OPTIMIZACIÓN NUMÉRICA I
PROYECTO I

Condiciones para entregar el proyecto

1. Cada equipo debe tener entre de 2 o 3 miembros.

2. Cada equipo debe **registrarse** con un correo electrónico (a mi).
   Esto es para que yo pueda configurar la entrega vía CANVAS.

3. Cada equipo debe implementar el método del conjunto activo descrito en las clases (10 – 12).
   Obligatorio:
   - la función debe aceptar un parámetro que sirve para limitar el máximo de iteraciones (default value 100).
   - En la decisión de tomar Rama 1 o Rama 2, considerar $\vec{\boldsymbol{d}}_k = \vec{\mathbf{0}}$ si $\left\|\vec{\boldsymbol{d}}_k\right\|_\infty < 10^{-9}$.
   - Cada vez que el algoritmo pasa por Rama 1 imprimir lo siguiente **en una linea** (usar el parámetro `end` en `print`):
     - "Rama 1"
     - la longitud $\left\|\vec{\boldsymbol{d}}_k\right\|_\infty$,
     - el nuevo valor objetivo $q(\vec{\boldsymbol{x}}_{k+1})$,
     - el valor $\check{\alpha}$
     - y si $\check{\alpha} < 1$ entonces también el índice j que entra a W_{k+1}
   - Cada vez que el algoritmo pasa por Rama 2 imprimir
     - "Rama 2"
     - En caso que existe $\mu_j < 0$ imprimir los valores j y μ_j .

4. documentar los resultados (en un PDF) para los problemas descritos abajo.
   Obligatorio: Para cada problema hay que entregar un *script* (puede ser en formato `.py`, o `.ipynb` en Python, compatible con la versión Python 3.8.8 usada en anaconda). Los scripts deben reproducir los resultados que incluyeron en la documentación (documento PDF). Pido 3 *scripts*, puesto que son 3 problemas. Ademas, hay que entregar el método en un archivo `mActiveSet.py` y ese se debe importar y aplicar en sus *scripts* para resolver los problemas. Esto se hace como en el ejemplo que les comparto.

Otras condiciones:

- Fecha límite de entrega: Ver CANVAS (en tareas).
- Entregar todo en CANVAS, los (*scripts* y el programa) en un archivo `zip` y **por separado** el documento PDF que documenta los resultados. También, deben incluir el archivo que contiene los datos (A, b, c, L, U) para el ultimo el problema y el código que yo les comparto.

Dr. Andreas Wachtel, Otoño 2021.

1. Problemas

Usen su implementación del método del conjunto activo para resolver los siguientes problemas:

P1) *Un problema chico:*

El problema 6a de los Ejercicios T2.1. Deben documentar

- todo el output (descrito en punto 3)
- y adicionalmente el numero de iteraciones y valor óptimo obtenido.

Para reproducir estos resultados les pido un *script* llamado `script1.py` o `script1.ipynb`.

P2) *El problema de Klee-Minty:*

Sea $G = 10^{-4} I$ y el problema cuadrático de Klee-Minty dado por

$$\begin{aligned} \text{minimizar} \quad & \frac{1}{2}\vec{\boldsymbol{x}}^{\top} G \vec{\boldsymbol{x}} - \sum_{i=1}^{n} x_i \\ \text{sujeto a} \quad & x_1 \leq 1 \\ & 2\sum_{j=1}^{i-1} x_j + x_i \leq 2^i - 1 \quad , \; i = 2, \ldots, n \\ & x_1 \, , \; \ldots \, , \; x_n \geq 0 \, . \end{aligned}$$

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

- W_0
- el numero de iteraciones
- el valor óptimo obtenido
- el valor óptimo del método `quadprog` del ambiente (MatLab u Octave).

Para reproducir estos resultados les pido un *script* llamado `script2.py` o `script2.ipynb`.

Consejo: Antes de entregar, pueden verificar el valor óptimo en MatLab u Octave con el método `quadprog`. (En Octave, eso requiere importar el paquete `optim` con `pkg load optim`.)

P3) Primero, consiguen el problema *woodinfe* en formato `.mat` de la liga (o del proyecto)

`https://www.cise.ufl.edu/research/sparse/matrices/LPnetlib/`

Ese problema tiene la siguiente estructura

$$\begin{aligned} \text{minimizar} \quad & \vec{c}^{\top}\vec{x} \\ \text{sujeto a} \quad & A\vec{x} = \vec{b} \\ & \vec{\ell} \le \vec{x} \le \vec{u} \end{aligned} \quad .$$

Los parámetros $\vec{c}, A, \vec{b}, \vec{\ell}, \vec{u}$ se pueden obtener en Python con el código que les comparto en CANVAS. Tomando esos datos y $G = I \in \mathbb{R}^{n\times n}$, define el siguiente problema

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

Esto se puede hacer en una función (separada). Una vez definido, ese problema se debe resolver con su algoritmo: Para empezar

- Encuentre x_0 con programación lineal (`linprog`) solo usando $A, \vec{b}, \vec{\ell}, \vec{u}$.
- Definimos $J \subset I$ como sigue:
  - Para $x_j \ge \ell_j$ definimos $|g_j(x_0)| \le 8\,\varepsilon_m\,\text{máx}\,\{|\ell_j|, 1\} \implies j \in J$
  - Para $x_j \le u_j$ definimos $|g_j(x_0)| \le 8\,\varepsilon_m\,\text{máx}\,\{|u_j|, 1\} \implies j \in J$

  Donde ε_m es el épsilon de la máquina (`numpy.finfo(float).eps`).
- Define $W_0 \coloneqq \mathcal{E} \cup \{j\}$ para un índice aleatorio $j \in J$.

En el PDF se deben documentar

- W_0
- el numero de iteraciones
- el valor óptimo obtenido.

Para reproducir estos resultados les pido un *script* llamado `script3.py` o `script3.ipynb`.

Consejo: Antes de entregar, pueden verificar el valor óptimo en MatLab u Octave con el método `quadprog`. (En Octave, eso requiere importar el paquete `optim` con `pkg load optim`.)

En Matlab u Octave: Después, de la instrucción `load lpi_woodinfe.mat` tendrán acceso a $A, \vec{b}, \vec{c}, \vec{\ell}, \vec{u}$ con las instrucciones

`Problem.A`, `Problem.b`, `Problem.aux.c`, `Problem.aux.lo` y `Problem.aux.hi`

donde `Problem.aux.lo` da el vector $\vec{\ell}$ y `Problem.aux.hi` da el vector $\vec{u}$.