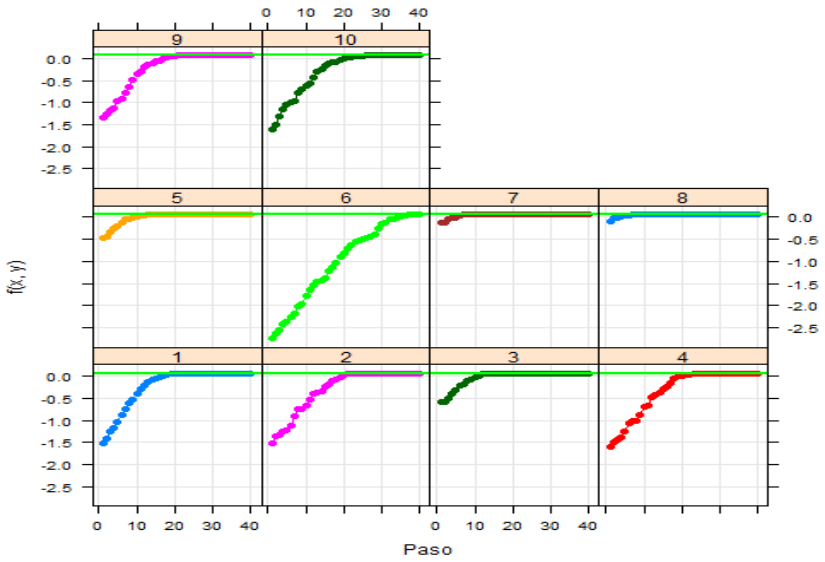


Figura 2. Diagrama de datos para cada una de las réplicas de la búsqueda del máximo

Para realizar los gráficos se incluyeron dos librerías *lattice* y *latticeextra*

```
#Graficar

png("p7.png",width = 500,height = 300)
xyplot(data=caminatas,f(x,y)~Paso,groups = Replicas,
       panel=function(x,y,subscripts,groups){
         panel.grid(h=-1,v=-1)
         panel.xyplot(x,y)
         panel.stripplot(x,y,
                         groups = groups,
  subscripts = subscripts,pch=19,type="o")
           panel.abline(h=wolfram,col="Green",lwd=2)
         })
     graphics.off()


     png("p7t.png",width = 500,height = 500)
     xyplot(data=caminatas,f(x,y)~Paso |Replicas,groups = Replicas
  ,
            panel=function(x,y,subscripts,groups){
              panel.grid(h=-1,v=-1)
              panel.xyplot(x,y)
              panel.stripplot(x,y,
                              groups = groups,
  subscripts = subscripts,pch=19,type="o")
              panel.abline(h=wolfram,col="Green",lwd=2)
            })
     graphics.off()
```

Reto 1

Para el primer reto se busca demostrar de manera visual como es que el código encuentra el valor máximo para la función g(x,y). Primero se utilizó como base el código dado por la práctica 7 para graficar la figura 3 al que llamaremos *paisaje*, grafico de tipo calor donde un color azul nos representa los valores máximos y los colores de un tono rosado nos representa los mínimos. Debido a la forma de la función el valor máximo dado por Wolfram Alpha se encuentra en el centro del paisaje en la zona azul.

```
1. f <- function(x, y) {
2.   return(((x + 0.5)^4 - 30 * x^2 - 20 * x + (y + 0.5)^4 - 30 * y^2
   - 20 * y)/100)
3. }
4. x <- seq(-6, 5, 0.25)
5. y <-  x
6. z <- outer(x, y, f)
7. colnames(z)=x
8. rownames(z)=y
9. d <- melt(z)
10.      names(d) <- c("x", "y", "z")
11.      png("p7_flat_2.png", width=500, height=500)
12.      levelplot(z ~ x * y, data = d)
13.      graphics.off()
```

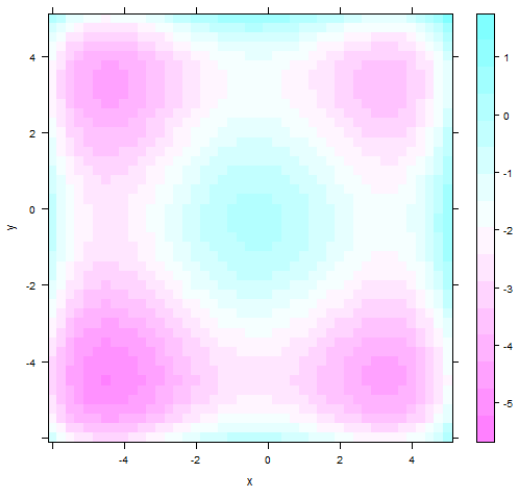
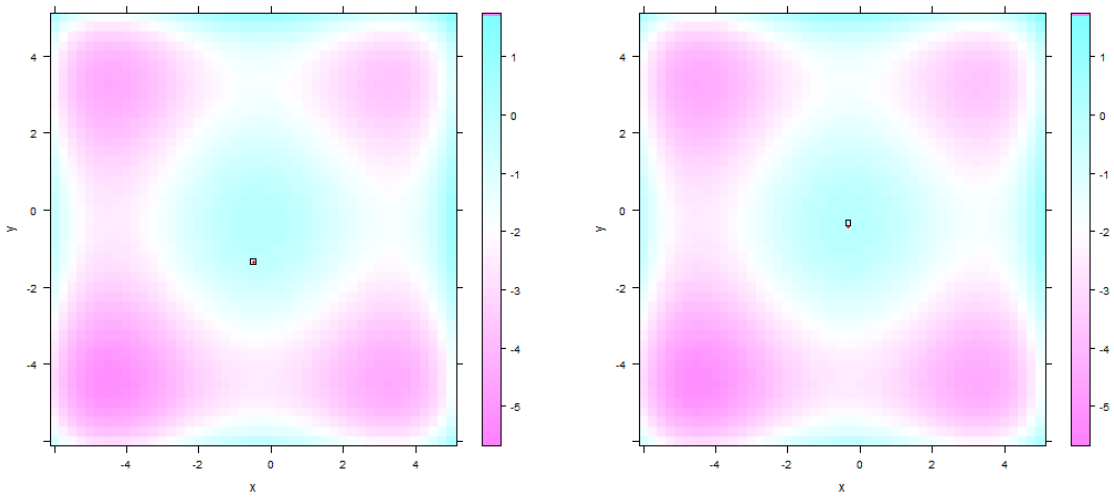


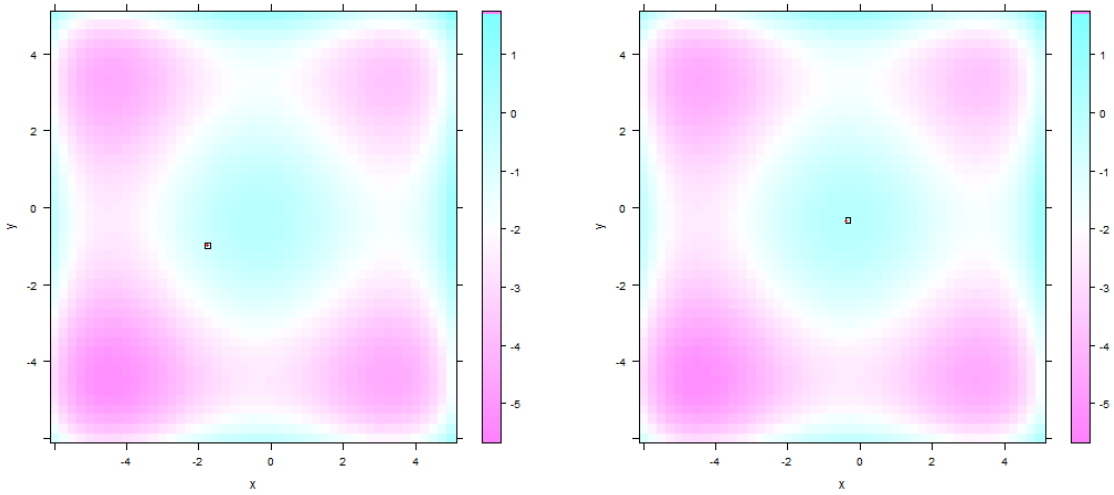
Figura 3. Paisaje realizado con la función g (x, y)

Se utilizó parte del código de la tarea base, concretamente la parte de la función llamada *BusquedaLocal* donde se agregó antes de comenzar cada ciclo del *for* graficar cada punto conforme se iban generando. En la figura 4 se puede apreciar 3 réplicas cada un representada por una fila donde la imagen de la izquierda de la fila es el paso inicial y la figura de la derecha es el último paso. Así mismo, se aprecian dos figuras dentro del paisaje el círculo rojo representa el posible nuevo mejor valor y el cuadrado negro sin relleno representa el mejor valor encontrado en el paso anterior.

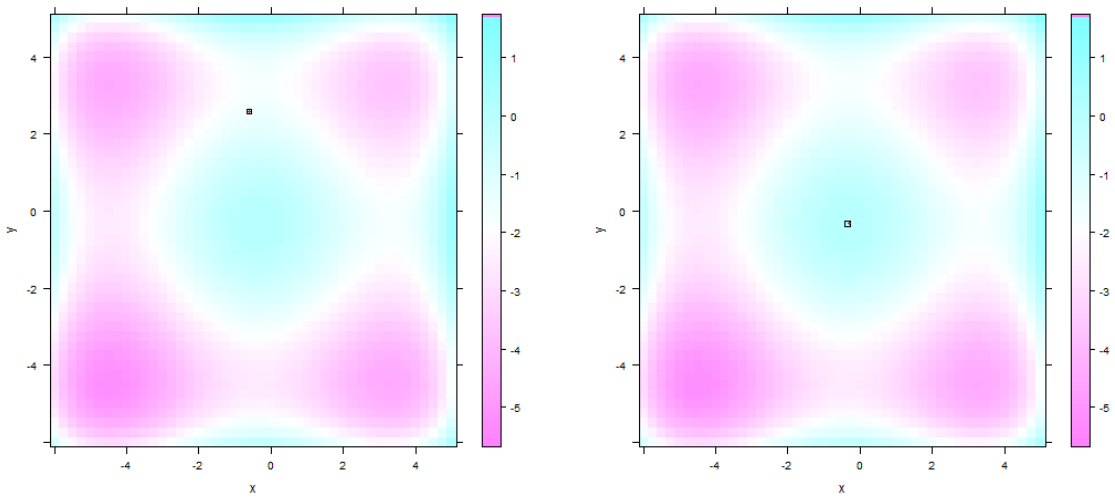
Se puede observar más a detalle el comportamiento del experimento si se visualizan los gifs que se encuentran dentro de la carpeta de la práctica 7 en la dirección *Reto1/Gifs* Se puede comprobar que sin importar la posición en donde se inicie la simulación en cada replica, siempre se alcanza el valor máximo aproximadamente en el paso 25.



a)



b)



c)

Figura 4. Ilustración del proceso de búsqueda de máximo para la función g(x, y)
a) Replica 1, b) Replica 2 c) Replica 3

Reto 2

Para el segundo reto se cambió la forma de elegir los vecinos utilizando el mecanismo de recocido simulado, donde en lugar de tener cuatro vecinos probables ahora se encuentra un mejor vecino por cada paso, este nuevo vecino puede representar una mejora (un valor más máximo) o una empeora (un valor más pequeño) si el valor evaluado en la función f (x) representa una mejora este será nuestro nuevo punto de partida para el siguiente paso en caso contrario, se podrá elegir ese vecino con una probabilidad representada por $\exp(-\delta/T)$ donde cada vez que se elija una empeora la temperatura (T) disminuirá lo que ocasionará que cada vez sea más difícil optar por una empeora. El código modificado se muestra a continuación:

```
for (tiempo in 1:tmax) {

  valor=runif(1)
  prob=exp(-d/Temp)
  delta <- runif(1, -step, step)
  xp <- x + delta
  d=f(xp)-f(x)
  resul=cbind(x,xp,Temp)


      if (d>0){
        x=xp
      }
      else{
        if  (valor<prob){
          Temp=Temp*e
          x=xp
        }
      }
      if (f(x)>f(best)){
```

```
21.         best=x
22.         }
23.       ##################################################################
   ################
24.       #for (tiempo in 1:tmax) {
25.         #delta <- runif(1, 0, step)
26.         #left <- curr - delta
27.         #right <- curr + delta
28.         #fl <- f(left)
29.         #fr <- f(right)
30.         #if (fl < fr) {
31.          # curr <- left
32.         #} else {
33.          # curr <- right
34.         #}
35.         #if (f(curr) < f(best)) { # minimizamos
36.           #best <- curr
37.         #}
38.       ##################################################################
   ################
39.
```

Dónde la parte verde del código es el movimiento en x como se encontraba originalmente y la parte que no se encuentra comentada es la nueva donde aplicamos la teoría de un recocido simulado.

En la figura 5 se muestra el comportamiento luego de calcular para 100 pasos el valor máximo y graficarlo en función de la temperatura inicial y a diferentes valores de E. Se observa que entre mayor es la temperatura el valor del máximo incrementa, esto es debido a que si lo comparamos con una partícula en movimiento al estar en un ambiente con mayor temperatura la partícula tendrá mayor energía y se moverá de manera más errática dando saltos más grandes. El comportamiento de estos gráficos varía en cada corrida pero para la mayoría de las corridas sigue esta misma tendencia.

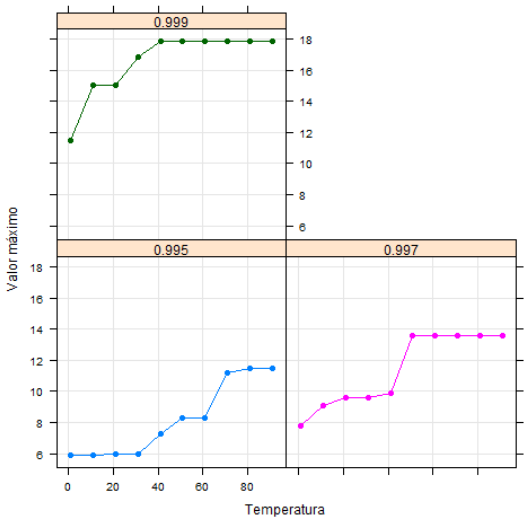
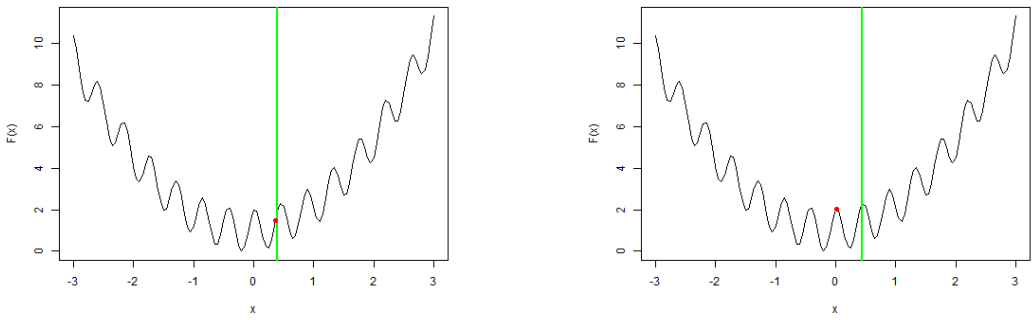
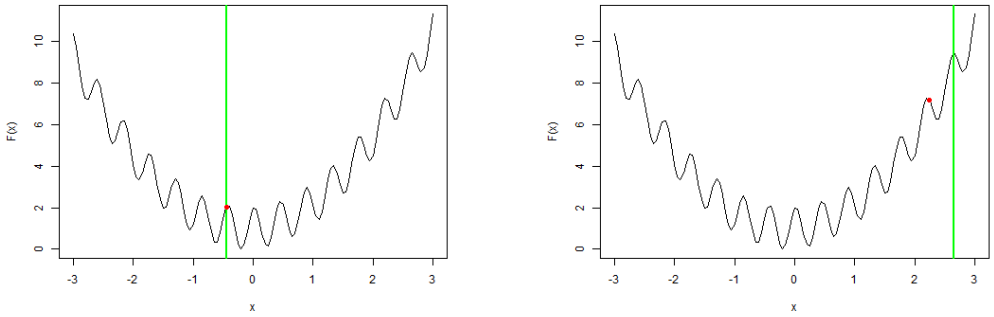


Figura 5. Gráfico de valor máximo por cada 100 pasos en función de la temperatura.

Para comprobar lo anterior el código modificado con la nueva forma de elegir los vecinos fue puesto a prueba con la función de f (x). En la figura 6 se observa el primer y último paso para 2 réplicas donde se observa que cuando la temperatura inicial es mayor, la probabilidad de alcanzar un valor de máximo más alto aumenta mientras que a menor temperatura el valor del mejor máximo encontrado no varía mucho del mejor valor encontrado en el paso 1. Ambas replicas se realizaron con un step igual a 0.3, una E= 0.995 y 100 pasos. Cabe mencionar que para algunas réplicas antes de llegar al paso cien la línea del mejor valor sale fuera de la función graficada y no tengo idea el por qué, algún error en la lógica del código o falta de alguna condición que limite el paso a la función.



a)



b)

Figura 6. Ilustración de la búsqueda del máximo para f (x). a) T=1°C, b) T=1000°C