Instituto Tecnológico de Costa Rica

Análisis Numérico para Ingeniería

Tarea No. 2

Grupo 6

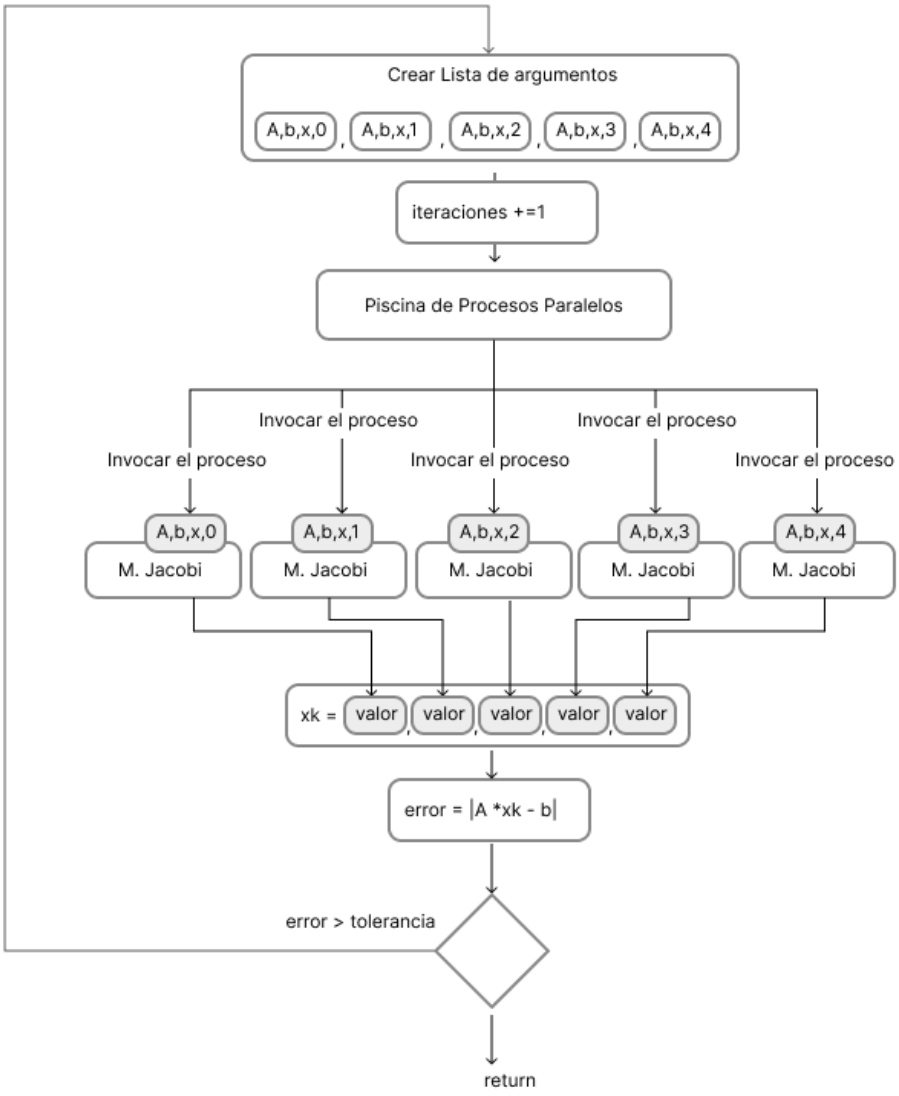
Luis Andrey Zuñiga Hernández
Adrián González Jimenez
Brian Wagemans Alvarado

Semestre II

Para describir el proceso del método paralelo de Jacobi recurrimos al siguiente diagrama, los detalles de implementación variarán de lenguaje de programación a otro, ya que este contexto adaptado a python utiliza el concepto de piscinas de procesos, en síntesis, se puede crear una piscina y hacer que ejecute simultáneamente una cantidad determinada de veces un proceso. El siguiente diagrama se explicará en orden de aparición de los bloques.

Con la capacidad de invocar simultáneamente un proceso, debemos crear la lista de argumentos necesarios para el proceso, en este caso corresponde a la ecuación brindada para aproximar el método de Jacobi, por lo que se requiere de la matriz A, el vector b, la aproximación actual x y la posición del valor dentro del vector xk. Cabe destacar que se contará como una iteración al proceso que encierra la ejecución en paralelo de todos los procesos y una vez que todos hayan finalizado sus valores retornados conformarán el vector xk.

Para determinar el error utilizamos la norma de Frobenius y si su grado de error es mayor que la mínima tolerancia se invocará de nuevo el proceso para obtener un resultado más preciso.



```python
def jacobiParallel(A,x,b,iterMax,tol,processor):
    start=time.time()
    pool=multiprocessing.Pool(processes=processor)
    iteraciones=0
    error=0
    for k in range(iterMax):
        iteraciones=k+1
        args=[(A,b,x,i) for i in range(len(x))]
        results=pool.starmap(calculateElement, args)
        x=np.transpose([results])
        mat=np.subtract(np.dot(A,x),b)
        error=np.linalg.norm(mat)
        if(error<tol):
            break
    end=time.time()
    elapsed=end-start
    return x,iteraciones,error,elapsed

def calculateElement(A,b,x,i):

    sumaParcial=0
    for j in range(len(A)):
        if(j!=i):
            sumaParcial+=A[i][j]*x[j][0]
    return (1/(A[i][i]))*(b[i][0]-sumaParcial)
```

Para este caso y como se muestra en la figura anterior, la función que se paraleliza es el calculate element, que como se habia indicado calcula la i esima posición del vector de la k-esima aproximación. En este caso y como se puede observar se utiliza el multiprocesing sincrónico para calcula dicha aproximación, tal y como se había planteado en el diagrama de la solución, posteriormente se espera a que el cálculo de todos los anteriormente mencionados sean calculados, para proceder a calcula el error de dicha solución con el fin de verificar si es necesario proceder con la siguiente aproximación.