Análisis Numérico
Laboratorio Parcial 2

Igor Andruskiewitsch

June 22, 2020

Cada ejercicio tiene su propio archivo, donde se encuentran las funciones pedidas. Además, cada archivo puede ejecutarse como un script independiente, donde se prueban las funciones pedidas, se imprimen los distintos resultados y se plotean los gráficos.

1 Ejercicio 1)

1.1 (a)

Para el ejercicio (a), se utiliza directamente la clase `CubicSpline` de `scipy`, sin extrapolación ya que no es necesaria.

1.2 (b)

Al tener la necesidad de modificar la regla del trapecio compuesta para que no sea necesario contar con puntos equidistantes, se utilizó la regla del trapecio común:

$$\int_a^b f(x)dx \approx \frac{(b-a)}{2}(f(a)+f(b))$$

y la regla que indica que:

$$\int_a^b f(x)dx + \int_b^c f(x)dx = \int_a^c f(x)dx$$

Luego, se calculan todas las integrales entre los puntos y se suma el resultado para lograr una aproximación, sin utilizar la fórmula de de la regla del trapecio ocompuesta.

1.3 (c)

En el ejercicio (c), se utiliza la función `trapecio_adaptativo` del ej (b) y se calcula:

$$\int_0^t vs(x)dx$$

Para cada t en ts, ya que sabemos que la posición de la particula para cada t está dada por esta integral.

2 Ejercicio 2)

Al correr el archivo `ejercicio_2.py`, se prueban las funciones `gseidel` y `sor`, utilizando un sistema conocido, para verificar el resultado:

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix} \begin{pmatrix} 1 \\ -3 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 12 \\ 34 \\ 27 \\ -38 \end{pmatrix}$$

2.1 (a)

Para el punto (a), utilizamos que para el paso iterativo de **Gauss-Seidel** se usa el vector $x^{(n)}$ tq:

$$Mx^{(n)} = b - Nx^{(n-1)}$$

Como sabemos que $M = L + D$, tenemos garantizado que esta matriz es triangular inferior y podemos utilizar `soltrinf` para resolver este sistema.

2.2 (b)

Aquí directamente se usa el paso iterativo dado en el ejercicio. Con el mismo concepto dicho anteriormente.

3 Ejercicio 3)

3.1 (a)

Se plantea el problema como:

$$\begin{array}{ll} \text{maximizar} & \text{z} = 500\text{x} + 300\text{y} \\ \text{sujeto a} & 2\text{x} + \text{y}\ 40 \leq 40 \\ & \text{x} + 2\text{y}\ 50 \leq 50 \end{array}$$

Luego se utiliza el módulo `scipy.optimize` para resolver este problema.

3.2 (b)

Para saber la respuesta a esta pregunta, se plantea un nuevo problema, basicamente el mismo que el anterior, pero agregando una variable h que representa la cantidad de horas trabajadas por nuestro empleado.

$$\begin{array}{ll} \text{maximizar} & \text{z} = 500\text{x} + 300\text{y} - 200\text{h} \\ \text{sujeto a} & 2\text{x} + \text{y}\ 40 \leq 40 + \text{h} \\ & \text{x} + 2\text{y}\ 50 \leq 50 \\ & \text{h} \leq 40 \end{array}$$

Sabemos que si el valor de z maximizado es mayor que el del ejercicio (a), entonces conviene contratar un empleado, la cantidad de horas convenientes está dada por el valor de h que maximiza este nuevo z.