

CE3102: Analisis numérico para ingeniería

II Semestre 2021

Profesor: Juan Pablo Soto Quiros

Grupo 05

Integrantes:

-Carlos Adrián Araya Ramírez

-Michael Shakime Richards Sparks

-Sebastian Mora Godinez

-David Cordero

Implementación en paralelo
del método de Jacobi.

Marco teórico

Método de Jacobi

El método de Jacobi resuelve sistemas de ecuaciones lineales de la forma $Ax = b$, donde $A \in \mathbb{R}^{m \times m}$ es una matriz diagonalmente dominante. La implementación realizada en esta tarea evita la representación matricial y esta dice que si se tiene que $x^{(k)} = [x_1^{(k)} x_2^{(k)} \dots x_m^{(k)}]^T$ es una aproximación generada por el método de Jacobi en la k -ésima iteración, entonces para calcular la $k + 1$ ésima iteración $x^{(k+1)} = [x_1^{(k+1)} x_2^{(k+1)} \dots x_m^{(k+1)}]^T$ se puede calcular mediante la formula:

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

Para cada $i = 1, 2, \dots, m$.

Función pararrayfun del paquete parallel en Octave

Este paquete contiene funciones para la ejecución paralela local explícita y funciones para la ejecución paralela utilizando varios núcleos de la computadora [1]. La función *pararrayfun* recibe como parámetros **la cantidad de procesadores** (núcleos o procesadores lógicos) que se utilizarán en el proceso, **un vector de elementos** y **una función** que se utilizará para evaluar el vector de elementos y devuelve el vector de elementos resultante.

A continuación, en la siguiente imagen se muestra un ejemplo del funcionamiento de *pararrayfun* utilizando para la evaluación la función $f(x) = x^2$

```
# fun is the function to apply
fun = @(x) x^2;

vector_x = 1:10;

vector_y = pararrayfun(nproc, fun, vector_x)
```

Imagen 1. Ejemplo pararrayfun, tomada de [2].

En la imagen 1, el *vector_x* es evaluado en cada uno de sus elementos por la función *fun* y el vector resultante es guardado en el *vector_y*. Este código al ser ejecutado produce la salida que se muestra en la siguiente imagen.

```
parcellfun: 10/10 jobs done

vector_y =

    1     4     9    16    25    36    49    64    81   100
```

Imagen 2. Salida del ejemplo, tomada de [2].

La función *pararrayfun* también permite utilizar como función de evaluación a una función externa guardada en un archivo local con extensión *.m*, para realizar esto se debe colocar un *@* como prefijo la función de evaluación en los parámetros de *pararrayfun*. Además, es importante, para esta implementación destacar que también es posible pasar un por parámetro a la función de evaluación **variables y el valor del elemento del arreglo inicial en el índice actual (n debe ser el primer parámetro de la función externa y también debe agregarse entre paréntesis en el prefijo de la función externa después del @ y entre parentesis)**. En la siguiente imagen se muestra un ejemplo de lo mencionado.

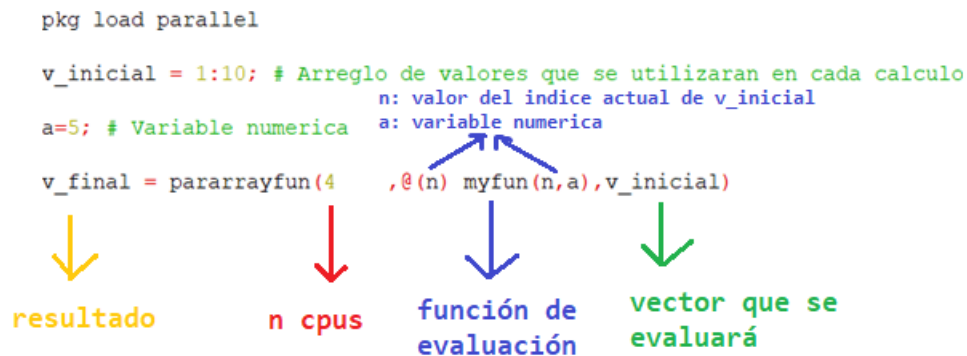


Imagen 3. Ejemplo de *pararrayfun* con función de evaluación externa.

En este caso *n* serán los valores del 1 al 10 del vector *v_inicial*, esto permite que la función *myfun* utilice los valores actuales del vector *v_inicial* para realizar las operaciones que producirán el *v_final* resultante. En la siguiente imagen se muestra la función que evalúa cada elemento del vector *v_inicial* y el vector *v_final* resultante.

```
function v_final_i = myfun(n, a);
    v_final_i=n*a;
endfunction
```

Imagen 4. Función de evaluación externa del ejemplo de *pararrayfun*.

```
>> prueba_par_array  
  
v_final_i =  
5  10  15  20  25  30  35  40  45  50
```

Imagen 5. Resultado del ejemplo de pararrayfun con función externa.

En este caso se realiza la multiplicación de cada elemento del `v_inicial` por la constante `a = 5`, es decir, en cada cálculo en paralelo se realiza las siguientes operaciones que se observan en la siguiente imagen:

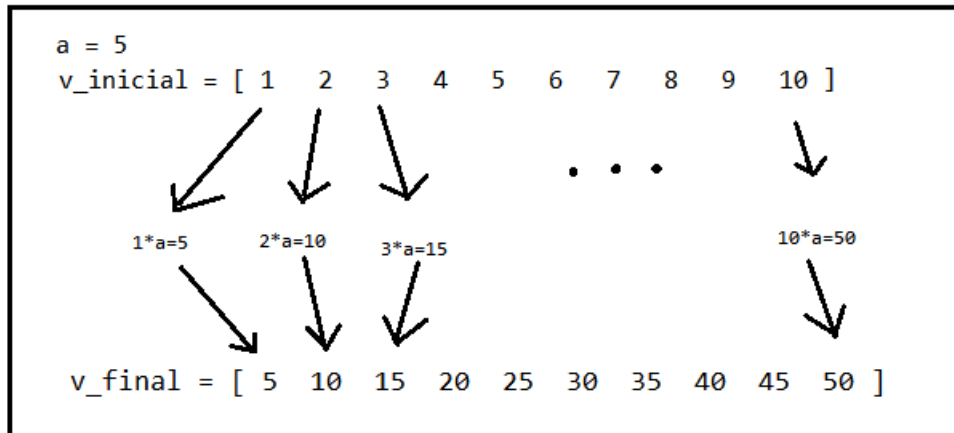


Imagen 6. Explicación del ejemplo de pararrayfun con función externa.

Implementación en paralelo del método de Jacobi

Para realizar esta implementación se debe realizar el cálculo en paralelo de todos los x_i hasta x_m de cada iteración $x^{(k+1)}$ donde estos cálculos únicamente dependen del valor anterior, es decir, de $x^{(k)}$. En el siguiente diagrama se puede observar como se debe realizar el cálculo en paralelo.

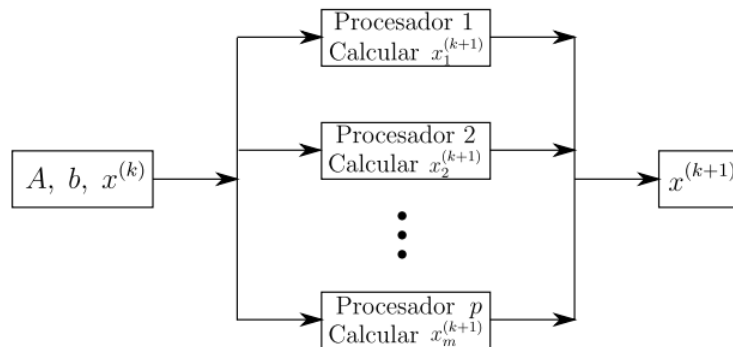


Imagen 7. Diagrama de la implementación del método de Jacobi en paralelo.

Esta implementación se basa en el uso de la función `pararrayfun` del paquete `parallel` y la formula (1) utilizando una función de evaluación externa aplicada a un vector inicial que contiene los valores que van de [1 hasta m], donde m es el tamaño de la matriz A . Es importante resaltar que acá se utilizará la técnica descrita en el marco teórico sobre el uso de funciones de evaluación externa donde se pasa como parámetro una variable que contiene el valor actual del vector inicial pasado por parámetro.

A continuación, se muestra la implementación del método de Jacobi en paralelo en GNU Octave.

```
function [xk,k,error]=partel_p3(A,b,tol,iterMax,x0,cpus)
```

```
    m=length(A);
```

```
    xk=x0;
```

```
    error = tol+1;
```

```
    k=0;
```

```
    while k<iterMax & error>tol
```

```
        ni = 1:m; → i=1,2,...,m
```

Valor de xk+1 después del cálculo en paralelo

```
        xk = pararrayfun(cpus,@(i) xk_plus_1(i,A,b,xk),ni);
```

```
        error = norm((A*xk')-b);
```

```
        if error < tol | k >= iterMax
            return
        endif
```

```
        k+=1;
```

```
    endwhile
```

```
end
```

$$x_i^{(k+1)} = \frac{1}{A_{i,i}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^m A_{i,j} x_j^{(k)} \right) \text{ para cada } i = 1, 2, \dots, m.$$

Vector xk, en cada calculo será utilizado el x_i con cada elemento de ni

La matriz de coeficientes A y el vector de términos independientes b utilizados en la formula

Imagen 8. Implementación del método de Jacobi en paralelo.

```

function xk = xk_plus_1(i,A,b,xk)

    aux = 0;
    m=length(A);

    for j=1:m

        if j!=i

            aux += A(i,j)*xk(j);

        endif

    endfor

    xk = (1/A(i,i))*(b(i)-aux);
endfunction

```

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

para cada $i = 1, 2, \dots, m$.

Imagen 9. Función de evaluación exterior del método de Jacobi en paralelo.

En la imagen 8 se puede observar como el valor de x_k es calculado mediante la función *pararrayfun* del paquete parallel, y a esta se le pasa por parámetros los **procesadores a utilizar**, la **función de evaluación externa** con sus variables numéricas necesarias en sus parámetros para realizar el cálculo de la formula (1) y **el vector inicial ni**. Además, se le pasa el parámetro i que es el valor actual del elemento al cual se le va a aplicar la función externa.

En la imagen 9 se puede observar la función de evaluación externa que utiliza la función *pararrayfun* para calcular los valores de cada elemento x_i hasta x_m de $x^{(k+1)}$. Acá se puede observar cómo se utiliza el valor de i para cada calculo según la formula (1).

Finalmente, en la siguiente imagen se muestra una explicación del método de Jacobi paralelo.

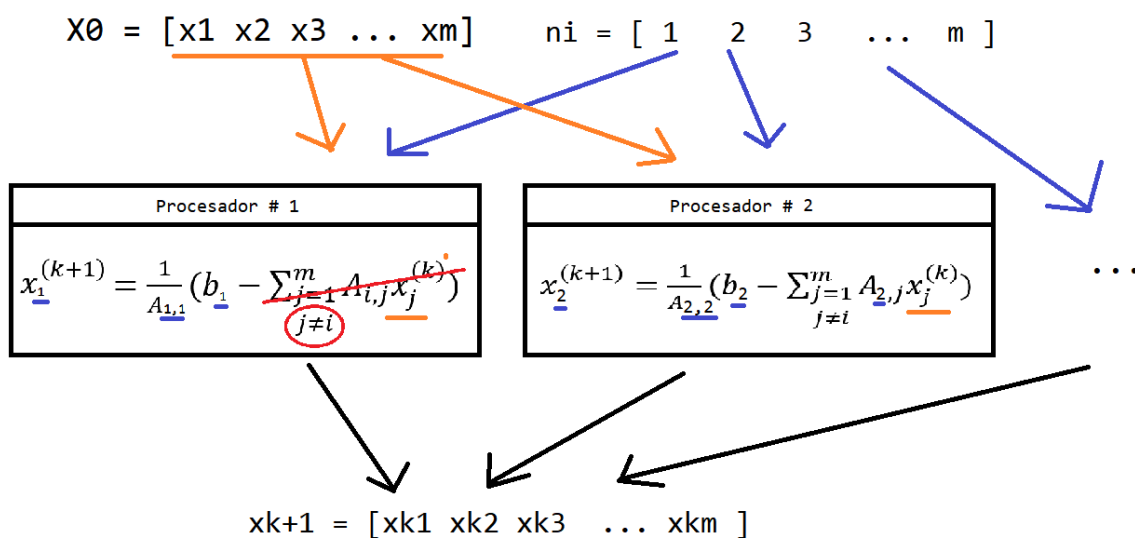


Imagen 10. Explicación del método de Jacobi en paralelo.

Referencias

- [1] "Top (parallel_doc)", *Octave.sourceforge.io*, 2021. [Online]. Available: https://octave.sourceforge.io/parallel/package_doc/. [Accessed: 07- Oct- 2021].
- [2] "Parallel package - Octave", *Wiki.octave.org*, 2021. [Online]. Available: https://wiki.octave.org/Parallel_package. [Accessed: 07- Oct- 2021].