Instituto Tecnológico de Costa Rica
Engineering School
Computer Engineering
CE 3102 — Análisis Numérico
II Semester 2020

Tarea 2

Paralelización del método de Jacobi

Hernaldo Ramirez, Gabriel Abarca, Alejandra Castrillo, Bryan Masís
Group 01
Profesor: Juan Pablo Soto

29 de noviembre de 2020

Índice

1. Algoritmo

El algoritmo a paralelizar como se mencionó en el título, corresponde a la versión modificada del método numérico de Jacobi que fue visto en clases, el cual; en lugar de trabajar sobre representaciones matriciales, trabaja sobre un vector de valores, el cual se puede representar mediante:

$$x^{(k)} = [x_1^{(k)}, x_2^{(k)}, ..., x_m^{(k)}]^T$$

El vector anterior corresponde a la k-ésima iteración y sirve como base para la $k+1$-ésima iteración, la cuál se calcula mediante:

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

Donde:

- A**:** Matriz cuadrada mxm diagonalmente dominante.

2. Modificaciones

El método base de Jacobi anterior, trabaja calculando un valor del vector x_k por ciclo, o sea; completamente serial.
Sin embargo; la independencia de datos entre iteraciones permite realizar una paralelización de la fórmula fundamental, logrando un potencial nivel de paralelización m, donde m representa la cantidad y filas y columnas de la matriz A.
Para efectos de esta implementación, se paralelizará a solamente a 4, ya que es el número de núcleos físicos que posee la computadora más rápida del grupo.
Para realizar la paralelización del método se debe utilizó la biblioteca *parallel* de Octave, la cual posee una función que permite evaluar una función para todos los valores dentro de un vector (pararrayfun).

Para paralelizar el método, se generó un archivo que contiene una función auxiliar que recibe los mismos parámetros que el método serial de Jacobi, además, de un valor i que determina cuál x_i va a calcular.

Debemos recordar que nosotros poseemos un valor de m que determina la cantidad de x_i a calcular por iteración del método, por lo tanto; si se genera un arreglo en octave que va de 1 a 242, se puede utilizar como insumo para la función *pararrayfun*, la cuál tiene el siguiente formato:

result=pararrayfun(n,fun,array);

Donde:

- **result:** Arreglo que almacenará todos los resultados obtenidos en todos los cálculos.
- **n:** Número de procesadores que trabajarán por ejecución.
- **fun:** Función a evaluar.

- **array:** Arreglo de valores por evaluar en la función objetivo.

Una vez esclarecido el funcionamiento del método para paralelizar de octave, el código completo sigue el comportamiento del siguiente pseudo-código:

```
function parte1_p4()
    A = tridiagonal(p,q,m);
    p=q=[1:0.1:25];
    b=ones(m,1);
    tmpXk = zeros(242, 1);
    tol = 105;
    mArray = 1:m;
    while(condicionDeParada< tol)
        tmpXk=ejecuciónParalela(jacobi, mArray);
        condicionDeParada = norma(A*tmpXk-b);
    return tmpXk;
```

Como se puede observar el código anterior, la primera parte es dedicada únicamente a la inicialización de las variables del problema, mientras existe un único bloque de *ejecución* para todo el problema, el cual se encarga de evaluar la función de Jacobi en todo los valores dentro del vector $mArray$ el cual va de 1 a 242.
La función Jacobi llamada en el pseudocódigo, es la función auxiliar que se comporta de acuerdo al siguiente pseudocódigo:

```
function jacobi_aux(A,xk, m,i)
    for j de 1 a m:
        if(j!=i)
            sum = sum + A(i,j)*xk(j); //formula de jacobi
    return result = (1/A(i,i))*(b(i)-sum)
```

Básicamente la función realiza uno sólo de los dos *for* anidados necesarios para realizar la iteración del método, repartiendo el ciclo principal entre los procesadores.
Cabe destacar que todo el peso de calendarización y ejecución de cada uno de los cálculos es manejados por la biblioteca *parallel* como tal, librando al programador de esa responsabilidad.

3. Restricciones

La biblioteca de paralelización de octave tiene limitantes, las cuales son:

- La función a paralelizar debe estar en un archivo aparte.
- La función a paralelizar debe tener un manejador, por lo que en el código se genera uno al agregar la etiqueta @(i), donde i es el parámetro que varía en la función auxiliar.
- El vector resultante obtenido por ejecución es un vector fila, por lo que antes de comprobar si la iteración cumple se calcula la traspuesta del vector resultante, para convertirlo en un vector columna.