Trabalho Computacional I & II
de
Otimização Não-Linear

Thiago Malta Coutinho - 2014123335

7 de dezembro de 2018

Contents

Trabalho I

Exercicio 1:

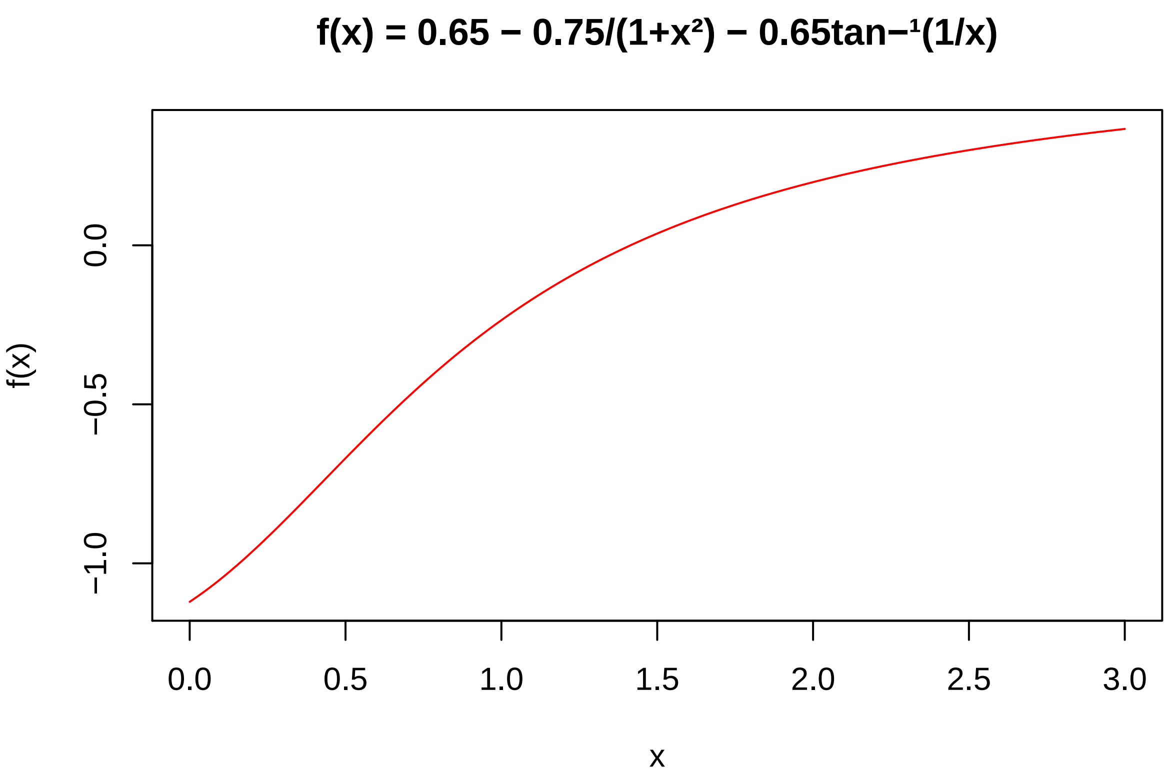
Encontre o minimo da funcao a seguir no intervalo [0,3]:

$f(x) = 0.65 - 0.75/(1+x^2) - 0.65\tan^{-1}(1/x)$

```r
f1x <- function(x){

  a = 0.75/(1+x^2)
  b = 0.65*atan(1/x)

  return(0.65 - a - b)
}
```

i - Plote o grafico da funcao no intervalo indicado.

```r
xseq <- seq(0, 3, 0.01)
f1_xseq <- f1x(xseq)

plot(xseq, f1_xseq, type='l', main = 'f(x) = 0.65 - 0.75/(1+x²) - 0.65tan-¹(1/x)',
     xlab = "x", ylab = "f(x)", col=c("red"))
```



ii - Usando o metodo da bissecao:

```
bissection_method <- function(a, b, fx, tol = 1e-6){
  ## Especifica tres pontos: u, c e v, igualmente espacados no intervalo inicial (a,b)
  tam_intervalo <- b-a
  k <- 0
  while(tam_intervalo > tol){
    ## Divide o intervalo atual por 3 para encontrar o passo.
    passo <- tam_intervalo/3

    ## Calcula os valores de u, c e v.
    u <- a + 1*passo
    c <- a + 2*passo
    v <- a + 3*passo

    ## Calcula os valores da funcao nos pontos u, c e v.
    fx_u <- fx(u)
    fx_c <- fx(c)
    fx_v <- fx(v)

    ## Verifica condicoes para atualizacao do intervalo.
    if( (fx_c < fx_v) && (fx_u < fx_c) ){
      b = c
      c = u
    }
    if( (fx_u > fx_c) && (fx_c > fx_v) ){
      a = c
      c = v
    }
    if( (fx_u > fx_c) && (fx_v > fx_c) ){
      a = u
      b = v
    }

    ## Encontra o tamanho (L) do intervalo atual.
    tam_intervalo <- b-a
    k <- k + 1
  }
  return(c(c(a,b),k))
}

min_f1x_bissecao <- bissection_method(a = 0, b = 3, fx = f1x, tol = 1e-6)
cat("INTERVALO ENCONTRADO: \n")
```

```
## INTERVALO ENCONTRADO:
```

```
print(min_f1x_bissecao[1])
```

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

```
print(min_f1x_bissecao[2])
```

```
## [1] 9.15682e-07
```

```
cat("VALOR DA FUNCAO: \n")
```

```
## VALOR DA FUNCAO:
```

```r
print(f1x(min_f1x_bissecao[1]))
```

```
## [1] -1.121018
```

iii - Usando o metodo da secao aurea:

```r
aurea_method <- function(a, b, fx, tol = 1e-6){

  tam_intervalo <- b-a
  k <- 0
  while(tam_intervalo > tol){

    u = b - 0.618*tam_intervalo
    v = a + 0.618*tam_intervalo

    fx_u <- fx(u)
    fx_v <- fx(v)

    if(fx_u < fx_v){
      b = v
    }else{ ## fx_u >= fx_v
      a = u
    }

    tam_intervalo = b-a
    k <- k + 1
  }

  return(c(c(a,b),k))
}

min_f1x_aurea <- aurea_method(a = 0, b = 3, fx = f1x, tol = 1e-6)

cat("INTERVALO ENCONTRADO: \n")
```

```
## INTERVALO ENCONTRADO:
```

```r
print(min_f1x_aurea[1])
```

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

```r
print(min_f1x_aurea[2])
```

```
## [1] 9.948646e-07
```

```r
cat("VALOR DA FUNCAO: \n")
```

```
## VALOR DA FUNCAO:
```

```r
print(f1x(min_f1x_aurea[1]))
```

```
## [1] -1.121018
```

iv - Compare os resultados obtidos pelos dois metodos

```r
print("NUMERO DE ITERACOES METODO DA BISSECAO")
```

```
## [1] "NUMERO DE ITERACOES METODO DA BISSECAO"
```

```r
print(min_f1x_bissecao[3])
```

```
## [1] 37
```

```r
print("NUMERO DE ITERACOES METODO DA SECAO AUREA")
```

```
## [1] "NUMERO DE ITERACOES METODO DA SECAO AUREA"
```

```r
print(min_f1x_aurea[3])
```

```
## [1] 31
```

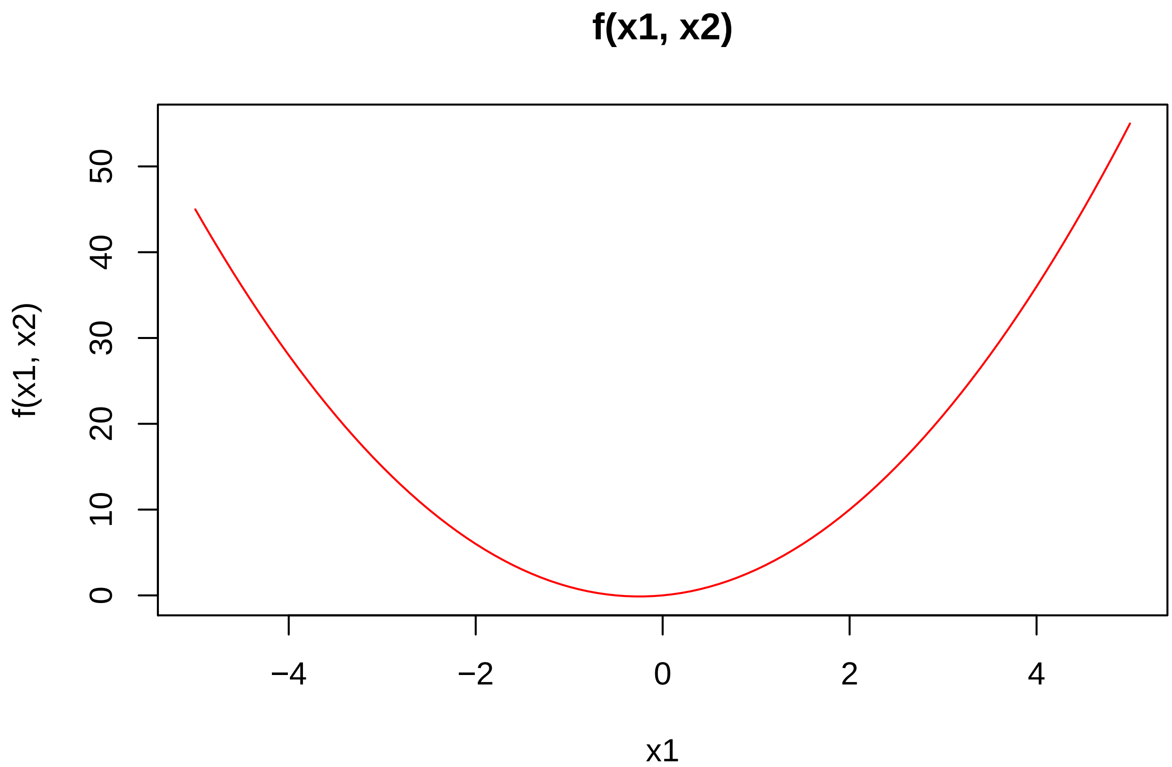
Exercicio 2:

Minimize a funcao f(x) = x1 - x2 + 2x1² + 2x1x2 + x2² a partir do ponto xk = (0,0) na direcao dk = (-1, 0).

```
f2x <- function(x1, x2 = 0){
  return(x1 - x2 + 2*x1^2 + 2*x1*x2 + x2^2)
}
```

i - Plote o grafico da funcao no intervalo [-5, 5]

```
xseq <- seq(-5, 5, 0.1)
plot(xseq, f2x(xseq), type = 'l', col = c('red'), main = 'f(x1, x2)', xlab='x1', ylab='f(x1, x2)')
```



ii - Usando o metodo da bissecao

```
min_f2x_bissecao <- bissection_method(a = -5, b = 5, fx = f2x, tol = 1e-6)
cat("INTERVALO ENCONTRADO: \n")
```

```
## INTERVALO ENCONTRADO:
```

```
print(min_f2x_bissecao[1])
```

```
## [1] -0.2500005
```

```
print(min_f2x_bissecao[2])
```

```
## [1] -0.2499996
```

```
cat("VALOR DA FUNCAO: \n")
```

```
## VALOR DA FUNCAO:
```

```r
print(f2x(min_f2x_bissecao[1]))
```

```
## [1] -0.125
```

iii - Usando o metodo da secao aurea

```r
min_f2x_aurea <- aurea_method(a = -5, b = 5, fx = f2x, tol = 1e-6)
cat("INTERVALO ENCONTRADO: \n")
```

```
## INTERVALO ENCONTRADO:
```

```r
print(min_f2x_aurea[1])
```

```
## [1] -0.2500004
```

```r
print(min_f2x_aurea[2])
```

```
## [1] -0.2499996
```

```r
cat("VALOR DA FUNCAO: \n")
```

```
## VALOR DA FUNCAO:
```

```r
print(f2x(min_f2x_aurea[1]))
```

```
## [1] -0.125
```

iv - Compare os resultados obtidos pelos dois metodos

Numero de iteracoes feitas pelo metodo da secao aurea.

```r
print("NUMERO DE ITERACOES METODO DA BISSECAO")
```

```
## [1] "NUMERO DE ITERACOES METODO DA BISSECAO"
```

```r
print(min_f2x_bissecao[3])
```

```
## [1] 40
```

```r
print("NUMERO DE ITERACOES METODO DA SECAO AUREA")
```

```
## [1] "NUMERO DE ITERACOES METODO DA SECAO AUREA"
```

```r
print(min_f2x_aurea[3])
```

```
## [1] 34
```

Trabalho II

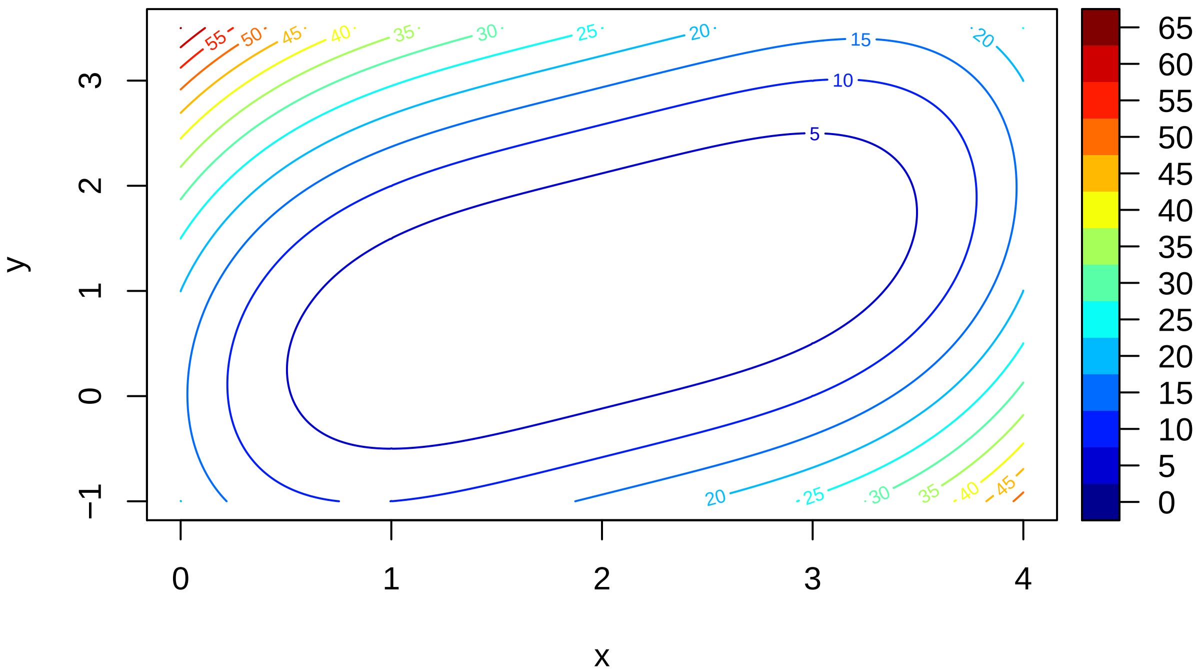
Exercicio 1:

Considere o problema de minimizar f(x) = (x1-2)^4 + (x1-2x2)^2 a partir do ponto x0 = (3,2).

```r
library(plot3D)
f3x <- function(x){
  x1 <- x[1]
  x2 <- x[2]
  return((x1-2)^4 + (x1-2*x2)^2)
}
```

i - Plote as curvas de nivel da funcao

```
xseq <- seq(0, 4, 0.005)
yseq <- seq(-1, 3.5, 0.005)

M_f3x <- matrix(-1, nrow = length(xseq), ncol = length(yseq))

for(i in 1:length(xseq)){
  x1 <- xseq[i]
  for(j in 1:length(yseq)){
    x2 <- yseq[j]
    x <- c(x1,x2)
    M_f3x[i, j] <- f3x(x)
  }
}

contour2D(z = M_f3x, x = xseq, y = yseq)
```



ii - Implementacao do metodo do gradiente

Para a implementacao do metodo do gradiente, foram feitas alteracoes na funcao de secao aurea e de bissecao para que as mesmas aceitem vetores como parametros ao inves de duas coordenadas. A funcao de busca irrestrita foi implementada para encontrar o intervalo inicial a ser otimizado pelas funcoes de otimizacao unidimensional. O gradiente da funcao no ponto avaliado foi calculado utilizando o metodo das diferencas finitas. E o criterio de parada escolhido foi 5 avaliacoes de funcoes menores que a tolerancia.

```
vec_bissection_method <- function(a, b, v1, v2, fx, tol = 1e-6){
  ## Especifica tres pontos: u, c e v, igualmente espacados no intervalo inicial (a,b)

  tam_intervalo <- b-a
  k <- 0
  while(tam_intervalo > tol){
```

```
    ## Divide o intervalo atual por 3 para encontrar o passo.
    passo <- tam_intervalo/3

    ## Calcula os valores de u, c e v.
    u <- a + 1*passo
    c <- a + 2*passo
    v <- a + 3*passo

    ## Calcula os valores da funcao nos pontos u, c e v.
    fx_u <- fx(v1 + u*v2)
    fx_c <- fx(v1 + c*v2)
    fx_v <- fx(v1 + v*v2)

    ## Verifica condicoes para atualizacao do intervalo.
    if((fx_c < fx_v) && (fx_u < fx_c)){
      b = c
      c = u
    }
    if((fx_u > fx_c) && (fx_c > fx_v)){
      a = c
      c = v
    }
    if((fx_u > fx_c) && (fx_v > fx_c)){
      a = u
      b = v
    }

    ## Encontra o tamanho (L) do intervalo atual.
    tam_intervalo <- b-a
    k <- k + 1
  }
  return(c(c(a,b),k))
}

vec_aurea_method <- function(a, b, v1, v2, fx, tol = 1e-6){

  tam_intervalo <- b-a
  k <- 0
  while(tam_intervalo > tol){

    u = b - 0.618*tam_intervalo
    v = a + 0.618*tam_intervalo

    fx_u <- fx(v1 + u*v2)
    fx_v <- fx(v1 + v*v2)

    if(fx_u < fx_v){
      b = v
    }else{ ## fx_u >= fx_v
      a = u
    }

    tam_intervalo = b-a
```

```
    k <- k + 1
  }


  return(c(c(a,b), k))
}

unrestricted_search <- function(x0, d, step, fx){

  i <- 1
  last_value <- fx(x0)
  curr_step <- matrix(rep(1+(i*step), length(x0)), nrow=1)
  curr_value <- fx(x0 + (curr_step*d))

  while(curr_value < last_value){

    i = i + 1
    last_value = curr_value
    curr_step <- matrix(rep(1+(i*step), length(x0)), nrow=1)
    curr_value <- fx(x0 + (curr_step*d) )

  }

  return(curr_step)
}

gradient <- function(x, fx, delta=1e-6){

  ## Pega numero de dimensoes
  m <- length(x)
  grad <- matrix(0, nrow=1, ncol=m)

  x_plus_delta <- c()
  x_minus_delta <- c()

  for(i in 1:m){
      x_plus_delta <- x
      x_minus_delta <- x
      x_plus_delta[i] <- x[i] + delta
      x_minus_delta[i] <- x[i] - delta
      grad[i] <- (fx(x_plus_delta) - fx(x_minus_delta))/(2*delta)
  }

  return(grad)
}

max_diff <- function(fx_vec, k = 5){

  n <- length(fx_vec)
  if(n < k){
    return(1e10)
  }else{
    fx_vec <- unlist(fx_vec)
```

```r
    fk_plus <- max(fx_vec[(n-k):k])
    fk_minus <- min(fx_vec[(n-k):k])

    return(fk_plus-fk_minus)
  }
}

gradient_method <- function(x, fx, unidim_min_func, tol){

  grad <- list()
  d <- list()
  alpha <- list()
  x_ <- list()
  fx_vec <- list()

  x_[[1]] = x
  fx_vec[1] = fx(x_[[1]])

  vec_k <- list()

  k <- 1
  while(max_diff(fx_vec) > tol){
    ## TIRA O GRADIENTE
    grad[[k]] <- gradient(x = x_[[k]], fx = fx)
    ## ATRIBUI DIREÇÂO D
    d[[k]] <- -grad[[k]]
    ## FAZ BUSCA INICIAL IRRESTRITA NA DIRECAO D
    intervalo <- c(0, unrestricted_search(x_[[k]], d[[k]], tol/2, fx) )
    ## PROCURA ALFA MINIMO
    aux <- unidim_min_func(intervalo[1], intervalo[2], x_[[k]], d[[k]], fx)
    alpha[[k]] <- c(aux[1], aux[2])
    vec_k[[k]] <- aux[3]
    ## ATUALIZA X
    x_[[k+1]] <- x_[[k]] + alpha[[k]]*d[[k]]
    ## AVALIA FUNCAO NO PONTO X ATUAL
    fx_vec[k+1] <- fx(x_[[k+1]])

    k <- k + 1
  }

  return(list(fx_vec = fx_vec,
              alpha = alpha,
              grad = grad,
              x_ = x_,
              d = d,
              k = k,
              vec_k = vec_k))
}
```

ii.i - Resolva o problema usando o metodo do gradiente acoplando: metodo da secao aurea

```
grad_aurea <- gradient_method(x = c(3,2), fx = f3x, unidim_min_func = vec_aurea_method, tol = 1e-6)
cat("MINIMO ENCONTRADO EM: \n")
```

```
## MINIMO ENCONTRADO EM:
```

```
n <- length(grad_aurea$x_)
print(grad_aurea$x_[n])
```

```
## [[1]]
##          [,1]     [,2]
## [1,] 2.269142 1.128072
```

ii.ii - Resolva o problema usando o metodo do gradiente acoplando: metodo de bissecao de intervalos

```
grad_bissection <- gradient_method(x = c(3,2), fx = f3x, unidim_min_func = vec_bissection_method, tol =
cat("MINIMO ENCONTRADO EM: \n")
```

```
## MINIMO ENCONTRADO EM:
```

```
n <- length(grad_bissection$x_)
print(grad_bissection$x_[n])
```

```
## [[1]]
##          [,1]     [,2]
## [1,] 2.269144 1.128073
```

iii - Compare os resultados obtidos com as diferentes estrategias de minimizacao unidimensional em termos de numeros de iteracoes e numeros de avaliacoes de funcao

```
cat("NUMERO DE ITERACOES METODO GRADIENTE COM SECAO AUREA:")
```

```
## NUMERO DE ITERACOES METODO GRADIENTE COM SECAO AUREA:
```

```
print(grad_aurea$k)
```

```
## [1] 10
```

```
cat("NUMERO DE ITERACOES METODO GRADIENTE COM METODO BISSECAO:")
```

```
## NUMERO DE ITERACOES METODO GRADIENTE COM METODO BISSECAO:
```

```
print(grad_bissection$k)
```

```
## [1] 10
```

```
cat("\n\n")
```

```
cat("NUMERO DE ITERACOES DA SECAO AUREA POR ITERACAO DO METODO DO GRADIENTE")
```

```
## NUMERO DE ITERACOES DA SECAO AUREA POR ITERACAO DO METODO DO GRADIENTE
```

```
print(grad_aurea$vec_k)
```

```
## [[1]]
## [1] 29
```

```
##
## [[2]]
## [1] 29
##
## [[3]]
## [1] 29
##
## [[4]]
## [1] 29
##
## [[5]]
## [1] 29
##
## [[6]]
## [1] 29
##
## [[7]]
## [1] 29
##
## [[8]]
## [1] 29
##
## [[9]]
## [1] 29
```

```
cat("NUMERO DE ITERACOES DA BISSECAO POR ITERACAO DO METODO DO GRADIENTE")
```

```
## NUMERO DE ITERACOES DA BISSECAO POR ITERACAO DO METODO DO GRADIENTE
```

```
print(grad_bissection$vec_k)
```

```
## [[1]]
## [1] 35
##
## [[2]]
## [1] 35
##
## [[3]]
## [1] 35
##
## [[4]]
## [1] 35
##
## [[5]]
## [1] 35
##
## [[6]]
## [1] 35
##
## [[7]]
## [1] 35
##
## [[8]]
## [1] 35
##
```

```
## [[9]]
## [1] 35
```