▶ Plot Functions

▶ Error Handle Functions

▶ Direct Search Functions

INSTITUTO TECNOLÓGICO DE AERONÁUTICA

MP-288 - Exercises on Conjugate Gradient Method

Prof.: Rafael T. L. Ferreira

1) Consider the function $f(\mathbf{x}) = f(x_1, x_2) = 10x_1^4 - 20x_1^2x_2 + 10x_2^2 - 2x_1 + 5. + x_1^2$

Starting from $\mathbf{x}^0 = (-1.5, 3)$, use both the steepest descent and conjugate gradient methods to find a minimum point of $f(\mathbf{x})$. Perform exact line search (use $df/d\alpha = 0$ at $\mathbf{x}_k$ to find all the α^{*k}). Plot a graph showing $f(\mathbf{x})$ and the $\alpha^{*k}\mathbf{d}^k$ directions used over iterations of both methods.

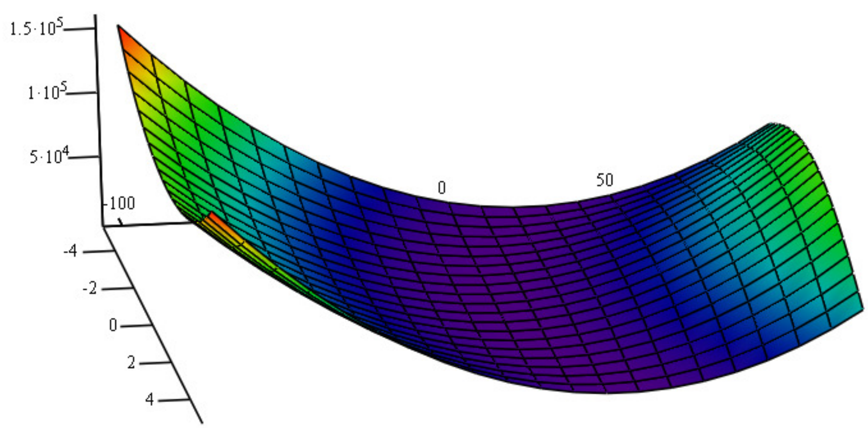
$f(x) := 10\cdot(x_1)^4 - 20\cdot(x_1)^2\cdot x_2 + 10\cdot(x_2)^2 + (x_1)^2 - 2\cdot x_1 + 5$

$f^*(x_1, x_2) := f\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right)$

$zero(x_1, x_2) := 0$

$x_0 := \begin{pmatrix} -1.5 \\ 3 \end{pmatrix}$

Plotando a função :



f*

Solução analítica :

$G(x) := \nabla_x f(x)$

$$G(x) = \begin{bmatrix} 40\cdot(x_1)^3 - 40\cdot x_1\cdot x_2 + 2\cdot x_1 - 2 \\ 20\cdot x_2 - 20\cdot(x_1)^2 \end{bmatrix}$$

Resolvendo o gradiente :

$\text{Extreme} := G(x) = 0 \text{ solve}, x_1, x_2 \ = (1 \quad 1)$

Calculamos o Hessiano :

$$H := \text{Hessian}\left(f, \text{Extreme}^T\right) = \begin{pmatrix} 82 & -40 \\ -40 & 20 \end{pmatrix}$$

Os autovalores do Hessiano são :

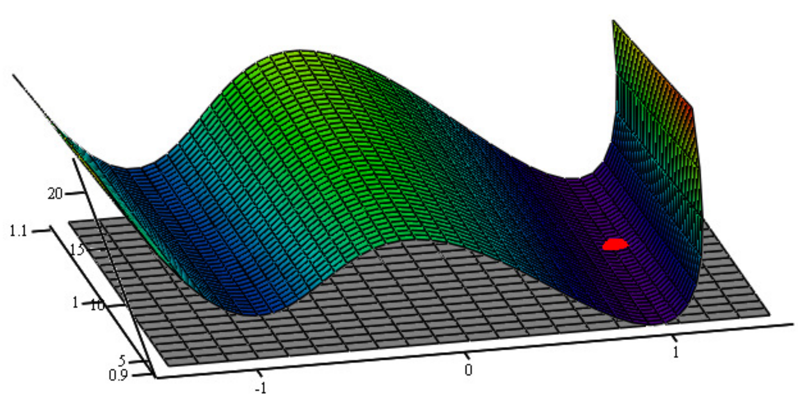
$$\text{eigenvals}(H) = \begin{pmatrix} 101.606 \\ 0.394 \end{pmatrix}$$

Como os autovalores são positivos, podemos dizer que o ponto é um ponto de mínimo :

Plotamos o ponto de mínimo :

$$P := \begin{bmatrix} \text{Extreme}^{\langle 1 \rangle} \\ \text{Extreme}^{\langle 2 \rangle} \\ \left(f\left(\text{Extreme}^T\right)\right) \end{bmatrix}$$

$$f_{min}(x_1, x_2) := f\left(\text{Extreme}^T\right)$$



f^*, P, f_{min}

Método numérico "steepest descent" para encontrar o mínimo da função objetiva :

```
steepest_descent(f, x0) := | trace("...", 0)
                           | trace("...", 0)
                           | trace("...", 0)
                           | trace("Start new |steepest_descent| function....", 0)
                           | "Maximum number of iterations :"
                           | max_iter ← 2000
                           | "Parameters and tolerance :"
                           | ε ← 10^(−5)
                           | "General variables to record values along function :"
                           | Total_iter ← 0
                           | Local_iter ← 0
                           | Trace_Steps ← 0
                           | "At beginning :"
                           | x_α* ← x_0
                           | f_α* ← f(x_α*)
                           | "Trace Value :"
                           | Total_iter ← 1
                           | "Record x alfa* e f alfa* :"
                           | Trace_Steps ← Line_Plot(( x_α* ; f_α* ))
                           | "Start looping :"
                           | for k ∈ 1 .. max_iter
                           |    | "Calculate the gradient :"
                           |    | X_k ← x_α*
                           |    | x_k ← X_k
                           |    | ∇f_k ← ∇_(x_k) f(x_k)
                           |    | "Calculate the length of the gradient :"
                           |    | ||∇fk|| ← |∇f_k|
                           |    | "Check if length is smaller than tolerance :"
                           |    | if ||∇fk|| < ε
                           |    |    | "Stop looping :"
                           |    |    | break
                           |    | otherwise
                           |    |    | "Set the search direction :"
                           |    |    | d_k ← −∇f_k
                           |    |    | "Normalize direction :"
```

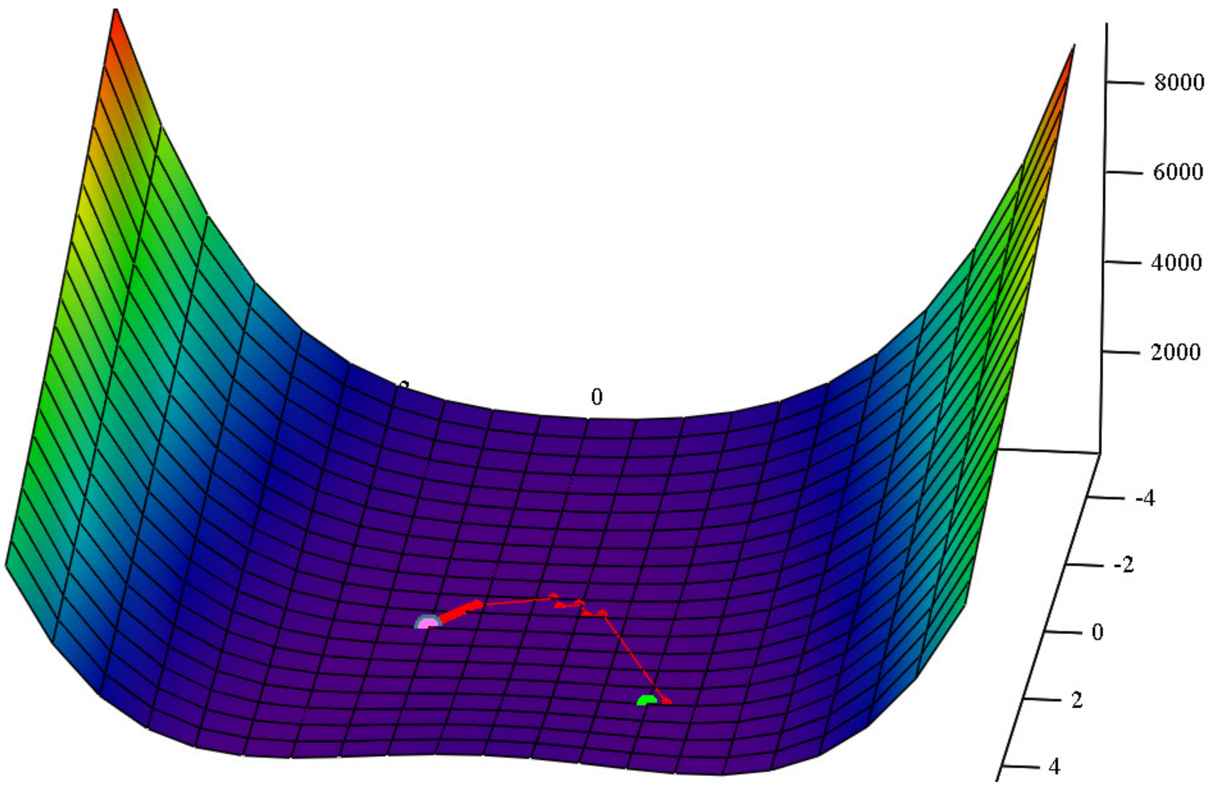
```
d_k ← d_k / |d_k|
"Calculate the step size :"
trace("x.k = {0}", x_k)
trace("d.k = {0}", d_k)
trace("gradiente modulus = {0}", ||∇fk||)
f^(α) ← f(x_k + α·d_k)
f'^(α) ← d/dα f^(α)
α_guess ← 0
on error α*_k ← root(f'^(α_guess), α_guess)
    "Error handle for built-in solver :"
    rc ← ERROR
trace("rc = {0}", rc)
trace("alfa* = {0}", α*_k)
"Check if result is not an error :"
if  rc ≠ ERROR
    "Register X value for alfa* :"
    x_α* ← x_k + α*_k·d_k
    "Register objective function value for X alfa* :"
    f_α* ← f(x_α*)
    "Update the objective function evaluation counter :"
    Total_iter ← Total_iter + 1
    "Record the direction :"
    D^⟨k⟩ ← d_k
    "Record x alfa* e f alfa* :"
    Trace_Steps ← Plot_Augment[Trace_Steps, Line_Plot((x_α*; f_α*))]
otherwise
    "Terminate function :"
    R ← stack(Error(4,1), Total_iter)
    R ← stack_2(R, Trace_Steps)
    return  R
"Check tolerance :"
rc ← ERROR   if  ||∇fk|| > ε
"Function output :"
(x_α*  X  D  rc  Total_iter  Trace_Steps)^T
```

$R := \text{steepest\_descent}(f, x_0)$

$$R = \begin{pmatrix} \{2,1\} \\ \{1270,1\} \\ \{2,1269\} \\ 0 \\ 1.27 \times 10^3 \\ \{3,1\} \end{pmatrix}$$

$R_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$L := R_6$



$$f^*, L, \begin{bmatrix} \left(x_{0_1}\right) \\ \left(x_{0_2}\right) \\ \left(f(x_0)\right) \end{bmatrix}, P, \begin{bmatrix} \left[\left(R_1\right)_1\right] \\ \left[\left(R_1\right)_2\right] \\ \left(f(R_1)\right) \end{bmatrix}$$

$$\begin{pmatrix} x_{\alpha^*} \\ X \\ D \\ rc \\ Total_{iter} \\ Trace_{Steps} \end{pmatrix} := \text{steepest\_descent}(f, x_0)$$

$$\begin{pmatrix} x_{\alpha^*} \\ X \\ D \\ rc \\ Total_{iter} \\ Trace_{Steps} \end{pmatrix} = \begin{pmatrix} \{2,1\} \\ \{1270,1\} \\ \{2,1269\} \\ 0 \\ 1.27 \times 10^3 \\ \{3,1\} \end{pmatrix}$$

$x_{\alpha^*} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

D =

	1	2	3	4	5	6	7	8
1	-0.936	0.351	0.936	0.351	0.936	0.351	0.936	-0.351
2	-0.351	-0.936	0.351	-0.936	0.351	-0.936	0.351	...

```
steepest_descent_num(f, x_0) := | trace("...", 0)
                                 | trace("...", 0)
                                 | trace("...", 0)
                                 | trace("Start new |steepest_descent| function....", 0)
                                 | "Maximum number of iterations :"
                                 | max_iter ← 1000
                                 | "Parameters and tolerance :"
                                 | ε ← 10^(−3)
                                 | I ← 10^(−5)
                                 | δ ← 0.00001
                                 | "General variables to record values along function :"
                                 | Total_iter ← 0
                                 | Local_iter ← 0
                                 | Trace_Local ← 0
                                 | Trace_Global ← 0
                                 | Trace_Steps ← 0
                                 | "At beginning :"
                                 | x_α* ← x_0
                                 | f_α* ← f(x_α*)
                                 | "Trace Value :"
                                 | Total_iter ← 1
                                 | Trace_Global ← Line_Plot((x_α*, f_α*))
                                 | "Record x alfa* e f alfa* :"
                                 | Trace_Steps ← Line_Plot((x_α*, f_α*))
                                 | "Start looping :"
                                 | for k ∈ 1 .. max_iter
                                 |   | "Calculate the gradient :"
                                 |   | X_k ← x_α*
                                 |   | x_k ← X_k
                                 |   | ∇f_k ← ∇_(x_k) f(x_k)
                                 |   | "Calculate the length of the gradient :"
                                 |   | ||∇fk|| ← |∇f_k|
```

```
"Check if length is smaller than tolerance :"
if ||∇fk|| < ε
    "Stop looping :"
    break
otherwise
    "Set the search direction :"
    d_k ← −∇f_k
    "Normalize direction :"
    d_k ← d_k / |d_k|
    "Calculate the step size :"
    trace("x.k = {0}", x_k)
    trace("d.k = {0}", d_k)
    trace("gradiente modulus = {0}", ||∇fk||)
    (α*_k, I', Local_Iter, rc, Trace_Local) ← golden_section(f, x_k, d_k, δ, I)
    trace("line section result = {0}", rc)
    "Trace Value :"
    Trace_Global ← Plot_Augment(Trace_Global, Trace_Local)
    "Check if result is not an error :"
    if rc ≠ ERROR
        "Register X value for alfa* :"
        x_α* ← x_k + α*_k·d_k
        "Register objective function value for X alfa* :"
        f_α* ← f(x_α*)
        "Update the objective function evaluation counter :"
        Total_iter ← Total_iter + Local_iter + 1
        "Record the direction :"
        D^⟨k⟩ ← d_k
        "Record x alfa* e f alfa* :"
        Trace_Steps ← Plot_Augment[Trace_Steps, Line_Plot(((x_α*, f_α*)))]
    otherwise
        "Terminate function :"
        R ← stack(Error(4, 1), Total_iter)
        R ← stack_2(R, Trace_Steps)
        R ← stack_2(R, Trace_Global)
```

```
| | | return R
"Check tolerance :"
rc ← ERROR  if ||∇fk|| > ε
"Function output :"
(x_α*  X  D  rc  Total_iter  Trace_Steps  Trace_Global)^T
```