Conciencia Digital
www.concienciadigital.org

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

Una propuesta para el cálculo del pseudoespectro en unidades de procesamiento gráfico

A proposal for pseudospectra computation on graphic processor units

Zenaida Natividad Castillo Marrero. [1], Gustavo Adolfo Colmenares Pacheco. [2], Paulina Elizabeth Valverde Aguirre. [3] & Víctor Oswaldo Cevallos Vique. [4]

Recibido: 09-03-2021 / Revisado: 16-03-2021 /Aceptado: 06-04-2021/ Publicado: 05-05-2021

Abstract.

DOI: https://doi.org/10.33262/concienciadigital.v4i2.1.1703

Introduction. Computation of matrix pseudospectra is required in many applications when modeling by differential equations. This computation is really expensive, especially for large matrices, for which highly parallelizable algorithms have been successfully implemented on high performance computers. **Objective**. We present an exploratory analysis of the pseudospectra computation in a hybrid architecture CPU-GPU where the graphics processing unit performs the massive parallel computation. **Methodology**. A proposal is formulated after analyzing some parallel implementations on high performance computers, methods based on Krylov methods, and the capacities of the graphics processor units for massive computation in the large-scale setting. **Results**. The proposal is attractive since the graphics processing units currently can be found on a wide range of computers, or can be adapted to any computer at a very a low cost. **Conclusions**. In this document we describe a general scheme for the parallel computation of pseudospectra on a hybrid architecture CPU-GPU.

Keywords: Pseudospectra, Eigenvalues, Krylov methods, GPU, NVIDIA.

[1] Escuela Superior Politécnica de Chimborazo, Facultad de Ciencias, Escuela de Estadística, Grupo CIDED. Riobamba, Ecuador, zenaida.castillo@espoch.edu.ec, https://orcid.org/0000-0002-4424-8652

[2] Universidad de Investigación Experimental Yachay, Escuela de Matemáticas y Ciencias Computacionales, Urcuquí, Ecuador, gcolmenares@yachaytech.edu.ec, https://orcid.org/0000-0003-4789-0859

[3] Escuela Superior Politécnica de Chimborazo, Facultad de Ciencias, Grupo CIDED. Riobamba, Ecuador, paulina.valverde@espoch.edu.ec, https://orcid.org/0000-0003-0458-7083

[4] Escuela Superior Politécnica de Chimborazo, Facultad de Administración de Empresas, Escuela de Finanzas, Grupo CIDED, Riobamba, Ecuador, victor.cevallos@espoch.edu.ec, https://orcid.org/0000-0001-5525-5818

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

Resumen.

Introducción. El cálculo del pseudoespectro de matrices es requerido en muchas aplicaciones modeladas por ecuaciones diferenciales y discretizadas en tiempo y espacio. Este cálculo resulta ser muy costoso computacionalmente, sobre todo para matrices de gran magnitud, para las cuales se han implementado con éxito métodos altamente paralelizables ejecutados en máquinas de alto rendimiento. **Objetivo**. Se presenta un análisis exploratorio del cálculo del pseudoespectro y su posible implementación en una arquitectura híbrida CPU-GPU, en la cual el cómputo masivo y paralelo se realice en las unidades de procesamiento gráfico. **Metodología**. Para formular la propuesta se analizan algunas implementaciones de este cálculo que han resultado efectivas en máquinas de alto rendimientos, el uso de métodos basados en métodos de Krylov, y las capacidades de las unidades de procesamiento gráfico en el cómputo masivo. **Resultados**. La propuesta resulta de interés debido a que la mayor parte de este cálculo puede realizarse en unidades de procesamiento gráfico, que en la actualidad son fáciles de adquirir a bajo costo, y están incluidas en la mayoría de las marcas y modelos presentes en el mercado. **Conclusiones**. En este documento se describe un esquema general para la paralelización del cálculo del pseudoespectro en una arquitectura híbrida CPU-GPU.

Palabras claves: Pseudoespectro, Autovectores, Métodos de Krylov, GPU, NVIDIA.

Introducción

En los modelos lineales la investigación se basa en la información provista por los autovalores, y para muchos problemas relacionados con la ciencia y la ingeniería este análisis es satisfactorio, sobre todo cuando las matrices son normales. Sin embargo, en algunas áreas de la industria, las matrices que se derivan del modelaje matemático, no poseen columnas ortogonales, y en caso de que puedan ortogonalizarse el proceso numérico es costoso o resulta en aproximaciones muy pobres o poco valederas para la toma de decisiones. Adicionalmente, la no-normalidad influye negativamente en la convergencia de los métodos iterativos. En estos casos el pseudoespectro es una herramienta útil, que proporciona una alternativa visual para reforzar el análisis espectral. Áreas de aplicación exitosa de técnicas con autovalores y pseudoautovalores incluyen el estudio de la acústica, análisis estructural, mecánica cuántica, mecánica de fluidos, procesos de revestimiento de superficies, entre otros.

Con la llegada de los ordenadores de alto rendimiento, que permiten cálculos en paralelo, se abrió la posibilidad de avanzar con el procesamiento masivo de datos en aplicaciones a gran escala. De esta manera muchos autores se avocaron a la tarea de redefinir o adaptar sus algoritmos para sacar el mayor provecho al uso de estas tecnologías, entre ellos vale la pena mencionar los trabajos de Bekas et al. (2001), Mezher & Philippe (2011), y Noschese & Reichel (2015). Estos algoritmos han sido programados en su mayoría a través de una arquitectura de pase de mensajes como MPI o OpenMP, ver por ejemplo el

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

trabajo de Bekas et al. (2002) con MPI_Matlab o el de Minini et al. (2011) con un híbrido MPI-OpenMP, ambos implementados en computadores con decenas de miles de procesadores.

Por otra parte, en la última década, las unidades de procesamiento gráfico o GPU (por sus siglas en inglés) adaptadas a la gran mayoría de microcomputadores y computadores personales, han probado tener una capacidad de cómputo que puede ser usada para aligerar significativamente la carga de la unidad principal de procesamiento o CPU. Este hecho, sumado a que no siempre se dispone de una máquina de alto rendimiento, nos lleva a considerar la implementación de estos métodos bajo un esquema híbrido que involucre cálculos numéricos densos en las GPU y manejo de entrada-salida a cargo del CPU.

En este trabajo se describe una propuesta para la implementación de un esquema paralelo para el cálculo del pseudoespectro de matrices en un modelo CPU-GPU.

Metodologia

Para el desarrollo de la propuesta se describe el problema de hallar el pseudoespectro de matrices de gran tamaño y se mencionan algunos métodos que han resultado efectivos en su resolución en computadores de alto rendimiento. En particular se detalla el funcionamiento de los algoritmos que integran la propuesta y su posible implementación en un modelo de arquitectura CPU-GPU.

El problema de los autovalores y el pseudoespectro

El espectro de una matriz $A \in \mathbb{C}^{nxn}$ es el conjunto de escalares *z* tales que existe un vector no nulo *x* tal que $Ax = zx$. El escalar *z*, generalmente complejo, se conoce como autovalor y se dice que *x* es su autovector asociado. También pudiéramos decir que *z* es autovalor de *A* si y sólo si $det\ (zI - A) = 0$. Esto conlleva a la siguiente definición.

Definición 1: Dada una matriz $A \in \mathbb{C}^{nxn}$, se define el espectro de *A*, denotándolo como $\Lambda(A)$, de la siguiente manera:

$$\Lambda(A) = \{z \in \mathbb{C}: det\ (zI - A) = 0\}$$

Ahora bien, si $det\ (zI - A) = 0$ entonces la matriz $(zI - A)^{-1}$ no está definida, y a tal efecto se define su norma como infinita ($\|(zI - A)^{-1}\| = \infty$). Pero, ¿qué ocurre cuando $\|(zI - A)^{-1}\|$ es finita pero muy grande?, la formulación de esta pregunta es la que permite construir la primera definición de pseudoespectro:

Definición 2: Dada una matriz $A \in \mathbb{C}^{nxn}$, se define el ε-pseudoespectro de *A*, para ε > 0, denotándolo como $\Lambda_\varepsilon(A)$, de la siguiente manera:

$$\Lambda_\varepsilon(A) = \{ z \in \mathbb{C}: \|(zI - A)^{-1}\| \geq \frac{1}{\varepsilon} \}$$

Otra definición equivalente, muy usada en teoría de perturbaciones, expresa al pseudoespectro en términos de los autovalores de una matriz perturbada en un épsilon (ε).

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

Definición 3: Dada una matriz $A \in \mathbb{C}^{nxn}$, se define el ε-pseudoespectro de A, denotándolo como $\Lambda_\varepsilon(A)$, de la siguiente manera:

$$\Lambda_\varepsilon(A) = \{ z \in \mathbb{C}: z \in \Lambda(A+E) \ / \ \|E\| < \varepsilon \}$$

Si z se encuentra en el pseudoespectro, entonces se denomina pseudoautovalor de A. A cada pseudoautovalor z puede asociarse un pseudoautovector v, en general, no único. Esto permite dar otra definición del pseudoespectro:

Definición 4: Dada una matriz $A \in \mathbb{C}^{nxn}$, se define el ε-pseudoespectro de A, denotándolo como $\Lambda_\varepsilon(A)$, de la siguiente manera:

$$\Lambda_\varepsilon(A) = \{z \in \mathbb{C}: \|(zI - A)v\| \leq \varepsilon, v \in \mathbb{C}^n, \ \|v\| = 1\}$$

Los diferentes métodos para el cálculo del pseudoespectro de una matriz usan estas definiciones u otras equivalentes. En este trabajo se propone un esquema para el cálculo del pseudoespectro usando esta última definición basada en el valor singular mínimo de la matriz $(zI - A)$.

Representación Gráfica del pseudoespectro

Uno de los aportes del pseudoespectro es que el análisis puede realizarse al observar la gráfica que se genera variando los valores de épsilon (ε). Cada valor de ε representa una perturbación de la matriz A que genera una curva en el plano complejo que encierra el área donde se encontrarían los autovalores de la matriz perturbada. La curva representa entonces la frontera de movimiento de los autovalores en el plano complejo.

Si una matriz u operador A es normal, es decir, si posee una base ortogonal de autovectores, entonces su pseudoespectro (en norma 2), está conformado por bolas cerradas de radio ε alrededor de los autovalores. Para el caso de matrices u operadores no normales, el pseudoespectro se presenta de forma menos predecible, lo cual hace su análisis más interesante.

La figura 1 muestra curvas de aproximación del pseudoespectro de una matriz hermitiana de orden $n = 50$ conocida como matriz 'smoke' (a), y una matriz Toepliz no simétrica de orden $n = 50$ conocida como 'grcar' (b). Ambas matrices son usadas frecuentemente en pruebas de algoritmos para el cálculo del pseudoespectro debido a su carácter normal y no-normal que se refleja en las curvas de su pseudoespectro. Información más detallada de las características de estas matrices, y las aplicaciones que modelan, se sugiere ver la página de Pseudospectra Gateway, mantenida por Embree & Trefethen (2021).

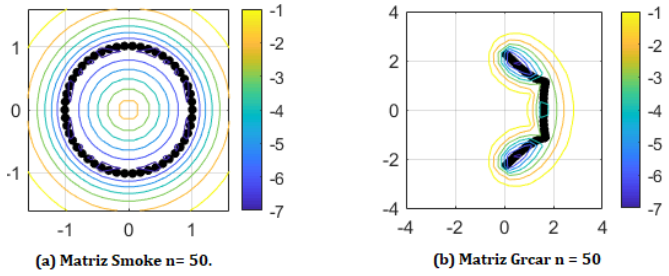

Figura 1. Pseudoespectro de matrices Normales (a) y No-Normales (b)
Fuente: Elaboración propia.

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

Las gráficas se presentan en el plano complejo, y los puntos negros representan los autovalores de la matriz, mientras que los trazos coloreados representan los límites del pseudoespectro para cada valor de épsilon, el cual se indica en la barra de colores a la derecha. Por ejemplo, la curva más externa de color amarillo representa una perturbación de la matriz en $\varepsilon = 10^{-1}$, mientras que la curva en color azul representa una perturbación en $\varepsilon = 10^{-7}$. Se puede apreciar que mientras más normal[5] sea la matriz, las curvas más cercanas a los autovalores son aquellas correspondientes a valores de $\varepsilon \approx 0$.

La mayor utilidad del Pseudoespectro como herramienta de análisis se presenta en aquellas aplicaciones modeladas con matrices no-normales. Tal es el caso de las matrices Toeplitz en banda, como la matriz Grcar, cuyo pseudoespectro se presenta en la figura 1 (b). Estas matrices suelen aparecer después de la discretización de ecuaciones diferenciales, y a menudo es de interés conocer los autovalores para determinar la estabilidad de las soluciones de un problema de valor inicial. La estructura de esta matriz se presenta en la figura 2.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & & & & \\ -1 & 1 & 1 & 1 & 1 & & & \\ & -1 & 1 & 1 & 1 & 1 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & -1 & 1 & 1 & 1 & 1 \\ & & & & -1 & 1 & 1 & 1 \\ & & & & & -1 & 1 & 1 \\ & & & & & & -1 & 1 \end{pmatrix}$$

Figura 2. Estructura de la matriz Toeplitz en banda
Fuente: Elaboración propia.

Los autovalores de estas matrices son altamente sensibles a las perturbaciones, de manera que una pequeña perturbación puede resultar en un cambio apreciable de sus autovalores, y en consecuencia el análisis que se hace con estos autovalores puede conllevar a decisiones no apropiadas. Adicionalmente la convergencia de los métodos resulta comprometida.

Una vez entendida la teoría detrás del Pseudoespectro, como herramienta de análisis en aplicaciones con matrices no-normales, muchos investigadores se dedicaron a la efectividad de su cálculo mediante diversos métodos. A continuación se resume el estado del arte de este cálculo.

Cálculo del pseudoespectro

Generalmente el cálculo de los autovalores y de los pseudoautovalores de matrices resulta en procesos computacionalmente costosos, en términos de requerimientos de memoria y tiempo de cómputo. Por esta razón, en las últimas décadas la atención se ha dirigido hacia métodos de proyección en espacios de menor dimensión, los cuales permiten disminuir

[5] La norma de $(A^TA - AA^T)$ proporciona una medida de la no-normalidad.

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

los requerimientos de memoria, y además contienen un nivel de paralelismo intrínseco que puede ser usado para reducir el tiempo total de cálculo**.**

De particular interés han resultado los métodos basados en proyecciones de la matriz original en espacios de Krylov, ya que permiten hallar aproximaciones con matrices de menor dimensión que contienen la información espectral de relevancia de la matriz original, pero su tamaño y estructura aligeran el cálculo, el cual se reduce principalmente a operaciones matriz-vector con alto grado de paralelismo. En la actualidad, diversos métodos han resultado exitosos cuando se implementan en máquinas de alto rendimiento; ver por ejemplo los trabajos de Mezher & Philippe (2002) y Minini et al. (2011).

Todos estos trabajos fueron implementados en computadores con una gran capacidad de cómputo, los cuales son parte de una tecnología costosa y de difícil mantenimiento, que no siempre está disponible para muchas instituciones. Por esta razón se hace necesaria la adaptación de estos métodos a arquitecturas de gama media y baja en donde puedan resolverse algunos problemas a mediana escala y quizás también puedan ejecutarse cálculos preliminares de problemas a gran escala, antes de su efectiva implementación en computadores de alto rendimiento.

En este documento se extienden las ideas tratadas por algunos autores para la aproximación visual y numérica del pseudoespectro en paralelo. En particular, partimos del trabajo realizado por Otero et al. (2015), en el cual se paraleliza el cálculo del pseudoespectro en un arreglo de clústeres usando una partición del dominio de interés.

Paralelización del cálculo del pseudoespectro

Entre las implementaciones más usadas para el cálculo en paralelo del pseudoespectro se encuentran aquellas basadas en métodos de continuación; las cuales comienzan hallando un punto en la frontera del pseudoespectro para un valor específico de ε y a partir de allí siguen la trayectoria definida por una curva en el plano complejo para la cual el valor $\sigma_{min}(zI - A)$ sea constante, ver por ejemplo Lui (1997). La principal debilidad de estas implementaciones es la obtención del punto en la frontera del pseudoespectro. Adicionalmente, el tiempo de cómputo se incrementa cuando el pseudoespectro tiene componentes conexas, ya que cada componente debe ser tratada particularmente.

Por otra parte, tenemos los algoritmos de malla, que tienen un esquema simple y organizado de ejecución. En estos se define una región de interés en el plano complejo, la cual se discretiza mediante una malla tan refinada como se desee y por cada punto de la malla se calcula $\sigma_{min}(zI - A)$. La elección de la región del plano complejo y la definición del número de puntos a procesar son claves en estos métodos, a fin de obtener contornos con la información deseada.

A continuación, en la figura 3, adaptada de Otero et. al (2015), se presenta un esquema básico del cálculo, basados en algoritmo de malla, para el cálculo del pseudoespectro. La idea general de estos esquemas fue propuesta por Trefethen (1999). Algunos autores han

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

implementado con éxito este tipo de métodos, véase por ejemplo Trefethen & Wright (2001) y las referencias en Trefethen & Embree (2005).

Algoritmo: Cálculo del pseudospectro a gran escala

1: Definir y discretizar la región de interés K en el plano complejo
2: Usar Arnoldi con reinicio implícito sobre la matriz A de orden n
para obtener una matriz H_m con $m << n$
3: **Para** cada punto $z \in K$ de la región discretizada **hacer**
4: Calcular la descomposición QR de $zI - H_m$
5: Obtener $\lambda_{max}(z)$ usando Lanczos inverso sobre R^*R
6: Obtener $\sigma_{min}(z) = \frac{1}{sqrt\ (\lambda_{max}(z))}$
7: **Fin para**
8: Visualizar los contornos del pseudoespectro en K

Figura 3. Algoritmo propuesto para matrices de gran tamaño
Fuente: Elaboración propia.

Los algoritmos que proponemos contienen intrínsecamente paralelismo de grano grueso (mínima comunicación entre unidades) una vez que están basados en la discretización de la región de interés, y paralelismo fino (mucha comunicación entre subtareas) en las operaciones matriz-vector que se utilizan para el cálculo de $\sigma_{min}(zI - A)$. En las próximas secciones se presenta el método de proyección que se propone para obtener matrices de menor dimensión que contengan información espectral de la matriz original A, y la propuesta de discretización de la región de interés K.

Proyecciones en espacios de Krylov

Los métodos basados en Arnoldi se fundamentan en la descomposición Hessenberg de la matriz A o $AV = VH$, donde V es una matriz Ortogonal; es decir, $V^tV = I$ y H es una matriz Hessenberg superior. Esta descomposición es ideal para la resolución simultánea de los problemas $Ax = b$ y $Av = \lambda v$, ya que provee un mecanismo mediante el cual se trata un sistema de menor dimensión, además de tomar ventaja de realizar operaciones del tipo producto matriz-vector. Para una mejor comprensión de esta descomposición se sugiere ver Treffethen & Bau (1997) y Golub & Van Loan (1996).

Para obtener la descomposición de Hessenberg $AV = VH$, el método de Arnoldi construye V usando una base ortogonal del subespacio de Krylov:

$$K_m(A, v_1) \equiv Span\{v_1, Av_1, A^2v_1, A^3v_1, \dots, A^{2m-1}v_1\}$$

En este caso, dado un vector inicial v_1 se añade un nuevo vector a la base en cada iteración, multiplicando el último vector por la matriz A. En la m-ésima iteración de este algoritmo se satisface la ecuación $AV_m = V_mH_m + f_me_m^t$ llamada factorización de

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

Arnoldi de orden m, donde V_m es una matriz de orden $n\,x\,m$, y H_m es una matriz Hessenberg de orden m.

Cuando la matriz A es simétrica, la matriz H_m toma la forma de una tridiagonal y se habla entonces del método de Lanczos. En el caso que nos compete se recomienda usar el método de Lanczos inverso para hallar el valor singular mínimo de $(zI - A)$ o de su proyección $(zI - H_m)$, considerando que este cálculo se traduce en hallar los autovalores de la matriz $(zI - A)^T(zI - A)$ o de $(zI - H_m)^T(zI - H_m)$, las cuales son simétricas.

La precisión y la velocidad de convergencia de este tipo de métodos mejora con el incremento de m. Sin embargo, cuando m aumenta, también aumenta el número de vectores de Arnoldi y el tamaño de la matriz Hessenberg. Esto compromete los recursos de memoria y el costo de CPU, por lo que en la práctica se sugiere mantener m en un valor bajo $(m << n)$ hasta lograr una precisión aceptable, o en algunos casos se produce un reinicio del proceso usando como vector inicial la última información obtenida. En la figura 4 se muestra el tiempo de cómputo de la factorización a medida que se incrementa el valor de m para una matriz aleatoria de orden $n = 1000$. Se usa el método de Arnoldi con reinicio implícito propuesto por Sorensen (1997), y en todas las pruebas el cómputo se detiene al alcanzar una tolerancia de 10^{-8} en la aproximación de la norma de $(AV_m - V_mH_m)$. Implementaciones en bloque para matrices no simétricas y de gran magnitud también han sido propuestas, ver por el ejemplo el trabajo de Castillo (2004).

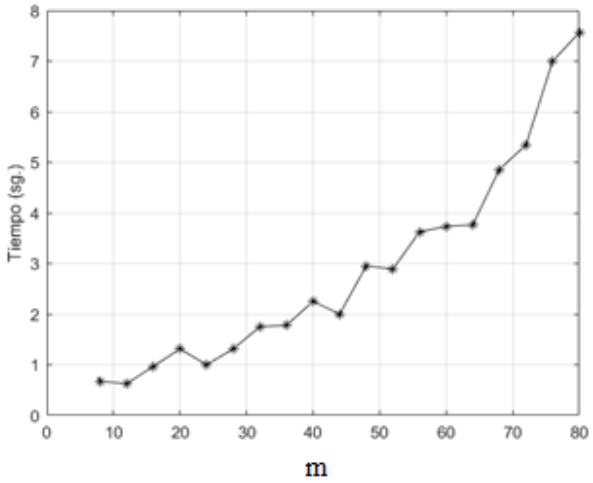

Figura 4. Tiempo de CPU: Iteración *m* de Arnoldi

Fuente: Elaboración propia.

Discretización de la región de interés

De acuerdo a la aplicación, el investigador pudiera estar interesado en la parte del espectro donde los autovalores toman un valor en particular, por ejemplo, los de mayor módulo en estudios de estabilidad, o para el análisis de convergencia de métodos iterativos. Este interés define la región en el plano en la cual se analiza el pseudoespectro.

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

Una vez definida la región de interés K el próximo paso será la discretización o definición de una malla de puntos. Supongamos que se define una malla de dimensión $m_x \times n_y$ en la región de interés K, tal como se ilustra en la figura 5(a), adaptada de Guevara (2012).

Cada punto (i,j) con $1 \leq i \leq m_x, 1 \leq j \leq n_y$, de esta malla, representa a un número $z \in \mathbb{C}$, con a $z = z(i,j)$.

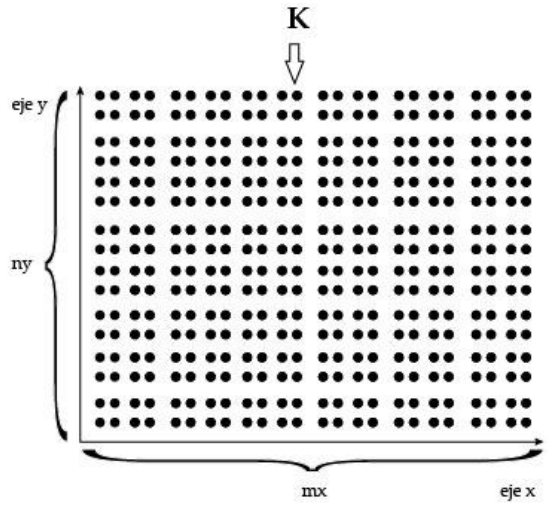

Figura 5(a). Malla de $m_x \times n_y$ puntos en la región K
Fuente: Elaboración propia.

Consideremos también que tenemos una arquitectura paralela con n procesadores, etiquetados P_0, P_1, …, P_{n-1}. Se define entonces un conjunto de n subregiones en K, etiquetadas desde 0 hasta $n-1$, tal como se presenta en la figura 5(b), adaptada de Guevara (2012).

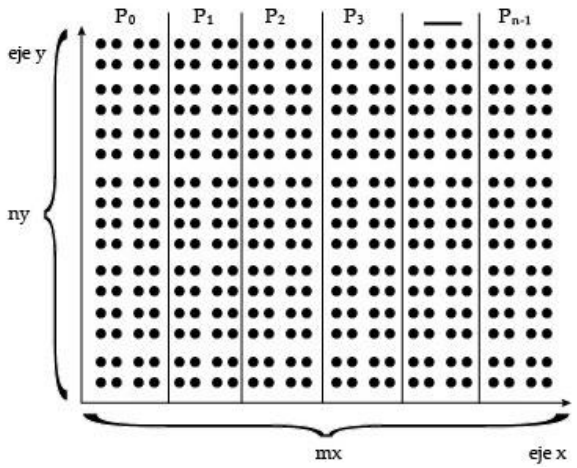

Figura 5(b). Distribución de la carga en los procesadores
Fuente: Elaboración propia.

Cada subregión se asocia a un proceso y cada proceso calcula $\sigma_{min}(zI - A)$ para todos los puntos $z(i,j)$ de su subregión. Este cálculo es simultáneo y representa un paralelismo de grano grueso soportado por la independencia de los datos. Los procesos terminan al enviar los valores singulares calculados al proceso maestro que asumirá la generación del

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

gráfico del pseudoespectro. Este esquema, basado en la discretización mostrada en la figura 5, permite disminuir significativamente el tiempo de cómputo.

Siempre podremos hacer una distribución equitativa del número de puntos que atiende cada procesador si el número total de puntos $m_x \,.\, n_y$ es divisible por n. En caso de que de $m_x \,.\, n_y$ no sea divisible por n podríamos por ejemplo recargar al procesador P_0 con $(m_x \,.\, n_y \ mod\ n)$ puntos adicionales.

Adicionalmente, cada cálculo de $\sigma_{min}(zI - A)$ se hace con un código basado en multiplicaciones matriz-vector, que se propone hacer en paralelo también representando un nivel de paralelismo de grano fino.

Unidades de Procesamiento Gráfico (GPUs)

En la actualidad los computadores personales y de oficina poseen múltiples núcleos o cores (procesadores) en la llamada Unidad Central de Procesamiento o CPU, además la tecnología incorpora, a bajo costo, tarjetas gráficas con una Unidad de Procesamiento Gráfico o GPU que también contiene muchos núcleos diseñados para el cómputo masivo, por lo que pudiéramos aceptar que la computación paralela está a nuestro alcance.

Los procesadores que se encuentran en la GPU pueden ser administrados eficientemente desde la unidad central de procesamiento o CPU para realizar cómputo intensivo. La idea es sustituir los procesos centralizados por procesos distribuidos entre los procesadores (cores) del CPU y los de las GPU. Para este fin se utilizan arquitecturas vectoriales o paralelas manejadas por plataformas de enlace que permiten consolidar resultados entre CPU y GPU.

En el año 2006 la compañía NVIDIA introduce la Arquitectura de Dispositivos de Cómputo Unificado o CUDA (por sus siglas en inglés), y hace posible el manejo efectivo de los núcleos de la GPU a través de un modelo de programación que permite la interacción CPU-GPU desde un programa de usuario. Detalles de esta tecnología, su uso y programación pueden encontrarse en la guía de programación de CUDA en NVIDIA (2021).

Con estas herramientas tecnológicas el potencial de la tarjeta gráfica se extiende a la implementación de algoritmos altamente paralelizables que consumen mucho tiempo de CPU; tal es el caso de los algoritmos basados en productos de matrices y de vectores, que aparecen frecuentemente en muchas aplicaciones del álgebra lineal, como el cálculo del espectro y/o el pseudoespectro de matrices dispersas y de gran magnitud.

Algunos investigadores se han dedicado desde entonces a la implementación de estos algoritmos, véase por ejemplo el trabajo de Angeles et al. (2011). Adicionalmente existen librerías de acceso libre con los códigos más usados. Estas librerías poseen interfaces para brindar flexibilidad en la programación y permitir la ejecución de instrucciones para la CPU y para la GPU en el mismo programa. Como ejemplo, véase Thrust, librería desarrollada por Bell y Hoberock (2011), que permite manejar estructuras de datos en algoritmos en paralelo, y la librería Cusp que contiene módulos para el manejo eficiente

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

de cálculo matricial y la ejecución de métodos clásicos para la resolución de sistemas lineales, desarrollada por Bell y Garland (2009).

El modelo de arquitectura induce a dejar las tareas con cálculos secuenciales, como el manejo de entrada y salida de datos, para que se ejecuten en la CPU (usualmente llamada host) y las tareas de cómputo masivo paralelizable que se ejecuten sobre la GPU (denominada device). Actualmente se cuenta con computadores de escritorio de 8 cores con una tarjeta Nvidia de miles de cores y este gap seguirá ampliándose en el futuro.

Cada núcleo de la GPU tiene recursos compartidos, como registros y memoria, integrados en el mismo chip, lo cual permite que las tareas (hilos) que se ejecutan en un mismo núcleo compartan datos sin usar un bus de memoria. La arquitectura permite al programador usar funciones escritas en lenguajes clásicos de alto nivel como C y C++ (conocidas como kernels) que luego se ejecutan en paralelo en la GPU como un conjunto de hilos que organizan dentro una jerarquía de bloques.

Para la programación en CUDA se debe considerar los siguientes elementos: jerarquía de bloques de hilos, compartición de memoria y sincronización.

Estos elementos dirigen la programación hacia la división del problema en tareas que pueden ser ejecutadas en paralelo por bloques de hilos independientes, que a su vez podrían agruparse en una malla de procesadores (grid), véase figura 6. La sincronización es manejada por el programador usando herramientas de CUDA que garantizan la escalabilidad; es decir, la aplicación de usuario debe soportar el incremento de cores en la GPU sin necesidad de reprogramación.

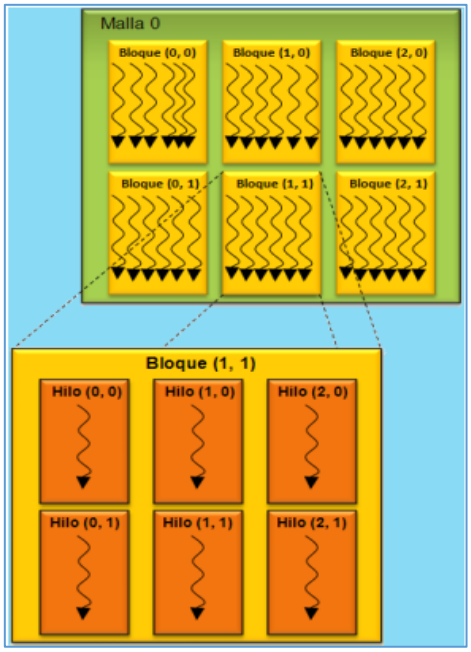

Figura 6. Malla de bloques de hilos
Fuente: Adaptada de NVIDIA (2021).

La propuesta de llevar estos cálculos a un esquema híbrido que involucre el cómputo masivo en la GPU no solo se sustenta en la posibilidad de incrementar el nivel de paralelismo y disminuir los tiempos de respuesta, sino que además el poder de cómputo

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

de los cores en las actuales GPU es muy superior a los de la CPU, véase por ejemplo el trabajo de Arce et. al (2011).

Resultados

Una vez analizados cada componente involucrado en el cálculo del pseudoespectro, se resume a continuación las bases de la propuesta:

1) Entrada de datos (CPU):
   1.1) Lectura o definición de la matriz A.
   1.2) Lectura o definición de los parámetros de Arnoldi.
   1.3) Distribución de la matriz A en los cores. (CPU-GPU).
2) Proyección de la matriz A, de orden n, en un espacio de dimensión $m << n$ (CPU-GPU).
   2.1) Distribución de la matriz A en los cores. (CPU-GPU).
   2.2) Ejecución de bloques de operaciones matriz-vector para aplicación del método de Arnoldi con reinicio implícito a la matriz A. (GPU).
   2.3) Generación de matriz H_m $(m << n)$ y su distribución o carga en los cores de la GPU. (CPU-GPU)
3) Discretización o definición de malla.
   3.1) Lectura o definición de parámetros de la malla. (CPU).
   3.2) Distribución de regiones en los cores la GPU. (GPU)
4) Cálculo del mínimo valor singular de los puntos en cada región (GPU).
   4.1) Para cada punto z de la subregión, ejecutar el método de Lanczos para hallar el valor singular mínimo de $\sigma_{min}(zI - H_m)$, a través del autovalor máximo de $(zI - H_m)^T(zI - H_m)$.
   4.2) Envío de la submatriz correspondiente a los $\sigma_{min}(zI - H_m)$ de cada z en la subregión.
5) Visualización de contornos (CPU)
   5.1) Graficación de curvas de contornos $\left(i, j, \sigma_{min}(zI - H_m)\right)$ con $z = z(i, j)$.

Conclusiones

- Se han descrito los aspectos fundamentales de una propuesta para el cálculo del pseudoespectro de matrices usando un esquema híbrido CPU-GPU que añade un nivel adicional de paralelismo con el fin de acelerar este cálculo que generalmente resulta costoso computacionalmente. Adicionalmente se han expuestos métodos que han logrado tener éxito en todas las implementaciones paralelas del pseudoespectro en máquinas de alto rendimiento. Se espera implementar estos algoritmos en un futuro inmediato bajo una plataforma CUDA con MPI usando GPU dispuestas en tarjetas NVIDIA en computadores de gama media o baja, a fin de probar el potencial de la propuesta.
- La propuesta conserva características generales y cada módulo puede ser implementado o bien utilizando los métodos recomendados en cada tarea, u otros métodos de preferencia.

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

El presente documento estuvo dirigido a:

1. Establecer los fundamentos teóricos para el cálculo del pseudoespectro; cubriendo detalles sobre su uso e importancia.
2. Justificar el uso de la herramienta y de los métodos de cálculo.
3. Desarrollar una propuesta general que contemple las distintas etapas del cálculo del pseudoespectro.
4. Proponer algoritmos básicos en cada actividad.
5. Describir las características de las unidades de procesamiento gráfico, y su utilidad para hacer cálculos en paralelo.
6. Señalar detalles de implementación y consideraciones de rigor en base a la arquitectura propuesta.

Referencias bibliográficas

Ángeles, M., Flores, G., Vidal, A. (2011). Implementación CPU-GPU y comparativa de las bibliotecas BLAS-CUBLAS, LAPACK-CULA, Universidad Politécnica de Valencia.

Bekas, C., Kokiopoulou, E., Koutis, I., Gallopoulus, E. (2001) Towards the effective Parallel Computation of Matrix Pseudospectra. IC's 01: Proceeding of the 15th International Conference on Supercomputing, June 2001, 203-226.

Bekas, C., Kokiopoulou, E., Gallopoulos E., Simoncini, V., Parallel Computation of Pseudospectra using Transfer Functions on a MATLAB-MPI Cluster Platform. Lecture Notes in Computer Science. DOI:10.1007/3-540-45825-5_35.

Bell, N., Garland, M. (2009). Implementing sparse matrix-vector multiplication on throughput-oriented processors. SC'09: Proceeding of the Conference on High Performance Computing Networking. Storage and Analysis, New York, NY, USA (ACM), 1-11.

Bell, N., Hoberock, J. (2011). Thrust: A Productivity-Oriented Library for CUDA. GPU Computing Gems, Jade Edition. Editores Wen-mei W. Hwu.

Castillo, Z. (2004). A new algorithm for continuation and bifurcation analysis of large-scale free surface flows. PhD thesis, Rice University, Houston, Texas.

Embree, M., Trefethen. L. N., Pseudospectra Gateway (2021). [Online]. Disponible en: https://www.cs.ox.ac.uk20/pseudospectra/index.html.

Guevara, R. (2012). Paralelización de datos para el cálculo del pseudoespectro. Tesis de Licenciatura. Universidad Central de Venezuela, Escuela de Computación, Caracas, Venezuela.

Golub, G., & Van Loan, C. (1996). Matrix Computations (3rd ed.). The Johns Hopkins University.

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

Lui, S. (1997). Computation of pseudospectra by continuation. SIAM J. Sci. Comput., 18, 2(1997), 565–573.

Mezher, D., Philippe, B. (2002). Parallel computation of pseudospectra of large sparse matrices. *Parallel Computing*, 28, 199-221.

Minini, P., Rosenberg, D., Reddy, R., Pouquet, A. (2011). A hybrid MPI-OpenMP scheme for scalable parallel pseudospectral computations for fluid turbulence. Parallel Computing, 37(6-7), 316-312.

Noschese, S. & Reichel, L. (2015). Approximated structured pseudospectra. *Numerical Linear Algebra and Applications*, 00, 1-15.

NVIDIA Corporation, NVIDIA CUDA C Programming Guide (Version 11.3.0) (2021). [Online]. Disponible en: https://docs.nvidia.com/cuda/cuda-c-programming-guide

Otero, B., Astudillo, R., Castillo, Z. (2015). Un esquema paralelo para el cálculo del pseudoespectro de matrices de gran magnitud. *Revista Internacional de Métodos Numéricos para el Cálculo y Diseño en Ingeniería*. 31-1, 8-12.

Sorensen D. C. (1997). Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations, *Parallel Numerical Algorithms*, Springer, Dordrecht, 119-165.

Trefethen, L. N. & Bau III, D. (1997). Numerical linear algebra. Siam, Philadelphia.

Trefethen, L.N. & Embree, M. (2005). Spectra and Pseudospectra: the behavior of nonnormal Matrices and Operators, Princeton University Press, New Jersey, USA.

Trefethen L.N & Wright, T. (2001). Large-scale computation of pseudospectra using ARPACK and eigs, *SIAM J. Sci. Comput*. 23 (2001), 591–605.

Trefethen, L.N. (1999). Computation of pseudospectra. Acta Numerica 8, 1, 247–295.

ISSN: 2600-5859
Vol. 4, N°2.1, p. 6-20, mayo, 2021

PARA CITAR EL ARTÍCULO INDEXADO.

Castillo Marrero, Z. N., Colmenares Pacheco, G. A., Valverde Aguirre, P. E., & Cevallos Vique, V. O. (2021). Una propuesta para el cálculo del pseudoespectro en unidades de procesamiento gráfico. ConcienciaDigital, 4(2.1), 6-20. https://doi.org/10.33262/concienciadigital.v4i2.1.1703