Trabajo Final - Análisis Numérico

Paralelización de Red-Black Gauss-Seidel

Integrantes:

- Fantini, Luis Mariano
- Mateos, Wenceslao
- Pablos Di Marco, Braulio Leonel

Fecha de entrega: 26/11/2019

Introducción

Los métodos para la resolución de sistemas ecuaciones lineales tienen un uso ciertamente extendido debido a la diversidad de problemas de ingeniería en donde se presenta este problema, sin importar la disciplina particular. Sin embargo, los vistos en la materia, no escalan adecuadamente cuando el sistema a resolver es de un orden considerablemente grande, ya sea porque el desempeño de los algoritmos aplicados se ve mermado al tener que trabajar con arreglos de gran tamaño o debido a la inestabilidad subyacente.

En este trabajo se presenta la aplicación de paralelismo para atender a esta problemática. Particularmente, se implementó la variante del método de Gauss-Seidel conocida como **Red-Black Gauss-Seidel** en su forma para problemas unidimensionales, el cual aprovecha la independencia de ciertos grupos de incógnitas para calcularlas en simultáneo.

Por otro lado, también se explica el trasfondo de su codificación en Fortran mediante el uso de coarrays, una herramienta brindada desde el lenguaje para el desarrollo de programas de ejecución paralela.

Finalmente, se exhiben los resultados obtenidos tras ejecutar una implementación particular de Gauss-Seidel preparada para problemas unidimensionales, Thomas y Red-Black.

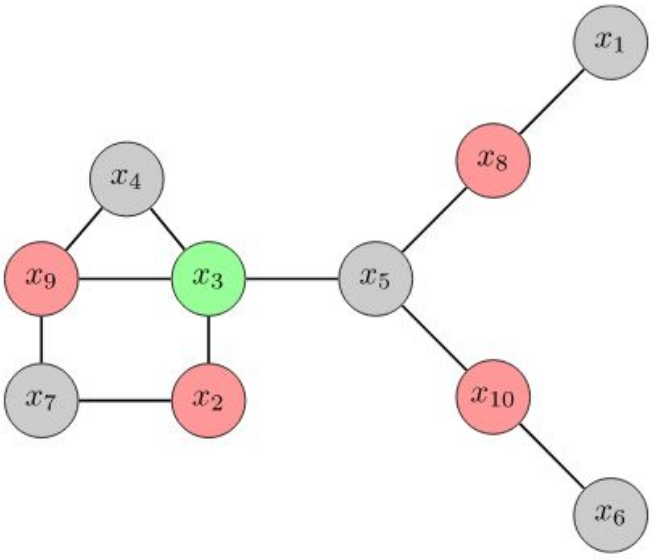
Gauss-Seidel coloreado

Los métodos de Gauss-Seidel coloreados surgen para posibilitar la aplicación de paralelismo en Gauss-Seidel. Se basan en la identificación de conjuntos o “colores” de incógnitas que no tienen una dependencia directa entre ellas. Una vez reconocidos estos conjuntos, se los ordena en una secuencia y para resolver el sistema de ecuaciones en cuestión se calculan las incógnitas de un cierto color en simultáneo antes de proceder a calcular los del color siguiente. Esta forma de encontrar el valor de las variables hace que sea un método semi-paralelo, ya que entre conjuntos sigue existiendo secuencialidad.

Por ejemplo, si se tiene la siguiente matriz de coeficientes:

$$\begin{bmatrix}
3.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 3.0 & 0.0 & 0.0 \\
0.0 & 15.0 & 7.0 & 0.0 & 0.0 & 0.0 & 8.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 7.0 & 28.0 & 9.0 & 9.0 & 0.0 & 0.0 & 0.0 & 3.0 & 0.0 \\
0.0 & 0.0 & 9.0 & 18.0 & 0.0 & 0.0 & 0.0 & 0.0 & 9.0 & 0.0 \\
0.0 & 0.0 & 9.0 & 0.0 & 20.0 & 0.0 & 0.0 & 8.0 & 0.0 & 3.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 8.0 & 0.0 & 0.0 & 0.0 & 8.0 \\
0.0 & 8.0 & 0.0 & 0.0 & 0.0 & 0.0 & 14.0 & 0.0 & 6.0 & 0.0 \\
3.0 & 0.0 & 0.0 & 0.0 & 8.0 & 0.0 & 0.0 & 11.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 3.0 & 9.0 & 0.0 & 0.0 & 6.0 & 0.0 & 18.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 3.0 & 8.0 & 0.0 & 0.0 & 0.0 & 11.0
\end{bmatrix}$$

Si se representan a las incógnitas del sistema como vértices de un grafo cuyas aristas indican las relaciones directas de los mismos, resultaría el siguiente gráfico:



El grafo ya presenta a las variables con un color asignado, lo que identifica a los grupos independientes entre sí. Si se elige una secuencia gris, rojo y verde, entonces por cada iteración del método se calcularían primero *x1, x4, x5, x6* y *x7* en paralelo, luego *x2, x8, x9* y *x10* y finalmente *x3*.

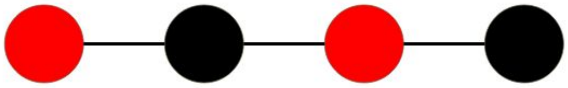
La aplicación práctica de Gauss-Seidel coloreado se ve limitada, debido a que encontrar los conjuntos de incógnitas independientes no solo es algorítmicamente complejo, sino que puede resultar más costoso que simplemente tratar de resolver el sistema de ecuaciones con métodos secuenciales. Sin embargo, existen problemas donde naturalmente se dan estas independencias sin necesidad de esclarecerlas de antemano, dando lugar así a Red-Black Gauss-Seidel.

Red-Black Gauss-Seidel

Se trata de un esquema particular de Gauss-Seidel coloreado, donde las incógnitas pertenecen a un único conjunto de independencia, “rojo” o “negro”. Primero, se actualizan todas las variables “rojas” en simultáneo, seguidas luego por las “negras”. Se presentan tres esquemas variantes que serán descriptas a continuación.

Red-Black Gauss-Seidel unidimensional

El algoritmo paralelo implementado en este trabajo corresponde a esta variación. Se utiliza en problemas donde las incógnitas son nodos que conforman una línea. De esta manera, cada variable depende de forma directa solo de la anterior y de la siguiente:



La matriz de coeficientes que representa a esta configuración es tridiagonal.

Es aplicable en los métodos de diferencias finitas para la resolución de ecuaciones diferenciales en derivadas parciales elípticas y parabólicas (método de Crank-Nicolson), para situaciones que puedan representarse en una sola dimensión (como la transmisión de calor en una varilla). También se puede utilizar para encontrar splines cúbicos.

Tomando *a* como la matriz de coeficientes, *k* como la iteración actual, *b* como el conjunto de términos independientes y *x* como el de incógnitas, la fórmula general calcular cada nodo *i* rojo o impar es:

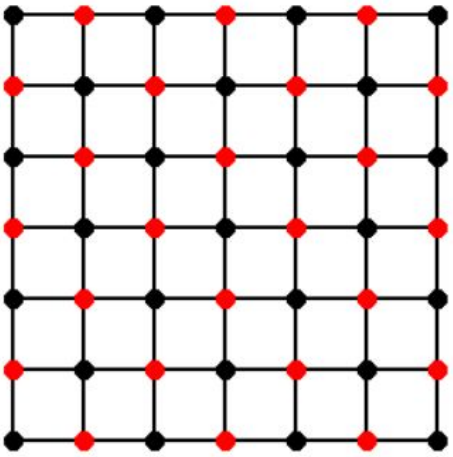
$$x_i^{k+1} = \frac{1}{a_{i,i}} \cdot \left(b_i - a_{i,i-1} \cdot x_{i-1}^{k} - a_{i,i+1} \cdot x_{i+1}^{k} \right)$$

La ecuación general para calcular los nodos negros o pares es:

$$x_i^{k+1} = \frac{1}{a_{i,i}} \cdot \left(b_i - a_{i,i-1} \cdot x_{i-1}^{k+1} - a_{i,i+1} \cdot x_{i+1}^{k+1} \right)$$

Red-Black Gauss-Seidel bidimensional

Resuelve sistemas de ecuaciones lineales donde las variables conforman un mallado rectangular, por lo que cada incógnita depende de las presentes en sus laterales:

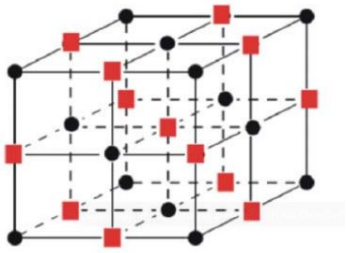


La matriz de coeficientes para este esquema es de 5 bandas.

Esta es la variante del método más popular, ya que se utiliza principalmente para resolver ecuaciones diferenciales en derivadas parciales elípticas en regiones rectangulares. También puede ser aplicada para implementar Crank-Nicolson en dichos sectores. En ambos casos, aplicando diferencias finitas.

Red-Black Gauss-Seidel tridimensional

Se aplica para situaciones donde se presentan nodos distribuidos en una región cúbica:



La matriz de coeficientes asociada es de 7 bandas.

Nuevamente puede ser aplicada para la resolución mediante diferencias finitas de ecuaciones diferenciales en derivadas parciales elípticas o parabólicas mediante Crank-Nicolson, esta vez para sectores volumétricos.

Paralelismo en Fortran

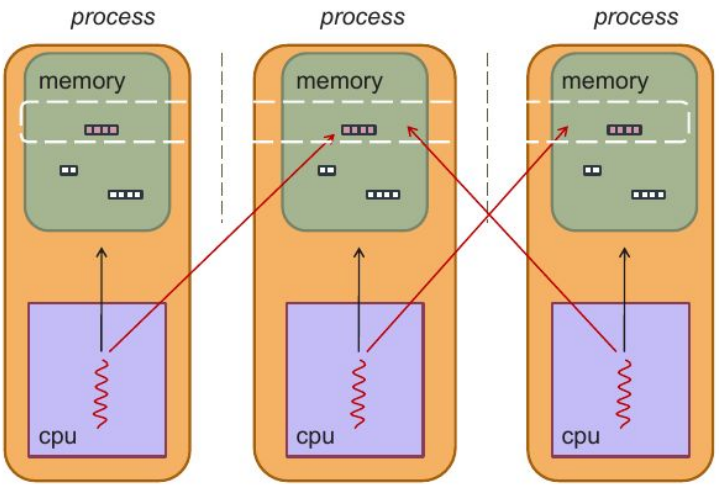
Fortran es uno de los pocos lenguajes de programación que brindan mecanismos de paralelización como parte del mismo mediante los denominados *coarrays* o coarreglos, los cuales fueron incorporados al estándar ISO del 2008.

Este esquema de paralelización se basa en replicar el mismo proceso una cierta cantidad de veces, denominando imágenes a dichas copias. Cada una de ellas se ejecuta de forma asíncrona, no necesariamente siguiendo el mismo curso. Debido a esto, el modelo de paralelismo presentado responde al SPMD (Single Program, Multiple Data o Programa Único, Datos Múltiples).

Fortran aporta las sentencias SYNC ALL y SYNC IMAGES para la sincronización explícita de todas las imágenes o de algunas particulares, respectivamente. También cuenta con mecanismos para bloquear ciertos datos compartidos o regiones de código para solventar las condiciones de carrera presentes en sistemas concurrentes.

Los coarreglos mencionados previamente se tratan de variables (tanto escalares como arreglos) que cada imagen mantiene localmente pero que a su vez puede acceder remotamente a la partición correspondiente en otra imagen. Se trata de un caso de memoria global compartida con verificación de tipo, donde a cada imagen le corresponde una división equitativa de la misma. De esta manera, el esquema de coarreglos responde al modelo PGAS (Partitioned Global Address Space o Espacio de Direcciones Global Particionado).

La siguiente figura representa la implementación de paralelismo en Fortran:



A continuación se demuestra una declaración de coarreglos:

```
real, dimension(10), codimension[*] :: x
integer :: y[*]
```

El símbolo '*' que aparece en ambas declaraciones implica que el rango del coarreglo sea el número de imágenes, por lo que cada una de ellas mantendrá de forma local una variable *x* e *y* de los tipos y dimensiones correspondientes. Como ya se mencionó antes, el acceso remoto es posible (por ejemplo, para referenciar a la variable *y* de la imagen 1 basta con escribir *y[1]*). Se destaca el uso de corchetes para diferenciar la sintaxis para arreglos y para coarreglos.

Procedimiento

Se codificó en GNU Fortran 7.4.0 una variación del método de Gauss-Seidel optimizada para matrices tridiagonales, con el fin de usarla como base para el desarrollo de Red-Black en una dimensión y para más adelante comparar el rendimiento de ambos métodos. Dicha versión de Fortran ya implementa gran parte del estándar 2008, siendo de particular interés para este trabajo la incorporación de coarreglos mediante la librería OpenCoarrays.

Para la generación de casos de prueba se desarrolló un módulo capaz de producir siempre la misma sucesión aleatoria de términos independientes y valores iniciales en la cantidad correspondiente a un orden especificado para el sistema de ecuaciones que se busca obtener. Dicha secuencia se genera a partir de una semilla, con el fin de poder repetir los experimentos. La diagonal principal de la matriz de coeficientes se fijó en 4 y las diagonales superiores e inferiores en -1, con la intención de asegurar la convergencia de los métodos indirectos.

Se seleccionó a la imagen número 1 no solo para calcular y mostrar los tiempos de los otros métodos no paralelizados a comparar, sino que también para transferir los datos iniciales al resto de las imágenes.

Una vez concluidas las librerías y los programas del caso unidimensional se procedió a la prueba, donde fueron comparados Red-Black Gauss-Seidel con Thomas refinado iterativamente (aprovechando que se estaba trabajando con matrices tridiagonales) y con Gauss-Seidel sin paralelización. En cada test con un sistema de ecuaciones particular cada método fue aplicado 50 veces con una tolerancia de error de 1E-5, para obtener un promedio de su tiempo de ejecución. Para Red-Black además se registró un segundo tiempo adicionándole el tiempo de distribución de datos en las distintas imágenes. La ejecución fue realizada en un sistema GNU/Linux, específicamente Ubuntu 18.04.3, sobre un Intel Celeron N2840 a 2.16GHz con 4 GB de RAM y un Intel i7-65000U a 2.50GHz con 6 GB de RAM.

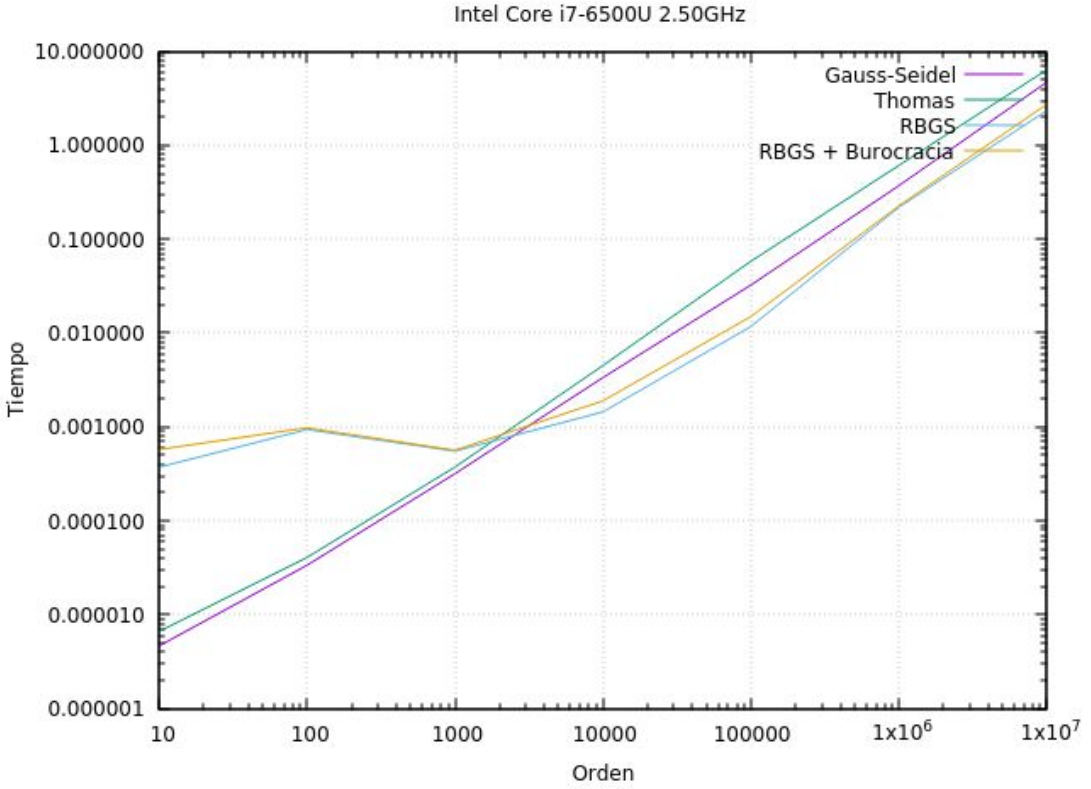
Resultados obtenidos

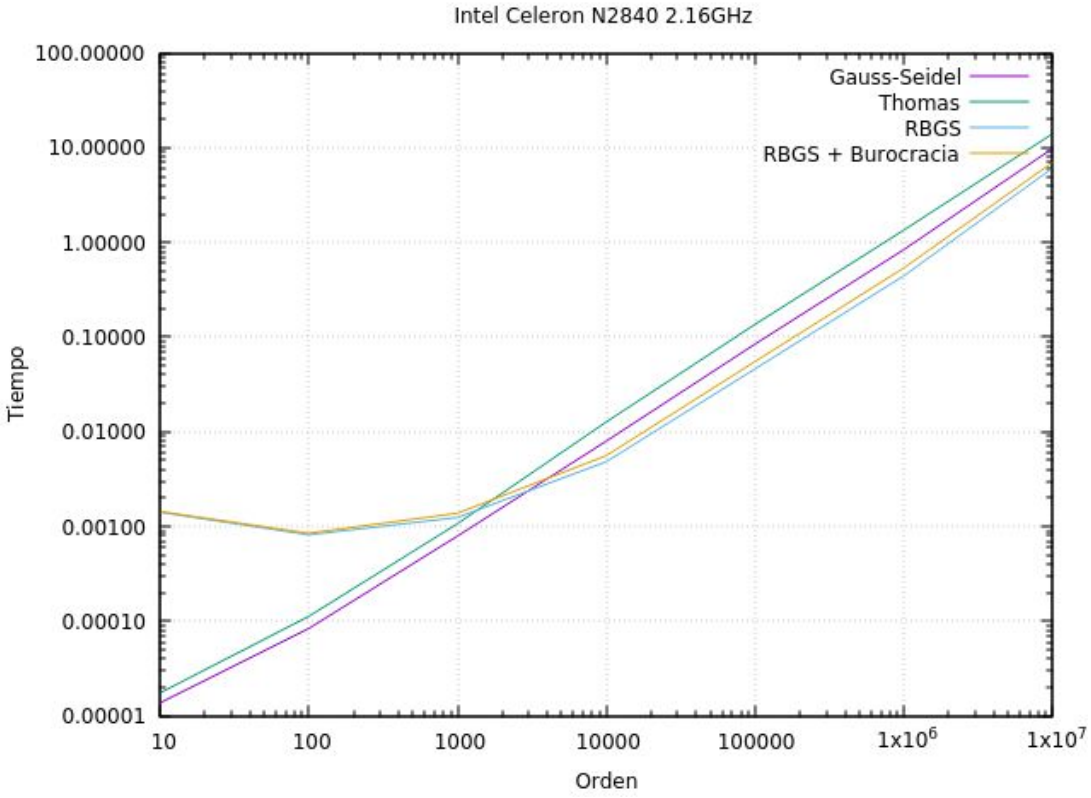
A continuación se presentan los tiempos obtenidos bajo las condiciones previamente descritas:

Tiempo de ejecución en segundos para un Intel Celeron N2840 a 2.16GHz							
Orden	**10**	**100**	**1.000**	**10.000**	**100.000**	**1.000.000**	**10.000.000**
Gauss-Seidel	0.0000136	0.0000831	0.0007897	0.0079505	0.0842747	0.8394571	9.8708213
Thomas	0.0000173	0.0001117	0.0010690	0.0126604	0.1363316	1.3516172	14.1192467
RBGS	0.0014079	0.0008153	0.0012384	0.0048110	0.0461554	0.4454729	6.1156710
RBGS + dist.	0.0014319	0.0008433	0.0013744	0.0055670	0.0552564	0.5374609	7.0435190

Tiempo de ejecución en segundos para un Intel i7-65000U a 2.50GHz							
Orden	**10**	**100**	**1.000**	**10.000**	**100.000**	**1.000.000**	**10.000.000**
Gauss-Seidel	0.00000462	0.00003338	0.00031680	0.00335062	0.03236960	0.37168518	4.73051810
Thomas	0.00000656	0.00004038	0.00037188	0.00450580	0.05807882	0.61088038	6.39190262
RBGS	0.00037004	0.00093490	0.00055112	0.00143964	0.01174246	0.21916548	2.34693402
RBGS + dist.	0.00057256	0.00097428	0.00056332	0.00188730	0.01493466	0.22686342	2.74800002

Los datos arrojados se presentan en las siguientes imágenes:





Conclusión

Es destacable que Red-Black Gauss-Seidel no siempre fue más veloz que Gauss-Seidel. Su rendimiento se vio afectado por el orden de la matriz sistema a calcular y de la cantidad de núcleos del procesador desde donde se ejecuta el código.

Tanto en el procesador Intel Celeron como en el Intel i7 utilizados Red-Black Gauss-Seidel demoró más tiempo que Gauss-Seidel para órdenes menores a 10.000. Esto es así ya que el método introduce un costo computacional adicional para preparar los coarreglos y si el orden del problema es pequeño, este gasto se vuelve significativo. Además, la espera para la sincronización entre imágenes también introduce demoras importantes para sistemas poco extensos.

En cambio, en los ejemplos en los que el sistema a calcular empezó a tomar dimensiones considerables, RBGS logró mejores tiempos debido a que el costo de armado de coarreglos y de sincronización pierden relevancia frente a la velocidad de cómputo adquirida por paralelismo.

Además, es apreciable que RBGS se desempeñó mejor en el procesador de cuatro núcleos frente al de dos. En los casos de prueba en donde se tenían dos núcleos, la mejora tendió a reducir el tiempo a la mitad, mientras que cuando se tenían cuatro, el tiempo de ejecución se acercó a la tercera parte del conseguido mediante Gauss-Seidel.

Cabe destacar que RBGS fue comparado contra una implementación particular de Gauss-Seidel optimizada para matrices tridiagonales. Se espera que un desarrollo similar para problemas en dos y tres dimensiones brinde diferencias más marcadas no solo a favor de Red-Black, sino también para un Gauss-Seidel similarmente optimizado.

Por otro lado, debido a que las pruebas se realizaron en un sistema operativo multipropósito, los valores obtenidos son propensos a variar dependiendo la carga de tareas de dicho sistema y las decisiones que tome el planificador de procesos asociado.

Se encontró una barrera a la hora a la hora de realizar las pruebas. Los equipos utilizados para realizarlas no poseen gran cantidad de memoria RAM, por lo que no fue posible obtener resultados para órdenes de magnitud mayor a diez millones. En estos casos o el sistema operativo terminó matando a los procesos de cálculo debido a insuficiencia de memoria o, en el caso de que el tamaño de los datos aún hiciera posible operar mediante memoria virtual, la hiperpaginación consecuente no permitió obtener tiempos confiables.

Como observación Red-Black Gauss-Seidel y Gauss-Seidel presentan diferencias en sus resultados más allá de la tolerancia aplicada. Por otro lado, se notaron variaciones de

hasta dos iteraciones entre la convergencia de ambos métodos. Esto se debe a las distintas formas de obtener los valores de las incógnitas en cada uno de ellos. A grandes rasgos, se podría decir que Red-Black es un híbrido entre Jacobi y Gauss-Seidel, dado que la mitad de los nodos se calculan a partir del resultado conseguido en la iteración anterior y la otra mitad con valores de la actual.

Por último y enfocado al desarrollo realizado, los coarreglos brindan una herramienta sencilla y en sintonía con la sintaxis del lenguaje Fortran que facilita la codificación de algoritmos paralelos. Sin embargo, se pudo observar al investigar su aplicación que no existen ejemplos complejos de uso en la web y que la documentación relevante no refleja del todo bien su funcionamiento, debido a que los compiladores del lenguaje presentan distintos grados de avance en su implementación.