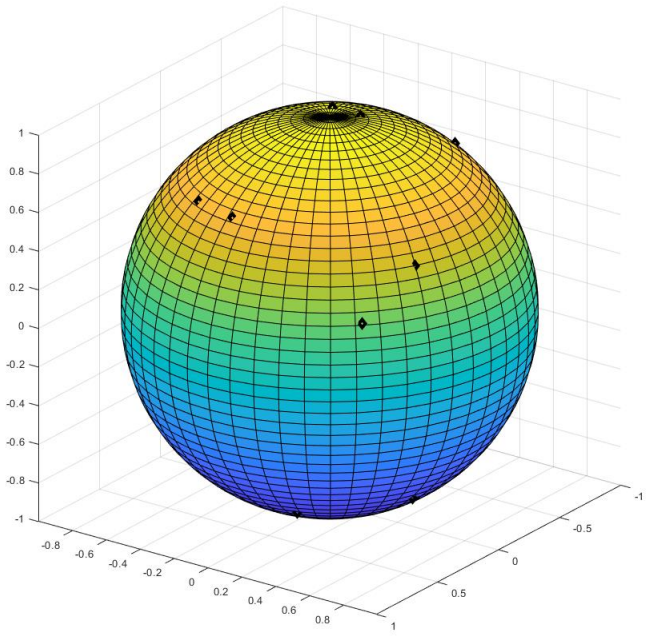
Optimización Numérica
Proyecto
Programación Cuadrática Sucesiva
Dr. Zeferino Parada García

1 Introducción

Colocar $(np+1)$ puntos en la esfera unitaria en $\mathbb{R}^3$ que minimice la suma de las repulsiones entre cualquier par de puntos, se modela como :

Sean $x^k = (x_{k1},\ x_{k2},\ x_{k3})^T,\ k = 0,\ 1,\ 2,\ \ldots, np,$, las coordenadas cartesianas del punto número k, se define el problema,

$$\begin{aligned} \textbf{Min} \quad & \sum_{k=0}^{np} \sum_{k<j} \frac{1}{[(x_{k1}-x_{j1})^2+(x_{k2}-x_{j2})^2+(x_{k3}-x_{j3})^2]^{-1/2}} \\ \textbf{Sujeto a} \quad & \\ & x_{k1}^2 + x_{k2}^2 + x_{k3}^2 = 1,\ \ k = 1,\ \ldots,\ np \end{aligned} \tag{1}$$



El número de variables es $n = 3 * (np + 1)$ y el número de restricciones no lineales es $m = np + 1$

Para evitar diferentes soluciones, anclamos el primer punto en, $x^0 = (1,\ 0,\ 0)$. El modelo ahora tiene $n = 3 * np$ variables y $m = np$ restricciones.

2 Proyecto

Resolver el modelo (1) con **Python** para 21 puntos en la esfera en las dos versiones:

V1.- Con el pograma, hecho en laboratorio y completado por ustedes, **pcs_global.py**, que es programación cuadrática sucesiva con búsqueda de línea. Aquí debe usar las funciones de clase, **gradiente.py** y **jacobiana.py**.

V2.- Con el módulo de optimización en **scipy** y la función **minimize.py**. Ver documentación en:
https://docs.scipy.org/doc/scipy/tutorial/optimize.html

3 Programas en Python

1. Programe la función objetivo:

```
def f_electron(x)
    #
    # x es un vector columna de dimensión 3 * np.
    # El punto #, k,  x^k que quiere colocarse en la esfera,
    # tiene coordenadas (x_{3*k}, x_{3k+1}, x_{3*k+2}).
    # Return:
    # f_x valor de la función a minimizar.
    #-----------------------------------------
```

2. Programe la función de restricciones:

```
def h_esfera(x)
    # versión para usar en programación cuadrática sucesiva
    # con búsqueda de línea.
    # x es un vector columna de dimensión 3 * np.
    # El punto #, k,  x^k que quiere colocarse en la esfera,
    # tiene coordenadas (x[3 * k], x[3k + 1], x[3 * k + 2]).
    # Return h_x .
```

h_x es el vector de dimensión np que indica
si el punto x^k está cerca o no de la esfera.
$h\_x[k] = x[3*k]**(2) + x[3k+1]**(2) + x[3*k+2]**(2) - 1$
#----------

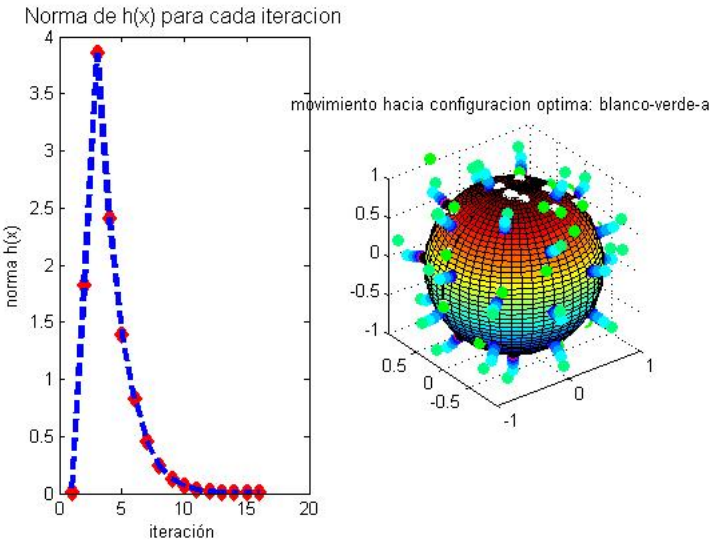
3. Resuelva el problema para $np = 21$, uno de los puntos está anclado a $(1,\ 0,\ 0)^T$.

4. Se requiere un punto inicial de $n = 3 * np = 60$ variables. Usted debe proponer el vector inicial de 60 puntos.

5. Escriba una tabla, por cada programa **pcs_global.py** y **minimize.py** con cada uno de valores obtenidos:

np	$\|CNPO\|$	$f(x^*)$	cpu time

6. Grafique la esfera y los puntos solución sobre la misma en las dos versiones.



4 Entrega del Proyecto

Equipos de a lo más tres personas.

1. Programas con los nombres y clave única de los integrantes del equipo y documentación de los códigos.

2. Archivo, **pcs_esfera.py**, con la solución obtenida por **pcs_global.py** . También con la graficación de la solución.

3. Archivo, **pcs_minimize.py**, con la solución obtenida por **minimize.py** . También con la graficación de la solución.

4. Enviar todos los programas en **Python** para que sus proyectos trabajen. El Profesor no modificará sus programas para que estos funcionen.

5. Entrega por Canvas, **miércoles 15 de mayo de 2024 a las 18:00 horas**. Una entrega por cada equipo. Carpeta empaquetada, **ProyOptNUm2.zip** o **ProyOptNUm2.rar**.

5 Calificación

Concepto	**Comentarios y Sugerencias**	**Ponderaión**	**Calificación**
Parte Teórica		**20**	
Programación		**30**	
Resultado Numérico		**30**	
Graficación		**10**	
Presentación		**10**	
TOTAL de puntos		**100**	