Proyecto Final de Análisis Numérico y Computación Científica

INTEGRANTES: Sergio Andrés García y Juan José Ruiz Triana

A continuación, se presenta la explicación del código de nuestro proyecto final lo cual consta de 4 funciones principales:

- Jacobi: esta función implementa el método de Jacobi para resolver un sistema de ecuaciones lineales, la cual inicializa un vector solución inicial con ceros y un error inicial que iniciamos como infinito. Mientras que el error inicial el cual se actualiza como la norma del vector de la diferencia entre la nueva solución y la anterior sea mayor a la tolerancia dada por el usuario calcula una nueva aproximación siguiendo la formula vista en clase y cuando el error inicial sea menor a la tolerancia hemos encontrado la solución aproximada.

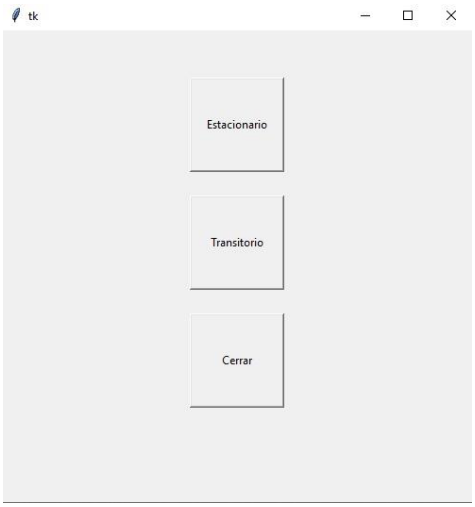
- gauss_elimination: esta función emplea el método de eliminación gaussiana para resolver un sistema de ecuaciones lineales de la forma Ax = b. Primero se comprueba que la matriz A sea cuadrada y que coincida el tamaño con el del vector b para luego crear la matriz aumentada de A y b, el siguiente paso es según la opción de pivoteo dado por el usuario ya sea escalado, parcial o ninguna llevar la matriz a la forma triangular superior para asi llegar al último paso que es la sustitución hacia atrás y llegar a la solución del sistema lineal.

- sol_estacionaria: esta función resuelve un problema de difusión estacionaria de un disipador de calor utilizando el método de las diferencias finitas y el método de eliminación gaussiana(gauss_elimination). Inicia calculando el tamaño de los intervalos entre los nodos, inicializa dos matrices para poder representar el sistema de ecuaciones, crea el termino común de la discretización para este problema(β) y llena la matriz anteriormente creada con coeficientes según la discretización del problema en este caso la diagonal principal y las demás posiciones las llena de 1. Ahora según el caso escogido por el usuario se configuran las condiciones de contorno, si es para el primer caso se establece una temperatura fija en el primer nodo y se solicita al usuario ingresar la temperatura en el último nodo para luego crear la matriz que contiene los coeficientes y las condiciones de contorno y el vector b que contine la temperatura inicial fija y la temperatura preguntada al usuario, si el para el segundo caso se fija la temperatura en el primer nodo y se iguala la derivada en el último nodo a cero, para el último caso iguala la derivada en el primer nodo a cero y se fija la temperatura en el último nodo y en este caso ya no se necesita el β. Luego de creadas las matrices dependiendo el caso se resuelve el sistema lineal con la función gauss_elimination retornando las temperaturas estimadas en cada nodo.

- sol_transitoria: esta función resuelve un problema de difusión de un disipador de calor utilizando el método de las diferencias finitas en el tiempo y el método iterativo de Jacobi para la discretización temporal. La función empieza calculando los pasos espaciales y temporales según el número de nodos, el tiempo y la longitud L, luego se guardan los coeficientes β_1 y β_2 utilizados en la discretización de la ecuación de difusión en diferencias finitas en el tiempo. Como siguiente paso se crea una matriz A que representa el sistema de ecuaciones resultante de la discretización en diferencias finitas en el tiempo y además se inicializa un vector de ceros el cual es la aproximación inicial para el método de Jacobi. Ahora dependiendo el caso se crea el sistema de ecuaciones, si es para el primer caso se establece que la temperatura en los nodos extremos de la barra es fija. Se solicita al usuario la temperatura en el último nodo y se establecen las condiciones de contorno en la matriz y la matriz de temperaturas, en el segundo caso se fija la temperatura en el primer nodo y se iguala la derivada en el último nodo a cero, para el último caso se iguala la derivada en el primer nodo a cero y se fija la temperatura en el último nodo. Luego de se soluciona el sistema de ecuaciones empleando el método de Jacobi el cual nos retorna un vector la matriz de temperaturas que contiene las temperaturas estimadas en cada nodo para cada paso de tiempo.

A continuación, adjuntamos los pantallazos de las pruebas realizadas:

Interfaz:

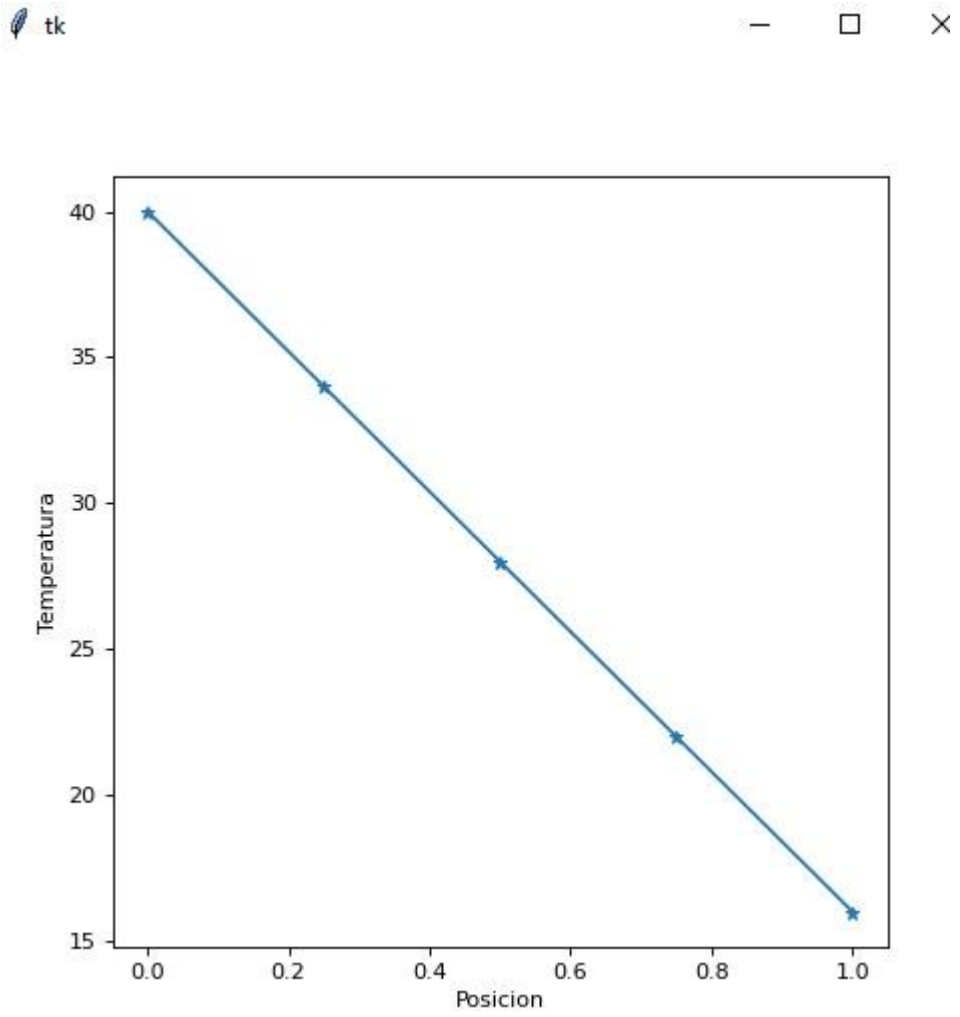


Datos ingresados para el caso estacionario:

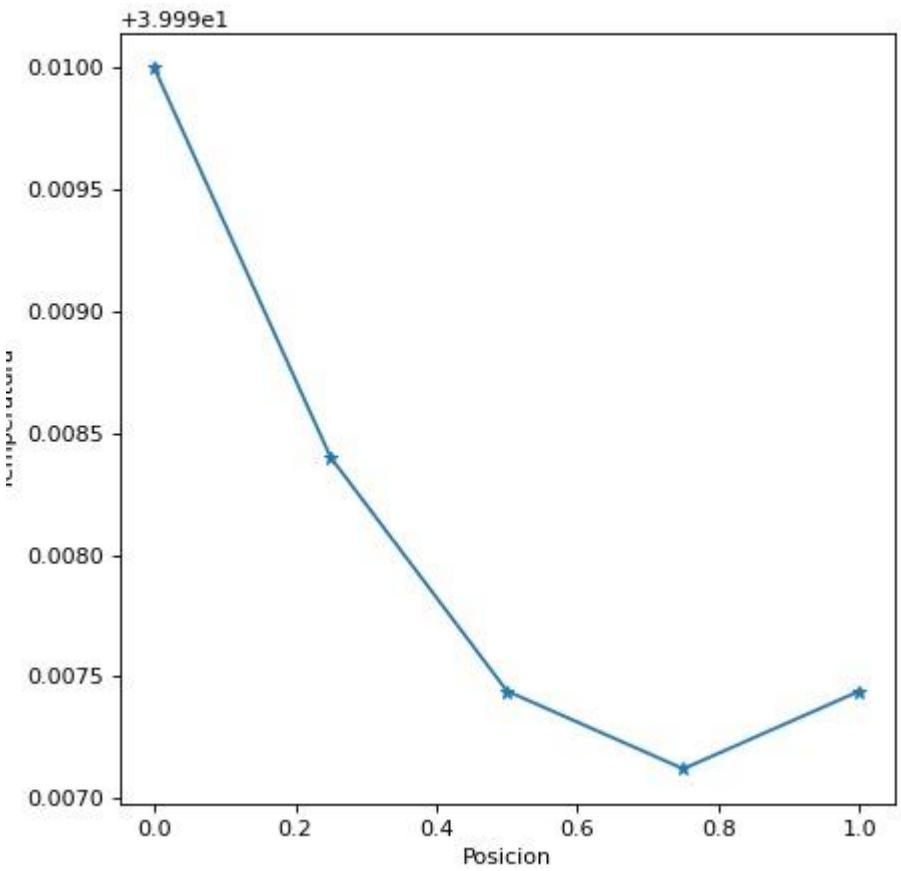
```
L = 1
nodos = 5
alpha = 0.02
T_contorno = 40
T_contorno2 = 10
opcion = 1
```

Resultados:

Caso 1:



Caso 2:



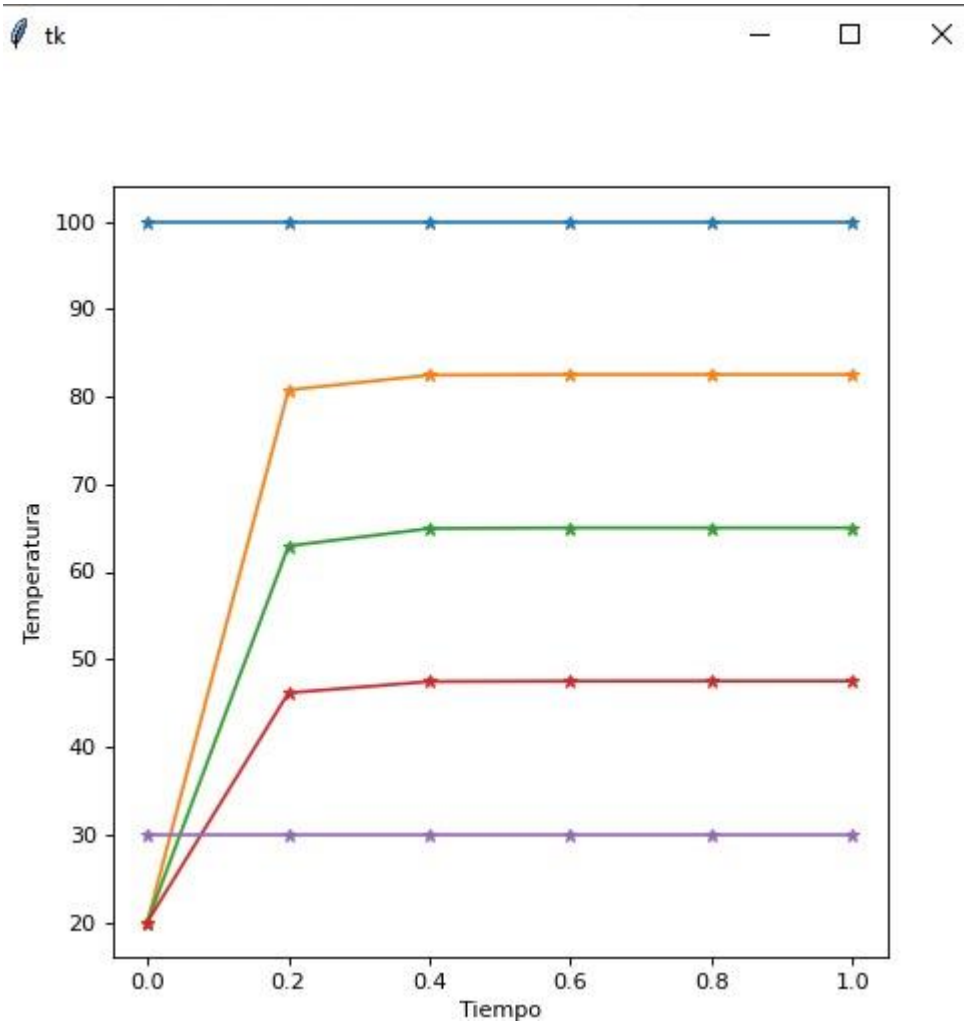
Datos ingresados para el caso transitorio:

```
transitorio

L = 1
tiempo = 10
nodos_espacio = 5
nodos_tiempo = 6
alpha = 0.02
T_contorno = 100
T_inicial = 20
T_ultimo nodo = 30
```

Resultados:

Caso 1



Caso 2:

Universidad del Rosario

tk