Instituto Tecnológico de Costa Rica
Ingeniería en Computadores
Puntaje Total: 100 puntos

CE-3102: Análisis Numéricos para Ingeniería
Semestre: II - 2021
Valor Porcentual: 15 %

Tarea 2

Instrucciones generales

- La tarea se realiza en grupos de máximo 4 personas. Cada grupo debe escribir el nombre de los integrantes del grupo en la siguiente dirección electrónica:

  `https://docs.google.com/spreadsheets/d/1bxDpqztI9lU70-fno_LzAcLe7MHKYJUZ`

- Todos los archivos de esta tarea se encuentran en la carpeta de One Drive del curso.

- Los archivos computacionales implementados en GNU Octave, Pyhton y C++ deben estar correctamente comentados. Por cada archivo que no este documentado correctamente, se restaran 5 puntos de la nota final. **Si alguna función o archivo computacional está incompleto o genera error al momento de compilar, entonces pierde el 75% del puntaje de la pregunta asignada.**

Parte 1: Método de Jacobi en Paralelo con GNU Octave

Descripción General

Durante este semestre, se estudio el método de Jacobi para resolver un sistemas de ecuaciones lineal $Ax = b$, donde $A \in \mathbb{R}^{m \times m}$ es una matriz diagonalmente dominante. La implementación computacional se realizó utilizando una representación matricial. Sin embargo, existe una manera de realizar la implementación del método de Jacobi evitando dicha representación matricial. Si $x^{(k)} = [x_1^{(k)}\ x_2^{(k)}\ ...\ x_m^{(k)}]^T$ es la aproximación generada por el método de Jacobi en la $k-$ésima iteración, entonces la $k+1-$ésima iteración $x^{(k+1)} = [x_1^{(k+1)}\ x_2^{(k+1)}\ ...\ x_m^{(k+1)}]^T$ se puede calcular a través de la siguiente fórmula:

$$x_i^{(k+1)} = \frac{1}{A_{i,i}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{m} A_{i,j} x_j^{(k)} \right), \tag{1}$$

para cada $i = 1, 2, ..., m$. Note que para calcular cada $x_i^{(k+1)}$ solo se necesitan los valores de la iteración anterior $x^{(k)}$. Por lo tanto, se puede realizar una implementación en paralelo, calculando cada $x_i^{(k+1)}$ en un procesador (núcleo). Una representación gráfica de lo explicado anteriormente se presenta a continuación:

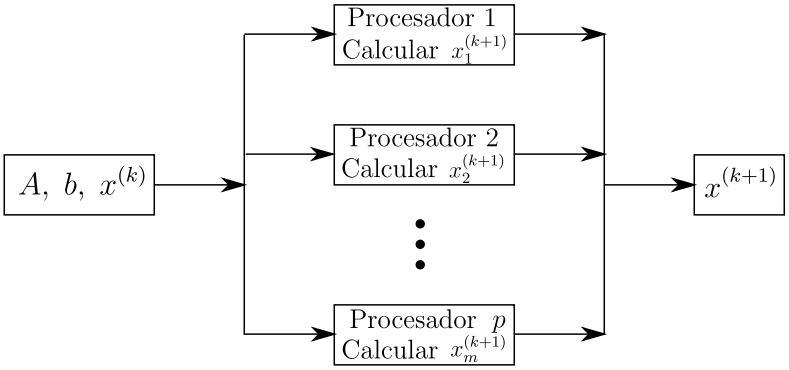

Figura 1: Representación Gráfica de la Implementación en Paralelo del Método de Jacobi

Preguntas

1. **[4 puntos]** Implemente en GNU Octave una función `A=tridiagonal(p,q,m)`, donde los parámetros iniciales son:

   - $p = [p_2\ p_3\ ...\ p_m]^T \in \mathbb{R}^{m-1}$
   - $q = [q_1\ q_2\ ...\ q_{m-1}]^T \in \mathbb{R}^{m-1}$
   - m es un número entero positivo mayor o igual a 3.

   El parámetro de salida A es una matriz tridiagonal de tamaño $m \times m$ tal que

$$A = \begin{pmatrix} 2q_1 & q_1 & & & & \\ p_2 & 2(p_2+q_2) & q_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & p_{m-1} & 2(p_{m-1}+q_{m-1}) & q_{m-1} & \\ & & & p_m & 2p_m & \end{pmatrix}.$$

2. **[7 puntos]** Realice una implementación en GNU Octave del método de Jacobi basado en la ecuación (1) de manera secuencial (es decir, no paralelo) para resolver el sistema $Ax = b$, donde A es una matriz generada por la función `tridiagonal` con parámetros `p=q=[1:0.1:25]`, `m=242`, y `b=ones(m,1)`. Utilizarán como valor inicial el vector $x^{(0)} =$ `zeros(242,1)`, tolerancia $tol = 10^{-5}$, criterio de parada $\|Ax^{(k)} - b\|_2 < tol$ e iteraciones máximas igual a 1000. Deben guardar dicha implementación en un archivo con nombre `parte1_p2.m`.

3. **[25 puntos]** Esta pregunta consiste en la implementación en paralelo del método de Jacobi.

   (a) **[5 puntos]** En un documento escrito, con nombre `parte1_p3.pdf`, expliquen como se puede realizar la implementación en paralelo del método de Jacobi, según se presenta en la ecuación (1). Dicha explicación es libre: pueden usar diagramas de flujos, pseudocódigos, explicar paso por paso la implementación en paralelo, o desarrollar una infografías del método. **Lo importante es que dicha explicación sea clara.**

   (b) **[20 puntos]** Realice una implementación **en paralelo** en GNU Octave del método de Jacobi basado en la ecuación (1) para resolver el sistema $Ax = b$, donde A es una matriz generada por la función `tridiagonal` con parámetros `p=q=[1:0.1:25]`, `m=242`, y `b=ones(m,1)`. Utilizarán como valor inicial el vector $x^{(0)} =$ `zeros(242,1)`, tolerancia $tol = 10^{-5}$, criterio de parada $\|Ax^{(k)} - b\| < tol$ e iteraciones máximas igual a 1000. Deben guardar dicha implementación en un archivo con nombre `parte1_p3.m`. La implementación en paralelo debe realizarse usando el comando `pararrayfun` del paquete `parallel` de GNU Octave, **utilizando el máximo número de procesadores disponibles en su computadora (número de núcleos o el doble con procesadores lógicos)**. Para esto, deben leer la explicación y los ejemplos que se encuentran en los siguientes enlaces:

`https://octave.sourceforge.io/parallel/overview.html`

`https://wiki.octave.org/Parallel_package`

Deben guardar dicha implementación en un archivo con nombre `parte1_p3.m`.

4. **[4 puntos]** Para medir la aceleración S_p de un método implementado en paralelo, se utiliza la fórmula

$$S_p = \frac{T_s}{T_p},$$

donde T_s es el tiempo de ejecución del método sin paralelismo y T_p es el tiempo de ejecución del método con paralelismo, usando p procesadores. Calcule la aceleración de la implementación en paralelo del método de Jacobi, utilizando $p = 1, 2, ..., p_{max}$, donde p_{max} es la cantidad máxima de procesadores disponibles de su computadora. Deben guardar dicha implementación en un archivo con nombre `parte1_p4.m`.

Parte 2: Solución de Sistemas de Ecuaciones No Lineales en Pyhton

Descripción General

Esta parte de la tarea consiste en dar solución a un sistema de m ecuaciones no lineales con m incógnitas. El problema se puede representar matemáticamente como

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}_m, \tag{2}$$

donde $\mathbf{0}_m = (0, 0, ..., 0)^T \in \mathbb{R}^m$,

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} f_1(x_1, x_2, ..., x_m) \\ f_2(x_1, x_2, ..., x_m) \\ \vdots \\ f_m(x_1, x_2, ..., x_m) \end{pmatrix},$$

y cada $f_i(x_1, x_2, ..., x_m) : \mathbb{R}^m \to \mathbb{R}$ es una función no lineal. Un método para resolver sistemas de ecuaciones no lineales es el método de Newton-Raphson. En los archivos `Método Newton-Raphson - 1.pdf` y `Método Newton-Raphson - 2.pdf` se encuentra una explicación de dichos métodos.

Preguntas

1. **[5 puntos]** En un documento con nombre `parte2_p1.pdf`, realizar un resumen del método iterativo de Newton-Raphson. Dicho resumen debe incluir el problema a resolver, formulación matemática del método, además de los valores iniciales y pseudocódigo.

2. **[20 puntos]** Implemente en Python el método de Newton-Raphson a través de la función con nombre `newton_raphson`. Los parámetros iniciales de la función son:

   - Vector inicial $\mathbf{x}^{(0)} \in \mathbb{R}^m$.
   - Vector *string* $\mathbf{f} = (f_1, f_2, ..., f_m)$, donde cada $f_i(x_1, x_2, ..., x_m) : \mathbb{R}^m \to \mathbb{R}$ es una función no lineal.
   - Vector *string* $\mathbf{x} = (x_1, x_2, ..., x_m)$, que representan las variables a utilizar.
   - $tol > 0$, que representa la tolerancia para el criterio $\|\mathbf{F}(\mathbf{x}^{(k)})\|_2 < tol$.

- $iterMax > 0$, que representa las cantidades máximas de iteraciones.

Los parámetros de salida son:

- Vector $\mathbf{x}^{(k)} \in \mathbb{R}^m$, que aproxima a la solución del Problema (1).
- Cantidad de iteraciones k.
- Error e_k, que cumple la desigualdad $e_k = \|\mathbf{F}(\mathbf{x}^{(k)})\|_2 < tol$.
- Una gráfica de iteraciones versus errores.

Nota: El pseudocódigo y la implementación computacional no debe realizar el cálculo de la inversa del Jacobiano, sino resolver un sistema de ecuaciones lineales. Busque la manera más óptima para resolver sistemas de ecuaciones lineales en Python .

3. **[5 puntos]** Utilizar la función `newton_raphson`, aproxime el cero de las funciones que se encuentran en la Sección 4 del documento `ejemplos.pdf`, **excepto la función del punto (f)**. El nombre del archivo que contiene la solución de todos los ejercicios debe ser `parte2_p3.py`.

Parte 3: Método Iterativo para Aproxima la Pseudoinversa en C++

Descripción General

En el artículo *Generalized Inverses Estimations by Means of Iterative Methods with Memory* desarrollado por Santiago Artidiello, Alicia Cordero, Juan R. Torregrosa y María P. Vassileva, presentan un método iterativo para aproximar la pseudoinversa de una matriz $A \in \mathbb{R}^{m\times n}$, el cual se deduce del método de la secante para aproximar un cero de la función no lineal. Dicho método iterativo es el siguiente:

$$X_{k+1} = X_{k-1} + X_k - X_{k-1}AX_k, \tag{3}$$

para $k = 1, 2, ...$, con valores iniciales $X_0 = \alpha_1 A^T$ y $X_1 = \alpha_2 A^T$, donde α_1 y α_2 son dos constante no negativas.

Preguntas

1. **[15 puntos]** Implemente en C++ el método iterativo de la ecuación (3) para aproximar la pseudoinversa de la matriz $A \in \mathbb{R}^{45\times 30}$, donde $A_{i,j} = i^2 + j^2$, para $i = 1, 2, ..., 45$ y $j = 1, 2, ..., 30$. Utilice $\alpha_0 = 5 \times 10^{-10}$ y $\alpha_1 = 2 \times 10^{-11}$. El criterio de parada debe ser $\|AX_kA - A\|_{fr} < 10^{-5}$, donde $\|\cdot\|_{fr}$ es la norma de Frobenius. Dicha implementación debe realizarse usando la librería **Armadillo**. La documentación completa de esta librería se encuentra en los siguientes enlaces:

   `http://arma.sourceforge.net/docs.html`

   `http://arma.sourceforge.net/armadillo_joss_2016.pdf`

   El nombre del archivo que contiene dicha implementación debe ser `parte3_p1.cpp`.

2. **[15 puntos]** Investigue un problema aplicado a la vida real que necesite el cálculo de la pseudoinversa. Detalles de la parte escrita y de programación son los siguientes:

   - **Cada grupo debe utilizar un problema diferente**. Indicar el título del problema y su referencia bibliográfica respectiva en el archivo donde indicaron los integrantes del grupo (Ver enlace en **Instrucciones Generales**).

- La parte escrita debe indicar el problema a resolver, explicado en forma simple y resumida. Deben incluir la referencia bibliográfica de donde se seleccionó dicho problema. El nombre del archivo debe ser `parte4_p2.pdf`.
- La parte escrita debe indicar el problema matemático, además de su respectiva solución.
- Solución del problema usando la implementación computacional de la pregunta 1 de esta parte en **C++**. El nombre del archivo debe ser `parte3_p2.cpp`.

Información de la Entrega

- **Fecha y hora límite**: Lunes 11 de Octubre del 2021 a las 11:59 pm.
- Los documentos deben estar en una carpeta principal con nombre `Tarea 2 - Grupo #`, donde `#` es el número de cada grupo. Dentro de esta carpeta debe existir tres carpetas con nombres `Parte 1`, `Parte 2` y `Parte 3` . En cada una de estas carpetas estarán todos los archivos necesarios para el desarrollo de las preguntas mencionadas anteriormente.
- Deben enviar la carpeta `Tarea 2 - Grupo #` en formato `zip` al correo `jusoto@tec.ac.cr`, con el encabezado `Entrega Tarea 2 - Grupo # - ANPI`. En el cuerpo del correo deben indicar el nombre completo de los miembros del grupo.
- **OBSERVACIÓN IMPORTANTE: Las entregas tardías se penalizarán con una reducción del 25% de la nota máxima por día de atraso. A las tareas que excedan el plazo de entrega en 3 días o más después de la fecha límite, se les asignará la nota de 0**.

Defensa

- Cada grupo debe defender esta tarea frente al profesor. Para eso deben seleccionar un horario de la siguiente dirección electrónica:

  `https://doodle.com/poll/f37g2r5ap4rwzvtm`

- Deben escribir el nombre (sin apellidos) de todos los miembros del grupo y seleccionar uno de los horarios disponibles.
- Todos los miembros del grupo deben estar presentes para defender cada una de las preguntas. Si un estudiante no esta presente, entonces el estudiante perderá 35 puntos de la nota final.