Algoritmo de ajuste polinomial cúbico

Algorithm for Cubic Polynomial Fit

Step 1: Given x, ε, and Δx

Step 2: Compute $\alpha = \frac{a+b}{2}$, $f'(a)$ and $f'(\alpha)$

    If $f'(a)\,f'(\alpha) < 0$
        then $b = \alpha$
        else $a = \alpha$

Step 3: Repeat Step 2 until $f'(a)\,f'(\alpha) < 0$

Step 4: Using $f(a), f'(a), f(b), f'(b)$, compute μ, z, and w

Step 5: Compute $\bar{x}$

    If $|f'(\bar{x})| < \varepsilon$ goto Step 6
    If $f'(a)\,f'(\bar{x}) < 0$
        then $b = \bar{x}$
        else $a = \bar{x}$
            goto Step 4

Step 6: Converged. Print $x^* = \bar{x}$, $f(x^*) = f(\bar{x})$

Algoritmo de ajuste polinomial cúbico

$$\bar{x} = \left\{ \begin{array}{cl} x_2 & \text{if } \mu < 0 \\ x_2 - \mu\,(x_2 - x_1) & \text{if } 0 \leq \mu \leq 1 \\ x_1 & \text{if } \mu > 0 \end{array} \right\}$$

$$\mu = \frac{f'(x_2) + w - z}{f'(x_2) - f'(x_1) + 2w}$$

$$z = \frac{3\,(f(x_1) - f(x_2))}{x_2 - x_1} + f'(x_1) + f'(x_2)$$

$$w = \frac{x_2 - x_1}{|x_2 - x_1|} + \sqrt{z^2 - f'(x_1)\,f'(x_2)}$$

Consumo de gasolina

Un automóvil consume gasolina a una tasa de $\left(\frac{300}{x} + \frac{x}{3}\right)$ litros por cada 100 km, donde x es la velocidad en km/h. El costo del combustible es de un dólar por litro y el chofer paga una renta de 7 dólares por hora. Encontrar la velocidad constante que minimizará el costo total de un viaje de 600 km.

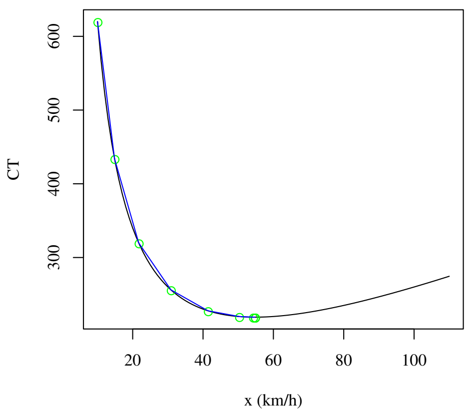
Consumo de gasolina

Un automóvil consume gasolina a una tasa de $\left(\frac{300}{x} + \frac{x}{3}\right)$ litros por cada 100 km, donde x es la velocidad en km/h. El costo del combustible es de un dólar por litro y el chofer paga una renta de 7 dólares por hora. Encontrar la velocidad constante que minimizará el costo total de un viaje de 600 km.

El costo total esta dado por:

$$CT = \frac{600}{100}\left(\frac{300}{x} + \frac{x}{3}\right) + 7\frac{600}{x}$$

Consumo de gasolina

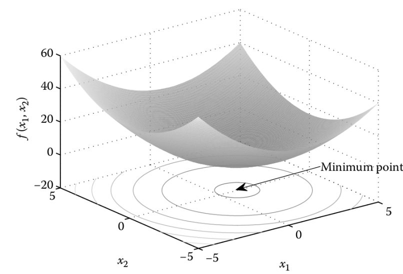


x	f(x)	f1(x)	f2(x)
10.00000	620.0000	-5.800000e+01	12.00000000
14.83333	434.1610	-2.526928e+01	3.67675742
21.70604	319.8328	-1.073474e+01	1.17338168
30.85459	256.1697	-4.302484e+00	0.40852814
41.38626	227.7482	-1.502990e+00	0.16928276
50.26484	219.8974	-3.747763e-01	0.09449056
54.23112	219.0998	-4.011249e-02	0.07523771
54.76426	219.0890	-5.838566e-04	0.07306166
54.77225	219.0890	-1.277710e-07	0.07302968
54.77226	219.0890	-6.217249e-15	0.07302967

Caso multidimensional

Consideremos un problema de optimización con dos variables,

$$f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1$$



Su gradiente o las derivadas parciales con respecto a cada variables es:

$$\nabla f(x_1, x_2) = \begin{bmatrix} 2x_1 - 2 \\ 2x_2 \end{bmatrix}$$

Caso multidimensional

El hessiano o la matriz de las segundas derivadas es:

$$[H] = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Si $x^T H x > 0$ es un **mínimo**. Si $x^T H x < 0$ es un **máximo**. Si $x^T H x = 0$ es un **punto silla**.

Algoritmo de Newton

Algorithm for Newton's Method

Step 1: Given $\boldsymbol{x}_i$ (starting value of design variable)
ε_1 (tolerance of function value from previous iteration)
ε_2 (tolerance on gradient value)
$\Delta\boldsymbol{x}$ (required for gradient computation)

Step 2: Compute $f(\boldsymbol{x}_i)$, $\nabla f(\boldsymbol{x}_i)$, and $[\boldsymbol{H}]$ (function, gradient, and Hessian)
$\boldsymbol{S}_i = -[\boldsymbol{H}]^{-1}\nabla f(\boldsymbol{x}_i)$ (search direction)
$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{S}_i$ (update the design vector)
If $|f(\boldsymbol{x}_{i+1}) - f(\boldsymbol{x}_i)| > \varepsilon_1$ or $\|\nabla f(\boldsymbol{x}_i)\| > \varepsilon_2$
then goto Step 2
else goto Step 3

Step 3: Converged. Print $\boldsymbol{x}^* = \boldsymbol{x}_{i+1}$, $f(\boldsymbol{x}^*) = f(\boldsymbol{x}_{i+1})$

Algoritmo de Newton

```r
library('Deriv')

# funcion a optimizar
f <- function(x1,x2){
  100*(x2-x1^2)^2 + (1-x1)^2 # optimo en (1,1)
}

iter <- 1 # contador
x <- c(3,0.5) # x inicial
xprev <- c(2,2) # poner xprev diferente a x
epsilon1 <- 0.0000001 # para la funcion
epsilon2 <- 0.0001 # para el gradiente

# primer derivada
f1 <- Deriv(f)
# segunda derivada
f2 <- Deriv(f, nderiv=2)

m <- f2(x[1],x[2]) # x evaluada en el hessiano
H <- matrix(c(m[1],m[2],m[3],m[4]),nrow=2) # convertir m a matriz
historial <- c(iter, x, f(x[1],x[2]), t(x)%*%H%*%x, sum(f1(x[1],x[2])^2)^(1/2))

while(abs(f(x[1],x[2])-f(xprev[1],xprev[2])) > epsilon1 || sum(f1(x[1],x[2])^2)^(1/2) > epsilon2){

  m <- f2(x[1],x[2]) # x evaluado en el hessiano
  H <- matrix(c(m[1],m[2],m[3],m[4]),nrow=2) # convertir m a matriz 2x2
```


Algoritmo de Newton

```r
  xprev <- x # copia de x
  S <- -solve(H)%*%f1(x[1],x[2]) # search direction
  x <- x + S # nueva x
  iter <- iter + 1 # contador
  historial <- rbind(historial, c(iter, x, f(x[1],x[2]), t(x)%*%H%*%x, sum(f1(x[1],x[2])^2)^(1/2)))
} # while

colnames(historial) <- c("iteracion","x1", "x2", "f(x)","t(x)*H*x", "norma")
print(historial)
```


Algoritmo de Newton

Consideremos el siguiente problema de minimización

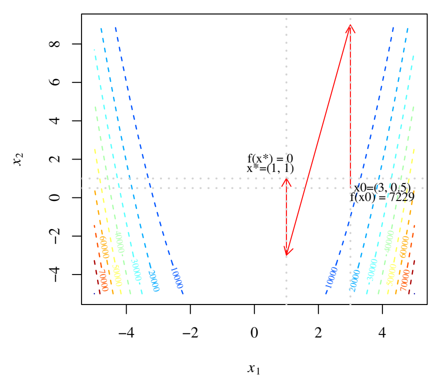
$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

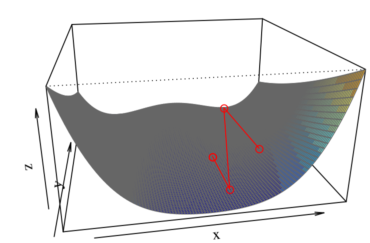
iter	x1	x2	f(x)	t(x)*H*x	norma
1	3.000000	0.5000000	7.229000e+03	91868.0000	1.034464e+04
2	2.998824	8.9929453	3.995298e+00	46794.0047	3.999307e+00
3	1.000553	-2.9919845	1.594477e+03	16176.6209	1.786554e+03
4	1.000552	1.0011039	3.044983e-07	1801.4438	1.103627e-03
5	1.000000	0.9999997	9.271921e-12	202.4418	1.361758e-04
6	1.000000	1.0000000	0.000000e+00	202.0001	0.000000e+00

Algoritmo de Newton

Consideremos el siguiente problema de minimización

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$





Algorithm for the Levenberg–Marquardt Method

Step 1: Given x_i (starting value of design variable)
ε_1 (tolerance of function value from previous iteration)
ε_2 (tolerance on gradient value)
Δx (required for gradient computation)

Step 2: Compute $f(x_i)$, $\nabla f(x_i)$, and $[H]$ (function, gradient, and Hessian)
$S_i = -[H + \lambda I]^{-1} \nabla f(x_i)$ (search direction)
$x_{i+1} = x_i + S_i$ (update the design vector)
If $f(x_{i+1}) < f(x_i)$
then change the value of λ as $\lambda/2$
else change the value of λ as 2λ
If $\left|f(x_{i+1}) - f(x_i)\right| > \varepsilon_1$ or $\|\nabla f(x_i)\| > \varepsilon_2$
then goto Step 2
else goto Step 3

Step 3: Converged. Print $x^* = x_{i+1}$, $f(x^*) = f(x_{i+1})$

Optimización restringida

Optimización restringida

Considere el siguiente problema de optimización.

Minimizar

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 5)^2$$

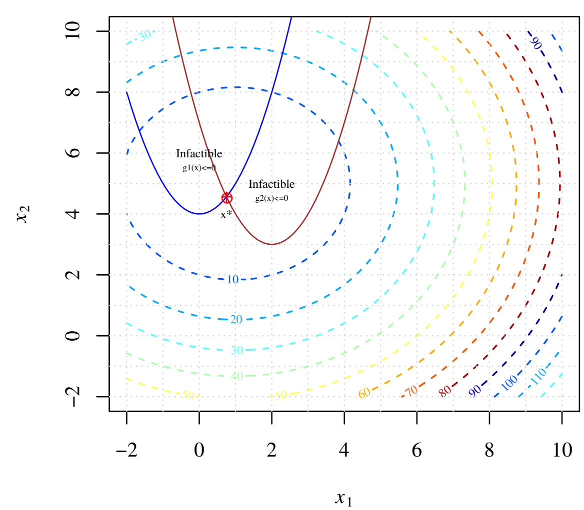
Sujeto a

$$g_1(x_1, x_2) = -x_1^2 + x_2 - 4 \leq 0$$

$$g_2(x_1, x_2) = -(x_1 - 2)^2 + x_2 - 3 \leq 0$$

donde los valores óptimos de $x_1 = 0.75$ y $x_2 = 4.5625$, con valor de la función objetivo $f(x_1, x_2) = 0.2539624$.

Optimización restringida



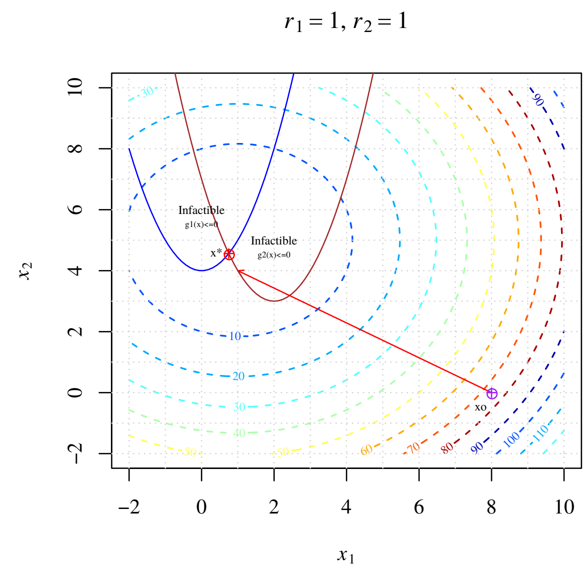
Manejo de restricciones

Un enfoque simple para el manejo de las restricciones es usar un método de penalización para resolver un problema de optimización con restricciones mediante un algoritmo para problemas no restringidos. La función objetivo modificada quedaría de la siguiente forma:

$$F(x) = f(x) + r_1 \cdot g_1(x) + r_2 \cdot g_2(x)$$

donde $r_1, r_2 > 0$ son los parámetros de penalidad.

Optimización restringida



Manejo de restricciones

La idea principal es encontrar los valores de r_1 y r_2 tal que se conserve la factibilidad de las restricciones g_1 y g_2. La condición de optimalidad esta dada por,

$$-f'(x_1, x_2) = r_1 \cdot g_1'(x_1, x_2) + r_2 \cdot g_2'(x_1, x_2)$$

Manejo de restricciones

La idea principal es encontrar los valores de r_1 y r_2 tal que se conserve la factibilidad de las restricciones g_1 y g_2. La condición de optimalidad esta dada por,

$$-f'(x_1, x_2) = r_1 \cdot g_1'(x_1, x_2) + r_2 \cdot g_2'(x_1, x_2)$$

Para el ejemplo, se tiene lo siguiente:

$$\begin{bmatrix} 0.5 \\ 0.875 \end{bmatrix} \approx 0.4218 \begin{bmatrix} -1.5 \\ 1 \end{bmatrix} + 0.4531 \begin{bmatrix} 2.5 \\ 1 \end{bmatrix}$$

Manejo de restricciones

La idea principal es encontrar los valores de r_1 y r_2 tal que se conserve la factibilidad de las restricciones g_1 y g_2. La condición de optimalidad esta dada por,

$$-f'(x_1, x_2) = r_1 \cdot g_1'(x_1, x_2) + r_2 \cdot g_2'(x_1, x_2)$$

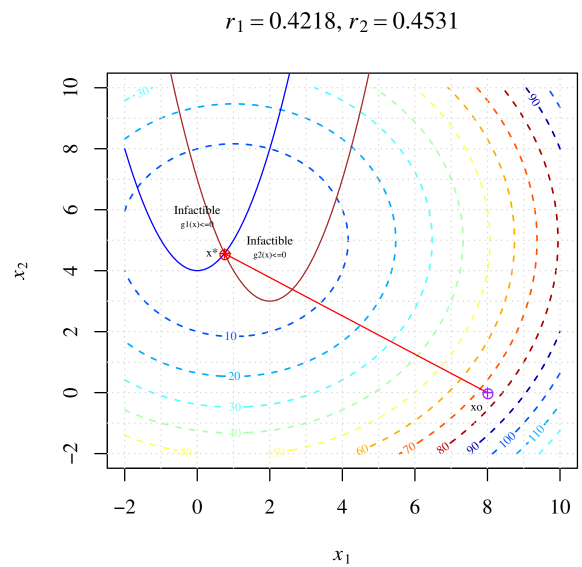
Para el ejemplo, se tiene lo siguiente:

$$\begin{bmatrix} 0.5 \\ 0.875 \end{bmatrix} \approx 0.4218 \begin{bmatrix} -1.5 \\ 1 \end{bmatrix} + 0.4531 \begin{bmatrix} 2.5 \\ 1 \end{bmatrix}$$

Verificando la segunda derivada en $F(x)$ se tiene una matriz definida positiva, por lo que se determina que es un mínimo,

$$\begin{bmatrix} 0.25 & 0 \\ 0 & 2 \end{bmatrix}$$

Optimización restringida



Optimización sin derivadas

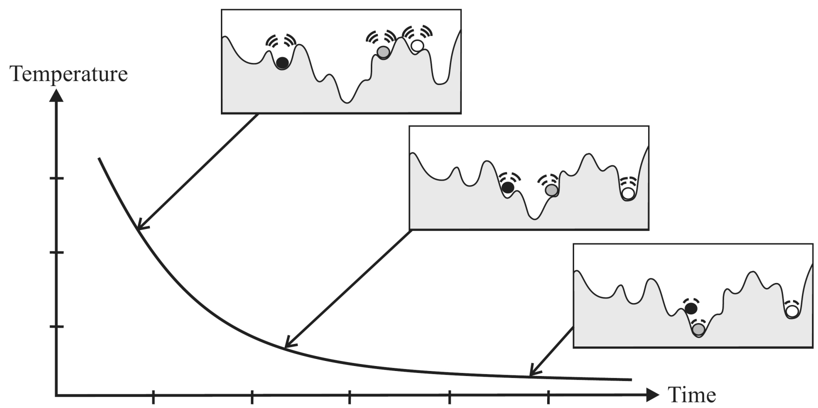
Recocido simulado

El recocido simulado (SA por sus siglas en inglés, *Simulated Annealing*) es una metaheurística que simula el recocido de un metal, en el que el metal se calienta a una temperatura cerca de su punto de fusión y luego se enfría lentamente para permitir que las partículas se muevan hacia un estado de mínima energía.

A continuación, se presenta la analogía del recocido en la naturaleza y el recocido simulado en optimización.

configuraciones atómicas (estados del sistema) $\longleftrightarrow$ soluciones candidatas
temperatura $\longleftrightarrow$ tendencia para explorar el espacio de búsqueda
enfriamiento $\longleftrightarrow$ decrecimiento de la tendencia para explorar
energía del estado $\longleftrightarrow$ el costo de la solución candidata
estado de mínima energía $\longleftrightarrow$ minimización de la función de costo

Proceso de enfriamiento



Inicialmente valores grandes de T aceptan cualquier solución (estado). Cuando T tiende a cero, se dejan de aceptar soluciones. (Exploración $\rightarrow$ Explotación)

Criterio de aceptación-rechazo de Metrópolis

El criterio de aceptación determina si $\hat{\mathbf{x}}$ se acepta o rechaza como solución en la iteración $k+1$, con la siguiente probabilidad:

$$\Pr(\text{aceptar } \hat{\mathbf{x}}) = \begin{cases} 1, & \text{si } f(\hat{\mathbf{x}}) < f(\mathbf{x}_k) \\ \exp\left(\dfrac{f(\mathbf{x}_k) - f(\hat{\mathbf{x}})}{T}\right), & \text{si } f(\hat{\mathbf{x}}) \geq f(\mathbf{x}_k) \end{cases}$$

Para implementar lo anterior, en el caso que $f(\hat{\mathbf{x}}) \geq f(\mathbf{x}_k)$, es necesario realizar lo siguiente:

$$\text{Si} \quad \mathcal{U}(0,1) < \exp[(f(\mathbf{x}_k) - f(\hat{\mathbf{x}}))/T]$$

definir $\mathbf{x}_{k+1} \leftarrow \hat{\mathbf{x}}$. En caso contrario, definir $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$.

Algoritmo SA

Paso 0 **(Inicialización)** Elegir un vector inicial $\mathbf{x}_0$ de manera aleatoria o determinista. Definir $k \leftarrow 0$ y una temperatura inicial T.

Paso 1 Generar un vector independiente $\mathbf{d}_k \sim \mathcal{N}(0, \sigma^2)$ y sumarlo al vector actual $\mathbf{x}_k$, es decir, $\hat{\mathbf{x}} \leftarrow \mathbf{x}_k + \mathbf{d}_k$.

Paso 2 Si $f(\hat{\mathbf{x}}) < f(\mathbf{x}_k)$, definir $\mathbf{x}_{k+1} \leftarrow \hat{\mathbf{x}}$. En caso contrario, si $\mathcal{U}(0,1) < \exp[(f(\mathbf{x}_k) - f(\hat{\mathbf{x}}))/T]$ definir $\mathbf{x}_{k+1} \leftarrow \hat{\mathbf{x}}$; si no $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$.

Paso 3 Reducir la temperatura T de acuerdo a un programa de enfriamiento, por ejemplo, $T \leftarrow \alpha T$, donde $\alpha \in (0,1)$.

Paso 4 Detener el algoritmo hasta alcanzar K iteraciones o un valor mínimo de T. En caso contrario, ir al **Paso 1**. Definir $k \leftarrow k+1$.

Algoritmo SA

```r
# funcion objetivo con penalizacion de restricciones g1 y g2
f1 <- function(x){
  res <- (x[1]-1)^2 + (x[2]-5)^2
  if((-x[1]^2 + x[2]-4)> 0) res <- res + 1000 # g1
  if((-(x[1]-2)^2 + x[2]-3)> 0) res <- res + 1000 # g2
return(res)
}

# Restriccion g1
g1 <- function(x) -x[1]^2 + x[2] - 4
# Restriccion g2
g2 <- function(x) -(x[1]-2)^2 + x[2] - 3

RecocidoS <- function(iteraciones){
  # Paso 0
  mejorX <- c(-1.0,1.0)
  mejorFx <- f1(mejorX)
  historial <- c(1, mejorX, f1(mejorX),g1(mejorX),g2(mejorX))
  temp <- 1000

  for(i in 1:iteraciones){
    # Paso 1
    nuevoX <- mejorX + rnorm(2, 0, 1)

    # Paso 2
    if(f1(nuevoX) < f1(mejorX)){
      mejorX <- nuevoX
      mejorFx <- f1(nuevoX)
```

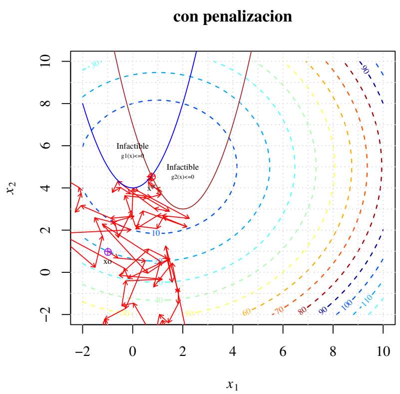
Algoritmo SA

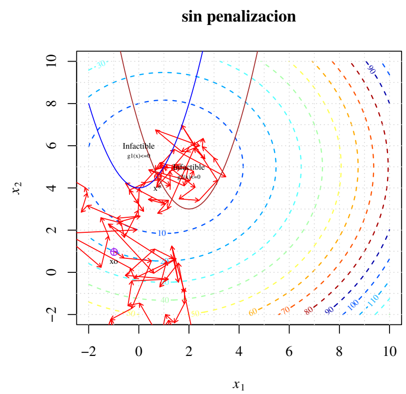
```
      historial <- rbind(historial, c(i+1, mejorX, f1(mejorX),g1(mejorX),g2(mejorX)))
    }else{
      aleatorio <- runif(1)
      if(aleatorio < exp((f1(mejorX)-f1(nuevoX))/temp)){
        mejorX <- nuevoX
        mejorFx <- f1(nuevoX)
        historial <- rbind(historial, c(i+1, mejorX, f1(mejorX),g1(mejorX),g2(mejorX)))
      }
    }
    # Paso 3
    temp <- 0.95*temp
  }

  colnames(historial) <- c("x","x1","x2","f1(x)","g1(x)","g2(x)")
  print(historial)
  cat('x1 = ',mejorX[1], ' x2 = ', mejorX[2], 'con f(x1, x2)=', mejorFx,'\n')
}

set.seed(1984)
RecocidoS(20000)
```

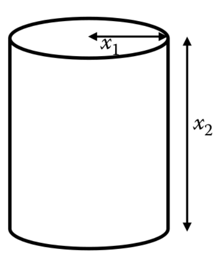

Solución con el algoritmo SA





Caso práctico

Una empresa desea diseñar un envase cilíndrico para las bebidas que produce. El envase debe contener 330 mililitros de líquido. Debido al costo del material, la empresa desea minimizar las dimensiones de diseño, cumpliendo con el volumen requerido para el líquido.



Caso práctico

Se tienen dos tapas circulares, el área de las tapas son

Caso práctico

Se tienen dos tapas circulares, el área de las tapas son

$$2\pi x_1^2.$$

Caso práctico

Se tienen dos tapas circulares, el área de las tapas son

$$2\pi x_1^2.$$

Se tienen dos partes curveadas para el cuerpo del envase, el área es

Caso práctico

Se tienen dos tapas circulares, el área de las tapas son

$$2\pi x_1^2.$$

Se tienen dos partes curveadas para el cuerpo del envase, el área es

$$2\pi x_1 x_2.$$

Caso práctico

Se tienen dos tapas circulares, el área de las tapas son

$$2\pi x_1^2.$$

Se tienen dos partes curveadas para el cuerpo del envase, el área es

$$2\pi x_1 x_2.$$

El volumen del envase cilídrico esta dado por

Caso práctico

Se tienen dos tapas circulares, el área de las tapas son

$$2\pi x_1^2.$$

Se tienen dos partes curveadas para el cuerpo del envase, el área es

$$2\pi x_1 x_2.$$

El volumen del envase cilídrico esta dado por

$$\pi x_1^2 x_2.$$

Caso práctico

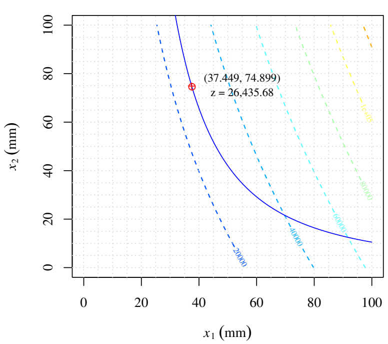
El modelo matemático para el diseño del envase cilíndrico es

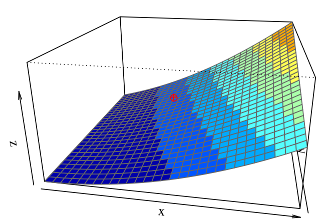
$$\min \quad 2\pi x_1^2 + 2\pi x_1 x_2$$

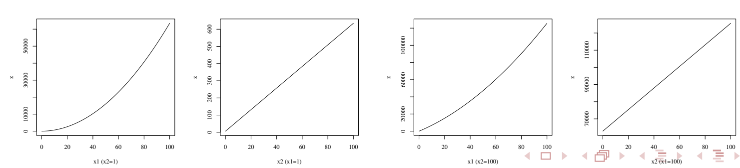
sujeto a

$$\pi x_1^2 x_2 = 330000$$

Caso práctico







Solución con el algoritmo SA