INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Optimización Numérica | Proyecto 1 – Reporte

Juan Carlos Sigler Alonso Martinez Paulina Carretero

Índice general

1 Marco teórico

En el presente documento discutimos una implementación del algoritmo del conjunto activo para resolver problemas de programación cuadrática con restricciones tanto de igualdad como desigualdad, como en el problema (P) a continuación:

$$\min \frac{1}{2}\vec{x}^\top G\vec{x} + \vec{c}^\top \vec{x} \tag{P}$$

1.1 Algunos comentarios sobre la elección de lenguaje de programación

2 Problemas

Probamos nuestra implementación con tres problemas y presentamos los resultados de acuerdo a lo especificado en la asignación del proyecto.

2.1 Problema 1: Problema chico

$$\min q(x) = (x-1)^2 + (y-2.5)^2$$

Sujeto a

$$\begin{aligned} -x + 2y - 2 &\leq 0 \\ x + 2y - 6 &\leq 0 \\ x - 2y - 2 &\leq 0 \\ -x &\leq 0 \\ -y &\leq 0 \end{aligned}$$

Con $x_0 = (2,0)^\top$ & $W_0 = \{3\}$.

Para este problema, nuestro algoritmo imprime lo siguiente:

```
Rama 1. ||d_k|| = 2.5, q(x) = -1.8125, α=0.5 k = 0
Rama 1. ||d_k|| = 0.9, q(x) = -0.8000000000000007, α=1.6666666666666667
Rama 2. j = 2, μ=0.0
```

Obtenemos que el punto Karush-Kuhn-Tucker (KKT) es:

$$(\vec{x}^\star, \vec{\mu}^\star) = \left(\begin{bmatrix} 1.4 \\ 1.7 \end{bmatrix}, \begin{bmatrix} 0.399 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

2.2 Problema 2: Klee-Minty

$$\begin{array}{ll} \min & \frac{1}{2}\vec{x}^\top G \vec{x} - \sum_{i=1}^{n} x_i \\ \text{sujeto a} & x_1 \leq 1 \\ & 2\sum_{j=1}^{i-1} x_j + x_i \leq 2^i - 1 \quad , i = 2, \dots, n \\ x_1, \dots, x_n \geq 0 & \end{array}$$

Sea $n = 15$. Aplica el método empezando con W_0 un subconjunto aleatorio de 5 entradas de los últimos 10 restricciones de positividad.

Falta documentar:

- W_0
- Numero de iteraciones

Para este problema, nuestro algoritmo imprime lo siguiente:

```
Rama 2. j = 1, μ=0.0
```

y muestra que el óptimo es

$$(\vec{x}^\star, \vec{\mu}^\star) = \left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \right)$$

.

La representación completa del vector $\vec{\mu}$ es

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

.

El valor óptimo de la función objetivo en $\vec{x}_\star$ es 0.0.

Para este problema la rutina `quadprog` de Matlab imprime:

```
>> quadprog(G,c,A,b)

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.


ans =

   1.0e-10 *

    0.2220
    0.1175
    0.0685
    0.0744
    0.1014
```

Nuestra solución coincide con la de Matlab salvo error de redondeo.

2.3 Problema 2: Problema Woodinfe

$$
\begin{array}{ll}
\text{minimizar} & \frac{1}{2}\vec{x}^\top G\vec{x} + \vec{c}^\top\vec{x} \\
\text{sujeto a} & A\vec{x} = \vec{b} \\
 & x_i \geq \ell_i \quad \text{si } \ell_i \text{ es finito,} \\
 & x_i \leq u_i \quad \text{si } u_i \text{ es finito.}
\end{array}
$$

- Definimos $J \subset I$ como sigue:
  - Para $x_j \geq \ell_j$ definimos $\left|g_j\left(x_0\right)\right| \leq 8\varepsilon_m \max\left\{\left|\ell_j\right|, 1\right\} \Longrightarrow j \in J$
  - Para $x_j \leq u_j$ definimos $\left|g_j\left(x_0\right)\right| \leq 8\varepsilon_m \max\left\{\left|u_j\right|, 1\right\} \Longrightarrow j \in J$

Donde ε_m es el épsilon de la máquina `eps(Float64)`.

A continuación presentamos el código que se utilizó para construir el problema y encontrar las restricciones que pertenecen a J. El código se presenta recortado (excluímos el código para obtener los datos del archivo .mat y comentarios aclaratorios) en interés de la brevedad, pero se puede encontarar código completo en el script incluído `script3.ipynb`.

```
n_eq = length(b)
A_eq = problem["A"]
G = I(length(c))

# Como all(isfinite.(l)) == true, se complen todas las cotas inferiores
A = [A_eq; -I(length(l))]
b = [b; l]
```

```julia

mask = isfinite.(u)
A = [A; I(length(u))[mask, :]]
b = [b; u[mask]]
b = b[:]

x_0 = linprog(A, b, n_eq)

# Shorthand para epsilon de maquinaε⊠
 = eps(Float64)
J_l = abs.(l - x_0) .<= 8 * ε⊠ *  max.(abs.(l), ones(length(l)))

J_u = abs.(x_0 - u) .<= 8 * ε⊠ *  max.(abs.(u), ones(length(u)))
J_u = J_u[isfinite.(u)]

W_0 = [trues(n_eq); J_l; J_u]
```

De el cómputo anterior obtenemos W_0 que documentamos a continuación, incluídas las 138 entradas.

```julia
julia> println(W_0)
Bool[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
     1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

El punto óptimo es:

```julia
julia> println(x_star)
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0]
```

Por documentar:

- Número de iteraciones

3 Conclusiones (?)