Departamento de Sistemas Informáticos y Computación

Estudio de Librerías Paralelas de Libre Distribución y Algoritmos Paralelos Iterativos Multipaso para la Resolución de Sistemas de Ecuaciones Lineales Dispersos. Aplicación a la Ecuación de Difusión Neutrónica

Tesis Doctoral

Presentada por Omar Flores Sánchez

Dirigida por Dr. Vicente E. Vidal Gimeno

Valencia, Febrero del 2009

A Georgina

A mis padres, abuelitos y hermanos

A mi Tec

A mi Patria

Con mi más sincero reconocimiento y agradecimiento al
Dr. Vicente E. Vidal Gimeno, por su guía durante
la preparación y elaboración de
esta tesis doctoral

Al Dr. Gumersindo J. Verdú Martín y al
Dr. Antonio M. Vidal Maciá por sus
palabras de aliento para continuar
y terminar esta aventura

Al Dr. Leroy Anthony Drummond del NERSC en Berkeley, California
por su amistad y por darme la oportunidad de utilizar
los recursos informáticos del cluster Jacquard

Al programa SUPERA del convenio SES–ANUIES en México
y al Ministerio de Educación y Ciencia en España
por los apoyos económicos recibidos durante la
realización de este trabajo doctoral

"Homo faber suae quisque fortunae"
Appius Claudius Caecus

"Vale maye toyo"
Anónimo

"El flojo siempre trabaja dos veces"
Profra. Silvia Sánchez Cruz

"¿Quién sabe más?...¿un burro preguntando o un sabio contestando?"
Sr. Pedro Flores González

"¡En la selva, hay que ser salvaje!"
Ing. Othón Valeriano Soto

"Sin prisa, pero sin pausa"
Dicho Popular Valenciano

RESUMEN

En esta tesis se abordan dos problemas fundamentales relacionados con los estudios de estabilidad y seguridad de reactores nucleares, donde es necesario resolver eficientemente sistemas de ecuaciones lineales dispersos de gran dimensión. El primero de ellos está relacionado con el problema de los modos Lambda de la ecuación de difusión neutrónica aplicada a un caso de estudio (reactor Ringhals–I, tipo *agua en ebullición o BWR*), que constituye un problema de valores propios generalizado. El segundo problema está relacionado con la resolución de un sistema de ecuaciones lineales disperso de gran dimensión que surge de la discretización temporal de la ecuación de difusión neutrónica aplicada a otro caso de estudio (reactor Leibstadt, tipo *BWR*) y que debe resolverse en distintos pasos de tiempo.

Para la resolución de los sistemas de ecuaciones lineales dispersos de gran dimensión asociados al problema de los modos Lambda, en esta tesis se ha realizado un estudio numérico del comportamiento secuencial y paralelo de algunos de los métodos que resuelven este tipo de problemas, tales como: métodos directos, métodos iterativos y métodos basados en subespacios de Krylov. Para realizar el estudio se han utilizando librerías de libre distribución, tanto secuenciales como paralelas. Con los resultados obtenidos, se han identificado aquellos métodos y librerías que resuelven más eficientemente los sistemas lineales para el caso de estudio seleccionado.

Para la resolución de los sistemas de ecuaciones lineales dispersos del caso dinámico, en esta tesis se han propuesto métodos iterativos multipaso para la aceleración de su resolución, los cuales también se han implementado secuencial y paralelamente utilizando librerías de libre distribución. En la experimentación de estos métodos iterativos multipaso propuestos se ha podido comprobar que se ha alcanzado una aceleración considerable y que pueden ser una opción apropiada para llevar a cabo simulaciones más complejas.

Los algoritmos que resuelven los problemas planteados en esta tesis y sus implementaciones paralelas se han evaluado experimentalmente con respecto a tiempo de ejecución, aceleración (speedup) y eficiencia en dos plataformas paralelas de memoria distribuida. Los resultados han mostrado que las implementaciones paralelas disminuyen considerablemente los tiempos de ejecución secuenciales, y las aceleraciones y eficiencias han sido aceptables.

SUMMARY

This thesis addresses two key problems related to stability and security studies of nuclear reactor cores, where it is necessary to solve very-large sparse linear equations systems efficiently. The first one is related to Lambda modes equation problem of the neutron diffusion equation applied to a realistic test case (Ringhals–I nuclear reactor, which is a *boiling water reactor or BWR type*), and it constitutes a generalized eigenvalue problem. The second one is related to the solution of very-large sparse linear equation systems that arise from the temporal discretization of the neutron diffusion equation applied to another realistic test case (Leibstadt nuclear reactor, which is a *BWR* type) and they must be solved in different time steps.

In order to solve the very–large sparse linear equations systems associated with the Lambda modes problem, in this thesis we have carried out a numerical study of the sequential and parallel performance of direct, iterative and Krylov–based iterative methods. This study has been done using sequential and parallel free distribution libraries. With the numerical results from this study we have identified all those methods and libraries that solve efficiently the sparse linear equation systems for the test case.

On the other hand, to solve the sparse linear equation systems related to second problem, in this thesis we have proposed second–degree iterative methods to speedup their solution. These methods have been implemented using sequential and parallel free distribution libraries. Several experiments carried out with second–degree iterative methods have showed a significant speedup of the solution process and they can be an option to carry out more complex simulations.

The algorithms which solve the problems presented in this thesis and their parallel implementations have been evaluated with respect to execution time, speedup and efficiency in two distributed memory parallel platform. Numerical experiments have showed that parallel implementations reduce significantly sequential execution times and overall parallel performance has been satisfactory.

RESUM

En aquesta tesis s'aborden dos problemes fonamentals relacionats amb els estudis d'estabilitat i seguretat de reactors nuclears, on esdevé necessari resoldre eficientment sistemes d'equacions linials dispersos de gran dimensió. El primer d'ells està relacionat amb el problema dels modes Lambda de la equació de difusió neutrònica aplicada a una cas d'estudi (reactor Ringhals–I tipus *aigua en ebullició o BWR*), que constitueix un problema de valors propis generalitzats. El segon problema està relacionat amb la resolució d'un sistema d'equacions linials disperses de gran dimensió que surt de la discretització temporal de la equació de difusión neutrònica aplicada a un altre cas d'estudi (reactor Leibstadt, tipus BWR) i que ha de resoldres en diferents passos de temps.

Per a la resolució dels sistemes d'equacions linials dispersos de gran dimensió associats al problema dels modes Lambda, en aquesta tesis s'ha realitzat un estudi numèric del comportament seqüencial i paral·lel d'alguns dels mètodes que resolen aquests tipus de problemes, tals com: mètodes directes, mètodes iteratius i mètodes basats en subespais de Krylov. Per a realitzar aquest estudi s'han emprat llibreries de lliure distribució, tant seqüencials com paral·leles. Amb els resultats obtesos, s'han identificat aquells mètodes i llibreries que resolen més eficientment els sistemes linials per al cas d'estudi seleccionat.

Per a la resolució dels sistemes d'equacions linials dispersos del cas dinàmic, en aquesta tesis s'han propost mètodes iteratius multi–pas per a l'acceleració de la seua resolució, els quals també s'han implementat sequencialment i paral·lelament utilitzant llibreries de lliure distribució. En l'experimentació d'aquestos mètodes iteratius multipas propostos s'ha pogut comprobar que s'ha assolit una acceleració considerable i que poden ser una opció apropiada per dur a terme simulacions més complexes.

Els algorismes que resolen aquestos problemes plantejats en aquesta tesis juntamente amb les seues implementacions paral·leles han estat avaluats experimentalment respecte al temps d'execució, acceleració (*speed–up*) i eficiència en dues plataformes paral·leles de memòria distribuïda. Els resultats han mostrat que les implementacions paralel·leles disminueixen considerablement els temps d'execució seqüencials, i les acceleracions i les eficiències han estat acceptables.

Índice general

Índice de tablas

Índice de figuras

Índice de algoritmos

Capítulo 1

Introducción

La modelización matemática de muchos fenómenos físicos involucra la utilización de sistemas de ecuaciones en derivadas parciales (EDP). Uno de los fenómenos físicos con más interés actualmente, es la modelización de la difusión de neutrones en el interior del núcleo de un reactor, especialmente reactores de agua en ebullición o tipo BWR (del Inglés Boiling Water Reactor) [85] [98], dado su impacto en la mejora de la seguridad en las centrales nucleares, y donde la resolución numérica rápida y eficiente de la ecuación de la difusión neutrónica (EDN) constituye una herramienta fundamental para la simulación y la prevención de desastres nucleares.

En el proceso de la resolución de la EDN pueden identificarse dos cálculos distintos aunque complementarios. El primero de ellos tiene como objetivo determinar la configuración estática del reactor en un instante de tiempo, y que toma la forma de un problema de valores propios generalizado. El otro tipo de cálculo se realiza para el estudio de un transitorio a partir de una perturbación efectuada sobre una configuración estática del reactor, utilizando para ello la EDN dependiente del tiempo.

La forma común de resolver la EDN es discretizándola, es decir, aproximando las EDP que la conforman mediante ecuaciones algebraicas que involucran un número finito de incógnitas. Para realizar estas aproximaciones se pueden utilizar diversos métodos, como métodos en diferencias finitas [128] [17], métodos nodales [49] [117] [60] y métodos basados en elementos finitos [59]. Con la discretización es posible reducir el problema original, al problema de resolver sistemas de ecuaciones lineales cuyas matrices de coeficientes se caracterizan, generalmente, por ser de gran tamaño y dispersas.

Una vez discretizadas las EDP involucradas, se hace necesario resolver numéricamente los sistemas de ecuaciones lineales resultantes. Para la resolución numérica de estos sistemas existen también diversos métodos, tanto directos [124] [21] [125] [24] (Gauss, Crout, etc.) como iterativos clásicos [126] [113] (Jacobi, Gauss–Seidel, Sobrerrelajación) e iterativos basados en subespacios de Krylov [91] [84] [66] (Gradiente Conjugado, Re-

siduo Mínimo Generalizado, entre otros).

Los problemas planteados en párrafos anteriores se caracterizan por ser problemas de gran dimensión, y la automatización de sus soluciones mediante cómputo secuencial no es factible con respecto a los tiempos de respuesta ideales para efectos de simulación o predicción. Así pues, para predecir con precisión el comportamiento estático y dinámico de los reactores BWR, se hace necesario desarrollar, además de métodos computacionales secuenciales, métodos computacionales paralelos que permitan reducir los costes asociados a la resolución numérica de los sistemas de ecuaciones lineales dispersos (SELD) de gran dimensión que surgen de la discretización de la EDN en su forma estacionaria y dinámica. La siguiente sección establece los objetivos principales de la presente memoria.

1.1. Objetivos

Conforme a lo dicho anteriormente, los objetivos principales de este trabajo de investigación son

- ★ Estudiar, probar y evaluar distintas librerías numéricas de libre distribución que permitan identificar los métodos más eficientes en la resolución de los sistemas de ecuaciones lineales dispersos de gran dimensión asociados con la EDN multigrupo en su forma estacionaria de un caso real, y
- ★ Diseñar, implementar, evaluar y extender algoritmos secuenciales y paralelos iterativos multipaso para resolver eficientemente los sistemas de ecuaciones lineales dispersos de gran dimensión que surgen en la discretización de la EDN multigrupo dependiente del tiempo de un caso real.

Para alcanzar los objetivos de este trabajo de tesis, se han definido los siguientes objetivos específicos:

- Identificar librerías del álgebra lineal numérica, tanto secuenciales como paralelas, que permitan la representación, la manipulación de operaciones y la resolución de sistemas dispersos de ecuaciones lineales de gran dimensión.
- Conformar una batería de matrices prueba, obtenidas de la discretización 3D de la EDN multigrupo partiendo de núcleos de reactores de agua en ebullición (tipo BWR) reales, que aborden tanto el caso estacionario como el caso dinámico.

- Realizar un estudio experimental que permita observar el desempeño ofrecido por métodos directos e iterativos contenidos en librerías numéricas de libre distribución, para la resolución secuencial y paralela de los SELD asociados a la EDN multigrupo, tanto para el problema estacionario como dinámico de los casos de estudio abordados.
- Con base en el estudio experimental, seleccionar aquellos métodos y librerías, tanto secuenciales como paralelas, que resulten más eficientes para resolver los SELD de los casos de estudio abordados.
- Investigar los fundamentos de los métodos iterativos multipaso y su aplicación, a fin de **proponer métodos iterativos multipaso** que busquen resolver eficientemente los sistemas de ecuaciones lineales dispersos que surgen de la discretización de la EDN multigrupo en el caso dinámico.
- Implementar secuencial y paralelamente, utilizando las librerías más eficientes y adecuadas, los métodos iterativos multipaso propuestos para resolver los SELD de los casos de estudio.
- Evaluar el desempeño secuencial y paralelo de los métodos iterativos multipaso propuestos con base en criterios de convergencia y métricas de paralelismo como tiempo de ejecución, speedup y eficiencia.

1.2. Motivación

Los problemas de estabilidad de los reactores tipo BWR han sido una gran preocupación desde los primeros experimentos de su diseño alrededor del año 1950. Generalmente estos reactores producen oscilaciones en la potencia y el caudal, durante su arranque o parada. Tanto en pruebas de estabilidad como en algunos sucesos de inestabilidad durante la operación comercial de estos reactores se han observado dos tipos de oscilaciones. En el primer tipo de oscilaciones detectadas la potencia del reactor varía de la misma forma con una frecuencia cercana a los 0.5 Hz. y se conoce con el nombre de oscilaciones en fase. En el otro tipo de oscilaciones se observa que la potencia de una parte del reactor crece mientras que en la otra parte del reactor decrece manteniéndose la potencia media, aproximadamente constante y se denominan oscilaciones fuera de fase. La detección y predicción de estas oscilaciones mediante una correcta modelización del fenómeno físico es de vital importancia para mejorar la seguridad de los reactores BWR.

En particular se suele utilizar la EDN en la aproximación de dos grupos de energía: el rápido y el térmico; y se supone que los neutrones se producen en el grupo rápido de energía y no hay procesos de dispersión (o trasvase de energía) de neutrones del grupo térmico al rápido. Este modelo consiste en un conjunto de sistemas de ecuaciones

en derivadas parciales lineales con coeficientes variables que hay que integrar para la geometría del núcleo del reactor.

Un primer problema asociado con este modelo, es la resolución de la ecuación de los modos Lambda que es un problema parcial de valores propios generalizado asociado a un operador diferencial no autoadjunto. La obtención de los valores propios dominantes y las correspondientes funciones propias asociadas, tanto a este problema como al problema adjunto, es de interés ya que el valor propio dominante, conocido como constante efectiva del reactor (*keff*) [98], proporciona una idea de la distancia de la configuración estudiada del reactor de una configuración crítica en la que la reacción en cadena se mantiene. La correspondiente función propia (o modo fundamental), describe el estado estacionario del reactor para una configuración dada, siendo el punto de partida para cualquier transitorio que se quiera estudiar, y constituye el segundo problema asociado al modelo [1]. Este segundo problema, que está relacionado con el estudio de transitorios durante el funcionamiento del núcleo del reactor, se caracteriza por la necesidad de resolver eficientemente SELD de gran dimensión durante distintos pasos de tiempo. Sin embargo, recientemente existe el interés por extender estudios de la EDN utilizando 4, 7 o más grupos de energía [23], lo cual hace aun más desafiante los problemas que plantea.

Por tal razón, la resolución rápida y eficiente de los SELD involucrados, el desarrollo de nuevos algoritmos, así como la identificación de todas aquellas herramientas hardware y software más apropiadas para este propósito, será de gran beneficio cuando se intente resolver ambos problemas fundamentales.

1.3. Estado del Arte

Dentro de los métodos de discretización que comúnmente se han utilizado para realizar cálculos estáticos y dinámicos de reactores están los métodos nodales [2], que se clasifican en métodos nodales avanzados analíticos (ANMs) [105], y los métodos nodales basados en desarrollos (NEMs) [72]. Otros métodos son los basados en el desarrollo del flujo neutrónico en polinomios cuyos coeficientes se calculan mediante una técnica de pesado de residuos [77]. Otra aproximación utilizada en programas comerciales consiste en resolver la ecuación de la difusión de un grupo modificada dependiente del tiempo mediante una aproximación cuasi–estática [127].

Se han desarrollado métodos nodales consistentes para el cálculo de transitorios que hacen uso de desarrollos de los flujos neutrónicos y de los flujos transversales en cada nodo en términos de polinomios de Legendre [117]. Dentro de las propiedades importantes de este último, que es un método de colocación nodal lineal, es que es idéntico al método de diferencias finitas centradas en las caras de la red del proceso de discretización [60].

Terminada la etapa de discretización, se hace necesario resolver numéricamente los sistemas de ecuaciones lineales resultantes de este proceso. Para la resolución numérica

de estos sistemas existen métodos directos y métodos iterativos. Los mayoría de los métodos directos para el caso disperso ejecutan una factorización LU [21] [125] [24] sobre la matriz original y tratan de reducir el costo mediante la minimización del relleno (*fill-in*), es decir, la introducción de elementos diferentes de cero, durante el proceso de eliminación en posiciones en que inicialmente habían ceros.

Existen aplicaciones en donde los métodos directos, se han estado utilizando preferentemente sobre los métodos iterativos debido a su robustez y a la precisión de las soluciones encontradas. Sin embargo, los métodos iterativos han demostrado buena competencia con la aparición de métodos tipo Gradiente Conjugado (GC), que combinadas con iteraciones sobre subespacios de Krylov pueden proporcionar procedimientos de propósito general eficientes y sencillos, alcanzando la calidad de sus contrapartes directas. Los métodos iterativos también se han caracterizado por su relativa fácilidad para ser implementados sobre computadoras de alto desempeño, más que los métodos directos [91]. Dentro del conjunto de métodos iterativos para resolver sistemas lineales dispersos de gran dimensión, se encuentran:

- Los métodos iterativos clásicos como [126] [113]: Jacobi, Gauss–Seidel y Sobrerelajación.

- Los métodos iterativos basados en subespacios de Krylov, como: Gradiente Conjugado (GC) [66] [76], Residuo Mínimo [84] (basado en el Método de Lanczos), Residuo Mínimo Generalizado (GMRES) [93] [122] (basado en el algoritmo de Arnoldi [7] [76] [91] [53]).

- Métodos multinivel, que intentan mejorar la eficiencia de los métodos iterativos clásicos en función del uso de diferentes tamaños de mallas en la discretización del problema tratado [12] [58] [73] [86].

- Otros métodos basados en subespacios de Krylov para matrices no simétricas: Gradiente Biconjugado (BCG) [31], Gradiente Conjugado Cuadrado (CGS) [107], Residuo Cuasi–Mínimo Libre Transpuesto (TFQMR) [38] [39], Gradiente Biconjugado Estabilizado (BICGSTAB) [112] y GMRES reiniciado [93].

Se pueden encontrar trabajos de investigación donde se emplean métodos iterativos para resolver diversos problemas. Por ejemplo, en [82] se propone un algoritmo paralelo basado en una versión asíncrona de un esquema iterativo de segundo grado; en [6] se propone un conjunto de métodos iterativos no estacionarios paralelos para la resolución de sistemas no lineales; en [16] se pueden encontrar métodos iterativos paralelos basados en multipartición para resolver sistemas lineales hermíticos y definidos positivos.

El problema estacionario de la EDN ha recibido gran atención por su importancia como paso previo que hay que resolver, para estudios de estabilidad y seguridad en reactores nucleares. El problema estacionario, implica en realidad un problema de valores propios generalizado. Este problema ha dado lugar a estrategias de resolución aplicadas con éxito en métodos como Jacobi–Davidson [115], técnicas basadas en la Iteración del Subespacio [120] [119], Arnoldi con reinicio implícito [62] [63] [108] así como métodos basados en Homotopía [65].

Por otro lado, entre los trabajos que intentan resolver la EDN dependiente del tiempo, usando códigos de dominio público, se encuentran los que usan los códigos DASPK [13] y FCVODE [67]. Estos códigos han sido utilizados como base en trabajos como [40] [41], donde se aplican a casos pequeños 2D y 3D; o en trabajos como [42], donde se describen las actividades que se están llevando a cabo en la Universidad Politécnica de Valencia, para acelerar y extender códigos para la solución de la EDN modelada con 2 grupos de energía en casos 3D grandes de reactores reales.

Es importante mencionar que la técnica de discretización utilizada dicta, muchas de la veces, los métodos numéricos que serán utilizados para resolver los sistemas de ecuaciones que surgen. Por ejemplo, en [102] la EDN es discretizada usando un método de volumen finito mixto de celdas centradas (NEM–M0) en el espacio y un método que combina Crank-Nicholson así como un método de diferencias finitas hacia atrás de dos pasos (BDF–2) en el tiempo, donde los sistemas de ecuaciones que surgen son resueltos con técnicas multimalla, así como con el método del Gradiente Biconjungado estabilizado (BICGSTAB) precondicionado.

Sin embargo, un problema latente es que conforme los problemas 2D y 3D de la EDN se modelan con más de 2 grupos de energía, los sistemas de ecuaciones dispersos, como producto de la discretización, son de mayor dimensión, creando nuevos desafíos y la necesidad de desarrollar e implementar nuevos métodos numéricos. Por esta razón, el presente trabajo propone la utilización y paralelización de métodos multipaso [126], que aprovechen las características de los bloques que conforman los sistemas de ecuaciones a resolver.

La aplicación de métodos multipaso a problemas relacionados con la resolución de los sistemas dispersos de la EDN dependiente del tiempo puede encontrarse en trabajos como [14], donde se exponen los resultados de la aplicación de estos métodos a problemas dinámicos de pequeños reactores 2D y 3D.

1.4. Metodología e Hipótesis

La metodología del trabajo ha estado fundamentada especialmente sobre actividades como: lectura de artículos científico–técnicos y tesis de doctorado relacionados con la EDN y los problemas asociados a ella (problema estacionario y dinámico); investigación

sobre los fundamentos y características principales de los métodos de resolución de SELD (métodos directos e iterativos); investigación, práctica y uso de librerías numéricas secuenciales y paralelas que manipulen matrices dispersas; aplicación y diseño de métodos iterativos multipaso; así como aprendizaje y entrenamiento en el uso de recursos hardware y software disponibles en centros educativos como la Universidad Politécnica de Valencia y otros centros de investigación como el National Energy Research Scientific Computing Center (NERSC) en los Estados Unidos de Norteamérica.

1.4.1. Métricas de Evaluación

Para la evaluación secuencial y paralela de los métodos implementados en esta tesis se han utilizado las siguientes métricas que son [74]:

- **Tiempo de ejecución,** que en esta tesis se denotará con T_s el tiempo de ejecución secuencial mejor y con T_1 el tiempo de ejecución del programa paralelo con $p = 1$ procesador y, en general, T_p el tiempo de ejecución paralelo con p procesadores.
- **Aceleración de ejecución (Speedup),** que mide la ganancia de velocidad de ejecución del algoritmo paralelo con respecto al mejor algoritmo secuencial que resuelve el mismo problema. El speedup, por tanto, está dado por:
$$S_p = \frac{T_s}{T_p};$$
idealmente se espera que el Speedup sea igual al número de procesadores empleados [74].
- **Eficiencia,** que mide el porcentaje del tiempo de ejecución que los procesadores se mantienen útilmente empleados. La eficiencia se expresa como:
$$E_p = \frac{S_p}{p};$$
donde la eficiencia ideal es igual a 1 [74].

1.4.2. Hipótesis

La hipótesis planteada en esta tesis es que el uso de métodos iterativos basados en subespacios de Krylov y métodos iterativos multipaso combinados con el poder que ofrecen las herramientas de computación de altas prestaciones pueden ayudar a resolver eficientemente los SELD de gran dimensión que surgen de la discretización de la EDN, tanto en el caso estacionario como en el dinámico. Aunado a esto, y dada la estructura a bloques del problema, es posible aprovechar las características de los métodos iterativos

multipaso, para realizar simulaciones de este tipo con más de dos grupos de energía en un futuro cercano, lo que redundará en simulaciónes más realistas y complejas que permitirán un mejor conocimiento de los aspectos de seguridad y control en la industria nuclear.

1.5. Organización de la Tesis

Este documento está organizado como sigue: en el capítulo 2 se plantean los problemas asociados a la resolución de la EDN que se abordan en esta tesis: por una parte, la resolución de los sistemas de ecuaciones lineales de gran dimensión que aparecen en la solución de la ecuación de los modos Lambda, resultado de discretizar la EDN con dos grupos de energía en estado estacionario; por otra parte, la resolución de un sistema de ecuaciones lineales disperso (SELD) de gran dimensión para el estudio de transitorios, resultado de discretizar la EDN en el tiempo.

En el capítulo 3 se presenta una descripción general de los diversos métodos y precondicionadores para la resolución de SELD: métodos directos, métodos iterativos (estacionarios, no estacionarios) así como también los métodos multipaso.

En el capítulo 4 se especifican las plataformas computacionales, tanto secuenciales como paralelas, utilizadas en los experimentos numéricos para resolver los problemas antes mencionados. Además, se introducen las librerías numéricas de libre distribución, tanto secuenciales como paralelas, que serán aplicadas y evaluadas a lo largo del presente trabajo.

La presentación, desempeño y análisis de los distintos experimentos secuenciales y paralelos realizados con las librerías numéricas para resolver los sistemas de ecuaciones dispersos asociados a la EDN en su forma estacionaria, así como los resultados obtenidos de la aplicación de los distintos métodos multipaso propuestos por este trabajo de tesis para el caso dinámico de la EDN multigrupo se muestran en el capítulo 5. Además, se presentan los coeficientes de desempeño paralelo obtenidos de las diferentes pruebas aquí descritas.

El capítulo 6 muestra los resultados de un estudio, basado en ciertas suposiciones acerca de los flujos neutrónicos, de la estructura que tendrían las matrices asociadas a la EDN, tanto en el caso estacionario como dinámico, utilizando más de dos grupos de energía. En particular, se presenta un estudio de las estrategias y algoritmos de paralelización que pueden utilizarse para el caso hipotético con cuatro grupos de energía.

Por último, en el capítulo 7 se presentan las conclusiones finales y trabajos futuros que se derivan de este trabajo de tesis.

Capítulo 2

Planteamiento de los Problemas asociados a la Ecuación de Difusión Neutrónica

2.1. Introducción

Este capítulo tiene como objetivo presentar la ecuación de difusión neutrónica (EDN) dependiente del tiempo utilizada para estudiar fenómenos relacionados con la distribución de neutrones en el interior del núcleo de un reactor. La distribución de neutrones en un reactor nuclear, es función de la posición, la energía y el tiempo. La ecuación constituye un sistema de ecuaciones en derivadas parciales, por lo que su resolución numérica precisa de métodos de discretización, tanto para la parte espacial como para la parte temporal. Mucha de la información mostrada en este capítulo puede encontrarse en forma más detallada en [81] [47] [117] [116].

El presente capítulo se ha organizado de la siguiente manera. La sección 2.2 presenta los fundamentos de la EDN. En la sección 2.3, se presenta el problema estacionario relacionado con la EDN, conocido también como el problema de la ecuación de los modos Lambda. En la sección 2.4, se presenta el problema asociado al caso dinámico de la EDN y que es fundamental para estudios de estabilidad y seguridad en los reactores nucleares. Por último, en la sección 2.5 se enuncian algunas conclusiones del capítulo.

2.2. Ecuación de Difusión Neutrónica

En la teoría del transporte, el neutrón es considerado como una partícula puntual, que puede describirse completamente conociendo su posición y su velocidad. Desde un punto de vista macroscópico, se estudia la interacción entre los neutrones y los núcleos atómicos, ignorando los detalles del proceso de interacción dentro del núcleo y considerando que ésta se produce instantáneamente. Aunado a esto, se definen secciones eficaces asociadas a la probabilidad de que tenga lugar un determinado tipo de reacción, tal como: la captura de neutrones, la dispersión elástica de los mismos, etc.

La ecuación de balance en un volumen de control $dVdEd\Omega$ del espacio de fases, se obtiene considerando cómo varía la densidad de neutrones dentro del volumen con respecto al tiempo, la cual está en función de la proporción de neutrones que entran menos la proporción de neutrones que salen del volumen. Para esto, es necesario describir la población de neutrones, que se hace mediante la *densidad angular de neutrones*, que es una magnitud denotada con $N(\vec{r}, E, \Omega, t)$ y definida como el número esperado de neutrones en la posición $\vec{r}$, con dirección Ω, y energía E, en el tiempo t, por unidad de volumen, unidad de ángulo sólido y unidad de energía. Ahora, en función de la densidad angular de neutrones, puede definirse el *flujo neutrónico angular* como:

$$\Phi(\vec{r}, E, \Omega, t) \equiv vN(\vec{r}, E, \Omega, t),$$

donde v es el módulo de la velocidad. Así, la ecuación de balance en el volumen de control da lugar a la ecuación de transporte de neutrones que se expresa como:

$$\begin{aligned}
\frac{1}{v}\frac{\partial \Phi}{\partial t}(\vec{r}, E, \Omega, t) &= -\vec{u}_\Omega \cdot \vec{\nabla}\Phi(\vec{r}, E, \Omega, t) - \Sigma_T(\vec{r}, E, t)\,\Phi(\vec{r}, E, \Omega, t) + Q(\vec{r}, E, \Omega, t) + \\
&+ (1-\beta)\frac{\chi_P(E)}{4\pi}\int_0^\infty dE'\nu\Sigma_F(\vec{r}, E', t)\int_{\Omega'} d\Omega'\Phi(\vec{r}, E', \Omega', t) + \\
&+ \int_0^\infty dE'\int_{\Omega'} d\Omega'\Sigma_S(\vec{r}; E', \Omega' \to E, \Omega; t)\,\Phi(\vec{r}, E', \Omega', t) + \\
&+ \sum_{k=1}^{K}\lambda_k\frac{\chi_k(E)}{4\pi}\mathcal{C}_k(\vec{r}, t). \qquad (2.1)
\end{aligned}$$

El primer término de la parte derecha de la ecuación (2.1), representa la advección neta de neutrones fuera del volumen de control (donde $\vec{u}_\Omega$ es el vector unitario en la dirección especificada por Ω); el segundo término describe la proporción a la que salen los neutrones del volumen de control por procesos de absorción o dispersión; en el tercer término, Q representa una posible fuente externa de neutrones; en el cuarto término, la tasa de neutrones de fisión que se producen en el volumen es representada, suponiendo que la distribución de neutrones de fisión es isótropa en todas direcciones; el quinto

término representa la tasa de neutrones que se introducen en el volumen por procesos de dispersión; por último, el sexto término da cuenta de los neutrones diferidos que aparecen en el volumen al decaer los precursores.

En la ecuación (2.1) también se consideran los siguientes parámetros:

$$\begin{aligned}
\Sigma_T &\quad , \quad \text{denota la sección eficaz total,} \\
\Sigma_F &\quad , \quad \text{denota la sección eficaz de fisión,} \\
\chi_P &\quad , \quad \text{es el espectro de neutrones producidos al decaer los precursores,} \\
\Sigma_S\left(\vec{r}; E', \Omega' \to E, \Omega; t\right) &\quad , \quad \text{es la probabilidad de que un neutrón se disperse de un volumen } dV'dE'd\Omega' \text{ a otro } dVdEd\Omega, \\
\beta = \sum_{k=1}^{K} \beta_k &\quad , \quad K \text{ es el número de precursores considerados y} \\
\beta_k &\quad , \quad \text{es la proporción de neutrones de fisión diferidos por la transformación de un precursor tipo } k, \\
\lambda_k &\quad , \quad \text{es la tasa con la que un precursor de tipo } k \text{ decae.}
\end{aligned}$$

La concentración de precursores de neutrones diferidos satisface la ecuación de balance:

$$\frac{\partial \mathcal{C}_k}{\partial t}(\vec{r}, t) = \beta_k \int_0^\infty dE \int_\Omega d\Omega \nu \Sigma_F\left(\vec{r}, E, \Omega, t\right) \Phi\left(\vec{r}, E, \Omega, t\right) - \lambda_k \mathcal{C}_k\left(\vec{r}, t\right), \tag{2.2}$$

donde ν es el número promedio de neutrones que se producen en una fisión, y $k = 1, ..., K$.

Cabe hacer notar que hay una dependencia angular de la dispersión de neutrones, la cual se debe al ángulo formado entre la dirección del neutrón incidente ($\vec{u}_{\Omega'}$) y del neutrón emergente ($\vec{u}_\Omega$). Para eliminar esta dependencia angular:

1. Se desarrolla la sección eficaz de dispersión como:

$$\Sigma_S\left(\vec{r}; E', \Omega' \to E, \Omega; t\right) = \sum_{l=0}^{\infty} \frac{2l+1}{4\pi} \Sigma_{Sl}\left(\vec{r}; E' \to E; t\right) P_l(\mu^*), \tag{2.3}$$

donde $\mu^* = \vec{u}_{\Omega'}$ y P_l son los polinomios de Legendre:

$$P_l(\mu^*) = P_l(\mu)P_l(\mu') + 2\sum_{m=1}^{l} \frac{(l-m)!}{(l+m)!} P_l^m(\mu)P_l^m(\mu')cos(m(\gamma - \gamma')), \tag{2.4}$$

donde γ es el ángulo azimutal y P_l^m los polinomios asociados de Legendre.

2. Se desarrollan armónicos esféricos [114] para expresar $\Phi(\vec{r}, E, \Omega, t)$, en primera aproximación (aproximación P_1) como:

$$\Phi(\vec{r}, E, \Omega, t) = \frac{1}{4\pi}\left[\phi(\vec{r}, E, t) + 3\vec{u}_\Omega \vec{J}(\vec{r}, E, t)\right], \tag{2.5}$$

donde $\phi(\vec{r}, E, t)$ es el flujo neutrónico escalar, definido como:

$$\phi(\vec{r}, E, t) = \int_\Omega d\Omega \Phi(\vec{r}, E, \Omega, t), \tag{2.6}$$

y $\vec{J}(\vec{r}, E, t)$ es la corriente neutrónica definida como

$$\vec{J}(\vec{r}, E, t) = \int_\Omega d\Omega \vec{u}_\Omega \Phi(\vec{r}, E, \Omega, t). \tag{2.7}$$

Con las aproximaciones (2.3) y (2.5) se obtiene la ecuación de transporte en la aproximación P_1:

$$\begin{aligned}
\frac{1}{v}\frac{\partial}{\partial t}\left(\frac{1}{4\pi}\left[\phi + 3\vec{u}_\Omega\vec{J}\right]\right) &= -\left(\vec{u}_\Omega\vec{\nabla}\right)\frac{1}{4\pi}\left[\phi + 3\vec{u}_\Omega\vec{J}\right] - \Sigma_T\frac{1}{4\pi}\left[\phi + 3\vec{u}_\Omega\vec{J}\right] + \\
&+ Q + (1-\beta)\frac{\chi_P}{4\pi}\int_0^\infty dE'\nu\Sigma_F\int_{\Omega'}d\Omega'\frac{1}{4\pi}\left[\phi + 3\vec{u}_{\Omega'}\vec{J}\right] + \\
&+ \int_0^\infty dE'\int_{\Omega'}d\Omega'\left(\sum_{l=0}^\infty\frac{2l+1}{4\pi}\Sigma_{Sl}(\vec{r}, E'\to E, t)\,P_l(\mu^*)\right) \\
&\quad \frac{1}{4\pi}\left[\phi + 3\vec{u}_{\Omega'}\vec{J}\right] + \\
&+ \sum_{k=1}^K \lambda_k\frac{\chi_k}{4\pi}\mathcal{C}_k.
\end{aligned} \tag{2.8}$$

Ahora, para obtener la ecuación para (2.6), se integra (2.8) para todas las posibles direcciones de Ω, para esto:

1. Se consideran las identidades [81] [121] [9] [47]:

$$\begin{aligned}
\int_\Omega d\Omega &= 4\pi, \\
\int_\Omega \vec{u}_\Omega d\Omega &= 0, \\
\int_\Omega \vec{u}_\Omega\left(\vec{u}_\Omega\vec{A}\right)d\Omega &= \frac{4\pi}{3}\vec{A}, \\
\int_\Omega \left(\vec{u}_\Omega\vec{A}\right)\left(\vec{u}_\Omega\vec{B}\right)d\Omega &= \frac{4\pi}{3}\vec{A}\vec{B};
\end{aligned}$$

2. Se introduce una fuente de neutrones promedio:

$$\int_{\Omega} d\Omega Q\left(\vec{r}, E, \Omega, t\right) = \tilde{Q}\left(\vec{r}, E, t\right);$$

3. Y se consideran las relaciones de ortogonalidad de los polinomios y de los polinomios asociados de Legendre;

llegando así a que [81] [121] [47] [123]:

$$\begin{aligned}
\frac{1}{v}\frac{\partial \phi}{\partial t}\left(\vec{r}, E, t\right) &= -\vec{\nabla}\ \vec{J}\left(\vec{r}, E, t\right) - \Sigma_T\left(\vec{r}, E, t\right)\phi\left(\vec{r}, E, t\right) + \tilde{Q}\left(\vec{r}, E, t\right) + \\
&+ (1-\beta)\chi_P(E)\int_0^{\infty} dE'\nu\Sigma_F\left(\vec{r}, E', t\right)\Phi\left(\vec{r}, E', t\right) + \\
&+ \int_0^{\infty} dE'\Sigma_{S0}\left(\vec{r}, E' \rightarrow E, t\right)\phi\left(\vec{r}, E', t\right) \\
&+ \sum_{k=1}^{K}\lambda_k\chi_k(E)\mathcal{C}_k\left(\vec{r}, t\right). \qquad (2.9)
\end{aligned}$$

Ahora bien, para obtener la ecuación para (2.7), se multiplica (2.8) por $\vec{u}_{\Omega}$ y se integra para toda dirección de Ω, obteniendo así [81] [121] [123]:

$$\begin{aligned}
\frac{1}{v}\frac{\partial \vec{J}}{\partial t}\left(\vec{r}, E, t\right) &= -\frac{1}{3}\vec{\nabla}\phi\left(\vec{r}, E, t\right) - \Sigma_T\left(\vec{r}, E, t\right)\vec{J}\left(\vec{r}, E, t\right) + \\
&+ \int_0^{\infty} dE'\Sigma_{S1}\left(\vec{r}, E' \rightarrow E, t\right)\vec{J}\left(\vec{r}, E', t\right). \qquad (2.10)
\end{aligned}$$

Finalmente, considerando:

1. La aproximación:

$$\frac{\partial \vec{J}}{\partial t}\left(\vec{r}, E, t\right) = \vec{0};\ \text{y}$$

2. Que la dispersión inelástica de los neutrones es isótropa, es decir que Σ_{S1} describe sólamente la dispersión elástica, entonces la dispersión elástica anisótropa se da sin cambio en la energía de los neutrones y, por lo tanto, se puede expresar como:

$$\int_0^{\infty} dE'\Sigma_{S1}\left(\vec{r}, E' \rightarrow E, t\right)\vec{J}\left(\vec{r}, E', t\right) = \tilde{\Sigma}_{S1}\left(\vec{r}, E, t\right)\vec{J}\left(\vec{r}, E, t\right);$$

3. Que la definición de la sección eficaz de transporte es:

$$\Sigma_{tr}\left(\vec{r}, E, t\right) = \Sigma_T\left(\vec{r}, E, t\right) - \tilde{\Sigma}_{S1}\left(\vec{r}, E, t\right);$$

4. Y que el coeficiente de difusión está definido como:

$$D\left(\vec{r},E,t\right)=\frac{1}{3\Sigma_{tr}\left(\vec{r},E,t\right)};$$

las ecuaciones (2.6) y (2.7) se pueden reducir a la EDN:

$$\begin{aligned}\frac{1}{v}\frac{\partial\phi_g\left(\vec{r},E,t\right)}{\partial t} &= -\vec{\nabla}\left(D(\vec{r},E,t)\vec{\nabla}\phi(\vec{r},E,t)\right)-\Sigma_T(\vec{r},E,t)\phi(\vec{r},E,t)+\\ &+ \int_0^\infty dE'\Sigma_{S0}(\vec{r},E'\rightarrow E,t)\phi(\vec{r},E',t)+\\ &+ (1-\beta)\chi_P(E)\int_0^\infty dE'\nu\Sigma_F(\vec{r},E',t)\phi(\vec{r},E',t)+\\ &+ \sum_{k=1}^{K}\lambda_k\chi_k(E)\mathcal{C}_k(\vec{r},t)+\tilde{Q}\left(\vec{r},E,t\right). \qquad (2.11)\end{aligned}$$

Para simplificar (2.11), se utiliza una *aproximación multigrupo* [47] [123], que consiste en obtener una ecuación para los neutrones cuya energía esté comprendida en cada intervalo $[E_g,E_{g+1}]$, $g=1,...,G-1$, donde G es el número de grupos de energía considerados. Esto es posible porque, en general, las secciones eficaces están en función de la energía de los neutrones [47].

Antes de simplificar, se definen las magnitudes asociadas al grupo de energía g:

$$\begin{aligned}\phi_g\left(\vec{r},t\right) &= \int_{E_g}^{E_{g+1}}dE\phi\left(\vec{r},E,t\right),\\ \frac{1}{v_g} &= \int_{E_g}^{E_{g+1}}dE\frac{1}{v}\frac{\phi\left(\vec{r},E,t\right)}{\phi_g\left(\vec{r},t\right)},\\ \Sigma_{T_g}\left(\vec{r},t\right) &= \int_{E_g}^{E_{g+1}}dE\Sigma_T\left(\vec{r},E,t\right)\frac{\phi\left(\vec{r},E,t\right)}{\phi_g\left(\vec{r},t\right)},\\ \nu\Sigma_{fg}\left(\vec{r},t\right) &= \int_{E_g}^{E_{g+1}}dE\nu\Sigma_F\left(\vec{r},E,t\right)\frac{\phi\left(\vec{r},E,t\right)}{\phi_g\left(\vec{r},t\right)},\\ \tilde{Q}_g\left(\vec{r},t\right) &= \int_{E_g}^{E_{g+1}}dE\tilde{Q}\left(\vec{r},E,t\right),\\ \chi_{pg} &= \int_{E_g}^{E_{g+1}}dE\chi_P\left(E\right),\\ \chi_{kg} &= \int_{E_g}^{E_{g+1}}dE\chi_k\left(E\right),\\ \Sigma_{g'g}\left(\vec{r},t\right) &= \int_{E_{g'}}^{E_{g'+1}}dE'\int_{E_g}^{E_{g+1}}dE\Sigma_{S0}\left(\vec{r},E\rightarrow E',t\right)\frac{\phi\left(\vec{r},E',t\right)}{\phi_{g'}\left(\vec{r},t\right)};\end{aligned}$$

y el coeficiente de difusión del grupo g por cada dirección espacial j:

$$D_g(\vec{r},t) = \int_{E_g}^{E_{g+1}} dE D(\vec{r},E,t)\,\frac{\partial_j \phi(\vec{r},E,t)}{\partial_j \phi_g(\vec{r},t)}.$$

La sección eficaz total se escribe como suma de un término de absorción y términos de dispersión como sigue:

$$\Sigma_{Tg}(\vec{r},t) = \Sigma_{ag}(\vec{r},t) + \sum_{g'=1}^{G} \Sigma_{g'g}(\vec{r},t),$$

así, la sección eficaz de dispersión del grupo g es:

$$\Sigma_{Sg}(\vec{r},t) = \sum_{g'\neq g}^{G} \Sigma_{g'g}(\vec{r},t).$$

Integrando la ecuación (2.11) entre E_g y E_{g+1}, y utilizando las definiciones anteriores, se obtiene la EDN para el grupo g, que se expresa como:

$$\begin{aligned}\frac{1}{v_g}\frac{\partial \phi_g(\vec{r},t)}{\partial t} &= \vec{\nabla}\left(D_g(\vec{r},t)\vec{\nabla}\phi_g(\vec{r},t)\right) - \left(\Sigma_{ag}(\vec{r},t) + \Sigma_{Sg}(\vec{r},t)\right)\phi_g(\vec{r},t) + \\ &+ \sum_{g'\neq g}^{G} \Sigma_{g'g}(\vec{r},t)\phi_{g'}(\vec{r},t) + (1-\beta)\chi_{pg}\sum_{g'=1}^{G}\nu\Sigma_{fg'}(\vec{r},t)\phi_{g'}(\vec{r},t) + \\ &+ \sum_{k=1}^{K}\lambda_k\chi_{kg}\mathcal{C}_k(\vec{r},t) + \tilde{Q}(\vec{r},t); \qquad (2.12)\end{aligned}$$

mientras que la ecuación de la concentración de precursores para el grupo g es:

$$\frac{\partial \mathcal{C}_k(\vec{r},t)}{\partial t} = \beta_k \sum_{g=1}^{G} \nu\Sigma_{fg}(\vec{r},t)\,\phi_g(\vec{r},t) - \lambda_k \mathcal{C}_k(\vec{r},t). \qquad (2.13)$$

De esta forma, con las ecuaciones (2.12) y (2.13) se aproximan las ecuaciones (2.1) y (2.2), reduciendo el espacio de trabajo de $(\vec{r},E,\Omega,t)$ a $(\vec{r},t)$.

En este trabajo de tesis, se considera el espectro de energía divido en dos grupos [47] [48]:

Grupo rápido (grupo 1), representado con $\phi_1(\vec{r},t)$,

Grupo térmico (grupo 2), representado con $\phi_2(\vec{r},t)$;

así, las ecuaciones de difusión neutrónica para estos dos grupos son:

$$\begin{aligned}\frac{1}{v_1}\frac{\partial\phi_1(\vec{r},t)}{\partial t} &- \vec{\nabla}\left(D_1(\vec{r},t)\vec{\nabla}\phi_1(\vec{r},t)\right) + \left(\Sigma_{a1}(\vec{r},t)+\Sigma_{s1}(\vec{r},t)\right)\phi_1(\vec{r},t) = \\ &= \Sigma_{21}(\vec{r},t)\phi_2(\vec{r},t) + (1-\beta)\chi_{p1}\left(\nu\Sigma_{f1}(\vec{r},t)\phi_1(\vec{r},t)+\nu\Sigma_{f2}(\vec{r},t)\phi_2(\vec{r},t)\right) + \\ &+ \sum_{k=1}^{K}\lambda_k\chi_{k1}\mathcal{C}_k(\vec{r},t) + \tilde{Q}\left(\vec{r},t\right) \qquad (2.14)\end{aligned}$$

y

$$\begin{aligned}\frac{1}{v_2}\frac{\partial\phi_2(\vec{r},t)}{\partial t} &- \vec{\nabla}\left(D_2(\vec{r},t)\vec{\nabla}\phi_2(\vec{r},t)\right) + \left(\Sigma_{a2}(\vec{r},t)+\Sigma_{s2}(\vec{r},t)\right)\phi_2(\vec{r},t) = \\ &= \Sigma_{12}(\vec{r},t)\phi_1(\vec{r},t) + (1-\beta)\chi_{p2}\left(\nu\Sigma_{f1}(\vec{r},t)\phi_1(\vec{r},t)+\nu\Sigma_{f2}(\vec{r},t)\phi_2(\vec{r},t)\right) + \\ &+ \sum_{k=1}^{K}\lambda_k\chi_{k2}\mathcal{C}_k(\vec{r},t) + \tilde{Q}\left(\vec{r},t\right). \qquad (2.15)\end{aligned}$$

Considerando que no hay fuente externa de neutrones $\left(\tilde{Q}\left(\vec{r},t\right)=0\right)$ en (2.14) y (2.15), y que no hay procesos de dispersión (o trasvase de energía) del grupo térmico (grupo 2) al grupo rápido (grupo 1) ($\Sigma_{21}\left(\vec{r},t\right)=0$ en (2.14)), (2.14) y (2.15) pueden escribirse vectorialmente como:

$$\begin{aligned}\begin{bmatrix} \frac{1}{v_1}\frac{\partial\phi_1(\vec{r},t)}{\partial t} \\ \frac{1}{v_2}\frac{\partial\phi_2(\vec{r},t)}{\partial t}\end{bmatrix} &+ \begin{bmatrix} \left(-\vec{\nabla}D_1(\vec{r},t)\vec{\nabla}+\Sigma_{a1}(\vec{r},t)+\Sigma_{12}(\vec{r},t)\right)\phi_1(\vec{r},t) \\ \left(-\vec{\nabla}D_2(\vec{r},t)\vec{\nabla}+\Sigma_{a2}(\vec{r},t)\right)\phi_2(\vec{r},t)-\Sigma_{12}\phi_1(\vec{r},t)\end{bmatrix} = \\ &= \begin{bmatrix} (1-\beta)\chi_{p1}\left(\nu\Sigma_{f1'}(\vec{r},t)\phi_1(\vec{r},t)+\nu\Sigma_{f2}(\vec{r},t)\phi_2(\vec{r},t)\right) \\ (1-\beta)\chi_{p2}\left(\nu\Sigma_{f1'}(\vec{r},t)\phi_1(\vec{r},t)+\nu\Sigma_{f2}(\vec{r},t)\phi_2(\vec{r},t)\right)\end{bmatrix} + \\ &+ \begin{bmatrix} \sum_{k=1}^{K}\lambda_k\chi_{k1}\mathcal{C}_k(\vec{r},t) \\ \sum_{k=1}^{K}\lambda_k\chi_{k2}\mathcal{C}_k(\vec{r},t)\end{bmatrix}. \qquad (2.16)\end{aligned}$$

Considerando también que no se producen neutrones en el grupo térmico (grupo 2) ($\chi_{p2}=\chi_{k2}=0$), la expresión matricial de (2.16) es:

$$\begin{aligned}&\begin{bmatrix} \frac{1}{v_1} & 0 \\ 0 & \frac{1}{v_2}\end{bmatrix}\begin{bmatrix} \frac{\partial\phi_1(\vec{r},t)}{\partial t} \\ \frac{\partial\phi_2(\vec{r},t)}{\partial t}\end{bmatrix} + \\ &+\begin{bmatrix} -\vec{\nabla}D_1(\vec{r},t)\vec{\nabla}+\Sigma_{a1}(\vec{r},t)+\Sigma_{12}(\vec{r},t) & 0 \\ -\Sigma_{12}(\vec{r},t) & -\vec{\nabla}D_2(\vec{r},t)\vec{\nabla}+\Sigma_{a2}(\vec{r},t)\end{bmatrix}\begin{bmatrix} \phi_1(\vec{r},t) \\ \phi_2(\vec{r},t)\end{bmatrix} = \\ &\qquad = (1-\beta)\begin{bmatrix} \nu\Sigma_{f1}(\vec{r},t) & \nu\Sigma_{f2}(\vec{r},t) \\ 0 & 0\end{bmatrix}\begin{bmatrix} \phi_1(\vec{r},t) \\ \phi_2(\vec{r},t)\end{bmatrix} + \sum_{k=1}^{K}\lambda_k\mathcal{C}_k\begin{bmatrix} 1 \\ 0\end{bmatrix},\end{aligned}$$

que al renombrar términos queda como:

$$\left[v^{-1}\right]\dot{\Phi} + \mathcal{L}\Phi = (1-\beta)\mathcal{M}\Phi + \sum_{k=1}^{K}\lambda_k\mathcal{C}_k\chi. \tag{2.17}$$

Por otra parte, la ecuación de la concentración de precursores para dos grupos es:

$$\frac{\partial\mathcal{C}_k(\vec{r},t)}{\partial t} = \beta_k\left(\nu\Sigma_{f1}\left(\vec{r},t\right)\phi_1\left(\vec{r},t\right) + \nu\Sigma_{f2}\left(\vec{r},t\right)\phi_2\left(\vec{r},t\right)\right) - \lambda_k\mathcal{C}_k\left(\vec{r},t\right)$$

que vectorialmente se expresa como:

$$\frac{\partial\mathcal{C}_k(\vec{r},t)}{\partial t} = \beta_k\left[\begin{array}{cc}\nu\Sigma_{f1}\left(\vec{r},t\right) & \nu\Sigma_{f2}\left(\vec{r},t\right)\end{array}\right]\left[\begin{array}{c}\phi_1\left(\vec{r},t\right)\\ \phi_2\left(\vec{r},t\right)\end{array}\right] - \lambda_k\mathcal{C}_k\left(\vec{r},t\right)$$

y renombrando queda como:

$$\dot{\mathcal{C}}_k = \beta_k\left[\begin{array}{cc}\nu\Sigma_{f1} & \nu\Sigma_{f2}\end{array}\right]\Phi - \lambda_k\mathcal{C}_k. \tag{2.18}$$

En la integración de las ecuaciones dinámicas representadas en (2.17) y (2.18), se utiliza un desarrollo de la solución en términos de los modos Lambda del reactor, por lo que la determinación de estos modos, se considera un problema previo para el estudio de las características de la EDN en su forma dinámica. La siguiente sección aborda el primer problema.

2.3. Ecuación de los Modos Lambda. Problema Estacionario

Ahora bien, resulta fundamental poder estudiar el comportamiento de un reactor en estado crítico. Se dice que un reactor está en estado crítico cuando la proporción a la que se producen los neutrones en su interior es igual a la proporción a la que se pierden [47] [98]. En estas condiciones el reactor se encuentra en estado estacionario.

Así pues, para estudiar el estado estacionario de un reactor dado, se puede forzar la criticidad del mismo de un modo artificial, multiplicando las secciones eficaces, relacionados con la producción de los neutrones por procesos de fisión, por un número λ (o bien dividiéndolas por un número k, λ=1/k) [47]. De esta forma, se espera que exista un número λ que satisfaga las ecuaciones (2.17) y (2.18), mismas que pueden reescribirse de la siguiente manera:

$$\mathcal{L}\Phi = \lambda(1-\beta)M\Phi + \sum_{k=1}^{K}\lambda_k\mathcal{C}_k\chi \tag{2.19}$$

$$0 = \lambda\beta_k\left[\begin{array}{cc}\nu\Sigma_{f1} & \nu\Sigma_{f2}\end{array}\right]\Phi - \lambda_k\mathcal{C}_k; \tag{2.20}$$

así, para el caso de usar $G = 2$ grupos de energía, la expresión (2.19) se puede expresar como:

$$\begin{aligned}\begin{bmatrix} -\nabla D_1 \nabla + \Sigma_{a1} + \Sigma_{12} & 0 \end{bmatrix} \Phi &= \lambda(1-\beta) \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi + \\ &+ 1 \sum_{k=1}^{K} \lambda_k \mathcal{C}_k \qquad (2.21)\\ \begin{bmatrix} -\Sigma_{12} & -\nabla D_2 \nabla + \Sigma_{a2} \end{bmatrix} \Phi &= \lambda(1-\beta) \begin{bmatrix} 0 & 0 \end{bmatrix} \Phi + \\ &+ 0 \sum_{k=1}^{K} \lambda_k \mathcal{C}_k \qquad (2.22)\end{aligned}$$

y la expresión en (2.20) como:

$$\lambda_k \mathcal{C}_k = \lambda \beta_k \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi; \qquad (2.23)$$

sustituyendo (2.23) en (2.21) y (2.22), éstas quedan como:

$$\begin{aligned}\begin{bmatrix} -\nabla D_1 \nabla + \Sigma_{a1} + \Sigma_{12} & 0 \end{bmatrix} \Phi &= \lambda \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi - \lambda\beta \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi + \\ &+ 1 \sum_{k=1}^{K} \lambda\beta_k \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi \qquad (2.24)\\ \begin{bmatrix} -\Sigma_{12} & -\nabla D_2 \nabla + \Sigma_{a2} \end{bmatrix} \Phi &= \lambda \begin{bmatrix} 0 & 0 \end{bmatrix} \Phi - \lambda\beta \begin{bmatrix} 0 & 0 \end{bmatrix} \Phi + \\ &+ 0 \sum_{k=1}^{K} \lambda\beta_k \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi; \qquad (2.25)\end{aligned}$$

las cuales, como $\beta = \sum_{k=1}^{K} \beta_k$, se expresan como:

$$\begin{aligned}\begin{bmatrix} -\nabla D_1 \nabla + \Sigma_{a1} + \Sigma_{12} & 0 \end{bmatrix} \Phi &= \lambda \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi - \lambda\beta \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi + \\ &+ 1\lambda\beta \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi \qquad (2.26)\\ \begin{bmatrix} -\Sigma_{12} & -\nabla D_2 \nabla + \Sigma_{a2} \end{bmatrix} \Phi &= \lambda \begin{bmatrix} 0 & 0 \end{bmatrix} \Phi - \lambda\beta \begin{bmatrix} 0 & 0 \end{bmatrix} \Phi + \\ &+ 0\lambda\beta \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \end{bmatrix} \Phi, \qquad (2.27)\end{aligned}$$

obteniendo así la ecuación de los modos Lambda del reactor:

$$\mathcal{L}\Phi = \lambda \mathcal{M} \Phi, \qquad (2.28)$$

que es un problema de valores propios generalizado, asociado al operador diferencial $\mathcal{L}$,

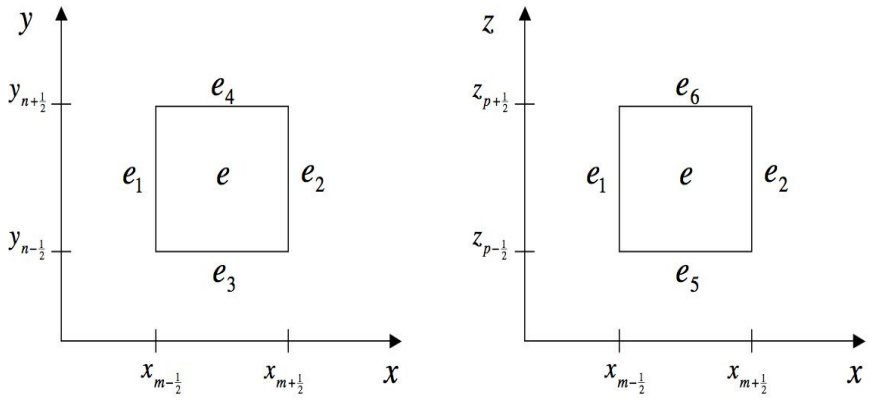

Figura 2.1: Posición de los nodos adyascentes al nodo e.

donde

$$\begin{aligned}
\mathcal{L} &= \begin{bmatrix} -\nabla D_1 \nabla + \Sigma_{a1} + \Sigma_{12} & 0 \\ -\Sigma_{12} & -\nabla D_2 \nabla + \Sigma_{a2} \end{bmatrix}, \\
\Phi &= \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}, \\
\mathcal{M} &= \begin{bmatrix} \nu\Sigma_{f1} & \nu\Sigma_{f2} \\ 0 & 0 \end{bmatrix}.
\end{aligned}$$

Los valores propios λ_n asociados a esta ecuación se interpretan como factores multiplicativos de las secciones eficaces de fisión, lo que significa que son números reales y, por lo tanto, las funciones propias Φ_n también son reales.

Por otra parte, existen diversos métodos para discretizar la ecuación (2.28). En [47] se mencionan varios, entre los que se encuentran: métodos en diferencias finitas, métodos nodales, métodos basados en elementos finitos, por mencionar algunos. En este trabajo de tesis, los casos de estudio seleccionados en las pruebas experimentales utilizan métodos nodales basados en desarrollos de polinomios de Legendre de los flujos neutrónicos en cada celda en que se ha discretizado el reactor. Esta discretización es como sigue.

En el caso tridimensional, para un autovalor dado λ y un paralepípedo e (o nodo e, ver Figura 2.1), la ecuación (2.28) es:

$$\begin{aligned}
-\nabla D_{1e} \nabla \phi_{1e} + (\Sigma_{a1,e} + \Sigma_{12,e})\, \phi_{1e} &= \lambda \left(\nu\Sigma_{f1,e}\phi 1e + \nu\Sigma_{f2,e}\phi_{2e}\right) \\
-\nabla D_{2e} \nabla \phi_{2e} + \Sigma_{a2,e}\phi_{2e} &= \Sigma_{12,e}\phi 1e
\end{aligned} \tag{2.29}$$

Para desarrollar el método nodal se considerará una sóla ecuación genérica:

$$-D_{x,e}\frac{\partial^2\phi_e}{\partial x^2} - D_{y,e}\frac{\partial^2\phi_e}{\partial y^2} - D_{x,e}\frac{\partial^2\phi_e}{\partial z^2} + \Sigma_{r,e}\phi_e = S_e(\phi_e) \tag{2.30}$$

Realizando el cambio de variable

$$\begin{aligned} u &= \frac{1}{dx_e}\left[x - \frac{x_{m-\frac{1}{2}} + x_{m+\frac{1}{2}}}{2}\right] \\ v &= \frac{1}{dy_e}\left[y - \frac{y_{n-\frac{1}{2}} + y_{n+\frac{1}{2}}}{2}\right] \\ w &= \frac{1}{dz_e}\left[z - \frac{z_{p-\frac{1}{2}} + z_{p+\frac{1}{2}}}{2}\right] \end{aligned}$$

donde

$$dx_e = x_{m+\frac{1}{2}} - x_{m-\frac{1}{2}},$$

la ecuación (2.30) se escribe como:

$$-\frac{dy_e dz_e}{dx_e}D_{x,e}\frac{\partial^2\phi_e}{\partial u^2} - \frac{dx_e dz_e}{dy_e}D_{y,e}\frac{\partial^2\phi_e}{\partial v^2} - \frac{dx_e dy_e}{dx_e}D_{z,e}\frac{\partial^2\phi_e}{\partial w^2} + \Sigma_{r,e}V_e\phi_e = V_e S_e(\phi_e). \tag{2.31}$$

Suponer que la solución en cada celda (o nodo) e utiliza

$$\phi_e(u,v,w) = \sum_{k1=0}^{K}\sum_{k2=0}^{K}\sum_{k3=0}^{K}\phi_e^{k1,k2,k3}P_{k1}(u)P_{k2}(v)P_{k3}(w) \tag{2.32}$$

$$S_e(u,v,w) = \sum_{k1=0}^{K}\sum_{k2=0}^{K}\sum_{k3=0}^{K}S_e^{k1,k2,k3}P_{k1}(u)P_{k2}(v)P_{k3}(w), \tag{2.33}$$

donde $P_k(u)$ son los polinomios de Legendre ortonormales, es decir, que cumplen

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} P_n(u)P_m(u)du = \sigma_{n,m} = 1, \tag{2.34}$$

definidos como

$$P_0(u) = 1, \tag{2.35}$$

$$P_1(u) = 2\sqrt{3}u, \tag{2.36}$$

$$P_{k+1}(u) = 2\sqrt{\frac{2k+3}{2k+1}}\frac{2k+1}{k+1}uP_k(u) - \sqrt{\frac{2k+3}{2k-1}}\frac{k}{k+1}P_{k-1}(u), \quad k \geq 1, \tag{2.37}$$

que satisfacen:

$$P_n(\frac{1}{2}) = \sqrt{2n+1}, \qquad P_n(-\frac{1}{2}) = (-1)^n\sqrt{2n+1}, \tag{2.38}$$

$$P_n'(\frac{1}{2}) = n(n+1)\sqrt{2n+1}, \qquad P_n'(-\frac{1}{2}) = (-1)^{n+1}n(n+1)\sqrt{2n+1}. \tag{2.39}$$

Entonces, sustituyendo (2.32) y (2.33) en (2.31), ésta se expresa como:

$$
\begin{aligned}
&-\frac{dy_e dz_e}{dx_e} D_{x,e} \frac{\partial^2}{\partial u^2} \sum_{k1=0}^{K}\sum_{k2=0}^{K}\sum_{k3=0}^{K} \phi_e^{k1,k2,k3} P_{k1}(u)P_{k2}(v)P_{k3}(w) - \\
&\quad -\frac{dx_e dz_e}{dy_e} D_{y,e} \frac{\partial^2}{\partial v^2} \sum_{k1=0}^{K}\sum_{k2=0}^{K}\sum_{k3=0}^{K} \phi_e^{k1,k2,k3} P_{k1}(u)P_{k2}(v)P_{k3}(w) - \\
&\quad -\frac{dx_e dy_e}{dx_e} D_{z,e} \frac{\partial^2}{\partial w^2} \sum_{k1=0}^{K}\sum_{k2=0}^{K}\sum_{k3=0}^{K} \phi_e^{k1,k2,k3} P_{k1}(u)P_{k2}(v)P_{k3}(w) + \\
&\quad + \Sigma_{r,e} V_e \sum_{k1=0}^{K}\sum_{k2=0}^{K}\sum_{k3=0}^{K} \phi_e^{k1,k2,k3} P_{k1}(u)P_{k2}(v)P_{k3}(w) = \\
&\quad = V_e \sum_{k1=0}^{K}\sum_{k2=0}^{K}\sum_{k3=0}^{K} S_e^{k1,k2,k3} P_{k1}(u)P_{k2}(v)P_{k3}(w). \quad (2.40)
\end{aligned}
$$

Ahora, multiplicando (2.40) por $W_{k1,k2,k3}(u,v,w) = P_{k1}(u)P_{k2}(v)P_{k3}(w)$, integrando sobre todo el volumen e y utlizando la relación de ortonormalidad de los Polinomios de Legendre ya definidos, se tiene que:

$$
\begin{aligned}
&-\iiint_{-\frac{1}{2}}^{\frac{1}{2}} dudvdw W_{k_1,k_2,k_3}(u,v,w) \frac{dy_e dz_e}{dx_e} D_{x,e} \frac{\partial^2}{\partial u^2} \sum_{k_1=0}^{K}\sum_{k_2=0}^{K}\sum_{k_3=0}^{K} \phi_e^{k_1,k_2,k_3} P_{k_1}(u)P_{k_2}(v)P_{k_3}(w) - \\
&-\iiint_{-\frac{1}{2}}^{\frac{1}{2}} dudvdw W_{k_1,k_2,k_3}(u,v,w) \frac{dx_e dz_e}{dy_e} D_{y,e} \frac{\partial^2}{\partial v^2} \sum_{k_1=0}^{K}\sum_{k_2=0}^{K}\sum_{k_3=0}^{K} \phi_e^{k_1,k_2,k_3} P_{k_1}(u)P_{k_2}(v)P_{k_3}(w) - \\
&-\iiint_{-\frac{1}{2}}^{\frac{1}{2}} dudvdw W_{k_1,k_2,k_3}(u,v,w) \frac{dx_e dy_e}{dx_e} D_{z,e} \frac{\partial^2}{\partial w^2} \sum_{k_1=0}^{K}\sum_{k_2=0}^{K}\sum_{k_3=0}^{K} \phi_e^{k_1,k_2,k_3} P_{k_1}(u)P_{k_2}(v)P_{k_3}(w) + \\
&+\iiint_{-\frac{1}{2}}^{\frac{1}{2}} dudvdw W_{k_1,k_2,k_3}(u,v,w) \Sigma_{r,e} V_e \sum_{k_1=0}^{K}\sum_{k_2=0}^{K}\sum_{k_3=0}^{K} \phi_e^{k_1,k_2,k_3} P_{k_1}(u)P_{k_2}(v)P_{k_3}(w) = \\
&\quad = \iiint_{-\frac{1}{2}}^{\frac{1}{2}} dudvdw W_{k_1,k_2,k_3}(u,v,w) V_e \sum_{k_1=0}^{K}\sum_{k_2=0}^{K}\sum_{k_3=0}^{K} S_e^{k_1,k_2,k_3} P_{k_1}(u)P_{k_2}(v)P_{k_3}(w),
\end{aligned}
$$

que es igual que

$$
\begin{aligned}
-\int_{-\frac{1}{2}}^{\frac{1}{2}} du P_{k1}(u,v,w)\frac{dy_e dz_e}{dx_e} D_{x,e}\frac{d^2}{du^2}\sum_{k1=0}^{K}\phi_e^{k1,k2,k3}P_{k1}(u) - \\
-\int_{-\frac{1}{2}}^{\frac{1}{2}} dv P_{k2}(u,v,w)\frac{dx_e dz_e}{dy_e} D_{y,e}\frac{d^2}{dv^2}\sum_{k2=0}^{K}\phi_e^{k1,k2,k3}P_{k2}(v) - \\
-\int_{-\frac{1}{2}}^{\frac{1}{2}} dw P_{k3}(u,v,w)\frac{dx_e dy_e}{dz_e} D_{z,e}\frac{d^2}{dw^2}\sum_{k3=0}^{K}\phi_e^{k1,k2,k3}P_{k3}(w) + \\
+\Sigma_{r,e}V_e\Phi_e^{k1,k2,k3} = V_e S_e^{k1,k2,k3};
\end{aligned}
$$

así, esta ecuación se puede reescribir como:

$$
\begin{aligned}
-dy_e dz_e\frac{D_{x,e}}{dx_e}\int_{-\frac{1}{2}}^{\frac{1}{2}} du P_{k1}(u,v,w)\frac{d^2}{du^2}\sum_{k=0}^{K}\phi_e^{k,k2,k3}P_k(u) - \\
-dx_e dz_e\frac{D_{y,e}}{dy_e}\int_{-\frac{1}{2}}^{\frac{1}{2}} dv P_{k2}(u,v,w)\frac{d^2}{dv^2}\sum_{k=0}^{K}\phi_e^{k1,k,k3}P_k(v) - \\
-dx_e dy_e\frac{D_{z,e}}{dz_e}\int_{-\frac{1}{2}}^{\frac{1}{2}} dw P_{k3}(u,v,w)\frac{d^2}{dw^2}\sum_{k=0}^{K}\phi_e^{k1,k2,k}P_k(w) + \\
+\Sigma_{r,e}V_e\phi_e^{k1,k2,k3} = V_e S_e^{k1,k2,k3}. \quad (2.41)
\end{aligned}
$$

Ahora, definiendo:

$$
\begin{aligned}
\phi_{ex}^{k2,k3} &= \sum_{k=0}^{K}\phi_e^{k,k2,k3}P_k(u), \\
\phi_{ey}^{k1,k3} &= \sum_{k=0}^{K}\phi_e^{k1,k,k3}P_k(v), \\
\phi_{ez}^{k1,k2} &= \sum_{k=0}^{K}\phi_e^{k1,k2,k}P_k(w),
\end{aligned}
$$

$$
\begin{aligned}
L_{k1} &= \int_{-\frac{1}{2}}^{\frac{1}{2}} du P_{k1}(u,v,w)\frac{d^2}{du^2}\phi_{ex}^{k2,k3}, \\
L_{k2} &= \int_{-\frac{1}{2}}^{\frac{1}{2}} dv P_{k2}(u,v,w)\frac{d^2}{dv^2}\phi_{ey}^{k1,k3}, \\
L_{k3} &= \int_{-\frac{1}{2}}^{\frac{1}{2}} dw P_{k3}(u,v,w)\frac{d^2}{dw^2}\phi_{ez}^{k1,k2},
\end{aligned}
$$

$$\begin{aligned} F_{ex}^{k1,k2,k3} &= \frac{D_{x,e}}{dx_e} L_{k1}, \\ F_{ey}^{k1,k2,k3} &= \frac{D_{y,e}}{dy_e} L_{k2}, \\ F_{ez}^{k1,k2,k3} &= \frac{D_{z,e}}{dx_e} L_{k3}, \end{aligned}$$

la ecuación (2.41) se expresa sintéticamente como:

$$-dy_e dz_e F_{ex}^{k1,k2,k3} - dx_e dz_e F_{ey}^{k1,k2,k3} - dx_e dy_e F_{ez}^{k1,k2,k3} + \Sigma_{r,e} V_e \phi_e^{k1,k2,k3} = V_e S_e^{k1,k2,k3}. \tag{2.42}$$

Por otra parte, las ecuaciones L_{k1}, L_{k2} y L_{k3} tienen la forma

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} du P_k(u) \frac{d^2}{du^2} f(u),$$

donde:

$$f(u) = \sum_{k=0}^{\infty} F_l P_l(u),$$

cuya solución es:

$$\begin{aligned} &\int_{-\frac{1}{2}}^{\frac{1}{2}} du P_k(u) \frac{d^2}{du^2} f(u) = \sqrt{2k+1} \left\{ (-1)^{k+1} \left[k(k+1) f(\frac{-1}{2}) + \frac{d}{du} f(\frac{-1}{2}) \right] - \right. \\ &\left. - \left[k(k+1) f(\frac{1}{2}) - \frac{d}{du} f(\frac{1}{2}) \right] + \sum_{l=0}^{K-2} \left[1 + (-1)^{k+l} \right] \sqrt{2l+1} \left[k(k+1) - l(l+1) \right] F_l \right\}. \end{aligned} \tag{2.43}$$

Considerando la celda (o nodo) e_1, vecina de e (recordar la Figura 2.1), para la frontera en común las condiciones de continuidad establecen que:

$$\phi_{e1} \left(\frac{1}{2}, v, w \right) = \phi_e \left(-\frac{1}{2}, v, w \right), \tag{2.44}$$

y para la corriente se tiene que:

$$\frac{D_{x,e1}}{dx_{e1}} \frac{\partial}{\partial u} \phi_{e1} \left(\frac{1}{2}, v, w \right) = \frac{D_{x,e}}{dx_e} \frac{\partial}{\partial u} \phi_e \left(-\frac{1}{2}, v, w \right). \tag{2.45}$$

Multiplicando (2.44) y (2.45) por $W_{k2,k3} = P_{k2}(v) P_{k3}(w)$ e integrando sobre la superficie común:

$$\iint_{-\frac{1}{2}}^{\frac{1}{2}} P_{k2}(v) P_{k3}(w) \phi_{e1} \left(\frac{1}{2}, v, w \right) dv dw = \iint_{-\frac{1}{2}}^{\frac{1}{2}} P_{k2}(v) P_{k3}(w) \phi_e \left(-\frac{1}{2}, v, w \right) dv dw$$

y

$$\int\int_{-\frac{1}{2}}^{\frac{1}{2}} P_{k2}(v)P_{k3}(w)\frac{D_{x,e1}}{dx_{e1}}\frac{\partial}{\partial u}\phi_{e1}\left(\frac{1}{2},v,w\right)dvdw =$$

$$\int\int_{-\frac{1}{2}}^{\frac{1}{2}} dvP_{k2}(u,v,w)P_{k2}(v)P_{k3}(w)\frac{D_{x,e}}{dx_e}\frac{\partial}{\partial u}\phi_e\left(-\frac{1}{2},v,w\right)dvdw,$$

entonces,

$$\phi_{e1,x}^{k2,k3}\left(\frac{1}{2}\right) = \phi_{e,x}^{k2,k3}\left(-\frac{1}{2}\right), \tag{2.46}$$

$$\frac{D_{x,e1}}{dx_{e1}}\frac{d}{du}\phi_{e1,x}^{k2,k3}\left(\frac{1}{2}\right) = \frac{D_{x,e}}{dx_e}\frac{d}{du}\phi_{e,x}^{k2,k3}\left(-\frac{1}{2}\right). \tag{2.47}$$

De acuerdo a (2.39), (2.43), (2.46) y (2.47) (los detalles pueden verse en [121] [81]), se sabe que:

$$\begin{aligned}
F_{ex}^{k,k2,k3} = \frac{\sqrt{2k+1}}{K(K+1)}\Bigg\{ & \frac{D_{xe}}{dx_e}\left[K(K+1)-k(k+1)\right]\times \\
& \times\sum_{l=0}^{k-1}\left(1+(-1)^{k+k}\right)\sqrt{2l+1}l(l+1)\phi_2^{l,k2,k3}+ \\
& +\frac{D_{xe}}{dx_e}k(k+1)\sum_{l=k}^{K-1}\left(1+(-1)^{k+l}\right)\sqrt{2l+1}\left[K(K+1)-l(l+1)\right]\phi_e^{l,k2,k3}+ \\
& +\left[K(K+1)-k(k+1)\right]\sum_{l=0}^{K-1}\sqrt{2l+1}\left[K(K+1)-l(l+1)\right]\times \\
& \times\Bigg[(-1)^k\frac{Dx_eDx_{e1}}{dx_eDx_{e1}+dx_{e1}Dx_e}\left[(-1)^l\phi_e^{l,k2,k3}-\phi_{e1}^{l,k2,k3}\right]- \\
& -\frac{Dx_eDx_{e2}}{dx_eDx_{e2}+dx_{e1}Dx_e}\left[(-1)^l\phi_{e2}^{l,k2,k3}-\phi_e^{l,k2,k3}\right]\Bigg]\Bigg\};
\end{aligned} \tag{2.48}$$

análogamente se obtienen las expresiones correspondientes para $F_{ey}^{k1,k,k3}$ y $F_{ez}^{k1,k2,k}$, entonces, con base en estas expresiones y renombrando términos se tiene que:

$$F_{ex}^{k,k2,k3} = \sum_{l=0}^{K-1}\left(A_{ex}^{k,l;K}\phi_{e1}^{l,k2,k3} - B_{ex}^{k,l;K}\phi_e^{l,k2,k3} + C_{ex}^{k,l;K}\phi_{e2}^{l,k2,k3}\right) \tag{2.49}$$

$$F_{ey}^{k1,k,k3} = \sum_{l=0}^{K-1}\left(A_{ey}^{k,l;K}\phi_{e3}^{k1,l,k3} - B_{ey}^{k,l;K}\phi_e^{k1,l,k3} + C_{ey}^{k,l;K}\phi_{e4}^{k1,l,k3}\right) \tag{2.50}$$

$$F_{ez}^{k1,k2,k} = \sum_{l=0}^{K-1}\left(A_{ez}^{k,l;K}\phi_{e5}^{k1,k2,l} - B_{ez}^{k,l;K}\phi_e^{k1,k2,l} + C_{ez}^{k,l;K}\phi_{e6}^{k1,k2,l}\right). \tag{2.51}$$

Sustituyendo (2.49) – (2.51) en (2.42) se obtiene una ecuación que involucra sólamente los coeficientes de Legendre ϕ_e^{ijk}.

Ahora, introduciendo una aproximación "serendib" [60] se tiene que:

$$F_{ex}^{k,k2,k3} = \sum_{l=0}^{K-1-k2-k3} \left(A_{ex}^{k,l;K-k2-k3} \phi_{e1}^{l,k2,k3} - B_{ex}^{k,l;K-k2-k3} \phi_e^{l,k2,k3} + C_{ex}^{k,l;K-k2-k3} \phi_{e2}^{l,k2,k3} \right) \quad (2.52)$$

$$F_{ey}^{k1,k,k3} = \sum_{l=0}^{K-1-k1-k3} \left(A_{ey}^{k,l;K-k1-k3} \phi_{e3}^{k1,l,k3} - B_{ey}^{k,l;K-k1-k3} \phi_e^{k1,l,k3} + C_{ey}^{k,l;K-k1-k3} \phi_{e4}^{k1,l,k3} \right) \quad (2.53)$$

$$F_{ez}^{k1,k2,k} = \sum_{l=0}^{K-1-k1-k2} \left(A_{ez}^{k,l;K-k1-k2} \phi_{e5}^{k1,k2,l} - B_{ez}^{k,l;K-k1-k2} \phi_e^{k1,k2,l} + C_{ez}^{k,l;K-k1-k2} \phi_{e6}^{k1,k2,l} \right) \quad (2.54)$$

Habiendo visto a qué son iguales los términos $F_{ex}^{k1,k2,k3}$, $F_{ey}^{k1,k2,k3}$ y $F_{ez}^{k1,k2,k3}$ de la ecuación (2.42), ésta puede generalizarse para dos grupos de energía, y ordenando convenientemente los índices, el sistema de ecuaciones diferenciales (2.29) se puede aproximar mediante un problema algebraico generalizado de valores propios, que se expresa como:

$$\begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \psi_{1_n} \\ \psi_{2_n} \end{bmatrix} = \frac{1}{k_n} \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_{1_n} \\ \psi_{2_n} \end{bmatrix}, \quad (2.55)$$

es decir:

$$L\psi_n = \frac{1}{k_n} M\psi_n.$$

El problema de valores propios generalizado representado en (2.55) es un problema fundamental que es necesario resolver para llevar a cabo estudios de estabilidad y seguridad en los reactores nucleares [118], pues permite el estudio del problema dinámico que se expone en el siguiente apartado.

2.4. Estudio de Transitorios. Problema Dinámico

Partiendo de las ecuaciones (2.17) y (2.18), la discretización espacial mediante el método de colocación nodal visto en la sección anterior da como resultado el siguiente sistema de ecuaciones diferenciales ordinarias:

$$\left[v^{-1}\right] \dot{\psi} + L\psi = (1-\beta) M\psi + X \sum_{k=1}^{K} \lambda_k C_k \quad (2.56)$$

$$\dot{C}_k = \beta_k \begin{bmatrix} M_{11} & M_{12} \end{bmatrix} \psi - \lambda_k C_k, \quad (2.57)$$

donde

$$L = \begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix},$$

$$M = \begin{bmatrix} M_{11} & M_{22} \\ 0 & 0 \end{bmatrix} \text{ y}$$

$$X = \begin{bmatrix} I \\ 0 \end{bmatrix}.$$

Se empezará con la discretización de (2.57). Por una parte, se sabe que con la serie de Taylor de primer grado se puede aproximar una función diferenciable $f(t)$ alrededor del punto t_n como sigue [15]:

$$f(t) \approx f(t_n) + \frac{f'(t_n)}{1!}(t - t_n).$$

Por otra parte, para integrar (2.57) desde el instante t_n al instante t_{n+1}, se supone que $\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}\psi$ varía linealmente entre estos instantes, y por su aproximación por la serie de Taylor de primer grado, se aproxima como:

$$\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}\psi \approx \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^n \psi^n + \left\{ \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}\psi \right\}' (t - t_n), \qquad (2.58)$$

y aplicando diferencias finitas hacia atrás para aproximar la derivada:

$$\left\{ \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}\psi \right\}' \approx \frac{\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^{n+1} \psi^{n+1} - \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^n \psi^n}{h_t},$$

donde $h_t = t_{n+1} - t_n$, entonces la ecuación (2.58) queda como:

$$\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}\psi \approx \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^n \psi^n + \\ + \frac{\begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^{n+1} \psi^{n+1} - \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^n \psi^n}{h_t}(t - t_n). \qquad (2.59)$$

Así, sustituyendo (2.59) en (2.57) se tiene que:

$$\dot{C}_k = \beta_k \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^n \psi^n + \\ + \left\{ \frac{\beta_k}{h_t} \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^{n+1} \psi^{n+1} - \frac{\beta_k}{h_t} \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^n \psi^n \right\} (t - t_n) - \lambda_k C_k,$$

es decir,

$$\dot{C}_k = \beta_k \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^n \psi^n + \\ + \frac{\beta_k}{h_t}(t - t_n) \left\{ \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^{n+1} \psi^{n+1} - \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}^n \psi^n \right\} - \lambda_k C_k,$$

que integrando se tiene la solución C_k en t_{n+1} que se puede expresar como:

$$C_k^{n+1} = C_k^n e^{-\lambda_k h_t} + \\ + \beta_k \left(a_k \left[\begin{array}{cc} M_{11} & M_{12} \end{array} \right]^n \psi^n + b_k \left[\begin{array}{cc} M_{11} & M_{12} \end{array} \right]^{n+1} \psi^{n+1} \right), \quad (2.60)$$

donde M_{11} y M_{22} son matrices diagonales y

$$\begin{aligned} a_k &= \frac{(1+\lambda_k h_t)\left(1-e^{-\lambda_k h_t}\right)}{\lambda_k^2 h_t} - \frac{1}{\lambda_k}, \\ b_k &= \frac{\lambda_k h_t - 1 + e^{-\lambda_k h_t}}{\lambda_k^2 h_t}. \end{aligned}$$

Ahora bien, para discretizar la ecuación (2.56) se aplican diferencias finitas hacia atrás de un paso al término de la derivada con respecto al tiempo $\left[v^{-1}\right]\dot{\psi}$, quedando:

$$\left[v^{-1}\right] \frac{\psi^{n+1} - \psi^n}{h_t} + L^{n+1}\psi^{n+1} = (1-\beta)M^{n+1}\psi^{n+1} + X\sum_{k=1}^{K} \lambda_k C_k^{n+1}. \quad (2.61)$$

Sustituyendo (2.60) en (2.61):

$$\left[v^{-1}\right] \frac{\psi^{n+1} - \psi^n}{h_t} + L^{n+1}\psi^{n+1} = (1-\beta)M^{n+1}\psi^{n+1} + \\ + X\sum_{k=1}^{K} \lambda_k \left(C_k^n e^{-\lambda_k h_t} + \beta_k \left(a_k \left[\begin{array}{cc} M_{11} & M_{12} \end{array} \right]^n \psi^n + b_k \left[\begin{array}{cc} M_{11} & M_{12} \end{array} \right]^{n+1} \psi^{n+1} \right) \right),$$

que desarrollando queda como:

$$\left\{ \frac{1}{h_t}\left[v^{-1}\right] + L^{n+1} - (1-\beta)M^{n+1} - X\sum_{k=1}^{K} \lambda_k \beta_k b_k \left[\begin{array}{cc} M_{11} & M_{12} \end{array} \right]^{n+1} \right\} \psi^{n+1} = \\ \left\{ \frac{1}{h_t}\left[v^{-1}\right] + X\sum_{k=1}^{K} \lambda_k \beta_k a_k \left[\begin{array}{cc} M_{11} & M_{12} \end{array} \right]^n \right\} \psi^n + X\sum_{k=1}^{K} \lambda_k C_k^n e^{-\lambda_k h t},$$

es decir:

$$\left\{ \frac{1}{h_t} \left[\begin{array}{cc} v_1^{-1} & 0 \\ 0 & v_2^{-1} \end{array} \right] + \left[\begin{array}{cc} L_{11}^{n+1} & 0 \\ L_{21}^{n+1} & L_{22}^{n+1} \end{array} \right] - (1-\beta) \left[\begin{array}{cc} M_{11}^{n+1} & M_{12}^{n+1} \\ 0 & 0 \end{array} \right] - \\ \left[\begin{array}{c} I \\ 0 \end{array} \right] \sum_{k=1}^{K} \lambda_k \beta_k b_k \left[\begin{array}{cc} M_{11}^{n+1} & M_{12}^{n+1} \end{array} \right] \right\} \psi^{n+1} = \\ \left\{ \frac{1}{h_t} \left[\begin{array}{cc} v_1^{-1} & 0 \\ 0 & v_2^{-1} \end{array} \right] + \left[\begin{array}{c} I \\ 0 \end{array} \right] \sum_{k=1}^{K} \lambda_k \beta_k a_k \left[\begin{array}{cc} M_{11}^{n} & M_{12}^{n} \end{array} \right] \right\} \psi^n + \\ \left[\begin{array}{c} I \\ 0 \end{array} \right] \sum_{k=1}^{K} \lambda_k C_k^n e^{-\lambda_k h t},$$

es decir:

$$\left\{ \begin{bmatrix} \frac{1}{h_t}v_1^{-1} & 0 \\ 0 & \frac{1}{h_t}v_2^{-1} \end{bmatrix} + \begin{bmatrix} L_{11}^{n+1} & 0 \\ L_{21}^{n+1} & L_{22}^{n+1} \end{bmatrix} - \begin{bmatrix} (1-\beta)M_{11}^{n+1} & (1-\beta)M_{12}^{n+1} \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \sum_{k=1}^{K}\lambda_k\beta_k b_k M_{11}^{n+1} & \sum_{k=1}^{K}\lambda_k\beta_k b_k M_{12}^{n+1} \\ 0 & 0 \end{bmatrix} \right\} \psi^{n+1} =$$

$$\left\{ \begin{bmatrix} \frac{1}{h_t}v_1^{-1} & 0 \\ 0 & \frac{1}{h_t}v_2^{-1} \end{bmatrix} + \begin{bmatrix} \sum_{k=1}^{K}\lambda_k\beta_k a_k M_{11}^{n} & \sum_{k=1}^{K}\lambda_k\beta_k a_k M_{12}^{n} \\ 0 & 0 \end{bmatrix} \right\} \psi^{n} + \sum_{k=1}^{K}\lambda_k e^{-\lambda_k ht} \begin{bmatrix} C_k^n \\ 0 \end{bmatrix}. \quad (2.62)$$

De (2.62) se obtiene el siguiente sistema de ecuaciones lineales:

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} \psi_1^{n+1} \\ \psi_2^{n+1} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \begin{bmatrix} \psi_1^{n} \\ \psi_2^{n} \end{bmatrix} + \sum_{k=1}^{K}\lambda_k e^{-\lambda_k ht} \begin{bmatrix} C_k^n \\ 0 \end{bmatrix}, \quad (2.63)$$

donde

$$\begin{aligned} T_{11} &= \frac{1}{h_t}v_1^{-1} + L_{11}^{n+1} - (1-\beta)M_{11}^{n+1} - \sum_{k=1}^{K}\lambda_k\beta_k b_k M_{11}^{n+1}, \\ T_{12} &= -(1-\beta)M_{12}^{n+1} - \sum_{k=1}^{K}\lambda_k\beta_k b_k M_{12}^{n+1}, \\ T_{21} &= L_{21}^{n+1}, \\ T_{22} &= \frac{1}{h_t}v_2^{-1} + L_{22}^{n+1}, \\ R_{11} &= \frac{1}{h_t}v_1^{-1} + \sum_{k=1}^{K}\lambda_k\beta_k a_k M_{11}^{n}, \\ R_{12} &= \sum_{k=1}^{K}\lambda_k\beta_k a_k M_{12}^{n}, \\ R_{22} &= \frac{1}{h_t}v_2^{-1}. \end{aligned}$$

Como puede observarse, el sistema de ecuaciones representado por la expresión (2.63) es un sistema disperso, de gran tamaño y no simétrico, que es necesario resolver para cada paso de tiempo en la simulación de un transitorio.

2.5. Conclusiones

El estudio del comportamiento de los reactores nucleares, así como la descripción de los distintos procesos que ocurren en su interior pueden describirse mediante la EDN. En

los estudios de la EDN se pueden distinguir dos problemas: el estacionario y el dinámico.

El problema estacionario, también conocido como problema de los modos Lambda, adquiere la forma de un problema generalizado de valores propios que puede resolverse con distintas técnicas. No obstante, muchas de estas técnicas, presentan como operación principal el producto matriz–vector. Por otro lado, los métodos de discretización nodal de los casos de estudio abordados por este trabajo, dan origen a que no se disponga de la matriz L (en la expresión (2.55)) en forma explícita, por lo que es necesario utilizar estrategias que aprovechen la estructura a bloques del problema planteado por la expresión (2.55). El éxito de estas estrategias, depende en gran medida del nivel de eficiencia para llevar a cabo este producto matriz–vector, y como se verá más adelante, éste producto depende a su vez de la eficiencia con que se resuelvan los sistemas de ecuaciones lineales dispersos que forman la matriz L, y que a su vez forman una parte principal de los estudios y experimentos numéricos que aborda el presente trabajo. Otro de los aspectos fundamentales involucrados en el estudio de la EDN, lo constituye el problema dinámico que plantea y que es de vital importancia en los estudios de estabilidad de los reactores nucleares. En este problema es necesario resolver el sistema de ecuaciones lineales asociado al bloque T (ver expresión en (2.63)), para cada paso de tiempo del proceso de simulación en un transitorio. Esta matriz T, tiene la particularidad de ser no simétrica, dispersa de gran dimensión y con estructura a bloques, lo cual puede dificultar su resolución mediante métodos iterativos y precondicionadores estándar [10] [20]. Por tal motivo, en el presente trabajo se aborda el estudio, análisis, prueba y diseño de métodos (concretamente multipaso) que aprovechen las propiedades estructurales de los sub–bloques de la matriz T y al mismo tiempo, permitan su eficiente resolución.

Capítulo 3

Métodos de Resolución de Sistemas de Ecuaciones Lineales Dispersos

3.1. Introducción

Los sistemas de ecuaciones lineales dispersos (SELD) surgen en muchas áreas de aplicación, tales como: problemas de difusión acústica, control de tráfico aéreo, dinámica de fluidos, astrofísica, óptica láser, bioquímica, física de circuitos, oceanografía, simulación por computadora, estudios de demografía, economía, así como en todo tipo de problemas de ingeniería nuclear, petroquímica, eléctrica, estructural, entre otras. Ejemplos de matrices dispersas que surgen de estas aplicaciones pueden encontrarse en la colección de matrices dispersas Matrix Market[1] (también conocida como colección Harwell–Boeing) propuesta en sus inicios por I. S. Duff, R. G. Grimes y J. G. Lewis [71] [25] [26].

En esta capítulo se revisan los métodos que resuelven los SELD relacionados con la discretización de la ecuación de difusión neutrónica (EDN), tanto en su estado estacionario como dinámico, y que se han presentado en el capítulo 2. Estos métodos están implementados en algunas librerías secuenciales y paralelas, que permiten realizar operaciones del álgebra lineal numérica dispersa.

Los métodos de resolución que se abordan en las siguientes secciones son:

- **Métodos Directos,** que encuentran una solución al sistema $Ax = b$ en un número finito de pasos, donde cada paso en las operaciones intermedias depende de los pasos previos (sección 3.2).

[1] http://math.nist.gov/MatrixMarket/

- **Métodos Iterativos,** que generan una sucesión de aproximaciones a la solución de $Ax = b$. Aquí pueden encontrarse

    - **Métodos Iterativos Estacionarios,** como Jacobi, Gauss–Seidel, Sobrerrelajación Sucesiva y Métodos Iterativos Multipaso (sección 3.3).
    - **Métodos iterativos basados en Subespacios de Krylov,** o Métodos No Estacionarios, como Residuo Mínimo Generalizado, Gradiente Conjugado, Gradiente Biconjugado, Gradiente Biconjugado Estabilizado, Residuo Cuasi–mínimo Libre Transpuesto. Como algunos de los métodos iterativos requieren de transformar la matriz del sistema a resolver en otra equivalente con mejores propiedades estructurales, se revisan también los siguientes: **Método de Arnoldi** y **Método de Lanczos** (sección 3.4) .

También se revisan algunos precondicionadores (sección 3.5), como:

- **Precondicionadores Basados en Métodos Iterativos,** como Jacobi, Gauss–Seidel y Sobrerelajación Susesiva Simétrica.
- **Precondicionadores basados en Factorizaciones Incompletas,** como LU Incompleta en sus variaciones: sin relleno, con K niveles de relleno y con umbral.

Por último, en la sección 3.6 se enuncian conclusiones referentes a este capítulo.

3.2. Métodos Directos

La mayoría de los métodos directos para el caso disperso están basados en la eliminación de Gauss. Esto quiere decir, que la mayoría de los algoritmos calculan una factorización LU de una permutación de los coeficientes de la matriz general A

$$PAQ = LU,$$

donde P y Q son matrices de permutación, L y U son matrices triangulares inferior y superior, respectivamente. L y U son los factores de A y se utilizan para resolver el sistema $Ax = b$ a través de: resolver $Ly = P^T b$ con una sustitución hacia adelante y enseguida resolver $U(Q^T x) = y$ con una sustitución hacia atrás. Para el caso en que A es simétrica y definida positiva la factorización toma la forma $PAP^T = LL^T$, también conocida como factorización de Cholesky.

De acuerdo a lo ya dicho, la solución del sistema de ecuaciones lineales disperso $Ax = b$ mediante un método directo puede dividirse en varias fases:

- **Fase de Preordenamiento**. Con esta fase se persiguen varios objetivos. Uno es explotar la estructura de la matriz A, por ejemplo haciendo un preordenamiento a bloques triangular o en forma diagonal a bloques para resolver el sistema eficientemente. Otro objetivo es aplicar una permutación a la matriz A para que los elementos de la diagonal (o los pivotes) de la matriz permutada tengan valores grandes en comparación con el resto de los elementos, consiguiendo resolver el sistema de forma estable. Algunos métodos para el preordenamiento son: Grado Mínimo y Disección Anidada [91] [24] [43] [80].

- **Fase de Factorización Simbólica**. En esta fase [44] [54] se analiza la matriz preordenada para pronosticar la estructura dispersa de los factores L y U de la matriz A, produciendo estructuras adecuadas para su representación.

- **Fase de Factorización Numérica.** Aquí es cuando se realizan las operaciones aritméticas sobre la matriz A para calcular los factores L y U, para ello se utiliza [50] [24] [22]: la eliminación Gaussiana cuando A es general, y la factorización de Cholesky cuando A es simétrica y definida positiva.

- **Fase de Solución.** En esta fase se obtiene la solución del sistema $Ax = b$, es decir, del sistema $LUx = b$, resolviendo $Ly = b$ con sustitución hacia adelante, y resolviendo $Ux = y$ con sustitución hacia atrás.

En la mayoría de los códigos que implementan un método directo, las fases anteriores se presentan muchas veces en forma combinada. Por ejemplo, las fases 2 y 3 suelen combinarse de modo que los valores numéricos están disponibles cuando el ordenamiento está siendo generado. La fase 3 por lo general, es la parte que consume más tiempo de computación, a diferencia de la fase de solución la cual es más rápida.

Para el caso donde A es densa y de orden n se requiere un almacenamiento de $O(n^2)$ y de $O(n^3)$ operaciones aritméticas en punto flotante. Para el caso de los algoritmos dispersos, su objetivo es resolver ecuaciones de la forma $Ax = b$ en un espacio y tiempo proporcional a $O(n)+O(NZ)$ para una matriz de orden n con NZ elementos diferentes de cero. Es por esta razón que los códigos para el caso disperso suelen ser complicados [22].

3.3. Métodos Iterativos Estacionarios

Los métodos iterativos generan una sucesión de aproximaciones a la solución del sistema lineal $Ax = b$ de $n \times n$, con $A = \{a_{ij}\}_{i,j=1,2,\ldots,n}$, $x = \{x_i\}_{i=1,2,\ldots,n}$, $b = \{b_i\}_{i=1,2,\ldots,n}$, para ello parten de una aproximación inicial $x^{(0)}$ a la solución x^* y generan una sucesión de vectores $\{x^{(k)}\}_{k=0}^{\infty}$ que converge a x^*. La mayoría de los métodos iterativos involucran un proceso que convierte el sistema $Ax = b$ en otro equivalente $x = Tx + c$ para alguna

$T \in \Re^{n\times n}$ y algún $c \in \Re^n$. Entonces, tomando $x^{(0)}$ la sucesión de aproximaciones a la solución se calcula resolviendo

$$x^{(k)} = Tx^{(k-1)} + c, \tag{3.1}$$

para cada $k = 1, 2, ...$, donde T se llama matriz de iteración. Los métodos iterativos son muy utilizados para resolver sistemas grandes y dispersos, ya que son eficientes en el aspecto temporal y espacial cuando se automatizan.

Los métodos que se revisan en esta sección son: Jacobi (J), Gauss–Seidel (GS) y Sobrerrelajación Sucesiva (SOR).

3.3.1. Método Iterativo de Jacobi (J)

El método de Jacobi (J) [15] (ver algoritmo 1) consiste en resolver la i–ésima ecuación de $Ax = b$ para x_i para obtener, siempre que $a_{ii} \neq 0$, que

$$x_i = \sum_{j=1, j\neq i}^{n} \left(-\frac{a_{ij}x_j}{a_{ii}}\right) + \frac{b_i}{a_{ii}}, \quad i = 1, 2, ..., n,$$

y generar cada $x_i^{(k)}$ de las componentes de $x^{(k-1)}$ para $k \geq 1$ con

$$x_i^{(k)} = \frac{\sum_{j=1, j\neq i}^{n} \left(-a_{ij}x_j^{(k-1)}\right) + b_i}{a_{ii}}, \quad i = 1, 2, ..., n. \tag{3.2}$$

Este método puede escribirse en la forma $x^{(k)} = Tx^{(k-1)} + c$, dividiendo A en su diagonal y los elementos fuera de ésta. Para ello, sea $D = diag(A)$ matriz diagonal de los elementos diagonales de A, $-L$ la triangular inferior estricta de A y $-U$ la triangular superior estricta de A, de forma que $A = D - L - U$ (ver figura 3.1). Entonces

$$Ax = b \;\Rightarrow\; (D - L - U)\, x = b \;\Rightarrow\; Dx = (L + U)\, x + b \;\Rightarrow\; x = D^{-1}\,(L + U)\, x + D^{-1}b,$$

lo que da lugar a la forma matricial del método de Jacobi (J)

$$x^{(k)} = B_J x^{(k)} + c_J = D^{-1}\,(L + U)\, x^{(k-1)} + D^{-1}b, \quad k = 1, 2, ... \tag{3.3}$$

donde la matriz $B_J = D^{-1}(L + U)$ se le conoce como matriz de iteración de Jacobi.

3.3.2. Método Iterativo de Gauss–Seidel (GS)

En la ecuación (3.2) se observa que para calcular la aproximación $x^{(k)}$ se utilizan los componentes de $x^{(k-1)}$. El método de Gauss–Seidel (GS) [15] aprovecha el hecho de que para $i > 1$, $x_j^{(k)}$ $(j = 1, 2, ..., i - 1)$ ya se han calculado y se supone son una

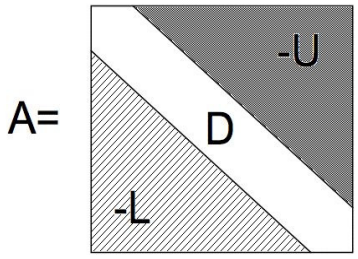

Figura 3.1: Partición de la matriz A en dos matrices triangulares L, U y una diagonal D.

Algoritmo 1 Método iterativo de Jacobi (J)

01 $X0 = x^{(0)}$
02 $k = 1$
03 Mientras $k \leq maxiter$
04 Para $i = 1, 2, ..., n$
05 $x_i = \frac{-\sum_{j=1, j\neq i}^{n}(a_{ij} XO_j) + b_i}{a_{ii}}$
06 Si $||x - XO||_p < TOL$ TERMINAR con solución $(x_1, ..., x_n)^T$
07 $k = k + 1$
08 $XO = x$
09 TERMINAR SIN ÉXITO, *maxiter* excedido

mejor aproximación a la solución, entonces se puede calcular $x_i^{(k)}$ con los valores más actualizados con los que se cuente, es decir

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1}\left(a_{ij}x_j^{(k)}\right) - \sum_{j=i+1}^{n}\left(a_{ij}x_j^{(k-1)}\right) + b_i}{a_{ii}}, \tag{3.4}$$

con $a_{ii} \neq 0$, para $i = 1, 2, ..., n$. Ahora, multiplicando (3.4) por a_{ii} se tiene que

$$a_{i1}x_1^{(k)} + a_{i2}x_2^{(k)} + ... + a_{ii}x_i^{(k)} = -a_{i,i+1}x_{i+1}^{(k-1)} - ... - a_{in}x_n^{(k-1)} + b_i,$$

para $i = 1, 2, ..., n$, y escribiendo las n ecuaciones se tiene que

$$\begin{aligned}
a_{11}x_1^{(k)} &= -a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - ... - a_{1n}x_n^{(k-1)} + b_1 \\
a_{21}x_1^{(k)} + a_{22}^{(k)} &= -a_{23}x_3^{(k-1)} - ... - a_{2n}x_n^{(k-1)} + b_2 \\
&\vdots \\
a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + ... + a_{nn}x_n^{(k)} &= b_n,
\end{aligned}$$

que en forma matricial puede escribirse como $(D-L)\,x^{(k)} = Ux^{(k-1)} + b$, donde D, L y U están definidos como en Jacobi, con lo que la iteración GS (ver algoritmo 2) es

$$x^{(k)} = B_{GS}x^{(k)} + c_{GS} = (D-L)^{-1}\,Ux^{(k-1)} + (D-L)^{-1}\,b. \tag{3.5}$$

donde la matriz $B_{GS} = (D-L)^{-1}U$ se le conoce como matriz de iteración de Gauss–Seidel.

Algoritmo 2 Método iterativo de Gauss–Seidel (GS)

01 $X0 = x^{(0)}$
02 $k = 1$
03 Mientras $k \leq maxiter$
04 Para $i = 1, 2, ..., n$
05 $x_i = \frac{-\sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}XO_j + b_i}{a_{ii}}$
06 Si $||x - XO||_p < TOL$ TERMINAR con solución $(x_1, ..., x_n)^T$
07 $k = k + 1$
08 $XO = x$
09 TERMINAR SIN ÉXITO, *maxiter* excedido

3.3.3. Convergencia de las Técnicas Generales Iterativas

En la literatura [91] [126] [15] se ha demostrado que para cualquier $x^{(0)} \in \Re^n$, la sucesión $\{x^{(0)}\}_{k=0}^{\infty}$ definida por

$$x^{(k)} = Tx^{(k-1)} + c, \quad \text{para cada } k \geq 1 \text{ y } c \neq 0,$$

converge a la solución única de $x = Tx + c$ si y sólo si el radio espectral de la matriz de iteración T es menor que 1, lo que se denota como $\rho(T) < 1$, y cuanto más próximo esté $\rho(T)$ a cero, mayor será la velocidad de convergencia.

También se ha demostrado [91] [126] [15] que si A es estrictamente diagonalmente dominante, entonces, para cualquier $x^{(0)}$ los métodos de Jacobi y Gauss–Seidel dan lugar a sucesiones $\{x^{(k)}\}_{k=0}^{\infty}$ que convergen a la solución de $Ax = b$.

Con respecto a los criterios de convergencia, se puede utilizar el que aparece en los algoritmos 1 y 2: $||x^{(k)} - x^{(k-1)}||_p < tol$, para alguna p–norma, pero también se puede utilizar uno que involucre al vector residual, que se define a continuación. Si $\tilde{x} \in \Re^n$ es una aproximación a la solución de $Ax = b$, el vector residual de $\tilde{x}$ con respecto a este sistema se define como

$$r = b - A\tilde{x}. \tag{3.6}$$

Entonces, los algoritmos de Jacobi y Gauss–Seidel pueden utilizar el siguiente criterio de convergencia

$$||b - Ax^{(k)}||_p < tol, \tag{3.7}$$

para alguna p–norma, el que se espera tienda a cero conforme el método iterativo calcule aproximaciones más cercanas a la solución del sistema.

3.3.4. Método Iterativo de Sobrerrelajación Sucesiva (SOR)

Se pueden derivar conexiones entre los vectores residuo de $x^{(k)}$ y el método iterativo GS. Por una parte, suponiendo que $r_i^{(k)} = \left(r_{1i}^{(k)}, r_{2i}^{(k)}, ..., r_{ni}^{(k)}\right)^T$ es el vector residual de GS de la aproximación $\left(x_1^{(k)}, x_2^{(k)}, ..., x_{i-1}^{(k)}, x_i^{(k-1)}, ..., x_n^{(k-1)}\right)^T$. La i–ésima componente de $r_i^{(k)}$ es

$$r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} - a_{ii}x_i^{(k-1)},$$

es decir

$$a_{ii}x_i^{(k-1)} + r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)},$$

y como la iteración de Gauss–Seidel tiene la forma (3.4), la ecuación anterior se escribe como $a_{ii}x_i^{(k-1)} + r_{ii}^{(k)} = a_{ii}x_i^{(k)}$, es decir

$$x_i^{(k)} = x_i^{(k-1)} + \frac{r_{ii}^{(k)}}{a_{ii}}. \tag{3.8}$$

Por otra parte, la i–ésima componente de $r_{i+1}^{(k)}$ es

$$r_{i,i+1}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} - a_{ii}x_i^{(k)},$$

que por la iteración de Gauss–Seidel (3.4) implica que $r_{i,i+1}^{(k)} = 0$, es decir, que Gauss–Seidel está ideado para requerir que la i–ésima componente de $r_{i+1}^{(k)}$ sea cero, lo que reduce la norma de este vector residual, aunque no necesariamente de forma eficiente. Por lo anterior, si se modifica Gauss–Seidel en la forma de (3.8) a

$$x_i^{(k)} = x_i^{(k-1)} + \omega\frac{r_{ii}^{(k)}}{a_{ii}}, \tag{3.9}$$

entonces, para cierto $\omega > 0$, la rapidez de convergencia será mayor.

Los métodos iterativos que emplean (3.9) se llaman métodos de relajación. Cuando $0 < \omega < 1$ los métodos se llaman de subrrelajación, y cuando $1 < \omega$ se llaman de sobrerrelajación sucesiva o SOR [15]. Los métodos SOR se utilizan para acelerar la convergencia de Gauss–Seidel.

Reescribiendo (3.9) en términos de los componentes de A, $x^{(k)}$, $x^{(k-1)}$ y b, la i–ésima componente de la k–ésima aproximación de SOR es

$$x_i^{(k)} = (1-\omega)\, x_i^{(k-1)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right), \tag{3.10}$$

que en forma matricial se escribe como

$$x^{(k)} = B_\omega x^{(k-1)} + c_\omega = (D - \omega L)^{-1} \left((1-\omega)\, D + \omega U \right) x^{(k-1)} + \omega \left(D - \omega L \right)^{-1} b, \tag{3.11}$$

donde D, L y U están definidos como en Jacobi y Gauss–Seidel.

La selección del valor apropiado para ω también se hace en función del espectro de la matriz de iteración T. Se sabe que [91] [126] [15] si $diag(A) \neq 0$, entonces $\rho(T_\omega) \geq |\omega - 1|$; esto implica que $\rho(T_\omega) < 1$ sólo si $0 < \omega < 2$. Entonces, si A es definida positiva y $0 < \omega < 2$, SOR converge para cualquier $x^{(0)}$. El algoritmo 3 da cuenta de la iteración de SOR, donde la prueba de convergencia de la línea 6 puede sustituirse por la expresión en (3.7).

Algoritmo 3 Método iterativo de sobrerrelajación sucesiva (SOR)

01 $X0 = x^{(0)}$
02 $k = 1$
03 Mientras $k \leq maxiter$
04 Para $i = 1, 2, ..., n$
05 $x_i = (1-\omega)\, XO_i + \frac{\omega\left(- \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} XO_j + b_i \right)}{a_{ii}}$
06 Si $||x - XO||_p < TOL$ TERMINAR con solución $(x_1, ..., x_n)^T$
07 $k = k + 1$
08 $XO = x$
09 TERMINAR SIN ÉXITO, *maxiter* excedido

3.3.5. Métodos Multipaso

Este apartado tiene por objetivo mostrar algunos fundamentos téoricos acerca de los métodos iterativos multipaso, también llamados de grado $\hat{s}$ [126]. También se revisan dos métodos multipaso de grado $\hat{s} = 2$ basados en las particiones aceleradas de Jacobi y Gauss–Seidel [81], los cuales han sido la base de los métodos propuestos en este trabajo de tesis (ver capítulo 5).

Los métodos multipaso han sido aplicados con éxito en distintos trabajos de investigación para la resolución de SELD asociados al problema dinámico de la EDN como en [14] [48] [81] [49] [117]. En estas mismas fuentes se puede encontrar información más detallada de lo que se expondrá a continuación.

3.3.5.1. Métodos Multipaso de Grado $\hat{s}$

Supóngase que se quiere resolver el sistema representado por

$$T\psi = e, \tag{3.12}$$

donde la matriz T es invertible y está dividida en $q \times q$ bloques de la forma

$$T = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1q} \\ T_{21} & T_{22} & \cdots & T_{2q} \\ \vdots & \vdots & & \vdots \\ T_{q1} & T_{q2} & \cdots & T_{qq} \end{bmatrix}, \tag{3.13}$$

y se cumple que los bloques diagonales T_{ii}, $i = 1, \ldots, q$, son matrices cuadradas e invertibles de dimensión n_i, $i = 1, \ldots, q$, por lo que la matriz global T es de tamaño $\sum_{i=1}^{q} n_i = n$.

Representando a la matriz T como la suma de matrices $T = D - L - U$, donde L y U son matrices con las partes triangular inferior y superior estrictas a bloques de la matriz $-T$, respectivamente, y D es la matriz diagonal de los bloques diagonales de la matriz T, es decir

$$T = \begin{bmatrix} T_{11} & 0 & \cdots & 0 \\ 0 & T_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & T_{qq} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -T_{21} & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \vdots \\ -T_{q1} & \ldots & -T_{qq-1} & 0 \end{bmatrix} - \begin{bmatrix} 0 & -T_{12} & \cdots & -T_{1q} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & 0 & -T_{q-1,q} \\ 0 & \ldots & 0 & 0 \end{bmatrix}, \tag{3.14}$$

como la matriz T es invertible, D también lo es, y pueden definirse las siguientes matrices de iteración para los métodos Jacobi(J) y Gauss–Seidel (GS) (vistos en los apartados 3.3.1, 3.3.2 y 3.3.4), así como la matriz de iteración de un método Gauss–Seidel acelerado (AGS) presentado en [14] [81]:

$$B_J = D^{-1}(L+U), \quad B_{GS} = (D-L)^{-1}U, \quad B_{AGS} = (D-\omega L)^{-1}((1-\omega)L+U). \tag{3.15}$$

Un método iterativo para resolver (3.12) se puede definir como un conjunto de funciones $\phi_0(T, e)$, $\phi_1(\psi^{(0)}; T, e)$, ..., $\phi_n(\psi^{(0)}, \ldots, \psi^{(n-1)}; T, e)$, en donde la secuencia de vectores iterados $\psi^{(0)}, \psi^{(1)}, \ldots, \psi^{(n)}$ se define como

$$\begin{aligned} \psi^{(0)} &= \phi_0(T, e) \\ \vdots \quad &\vdots \quad \vdots \\ \psi^{(n)} &= \phi_n(\psi^{(0)}, \psi^{(1)}, \ldots, \psi^{(n-1)}; T, e). \end{aligned}$$

Se dice que el método iterativo es estacionario de grado $\hat{s}$ si para algún $\hat{s} > 0$, ocurre que ϕ_n es independiente de n, es decir, si para todo $n \geq \hat{s}$, $\psi_{(n)}$ depende de

$\psi^{(n-1)}, \psi^{(n-2)}, \ldots, \psi^{(n-\hat{s})}$, pero no de $\psi^{(k)}$ para $k < n - \hat{s}$. Con lo anterior, se puede representar un método iterativo estacionario lineal multipaso (o multietapa) de grado $\hat{s}$ como

$$\psi^{(n)} = \sum_{i=1}^{\hat{s}} G_{\hat{s}-i}\psi^{(n-i)} + k, \tag{3.16}$$

donde $G_{\hat{s}-i}$ y k son funciones lineales de T y e.

En [126] se establecen las condiciones bajo las cuales un método multipaso de grado $\hat{s}$ converge a una solución aproximada de (3.12), por lo que, para estudiar el método representado en (3.16), es necesario establecer el siguiente sistema de ecuaciones lineales relacionado con (3.16)

$$\left(I - \sum_{i=1}^{\hat{s}} G_{\hat{s}-i}\right)\psi = (I - H)\,\psi = k, \tag{3.17}$$

en donde $H = \sum_{i=1}^{\hat{s}} G_{\hat{s}-i}$. Ahora, la relación que existe entre el conjunto de soluciones $\mathcal{S}(T, e)$ de la expresión (3.12) con el conjunto de soluciones $\mathcal{S}(I - H, k)$ de la expresión (3.17), está determinada por los siguientes teoremas.

Definición 1 *[81, Definición 24] Sea $\mathcal{S}(T, e)$ el conjunto de soluciones de* (3.12) *y sea $\mathcal{S}(I - H, k)$ el conjunto de soluciones de* (3.17)*. El esquema iterativo expresado por* (3.16) *se dice que es consistente con* (3.12) *si $\mathcal{S}(T, e) \subseteq \mathcal{S}(I - h, k)$, y completamente consistente si $\mathcal{S}(T, e) = \mathcal{S}(I - H, k)$.*

La definición 1 implica que si un método iterativo es completamente consistente, en el caso de que converja lo hace a una solución del sistema.

Teorema 1 *[81, Teorema 25] Si T en la expresión* (3.12) *es invertible, entonces el esquema iterativo representado en la expresión* (3.16) *es consistente con el sistema* (3.12) *si, y sólo si $k = (I - H)T^{-1}e$.*

Teorema 2 *[81, Teorema 26] Si T en* (3.12) *es invertible, entonces el esquema iterativo en la expresión* (3.16) *es completamente consistente con el sistema* (3.12) *si, y sólo si es consistente y además $I - H$ es invertible.*

Usando los teoremas 1 y 2, el método iterativo de grado $\hat{s}$ dado en la expresión (3.16) es completamente consistente con (3.12) si se cumple que

$$k = \left(I - \sum_{i=1}^{\hat{s}} G_{\hat{s}-i}\right)T^{-1}e,$$

y la matriz $I - \sum_{i=1}^{\hat{s}} G_{\hat{s}-i}$ es invertible, es decir, $1 \notin \rho(\sum_{i=1}^{\hat{s}} G_{\hat{s}-i})$.

En los métodos iterativos multipaso es importante garantizar la invertibilidad de la matriz $I - \sum_{i=1}^{\hat{s}} G_{\hat{s}-i}$, lo que se hace obteniendo el método de grado $\hat{s}$ a partir de un método de 1 paso convergente. Así, dada una partición de la matriz $T = M - N$ con M invertible, se obtiene la matriz $G = M^{-1}N$ asociada a la partición. Por lo que en caso de que $\rho(G) < 1$, es decir, se cumpla que la partición es convergente, el siguiente resultado proporciona una forma de elegir las matrices $G_{\hat{s}-i}$ para que el esquema iterativo de grado $\hat{s}$ en (3.16) sea completamente consistente.

Lema 1 *[81, Lema 27] Sea $T = M - N$ una partición de la matriz T del sistema* (3.12) *y sea la matriz $G = M^{-1}N$ tal que $\rho(G) < 1$. Sean las matrices $G_{\hat{s}-i}$ de la forma*

$$G_{\hat{s}-i} = GP_{\hat{s}-i}, i = 1, \ldots, \hat{s}, \tag{3.18}$$

con matrices $P_{\hat{s}-i}$ tales que $\sum_{i=1}^{\hat{s}} P_{\hat{s}-i} = I$. El esquema iterativo de grado $\hat{s}$ es completamente consistente con (3.12) *si, y sólo si $k = M^{-1}e$.*

La demostración del Lema 1 puede encontrarse en [81], de donde se desprende el siguiente corolario.

Corolario 1 *[Corolario 28 en 81] Sea $T = M - N$ una partición de la matriz T del sistema representado en* (3.12)*, y sea la matriz $G = M^{-1}N$ tal que $\rho(G) < 1$. Si $k = M^{-1}e$ y las matrices $P_{\hat{s}-i}$ representadas en la expresión* (3.18) *se toman de la forma $P_{\hat{s}-i} = \alpha_{\hat{s}-i}I$, en donde $\alpha_{\hat{s}-i} \in \Re, i = 1, \ldots, \hat{s}$, y $\sum_{i=1}^{\hat{s}} = 1$, entonces el esquema iterativo de grado $\hat{s}$ en* (3.16) *es completamente consistente con la expresión* (3.12)*.*

Con lo expuesto, un esquema iterativo multipaso de grado $\hat{s}$ puede obtenerse a partir de un esquema iterativo de grado $\hat{s} = 1$ convergente, como pueden ser los esquemas de las matrices de iteración definidas para los métodos de Jacobi o Gauss–Seidel.

Por ejemplo, un método iterativo estacionario multipaso de grado $\hat{s} = 2$ se define partiendo de la expresión en (3.16) como

$$\psi^{n+1} = G_1\psi^{(n)} + G_0\psi^{(n-1)} + k. \tag{3.19}$$

Ahora, considérese una partición de la matriz T de la forma $T = M - N$ y la matriz de iteración asociada $G = M^{-1}N$. Tomando las matrices G_0, G_1 y k de la forma [14] [81]

$$G_1 = \omega G,\ G_0 = (1 - \omega)G,\ k = M^{-1}e, \tag{3.20}$$

se obtiene el método iterativo de segundo grado dado por,

$$\psi^{n+1} = G(\omega\psi^{(n)} + (1 - \omega)\psi^{(n-1)}) + k. \tag{3.21}$$

Así, asumiendo que se cumple $\rho(G) < 1$, por el corolario 1, este método es completamente consistente (tomando como valores de $P_0 = (1 - \omega)I$ y $P_1 = \omega I$). Para el estudio de

los métodos multipaso de grado $\hat{s} = 2$ se utiliza el método auxiliar de grado $\hat{s} = 1$ siguiente [81],

$$\begin{bmatrix} \psi^{(n)} \\ \psi^{(n+1)} \end{bmatrix} = \begin{bmatrix} 0 & I \\ G_0 & G_1 \end{bmatrix} \begin{bmatrix} \psi^{(n-1)} \\ \psi^{(n)} \end{bmatrix} + \begin{bmatrix} 0 \\ k \end{bmatrix}, \tag{3.22}$$

en donde el método (3.22) es convergente para todo $\psi^{(0)}$ y $\psi^{(1)}$ si $\rho(\hat{G}_\omega) < 1$, donde la matriz de iteración del esquema (3.22) está representado por la expresión

$$\hat{G}_\omega = \begin{bmatrix} 0 & I \\ G_0 & G_I \end{bmatrix}.$$

En [81] se puede encontrar en forma detallada la demostración de que el método iterativo de primer grado (3.22) es convergente para todo $\psi^{(0)}$ y $\psi^{(1)}$, y mediante la relación que establece entre los valores propios de $\hat{G}_\omega$ y los valores propios de G, tomando en cuenta la definición de G_0 y G_1 dada por (3.20), demuestra que el método iterativo de segundo grado (3.21) también converge para valores determinados de ω.

En los siguientes apartados se muestran los métodos iterativos multipaso de grado $\hat{s} = 2$ basados en las particiones aceleradas de Jacobi y Gauss–Seidel [14] [81] que servirán de base de los métodos propuestos en esta tesis.

3.3.5.2. Método Multipaso de Jacobi (MMJ)

Para definir el esquema iterativo multipaso de grado $\hat{s} = 2$ basado en la partición de Jacobi, hay que considerar el sistema de ecuaciones (3.12) dividido en bloques como sigue

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}. \tag{3.23}$$

Dada la partición de la matriz $T = M - N$, en donde $M = D$ y $N = L + U$ correspondiente al método iterativo de Jacobi por bloques, es decir,

$$M = \begin{bmatrix} T_{11} & 0 \\ 0 & T_{22} \end{bmatrix}, \; N = \begin{bmatrix} 0 & -T_{12} \\ -T_{21} & 0 \end{bmatrix}.$$

Entonces, la matriz de iteración del método de Jacobi por bloques viene dada por,

$$B_J = \begin{bmatrix} 0 & -T_{11}^{-1}T_{12} \\ -T_{22}^{-1}T_{21} & 0 \end{bmatrix} = \begin{bmatrix} 0 & B_{12} \\ B_{21} & 0 \end{bmatrix}, \tag{3.24}$$

por lo que para simplificar se han introducido las matrices $B_{12} = -T_{11}^{-1}T_{12}$ y $B_{21} = -T_{22}^{-1}T_{21}$.

Si se considera el esquema iterativo multipaso de grado $\hat{s} = 2$ expuesto en la ecuación (3.19), con matrices $G_0 = \omega B_J$ y $(1 - \omega)B_J$, en donde ω es un factor de relajación y el

vector k está representado por $k = M^{-1}e = \begin{bmatrix} T_{11}^{-1}e_1 & T_{22}^{-1}e_2 \end{bmatrix}^T$, se obtiene el siguiente sistema de ecuaciones

$$\begin{aligned} T_{11}\psi_1^{l+1} &= e_1 - T_{12}(\omega\psi_2^l + (1-\omega)\psi_2^{l-1}, \\ T_{22}\psi_2^{l+1} &= e_2 - T_{21}(\omega\psi_1^l + (1-\omega)\psi_1^{l-1}, \end{aligned}$$

y que corresponde al cuerpo principal del método multipaso de Jacobi (llamado MMJ en este trabajo de tesis). En este caso, todos los resultados obtenidos para un esquema iterativo de segundo grado genérico (consistencia y convergencia) se aplican a este esquema particular. En el capítulo 5 se presenta un algoritmo basado en este método, así como las distintas pruebas experimentales realizadas en un entorno paralelo y que se aplicaron a un problema dinámico de la EDN multigrupo 3D.

3.3.5.3. Método Multipaso de Gauss–Seidel (MMGS)

En el esquema iterativo multipaso basado en Gauss–Seidel acelerado (MMGS) se pueden identificar las matrices G_0, G_1 y k que, sustituidas en la ecuación (3.19), permitan escribirlo como un esquema iterativo multietapa de grado $\hat{s} = 2$. Estas matrices son [14] [81]:

$$G_0 = \begin{bmatrix} 0 & -(1-\omega)B_{12} \\ 0 & (1-\omega)\omega B_{21}B_{12} \end{bmatrix}, G_1 = \begin{bmatrix} 0 & \omega B_{12} \\ -(1-\omega)B_{21} & \omega^2 B_{21}B_{12} \end{bmatrix}, \text{ y} \tag{3.25}$$

$$k = \begin{bmatrix} T_{11}^{-1}e_1 \\ T_{22}^{-1}e_2 - \omega B_{21}T_{11}^{-1}E_1 \end{bmatrix}, \tag{3.26}$$

en donde $B_{12} = T_{11}^{-1}T_{12}$, $B_{21} = T_{22}^{-1}T_{21}$ son los bloques que están fuera de la diagonal principal de la matriz de Jacobi por bloques en la ecuación (3.24). Al sustituir en (3.19) se obtiene el sistema de ecuaciones

$$\begin{aligned} T_{11}\psi_1^{l+1} &= e_1 - T_{12}(\omega\psi_2^l + (1-\omega)\psi_2^{l-1}), \\ T_{22}\psi_2^{l+1} &= e_2 - T_{21}(\omega\psi_1^{l+1} + (1-\omega)\psi_1^{l-1}), \end{aligned}$$

que se corresponden con el método multipaso Gauss–Seidel acelerado, abreviado como MMGS en este trabajo de tesis. Además, en los trabajos [14] y [81], las matrices G_0 y G_1 son factorizadas de la siguiente manera,

$$G_0 = \begin{bmatrix} 0 & -B_{12} \\ -(1-\omega)B_{21} & \omega B_{21}B_{12} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & (1-\omega)I \end{bmatrix} = H_\omega P_0 \tag{3.27}$$

$$G_1 = \begin{bmatrix} 0 & -B_{12} \\ -(1-\omega)B_{21} & \omega B_{21}B_{12} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \omega I \end{bmatrix} = H_\omega P_1 \tag{3.28}$$

por lo que la matriz del método ampliado equivalente (3.22) se puede escribir como

$$\hat{G}_\omega = \begin{bmatrix} 0 & I \\ H_\omega P_0 & H_\omega P_1 \end{bmatrix} \tag{3.29}$$

obteniéndose la relación

$$H_\omega P_0 + H_\omega P_1 = H_\omega (P_0 + P_1) = H_\omega.$$

Ahora, factorizando la matriz H_ω como

$$H_\omega = M^{-1}N = \begin{bmatrix} T_{11}^{-1} & 0 \\ -\omega B_{21} T_{11}^{-1} & T_{22}^{-1} \end{bmatrix} \begin{bmatrix} 0 & -T_{12} \\ -(1-\omega)T_{21} & 0 \end{bmatrix}, \tag{3.30}$$

se observa que el término k en (3.26) responde a la expresión

$$k = M^{-1}e = \begin{bmatrix} T_{11}^{-1} & 0 \\ -\omega B_{21} T_{11}^{-1} & T_{22}^{-1} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}.$$

Por otro lado, si además el radio espectral de la matriz H_ω es menor que 1, por el Lema 1, el método multipaso MMGS es completamente consistente con el sistema de ecuaciones en (3.23). Por otro lado, hay que considerar que la matriz H_ω se obtiene de una partición de la matriz T del sistema de ecuaciones mostrado en (3.23), $T = M_\omega - N_\omega$. En efecto, de la ecuación (3.30) se sigue que,

$$M_\omega = \begin{bmatrix} T_{11} & 0 \\ \omega T_{21} & T_{22} \end{bmatrix}, N_\omega = \begin{bmatrix} 0 & -T_{12} \\ -(1-\omega)T_{21} & 0 \end{bmatrix}. \tag{3.31}$$

Ahora, considerando las matrices,

$$D = \begin{bmatrix} T_{11} & 0 \\ 0 & T_{22} \end{bmatrix}, L = \begin{bmatrix} 0 & 0 \\ -T_{21} & 0 \end{bmatrix}, \text{ y } U = \begin{bmatrix} 0 & -T_{12} \\ 0 & 0 \end{bmatrix}, \tag{3.32}$$

las matrices T y H_ω pueden escribirse como

$$\begin{aligned} T &= (D - \omega L) - ((1-\omega)L + U), \\ H_\omega &= (D - \omega L)^{-1}((1-\omega)L + U). \end{aligned}$$

Hay que notar que la matriz de iteración H_ω corresponde al método de Gauss–Seidel acelerado representado en la ecuación (3.15), aplicado a la matriz T del sistema de ecuaciones representado en (3.23). En [81] se presentan resultados de convergencia de este método.

En este trabajo de tesis se han realizado experimentos numéricos con los métodos multipaso basados en particiones acelerados de Jacobi (MMJ) y Gauss-Seidel acelerado (MMGS). Los resultados de los experimentos se muestran en el capítulo 5, así como las distintas pruebas realizadas con las modificaciones a este último método propuestas por esta memoria y que son aporte de la misma.

3.4. Métodos Iterativos basados en Subespacios de Krylov

Algunos de los métodos más importantes que utilizan técnicas iterativas basadas en procesos de proyección para resolver sistemas de ecuaciones lineales de gran dimensión como el método del Residuo Mínimo Generalizado (GMRES) y Gradiente Conjugado (GC) se revisan en esta sección, ya que éstos y algunas variantes se implementan en las librerías que se han utilizado en este trabajo para resolver los sistemas de ecuaciones dispersos asociados a la ecuación de difusión neutrónica (EDN) . También se revisa el método de Arnoldi, que calcula una aproximación a los valores propios de una matriz y que puede utilizarse para resolver la ecuación de los modos Lambda (resultado de la discretización en estado estacionario de la EDN), cuya operación fundamental es la multiplicación matriz–vector.

Un proceso de proyección [91] representa una forma canónica de extracción de una aproximación a la solución de un sistema lineal $Ax = b$, con $A \in \Re^{n\times n}$ desde un subespacio m–dimensional $\mathcal{K}$ de $\Re^n$ ($\mathcal{K}$ es el subespacio de aproximaciones candidatas o el subespacio de búsqueda). En un proceso de proyección se restringe al residual $r = b - Ax$ para que sea ortogonal a m vectores linealmente independientes, lo que define otro subespacio m–dimensional $\mathcal{L}$ llamado subespacio de restricciones, de hecho, hay otros tipos de restricciones que se verán más adelante. Por cada paso que realiza el proceso de proyección, la dimensión del subespacio se incrementa en uno. Hay dos tipos de métodos de proyección: ortogonales ($\mathcal{L} = \mathcal{K}$) y oblícuos ($\mathcal{L} \neq \mathcal{K}$).

Para resolver $Ax = b$, en esta sección se abordan métodos de resolución basados en subespacios de Krylov, los cuales tienen la forma

$$\mathcal{K}_m(A, r_0) \equiv span\{r_0, Ar_0, A^2r_0, ..., A^{m-1}r_0\} \equiv \mathcal{K}_m. \tag{3.33}$$

Así, para resolver $Ax = b$ se obtiene una solución aproximada x_m desde el subespacio afín m–dimensional $x_0 + \mathcal{K}_m$ (donde x_0 es una aproximación inicial arbitraria de la solución), imponiendo alguna de las siguientes restricciones [22]:

- La aproximación Ritz–Galerkin, que establece que el residual de x_m sea ortogonal al subespacio $\mathcal{K}_m$: $b - Ax_m \perp \mathcal{K}_m$.
- La aproximación del residual mínimo, donde la norma del residual de x_m, $||b - Ax_m||$, debe ser mínimo sobre el subespacio $\mathcal{K}_m$.
- La aproximación Petrov–Galerkin, donde el residual de x_m debe ser ortogonal al subespacio $\mathcal{L}_m$: $b - Ax_m \perp \mathcal{L}_m$.
- La aproximación del error mínimo, que requiere que $||x - x_m||$ sea mínimo sobre $A\mathcal{K}_m$.

Los métodos de proyección también pueden utilizarse para hallar los valores propios de una matriz y generar subespacios de Krylov, tales como Arnoldi y Lanczos. Antes de presentar los métodos que resuelven $Ax = b$, se revisa el método de Arnoldi, cuya particularización es el método de Lanczos (que puede verse en [50] [91]).

3.4.1. Método de Arnoldi

El método de Arnoldi [7] [89] [50] se ideó para reducir una matriz densa no hermítica en una matriz de Hessenberg superior, sin embargo con su técnica se pueden calcular valores propios dominantes de matrices dispersas de gran dimensión.

Para tomar una base adecuada para el subespacio de Krylov (3.33) (ya que no es numéricamente conveniente tomar como base a A y r_0, pues los vectores $A_j r_0$ que se vayan agregando a (3.33) apuntarán más y más en la dirección del valor propio dominante, llegando a ser los vectores de la base linealmente dependientes en aritmética de precisión finita), se utiliza una base ortonormal. El método de Arnoldi realiza esto partiendo del hecho de que se tiene una base ortonormal $v_1, v_2, ..., v_j$ para $K_j(A, r_0)$, entonces la expande calculando Av_j y ortonormalizando este producto con respecto a $v_1, v_2, ..., v_j$. Este procedimiento se ve en el algoritmo 4, en el que se detecta que la operación más costosa es la multiplicación matriz–vector.

Algoritmo 4 Arnoldi básico.

01 Elegir un vector unitario inicial v_1
02 Elegir la dimensión del subespacio de krylov m
03 Para $j = 1, 2, \ldots, m$
04 $\quad h_{i,j} = v_i^T A v_j, \quad i = 1, 2, \ldots, j$
05 $\quad w_j = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$
06 $\quad h_{j+1,j} = \| w_j \|_2$, Si $h_{j+1,j} = 0$ TERMINAR.
07 $\quad v_{j+1} = w_j / h_{j+1,j}$

El algoritmo 4 tiene dos criterios de parada: o bien cuando se ha construido el subespacio de Krylov de orden m (término del ciclo de la línea 3 del algoritmo), o bien cuando uno de los vectores se anula (línea 6 del algoritmo). Para entender el segundo criterio de parada del algoritmo, se introducen las siguientes definiciones y proposiciones que pueden encontrarse en [91].

Definición 2 *El subespacio de Krylov $\mathcal{K}_m(A, v)$ es el subespacio de todos los vectores x en $\Re^n$, los cuales pueden ser expresados como $x = p(A)v$, donde p es un polinomio de grado que no excede a $m - 1$.*

Definición 3 *El polinomio mínimo de un vector v asociado a una matriz A es el polinomio mónico no nulo p de menor grado que verifica que $p(A)v = 0$.*

Proposición 1 *Sea μ el grado del polinomio mínimo de v asociado a A, entonces $\mathcal{K}_\mu(A, v)$ es un subespacio invariante sobre A y $\mathcal{K}_m(A, v) = \mathcal{K}_\mu(A, v)$ para todo $m \geq \mu$.*

Proposición 2 *El algoritmo de Arnoldi se detiene en el j–ésimo paso (línea 6 del algoritmo 4) si y sólo si el polinomio de v_1 asociado a la matriz A es de grado j. Más aún, en este caso el subespacio $\mathcal{K}_j$ es invariante bajo A.*

De acuerdo a las definiciones y proposiciones anteriores, la parada del algoritmo 4 por anulación de un vector (línea 6) se debe a que el grado del polinomio mínimo del vector v_1 es j, por lo que se ha conseguido el subespacio invariante $\mathcal{K}_j$ del que se pueden obtener j valores propios dominantes exactos de la matriz A.

A partir de la matriz Hessenberg $H = [h_{ij}]$, $i = 1, 2, ..., m$, $j = 1, 2, ..., m$, obtenida del algoritmo de Arnoldi se pueden calcular sus valores propios λ_i $(i = 1, 2, ..., m)$ mediante, por ejemplo, la iteración QR [50]. Si estos valores propios son buenas aproximaciones a los valores propios de A, entonces se pueden calcular los vectores propios asociados u_i $(i = 1, 2, ..., m)$ mediante $u_i = Vy_i$, donde $V = [v_1, v_2, ..., v_m]$ es la matriz formada por los vectores calculados en el algoritmo de Arnoldi, y_i es el i–ésimo vector propio de H asociado a λ_i $(i = 1, 2, ..., m)$. Sin embargo, si los valores propios calculados no son buenas aproximaciones a los valores propios dominantes de A, se puede aumentar el valor de m, es decir el número de iteraciones del algoritmo 4 en la línea 3, o se puede incluir un paso de reinicialización tomando un vector inicial v_1 con más información espectral. Este nuevo vector inicial se puede formar, por ejemplo, con una combinación lineal de la base $\{v_1, v_2, ..., v_m\}$ [87]; también es posible formarlo con u_1, el vector propio asociado con el valor propio dominante de A [121].

Otro caso a tomar en cuenta es cuando se trabaja con una matriz A con mal comportamiento. Para que el algoritmo de Arnoldi alcance una mejor estabilidad numérica se utiliza el proceso de Gram Schmidt modificado [91] en el cálculo de la matriz ortogonal V (ver algoritmo 5).

Algoritmo 5 Arnoldi con Gram Schmidt Modificado.

```
01 Elegir un vector unitario inicial v_1
02 Elegir la dimensión del subespacio de krylov m
03 Para j = 1, 2, ..., m
04     w = A v_j
05     Para i = 1, 2, ..., j
06         h_{i,j} = v_i^T w
07         w = w - h_{i,j} v_i
08     h_{j+1,j} = || w ||_2, Si h_{j+1,j} = 0 TERMINAR.
09     v_{j+1} = w / h_{j+1,j}
```

Con base en la Proposición 2 y en la Proposición 3 (que se da en el siguiente párrafo), está garantizado que el método de Arnoldi construye una base ortonormal del subespacio de Krylov $\mathcal{K}_m$ y también aproxima a los valores propios de una matriz dispersa de gran dimensión.

Proposición 3 *[91] Asumiendo que el algoritmo de Arnoldi no se detiene antes de m iteraciones (línea 3 del algoritmo 5), los vectores v_1, v_2, ..., v_m forman una base ortonormal del subespacio de Krylov $\mathcal{K}_m = span\{v_1, Av_1, A^2v_1, ..., A^{m-1}v_1\}$.*

Se presenta enseguida una proposición (que puede verse en [91]) que será útil para introducir el método del Residuo Mínimo Generalizado (GMRES) en secciones posteriores.

Proposición 4 *Sea V_m matriz $n \times m$ con columnas $v_1, v_2, ..., v_m$. Sea $\tilde{H}_m$ la matriz Hessenberg $(m+1) \times m$ con elementos h_{ij} diferentes de cero calculados por el algoritmo de Arnoldi. Sea H_m la matriz obtenida de $\tilde{H}_m$ al eliminar su último renglón (o fila). Entonces, se cumplen las siguientes relaciones:*

$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^T = V_{m+1}\tilde{H}_m \\ V_m^T A V_m &= H_m. \end{aligned} \tag{3.34}$$

3.4.2. Método del Residuo Mínimo Generalizado (GMRES)

El método del Residuo Mínimo Generalizado (GMRES) es un método de proyección utilizado para resolver sistemas no simétricos de la forma $Ax = b$, donde $A \in \Re^{n \times n}$ y $x, b \in \Re^n$, y que toma [91] $\mathcal{K} = \mathcal{K}_m$ y $\mathcal{L} = A\mathcal{K}_m$, siendo $\mathcal{K}_m(A, r_0/||r_0||_2)$ el subespacio de Krylov.

Como se ha visto, el método de Arnoldi reduce una matriz no simétrica a Hessenberg superior. GMRES aprovecha esta transformación para construir soluciones aproximadas para las cuales la norma del residual sea mínima con respecto a $x_0 + \mathcal{K}_m$ (restricción basada en la aproximación del residual mínimo [22]), como se ve a continuación.

Primero, definiendo $\beta = ||r_0||_2$, hay que recordar que

$$v_1 = \frac{r_0}{||r_0||_2} \;\Rightarrow\; r_0 = ||r_0||_2 v_1 \;\Rightarrow\; r_0 = \beta v_1. \tag{3.35}$$

Ahora, cualquier vector x en $x_0 + \mathcal{K}_m$ puede escribirse como la siguiente combinación lineal:

$$x = x_0 + y_1 v_1 + y_2 v_2 + ... + y_m v_m \;\Rightarrow\; x = x_0 + V_m y. \tag{3.36}$$

Definiendo

$$J(y) = ||b - Ax||_2 = ||b - A(x_0 + V_m y)||_2, \tag{3.37}$$

y utilizando (3.34) de la Proposición 4 y (3.35) en el residual $b - Ax$ se tiene que

$$b - Ax = b - A(x_0 + V_m y) = r_0 - AV_m y = \beta v_1 - V_{m+1}\tilde{H}_m y = V_{m+1}\left(\beta e_1 - \tilde{H}_m y\right).$$

Como las columnas de V_{m+1} son ortonormales, se tiene que

$$J(y) \equiv ||b - A\left(x_0 + V_m y\right)||_2 = ||\beta e_1 - \tilde{H}_m y||_2. \tag{3.38}$$

Entonces, como el método GMRES busca minimizar la norma residual, una vez resuelto el problema de mínimos cuadrados

$$\min_{y \in \Re^n} ||\beta e_1 - \tilde{H}_m y||_2, \tag{3.39}$$

con y_{min} como solución, es posible calcular una aproximación a la solución de $Ax = b$ con $x_m = x_0 + V_m y_{min}$.

Lo que queda por resolver es el problema de Mínimos Cuadrados (3.39), que equivale a resolver un sistema sobredeterminado, en este caso de la forma $\tilde{H}_m y = -\beta e_1$, donde $\tilde{H}_m \in \Re^{(m+1)\times m}$, $y \in \Re^m$ y $\beta e_1 \in \Re^m$.

Los problemas de mínimos cuadrados suelen resolverse reduciendo la matriz de coeficientes $\tilde{H}_m$ a alguna forma canónica, vía transformaciones ortogonales. Por ejemplo, aplicando rotaciones de Givens [50] [91], se puede obtener la factorización QR de $\tilde{H}_m$ [50] [91] $\tilde{H}_m = QR$, donde $Q \in \Re^{(m+1)\times(m+1)}$ es matriz ortogonal ($Q^{-1} = Q^T$) y $R \in \Re^{(m+1)\times m}$ es matriz triangular superior. De $\tilde{H}_m = QR$ se puede obtener $Q^T\tilde{H}_m = R$. Como la matriz Q es ortogonal y conserva la norma vectorial, se tiene que

$$||\tilde{H}_m y - \beta e_1||_2^2 = ||Q^T\left(\tilde{H}_m y - \beta e_1\right)||_2^2 = ||Ry - c||_2^2 = ||R_1 y - c_1||_2^2 + ||-c_2||_2^2,$$

donde

$$Q^T\tilde{H}_m = R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \quad Q^T b = c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix},$$

con $R_1 \in \Re^{m\times m}$, $c_1 \in \Re^m$, y $c_2 \in \Re$. Por lo tanto, el mínimo de (3.39) se obtiene resolviendo el sistema triangular $R_1 y = c_1$ [91], donde

$$\min_{y \in \Re^n} ||\tilde{H}_m y - \beta e_1||_2^2 = ||c_2||_2^2.$$

Todos los procesos referidos en párrafos anteriores del método GMRES se reúnen en el algoritmo 6. En este algoritmo, la salida del ciclo (que aplica Arnoldi para formar las matrices V y $\tilde{H}_m$) en la línea 12 indica, como ya se ha visto antes, que no se puede generar el vector v_{j+1}, pero también indica que al resolver el sistema $R_1 y = c_1$ y calcular $x_m = x_0 + Vy$, se obtiene la solución exacta del sistema $Ax = b$ [91].

Ya se sabe que el manejo de una m grande en la generación de V y $\tilde{H}_m$ con Arnoldi es poco práctico. Por ello, GMRES puede modificarse reiniciando la ortogonalización de Arnoldi, como en el algoritmo 7.

Algoritmo 6 Método del Residuo Mínimo Generalizado (GMRES).

01 $r_0 = b - Ax_0$
02 $\beta = ||r_0||_2$
03 $v_1 = r_0/\beta$
04 Definir $\tilde{H}$ de tamaño $(m+1) \times m$ y ponerla a 0
05 Para $j = 1, 2, \ldots, m$
06 $\quad w = Av_j$
07 $\quad$ Para $i = 1, 2, \ldots, j$
08 $\quad\quad h_{i,j} = v_i^T w$
09 $\quad\quad w = w - h_{i,j} v_i$
10 $\quad h_{j+1,j} = \| w \|_2$
11 $\quad$ Si $h_{j+1,j} = 0$
12 $\quad\quad m = j$, Ir a línea 14.
13 $\quad v_{j+1} = w/h_{j+1,j}$
14 Calcular la factorización $\tilde{H}_m = QR$
15 $c = Q^T \beta e_1$
16 Resolver el sistema triangular $R_1 y = c_1$
17 Calcular aproximación $x_m = x_0 + Vy$

Algoritmo 7 Método del Residuo Mínimo Generalizado (GMRES) con reinicio.

01 $r_0 = b - Ax_0$
02 $\beta = ||r_0||_2$
03 $v_1 = r_0/\beta$
04 $parar = 0$
05 Mientras $parar = 0$
06 $\quad$ Definir $\tilde{H}$ de tamaño $(m+1) \times m$ y ponerla a 0
07 $\quad$ Aplicar Arnoldi para calcular $\tilde{H}_m$ y V_m
08 $\quad$ Resolver el problema de mínimos cuadrados (3.39) con solución y_m
09 $\quad$ Calcular aproximación $x_m = x_0 + V_m y_m$
10 $\quad r_0 = b - Ax_m$
11 $\quad \beta = ||r_0||_2$
12 $\quad$ Si $\beta < tol$ $\;parar = 1$(TERMINAR con solución x_m)
13 $\quad$ Sino
14 $\quad\quad x_0 = x_m$
15 $\quad\quad v_1 = r_0/\beta$

3.4.3. Método Gradiente Conjugado (GC)

El método Gradiente Conjugado (GC) es un método utilizado para resolver sistemas lineales simétricos definidos positivos de la forma $Ax = b$, con $A \in \Re^{n \times n}$, $x, b \in \Re^n$. El método GC es un método de proyección que toma [91] $\mathcal{K} = \mathcal{L} = \mathcal{K}_m$, siendo $\mathcal{K}_m(A, r_0/||r_0||_2)$ subespacio de Krylov. El método GC cumple con la restricción

Ritz–Galerkin, que establece que el residual de x_m sea ortogonal al subespacio $\mathcal{K}_m$ [22].

Para resolver el sistema simétrico definido positivo $Ax = b$, el método GC reduce la matriz A a la forma Hessenberg aplicando el método de Lanczos [91] [50]. Procediendo análogamente al método de Arnoldi (ver expresiones (3.36) y (3.35)), una solución aproximada a $Ax = b$ está dada por

$$x_m = x_0 + V_m y_m, \tag{3.40}$$

para lo cual hay que calcular y_m; como el método GC trata de minimizar la norma del residual de $b - Ax$, al igual que Arnoldi (ver expresiones (3.37) y (3.38)), se tiene que

$$J_{GC}(y) = \| b - Ax \|_2 = \| \beta e_1 - \tilde{H}_m y \|_2,$$

pero como A es simétrica, la matriz Hessenberg asociada a A es tridiagonal, denotada con T_m, por lo que

$$J_{GC}(y) = \| \beta e_1 - T_m y \|_2 \ .$$

Entonces, como T_m es cuadrada, la minimización de $J_{GC}(y)$ consiste en resolver el sistema tridiagonal [91]

$$T_m y = \beta e_1. \tag{3.41}$$

Para resolver la ecuación (3.41) y calcular x_m se puede utilizar la factorización LU como sigue

$$x_m = x_0 + V_m y \ \Rightarrow \ x_m = x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1) \ \Rightarrow \ x_m = x_0 + P_m L_m^{-1}(\beta e_1),$$

donde

$$P_m = V_m U_m^{-1}, \tag{3.42}$$

y L_m y U_m tienen la forma

$$L_m = \begin{pmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & \lambda_{m-1} & 1 & \\ & & & \lambda_m & 1 \end{pmatrix}, \ U_m = \begin{pmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ & & \ddots & \ddots & \\ & & & \eta_{m-1} & \beta_m \\ & & & & \eta_m \end{pmatrix}.$$

Resolviendo el sistema triangular inferior $L_m z = \beta e_1$, la aproximación x_m queda como

$$x_m = x_0 + P_m z_m. \tag{3.43}$$

En [91] puede verse que la columna p_m de P_m puede calcularse utilizando la columnas previas y el vector v_m, es decir

$$p_m = \eta_m^{-1} \left[v_m - \beta_m p_{m-1} \right]. \tag{3.44}$$

Ahora hay que observar que el residual $r_m = b - Ax_m$ está en la dirección de v_{m+1}, pues [91] $b - Ax_m = -\beta_{m+1} e_m^T y_m v_{m+1}$, por lo que los vectores residuales son ortogonales entre sí. Lo mismo sucede con los vectores columna p_i de P_m (3.42), éstos son A–ortogonales [91]. Estos resultados se ven en la siguiente proposición [91].

Proposición 5 *Sean $r_m = b - Ax_m$, $m = 0, 1, ...$, vectores residuales obtenidos después de calcular un conjunto de aproximaciones x_m, $m = 0, 1, ...$ Sean p_m, $m = 0, 1, ...$, los vectores* (3.44)*. Entonces,*

1. *Cada vector residual es r_m es tal que $r_m = \sigma_m v_{m+1}$, donde σ_m es cierto escalar. Como resultado, los vectores residuales son ortogonales entre sí.*

2. *Los vectores p_i* (3.44) *forman un conjunto A–conjugado, es decir: $p_j^T(Ap_i) = 0$ para cada $i \neq j$.*

Entonces, imponiendo las condiciones de ortogonalidad y conjugación de la Proposición 5, para derivar el método GC se parte del hecho de que [91]

$$x_{j+1} = x_j + \alpha_j p_j. \tag{3.45}$$

Entonces, los vectores residuales satisfacen la recurrencia

$$r_{j+1} = r_j - \alpha_j A p_j. \tag{3.46}$$

Si las r_j son ortogonales, es necesario que $r_j^T(r_j - \alpha_j A p_j) = 0$, lo que implica que

$$\alpha_j = \frac{r_j^T r_j}{r_j^T (Ap_j)}. \tag{3.47}$$

Por la Proposición 5 se sabe que la dirección de búsqueda de p_{j+1} es una combinación lineal de r_{j+1} y p_j, y después de reescalar los vectores p apropiadamente, se tiene que

$$p_{j+1} = r_{j+1} + \beta_j p_j, \tag{3.48}$$

lo que tiene como consecuencia que $r_j^T(Ap_j) = (p_j - \beta_{j-1} p_{j-1})^T (Ap_j) = p_j^T (Ap_j)$, dado que Ap_j es ortogonal a p_{j-1}. Entonces, la ecuación (3.47) puede escribirse como

$$\alpha_j = \frac{r_j^T r_j}{p_j^T (Ap_j)}.$$

Además, tomando p_{j+1} de (3.48), como es ortogonal a Ap_j, entonces

$$\beta_j = -\frac{(Ap_j)^T r_{j+1}}{(Ap_j)^T p_j}.$$

De (3.46)

$$Ap_j = -\frac{1}{\alpha_j}(r_{j+1} - r_j) \tag{3.49}$$

y, de esta forma,

$$\beta_j = \frac{1}{\alpha_j}\frac{(r_{j+1} - r_j)^T r_{j+1}}{p_j^T(Ap_j)} = \frac{r_{j+1}^T r_{j+1}}{r_j^T r_j}. \tag{3.50}$$

El procedimiento anterior conforma al método del Gradiente Conjugado y se reúne en el algoritmo 8.

Algoritmo 8 Método Gradiente Conjugado (GC).

01 $r_0 = b - Ax_0$
02 $p_0 = r_0$
03 Para $j = 0, 1, ...$ Hasta convergencia
04 $\quad \alpha_j = r_j^T r_j / p_j^T(Ap_j)$
05 $\quad x_{j+1} = x_j + \alpha_j p_j$
06 $\quad r_{j+1} = r_j - \alpha_j Ap_j$
07 $\quad \beta_j = r_{j+1}^T r_{j+1} / r_j^T r_j$
08 $\quad p_{j+1} = r_{j+1} + \beta_j p_j$

3.4.4. Método Gradiente Biconjugado (BCG)

El método Gradiente Biconjugado (BCG) es un método utilizado para resolver sistemas lineales no simétricos de la forma $Ax = b$, con $A \in \Re^{n\ timesn}$, $b, x, \in \Re^n$. En general, la matriz de coeficientes de este tipo de sistemas no puede reducirse a una forma tridiagonal porque no es posible crear una base ortogonal para el subespacio de Krylov con relaciones de recurrencia de tres términos [22]. Sin embargo, se puede tratar de obtener una base no ortogonal adecuada con una recurrencia de tres términos, requiriendo que esta base sea ortogonal a alguna otra. Es por esto que el método BCG cumple con la restricción Petrov–Galerkin [22], es decir que busca una solución para $Ax = b$ en el subespacio de Krylov

$$\mathcal{K} = \mathcal{K}_m = span\{v_1, Av_1, ..., A^{m-1}v_1\},$$

con $v_1 = r_0/||r_0||2$, y restringe al residual de x_m para que sea ortogonal al subespacio

$$\mathcal{L}_m = span\{w_1, A^T w_1, ..., (A^T)^{m-1} w_1\},$$

con un vector arbitrario w_1 tal que $w_1^T v_1 \neq 0$. Implícitamente, BCG resuelve un sistema dual de la forma $A^T x^* = b^*$.

Para resolver el sistema $Ax = b$, el método BCG reduce la matriz A a una tridiagonal aplicando el método de biortogonalización de Lanczos [91], que es el encargado de construir el par de bases biortogonales para los subespacios $\mathcal{K}_m$ y $\mathcal{L}_m$.

La matriz tridiagonal que construye la biortogonalización de Lanczos tiene la forma

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix}, \tag{3.51}$$

donde sus componentes δ_j son positivos y sus componentes $\beta_j = \pm\delta_j$.

De forma análoga al método GC (ver ecuaciones (3.40) a (3.43)), después de reducir la matriz A a la matriz triangular T_m mediante Lanczos biortogonalizado, el método Gradiente Biconjugado (BCG) calcula la aproximación a la solución de $Ax = b$ mediante la expresión

$$x_m = x_0 + V_m y_m, \tag{3.52}$$

que requiere del cálculo de y_m, que es solución del sistema tridiagonal $T_m y = \beta e_1$; para esto, se calcula la descomposición LU de T_m, $T_m = L_m U_m$, con lo que la expresión (3.52) queda como $x_m = x_0 + P_m z_m$, con $P_m = V_m U_m^{-1}$ y $z_m = L_m^{-1}(\beta e_1)$. El cálculo de x_m se realiza tomando consideraciones análogas al método GC (ver ecuaciones (3.44) a (3.50)), como los hechos de que x_m es actualizable desde x_{m-1}, y cada vector residual r_j (r_j^* del sistema dual) está en la misma dirección que el vector v_{j+1} (w_{j+1}).

Similarmente, con respecto a la resolución del sistema dual se puede definir la matriz $P_m^* = W_m L_m^{-1}$, donde las columnas p_i^* de P_m^* y las columnas p_i de P_m son A–conjugadas, ya que [91] $(P_m^*)^T A P_m = L_m^{-1} W_m^T A V_m U_m^{-1} = L_m^{-1} T_m U_m^{-1} = I$.

Todas las operaciones concernientes al método BCG para resolver $Ax = b$ se reúnen en el algoritmo 9.

3.4.5. Método Gradiente Biconjugado Estabilizado (BCGSTAB)

Uno de los inconvenientes del método Gradiente Biconjugado (BCG) es que necesita para operar tanto de la matriz A como de su transpuesta A^T. El método Gradiente Conjugado Cuadrado (CGS), que puede encontrarse en [91] [107], propone expresar tanto los residuales r_j y r_j^* (calculados en las líneas 8 y 9 del algoritmo 9, respectivamente), como los vectores columna p_j y p_j^* (llamados vectores de direcciones de búsqueda, calculados en las líneas 11 y 12 del algoritmo 9, respectivamente) como recurrencias basadas en polinomios cuadráticos, como por ejemplo

$$r_j = \phi_j^2(A) r_0, \quad p_j = \pi_j^2(A) r_0,$$

Algoritmo 9 Método Gradiente Biconjugado (BCG)

01 $r_0 = b - Ax_0$
02 Elegir r_0^* tal que $(r_0^*)^T r_0 \neq 0$
03 $p_0 = r_0$;
04 $p_0^* = r_0^*$
05 Para $j = 0, 1, ...$ Hasta convergencia
06 $\quad \alpha_j = (r_j^*)^T r_j / (p_j^*)^T (Ap_j)$
07 $\quad x_{j+1} = x_j + \alpha_j p_j$
08 $\quad r_{j+1} = r_j - \alpha_j A p_j$
09 $\quad r_{j+1}^* = r_j^* - \alpha_j A^T p_j^*$
10 $\quad \beta_j = (r_{j+1}^*)^T r_{j+1} / (r_j^*)^T r_j$
11 $\quad p_{j+1} = r_{j+1} + \beta_j p_j$
12 $\quad p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*$

donde $\phi_j(A)$ y $\pi_j(A)$ son polinomios de grado j. Con esto, CGS evita el uso de A^T.

Aunque el método CGS trabaja bien en muchos casos, presenta la dificultad de los errores de redondeo provocados por el uso de polinomios cuadráticos, lo que puede afectar el desempeño del método.

El método Gradiente Biconjugado Estabilizado (BCGSTAB) evita el uso de la matriz transpuesta A^T y de los polinomios cuadráticos propuestos por el método CGS. El método BCGSTAB calcula una aproximación para un sistema lineal no simétrico $Ax = b$, con $A \in \Re^{n \times n}$ y $b, x \in \Re^n$, bajo las restricciones Petrov–Galerkin [22]. El método BCGSTAB propone expresar los residuales y las direcciones de búsqueda como

$$r_j = \psi_j(A)\phi_j(A)r_0, \quad p_j = \psi_j(A)\pi_j(A)r_0, \tag{3.53}$$

en donde $\phi_j(A)$ y $\pi_j(A)$ son polinomios asociados al método BCG definidos por el método CGS [91] como

$$\phi_{j+1}(A) = \phi_j(A) - \alpha_j A\pi_j(A), \quad \pi_{j+1}(A) = \phi_{j+1}(A) + \beta_j \pi_j(A), \tag{3.54}$$

y $\psi_j(A)$ es un nuevo polinomio que se define recursivamente en cada paso del algoritmo con el fin de estabilizar el comportamiento de la convergencia del mismo, y que BCGSTAB define con la siguiente recurrencia

$$\psi_{j+1}(A) = (I - \omega_j A)\psi_j(A), \tag{3.55}$$

donde ω_j es un escalar que toma un valor que permita dar un paso descendente en la dirección del residual.

Para derivar las relaciones de recurrencia de los residuales y las direcciones de búsqueda a partir de las ecuaciones (3.53), primero se desarrollan los términos $\psi_j(A)\phi_j(A)$ y

$\psi_j(A)\pi_j(A)$, utilizando las expresiones (3.54), como sigue

$$\begin{aligned}
\psi_{j+1}(A)\phi_{j+1}(A) &= (I-\omega_j A)\,\psi_j(A)\,(\phi_j(A)-\alpha_j A\pi_j(A)) \\
&= (I-\omega_j A)\,(\psi_j(A)\phi_j(A)-\alpha_j A\psi_j(A)\pi_j(A))\,, \qquad (3.56)\\
\psi_{j+1}(A)\pi_{j+1}(A) &= \psi_{j+1}(A)\,(\phi_{j+1}(A)+\beta_j\pi_j(A)) \\
&= \psi_{j+1}(A)\phi_{j+1}(A)+\beta_j\pi_{j+1}(A)\pi_j(A) \\
&= \psi_{j+1}(A)\phi_{j+1}(A)+\beta_j\,(I-\omega_j A)\,\psi_j(A)\pi_j(A). \qquad (3.57)
\end{aligned}$$

Sustituyendo las ecuaciones (3.56) y (3.57) en las ecuaciones (3.53), se tiene que

$$\begin{aligned}
r_{j+1} &= (I-\omega_j A)\,(\psi_j(A)\phi_j(A)r_0-\alpha_j A\psi_j(A)\pi_j(A)r_0) \\
&= (I-\omega_j A)\,(r_j-\alpha_j Ap_j)\,, \qquad (3.58)\\
p_{j+1} &= \psi_{j+1}(A)\phi_{j+1}(A)r_0+\beta_j\,(I-\omega_j A)\,\psi_j(A)\pi_j(A)r_0 \\
&= r_{j+1}+\beta_j\,(I-\omega_j A)\,p_j. \qquad (3.59)
\end{aligned}$$

Ahora, falta calcular los valores de los escalares β_j, α_j y ω_j (los detalles de estos cálculos pueden encontrarse en [91]). De acuerdo al método BCG

$$\beta_j=\frac{(r^*_{j+1})^T r_{j+1}}{(r^*_j)^T r_j},$$

donde $(r^*_j)^T r_j=(\phi_j(A^T)r^*_0)^T\phi_j(A)r_0=(r^*_0)^T\phi_j(A)^2r_0$, que no puede calcularse por no contar con los vectores $\phi_j(A^T)r^*_0$, $\phi_j(A)r_0$ o $\phi_j(A)^2r_0$; por esto, se relaciona con el siguiente escalar

$$\tilde{\rho}_j=(\psi_j(A^T)r^*_0)^T\phi_j(A)r_0=(r^*_0)^T(\psi_j(A^T)\phi_j(A)r_0)=(r^*_0)^T r_j.$$

Desarrollando $\phi_j(A^T)r^*$ con $\eta^j_1(A^T)^j r^*_0+\eta^j_2(A^T)^{j-1}r^*_0+...$, incorporando la ortogonalidad de $\phi_j(A)r_0$ respecto de $\left(A^T\right)^i r^*_0$, para toda $i<j$, y tomando en cuenta que sólo los coeficientes principales de los polinomios son relevantes, suponer que γ^j_1 es el principal coeficiente del polinomio $\phi_j(A)$, se tiene que

$$\tilde{\rho}_j=\left(\frac{\eta^j_1}{\gamma^j_1}\phi_j(A^T)r^*_0\right)^T(\phi_j(A)r_0)=\frac{\eta^j_1}{\gamma^j_1}(r^*_j)^T r_j.$$

Examinando las relaciones de recurrencia para $\phi_{j+1}(A)$ y $\psi_{j+1}(A)$, los coeficientes principales de estos polinomios cumplen que $\eta^{j+1}_1=-\omega_j\eta^j_1$, $\gamma^{j+1}_1=-\alpha_j\gamma^j_1$, con lo que

$$\frac{\tilde{\rho}_{j+1}}{\tilde{\rho}_j}=\frac{\omega_j}{\alpha_j}\frac{(\phi_{j+1}(A^T)r^*_0)^T\phi_{j+1}(A)r_0}{(\phi_j(A^T)r^*_0)^T\phi_j(A)r_0},\quad \beta_j=\frac{\tilde{\rho}_{j+1}}{\tilde{\rho}_j}\frac{\alpha_j}{\omega_j}.$$

De forma similar al cálculo de β_j, aplicando una simple fórmula de recurrencia, α_j se pude expresar como

$$\alpha_j=\frac{\left(\phi_j(A^T)r^*_0\right)^T(\phi_j(A)r_0)}{(\pi_j(A^T)r^*_0)^T(A\pi_j(A)r_0)},$$

desarrollando $\phi_j(A^T)r_0^*$ y $\pi_j(A^T)r_0^*$, tomando los principales coeficientes de sus polinomios (que resultan ser los mismos), se tiene que

$$\alpha_j = \frac{\left(\psi_j(A^T)r_0^*\right)^T\left(\phi_j(A)r_0\right)}{\left(\psi_j(A^T)r_0^*\right)^T\left(A\pi_j(A)r_0\right)} = \frac{\left(r_0^*\right)^T\left(\psi_j(A)\phi_j(A)r_0\right)}{\left(r_0^*\right)^T\left(A\psi_j(A)\pi_j(A)r_0\right)} = \frac{\tilde{\rho}_j}{\left(r_0^*\right)^T\left(Ap_j\right)}.$$

Ahora, para el cálculo de ω_j, a partir de la ecuación (3.58), si se define

$$s_j = r_j - \alpha_j A p_j, \tag{3.60}$$

el valor óptimo de ω_j que interviene en la construcción del polinomio reductor $\psi_j(A)$, se obtiene de minimizar la norma del residuo $r_{j+1} = s_j - \omega A s_j$, para esto

$$\begin{aligned} \| r_{j+1} \|^2 &= s_j^T s_j - 2\omega_j \left(As_j\right)^T s_j + \left(As_j\right)^T \left(As_j\right) \\ \Rightarrow \frac{\partial \| r_{j+1} \|^2}{\partial \omega_j} &= -2\left(As_j\right)^T s_j + 2\omega_j \left(As_j\right)^T \left(As_j\right), \end{aligned}$$

donde la derivada parcial es cero cuando

$$\omega_j = \frac{s_j^T\left(As_j\right)}{\left(As_j\right)^T\left(As_j\right)}. \tag{3.61}$$

Sustituyendo las expresiones (3.61) y (3.60) en la ecuación (3.58) se tiene que $r_{j+1} = r_j - \alpha_j A p_j - \omega_j A s_j$, con lo que se puede calcular la aproximación a la solución del sistema lineal no simétrico $Ax = b$

$$x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j.$$

En el algoritmo 10 se puede ver el procedimiento completo de el método BCGSTAB.

Algoritmo 10 Método Gradiente Biconjugado Estabilizado (BCGSTAB).

01 $r_0 = b - Ax_0$
02 Elegir r_0^* arbitrariamente
03 $p_0 = r_0$;
04 Para $j = 0, 1, \ldots$ Hasta convergencia
05 $\quad \alpha_j = (r_0^*)^T r_j / (r_0^*)^T (Ap_j)$
07 $\quad s_j = r_j - \alpha_j A p_j$
08 $\quad \omega_j = s_j^T (As_j) / (As_j)^T (As_j)$
09 $\quad x_{j+1}^* = x_j + \alpha_j p_j + \omega_j s_j$
10 $\quad r_{j+1} = s_j - \omega_j A s_j$
11 $\quad \beta_j = \left[(r_0^*)^T r_{j+1} / (r_0^*)^T r_j\right] [\alpha_j / \omega_j]$
12 $\quad p_{j+1} = r_{j+1} + \beta_j \left(p_j - \omega_j A p_j\right)$

3.4.6. Método Residuo Cuasi–mínimo Libre Transpuesto (TFQMR)

El método Residuo Cuasi–mínimo Libre Transpuesto (TFQMR) es un método que resuelve sistemas de ecuaciones no simétricos de la forma $Ax = b$, con $A \in \Re^{n\times n}$ y $x, b \in \Re^n$, calculando una aproximación en el subespacio de Krylov $\mathcal{K}_m$ y que cumple con las restricciones de Petrov–Galerkin y del residuo mínimo, por lo que algunos autores lo consideran un método híbrido [22].

El método TFQMR tiene como base el método CGS y para resolver un sistema lineal no simétrico $Ax = b$ parte de la recurrencia

$$x_m = x_{m-1} + \alpha_{m-1} u_{m-1}, \tag{3.62}$$

que equivale a

$$x_m = x_0 + U_m z_m, \tag{3.63}$$

donde se definen la matriz U_m y el vector z_m como $U_m = [u_0, u_1, ...u_{m-1}]$ y $z_m = [\alpha_0, \alpha_1, ..., \alpha_{m-1}]$.

Con base en las ecuaciones (3.62) y (3.63), el residual de x_m puede expresarse como

$$\begin{aligned} r_m &= r_0 - AU_m z_m & (3.64)\\ &= r_{m-1} - \alpha_{m-1} A u_{m-1} & (3.65) \end{aligned}$$

De acuerdo a la ecuación (3.65) se tiene que $Au_i = \frac{1}{\alpha_i}(r_i - r_{i+1})$, que en forma matricial se expresa como $AU_m = R_{m+1}\tilde{B}_m$, donde $R_{m+1} = [r_0, r_1, ..., r_m]$ $\tilde{B}_m$ es la matriz de tamaño $(m+1)\times m$ definida como

$$\tilde{B}_m = \begin{pmatrix} 1 & 0 & ... & & 0 \\ -1 & 1 & & & \vdots \\ 0 & -1 & 1 & ... & \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & -1 & 1 \\ 0 & ... & & & -1 \end{pmatrix} \times diag\left\{\frac{1}{\alpha_0}, \frac{1}{\alpha_1}, ..., \frac{1}{\alpha_{m-1}}\right\}.$$

Con estas definiciones, la ecuación (3.64) se expresa como

$$r_m = r_0 - AU_m z_m = r_0 - R_{m+1}\tilde{B}_m z_m = R_{m+1}e_1 - R_{m+1}\tilde{B}_m z_m = R_{m+1}\left(e_1 - \tilde{B}_m z_m\right). \tag{3.66}$$

Para que la 2–norma de cada columna de R_{m+1} sea unitaria, se le multiplica por la derecha por una matriz diagonal adecuada. Sea esa matriz diagonal la inversa de la

matriz $\Delta_{m+1} = diag\,[\delta_0, \delta_1, ..., \delta_m]$, entonces la ecuación (3.66) queda como

$$\begin{aligned} r_m &= R_{m+1}\Delta_{m+1}^{-1}\Delta_{m+1}\left(e_1 - \tilde{B}_m z_m\right) = R_{m+1}\Delta_{m+1}^{-1}\left(\delta_0 e_1 - \Delta_{m+1}\tilde{B}_m z_m\right) \\ &= R_{m+1}\Delta_{m+1}^{-1}\left(\delta_0 e_1 - \tilde{H}_m z_m\right). \end{aligned}$$

Al igual que los métodos vistos en secciones previas, el método TFQMR trata de reducir el residual. Si se intenta hacer al estilo del método GMRES, es decir, minimizando la 2–norma del residual, se necesita que R_{m+1} sea ortogonal para que la 2–norma puediera expresarse como sigue

$$\| r_m \|_2 = \| R_{m+1}\Delta_{m+1}^{-1}\left(\delta_0 e_1 - \tilde{H}_m z_m\right) \|_2 = \| \left(\delta_0 e_1 - \tilde{H}_m z_m\right) \|_2,$$

pero, en este caso, $R_{m+1}\Delta_{m+1}^{-1}$ no es ortogonal. Sin embargo, se puede intentar minimizar

$$\| \left(\delta_0 e_1 - \tilde{H}_m z_m\right) \|_2,$$

con lo que el método cuasi–minimiza el residual. Entonces, calculado el vector z_m se calcula, a su vez, x_m teniendo una aproximación a la solución de $Ax = b$. Con esto, formalmente, se define el método TFQRM.

Ahora, para expresar las iteraciones del método de manera progresiva, se introduce el Lema 2, que puede verse en [91].

Lema 2 *Sea $\tilde{H}_m$ una matriz Hessenberg superior $(m+1) \times m$. Sea Q_m el producto de matrices $Q_m = \Omega_m\Omega_{m-1}\ldots\Omega_1$, donde Ω_i $(i = 1, 2, ..., m)$ es una matriz de rotación para transformar a la matriz $\tilde{H}_m$ en una matriz triangular superior R_m. Sean $\tilde{R}_1$ la parte superior $m \times m$ de la matriz $Q_{m-1}\tilde{H}_m$ y $\tilde{c}_1$ el vector de los primeros m componentes del vector $Q_{m-1}\beta e_1$. Sean R_1 la parte superior $m \times m$ de la matriz $Q_m\tilde{H}_m$ y c_1 el vector de los primeros m componentes del vector $Q_m\beta e_1$. Sean $\tilde{y}_m = \tilde{R}_1^{-1}\tilde{c}_1$, $y_m = R_1^{-1}c_1$, los vectores obtenidos por los métodos CGS y TFQMR, respectivamente. Entonces*

$$y_m - \begin{bmatrix} y_{m-1} \\ 0 \end{bmatrix} = c_m^2\left(\tilde{y}_m - \begin{bmatrix} y_{m-1} \\ 0 \end{bmatrix}\right),$$

donde c_m es el coseno utilizado en la m–ésima rotación Ω_m.

Suponiendo que $\tilde{x}_m$ es vector calculado por el método CGS, por el Lema 2, la iteración del método TFQMR satisface la iteración

$$\begin{aligned} & x_m - x_{m-1} = c_m^2\left(\tilde{x}_m - x_{m-1}\right), \Rightarrow \frac{1}{\alpha_{m-1}}\left(x_m - x_{m-1}\right) = \frac{c_m^2}{\alpha_{m-1}}\left(\tilde{x}_m - x_{m-1}\right) \\ \Rightarrow \quad & \frac{1}{c_m^2\alpha_{m-1}}\left(x_m - x_{m-1}\right) = \frac{1}{\alpha_{m-1}}\left(\tilde{x}_m - x_{m-1}\right), \end{aligned}$$

definiendo $\eta_m = c_m^2 \alpha_{m-1}$ y $d_m = (\tilde{x}_m - x_{m-1})/\alpha_{m-1}$, se tiene que

$$\frac{1}{\eta_m}(x_m - x_{m-1}) = d_m \;\Rightarrow\; x_m = x_{m-1} + \eta_m d_m. \tag{3.67}$$

Ahora se necesita encontrar una recurrencia para d_m. Para esto, de acuerdo a la ecuación (3.62), $\tilde{x}_m$ puede expresarse como $\tilde{x}_m = \tilde{x}_{m-1} + \alpha_{m-1} u_{m-1}$. De la definición de d_m se tiene que

$$\begin{aligned} d_m &= \frac{1}{\alpha_{m-1}}(\tilde{x}_m - \tilde{x}_{m-1} + \tilde{x}_{m-1} - x_{m-1}) \\ &= u_{m-1} + \frac{1}{\alpha_{m-1}}(\tilde{x}_{m-1} - x_{m-2} - (x_{m-1} - x_{m-2})) \\ &= u_{m-1} + \frac{1 - c_{m-1}^2}{\alpha_{m-1}}(\tilde{x}_{m-1} - x_{m-2}) \\ &= u_{m-1} + \frac{\left(1 - c_{m-1}^2\right)\eta_{m-1}}{c_{m-1}^2 \alpha_{m-1}} d_{m-1}. \end{aligned}$$

En la expresión anterior, el término $\left(1 - c_{m-1}^2\right)/c_{m-1}^2$ es la tangente cuadrada del ángulo ϵ utilizado en la $(m-1)$–ésima rotación. A esta tangente se le denota con θ_{m-1}, entonces, para m

$$\begin{aligned} \theta_m = tan^2(\epsilon) = \frac{1 - c_m^2}{c_m^2} \;&\Rightarrow\; \theta_m = tan(\epsilon) = \sqrt{\frac{1}{c^2} - 1} \\ \Rightarrow\quad \theta_m = \frac{sen(\epsilon)}{cos(\epsilon)}, \;\text{ y }\; c_m^2 &= \frac{1}{\theta_m^2 + 1}. \end{aligned}$$

Para calcular los valores de $s_m = sen(\epsilon)$ y $c_m = cos(\epsilon)$, se examina la matriz $\tilde{H}_m$

$$\tilde{H}_m = \begin{pmatrix} \delta_0 & 0 & ... & & 0 \\ -\delta_1 & \delta_1 & & & \vdots \\ 0 & -\delta_2 & \delta_2 & ... & \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & -\delta_m & \delta_m \\ 0 & ... & & & -\delta_{m+1} \end{pmatrix} \times diag\left\{\frac{1}{\alpha_0}, \frac{1}{\alpha_1}, ..., \frac{1}{\alpha_{m-1}}\right\}.$$

Ignorando el escalamiento de las columnas de $\tilde{H}_m$ que hace la matriz diagonal $diag\{1/\alpha_i\}_{i=0,1,...,m-1}$, ya que no afecta la determinación de las rotaciones, $\tilde{H}_m$ después de j rotaciones tiene

la forma

$$\Omega_j ... \Omega_1 \tilde{H}_m = \begin{pmatrix} * & * & & & & & & \\ & * & * & & & & & \\ & & \ddots & \ddots & & & & \\ & & & \tau_j & 0 & & & \\ & & & -\delta_{j+1} & \delta_{j+1} & & & \\ & & & & \ddots & \ddots & & \\ & & & & & & -\delta_{m+1} \end{pmatrix}.$$

Entonces, la siguiente rotación está determinada por

$$s_{j+1} = \frac{-\delta_{j+1}}{\sqrt{\tau_j^2 + \delta_{j+1}^2}}, \quad c_{j+1} = \frac{\tau_j}{\sqrt{\tau_j^2 + \delta_{j+1}^2}}, \quad \theta_{j+1} = \frac{-\delta_{j+1}}{\tau_j}.$$

Entonces, después de aplicar la $j+1$–ésima rotación a la matriz anterior, se tiene que el elemento δ_{j+1} localizado en la posición $(j+1, j+1)$ de la matriz, se transforma en

$$\tau_{j+1} = \delta_{j+1} \times c_{j+1} = \frac{\tau_j \delta_{j+1}}{\sqrt{\tau_j^2 + \delta_{j+1}^2}} = -\tau_j s_{j+1} = -\tau_j \theta_{j+1} c_{j+1}.$$

Con todo el desarrollo anterior se puede expresar una recurrencia para d_m de la ecuación (3.67), que se presenta a continuación para d_{m+1}

$$d_{m+1} = u_m + \frac{\theta_m^2}{\alpha_m} \eta_m d_m,$$

donde $\theta_{m+1}^2 = \frac{\delta_{m+1}}{\tau_m}$, $c_{m+1} = \left(1 + \theta_{m+1}^2\right)^{\frac{1}{2}}$, $\tau_{m+1} = \tau_m \theta_{m+1} c_{m+1}$ y $\eta_{m+1} = c_{m+1}^2 \alpha_m$.

En el algoritmo 11 se presentan los pasos necesarios para calcular una aproximación a la solución del sistema $Ax = b$. Debe tomarse en cuenta que se ha cuasi–minimizado el residual, con lo que el término r_m del algoritmo lo representa parcialmente. Para ver detalles de este método consúltese [91].

3.5. Precondicionadores

La razón de convergencia de los métodos iterativos que resuelven sistemas de ecuaciones lineales de la forma $Ax = b$, depende de las propiedades espectrales de la matriz de coeficientes A. Si el sistema lineal se transforma en otro equivalente (que tenga la misma solución que el sistema original) con propiedades espectrales más favorables, se puede acelerar la convergencia de los métodos iterativos. Un precondicionador es una matriz que realiza dicha transformación.

Por ejemplo, si M es un precondicionador que aproxima a la matriz de coeficientes A en alguna forma, el sistema lineal transformado $M^{-1}Ax = M^{-1}b$ tiene la misma

Algoritmo 11 Método Residuo Cuasi–mínimo Libre Transpuesto (TFQMR).

01 $r_0 = u_0 = b - Ax_0$
02 $v_0 = Au_0$
03 $d_0 = 0$
04 Elegir r_0^* tal que $\rho_0 = r_0^T r_0^* \neq 0$
05 Para $m = 0, 1, ...$ Hasta convergencia
06 Si m es par
07 $\alpha_{m+1} = \alpha_m = \rho_m / (r_0^*)^T v_m$
08 $u_{m+1} = u_m - \alpha_m v_m$
09 $r_{m+1} = r_m - \alpha_m A u_m$
10 $d_{m+1} = u_m + (\theta_m^2 / \alpha_m) \eta_m d_m$
11 $\theta_{m+1} = \| r_{m+1} \|_2 / \tau_m$
12 $c_{m+1} = \left(1 + \theta_{m+1}^2\right)^{\frac{1}{2}}$
13 $\tau_{m+1} = \tau_m \theta_{m+1} c_{m=1}$
14 $\eta_{n+1} = c_{m+1}^2 \alpha_m$
15 $x_{m+1} = x_m + \eta_{m+1} d_{m+1}$
16 Si m es impar
17 $\rho_{m+1} = (r_0^*)^T r_{m+1}$
18 $\beta_{m-1} = \rho_{m+1} / \rho_{m-1}$
19 $u_{m+1} = r_{m+1} + \beta_{m-1} u_m$
20 $v_{m+1} = A u_{m+1} + \beta_{m+1} \left(A u_m + \beta_{m-1} v_{m-1}\right)$

solución que el sistema original, pero las propiedades espectrales de $M^{-1}A$ podrían ser más favorables.

Al utilizar precondicionadores se necesita contrastar el costo computacional extra que implica su construcción y aplicación, y la ganancia en la rapidez de convergencia. En el uso de precondicionadores se busca que su aplicación tenga un costo comparable al de una iteración de un método iterativo. Por otra parte, desde el punto de vista del paralelismo, se necesita valorar qué tan eficaz es un precondicionador en el sentido clásico frente a su eficiencia paralela, pues muchos precondicionadores tradicionales tienen largas secuencias de procesamientos no paralelizables.

Entonces, el problema de encontrar un precondicionador para un sistema lineal es encontrar una matriz M tal que [22]

1. M es una buena aproximación a A en algún sentido,

2. El costo de construir M no es prohibitivo,

3. El sistema $M^{-1}Ax = M^{-1}b$ es mucho más fácil de resolver que el sistema original.

Dependiendo de cómo se aplique un precondicionador al sistema, las formas de precondicionamiento pueden ser

1. Precondicionamiento por la izquierda: $M^{-1}Ax = M^{-1}b$.
2. Precondicionamiento por la derecha: $AM^{-1}Mx = b$.
3. Precondicionamiento por ambos lados: $M_1^{-1}AM_2^{-1}M_2x = M_1^{-1}b$, cuando M puede factorizarse en M_1M_2.

En esta sección se revisan precondicionadores basados en métodos iterativos estacionarios, como: los de Jacobi, Gauss-Sidel y SOR simétrico; también se revisan precondicionadores basados en factorizaciones incompletas, como: factorización sin relleno (ILU0), factorización con relleno en función de la estructura de A (ILUK, MILU) y factorización con relleno en función de umbrales numéricos (ILUT).

3.5.1. Precondicionadores Jacobi, Gauss–Seidel y SOR Simétrico

Una iteración de punto fijo para resolver un sistema lineal $Ax = b$ toma la forma general

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b, \tag{3.68}$$

con $A = M - N$. Por otra parte, se ha visto en la ecuación (3.1) de la sección 3.3 que una iteración de punto fijo también puede expresarse como

$$x^{(k+1)} = Tx^{(k)} + c, \tag{3.69}$$

con $c = M^{-1}b$ y $T = M^{-1}N = I - M^{-1}A$. Con la iteración (3.69) se resuelve el sistema $(I - T)x = c$, el cual, como $T = I - M^{-1}A$, se puede reescribir como

$$M^{-1}Ax = M^{-1}b. \tag{3.70}$$

El sistema (3.70) se llama *sistema precondicionado* asociado con $A = M - N$ y la iteración (3.69) se llama iteración de *punto fijo* sobre este sistema precondicionado, con matriz de iteración T.

Para Jacobi se ha obtenido la iteración de punto fijo (ecuación (3.3)) $x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b$, siendo D, L y U matrices con los elementos de la diagonal, de la triangular inferior estricta y de la triangular superior estricta de A, respectivamente (ver Figura 3.1). De esta iteración se ve claramente que el precondicionador y la matriz de iteración son, correspondientemente,

$$M_J = D, \quad T_J = D^{-1}(L + U) = I - D^{-1}A.$$

Para Gauss–Seidel también se ha obtenido la iteración de punto fijo (ecuación (3.5)) $x^{(k+1)} = (D - L)^{-1}Ux^{(k)} + (D - L)^{-1}b$, donde D, L y U están definidas como en

Jacobi. De esta iteración se vuelve a ver claramente que el precondicionador y la matriz de iteración son, respectivamente,

$$M_{GS} = (D - L), \quad T_{GS} = (D - L)^{-1} U = I - (D - L)^{-1} A.$$

Ahora, partiendo de la iteración de punto fijo de SOR (ecuación (3.11)), que es $x^{(k+1)} = (D - \omega L)^{-1} [(1 - \omega) D + \omega U] x^{(k)} + \omega (D - \omega L)^{-1} b$, donde D, L y U, nuevamente están definidas como en Jacobi y Gauss–Seidel, se puede definir la iteración simétrica de SOR (SSOR) como un paso de SOR seguido por un "paso hacia atrás" [91] de SOR como sigue

$$\begin{aligned} x^{(k+1/2)} &= (D - \omega L)^{-1} [(1 - \omega) D + \omega U] x^{(k)} + \omega (D - \omega L)^{-1} b, \\ x^{(k+1)} &= (D - \omega U)^{-1} [(1 - \omega) D + \omega L] x^{(k)} + \omega (D - \omega U)^{-1} b, \end{aligned}$$

con lo que se obtiene la iteración de SSOR de la forma $x^{(k+1)} = M^{-1} N x^{(k)} + M^{-1} b = T x^{(k)} + c$, donde

$$\begin{aligned} M_{SSOR} &= (D - \omega L) D^{-1} (D - \omega U), \\ T_{SSOR} &= (D - \omega U)^{-1} ((1 - \omega) D + \omega L) (D - \omega L)^{-1} ((1 - \omega) D + \omega U). \end{aligned}$$

Si se toma $\omega = 1$ se obtiene la iteración de Gauss-Seidel Simétrica (GSS).

3.5.2. Factorización LU Incompleta (ILU) General

En esta sección se revisa un precondicionador que realiza una factorización incompleta de una matriz general dispersa $A \in \Re^{n \times n}$. Una factorización incompleta de una matriz dispersa es deseable porque, a diferencia de una factorización completa que presenta el problema de relleno en las matrices factores (con lo que dejan de ser dispersas), las matrices que resultan de la factorización incompleta respetan el patrón de dispersión de la matriz original en cierta proporción, con lo que se busca controlar que la cantidad de procesamiento extra por el uso de precondicionadores no se incremente excesivamente. En concreto, se aborda la factorización LU Incompleta (ILU).

La factorización general ILU de A calcula las matrices L y U, que tienen la misma estructura diferente de cero que la triangular inferior y superior de A, respectivamente, tales que la matriz residual $R = LU - A$ satisface ciertas restricciones, como tener ceros en algunas de sus entradas.

Para introducir un algoritmo general de la factorización ILU, se considera que $A = \{a_{ij}\}_{i,j=1,2,\ldots,n}$ es una M–matriz, pues se ha demostrado [8] que una factorización incompleta existe para una M–matriz.

Renombrando a la matriz A con A_1, suponer que se aplica un paso de la eliminación Gaussiana a A_1 y que se obtiene como resultado la matriz $\tilde{A}_1$ (que también es M–matriz, como se demuestra en [91]) y suponer que algunos elementos fuera de la diagonal

se eliminan de $\tilde{A}_1$, obteniéndose $\tilde{\tilde{A}}_1 = \tilde{A}_1 + R$ ($\tilde{\tilde{A}}_1$ también es M–matriz, como se demuestra en [91]), donde los elementos de R son tales que $r_{ii} = 0$ $(i = 1, 2, ..., n)$ y $r_{ij} \geq 0$ $(i, j = 1, 2, ..., n$, con $i \neq j)$.

El proceso anterior se repite sobre la M–matriz de tamaño $(n-1) \times (n-1)$, $A_2 = \tilde{\tilde{A}}_1(2:n, 2:n)$ (A_2 se obtiene de $\tilde{\tilde{A}}_1$ eliminando su primer renglón y su primera columna), y así sucesivamente hasta obtener la factorización incompleta de A.

La selección de los elementos que se van eliminando de las matrices puede obedecer a un patrón de ceros estático, denotado con P, definido como

$$P \subset \{(i,j) | i \neq j;\ 1 \leq i, j \leq n\}.$$

Entonces, una factorización ILU_P se calcula con en el algoritmo 12, obteniéndose la factorización incompleta $A = LU - R$ (cuya demostración puede encontrarse en [91]). En el i–ésimo paso del algoritmo, el i–ésimo renglón de A se sobreescribe con los i–ésimos renglones de L y U incompletas, y se acceden a los $i-1$ renglones anteriores de A, donde se encuentran los renglones calculados de U y L.

Algoritmo 12 Factorización ILU General.

```
01 Para i = 2, 3, ...n
02     Para k = 1, ..., i − 1 y si (i, k) ∉ P
03         a_ik = a_ik / a_kk
04         Para j = k + 1, k + 2, ..., n y para (i, j) ∉ P
05             a_ij = a_ij − a_ik × a_kj
```

3.5.3. Factorización ILU sin Relleno (ILU0)

En esta factorización el patrón de ceros P es igual que el patrón de ceros de A, con lo que las matrices L y U tienen tantos ceros como A en las mismas posiciones de su triangular inferior y superior, respectivamente. Con esto, no necesariamente el resultado del producto LU es exactamente A, ya que LU puede *rellenar* entradas en las que A tenga ceros. Pero, si se ignoran estas entradas *rellenas* se pueden construir matrices L y U cuyo producto sea igual que A en las entradas diferentes de cero de A ($NZ(A)$). Entonces, la factorización ILU0 consiste justamente en construir L y U tales que los ceros de $A - LU$ están localizados en las entradas $NZ(A)$.

El algoritmo de la factorización ILU0 es el mismo que el algoritmo 12, en donde el patrón de ceros P debe ser exactament igual que el patrón de ceros de A.

La factorización ILU0 puede ser insuficiente para alcanzar una razón de convergencia adecuada al momento de resolver el sistema lineal (ver [91]), debido a que no permite

relleno en la factorización. Flexibilizar este punto es la política del método que se verá a continuación, lo que puede llevar a factorizaciones más exactas.

3.5.4. Factorización ILU con Nivel de Relleno K (ILUK)

La factorización ILUK permite cierto nivel de relleno (denotado con K) en la factorización. El relleno es permitido en función de la estructura de A. Así, por ejemplo, para aplicar una factorización con nivel $K = 1$, ILU(1), se aplica ILU0 y se obtienen L y U, se define P con el patrón de ceros del producto LU. Asignar a A un nuevo patrón de elementos diferentes de cero definido como

$$NZ_1(A) = NZ(A) + \text{ las posiciones } \textit{rellenas} \text{ del producto } LU,$$

donde $NZ(A)$ es el verdadero patrón de elementos diferentes de cero de A. Nuevamente se aplica ILU0 a la A con patrón de ceros $NZ_1(A)$, con lo que se obtienen los factores L_1 y U_1.

Para generalizar el proceso anterior y aplicarlo a matrices dispersas generales, se define el concepto de Nivel de Relleno, que se concibe como un atributo de un componente de $A = \{a_{ij}\}_{i,j=1,2,\ldots,n}$ que es modificado por eliminación Gaussiana.

Definición 4 *El nivel inicial de relleno de un elemento a_{ij} de una matriz dispersa A se define como*

$$nivel_{ij} = \begin{cases} 0 & si \;\; a_{ij} \neq 0 \;\; o \; si \;\; i = j, \\ \infty & en \; otro \; caso. \end{cases}$$

Cada vez que este elemento es modificado por eliminación de Gauss, su nivel de relleno se actualiza como $nivel_{ij} = min\{nivel_{ij}, nivel_{ik} + nivel_{kj} + 1\}$.

Por la definición anterior (que siempre toma los mínimos para actualizar el nivel de relleno), si un componente $a_{ij} \neq 0$ en la matriz original A, el nivel de relleno del componente es 0, nivel que no cambiará durante la factorización (el componente siempre será diferente de cero); por otra parte, la forma en que se actualizan los niveles de relleno es una estrategia natural para descartar componentes o mantenerlos durante la factorización. Con esto, todos los elementos con nivel de relleno menor que K se mantendrán. El patrón de ceros P_K se define antes de empezar el proceso de factorización como $P_K = \{(i,j) | nivel_{ij} > K\}$.

En el algoritmo 13 se reúne el proceso de la factorización ILUK. En este algoritmo se identifican tres inconvenientes:

1. No se puede predecir el nivel de relleno ni la cantidad de trabajo computacional para $K > 0$.

2. La actualización de los niveles de relleno puede ser costosa.

3. El nivel de relleno para matrices indefinidas puede no ser un buen indicador del tamaño de los componentes que han sido eliminados, con lo que si se eliminan elementos con valores grandes, se obtendrá una factorización inexacta en el sentido de que $R = LU - A$ no será pequeño (que podría llevar a un número grande de iteraciones para calcular la solución del sistema).

Algoritmo 13 Factorización ILU con nivel de relleno K (ILUK).

01 Definir $nivel_{ij}$, para $i, j = 1, 2, ..., n$, de acuerdo a la Definición 4
02 Para $i = 2, 3, ...n$
03 Para $k = 1, ..., i-1$ y para $nivel_{ik} \leq K$
04 $a_{ik} = a_{ik}/a_{kk}$
05 Para $j = 1, 2, ..., n$
06 $a_{ij} = a_{ij} - a_{ik} \times a_{kj}$
07 $nivel_{ij} = min\{nivel_{ij}, nivel_{ik} + nivel_{kj} + 1\}$
08 Asignar 0 a cualquier elemento en el renglón i con $nivel_{ij} > K$

3.5.5. Factorización ILU Modificada (MILU)

Las factorizaciones incompletas seleccionan elementos utilizando cierto criterio y, simplemente, se descartan asignándoles un cero. Hay estrategias que compensan esta asignación para reducir sus efectos. Una de ellas es tomar los elementos que se van eliminando durante el cálculo del i–ésimo renglón de las matrices L y U (durante la iteración k del algoritmo 13 o del algoritmo 12) y sumarlos a u_{ii}, la diagonal de U; a esta estrategia se le llama *compensación diagonal* y constituye un precondicionador ILU Modificado (MILU), que también permite cierto nivel de relleno en función de la estructura de A.

Un breve esquema de la compensación diagonal se da a continuación. Se denota con a_{i*} al renglón i–ésimo de A. Antes de comenzar el ciclo controlado por k en cualquiera de los algoritmos ILU, $u_{i*} = a_{i*}$. Las operaciones que se ejecutan durante la k–ésima iteración pueden representarse como $u_{i*} = u_{i*} - l_{i*}u_{k*}$, que en realidad se aplica sólamente a componentes $u_{ij} \neq 0$ $(j = k+1, k+2, ..., n)$, y para representar esto se tiene que $u_{i*} = u_{i*} - l_{ik}u_{k*} + r_{i*}^{(k)}$, donde

$$r_{ij} = \begin{cases} 0 & \text{si } (i,j) \notin P \text{ (si } U_{ij} \neq 0, \\ l_{ik}u_{kj} & \text{en otro caso.} \end{cases}$$

Con la definición de r_{ij} los componentes nulos de A se preservan. Así, al terminar el

ciclo controlado por k se tiene

$$u_{i*} = a_{i*} - \sum_{k=1}^{i-1} \left(l_{ik} u_{k*} - r_{i*}^{(k)} \right) = a_{i*} - \sum_{k=1}^{i-1} \left(l_{ik} u_{k*} \right) - r_{i*},$$

donde $r_{i*} = \sum_{k=1}^{i-1} \left(r_{i*}^{(k)} \right)$. Entonces, para el i–ésimo renglón procesado, los componentes que no se han considerado en la factorización porque se deben preservar los ceros de A están reunidos en el vector renglón u_{i*}; estos componentes se suman y el escalar resultante se suma, a su vez, al componente diagonal de U, es decir

$$u_{ii} = u_{ii} - (r_{i*} e), \quad \text{con } e = (1, 1, ..., 1)^T,$$

con lo que termina el cálculo del i–ésimo renglón de la factorización. En [90] se muestra que con esta estrategia $Ae = LUe$, es decir, cada renglón de A es igual que el correspondiente renglón de LU.

3.5.6. Factorización ILU con Umbral (ILUT)

A diferencia de las factorizaciones incompletas que permiten rellenar o no los componentes de una matriz en función de la estructura de A, la factorización ILU basada en umbral (ILUT) decide el relleno dependiendo de si la magnitud del componente es menor o mayor que cierto umbral. Con esto, el patrón de dispersión P es dinámico.

Un algoritmo ILUT genérico puede derivarse, por ejemplo, del algoritmo 12 (factorización ILU general) incluyéndole un conjunto de reglas para la eliminación de elementos pequeños. Así, si un elemento cumple con el conjunto de reglas, el elemento es sustituído por un cero. Las mismas reglas pueden aplicarse a todos los elementos de un renglón. En el algoritmo 14, w es un vector de trabajo que acumula combinaciones lineales de renglones dispersos en la eliminación, y w_k es su k–ésimo componente. a_i* denota el i–ésimo renglón de A.

En la línea 5 del algoritmo 14 podría aplicarse la siguiente regla: eliminar el elemento w_k (asignarle cero) si su magnitud es menor que cierto umbral τ, que puede ser, por ejemplo, el valor promedio del k–ésimo renglón o la 2–norma del k–ésimo renglón. En la línea 8 del algoritmo podría aplicarse un conjunto de reglas, por ejemplo, primero eliminar los componentes de w cuyas magnitudes sean menores que τ, después mantener sólamente los q componentes más grandes de w para L y los q más grandes para U, manteniendo siempre los componentes diagonales de U. Con esta combinación de reglas no sólo se controla el relleno con base en la magnitud de los componentes, sino que también se controla la cantidad de componentes diferentes de cero que se van a permitir.

Algoritmo 14 Factorización ILU con Umbral (ILUT).

01 Para $i = 1, 2, ...n$
02 $w = a_{i*}$
03 Para $k = 1, ..., i-1$ y cuando $w_{ik} \neq 0$
04 $w_k = w_k / a_{kk}$
05 Aplicar reglas de eliminación aw_k
06 Si $w_k \neq 0$
07 $w = w - w_k * u_k$
08 Aplicar reglas de eliminación aw
09 $l_{ij} = w_j$ Para $j = 1, 2, ..., i-1$
10 $u_{ij} = w_j$ Para $j = i, i+1, ..., n$
11 $w = 0$

3.6. Conclusiones

Existe una variedad muy amplia de métodos que resuelven y precondicionan sistemas lineales dispersos de gran dimensión. En este capítulo se han abordado algunos de los más utilizados y los que implementan las librerías empleadas en los experimentos numéricos de esta tesis.

Los métodos de resolución de sistemas lineales pueden clasificarse en directos e iterativos. Los métodos directos calculan la solución en un número finito de operaciones y los métodos iterativos generan una sucesión de aproximaciones que que pueden o no converger a una solución.

Los métodos directos realizan una descomposición de la matriz de coeficientes. Si esta matriz es dispersa, la descomposición trae como consecuencia el problema del relleno, con lo que la resolución del sistema requiere más tiempo de cómputo y más espacio de almacenamiento para su operación y representación. Para enfrentar este problema, existen técnicas de reordenamiento de las entradas de la matriz de coeficientes que intentan reducir este relleno. Además, a medida que aumenta la dimensión de los sistemas de ecuaciones, los métodos directos presentan problemas por los errores de redondeo que se acumulan durante el proceso. No obstante lo anterior, las soluciones encontradas por la aplicación de métodos directos resultan ser muy precisas.

El costo computacional de los métodos iterativos con un número moderado de iteraciones puede resultar menor que el costo de un método directo. Por ejemplo, la descomposición LU que realiza un método directo sobre una matriz densa es del orden de $O(n^3)$, mientras que los métodos iterativos estacionarios, como Jacobi, Gauss–Siedel y SOR, son del orden de $O(2 * NZ)$, donde NZ es el número de entradas diferentes de cero de la matriz de coeficientes. Por otra parte, el almacenamiento de los datos es menor en los métodos iterativos que en los directos, ya que los primeros preservan el caracter

disperso de los sistemas.

Dentro de los métodos iterativos se han revisado en este capítulo los estacionarios, como Jacobi (J), Gauss–Sidel (GS) y SOR; y los no estacionarios, como Residuo Mínimo Generalizado (GMRES), Gradiente Conjugado (GC), Gradiente Biconjugado (BCG), Gradiente Biconjugado Estabilizado (BCGSTAB) y Residuo Cuasi–mínimo Libre Transpuesto (TFQMR). Los métodos no estacionarios, a diferencia de los estacionarios, involucran en sus cálculos información que cambia en cada iteración. Por esto, los métodos iterativos no estacionarios presentan convergencia más rápida que los estacionarios. La convergencia de los métodos estacionarios depende de las propiedades de la matriz de coeficientes del sistema y la velocidad de convergencia depende del radio espectral de la matriz de iteración.

Los métodos iterativos no estacionarios vistos en este capítulo se basan en procesos de proyección (subespacios de Krylov) y tratan de minimizar cierta norma del vector residuo sobre un subespacio generado por la matriz de coeficientes del sistema así como ofrecer bajo costo en requerimientos de tiempo y espacio. Por otra parte, en estos métodos iterativos la operación fundamental es el producto matriz–vector.

Con los procesos de proyección no sólo se construyen métodos para resolver sistemas de ecuaciones lineales, sino métodos para calcular los valores propios de una matriz, que es uno de los problemas que surgen de la discretización de la EDN en el caso estacionario. Por este motivo, este capítulo ha introducido el método de Arnoldi, cuya operación fundamental es el producto matriz–vector.

La aceleración de la convergencia de los métodos iterativos puede obtenerse precondicionando los sistemas a resolver. En este capítulo se han revisado los precondicionadores de Jacobi, Gauss–Seidel y SOR Simétrico, y los basados en factorizaciones incompletas, como ILU0, ILUK, MILU e ILUT. ILU0 es una factorización que no permite relleno y requiere el mismo monto de almacenamiento que A, pero lleva a una representación cruda del sistema a resolver, lo que puede resultar en muchas iteraciones para converger a una solución [91]. Permitir ciertos niveles de relleno puede llevar a factorizaciones que permitan precondicionar al sistema más adecuadamente, por ejemplo ILUK y MILU permiten niveles de relleno con base en la estructura de la matriz. Otra política de relleno, que no se basa en la estructura de la matriz sino en las magnitudes de los valores de sus componentes es ILUT. Todas las variantes ILU propuestas tienen el objetivo de representar lo más fielmente posible las características del sistema a resolver, pero minimizando costos de operacón y almacenamiento. En general, las factorizaciones ILU requieren pocas iteraciones para converger, pero el costo de procesamiento de los factores puede ser alto [91].

Capítulo 4

Plataformas y Software de Computación

4.1. Introducción

Este capítulo muestra los elementos de hardware y software que fueron utilizados para la implementación secuencial y paralela de los métodos que resuelven los sistemas de ecuaciones lineales dispersos relacionados con la ecuación de difusión neutrónica multigrupo, tanto para el caso estacionario como dinámico y está organizado de la siguiente manera: La sección 4.2 describe las características de las máquinas secuenciales y paralelas sobre las que se realizaron las pruebas experimentales. En tanto que la sección 4.3 da cuenta de las distintas librerías numéricas de libre distribución que se han elegido para llevar a cabo las pruebas experimentales del capítulo 5, así como una descripción general de sus fundamentos y funcionamiento. Por último, en la sección 4.4 se enuncian algunas conclusiones de este capítulo.

4.2. Hardware

Los equipos hardware utilizados en esta memoria son Kubrick, Kefren y Jacquard. Estos son computadores paralelos con memoria distribuida que funcionan con el modelo de programación por paso de mensajes. A continuación se listan sus características más representativas.

4.2.1. Cluster Kubrick

Este es un cluster de memoria distribuida con 2 nodos biprocesador (figura 4.1). Los procesadores son de la marca Intel modelo Xeon a 2.2 GHz. Cada nodo comparte 4 Gbytes de RAM y están interconectados por una red Gigabit Ethernet bajo el sistema operativo LINUX Red Hat 8. En este cluster se han realizado experimentos secuenciales utilizado su versión de Matlab.

Figura 4.1: Cluster Intel Kubrick de la Universidad Politécnica de Valencia

4.2.2. Cluster Kefren

Kefren[1](figura 4.2) es un grupo de 20 nodos bi-procesador, donde cada nodo se encuentra conectado mediante una red SCI con topología Torus 2D en una malla de 4x5. Cada CPU es un Intel Xeon con una velocidad de 2GHz y 1 GB de memoria principal con sistema operativo LINUX Red Hat 8.0

El software usado en este cluster y que más adelante se explica con mayor amplitud es: SPARSKIT, PETSc, pARMS y SuperLU.

4.2.3. Cluster Jacquard

Jacquard[2] (figura 4.3) es un cluster AMD Opteron con 712 CPUs sobre el que se ejecuta el sistema operativo LINUX. La máquina es llamada así en honor de José María Jacquard, cuyo telar fué la primera máquina en usar tarjetas perforadas para controlar una secuencia de operaciones.

[1]http://www.grycap.upv.es
[2]http://www.nersc.gov/nusers/systems/jacquard/

Figura 4.2: Cluster Intel Kefren de la Universidad Politécnica de Valencia.

Jacquard tiene 356 nodos biprocesador para cálculos científicos. Cada procesador tiene una velocidad de reloj de 2.2GHz. Los procesadores en cada nodo comparten 6 GBytes de memoria y están interconectados con una red infiniband de alta velocidad. El software usado en este cluster corresponde a las librerías numéricas de PETSc e Hypre.

Figura 4.3: Cluster AMD Jacquard en el centro de investigación NERSC

4.3. Software

Entre los objetivos de la presente tesis, está la evaluación de software numérico secuencial y paralelo que permita resolver eficientemente los sistemas de ecuaciones lineales dispersos (SELD) que surgen de la discretización de la ecuación de difusión neutrónica en su forma estacionaria y dinámica. Por tal motivo, se ha tenido la oportunidad de trabajar con distintas herramientas software para cumplir los objetivos establecidos.

A continuación se exponen las librerías numéricas, tanto secuenciales como paralelas, así como un resumen de sus características de funcionamiento más importantes.

4.3.1. Matlab

MATLAB [28] Es un ambiente propietario de prueba y desarrollo de prototipos rápidos del álgebra lineal numérica. La versión utilizada fué 5.3.1 (R11.1)[3].

4.3.2. SuperLU

Se han realizado experimentos numéricos en esta memoria, utilizando la versión 2.0 de la librería SuperLU[4] [124] [125] [21]. SuperLU es una librería de libre distribución que implementa métodos directos (ver capítulo 3) para la resolución de SELD de ecuaciones de la forma $AX = B$, donde A es una matriz dispersa cuadrada de $n \times n$, no singular; X y B son matrices densas de $n \times nrhs$, donde $nrhs$ es el número de lados derechos y vectores solución.

La librería SuperLU usa eliminación de Gauss con técnicas de pivotamiento, donde las columnas de la matriz A puede ser preordenadas antes de la factorización. La librería SuperLU puede funcionar tanto con matrices reales como complejas y en precisión sencilla o doble. De este solver existen tres colecciones de subrutinas:

SuperLU. Rutina diseñada para procesadores secuenciales con una o más capas de jerarquías de memoria, y usa la técnica de pivotamiento parcial.

SuperLU_DIST. Rutina diseñada para arquitecturas paralelas con memoria distribuida que usan el estándar MPI [56] para la comunicación entre procesos. Aplica técnicas de pivotamiento estático.

SuperLU_MT. Rutina con tecnología multihilo (SMP), diseñada para multiprocesadores con memoria compartida. La técnica de pivotamiento es estática.

[3] Matlab es una marca registrada de The MathWorks, Inc.
[4] http://crd.lbl.gov/~xiaoye/superlu/

4.3.3. SPARSKIT

SPARSKIT[5] 2.0 [88] [90] es una librería numérica secuencial de libre distribución para la manipulación y trabajo con matrices dispersas y que nació como una idea del profesor Yousef Saad[6] mientras estuvo en el Center for Supercomputing Research and Development de la Universidad de Illinois durante los años 1986–1987. La librería SPARSKIT nace con el objetivo de facilitar el intercambio de matrices con fines de investigación entre otras cosas. La librería SPARSKIT en su versión 2.0 está escrita en FORTRAN 77 y los módulos que la conforman se muestran en la figura 4.4.

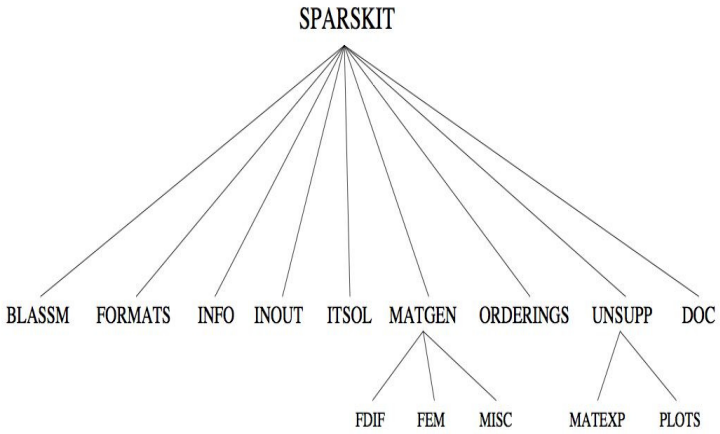

Figura 4.4: Módulos de la librería SPARSKIT.

Los módulos de la librería SPARSKIT que han sido usados en este trabajo de tesis son:

- FORMATS. Es el módulo de conversión de formatos, que actualmente maneja hasta 13 formatos diferentes de almacenamiento. Entre los formatos que se manejan están: formato denso (DNS), comprimido por fila (CSR), comprimido por columna (CSC), coordenado (COO), formato diagonal (DIA), formato en banda (BND), entre otros. En este trabajo, el formato que más se usó fué el comprimido por fila (CSR). Una descripción de los distintos formatos aquí mencionados puede encontrarse en [90] [88] [50].
- INOUT. Este módulo contiene rutinas para la lectura, escritura y visualización de la estructura de las matrices dispersas en formato Harwell–Boing [25].
- BLASSM. Este módulo contiene operaciones algebraicas básicas como $C = A + B$, $C = A + \beta B$, $C = AB$, entre otras. Operaciones muy comunes como las que involucran

[5] http://www-users.cs.umn.edu/saad/software/SPARSKIT/sparskit.html
[6] http://www-users.cs.umn.edu/~saad/

matrices dispersas y vectores, así como soluciones con sistemas triangulares se incluyen en el módulo MATVEC.

- INFO. Este módulo contiene rutinas que permiten conocer información básica y estadística de una matriz dispersa, como por ejemplo: el número de elementos diferentes de cero, número promedio de elementos distintos de cero por fila, ancho de banda, etc.
- ITSOL. Este módulo contiene una serie de rutinas que hacen uso de métodos iterativos basados en subespacios de Krylov básicos como: Gradiente Conjugado (GC), Residuo Mínimo Generalizado (GMRES), Gradiente Bi–Conjugado (BCG), Gradiente Bi–Conjugado estabilizado, entre otros. También contiene precondicionadores de tipo LU incompleto simple (ILU0), con umbral (ILUT), etc.

4.3.4. pARMS

La librería paralela *Parallel Algebraic Recursive Multilevel Solver* (pARMS) 0.2 [78] [95] es la versión paralela de ARMS [96] y es un conjunto de precondicionadores y solvers (aceleradores) iterativos de memoria distribuida enfocados a la solución de sistemas generales dispersos. Esta librería, también de libre distribución, tiene como elemento base las matrices dispersas distribuidas y se apoya en la solución de los sistemas del complemento de Schur distribuidos resultantes.

4.3.4.1. Sistemas Lineales Dispersos Distribuidos

Los sistemas lineales distribuidos [92] [94] proporcionan una representación algebraica de los sistemas de ecuaciones lineales dispersos que surgen en los métodos de descomposición de dominios. Un sistema distribuido típico surge, por ejemplo, de la discretización en elementos finitos de una ecuación diferencial parcial sobre un cierto dominio. La solución típica de estos sistemas en una computadora con memoria distribuida implica la partición de la malla del elemento finito y la asignación de un grupo de elementos que representan un sub–dominio físico a un procesador. Cada procesador ensambla solamente las ecuaciones locales correspondiente a su grupo de elementos asignados. Para el caso donde los sistemas son dados de manera algebraica, un grafo contiene vértices que corresponden a las filas de los sistemas lineales que pueden ser particionados. En cualquier caso, la idea general es que cada procesador mantiene un conjunto de ecuaciones (o filas del sistema global) y las variables incógnitas asociadas.

La figura 4.5 muestra una vista del 'dominio físico' de un sistema lineal disperso distribuido, para el cual una partición basada en vértices ha sido usada sin traslape de incógnitas. La librería pARMS distingue tres tipos de variables:

1. Variables interiores son aquellas que están acopladas (o emparejadas) únicamente con las variables locales por las ecuaciones;

2. Variables de interfase inter–dominio que son aquellas que están acopladas con variables no locales (externas) así como con variables locales; y

3. Variables de interfase externas que son aquellas que corresponden a los procesadores vecinos y que están acopladas con los tipos de variables locales anteriores.

Las ecuaciones locales pueden representarse como se muestra en la figura 4.6. La matriz de la figura 4.6 puede verse como una versión ordenada de las ecuaciones asociadas con una numeración local de pares ecuaciones/incógnitas.

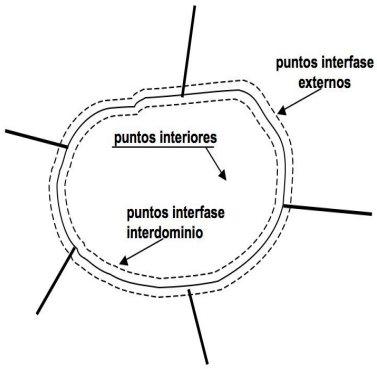

Figura 4.5: Vista local de una matriz dispersa distribuida

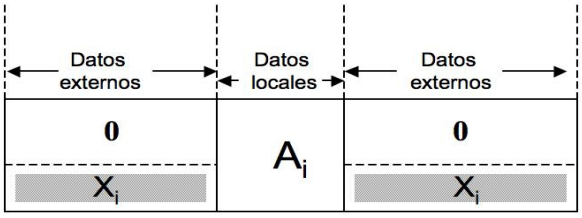

Figura 4.6: Matrices locales A_i y X_i.

Las filas de la matriz asignada a cierto procesador han sido divididas en dos partes: una matriz local A_i, que actúa solamente sobre las variables locales y la matriz interfase X_i, que actúa en las variables de interfase externa. Estas variables de interfase externa deben ser primero recibidas de los procesadores vecinos antes de que un producto matriz–vector distribuido pueda ser completado. Así, cada vector local de incógnitas $x_i (i = 1, \ldots, p)$ se divide también en dos partes: el sub–vector u_i de variables interiores seguidas por el sub–vector y_i de variables interfase inter–dominio. El lado derecho b_i se divide así mismo en en los sub–vectores f_i y g_i,

$$x_i = \begin{pmatrix} u_i \\ y_i \end{pmatrix} \; ; \; b_i = \begin{pmatrix} f_i \\ g_i \end{pmatrix}. \tag{4.1}$$

La matriz local A_i residente en el procesador i se organiza a bloques de acuerdo a esta partición, dando origen a

$$A_i = \left(\begin{array}{c|c} B_i & F_i \\ \hline E_i & C_i \end{array}\right).$$

Con esto, las ecuaciones asignadas al procesador i puede escribirse como sigue:

$$\begin{pmatrix} B_i & F_i \\ E_i & C_i \end{pmatrix} \begin{pmatrix} u_i \\ y_i \end{pmatrix} + \begin{pmatrix} 0 \\ \Sigma_{j \in N_i} E_{ij} y_j \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix}. \tag{4.2}$$

El término $E_{ij}y_j$ es la contribución de las ecuaciones locales de el sub–dominio vecino j, y N_i es el conjunto de sub–dominios que son vecinos del sub–dominio i. La suma de estas contribuciones, que están en la parte izquierda de (4.2), es el resultado de multiplicar la matriz de interfase X_i por las incógnitas de interfase externas

$$\sum_{j \in N_i} E_{ij} y_j \equiv X_i y_{i,ext}.$$

El resultado de esta multiplicación afecta únicamente a las incógnitas de interfase local, la cual está indicada por el cero en la parte superior del segundo término del lado izquierdo de (4.2).

Una vez que se tiene esta partición, puede aplicarse cualquiera de los precondicionadores de pARMS que se explican en las siguientes subsecciones.

4.3.4.2. Precondicionamiento ARMS Paralelo

Las técnicas del complemento de Schur multinivel de pARMS están basadas en técnicas que explotan los conjuntos independientes a bloques, como los descritos para los precondicionadores de ARMS.

La versión secuencial de ARMS tiene su base en la aplicación de técnicas tipo ILU multinivel (MILU) que explotan conjuntos *independientes*. Un conjunto independiente es un conjunto de incógnitas que no están acopladas con otras. Tales técnicas transforman el sistema original a un sistema de la forma

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{4.3}$$

donde B es una matriz diagonal. El concepto de conjuntos independientes se ha generalizado a *bloques* o *grupos* [97]. Un conjunto independiente a grupos es una colección

de subconjuntos (bloques) de incógnitas, tales que no existe acople entre incógnitas de cualesquiera dos grupos (o bloques) diferentes. Las incógnitas dentro del mismo grupo (o bloque) pueden estar acopladas. Cuando las incógnitas de los conjuntos independientes a grupos son etiquetados primero, la matriz A tendrá la estructura a bloques de la expresión (4.3), en el que el bloque B es diagonal a bloques en vez de diagonal.

En el caso del método ARMS, la siguiente factorización a bloques se calcula de manera 'aproximada'

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} \approx \begin{pmatrix} L & 0 \\ EU^{-1} & I \end{pmatrix} \times \begin{pmatrix} U & L^{-1} \\ 0 & A_1 \end{pmatrix}, \tag{4.4}$$

donde $LU \approx B$, y $A_1 \approx C - (EU^{-1})(L^{-1}F)$.

En general, el algoritmo de factorización en el método ARMS consiste esencialmente de dos pasos: primero, obtener un conjunto independiente a grupos y reordenar la matriz en la forma (4.3); segundo, obtener una factorización ILU del bloque B, $B \approx LU$, y aproximaciones a las matrices $L^{-1}F$, EU^{-1} y A_1. El proceso se repite recursivamente sobre la matriz A_1, hasta un cierto número de niveles. En el último nivel, puede aplicarse una factorización tipo ILUT o un método directo. La figura 4.7 esquematiza el procedimiento ARMS en el que las áreas más oscuras representan los complementos de Schur que van formándose consecutivamente.

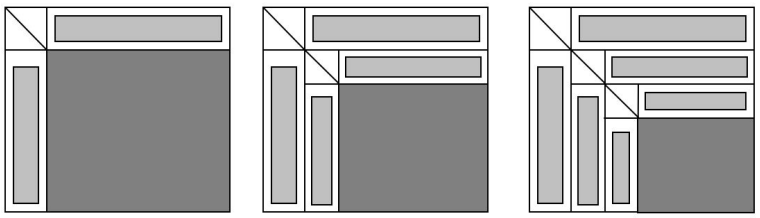

Figura 4.7: Un paso del procedimiento en la librería pARMS para formar complementos de Schur consecutivos

Las matrices A_1 del complemento de Schur consecutivas permanecen dispersas pero van tornándose densas conforme el número de niveles se incrementa, de modo que los elementos pequeños son eliminados durante las factorización a bloques a fin de mantener la estructura dispersa. Por otro lado, las matrices EU^{-1} y L^{-1} (o sus aproximaciones) no necesitan guardarse, dado que se utilizan para calcular la aproximación a A_1. Una vez hecho esto, se descartan. Las operaciones siguientes con $L^{-1}F$ y EU^{-1} se realizan usando U, L y los bloques E y F.

Distintos tipos de estrategias multinivel recursivas pueden definirse. Esencialmente, un paso de precondicionamiento equivale a una solución aproximada con la factorización

(4.3). Tal solución consiste de tres pasos. El primer paso es resolver con la primera matriz del lado derecho de (4.4). Esta operación producirá un nuevo lado derecho para la segunda parte de (4.3), que es $g_1 = g - EU^{-1}f$. Después, el sistema de complemento de Schur resultante $A_1 y = g_1$ se resuelve *iterativamente*. Por último, una sustitución hacia atrás es realizada para obtener la variable $u = U^{-1}[f - L^{-1}Fy]$. Dado que no se especifica la forma en la que el sistema complemento de Schur $A_1 y = g_1$ será resuelto, existen muchas posibles estrategias. Por ejemplo, la recursividad puede aplicarse: si está disponible otro nivel de factorización, el proceso puede repetirse y descender al próximo nivel, resolver (recursivamente), y ascender de regreso al nivel actual. Cuando se alcance el último nivel, el sistema puede resolverse mediante iteraciones del método GMRES precondicionado con ILUT o resolverse con los factores ILUT.

A continuación, se esquematiza la aplicación de un procedimiento ARMS como solver local en la construcción de un precondicionador de complemento de Schur global. Por sencillez, se considera la aplicación de un solo nivel del procedimiento ARMS actuando localmente en cada subdominio de la partición pARMS. En la figura 4.8, las líneas continuas establecen los límites de los subdominios. Los conjuntos locales independientes a grupos son construídos solamente sobre aquellas incógnitas desacopladas de las incógnitas externas mediante la producción de separadores locales (líneas con guiones y puntos en la figura 4.8). Las incógnitas de interfase interdominio (líneas con guiones en la figura 4.8) están asignadas *a priori* a los complementos de Schur por el procedimiento ARMS. Así en cada subdominio, la matriz del complemento de Schur local A_1, producida en el primer nivel de la reducción de ARMS, actuará sobre las variables de interfase interdominio más las variables separador, llamadas variables de interfase local. En otras palabras, las variables del complemento de Schur $y_i, i = 1, \ldots, p$ (ver expresión (4.7)), son las uniones de las variables de interfase interdominio y las variables separadoras los conjuntos independientes a grupos. Se denotará por *complemento expandido de Schur* el sistema que involucra la matriz A_1 que actúa sobre las incógnitas de interfase local e interdominio. Una vez resuelto el sistema global para todas las variables y, las variables interiores son obtenidas sin comunicación mediante sustitución hacia atrás en ARMS dentro de cada procesador.

4.3.4.3. Precondicionamiento Aditivo de Schwarz

Cuando un procedimiento aditivo de Schwarz se usa como precondicionador, el vector residuo local es actualizado por un solver iterativo global. El vector residual local resultante es usado como lado derecho local. Para esto, un intercambio de datos debe preceder la actualización del vector residual local, de modo tal que cada subdominio reciba de sus vecinos los últimos valores de sus variables de interfase interdominio. Cuando los últimos valores de las variables de interfase externas y el lado derecho local están disponibles,

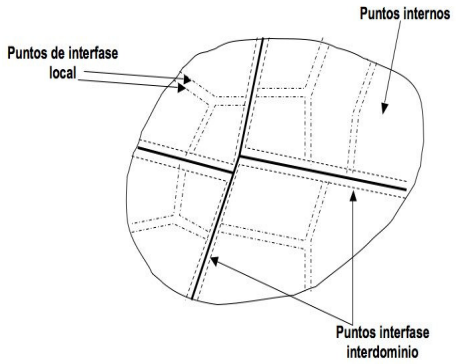

Figura 4.8: Clasificación de las incógnitas cuando se usa el método ARMS como solver local en el contexto de pARMS

el procedimiento aditivo de Schwarz puede usarse para encontrar una corrección a la solución local. El algoritmo 15 muestra el procedimiento descrito.

Algoritmo 15 Precondicionamiento aditivo de Schwarz.

01 Obtener datos externos $y_{i,ext}$
02 Actualizar residuo local$r_i = (b - Ax)_i = b_i - A_i x_i - X_i y_{i,ext}$
03 Resolver $A_i \delta_i = r_i$
04 Intercambiar δ_i entre los subdominios vecinos
05 Actualizar la solución $x_i = x_i + \delta_i$

El algoritmo 15 se ejecuta en cada procesador de manera paralela. El intercambio de información toma lugar en la línea 1 y el residuo se actualiza en la línea 2. Hay que notar que el residual es "actualizado" sólo en la parte y del lado derecho que cambia. Por otro lado, los sistemas locales $A_i \delta_i = r_i$ pueden resolverse de tres maneras: (1) mediante un método directo (disperso), (2) mediante un método de Krylov o (3) mediante una resolución con los factores triangulares resultantes de una factorización tipo ILU.

4.3.4.4. Precondicionamiento Complemento de Schur

Las técnicas del complemento de Schur se refieren a métodos que iteran únicamente sobre las incógnitas de interfase interdominio, implícitamente usando incógnitas interiores como variables intermedias. Estas técnicas forman la base para la aplicación de precondicionadores que serán explicados más adelante. Los sistemas del complemento de Schur se derivan mediante la eliminación de las variables u_i de la expresión en (4.2). Extrayendo de la primera ecuación $u_i = B_i^{-1}(F_i y_i)$, y sustituyendo en la segunda ecuación

se obtiene que

$$S_i y_i + \sum_{j \in N_i} E_{ij} y_j = g_i - E_i B_i^{-1} f_i \equiv g_i', \tag{4.5}$$

donde S_i es el complemento de Schur "local"

$$S_i = C_i - E_i B_i^{-1} F_i. \tag{4.6}$$

La ecuación (4.5) para todos los subdominios $i (i = 1, \ldots, p)$ constituye un sistema global de ecuaciones que involucra solamente los vectores y_i de incógnitas de interfase interdominio. Este sistema reducido global mantiene una estructura a bloques natural relacionada con los puntos de interfase interdominio para cada subdominio:

$$\begin{pmatrix} S_1 & E_{12} & \cdots & E_{1p} \\ E_{21} & S_2 & \cdots & E_{2p} \\ \vdots & & \ddots & \vdots \\ E_{p1} & E_{p-1,2} & \cdots & S_p \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix} = \begin{pmatrix} g_1' \\ g_2' \\ \vdots \\ g_p' \end{pmatrix}. \tag{4.7}$$

Los bloques diagonales en este sistema, las matrices $S_i (i = 1, \ldots, p)$, son densas en general. Los bloques E_{ij} fuera de la diagonal, que son idénticos con los involucados en (4.2) son dispersos.

El sistema (4.7) puede escribirse como $Sy = g'$, donde el vector y consiste de todas las variables de interfase interdominio $y_1, y_2, \ldots, y_p$ apiladas en un sólo vector. La matriz S es la matriz complemento de Schur "global". Los desarrolladores de pARMS consideran que pueden utilizarse métodos de resolución aproximados (como [29]) para el sistema reducido (4.7) para desarrollar precondicionadores del sistema distribuido (global) original. Una vez que el sistema complemento de Schur global (4.7) es (aproximadamente) resuelto, cada procesador calculará la parte u del vector solución en (4.1), mediante la resolución del sistema $B_i u_i = f_i - E_i y_i$ obtenido por substitución de (4.2). Se puede resumir la iteración del complemento de Schur mediante el algoritmo 16:

Algoritmo 16 Iteración del complemento de Schur.

01 *Adelante:* calcular lados derechos locales $g_i' = g_i - E_i B_i f_i$.
02 *Resolver:* el sistema del complemento de Schur global $Sy = g'$.
03 *Atrás:* substituir para obtener u_i, es decir, resolver $B_i u_i = f_i - E_i y_i$.

La ecuación (4.5) puede reescribirse como un sistema precondicionado con bloques diagonales:

$$y_i + S_i^{-1} \sum_{j \in N_i} E_{ij} y_j = S_i^{-1} [g_i - E_i B_i^{-1} f_i]. \tag{4.8}$$

Lo anterior puede verse como una versión de precondicionamiento Jacobi a bloques del complemento de Schur del sistema (4.5). El sistema (4.8), que acopla incógnitas externas y locales, puede resolverse por un método como el GMRES, requiriendo por lo tanto una solución con S_i para cada paso.

4.3.4.5. Precondicionadores y Solvers en pARMS

La siguiente es una lista de precondicionadores disponibles en pARMS:

- Tipo Schwarz aditivo: `add_ilu0, add_ilut, add_iluk, add_arms`. `add_X` indica precondicionador aditivo de Schwarz con precondicionador local X, donde X puede ser: ILU0, ILUT o ARMS.

- Tipo complemento de Schur: `lsch_ilu0, lsch_ilut, lsch_iluk, lsch_arms, rsch_ilu0, rsch_ilut, rsch_iluk, rsch_arms`. `lsch_X` indica precondicionador del complemento de Schur izquierdo con precondicionador local X, y `rsch_X` indica precondicionador del complemento de Schur derecho con precondicionador local X. Los experimentos presentados en este trabajo de tesis utilizaron precondicionamiento por la izquierda.

Por otro lado, pARMS contiene los siguientes aceleradores.

1. FGMRES es una versión distribuida del método GMRES flexible [91], que permite iteraciones en la aplicación del precondicionador.

2. DGMRES es una versión distribuida del método GMRES deflactado, que usa deflación de los valores propios.

3. BCGSTAB es una versión distribuida del método del Gradiente Biconjugado Estabilizado.

La librería paralela de pARMS ha sido aplicada con éxito en aplicaciones modernas de la Física [109], es por esta razón que fué elegida para llevar a cabo experimentos con los sistemas dispersos de la ecuación de difusión neutrónica, caso estacionario.

4.3.5. PETSc

La librería paralela Portable, Extensible Toolkit for Scientific Computation (PETSc) es un conjunto de estructuras de datos y rutinas de libre distribución, que proporcionan las bases para la implementación de códigos de aplicación a gran escala en computadoras paralelas (y secuenciales). PETSc usa el estándar de comunicación entre procesos de MPI [101] [100] [99].

4.3.5.1. Organización de PETSc

Algunos de los módulos de PETSc tratan con:

- Conjuntos índices, incluyendo permutaciones, para indexamiento en vectores, renumeración, etc.
- Vectores.
- Matrices (por lo general dispersas).
- Arreglos distribuidos.
- Métodos del subespacio de Krylov.
- Precondicionadores, incluyendo solvers multimalla (multigrid) y solvers dispersos.
- Integradores por paso de tiempo (*timesteppers*) para resolver ecuaciones diferenciales parciales (EDP) no lineales dependientes del tiempo.

Cada uno de estos módulos consiste de una interfase abstracta y una o más implementaciones usando estructuras de datos particulares. Con esto, PETSc proporciona códigos efectivos y limpios para las varias fases en la solución de las EDP, con un enfoque uniforme para cada tipo de problema. Además, este diseño permite la comparación y uso de diferentes algoritmos (por ejemplo, la experimentación con diferentes métodos de subespacio de Krylov, precondicionadores o métodos de Newton truncados).

La librería PETSc busca la fácil adaptación y extensión tanto de algoritmos como de su implementación. Este enfoque intenta promover el reuso del código y separar los elementos de paralelismo de la elección del algoritmo. La infraestructura de la librería PETSc crea un fundamento para la construcción de aplicaciones a gran escala.

La Figure 4.9 ilustra la organización jerárquica de la librería PETSc, que permite a los usuarios emplear los niveles de abstracción más apropiados para un problema particular.

La librería PETSc usa la tecnología del estándar MPI [56] como interfaz de paso de mensajes para la programación en paralelo. No obstante esto, el usuario es libre de emplear las rutinas de MPI conforme las necesite en su código de aplicación. Sin embargo, la librería PETSc oculta al usuario muchos de los detalles del paso de mensajes, que permanecen dentro de los objetos paralelos, tales como los vectores, matrices y solvers. Anexo a todo esto, la librería proporciona herramientas tales como arreglos distribuidos y operaciones de reunión/dispersión (gather/scatter) en vectores como ayuda en el manejo de datos en paralelo.

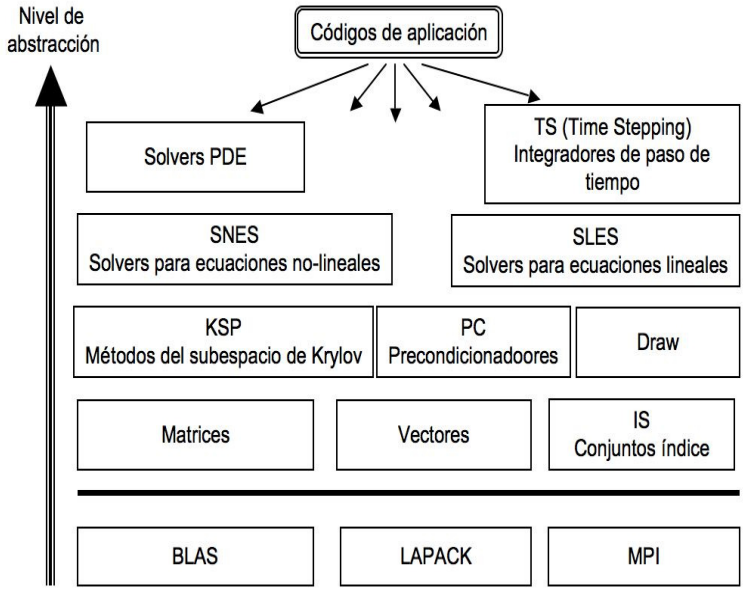

Figura 4.9: Organización de la librería PETSc.

4.3.5.2. Objetos Vector, Matriz y KSP en PETSc

PETSc construye vectores secuenciales o paralelos, `x`, de dimensión global `M` con los siguientes comandos

```
VecCreate(MPI_Comm comm,Vec *x);
VecSetSizes(Vec x, int m, int M);
```

donde `comm` representa el comunicador MPI y `m` es el tamaño local opcional o puede ser un valor que decida PETSc (`PETSC_DECIDE`).

Otro de los objetos fundamentales que maneja PETSc son las matrices, para lo cual implementa una variedad de éstas. La librería PETSc soporta actualmente almacenamiento denso así como almacenamiento disperso comprimido por fila (CSR o AIJ), además de varios formatos especializados. Existen formatos matriciales dispersos tipo AIJ (o CSR) secuenciales y paralelos en PETSc. Las matrices dispersas paralelas con formato AIJ pueden crearse mediante el comando:

```
MatCreateMPIAIJ(MPI_Comm comm,int m, int n, int M,int N,
                int d_nz,int *d_nnz,int o_nz,int *o_nnz, Mat *A);
```

donde `A` es la matriz recién creada, en tanto los argumentos `m, M` y `N`, indican el número

de filas local, y el número global de filas y columnas, respectivamente. En el esquema de partición de PETSc, todas las columnas de la matriz son locales y `n` es el número de columnas correspondiente a la parte local del vector paralelo. Cualquiera de los parámetros local o global pueden reemplazarse con la constante `PETSC_DECIDE`, de modo que la librería las determine. La matriz se almacena con un número de filas en cada procesador, dado por `m`, o determinado por PETSc si `m` es `PETSC_DECIDE`. Si `PETSC_DECIDE` no se usa en lugar de los `m` y `n`, entonces el usuario debe asegurarse de elegirlos de modo que sean compatibles con los vectores. Para asegurar esto, debe considerarse primero el producto matrix–vector $y = Ax$. El parámetro `m` que se usa en la rutina `MatCreateMPIAIJ()` para la creación de la matriz debe corresponderse con el tamaño local usado en la rutina de creación del vector `VecCreateMPI()` para `y`. Así mismo, el valor de `n` usado debe corresponderse con el usado como tamaño local en la rutina `VecCreateMPI()` para `x`. Por ejemplo, el esquema de particionamiento con el formato AIJ para una operación Ax, debe hacerse como sigue

$$
\begin{array}{ll}
p_0 & \\
p_1 & Ax = \\
p_2 &
\end{array}
\left(\begin{array}{ccc|ccc|cc}
1 & 2 & 0 & 0 & 3 & 0 & 0 & 4 \\
0 & 5 & 6 & 7 & 0 & 0 & 8 & 0 \\
9 & 0 & 10 & 11 & 0 & 0 & 12 & 0 \\
\hline
13 & 0 & 14 & 15 & 16 & 17 & 0 & 0 \\
0 & 18 & 0 & 19 & 20 & 21 & 0 & 0 \\
0 & 0 & 0 & 22 & 23 & 0 & 24 & 0 \\
\hline
25 & 26 & 27 & 0 & 0 & 28 & 29 & 0 \\
30 & 0 & 0 & 31 & 32 & 33 & 0 & 34
\end{array}\right)
\left(\begin{array}{c}
1 \\ 0 \\ 5 \\ \hline 7 \\ 9 \\ 0 \\ \hline 10 \\ 11
\end{array}\right)
\begin{array}{l}
p_0 \\ \\ \\ p_1 \\ \\ \\ p_2 \\ \\
\end{array}
$$

donde las partes locales de la matriz A y el vector x almacenado en el procesador p_0 son:

$$
A_{p_0} = \left(\begin{array}{ccc|ccc|cc}
1 & 2 & 0 & 0 & 3 & 0 & 0 & 4 \\
0 & 5 & 6 & 7 & 0 & 0 & 8 & 0 \\
9 & 0 & 10 & 11 & 0 & 0 & 12 & 0
\end{array}\right) \text{ y } \left(\begin{array}{c} 1 \\ 0 \\ 5 \end{array}\right) = x_{p_0},
$$

respectivamente.

En la librería PETSc, el usuario debe especificar un comunicador para la creación de cualquier objeto (ya sea un vector, una matriz o solver) para indicar el conjunto de procesadores sobre los cuales el objeto será distribuido.

La tabla 4.1 muestra algunas operaciones con objetos matrices y vectores en PETSc.

Después de crear las matrices y vectores que definen un sistema lineal, $Ax = b$, el usuario puede entonces usar un objeto tipo KSP para resolver el sistema con la siguiente secuencia de comandos:

```
KSPCreate(MPI_Comm comm,KSP *ksp);
```

Tabla 4.1: Operaciones de matriz en PETSc

Nombre de la función	**Operación**
MatAXPY(Mat Y, PetscScalar a, Mat X, MatStructure);	$Y = Y + a * X$
MatMult(Mat A, Vec x, Vec y);	$y = A * x$
MatMultAdd(Mat A, Vec x, Vec y, Vec z);	$z = y + A * x$
MatMultTranspose(Mat A, Vec x, Vec y);	$y = A^T * x$
MatMultTransposeAdd(Mat A, Vec x, Vec y, Vec z);	$z = y + A^T * x$
MatScale(Mat A, PetscScalar a);	$A = a * A$
MatCopy(Mat A, Mat B, MatStructure);	B=A

```
KSPSetOperators(KSP ksp, Mat A, Mat precA, MatStructure flag);
KSPSetFromOptions(KSP ksp);
KSPSetFromOptions(KSP ksp, Vec b, Vec x);
KSPDestroy(KSP ksp).
```

El usuario primero crea un contexto KSP y establece los operadores asociados con el sistema, es decir, la matriz del sistema lineal y opcionalmente una matriz de precondicionamiento diferente. Después, establece las opciones para resolver el sistema lineal, y finalmente destruye el contexto KSP. Por otro lado, el comando `KSPSetFromOptions()`, permite al usuario personalizar el método de solución durante la ejecución usando un conjunto de opciones de una base de datos. Esta base de datos permite seleccionar el método de solución y el precondicionador, así como la tolerancia de convergencia, entre otras cosas.

La figura 4.10 muestra más a detalle algunos de los objetos contenidos en PETSc, como los métodos de Krylov, precondicionadores, etc.

Entre los métodos iterativos de Krylov más populares contenidos en PETSc están el Gradiente Conjugado (GC), Gradiente Conjugado Cuadrado (BCS), Biconjugado estabilizado, Residuo Mínimo Generalizado (GMRES), entre otros. Por otro lado, PETSc ofrece precondicionadores tales como Schwarz aditivo, Jacobi a bloques, Jacobi, LU incompleto, Cholesky, etc. Las versiones de la librería PETSc que se han usado para los experimentos numéricos son 2.2.0 y 2.3.3.

4.3.6. Hypre

La librería numérica Hypre[7] (del Inglés High Performance Preconditioners) [110] [111] [30], también de libre distribución, es una conjunto de precondicionadores y solvers de alto desempeño dirigidos a la solución de grandes SELD sobre computadoras paralelas.

[7] https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html

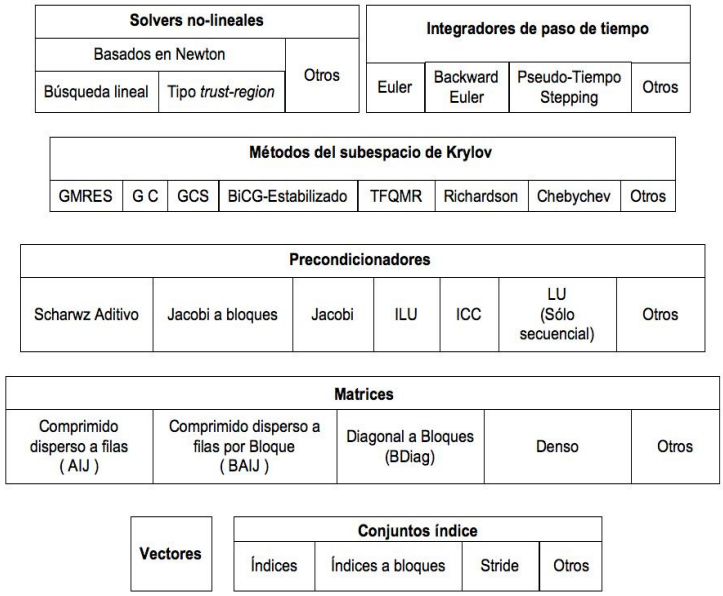

Figura 4.10: Componentes numéricos en la librería PETSc.

Entre otras cosas, la librería Hypre fue creada con el objetivo primordial de proporcionar a los usuarios precondicionadores paralelos avanzados. Aspectos como robustez, facilidad de uso, flexibilidad y e interoperabilidad también han sido tomados en cuenta durante su diseño. La librería se carateriza por contener solvers multimalla paralelos tanto para problemas estructurados como no estructurados.

4.3.6.1. Características Principales

Entre las características más destacadas de la librería Hypre se encuentran las siguientes:

- Contiene varias familias de algoritmos precondicionadores enfocados a la solución de grandes SELD. Hypre incluye, algoritmos que usan más que solo la matriz para resolver ciertas clases de problemas más eficientemente que las librerías de propósito general.
- Proporciona varios de los métodos iterativos más populares basados en subespacios de Krylov para ser usados con los precondicionadores escalables. Hypre incluye métodos para sistemas no simétricos tales como Residuo Mínimo Generalizado (GMRES) y métodos para matrices simétricas como el Gradiente Conjugado (GC).
- Busca mejorar la usabilidad de las primeras generaciones de librerías para resolver sistemas de ecuaciones lineales dispersos, de modo que el usuario no tenga que aprender

complicadas estructuras de datos para matrices dispersas. En lugar de ello, Hypre construye esas estructuras de datos para el usuario mediante una variedad de interfases conceptuales.

- Desarrollado en el lenguaje de programación C (con la excepción de la interfase FEI, la cual está escrita en el lenguaje C++), y usa Babel para proporcionar interfases para usuarios de Fortran 77, Fortran 90, C++, Phyton y Java.

4.3.6.2. Interfases Conceptuales

Los solvers y precondicionadores disponibles en Hypre pueden accederse desde el código de la aplicación mediante las interfases de sistemas lineales conceptuales de Hypre, las cuales permiten distintas descripciones de problemas de forma natural. Las interfases conceptuales intentan representar de modo natural, la forma en que los desarrolladores de la aplicación piensan acerca de su problema lineal y proporcionar interfases naturales a ellos para pasar los datos que definen a sus sistemas lineales en Hypre. De este modo, las interfases pueden ser de ayuda para que el usuario pueda construir las estructuras de datos apropiadas para los solvers y precondicionadores contenidos en Hypre. En la fila de la parte superior de la figura 4.11, se muestra un número de interfases conceptuales. Por lo general, las interfases conceptuales están representadas por distintos tipos de mallas computacionales, pero otro tipo de aplicación podría requerir de información de tipo geométrico. Por ejemplo, aplicaciones que usan mallas estructuradas (como las interfases de la izquierda en la figura 4.11), por lo general ven sus problemas en términos de esténcil y mallas. Por otro lado, las aplicaciones que usan mallas no estructuradas y elementos finitos normalmente ven sus problemas lineales en términos de elementos y matrices de elementos rígidos. Finalmente, la interfase del extremo derecho, es la forma estándar lineal–algebraica de ver el problema lineal.

La librería Hypre soporta actualmente cuatro interfases conceptuales, que se explican a continuación.

Sistema de malla estructurada (`Struct`): Esta interfase es apropiada para aplicaciones cuya malla consiste de redes de mallas lógicas rectangulares con un patrón de esténcil fijo de elementos diferentes de cero para cada punto de la malla.

Sistema de malla semi–estructurada (`SStruct`): Esta interfase es apropiada para aplicaciones cuyas mallas en su mayoría son estructuradas, pero con algunas características no estructuradas. Ejemplos incluyen mallas a bloques con estructura, mallas compuestas en aplicaciones con refinamiento de red adaptativo estructurado y mallas sobre–establecidas. Esta interfase soporta múltiples incógnitas por celda.

Interfase por elementos finitos (`FEI`): Apropiada para usuarios cuyos sistemas lineales parten de una discretización por elementos finitos. La interfase refleja estructuras

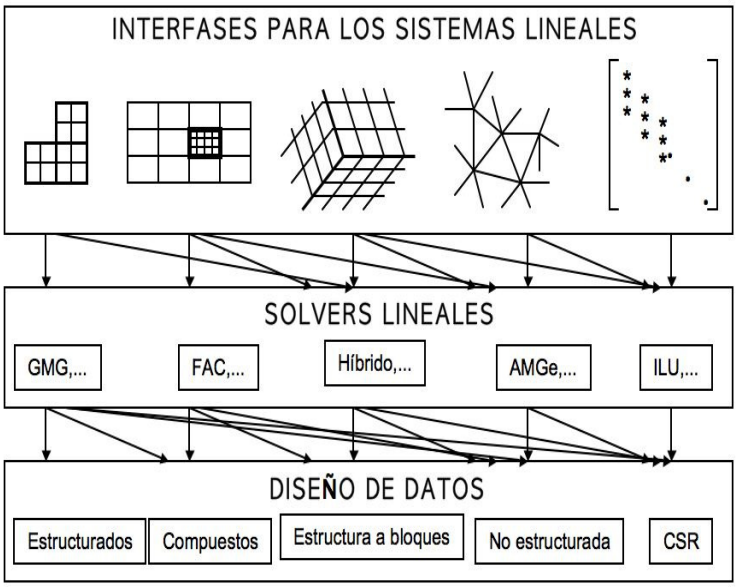

Figura 4.11: Interfases conceptuales en la librería Hypre.

de datos por elementos finitos típicos, incluyendo elementos de matrices rígidas.

Sistema lineal algebraico `(IJ)`: Esta es la interfase lineal algebraica común. Esta interfase requiere de más trabajo por parte del usuario.

4.3.6.3. La Interfase `IJ`

En este trabajo de tesis se ha utilizado la interfase IJ dado que las matrices de prueba están definidas en formato CSR. Bajo esta interfase, las matrices están distribuidas en bloques de filas conforme a lo siguiente:

$$\begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{P-1} \end{bmatrix}. \tag{4.9}$$

En la expresión (4.9), la matriz está distribuida a lo largo de los P procesos, $0, 1, \ldots, P-1$ por bloques de filas. Cada submatriz A_p is "propiedad" de un solo proceso y el número de su primera y última fila están dados por los índices globales `ilower` e `iupper` en la llamada de creación de la matriz.

4.3.6.4. Métodos y Precondicionadores en la Interfase `IJ`

Para la interfase conceptual IJ, Hypre proporciona métodos iterativos del subespacio de Krylov tales como Gradiente Conjugado (GC), Residuo Mínimo Generalizado (GMRES) y Gradiente Biconjugado Estabilizado (BICGSTAB). A continuación se presenta una breve descripción de los precondicionadores de Hypre usados en esta memoria.

- BoomerAMG. BoomerAMG es una implementación paralela del método multimalla algebraico [61] [86], y que puede ser usado como precondicionador o solver.
- ParaSails. ParaSails [19] [18] es una implementación paralela de un precondicionador inverso aproximado disperso, usando un patrón disperso *a priori* y una minimización (norma de Frobenius) por mínimos cuadrados. Los patrones dispersos son patrones de potencias de matrices dispersificadas. Además, ParaSails usa una técnica post–filtro para reducir los costos de aplicar el precondicionador.
- Euclid. La librería Euclid es una implementación escalable del algoritmo ILU paralelo presentado en publicado en forma extendida en la revista sobre Computación Científica del SIAM [70] y presentado por primera vez en [69].
- DS. Precondicionador por escalado diagonal.

4.4. Conclusiones

Existe una enorme cantidad de herramientas de software de libre distribución para la realización de cómputo numérico, tanto secuencial como paralelo. Este capítulo ha presentado los fundamentos y características de funcionamiento de algunas librerías que implementan métodos iterativos y directos. Concretamente se han presentado los fundamentos de la librería paralela SuperLU, la cual implementa un método directo. Así mismo, se han presentado librerías numéricas como SPARSKIT, PETSc, pARMS e Hypre que implementan métodos iterativos y precondicionadores basados en distintas técnicas. En esta tesis, se han utilizado los recursos de dichas librerías, para implementar y resolver de manera eficiente los sistemas de ecuaciones dispersos relacionados con la ecuación de difusión neutrónica multigrupo 3D. En el siguiente capítulo se muestran los resultados experimentales obtenidos.

Capítulo 5

Resultados Numéricos

5.1. Introducción

Este capítulo se ha dividido en dos partes. Una primera parte está representada por la sección 5.2, la cual presenta, analiza y compara los resultados numéricos obtenidos de la aplicación de diversas librerías numéricas de libre distribución, tanto secuenciales como paralelas (SPARSKIT, PETSc, SuperLU, pARMS) para resolver los grandes sistemas de ecuaciones lineales dispersos (SELD) relacionados con la ecuación de los modos Lambda (caso estacionario de la ecuación de difusión neutrónica (EDN)) para un caso de estudio real correspondiente al reactor de Ringhals–I (secciones 5.2.1 y 5.2.2). Dentro de los resultados obtenidos se muestran, por un lado, los tiempos de ejecución secuenciales con las distintas librerías numéricas, así como la determinación del mejor tiempo secuencial (sección 5.2.3). Una vez hecho lo anterior, se muestran los tiempos de ejecución obtenidos con las distintas librerías paralelas y se determinan los índices de aceleración (*speedup*) y eficiencia paralela, alcanzando con este análisis uno de los objetivos principales que persigue la presente tesis, el cual es determinar aquella librería, tanto secuencial como paralela que reuna las mejores características de facilidad de uso y desempeño paralelo para el caso de estudio presentado (sección 5.2.4). Se enuncian también algunas conclusiones obtenidas acerca de los resultados numéricos para el caso estacionario (sección 5.2.5).

La segunda parte de este capítulo está representada por la sección 5.3, que muestra los resultados numéricos obtenidos de la aplicación de dos algoritmos iterativos multipaso para resolver los SELD que surgen de la discretización en el tiempo de la EDN multigrupo. Además, se presentan los resultados numéricos experimentales obtenidos con una serie de modificaciones que el presente trabajo de tesis propone a los algoritmos iterativos multipaso, y que buscan acelerar el proceso de convergencia de dichos

algoritmos (sección 5.3.1). El caso de prueba elegido es el reactor de Leibstadt, y las matrices corresponden a una prueba de estabilidad realizada en el año de 1990 (sección 5.3.2). Los algoritmos iterativos multipaso, así como los cambios propuestos, han sido programados en el ambiente de desarrollo de Matlab (sección 5.3.3) y en las librerías numéricas paralelas de PETSc e Hypre (secciones 5.3.4 y 5.3.5). También se presentan los tiempos de ejecución secuenciales y paralelos obtenidos con los algoritmos iterativos multipaso, así como una comparativa de aceleración y eficiencia paralela entre ambas librerías. Por último en la sección 5.3.6 se enuncian algunas conclusiones relacionadas con los experimentos numéricos del caso dinámico.

Las plataformas secuenciales y paralelas utilizadas para los experimentos han sido Kubrick, Kefren y Jacquard (ver capítulo 4).

5.2. Solución de los Sistemas de Ecuaciones Lineales Dispersos de la Ecuación de Difusión Neutrónica. Caso Estacionario

El uso de herramientas de la computación de altas prestaciones tales como librerías numéricas paralelas y clusters de computadoras trae como consecuencia acelerar los procesos de cálculo en muchos problemas de ingeniería. En particular, en esta sección se han estudiado y aplicado las librerías numéricas de SPARSKIT [91] [90] [88], PETSc [101] [100] [99], pARMS [79] [95] [78] [96] y SuperLU [21] [125] a la resolución de los SELD relacionados con la ecuación de los modos Lambda.

5.2.1. Estrategias y Métodos

Un primer problema a resolver en el estudio de simulación de los reactores nucleares es la determinación de la distribución del flujo neutrónico en estado estacionario, así como los modos subcríticos responsables de las inestabilidades producidas en los mismos. Para esto, existen varios métodos de discretización y entre los más ampliamente usados, se encuentran los métodos de colocación nodal [116] [60].

En este trabajo de tesis, el método de colocación nodal utilizado es una adaptación de los métodos de colocación clásicos para la discretización de ecuaciones diferenciales parciales. Este método asume una expansión de los flujos neutrónicos en términos de polinomios de Legendre ortogonales. Como se ha establecido en el capítulo 2, tras imponer las condiciones de continuidad y contorno apropiadas a la ecuación de los modos Lambda

$$\mathcal{L}\Phi = \lambda \mathcal{M}\Phi,$$

y después de utilizar el método de discretización nodal, se obtiene un problema de valores propios generalizado, es decir, se tiene el problema de encontrar los valores propios y vectores propios de:

$$L\psi_n = \frac{1}{k_n} M\psi_n, \tag{5.1}$$

donde L y M son matrices de dimensión $2N$ para dos grupos de energía, con la estructura a bloques que ya se ha comentado en el capítulo 2. Este problema de valores propios generalizado puede expresarse como:

$$\begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \psi_{1_n} \\ \psi_{2_n} \end{bmatrix} = \frac{1}{k_n} \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_{1_n} \\ \psi_{2_n}. \end{bmatrix} \tag{5.2}$$

Independientemente de las condiciones de continuidad del flujo impuestas entre las celdas durante el proceso de discretización del núcleo, las matrices L_{11} y L_{22} siempre son simétricas en estructura y definidas positivas.

Por las condiciones de continuidad del flujo impuestas entre las celdas durante el proceso de discretización del núcleo, se sabe que las matrices L_{11} y L_{22} del caso de estudio abordado, son simétricas definidas positivas, y siempre son simétricas en estructura, además de ser dispersas y de gran dimensión.

Uno de los enfoques utilizados para resolver el problema de valores propios generalizado representado en la expresión (5.2), es reducirlo a un problema de valores propios ordinario de dimensión N. Lo anterior se consigue mediante las siguientes acciones:

1. Partiendo de la ecuación (5.2) se deducen las dos siguientes:

$$L_{11}\psi_{1_n} = \frac{1}{k_n}(M_{11}\psi_{1_n} + M_{12}\psi_{2_n}), \tag{5.3}$$

$$-L_{21}\psi_{1_n} + L_{22}\psi_{2_n} = 0. \tag{5.4}$$

2. Despejando ψ_{2_n} en la ecuación (5.4) y sustituyendo su valor en la ecuación (5.3), se obtiene la siguiente expresión

$$A\psi_{1_n} = k_n\psi_{1_n}, \tag{5.5}$$

donde la matriz A viene dada por

$$A = L_{11}^{-1}(M_{11} + M_{12}L_{22}^{-1}L_{21}). \tag{5.6}$$

Esta estrategia ha sido utilizada con éxito en técnicas basadas en la Iteración del Subespacio [120] [119], Arnoldi con Reinicio Implícito [62] [63] [108], métodos basados en Homotopía [65] y en métodos de Jacobi–Davidson [115].

La transformación anterior requiere del cálculo explícito de las inversas de los bloques L_{11} y L_{22}, lo cual numéricamente no es aconsejable por el relleno y los errores de redondeo

que se producen. Esto trae como consecuencia que no se disponga de la matriz A en forma explícita, por lo que para llevar a cabo un producto matriz–vector con la matriz A se requerirá de tres productos matriz diagonal por vector, la resolución de dos grandes SELD (con L_{11} y L_{22}) y la suma de dos vectores.

Algunas aplicaciones sólo requieren del cálculo de los valores propios dominantes, es decir, aquellos valores propios de A con mayor magnitud, así como sus correspondientes vectores propios. Lo anterior motiva la siguiente definición: dada una matriz A de dimensión n se llamará *problema parcial de valores propios* asociado a esta matriz, a un problema de la forma

$$AX = X\Lambda \tag{5.7}$$

donde Λ es una matriz diagonal de dimensión p con $p \leq n$ cuyos elementos son los valores propios dominantes de A, y X es una matriz de dimensión $n \times p$, cuyas columnas son los vectores propios asociados a los p valores propios dominantes de A [121].

Ahora bien, para resolver el problema ordinario de valores propios representado en la ecuación (5.5) se pueden usar distintos algoritmos dependiendo de si la matriz A es dispersa o densa. En nuestro caso la matriz es dispersa, por lo que se pueden usar algoritmos que no modifican su estructura original tales como el método de la potencia [50], iteración del subespacio [27] o basados en Subespacios de Krylov (como el método de Lanczos [75], Lanczos simétrico, Arnoldi [7], procedimiento Rayleigh–Ritz [89]). Recientemente se han incorportado mejoras algorítmicas a los métodos de proyección, dando origen a técnicas como Jacobi–Davidson [115] [106] o Arnoldi con Reinicio Implícito [108] [62] [63].

En esta memoria se ha elegido como base el algoritmo de Arnoldi por su sencillez y se han tomado elementos del análisis que se hace en [121], con el objetivo de identificar una de las partes críticas de este trabajo. Además, el método elegido representa una mayor estabilidad y efectividad para el cálculo de gran número de valores propios respecto de otros métodos.

5.2.1.1. Método de Arnoldi

Uno de los métodos más eficientes para el cálculo de valores propios y vectores propios de matrices de gran dimensión, no simétricas y dispersas, y que se apoya en la teoría de los subespacios de Krylov es el método de Arnoldi [7] [89] [50]. La idea del proceso de Arnoldi fue introducida en 1951, como medio para reducir una matriz densa a la forma de Hessenberg superior.

El método de Arnoldi es un algoritmo para contruir una base ortogonal del subespacio de Krylov, en el cual dada una matriz $A \in \Re^{n \times n}$ y un vector $x \in \Re^n$, $x \neq 0$, se define el espacio de Krylov de orden m, $\mathcal{K}_m$, como el espacio generado por los vectores obtenidos al ir multiplicando las distintas potencias de A por el vector x:

$$\mathcal{K}_m(A, x) = span\{x, Ax, A^2x, \ldots, A^{m-1}x\} \; . \tag{5.8}$$

Una característica interesante de los métodos de proyecciones ortogonales de Krylov es que pueden formularse en términos del polinomio característico del problema. Este polinomio se define para el caso de proyecciones ortogonales como el de la matriz $X^H AX$, donde X es la matriz cuyos vectores columna forman una base ortonormal de $\mathcal{K}_m(A, x)$ y X^H su matriz adjunta o transpuesta [50] [68].

En el capítulo 3 se presentó una formalización del método de Arnoldi y que se reproduce nuevamente a continuación.

Algoritmo 17 Arnoldi básico

01 Elegir un vector unitario inicial v_1
02 Elegir la dimensión del subespacio de krylov m
03 Para $j = 1, 2, \ldots, m$
04 $\quad h_{i,j} = v_i^T A v_j, \quad i = 1, 2, \ldots, j$
05 $\quad w_j = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$
06 $\quad h_{j+1,j} = \| \, w_j \, \|_2$, Si $h_{j+1,j} = 0$ TERMINAR.
07 $\quad v_{j+1} = w_j / h_{j+1,j}$

Normalmente, el valor de m es bastante menor que la dimensión n de la matriz A, por lo que si sólo se ejecuta una iteración del algoritmo, los valores propios de la matriz de Hessenberg H no son buenas aproximaciones de los valores propios dominantes de la matriz A, a menos que se elija una dimensión del subespacio de Krylov, m, muy grande. También, si se toman valores de m cercanos a n, el coste del algoritmo es prohibitivo.

Retomando la forma en la que está representada la matriz A del problema tratado en esta tesis (ver expresión (5.6)), la inversión de los bloques L_{11} y L_{22} es prohibitiva por el gran coste computacional y de almacenamiento que esto implica. Por tal razón, el cálculo del producto matriz por vector Av_j presente en el bucle del algoritmo de Arnoldi, debe realizarse con los pasos representados por el algoritmo 18.

Algoritmo 18 Cálculo del producto Av_j

01 $w_1 = L_{21} v_j$
02 $w_2 = L_{22}^{-1} w_1$
03 $w_3 = M_{12} w_2$
04 $w_4 = M_{11} v_j$
05 $r = L_{11}^{-1}(w_3 + w_4)$

Bajo la forma del algoritmo 18 se evita construir explícitamente la matriz A y se hace

uso de la resolución de dos SELD (líneas 2 y 5), tres multiplicaciones matriz diagonal–vector (líneas 1, 3, 4) y una suma de vectores (línea 5).

Del algoritmo 17, se observa que el coste computacional mayor está representado por el producto matriz–vector Av_j. Por otro lado, en el algoritmo 18, las operaciones críticas están representadas por los pasos 2 y 5, por lo que es fundamental realizar estas operaciones con el mínimo costo posible.

En esta tesis se ha realizado un estudio del empleo de diversas librerías del álgebra lineal numérica para matrices dispersas y de gran dimensión, que permita seleccionar los métodos que resuelvan más eficientemente los sistemas de ecuaciones de las líneas **2** y **5**, lo que permitirá mejorar la eficiencia del algoritmo de Arnoldi (o de cualquier otro método de cálculo de valores donde sea necesario realizar una multiplicación matriz–vector) e incidir, por lo tanto, en la eficiencia de la resolución de la ecuación de los modos Lambda.

5.2.2. Caso de Estudio: Reactor Ringhals–I

El caso de estudio elegido en esta sección es el reactor de la central nuclear sueca de Ringhals–I, que es del tipo agua en ebullición o BWR (*Boiling Water Reactor*). Este reactor se ha discretizado de forma tridimensional en 27 planos axiales de 14.72 cm. de longitud, 25 correspondientes al combustible, un plano superior y otro inferior correspondientes al reflector. Cada plano axial se ha discretizado en celdas de 15.275×15.275 cm. distribuidas, como se observa en la figura 5.1.

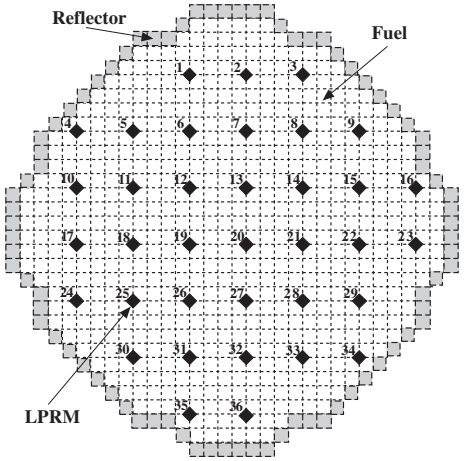

Figura 5.1: Plano axial del reactor Ringhals–I

Para esto se ha utilizando un método de colocación nodal basado en desarrollos de polinomios de Legendre grado $K = 2$. Esta discretización ha permitido dividir el reactor

en 20,844 nodos.

Utilizando un esquema de numeración natural para la malla de discretización, el patrón de elementos no nulos resultante es de tipo banda para los bloques L_{ii} $(i = 1, 2)$, como se aprecia en la figura 5.2.

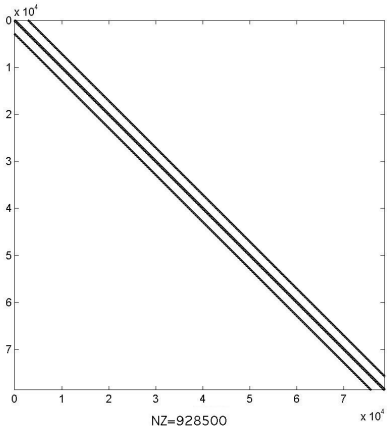

Figura 5.2: Patrón de dispersión del bloque L_{11} del caso Ringhals–I

Cuando se analizan métodos de resolución puede ser de gran ayuda contar con información acerca de las propiedades principales de una matriz dispersa, la cual en muchas ocasiones permite elegir entre los distintos formatos de almacenamiento a fin de hacer más eficiente el acceso a sus elementos o la realización de operaciones matriz–vector. Entre las propiedades más sencillas de calcular se encuentran: la dimensión de la matriz, el número total de elementos distintos de cero, el número promedio de elementos diferentes de cero por fila (*desviación estándar*), el ancho de banda, entre otras.

Muchas de las propiedades de una matriz que influyen en la elección del formato de almacenamiento para matrices dispersas se muestran en la tabla 5.1. Dicha tabla se ha obtenido con la rutina `info1.f` de SPARSKIT [88] [90] y muestra información de los bloques dispersos L_{ii} $(i = 1, 2)$ para el caso del reactor Ringhals–I.

En la tabla 5.1, la propiedad *Diagonales no evitables* (DNE) indica el número de diagonales que contiene al menos un elemento distinto de cero mientras que la propiedad *% diagonal dominante* indica el porcentaje de columnas diagonal dominantes.

Como se ha mencionado antes, se sabe que independientemente del orden K del desarrollo de Legendre y del benchmark utilizado, la matriz es simétrica con estructura en banda. Esta idea podría llevar a pensar que el formato en banda (`BND`)puede ser una buena alternativa al elegir el formato de almacenamiento. Pero un análisis más detenido indica que no sería el más conveniente, dado que bajo este formato, se necesitaría de

Tabla 5.1: Propiedades de los bloques L_{ii} del reactor Ringhals–I

GEOMETRÍA	**PROPIEDAD**	L_{11}	L_{22}
3D Con polinomio de Legendre Grado 2	Dimensión – n	83376	83376
	Elementos no nulos – nnz	928500	928500
	Ancho de banda – BW	6179	6179
	Diagonales no evitables – DNE	83	83
	Diagonal dominante (%)	100 %	100 %
	Grado de simetría	1.00	1.00
	Índice de dispersión	0.0001	0.0001

un arreglo de tamaño $BW \times n$, lo cual representa un número superior respecto de los elementos distintos de cero (nnz) que se necesitan almacenar. Otra alternativa es el almacenamiento diagonal (DIA). Bajo este formato, se utiliza un arreglo rectangular de tamaño $DNE \times n$ para almacenar los valores de las diagonales de la matriz, además de un arreglo (OFF) de enteros con los desplazamientos de las diagonales respectivas. Para este tipo de almacenamiento, nuevamente se observa que el espacio requerido supera en mucho el número de elementos no nulos presentes en la matriz. Otro método de almacenamiento es el CSR. Bajo el formato CSR por ejemplo, se manejan dos arreglos de enteros (IA, JA) y un arreglo de números reales que contendrá los valores distintos de cero de la matriz (AA). El arreglo JA de números enteros contendrá los índices columna de los elementos de la matriz. En tanto que el otro arreglo de números enteros IA, contendrá apuntadores a los inicios de cada fila, en el arreglo JA.

En este trabajo de tesis, se ha elegido el formato CSR debido a las razones expuestas con anterioridad, además de que la mayoría de las librerías numéricas que se utilizaron lo contemplan como formato de entrada para muchas de sus rutinas.

5.2.3. Estudio Secuencial del Comportamiento de las Librerías

Se sabe que por la forma de la discretización y las condiciones de contorno y flujo neutrónico impuestas sobre el reactor, las matrices L_{11} y L_{22} siempre son simétricas en estructura y definidas positivas. Estas propiedades garantizan su resolución mediante métodos directos o iterativos (ver capítulo 3).

Se sabe también que la resolución de SELD mediante métodos directos puede conducir a un relleno (*fill-in*) de la matriz que puede originar problemas con su almacenamiento en la memoria del computador y a errores de redondeo. Sin embargo, se han incorporado nuevas mejoras a las técnicas de métodos directos implementados en algunas librerías numéricas [54] [124] [125] [57], por lo que es de nuestro interés determinar el desempeño obtenido para nuestro caso de estudio usando uno de estos métodos. También se presenta

un análisis de desempeño de la aplicación de distintos métodos iterativos implementados en algunas librerías numéricas (ver capítulo 4).

Las rutinas y tiempos mostrados en este apartado fueron registrados con las librerías numéricas de software libre de SPARSKIT [88] [90], PETSc [100] [101] [99], pARMS [96] [78] [79] [95] y SuperLU [54] [124] [125] [21]. Algunos de los estudios descritos aquí se describen en [32] [34].

Para el caso de las librerías SPARSKIT y PETSc, los métodos del subespacio de Krylov probados fueron los contenidos en la tabla 5.2.

Tabla 5.2: Métodos iterativos utilizados en las librerías SPARSKIT y PETSc

MÉTODO	**IDENTIFICADOR**
Gradiente Conjugado	GC
Gradiente Biconjugado	BCG
Gradiente Biconjugado Estabilizado.	BCGSTAB
Residuo Quasi-Mínimo Libre Transpuesto	TFQMR
Residuo Mínimo Generalizado	GMRES

Para el caso de la librería numérica de pARMS, los métodos del subespacio de Krylov usados se muestran en la tabla 5.3.

Tabla 5.3: Métodos en la librería pARMS

MÉTODO	**IDENTIFICADOR**
GMRES flexible	FGMRES
GMRES deflactado	DGMRES
Gradiente biconjugado estabilizado	BCGSTAB

Por otro lado, la aplicación de precondicionadores a los métodos iterativos suelen mejorar la velocidad de convergencia a la solución. La tabla 5.4 muestra los precondicionadores de SPARSKIT usados para los experimentos; mientras que en la tabla 5.5 se muestran los precondicionadores para el caso de la librería de PETSc. Para el caso de la librería pARMS, los precondicionadores usados se muestran en la tabla 5.6.

Los cálculos numéricos de todas las pruebas secuenciales se han realizado en un procesador del cluster Kefren de la Universidad Politécnica de Valencia (ver sección 4.2.2), utilizando aritmética de punto flotante con doble precisión. Además, para los experimentos se ha elegido un test de convergencia basado en la l_2-norma del residuo, por lo que la convergencia se detecta en la iteración k–ésima si se cumple que

$$\| r_k \|_2 < \epsilon * \| b \|_2,$$

Tabla 5.4: Precondicionadores utilizados en la librería SPARSKIT

PRECONDICIONADOR	**IDENTIFICADOR**
Jacobi	JAC
LU incompleto (sin relleno)	ILU0
LU incompleto (con truncamiento)	ILUT
LU incompleto (modificado)	MILU
LU incompleto (multinivel)	ILUK

Tabla 5.5: Precondicionadores utilizados en la librería PETSc

PRECONDICIONADOR	**IDENTIFICADOR**
LU incompleto (sin relleno)	ILU0
Jacobi	JAC
Jacobi a Bloques	JACB
Mtodo Aditivo de Schwarz	ASM

Tabla 5.6: Métodos y precondicionadores usados en la librería pARMS

PRECONDICIONADOR	**IDENTIFICADOR**
Aditivo de Scharwz (con ILU0)	ADD_ILU0
Aditivo de Scharwz (con ILUT)	ADD_ILUT
Aditivo de Scharwz (con ILUK)	ADD_ILUK
Aditivo de Scharwz (con ARMS)	ADD_ARMS
Complemento de Schur (con ILU0)	LSCH_ILU0
Complemento de Schur (con ILUT)	LSCH_ILUT
Complemento de Schur (con ILUK)	LSCH_ILUK
Complemento de Schur (con ARMS)	LSCH_ARMS

donde $r_k = b - Ax_k$. La tolerancia exigida para los distintos métodos fué de $\epsilon = 10^{-12}$, dado que la librería numérica de SuperLU obtuvo resultados con dicha precisión, y es sólo con fines comparativos, pues para casos prácticos suelen usarse tolerancias alrededor de 10^{-6}. Por otro lado, el número máximo de iteraciones (*maxits*) se ha establecido a 10000 para todos los métodos y se ha tomado como base del subespacio de Krylov un valor m igual a 30 para los métodos basados en GMRES, que es el valor por default que usa PETSc. Aunque no se ha realizado un estudio del valor óptimo m del subespacio de Krylov para las matrices del caso de prueba, no se descarta este estudio como trabajo futuro.

Los tiempos de ejecución se han registrado con distintas rutinas, ya que en muchas ocasiones la misma librería numérica contaba con una función para tal efecto. Tal es el caso de PETSc y pARMS donde las funciones para el registro del tiempo fueron realizadas con las rutinas `PetscGetTime` y `dwalltime` respectivamente. En tanto que en el caso de SPARSKIT se ha utilizado la función `DTIME`. Los tiempos de las tablas se presentan en segundos (*segs*) a menos que se especifique lo contrario.

Se ha tomado como vector de entrada v_j, en el algoritmo 18, un vector aleatorio y las porciones de código del algoritmo 18 que se han medido en el tiempo corresponden únicamente a las partes donde tiene lugar la resolución de los SELD. En las tablas, las columnas T_a, T_b, T_t e ITS representan el tiempo de ejecución para resolver los sistemas L_{11}, L_{22}, el tiempo de ejecución total ($T_t = T_a + T_b$) y el número de iteraciones necesarias para obtener una solución aproximada.

Teniendo en cuenta lo anterior, se muestran a continuación los resultados obtenidos de aplicar al caso de estudio (la resolución de los sistemas de de las líneas 2 y 5 del algoritmo 18) los métodos seleccionados de las librerías ya mencionadas.

5.2.3.1. Análisis en la Librería SPARSKIT

Con el objeto de contrastar las ganancias obtenidas por la aplicación de los precondicionadores, primero se presentan y analizan los tiempos de ejecución sin precondicionador, y posteriormente con cada uno de los precondicionadores de la librería. Para el caso de SPARSKIT, que es una librería numérica completamente secuencial, se han obtenido los tiempos de ejecución de la tabla 5.7.

Tabla 5.7: Tiempos de ejecución (segs) de los métodos en la librería SPARSKIT sin precondicionar

MÉTODO	L_{22}		L_{11}		T_t
	ITS	T_a	ITS	T_b	
GC	59	0.58	121	1.20	**1.78**
BCG	116	1.16	240	2.44	3.60
GMRES	61	2.21	133	4.62	6.83
BCGSTAB	83	0.80	155	1.52	2.32
TFQMR	65	0.74	149	1.70	2.44

De la tabla 5.7 se puede observar que todos los métodos han convergido a una solución como se esperaba, dado que ambos bloques comparten la característica de ser diagonal dominante. No obstante lo anterior, el número de iteraciones necesarias para alcanzar una solución aproximada con el bloque L_{11} requiere de casi el doble de las iteraciones con las que se encuentra una solución con el bloque L_{22}, lo cual, en principio, indica que el bloque L_{22} presenta mejores propiedades espectrales que el bloque L_{11}. Por último,

de los métodos iterativos probados, el mejor tiempo está representado por el Gradiente Conjugado (GC) para ambos bloques de matrices, como era de esperar, dado que es un método apropiado para las características de las matrices. Al método del GC le siguen en eficiencia los métodos BCGSTAB y TFQMR. Por otro lado, el método menos eficiente está representado por el Residuo Mínimo Generalizado (GMRES) y, aunque no es el más apropiado para las matrices de ejemplo por sus características, se incluye con el objetivo de hacer más completo el estudio propuesto.

Por lo comentado en el capítulo 4, se sabe que la combinación de un precondicionador con un método basado en subespacios de Krylov puede ayudar a acelerar la tasa de convergencia a la solución. Uno de los precondicionadores que implica un coste bajo de construcción dado que se calcula con la diagonal principal de la matriz de coeficientes, lo representa el precondicionador de Jacobi (JAC) [91]. Los tiempos registrados por SPARSKIT con el uso de este precondicionador se muestran en la tabla 5.8.

Tabla 5.8: Tiempos de ejecución (segs) de los métodos en la librería SPARSKIT combinados con el precondicionador JAC

MÉTODO	L_{22}		L_{11}		T_t
	ITS	T_a	ITS	T_b	
GC	36	**0.45**	65	**0.80**	**1.25**
BCG	70	0.87	128	1.59	2.46
GMRES	43	1.41	82	2.48	2.42
BCGSTAB	49	0.60	95	1.15	1.75
TFQMR	39	0.53	75	1.01	1.54

Como se observa, se ha obtenido una disminución en el tiempo de convergencia respecto de los obtenidos sin el uso de precondicionador. A pesar de la aplicación del precondicionador, el método GC se mantiene a la cabeza en la solución de ambos sistemas de ecuaciones lineales registrando un tiempo total de 1.25 segundos, lo que significa una disminución en el tiempo de ejecución respecto al caso sin precondicionar (1.78 segs.) de un 30 %. Para este caso, nuevamente BCGSTAB y TFQMR son los métodos más eficientes después del GC; en tanto que GMRES Y BCG registran peores tiempos.

El empleo de un precondicionador tipo ILU modificado (MILU) ha dado lugar a los resultados mostrados en la tabla 5.9.

El efecto de este precondicionador puede notarse en la ligera disminución del tiempo para el bloque L_{22} respecto del precondicionador JAC de la tabla 5.8. No ocurre el mismo efecto para el caso de la solución con el bloque L_{11}, pues cuando el mejor método para este bloque en el caso del precondicionador JAC era el GC, la aplicación del precondicionador MILU lo ha empeorado, dando paso ahora como mejor método a TFQMR

Tabla 5.9: Tiempos de ejecución (segs) de los métodos en la librería SPARSKIT combinados con el precondicionador MILU

MÉTODO	L_{22}		L_{11}		T_t
	ITS	T_a	ITS	T_b	
GC	15	**0.43**	35	0.95	1.38
BCG	28	0.73	50	1.32	2.05
GMRES	15	0.64	26	1.27	1.91
BCGSTAB	17	0.48	33	0.87	1.35
TFQMR	17	0.51	29	**0.83**	1.34

seguido por BCGSTAB. A pesar de lo anterior, el tiempo de solución para el resto de los métodos se ve reducido respecto de los tiempos con el precondicionador JAC. También se observa que el número de iteraciones disminuye a más de la mitad en algunos casos, lo que no necesariamente se ha traducido en una disminución significativa del tiempo, dado que el costo de construcción y solución al operar con el precondicionador es distinto.

Al emplear un precondicionador tipo LU incompleto sin relleno (ILU0), se obtiene un ligero aumento en la rapidez de convergencia de manera diferente para cada solución con los bloques L_{ii} respecto del precondicionador MILU, como se observa en la tabla 5.10.

Tabla 5.10: Tiempos de ejecución (segs) de los métodos en la librería SPARSKIT combinados con el precondicionador ILU0

MÉTODO	L_{22}		L_{11}		T_t
	ITS	T_a	ITS	T_b	
GC	14	**0.39**	64	1.62	2.01
BCG	26	0.64	46	1.10	1.74
GMRES	14	0.60	24	1.15	1.75
BCGSTAB	15	0.42	29	**0.75**	1.17
TFQMR	15	0.43	29	0.80	1.23

Esta ligera ganancia en velocidad se nota para casi todos los métodos, a excepción del caso de la solución con el bloque L_{11} usando el GC, pues cuando se usa con el precondicionador tipo ILU0, el costo por iteración es mucho mayor que el costo de usar el precondicionador de Jacobi o MILU. Es interesante notar también que los métodos GMRES, BCGSTAB y TFQMR se han beneficiado de este último precondicionador al ver reducido ligeramente sus tiempos de solución con respecto de los casos JAC y MILU.

En la utilización del precondicionador ILUT, se realizaron pruebas con distintos nive-

les de relleno (*lfil*) y tolerancias de umbral (*droptol*) obteniéndose los tiempos mostrados en la tabla 5.11, en donde para el caso del método GC se presenta un "estancamiento" durante el proceso iterativo, por lo que no fué posible aproximar una solución. Para este caso en particular, el método agota el número máximo de iteraciones permitidas, indicando con esto que la aplicación del precondicionador empeora el número de condición para ambos bloques de matrices en todos los niveles de relleno y tolerancias de umbral probadas. Para el resto de los métodos se converge a una solución, destacando

Tabla 5.11: Tiempos de ejecución (segs) de los métodos en la librería SPARSKIT combinados con el precondicionador ILUT. El símbolo † indica no convergencia.

ILUT	**MÉTODOS**	L_{22}		L_{11}		T_t
(lfil,droptol)		ITS	T_a	ITS	T_b	
(1, 0.0)	GC	†	–	†	–	–
	BCG	66	1.04	148	2.23	3.27
	GMRES	34	1.38	74	2.83	4.21
	BCGSTAB	37	**0.62**	79	**1.22**	**1.84**
	TFQMR	37	0.69	89	1.51	2.20
(1, 10^{-1})	GC	†	–	†	–	–
	BCG	66	0.98	148	2.23	3.21
	GMRES	34	1.37	74	2.85	4.22
	BCGSTAB	37	**0.58**	79	**1.14**	**1.72**
	TFQMR	37	0.65	89	1.54	1.79
(1, 10^{-4})	GC	†	–	†	–	–
	BCG	66	1.08	148	2.23	3.31
	GMRES	34	1.42	74	2.84	4.26
	BCGSTAB	37	**0.63**	79	**1.21**	**1.84**
	TFQMR	37	0.70	89	1.51	2.21
(2, 0.0)	GC	†	–	†	–	–
	BCG	68	1.41	156	2.89	4.30
	GMRES	34	1.67	74	3.19	4.86
	BCGSTAB	35	**0.86**	79	**1.58**	**2.44**
	TFQMR	37	0.96	83	1.79	2.75
(4, 0.0)	GC	†	–	†	–	–
	BCG	48	14.08	138	15.91	29.99
	GMRES	24	14.00	68	16.24	30.24
	BCGSTAB	25	**13.46**	69	**14.31**	**27.77**
	TFQMR	27	13.65	79	14.70	28.35

en velocidad el método BCGSTAB para ambos bloques de matrices en todos los casos. Es interesante observar que cuando se usan valores de relleno altos, como en el caso de ILUT(4,0.0), el tiempo de aproximación se dispara debido a que el tiempo de construcción del precondicionador consume al menos el 90 % del tiempo total de cálculo para ambos bloques de matrices, porcentaje que se obtuvo en las pruebas experimentales. En general, el desempeño de ILUT respecto de los precondicionadores JAC, MILU e ILU0 no ha sido significativo, llegando incluso en algunos casos a ser peor, como se ha comentado anteriormente.

De las tiempos de ejecución con el precondicionador ILUK no se lograron reportar pruebas para niveles de relleno (*lfil*) iguales o mayores a 1 debido, entre otras cosas, a la modificación del patrón disperso del precondicionador durante su construcción y por problemas de una administración eficiente de la memoria.

Haciendo un resumen de todas las pruebas realizadas en SPARSKIT y eligiendo únicamente aquellos métodos que presentan los mejores tiempos de ejecución, tanto para el bloque L_{22} como L_{11}, así como los correspondientes precondicionadores, se obtiene la tabla 5.12, donde se resalta el renglón que registró el mejor tiempo, que en este caso resultó ser el método GC con precondicionador ILU0 (1.14 segundos).

Tabla 5.12: Resumen de mejores tiempos (segs) en la librería SPARSKIT

PC	**MÉTODO** en L_{22}	T_a	**MÉTODO** en L_{11}	T_b	T_t
No PC	GC	0.58	GC	1.20	1.78
JAC	GC	0.45	GC	0.80	1.25
MILU	GC	0.43	TFQMR	0.83	1.26
ILU0	**GC**	**0.39**	**BCGSTAB**	**0.75**	**1.14**
ILUT $(1, 10^{-2})$	BCGSTAB	0.58	BCGSTAB	1.14	1.72

5.2.3.2. Análisis en la Librería PETSc

Para el caso de la librería PETSc se ha realizado un estudio similar al de SPARSKIT, por lo que también se presentan los tiempos sin precondicionado y posteriormente los tiempos con el uso de algunos precondicionadores.

La tabla 5.13 muestra los tiempos de los métodos de Krylov probados en PETSc, donde se observa que el método del GC presenta el tiempo de solución menor para cada uno de los bloques y con respecto a todos los métodos probados de PETSc. Le siguen en desempeño los métodos de BCGSTAB y TFQMR. Por otro lado, los métodos GMRES y BCG presentan tiempos más altos.

Tabla 5.13: Tiempos de ejecución (segs) de los métodos en la librería PETSc sin precondicionador

MÉTODOS	L_{22} T_a	L_{11} T_b	T_t
GC	**0.64**	**1.37**	**2.01**
BCG	1.25	2.69	3.94
GMRES	1.35	3.08	4.42
BCGSTAB	0.99	1.68	2.67
TFQMR	0.87	2.04	2.91

Al aplicar un precondicionador de Jacobi en la librería PETSc se obtienen los tiempos de ejecución de la tabla 5.14, donde se observa el efecto del precondicionamiento, ya que se ha obtenido una reducción del tiempo de ejecución en todos los métodos, destacando en velocidad el método del GC, seguido por los métodos BCGSTAB y TFQMR. La combinación del método GC con este precondicionador (1.42 segundos) logra reducir en un 30 % el tiempo del GC sin precondicionar (2.01 segundos). Por otro lado, el GMRES y BCG continuan presentando los tiempos más altos.

Tabla 5.14: Tiempos de ejecución (segs) de los métodos en la librería PETSc combinados con el precondicionador JAC

MÉTODOS	L_{22} T_a	L_{11} T_b	T_t
GC	**0.50**	**0.92**	**1.42**
BCG	0.95	1.76	2.71
GMRES	0.96	1.84	2.80
BCGSTAB	0.66	1.22	1.88
TFQMR	0.61	1.22	1.84

La librería de PETSc cuenta también con un precondicionador tipo LU incompleto (ILU) secuencial. La tabla 5.15 reune los tiempos de ejecución para este precondicionador, en donde se observa una reducción mayor en el tiempo respecto de los casos sin precondicionar y Jacobi. Si se elige como referencia el tiempo registrado por el método GC, la aplicación de un precondicionador ILU0 (1.17 segundos) logra reducir el tiempo del caso sin precondicionar (2.01 segundos) y del caso con el precondicionador de Jacobi (1.42 segundos) en un 42 % y un 18 %, respectivamente.

Tabla 5.15: Tiempos de ejecución (segs) de los métodos en la librería PETSc combinados con el precondicionador ILU0 .

MÉTODOS	L_{22} T_a	L_{11} T_b	T_t
GC	**0.41**	**0.76**	**1.17**
BCG	0.78	1.45	2.23
GMRES	0.94	1.78	2.72
BCGSTAB	0.48	0.90	1.38
TFQMR	0.48	0.97	1.45

También se han realizado pruebas con un precondicionador tipo Scharwz aditivo (ASM) obteniéndose los tiempos reportados en la tabla 5.16. Sin embargo, aunque el tiempo obtenido con el método del GC combinado con el precondicionador ASM (1.50 segundos) reduce en un 25 % al tiempo de un método GC sin precondicionar (2.01 segundos), este 25 % es menor que la reducción del 30 % obtenida con el precondicionador JAC y menor que la reducción del 42 % con el precondicionador ILU0. Que el método del GC combinado con el precondicionador ASM no presente reducciones tan eficientes como los registrados por los precondicionadores JAC e ILU0 se debe a que las matrices de prueba no surgen de un proceso de discretización basado en un método de descomposición de dominios, por lo que es muy probable que bajo un método de discretización de esta naturaleza pudiera dar mejores resultados o en última instancia, resultados parecidos a los mostrados por la aplicación de un precondicionador ILU0.

También se han realizado pruebas en la librería de PETSc con un precondicionador tipo ILUK, con diferentes niveles de relleno *lfil*, obteniéndose los tiempos de ejecución mostrados en la tabla 5.17, destacando como mejor método el GC. Sin embargo los tiempos reportados no mejoran a los obtenidos con los precondicionadores JAC, JACB y ASM, debido a los efectos de relleno en la construcción del precondicionador.

La tabla 5.18 presenta un resumen de los mejores tiempos para cada uno de los precondicionadores probados en la librería de PETSc, donde se observa que a lo largo de las distintas pruebas, el método de Krylov que mejor desempeño mostró ha sido el

Tabla 5.16: Tiempos de ejecución (segs) de los métodos en la librería PETSc combinados con el precondicionador ASM

MÉTODOS	L_{22} T_a	L_{11} T_b	T_t
GC	**0.54**	**0.96**	**1.50**
BCG	0.96	1.72	2.68
GMRES	1.49	1.54	3.03
BCGSTAB	0.61	1.15	1.76
TFQMR	0.66	1.18	1.85

Tabla 5.17: Tiempos de ejecución (segs) de los métodos en la librería PETSc combinados con el precondicionador ILUK

ILUK (lfil)	**MÉTODOS**	L_{22} T_a	L_{11} T_b	T_t
	GC	**1.09**	**1.40**	**2.49**
	BCG	1.50	2.08	3.58
(1)	GMRES	1.44	2.52	3.96
	BCGSTAB	1.20	1.44	2.64
	TFQMR	1.21	1.48	2.69
	GC	3.14	3.47	6.61
	BCG	3.75	4.40	8.15
(2)	GMRES	4.69	3.58	8.27
	BCGSTAB	3.14	3.67	6.81
	TFQMR	3.15	3.70	6.85
	GC	20.72	20.89	41.61
	BCG	21.64	22.43	44.07
(4)	GMRES	22.26	20.80	43.06
	BCGSTAB	20.45	20.72	41.17
	TFQMR	20.37	20.68	41.05

GC combinado con el precondicionador ILU0.

5.2.3.3. Análisis en la Librería pARMS

Se ha realizado un estudio secuencial con los precondicionadores de la librería pARMS. Como se ha comentado, (ver capítulo 4), la librería pARMS contiene técnicas de precondicionamiento basadas en el método aditivo de Scharwz y en técnicas del complemento

Tabla 5.18: Resumen de mejores tiempos (segs) en la librería PETSc

PC	**MÉTODO** en L_{22}	T_a	**MÉTODO** en L_{11}	T_b	T_1
No PC	GC	0.64	GC	1.37	2.01
JAC	GC	0.50	GC	0.92	1.42
ILU0	**GC**	**0.41**	**GC**	**0.76**	**1.17**
ASM	GC	0.54	GC	0.96	1.50
ILUK (1)	GC	1.09	GC	1.40	2.49

de Schur.

En la librería pARMS, el método aditivo de Scharwz se utiliza en combinación con precondicionadores locales tipo ILU0, ILUT, ILUK y ARMS y que están designados como ADD_ILU0, ADD_ILUT, ADD_ILUK y ADD_ARMS, respectivamente.

Tabla 5.19: Tiempos de ejecución (segs) de los métodos en la librería pARMS combinados con técnicas de precondicionamiento tipo aditivo de Scharwz. Las siglas M.I. significan memoria insuficiente

PC LOCAL	**MÉTODOS**	L_{22} T_a	L_{11} T_b	T_t
ILU0	**BCGSTAB**	**0.56**	**0.91**	**1.47**
	DGMRES	0.66	1.43	2.09
	FGMRES	0.57	1.24	1.81
ILUK (lfil=1,dtol=0.1)	BCGSTAB	–	–	M.I.
	DGMRES	–	–	M.I.
	FGMRES	–	–	M.I.
ILUT (lfil=1,dtol=0.1)	BCGSTAB	0.72	1.77	2.49
	DGMRES	1.52	3.30	4.82
	FGMRES	1.33	2.79	4.12
ARMS (lfil=1,dtol=0.1)	BCGSTAB	1.38	3.33	4.71
	DGMRES	1.70	3.61	5.31
	FGMRES	1.84	3.81	5.65

La tabla 5.19 muestra los tiempos registrados por estos precondicionadores al caso de estudio. Como se puede observar, el precondicionador más efectivo está representado por el precondicionador ILU0, el cual, como se ha visto en resultados experimentales con las librerías SPARSKIT y PETSc, es el que mejor desempeño ha proporcionado. Con seguridad, es por esta razón que registra mejores tiempos de solución cuando se compara con el desempeño de otros precondicionadores como ILUK, ILUT y ARMS, ya que: para

el caso del precondicionador ILUK ni siquiera se tiene memoria suficiente para llevar a cabo la resolución de los sistemas, mientras que el precondicionador ILUT emplea tiempo de ejeución extra para la construcción del precondicionador con los niveles de relleno (*lfil*) especificados; para el caso del precondicionador ARMS, éste resulta ser el más costoso de todos pues, en términos generales, el precondicionador ARMS invierte tiempo extra de ejecución tanto para la búsqueda de conjuntos independientes a bloques, así como para ejecutar los niveles de recursividad especificados.

En el caso particular de los experimentos con el precondicionador ARMS se ha utilizado un sólo nivel de recursividad, es decir, se ha utilizado una factorización tipo ILUT (que ya se ha visto que va peor con respecto a otros prercondicionadores), por lo que los tiempos de ejecución observados con este precondicionador se deben al uso de ILUT y a la búsqueda de conjuntos independientes a bloques.

Respecto de la aplicación de los métodos BCGSTAB, DGMRES y FGMRES, se observa que, para el caso de los precondicionadores tipo aditivo de Scharwz, el método que mejor desempeño muestra, en términos generales, es el método BCGSTAB.

La aplicación de técnicas basadas en el complemento de Schur a las matrices del caso de estudio dan origen a los tiempos mostrados en la tabla 5.20.

Tabla 5.20: Tiempos de ejecución (segs) de los métodos en la librería pARMS combinados con técnicas de precondicionamiento tipo Complemento de Schur

PC LOCAL	**MÉTODOS**	L_{22} T_a	L_{11} T_b	T_t
ILU0	BCGSTAB	1.74	**1.13**	2.91
	DGMRES	0.77	1.51	2.28
	FGMRES	**0.63**	1.50	2.13
ILUK (lfil=1,dtol=0.1)	BCGSTAB	–	–	M.I.
	DGMRES	–	–	M.I.
	FGMRES	–	–	M.I.
ILUT (lfil=1,dtol=0.1)	BCGSTAB	0.91	2.10	3.01
	DGMRES	1.68	3.63	5.31
	FGMRES	1.47	3.13	4.60
ARMS (lfil=1,dtol=0.1)	BCGSTAB	1.95	3.92	5.87
	DGMRES	2.55	4.93	7.48
	FGMRES	2.55	4.37	6.92

Una comparativa de los tiempos totales (T_t) registrados por el método aditivo de Scharwz (tabla 5.19) con los tiempos obtenidos por las técnicas del complemento de Schur de la tabla 5.20 permite observar que estos últimos son ligeramente más altos que

los primeros. Esta situación se debe principalmente al gasto de tiempo invertido por la búsqueda de conjuntos independientes a bloques, como fase inicial para todas las técnicas de complemento de Schur. Es interesante notar también que al combinar la técnica del complemento de Schur con el precondicionador ILU0, el método que menor tiempo registra en la resolución con el sistema L_{22} es el método GMRES flexible (FGMRES), siendo que en el caso aditivo de Scharwz el mejor fué el método BCGSTAB. El resto de los precondicionadores (ILUK, ILUT y ARMS) tienen un comportamiento similar al caso aditivo de Scharwz.

Una vez realizadas las pruebas secuenciales, la tabla 5.21 muestra un resumen de los mejores tiempos obtenidos con ambas técnicas de precondicionamiento en pARMS. Para nuestro caso de estudio, el precondicionador con mejor tiempo de solución está representado por el método aditivo de Scharwz con precondicionador ILU0.

Tabla 5.21: Resumen de mejores tiempos (segs) en la librería pARMS

PC	**MÉTODO** en L_{22}	T_a	**MÉTODO** en L_{11}	T_b	T_t
Scharwz (Ad.)+ILU0	**BCGSTAB**	**0.56**	**BCGSTAB**	**0.91**	**1.47**
Schur (Comp.)+ILU0	GMRES	0.63	BCGSTAB	1.13	1.76

5.2.3.4. Mejor Tiempo Secuencial

Una vez que se han presentado los tiempos secuenciales registrados por las mejores combinaciones de método y precondicionador de las librerías estudiadas, se puede establecer el mejor tiempo secuencial (T_s) que se tomará como referencia a la hora de calcular las métricas paralelas de speedup y eficiencia paralela.

La tabla 5.22 presenta los mejores tiempos secuenciales registrados por las librerías de SPARSKIT, pARMS y PETSc.

Tabla 5.22: Resumen de mejores tiempos (segs) en las librerías SPARSKIT, PETSc y pARMS

LIBRERÍA	**PC**	**MÉTODO** en L_{22}	T_a	**MÉTODO** en L_{11}	T_b	T_t
SPARSKIT	**ILU0**	**GC**	**0.39**	**BCGSTAB**	**0.75**	**1.14**
PETSc	JACB	GC	0.41	GC	0.76	1.17
pARMS	Scharwz+ILU0	BCGSTAB	0.56	BCGSTAB	0.91	1.47

La figura 5.3 grafica los tiempos de la tabla 5.22, donde se puede observar que el menor tiempo secuencial lo registra la librería de SPARSKIT, seguido por PETSc y

pARMS. Por lo tanto, se ha elegido como mejor tiempo secuencial para el caso de estudio T_s =1.14 segundos. Para el caso de la librería SuperLU, no fué posible determinar el tiempo secuencial debido a los requerimientos de memoria para llevar a cabo la factorización.

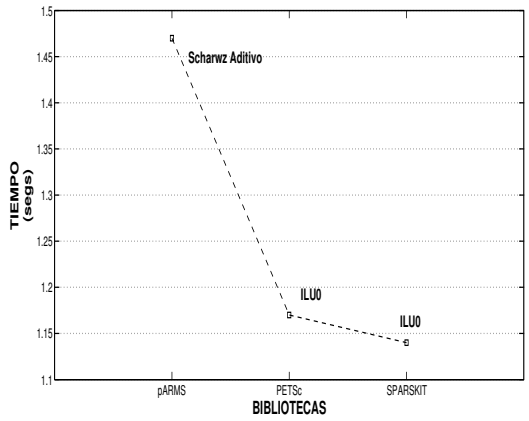

Figura 5.3: Gráfica de tiempos de ejecución en las librerías SPARSKIT, PETSc y pARMS

5.2.4. Estudio Paralelo

A continuación, se presentan los tiempos paralelos de ejecución obtenidos con las librerías de PETSc, pARMS y SuperLU aplicadas al caso de estudio. Se presentan también los tiempos obtenidos sin el uso de precondicionador y posteriormente usando los precondicionadores correspondientes. Los experimentos han sido realizados en el cluster Kefren de la Universidad Politécnica de Valencia (ver capítulo 4). En las tablas que contienen los resultados numéricos, la columna T_p representa la suma del tiempo de ejecución necesario para resolver los sistemas de ecuaciones con los bloques L_{11} y L_{22} usando p procesadores. Los tiempos han sido registrados con rutinas para ese propósito incluidas en las librerías y que han sido `PetscGetTime` y `dwalltime` en PETSc y pARMS, respectivamente. Al igual que en el análisis secuencial, los tiempos de las tablas se presentan en segundos (*segs*), a menos que se especifique lo contrario. Por otro lado, se han establecido las mismas condiciones de ejecución del caso secuencial en las pruebas paralelas aquí presentadas.

5.2.4.1. Análisis en la Librería PETSc

La tabla 5.23 reune los tiempos invertidos por los métodos implementados en la librería PETSc, sin el uso de precondicionadores, para diferentes números de procesadores p.

Tabla 5.23: Tiempos de ejecución (segs) en paralelo T_p de los métodos en la librería PETSc sin precondicionar

MÉTODOS	$p = 2$	$p = 4$	$p = 6$	$p = 8$	$p = 10$
GC	1.20	0.63	0.45	0.38	0.33
BCG	2.18	1.26	0.90	0.73	0.63
GMRES	2.26	1.26	0.84	0.63	0.55
BCGS	1.48	0.82	0.58	0.48	0.44
TFQMR	1.58	0.87	0.60	0.49	0.43

Las distintas pruebas experimentales muestran que las ejecuciones en paralelo del método Gradiente Conjugado aplicado para resolver los SELD L_{ii} $(i = 1, 2)$, mostró los mejores tiempos de ejecución en relación con el resto de los métodos del subespacio de Krylov.

Se puede observar también la reducción del tiempo secuencial por el uso de cómputo paralelo. Por ejemplo, si se contrasta el tiempo total (T_t) obtenido por el método del GC del caso secuencial sin precondicionar (2.01 segs de la tabla 5.13) y el tiempo total (T_p) obtenido por el método del GC usando p=10 procesadores (0.33 segs en la tabla 5.23), se obtiene una reducción de casi el 84 % del tiempo secuencial. En la figura 5.4, se muestra gráficamente la reducción de los tiempos de ejecución conforme aumenta el número de procesadores para el caso del método del GC sin precondicionar.

Los métodos de Krylov en PETSc se han combinado con la aplicación de un precondicionador tipo Jacobi (JAC), obteniéndose los tiempos mostrados en la tabla 5.24.

Tabla 5.24: Tiempos de ejecución (segs) en paralelo T_p de los métodos en la librería PETSc combinados con el precondicionador JAC

MÉTODOS	$p = 2$	$p = 4$	$p = 6$	$p = 8$	$p = 10$
GC	0.86	0.47	0.33	0.26	0.24
BICG	1.53	0.86	0.64	0.50	0.44
GMRES	1.47	0.80	0.55	0.42	0.36
BCGS	1.07	0.58	0.43	0.35	0.30
TFQMR	1.02	0.54	0.39	0.31	0.28

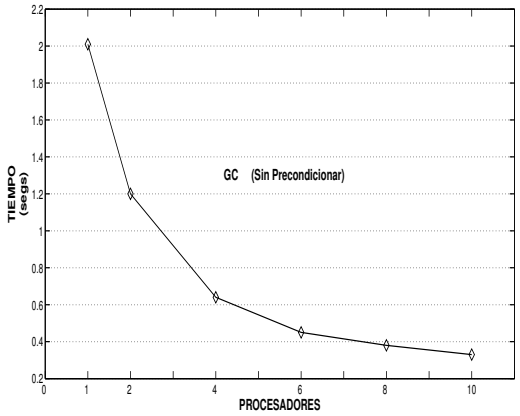

Figura 5.4: Gráfica de tiempos de ejecución en paralelo en la librería de PETSc usando el método GC

Comparando los tiempos obtenidos en pruebas paralelas con p=10 procesadores de las tablas 5.23 y 5.24 para el método del GC, el tiempo obtenido sin precondicionar de 0.33 segundos se reduce a 0.24 segundos con el precondicionador de Jacobi, lo que significa una reducción aproximada del 27 %. Esta ganancia se debe a la combinación de técnicas de paralelismo y precondicionado.

También se han hecho pruebas experimentales con la aplicación de un precondicionador Jacobi a bloques (JACB) para el caso de estudio, obteniéndose los tiempos mostrados en la tabla 5.25, donde se aprecia una mejoría, respecto de los tiempos con precondicionador Jacobi y, por consiguiente, sobre el caso sin precondicionar, como se verá para el método del GC que resultó el mejor de todos los métodos.

Tabla 5.25: Tiempos de ejecución (segs) en paralelo T_p de los métodos en la librería PETSc combinados con el precondicionador JACB

MÉTODOS	$p = 2$	$p = 4$	$p = 6$	$p = 8$	$p = 10$
GC	0.78	0.45	0.29	0.24	0.21
BICG	1.48	0.81	0.56	0.44	0.37
GMRES	1.34	0.64	0.39	0.30	0.28
BCGS	0.94	0.51	0.34	0.28	0.24
TFQMR	0.92	0.47	0.33	0.26	0.23

La aplicación del precondicionador Jacobi a Bloques logra reducir el tiempo del método GC sin precondicionar (0.33 segundos) a 0.21 segundos usando p=10 procesadores, lo que significa una reducción del tiempo del caso sin precondicionar de casi un 36 %.

Respecto del caso Jacobi (0.24 segundos), esta reducción de tiempo es del 13 %.

Continuando con la aplicación de los precondicionadores de PETSc propuestos, sólo resta mostrar los tiempos obtenidos con el precondicionador tipo Scharwz aditivo (PC ASM) aplicados al caso de estudio, obteniéndose los tiempos de la tabla 5.26.

Tabla 5.26: Tiempos de ejecución (segs) en paralelo T_p de los métodos en la librería PETSc combinados con el precondicionador ASM

MÉTODOS	$p = 2$	$p = 4$	$p = 6$	$p = 8$	$p = 10$
GC	0.92	0.57	0.42	0.36	0.37
BICG	1.54	0.96	0.71	0.60	0.54
GMRES	1.60	0.78	0.53	0.40	0.36
BCGS	1.01	0.62	0.47	0.39	0.36
TFQMR	1.05	0.63	0.49	0.40	0.36

En la tabla 5.26 se aprecia, entre otras cosas, que el método del Gradiente Conjugado es el que aporta los mejores tiempos de ejecución en general. Por otro lado, desde el punto de vista de la aplicación de los precondicionadores, el aporte en velocidad del precondicionador ASM no es muy significativo comparado con el precondicionador Jacobi (JAC) y Jacobi a bloques (JACB). Sin embargo, no se puede dejar de lado el efecto del uso de la computación de altas prestaciones, por ejemplo cuando se aplica el método del GC combinado con el precondicionador ASM en el caso secuencial se obtiene un tiempo de 1.50 segundos (ver tabla 5.16), en tanto que el tiempo al usar p=10 procesadores es de 0.37 segundos, lo que significa una reducción del 75 %.

A manera de resumen, la figura 5.5 muestra una gráfica comparativa de los mejores tiempos paralelos (T_p) del GC registrados en la aplicación de PETSc al caso de estudio.

Como puede observarse en la figura 5.5, la aplicación de precondicionadores puede reducir sustancialmente los tiempos de resolución de grandes SELD, como los abordados en este estudio. Por otro lado, puede apreciarse también la disminuación de los tiempos al usar computación de altas prestaciones. Como era de esperar y para el caso de PETSc, el método del Gradiente Conjugado ha sido el que mejores tiempos de ejecución ha reportado cuando se ha comparado con otros métodos de resolución, ya sea con o sin precondicionadores, ya sea con uno o muchos procesadores. En la gráfica se puede observar que el Gradiente Conjugado con precondicionador de Jacobi a bloques ha sido el más eficiente en la resolución de ambos sistemas de ecuaciones con bloques L_{11} y L_{22}.

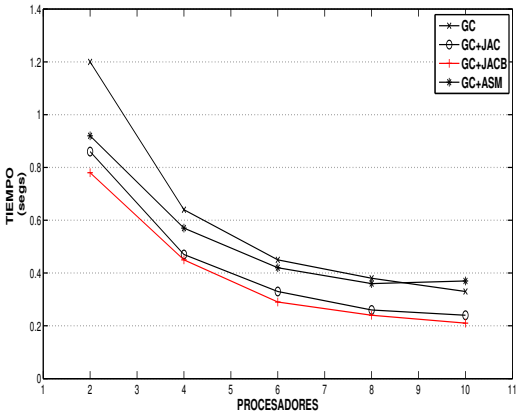

Figura 5.5: Gráfica de mejores tiempos en la librería de PETSc

5.2.4.2. Análisis en la Librería pARMS

Otra de las librerías utilizadas y aplicadas al caso del reactor Ringhals–I, descrita en el capítulo 4, es pARMS. Se han realizado experimentos paralelos en esta librería utilizando únicamente el método aditivo de Scharwz combinado con los precondicionadores locales ILU0, ILUT, pues como se ha visto en el análisis secuencial realizado anteriormente (ver apartado 5.2.3.3) son los que mejor desempeño ofrecen. La tabla 5.27, muestra los tiempos de ejecución invertidos para resolver los sistemas de ecuaciones L_{11} y L_{22}, que corresponden a pruebas paralelas con $p = 2, 4, 6, 8, 10$ procesadores.

Tabla 5.27: Tiempos de ejecución (segs) en paralelo T_p de los métodos en la librería pARMS

p	**PRECONDICIONADOR**	**ACELERADORES**		
		BCGSTAB	DGMRES	FGMRES
2	ADD_ILU0	1.52	1.95	1.48
	ADD_ILUT	1.94	2.37	2.04
4	ADD_ILU0	0.86	0.90	0.78
	ADD_ILUT	1.65	1.41	1.29
6	ADD_ILU0	0.81	0.84	0.55
	ADD_ILUT	1.40	1.23	0.84
8	ADD_ILU0	0.47	0.55	0.43
	ADD_ILUT	0.82	0.86	0.66
10	ADD_ILU0	0.44	0.51	0.36
	ADD_ILUT	0.70	0.75	0.54

Haciendo una comparativa de los tiempos registrados en la tabla 5.27 por la aplicación de los precondicionadores locales ILU0 e ILUT en paralelo, se observa que, en general, el método que registra los mejores tiempos está representado por la versión GMRES flexible (FGMRES). La gráfica de la figura 5.6 compara los tiempos de ejecución de la tabla 5.27, donde se aprecia la ganancia de tiempo obtenida por el precondicionador ILU0 sobre ILUT.

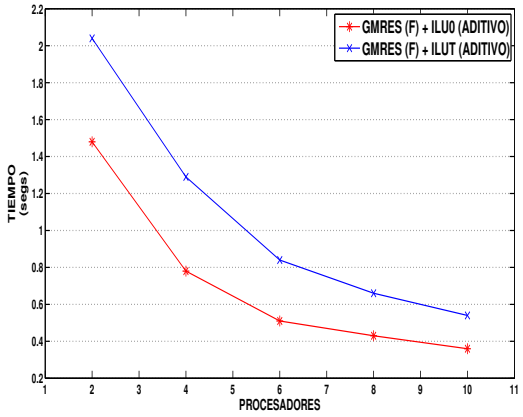

Figura 5.6: Gráfica de mejores tiempos en la librería de pARMS

5.2.4.3. Análisis en la Librería SuperLU

Como se ha descrito anteriormente (ver capítulo 4), se han realizado pruebas con un método directo de la librería SuperLU, obteniéndose los tiempos mostrados en la tabla 5.28. Una de las ventajas de esta técnica es la precisión de la solución encontrada, la cual ha sido de 12 dígitos en las soluciones al resolver los sistemas de ecuaciones lineales asociados a las matrices L_{11} y L_{22} del caso de estudio; sin embargo, las prestaciones obtenidas no son competitivas (ya que registra tiempos de ejecución en minutos), debido a que se precisa rapidez en la solución de los SELD, pues esta operación debe realizarse repetidas veces, como lo indica el algoritmo 17 de Arnoldi. Desde este punto de vista y para el caso de estudio, los métodos iterativos aquí utilizados, superan en mucho el desempeño ofrecido por la librería SuperLU.

5.2.4.4. Speedup y Eficiencia

Se ha podido comprobar experimentalmente que las librerías paralelas que mejor desempeño registran son PETSc y pARMS. En cambio, la librería numérica de SuperLU, no ofrece resultados competitivos para el caso de estudio.

Tabla 5.28: Tiempos de ejecución (mins) en paralelo T_p en la librería SuperLU

p	Precisión	T_p
2	3.62E-12	9.73
4	3.62E-12	5.00
6	3.64E-12	3.67
8	3.65E-12	2.95
10	3.64E-12	2.48

A continuación, y tomando en cuenta únicamente los mejores tiempos de PETSc y pARMS, se muestra el desempeño paralelo obtenido por ellos en la tabla 5.29, donde el mejor tiempo secuencial lo registró la librería numérica de SPARSKIT con un valor de $T_1 = T_s$=1.14 segundos (ver apartado 5.2.3).

Tabla 5.29: Coeficientes de speedup y eficiencia en las librerías PETSc y pARMS. Caso estacionario

p	PETSc			pARMS		
	T_p	S_p	$E_p(\%)$	T_p	S_p	$E_p(\%)$
1	**1.14**	1.00	100	**1.14**	1.00	100
2	0.78	1.46	73	1.48	0.77	39
4	0.45	2.53	63	0.78	1.46	37
6	0.29	3.93	66	0.55	2.07	35
8	0.24	4.75	59	0.43	2.65	30
10	0.21	5.43	54	0.36	3.17	29

La tabla 5.29 muestra que de las librerías PETSc y pARMS, la que ofrece los tiempos más competitivos para el caso de estudio es la librería de PETSc. La figura 5.7 ilustra mejor este hecho. En el caso de pARMS se ha encontrado que al menos para el caso de usar dos procesadores (p =2), el tiempo paralelo registrado (1.48 segundos) resulta ser muy costoso. Es probable que esta situación surga debido al gasto extra ocasionado por los reordenamientos de las submatrices que realiza pARMS para cada procesador en fases previas de aplicación de los precondicionadores.

Respecto de los coeficientes de speedup y eficiencia (ver figuras 5.8 y 5.9) obtenidos se observa que para la librería PETSc éstos son buenos cuando se utilizan hasta seis procesadores y aceptables cuando se usan ocho o diez procesadores. Por otro lado, y como se ha comentado anteriormente, en el caso de pARMS sus coeficientes de speedup y eficiencia caen debido a que, entre otras cosas, el mejor tiempo secuencial obtenido (en SPARSKIT) es muy eficiente.

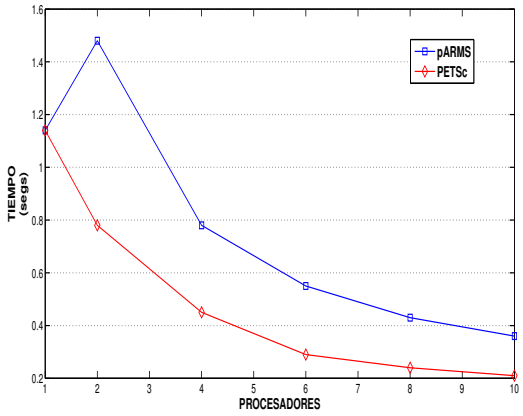

Figura 5.7: Gráfica de tiempos de ejecución en paralelo en las librerías PETSc y pARMS

5.2.5. Conclusiones del Caso Estacionario

Se han logrado identificar las herramientas software más adecuadas para resolver los SELD de gran dimensión (ecuaciones (5.3) y (5.4)) asociados a la EDN en su caso estacionario (ecuación de los modos Lambda (5.1)) para el caso de estudio.

Además, se ha visto la importancia que tiene el estudio, aplicación y evaluación de distintas librerías numéricas secuenciales y paralelas que pueden ayudar a mejorar el rendimiento en la resolución de problemas de Ingeniería donde se precisa de respuestas rápidas y eficientes.

En este trabajo se han plasmado resultados experimentales con algunas librerías numéricas (SPARSKIT, PETSc, pARMS y SuperLU); no obstante, es importante continuar con la investigación y prueba del desempeño ofrecido por otras librerías, pues continuamente surgen nuevos algoritmos que las implementan, de modo que los profesionales que no están relacionados con el uso de éstas, tengan los elementos necesarios para decidir cuál usar en función de trabajos de investigación como los que describe este trabajo de tesis.

Por otro lado, es de destacar las ventajas del uso de la computación paralela, y la necesidad de contar con profesionales que apliquen las metodologías necesarias para encontrar estrategias eficientes que resuelvan los desafíos de los problemas de ingeniería actuales.

Así mismo, es importante también decir que todas las librerías estudiadas y evaluadas en esta tesis son de libre distribución, y los resultados obtenidos ponen de manifiesto su calidad en aplicaciones prácticas.

La aplicación de métodos directos de resolución de SELD (como los contenidos en

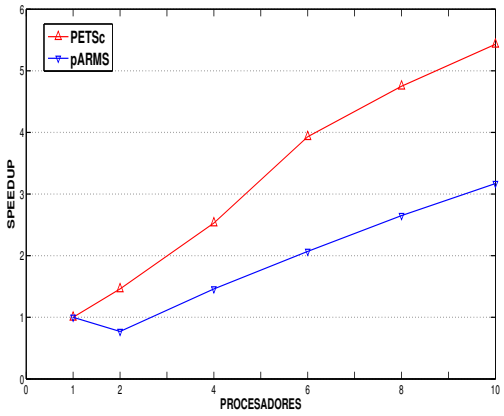

Figura 5.8: Gráfica de speedup en las librerías de PETSc y pARMS

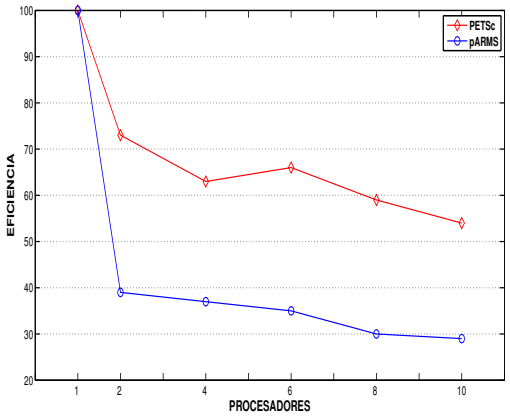

Figura 5.9: Gráfica de eficiencia en las librerías de PETSc y pARMS

la librería SuperLU) al caso de estudio ha demostrado que, a pesar de la calidad de la solución obtenida, el tiempo continua siendo prohibitivo para los problemas del caso de estudio aquí abordado.

Pruebas experimentales presentadas en trabajos como [64], muestran que para matrices prueba obtenidas de discretizaciones con métodos de colocación nodal en otros reactores, se obtienen resultados similares a los aquí mostrados.

Se puede decir que, para el caso de estudio expuesto, habrá que tomar en consideración la librería numérica de PETSc, así como recomendar el uso del método del Gradiente Conjugado con el precondicionador de Jacobi a bloques. Mención especial merece la librería numérica de SPARSKIT por su relativa eficiencia sobre PETSc en

lo concerniente a la ejecución secuencial. Relativa en el sentido de que el mejor tiempo secuencial registrado por la librería de PETSc (1.17 segundos), dista muy poco del registrado por SPARSKIT (1.14 segundos).

Los distintos experimentos realizados con PETSc muestran un alto grado de facilidad y flexibilidad en su uso al permitir de manera fácil que las implementaciones secuenciales sean ejecutadas prácticamente sin cambios en plataformas paralelas. En tanto que la librería SPARSKIT permite solamente implementaciones secuenciales. Además, en SPARSKIT se requiere de una mayor demanda de programación por parte del usuario que si se usara la librería PETSc. Por lo ya mencionado, puede decirse que para los casos de estudio, la librería PETSc es más adecuada que la librería SPARSKIT.

5.3. Solución de los Sistemas de Ecuaciones Lineales Dispersos de la Ecuación de Difusión Neutrónica. Caso Dinámico

En estudios de estabilidad y seguridad de los reactores nucleares, es fundamental estudiar su evolución en el tiempo. Por tal motivo, es importante resolver los SELD que aparecen como consecuencia de la discretización en el tiempo de la EDN 3D.

Esta sección presenta los resultados numéricos experimentales obtenidos de la aplicación de cuatro métodos iterativos multipaso, de los cuales dos forman parte principal de las aportaciones de este trabajo de tesis. Estos métodos se han codificado en el ambiente de desarrollo Matlab [28], y en las librerías numéricas de PETSc [101] [100] [99] e Hypre [110] [111].

5.3.1. Métodos Multipaso

En el capítulo 2, se ha establecido que para resolver la EDN en su caso dinámico después de aplicar discretizaciones espaciales y temporales, se requiere resolver en cada paso de tiempo un sistema de ecuaciones de la forma:

$$T\psi = E,$$

es decir:

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}, \tag{5.9}$$

donde el lado derecho de la ecuación depende de la solución en el paso de tiempo previo y del método de diferencias hacia atrás utilizado. Generalmente, la matriz de coeficientes del sistema (5.9) tiene propiedades similares a las matrices L y M de la ecuación (5.2) del caso estacionario. Por tal motivo, y debido a las condiciones de continuidad del flujo

y de la corriente neutrónica interpuestas en las caras de las celdas internas del reactor, así como las condiciones de contorno en los extremos, los bloques T_{11} y T_{22} son matrices simétricas definidas positivas. En tanto que los bloques T_{12} y T_{21} son matrices diagonales que pueden ser singulares o no y que dependen también de las condiciones del problema.

Dado el gran tamaño del sistema representado en (5.9), sería conveniente contar con un método que permita su resolución y que tomara en cuenta las propiedades de los bloques que forman el problema global. Por ejemplo, dadas las propiedades de las matrices T_{11} y T_{22}, un método basado en subespacios de Krylov (como el Gradiente Conjugado), sería apropiado para su resolución. No así la matriz total T, que no presenta ninguna de estas propiedades.

Por el motivo anterior, los métodos que se proponen en este trabajo, consisten en descomponer la resolución de la ecuación global (5.9) de tamaño $2N$, en términos de la resolución de dos sistemas de ecuaciones lineales de tamaño N. Con este método se evitaría en primer lugar tener que manipular el sistema global T, evitando así problemas de memoria, y en segundo lugar, permitiría la simulación con más grupos de energía.

A continuación se presentan y explican, cada uno de los métodos iterativos multipaso que permitieron implementar esta estrategia, así como las modificaciones que se proponen en esta memoria.

5.3.1.1. Método Multipaso de Jacobi (MMJ)

En el algoritmo 19, se muestran las sentencias de la implementación del algoritmo multipaso de Jacobi (MMJ) explicado en el capítulo 3.

Algoritmo 19 Método Multipaso de Jacobi (MMJ)

(* ψ_f^* y ψ_t^* soluciones de un paso previo *)
01 $\psi_f^0 := \psi_f^*$;
02 $\psi_t^0 := \psi_t^*$;

(* Resolver los grupos ψ_f^1 (rápido) y ψ_t^1 (térmico) *)
03 $T_{11}\psi_f^1 = e_1 - T_{12}\psi_t^0$
04 $T_{22}\psi_t^1 = e_2 - T_{21}\psi_f^0$

(* Ciclo externo *)
05 **do** l=1, 2, 3, ...
06 $T_{11}\psi_f^{l+1} = e_1 - T_{12}(\omega\psi_t^l + (1-\omega)\psi_t^{l-1})$; (* Ciclos internos de los *)
07 $T_{22}\psi_t^{l+1} = e_2 - T_{21}(\omega\psi_f^l + (1-\omega)\psi_f^{l-1})$; (* métodos de Krylov *)

(* Detectar convergencia *)
08 **until** $\|\psi_f^{l+1} - \psi_f^l\| < tol$ **and** $\|\psi_t^{l+1} - \psi_t^l\| < tol$

Este algoritmo tiene como primera etapa la inicialización de los flujos neutrónicos de tipo *rápido* (ψ_f^0) y *térmico* (ψ_t^0), con valores de los flujos de un paso de tiempo previo ψ_f^* y ψ_t^* (líneas 1 y 2). Posteriormente, se calculan nuevos valores para los flujos neutrónicos tomando en cuenta los otros bloques (T_{12} y T_{21}) del sistema global, presentes en la ecuación (5.9), de modo que se tengan dos soluciones consecutivas (líneas 3 y 4), con las que se puedan realizar las operaciones contenidas en el ciclo externo (líneas 5–8). Las operaciones contenidas en el ciclo externo son en sí ciclos internos que se corresponden con el cálculo de una solución de los flujos neutrónicos a través de una técnica de resolución de SELD como las basadas en subespacios de Krylov. Una vez realizado lo anterior, se verifica que el cambio entre la aproximación de una solución en la etapa $l+1$ respecto de la etapa l anterior es menor que cierta tolerancia (tol) establecida (línea 8). Aunque este criterio de convergencia es débil, se ha elegido por representar un bajo coste computacional en su cálculo.

5.3.1.2. Método Multipaso de Gauss–Seidel (MMGS)

El algoritmo que implementa el esquema iterativo multipaso de Gauss–Seidel acelerado se muestra en el algoritmo 20.

Algoritmo 20 Método Multipaso de Gauss–Seidel (MMGS)

(* ψ_f^* y ψ_t^* soluciones de un paso previo *)
01 $\psi_f^0 := \psi_f^*$;
02 $\psi_t^0 := \psi_t^*$;

(* Resolver para grupos ψ_f^1 y ψ_t^1 *)
03 $T_{11}\psi_f^1 = e_1 - T_{12}\psi_t^0$
04 $T_{22}\psi_t^1 = e_2 - T_{21}\psi_f^0$

(* Ciclo externo *)
05 **do** l=1, 2, 3, ...
06 $T_{11}\psi_f^{l+1} = e_1 - T_{12}(\omega\psi_t^l + (1-\omega)\psi_t^{l-1})$; (* Ciclos internos de los *)
07 $T_{22}\psi_t^{l+1} = e_2 - T_{21}(\omega\psi_f^{l+1} + (1-\omega)\psi_f^l)$; (* métodos de Krylov *)

(* Detectar convergencia *)
08 **until** $\|\psi_f^{l+1} - \psi_f^l\| < tol$ **and** $\|\psi_t^{l+1} - \psi_t^l\| < tol$

La diferencia principal entre los algoritmos 19 y 20 es que, en este último, tan pronto como está disponible la nueva solución para el grupo rápido (ψ_f^{l+1}), ésta es utilizada para calcular el grupo térmico (ψ_t^{l+1}) de la etapa $l+1$, como se observa en los pasos 6 y 7 del algoritmo 20 que corresponden a los ciclos internos del método de Krylov. Con este esquema puede esperarse una tasa de convergencia mayor, pero habría que tomar

en cuenta que, desde el punto de vista del paralelismo se pueden generar dependencias que pueden provocar tiempo ocioso en los procesadores.

5.3.1.3. Método Multipaso de Gauss–Seidel con Dos Parámetros de Relajación (MMGS2)

Una parte fundamental de este trabajo es la puesta en marcha de modificaciones a los dos métodos multipaso introducidos: MMJ y MMGS.

Una de estas modificaciones consiste en realizar pruebas con dos valores de ω diferentes para cada uno de los sistemas a resolver, con miras a acelerar la convergencia del método multipaso [33] [37] [36] [35]. Por ejemplo, el proceso de solución para el flujo rápido usará ω_1 (línea 6 del algoritmo 21), y el proceso de solución para el flujo térmico calculará con ω_2 (línea 7 del algoritmo 21). En esta propuesta, de la misma forma que en el algoritmo del método MMGS, el flujo ψ_t^{l+1} se calcula utilizando ψ_f^{l+1} (líneas 6 y 7 del algoritmo 21).

Algoritmo 21 Método Multipaso de Gauss–Seidel con 2 Parámetros de Relajación (MMGS2)

(* ψ_f^* y ψ_t^* soluciones de un paso previo *)
01 $\psi_f^0 := \psi_f^*$;
02 $\psi_t^0 := \psi_t^*$;

(* Resolver para grupos ψ_f^1 y ψ_t^1 *)
03 $T_{11}\psi_f^1 = e_1 - T_{12}\psi_t^0$
04 $T_{22}\psi_t^1 = e_2 - T_{21}\psi_f^0$

(* Ciclo externo *)
05 **do** l=1, 2, 3, ...
06 $T_{11}\psi_f^{l+1} = e_1 - T_{12}(\omega_1\psi_t^l + (1-\omega_1)\psi_t^{l-1})$; (* Ciclos internos de los *)
07 $T_{22}\psi_t^{l+1} = e_2 - T_{21}(\omega_2\psi_f^{l+1} + (1-\omega_2)\psi_f^l)$; (* métodos de Krylov *)

(* Detectar convergencia *)
08 **until** $\|\psi_f^{l+1} - \psi_f^l\| < tol$ **and** $\|\psi_t^{l+1} - \psi_t^l\| < tol$

5.3.1.4. Método Multipaso de Gauss–Seidel con Dos Parámetros de Relajación y Tolerancia Adaptativa (MMGS2A)

Otra aportación importante de este trabajo de tesis es el diseño de un algoritmo multipaso (basado en el método MMGS2), utilizando una técnica *adaptativa* [33] [37] [36]

[35]. Esta técnica, en palabras generales, consiste en que los sistemas lineales dispersos (T_{11} y T_{22}) se resuelven con una precisión inicial (ϵ_i) no restrictiva en etapas iniciales del método. Después, esta precisión es *adaptada* (o mejorada) hacia una precisión más demandante (ϵ_{i+1}), en iteraciones sucesivas. La aplicación de esta técnica al algoritmo 21, da origen al algoritmo 22.

Algoritmo 22 Método Multipaso Gauss–Seidel con 2 Parámetros de Relajación y Tolerancia Adaptativa (MMGS2A)

(* ψ_f^* y ψ_t^* soluciones de un paso previo *)
01 $\psi_f^0 := \psi_f^*$;
02 $\psi_t^0 := \psi_t^*$;

(* Establecer conjunto de tolerancias ϵ_i *)
03 $\epsilon_i = \{\epsilon_1, \epsilon_2, \epsilon_3, \ldots, \epsilon_n\}$
04 $i := 1$

(* Resolver para grupos ψ_f^1 y ψ_t^1 con tolerancia ϵ_1 *)
05 $T_{11}\psi_f^1 = e_1 - T_{12}\psi_t^0$
06 $T_{22}\psi_t^1 = e_2 - T_{21}\psi_f^0$

(* Ciclo externo *)
07 **do** l=1, 2, 3, ...
08 $T_{11}\psi_f^{l+1} = e_1 - T_{12}(\omega_1\psi_t^l + (1-\omega_1)\psi_t^{l-1})$; Resolver con precisión ϵ_i
09 $T_{22}\psi_t^{l+1} = e_2 - T_{21}(\omega_2\psi_f^{l+1} + (1-\omega_2)\psi_f^l)$; Resolver con precisión ϵ_i

10 **if** precisión en etapa $l+1$ es mejor que precisión en etapa l **then**
11 $i := i + 1$ (* mejorar ϵ_i*)
12 **end-if**

(* Detectar convergencia *)
13 **until** $\|\psi_f^{l+1} - \psi_f^l\| < tol$ **and** $\|\psi_t^{l+1} - \psi_t^l\| < tol$

Como en los algoritmos precedentes, el algoritmo 22 se inicializa con las soluciones de los flujos neutrónicos de un paso de tiempo previo (líneas 1–2). Después se establece el conjunto de tolerancias $\epsilon_i = \{\epsilon_1, \epsilon_2, \epsilon_1, \ldots, \epsilon_n\}$, así como la tolerancia ϵ_1 con la que se resolverán los sistemas inicialmente (líneas 3–6). Una vez calculados los flujos neutrónicos de la etapa $l+1$ (líneas 8–9) se determina la precisión obtenida en la etapa actual (líneas 10–12) y si resulta que es mejor que la precisión alcanzada en la etapa l, entonces se establece ϵ_i a una precisión ϵ_{i+1}. Por último, se verifica el criterio de parada (línea 13).

5.3.2. Caso de Estudio: Reactor Leibstadt

El caso de estudio a considerar ahora es el reactor nuclear suizo de Leibstadt [83] [11], abreviado KKL y que es un reactor del tipo agua en ebullición (del Inglés Boiling Water Reactor – BWR). Este reactor se ha discretizado utilizando un método de colocación nodal basado en desarrollos de polinomios de Legendre grado K =2 en forma 3D, imponiendo un discretizado espacial de 32*32 celdas por 27 planos axiales, de las cuales 19,656 corresponden al reactor. La figura 5.10 muestra uno de los planos axiales en los que se ha dividido el reactor.

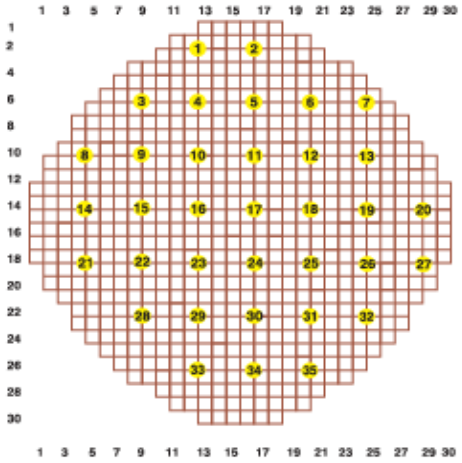

Figura 5.10: Plano axial del reactor Leibstadt

Así mismo, utilizando un esquema de numeración natural para la malla de discretización, el patrón de elementos no nulos resultante para los bloques diagonales $T_{ii}, i = 1, 2$ es el que se aprecia en la figura 5.11.

Al igual que como se ha hecho para las matrices del caso estacionario, la tabla 5.30 muestra algunas propiedades básicas de información de los bloques T_{ii} $(i = 1, 2)$ que surgen en el primer paso de tiempo, usando la rutina *info1.ex* de SPARSKIT. Haciendo un análisis análogo al realizado en el caso estacionario se encontrará que de optar por un formato tipo banda (`BND`) es necesario contar con un arreglo de tamaño $BW \times n$, lo que representa un número mucho mayor que el número de elementos distintos de cero (nnz) reales contenidos en los bloques T_{11} y T_{22}. Algo parecido ocurriría de optar por el almacenamiento tipo diagonal (`DIA`) en donde es necesario un arreglo de tamaño $DNE \times n$, el cual representa un número mayor del necesario para almacenar únicamente los elementos distintos de cero reales contenidos en los bloques $T_{ii}, i = 1, 2$. Por tal motivo, y al igual que en el caso estacionario, el formato CSR se ha elegido como formato de almacenamiento, el cual es contemplado por PETSc e Hypre.

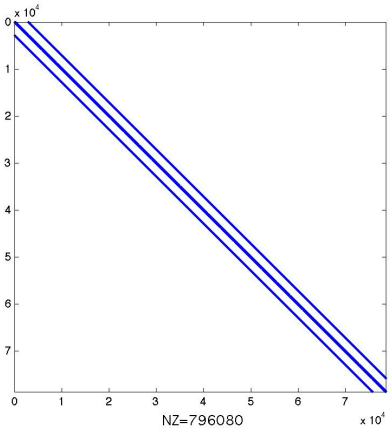

Figura 5.11: Patrón de dispersión del bloque T_{11} del caso Leibstadt

Tabla 5.30: Propiedades de los bloques T_{ii} del reactor Leibstadt

GEOMETRÍA	**PROPIEDAD**	T_{11}	T_{22}
	Dimensión – n	78624	78624
3D	Elementos no nulos – nnz	796080	796080
Con polinomio	Ancho de banda – BW	5826	5826
de Legendre	Diagonales no evitables –DNE	91	91
Grado 2	Diagonal dominante (%)	75 %	97 %
	Grado de simetría	1.00	1.00
	Índice de dispersión	0.00014	0.00014

Para probar el desempeño de los métodos MMJ, MMGS, MMGS2 y MMGS2A, se han elegido como matrices de prueba las correspondientes a cinco distintos pasos de tiempo (Si, donde $i = 0, 1, 2, 3, 4$), que corresponden a una prueba de estabilidad llevada a cabo en 1990 cuando el reactor oscilaba fuera de fase.

A continuación se presentarán los resultados experimentales obtenidos con los distintos métodos mencionados en el ambiente de Matlab, PETSc e Hypre.

5.3.3. Estudio Secuencial en Matlab

Se han realizado pruebas experimentales de los prototipos de los métodos MMJ, MMGS, MMGS2 y MMGS2A en el ambiente de programación Matlab con el objetivo de identificar los parámetros $\omega, \omega_1, \omega_2$ óptimos para cada uno de ellos [36] [37]. Por otro lado, para verificar la robustez de los métodos, se ha utilizado un vector aleatorio como

solución inicial (ψ^*) de los métodos iterativos multipaso para calcular la solución en el paso de tiempo S0, que a su vez se utilizará como solución inicial para calcular la solución en el tiempo S1, y así sucesivamente. Además, se ha comprobado la precisión alcanzada por los algoritmos mediante el cálculo de la norma del error residual (ERR.RES.) y el residuo relativo (ERR.REL.), formando la matriz original T con los bloques T_{11}, T_{22}, T_{12} y T_{21}. La solución de los sistemas dispersos con los bloques $T_{ii}, i = 1, 2$ en la parte que corresponde al ciclo externo, fue realizada utilizando el método de Krylov del Gradiente Conjugado (GC) sin precondicionador, exigiendo siempre que el residuo relativo sea menor o igual que 10^{-7} como criterio de convergencia. Las distintas pruebas experimentales se han realizado en el cluster Kubrick (ver sección 4.2.1) utilizando un $p = 1$ procesador y los tiempos se han registrado en segundos con la función Matlab de nombre `etime`.

5.3.3.1. Determinación de ω Óptimo en el método MMJ

Pruebas experimentales del prototipo del método MMJ en Matlab muestran que para los juegos de matrices del caso de estudio (reactor Leibstadt) el valor óptimo de ω está representado por 0.9 (ver tabla 5.31).

Tabla 5.31: Tiempos de ejecución (segs) en el método MMJ para distintos valores de ω. El símbolo † indica no convergencia

Si	ω =0.1	ω =0.5	ω =0.8	ω =0.9	ω =1.3	ω =1.9
S0	2996.63	2413.45	1963.14	**1791.38**	†	†
S1	537.87	464.55	397.72	**371.57**	†	†
S2	690.82	596.32	497.56	**460.74**	†	†
S3	654.71	567.64	473.48	**445.29**	†	†
S4	699.10	601.56	502.73	**468.78**	†	†

Se ha encontrado que la precisión alcanzada por el método MMJ es de 6 dígitos en el error residual y de 5 dígitos en el error relativo como lo muestra la tabla 5.32. Es importante comentar que el alto costo inicial asociado a la solución con el juego de matrices del paso de tiempo S0 (que es casi cuatro veces el costo en iteraciones para los otros juegos de matrices) es debido al uso de un vector aleatorio como vector solución inicial; sin embargo, este costo se ve disminuido en posteriores cálculos debido al uso de soluciones previas.

5.3.3.2. Determinación de ω Óptimo en el método MMGS

De los experimentos realizados con el algoritmo MMGS para determinar el valor óptimo de ω se ha encontrado que un valor de ω =1.3 proporciona el menor número

Tabla 5.32: Precisión y número de iteraciones externas en el método MMJ (ω óptimo)

Si	**ERR.RES.**	**ERR.REL.**	**ITS**$_{EXT}$
S0	1.54e-6	1.32e-5	4717
S1	2.52e-6	4.85e-5	1033
S2	2.50e-6	2.11e-5	1350
S3	2.54e-6	6.37e-5	1255
S4	2.52e-6	7.74e-5	1319

de iteraciones para las matrices del caso de estudio, como puede observarse en la tabla 5.33.

Tabla 5.33: Tiempos de ejecución (segs) con el método MMGS para distintos valores de ω

Si	$\omega = 0{,}1$	$\omega = 0{,}5$	$\omega = 0{,}9$	$\omega = 1{,}3$	$\omega = 1{,}5$	$\omega = 1{,}9$
S0	†	†	†	**411.60**	†	†
S1	†	†	493.47	**105.16**	†	†
S2	†	†	291.01	**121.43**	†	†
S3	338.28	415.24	286.01	**120.78**	451.42	†
S4	†	436.09	298.06	**125.50**	500.59	4875.58

Una comparación de tiempos entre los métodos MMJ (tabla 5.31) y los tiempos de la tabla 5.33 del método MMGS permiten observar que este último reduce los tiempos que registra el primero en al menos un 72 % en todos los conjuntos (Si) de matrices. Esto de alguna manera quiere decir que el método MMGS es, al menos para el caso de estudio abordado, 3 veces más rapido que MMJ, como lo indican los coeficientes de velocidad de MMGS respecto MMJ representados en tabla 5.34.

Tabla 5.34: Tiempos de ejecución (segs) y coeficientes de velocidad del método MMGS con respecto al método MMJ

Si	**MMJ**	**MMGS**	**COEFICIENTE VELOCIDAD**
S0	1791.38	411.60	4.35
S1	371.57	105.16	3.53
S2	460.74	121.43	3.79
S3	445.29	120.78	3.69
S4	468.78	125.50	3.74

Se ha determinado la precisión de la solución obtenida con el método MMGS, la cual es también de 6 dígitos en el error residual y de 5 dígitos en el error relativo (ver tabla 5.35).

Tabla 5.35: Precisión y número de iteraciones externas en el método MMGS (ω óptimo)

Si	**ERR.RES.**	**ERR.REL.**	**ITS**$_{EXT}$
S0	4.79e-6	4.12e-5	964
S1	4.48e-6	8.61e-5	277
S2	6.64e-6	5.62e-5	340
S3	4.69e-6	1.18e-4	323
S4	4.48e-6	1.37e-4	322

5.3.3.3. Determinación de ω_1 y ω_2 Óptimos en el método MMGS2

Para el caso del método MMGS2 se han realizado experimentos con ω_1 y ω_2 en el intervalo abierto $(1, 2)$, es decir, que $0 < \omega_1, \omega_2 < 2$. La tabla 5.36 muestra parte de los experimentos realizados para el conjunto de matrices que surgen en el paso de tiempo S0. No se consideraron pruebas para el resto de las matrices de los otros pasos dado que comparten características comunes.

La tabla 5.36 muestra que la combinación de valores de ω_1 y ω_2 que mejor resulta está dada por el test 5 (T5 en la tabla), donde para el ciclo externo (ITS$_{EXT}$) se obtienen 325 iteraciones con 149.43 segundos.

Tabla 5.36: Tiempos de ejecución (segs) para el método MMGS2 (ω_1, ω_2 óptimos)

TEST	ω_1	ω_2	**ITS**$_{EXT}$	**TIEMPO**	**ERR.RES.**	**ERR.REL.**
T1	1.3	0.1	2314	1164.80	7.22E-6	4.39E-4
T2	1.3	0.7	1766	888.72	6.06E-6	4.17E-4
T3	1.3	1.3	1109	557.98	6.05E-6	4.16E-4
T4	1.3	1.7	553	278.30	6.64E-6	4.28E-4
T5	**1.3**	**1.9**	**325**	**149.43**	**5.41E-6**	**4.69E-4**
T6	1.5	0.1	2314	1164.72	7.22E-6	4.39E-4
T7	1.5	0.3	2141	984.40	7.22E-6	4.29E-4
T8	1.5	0.5	1958	985.32	6.35E-6	4.22E-4

Sin embargo, durante los experimentos se ha encontrado que existen distintas combinaciones que presentan un comportamiento parecido al test T5. La tabla 5.37 reune tales combinaciones.

Tabla 5.37: Tiempos de ejecución (segs) para el método MMGS2 (ω_1, ω_2 óptimos) (BIS)

ω_1	ω_2	**ERR.RES.**	**ERR.REL.**	**ITS**$_{EXT}$	**TIEMPO**
0.3	1.9	5.51E-6	4.06E-4	327	151.01
0.5	1.9	5.85E-6	4.12E-4	326	149.96
0.7	1.9	5.76E-6	4.11E-4	326	149.61
0.9	1.9	5.68E-6	4.09E-4	326	149.95
1.3	1.9	5.41E-6	4.69E-4	325	149.43
1.5	1.9	5.85E-6	4.12E-4	325	149.59
1.7	1.9	5.76E-6	4.11E-4	325	149.55
1.9	1.9	5.68E-6	4.09E-4	325	149.96

Las pruebas experimentales con el método MMGS2 se han realizado con los valores óptimos ω_1 =1.3 y ω_1 =1.9 obteniéndose los tiempos de ejecución de la tabla 5.38, la cual deja ver que el método MMGS2 disminuye el tiempo de MMGS a lo más en un 64 % para todos los juegos de matrices en todos los pasos de tiempo. No obstante la ganancia de tiempo frente a MMGS (por consiguiente frente a MMJ), el método MMGS2 presenta una pérdida de precisión de 1 dígito para algunos casos en el error relativo, pero mantiene la precisión del error residual hallada en métodos anteriores.

Tabla 5.38: Precisión y número de iteraciones externas en el método MMGS2 (ω_1, ω_2 óptimos)

Si	**ERR.RES.**	**ERR.REL.**	**ITS**$_{EXT}$	**TIEMPO**
S0	5.39e-6	4.63e-4	**325**	**149.43**
S1	5.84e-6	1.12e-4	**123**	**49.76**
S2	8.78e-6	7.43e-5	**133**	**50.72**
S3	7.64e-6	1.92e-4	**130**	**51.95**
S4	7.49e-6	2.30e-5	**130**	**54.38**

La tabla 5.39 muestra los índices de velocidad del método MMGS2 sobre MMGS, donde se puede observar que MMGS2 obtiene la solución en al menos la mitad del tiempo invertido por MMGS.

5.3.3.4. Pruebas Experimentales con el método MMGS2A

Los experimentos numéricos mediante la técnica adaptable representada por el método MMGS2A y con los valores óptimos ω_1 y ω_2 obtenidos en el apartado anterior se presentan en la tabla 5.40.

Como puede verse, la eficiencia del método MMGS2A es ligeramente más eficiente

Tabla 5.39: Tiempos de ejecución (segs) y coeficientes de velocidad del método MMGS2 con respecto al método MMGS

Si	**MMGS**	**MMGS2**	**COEFICIENTE VELOCIDAD**
S0	411.60	149.43	2.75
S1	105.16	49.76	2.11
S2	121.43	50.72	2.39
S3	120.78	51.95	2.32
S4	125.50	54.38	2.31

Tabla 5.40: Precisión y tiempos de ejecución (segs) para el método MMGS2A con respecto al método MMGS2

Si	**ERR.RES.**	**ERR.REL.**	**ITS$_{EXT}$**	**TIEMPO**
S0	5.44e-6	4.67e-4	**352**	**82.58**
S1	5.95e-6	1.14e-4	**123**	**39.31**
S2	8.88e-6	7.51e-5	**133**	**39.56**
S3	7.77e-6	1.95e-4	**130**	**40.59**
S4	7.52e-6	2.31e-5	**130**	**42.89**

que el método MMGS2 debido a las tolerancias ϵ_i impuestas en las etapas iniciales del método, lo cual impacta directamente en el tiempo total de ejecución. También puede observarse que a pesar de manejar tolerancias distintas durante todo el proceso, el método adaptativo mantiene la misma precisión que MMGS2.

Una comparativa de los tiempos obtenidos por el método MMGS2A frente al método MMGS2 permite observar que el primero reduce el tiempo del segundo a lo más en un 45 % para todos los juegos de matrices del caso de estudio. Los índices de aceleración del MMGS2A con respecto al MMGS2 se representan en la tabla 5.41.

La utilización de Matlab ha sido importante para probar los distintos métodos, determinar los valores óptimos de ω y determinar la precisión alcanzada por los mismos, por lo que considerando los tiempos totales de ejecución para todos los pasos de tiempo, se observó que los métodos más eficientes están representados por los métodos MMGS2 y MMGS2A (ver figura 5.12).

5.3.4. Análisis en la Librería PETSc. Plataforma Kefren

Una vez encontrados los valores óptimos para los distintos métodos en el ambiente de Matlab, se ha procedido a implementarlos utilizando la librería numérica de PETSc,

Tabla 5.41: Tiempos de ejecución (segs) y coeficientes de velocidad del método MMGS2 con respecto al método MMGS

Si	**MMGS2**	**MMGS2A**	**COEFICIENTE VELOCIDAD**
S0	149.43	82.58	1.81
S1	49.76	39.31	1.27
S2	50.72	39.56	1.28
S3	51.95	40.59	1.28
S4	54.38	42.89	1.27

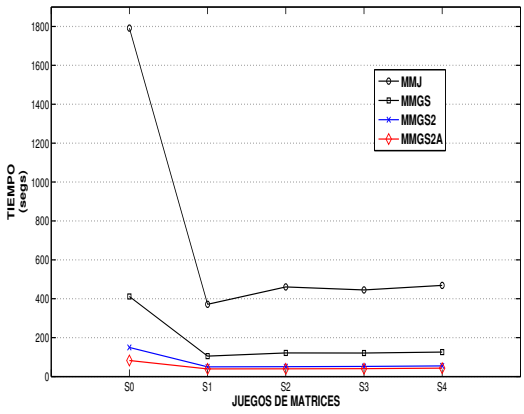

Figura 5.12: Gráfica de tiempos de ejecución de los métodos MMJ, MMGS, MMGS2 y MMGS2A

aplicándolos al juego de matrices S0 del caso de estudio. Estudios similares se presentan en [35].

Algunas operaciones en PETSc usadas para la implementación de los métodos han sido las siguientes:

- `VecNorm` Cálcula la norma de un vector,
- `VecPointwiseMult` Calcula la multiplicación *componente–a–componente* $w = x \times y$,
- `VecAYPX` Calcula suma de vectores $y = x + \alpha y$,
- `VecCopy` Copia un vector,
- `KSPSolve` Resuelve un sistema lineal.

La solución de los SELD asociados a las matrices T_{ii} ha sido realizada sin el uso de precondicionadores. A diferencia de los experimentos realizados en Matlab, en PETSc

todos los métodos tienen como vector inicial ψ^* un vector solución aproximado (que fue obtenido perturbando la solución del sistema en el tiempo S0) en vez de uno aleatorio, en el entendido de que los algoritmos trabajarán con un vector inicial cercano a la solución. A continuación, se presenta el análisis secuencial realizado en PETSc sobre la plataforma paralela Kefren, registrando los tiempos de ejecución en segundos.

5.3.4.1. Estudio Secuencial

La tabla 5.42 presenta los tiempos de ejecución secuencial para todos los métodos donde puede observarse que el método MMGS2A es el que mejor desempeño secuencial proporciona, lo que viene a confirmar los estudios experimentales realizados con Matlab. También es importante destacar que la precisión alcanzada utilizando el error relativo de los algoritmos ha sido de 5 dígitos.

Tabla 5.42: Tiempos de ejecución (segs) de los métodos multipaso con los valores óptimos ω_1 y ω_2.

MÉTODO	**ITS**$_{EXT}$	**TIEMPO**	**ERR.REL.**
MMJ	394	138.08	7.58e-5
MMGS	183	63.98	4.21e-5
MMGS2	89	36.29	5.42e-5
MMGS2A	90	27.31	5.42e-5

Resalta el hecho de que el método multipaso MMGS2A realiza más iteraciones que el método MMGS2, sin embargo estas son menos costosas debido a la técnica adaptativa de la precisión en la resolución de los SELD que implementa el método.

5.3.4.2. Estudio Paralelo

Respecto de las pruebas paralelas en el cluster Kefren en donde se han utilizado hasta p =12 procesadores, el método MMJ presenta un buen grado de paralelismo debido a que los diferentes sistemas lineales de ecuaciones pueden resolverse simultáneamente por diferentes grupos de procesadores, y después intercambiar las soluciones. Por tal razón, se han implementado dos versiones paralelas de este método basadas en dos diferentes rutinas de comunicacióne de MPI: *gather/scatter* y *send/recv*. Además, se ha hecho uso de una rutina contenida en MPI para administrar grupos de procesos mediante el uso de comunicadores. Por ejemplo, para el caso de usar $p = 2$ procesadores en el método MMJ, el procesador p_0 es dedicado a resolver el sistema con bloque T_{11} y el procesador p_1 se dedica a resolver el sistema con bloque T_{22}. En caso de usar $p = 4$ procesadores, entonces $\frac{p}{2}$ procesadores son dedicados a resolver el sistema con el bloque T_{11} y el resto de procesadores son dedicados a resolver el sistema con el bloque T_{22}. Los tiempos

resultantes de la aplicación de estas estrategias para diferentes números de procesadores (p) se registran en la tabla 5.43, donde pueden observarse que: por un lado, la estrategia basada en las rutinas de comunicación *send/recv* es ligeramente más eficiente que la estrategia basada en las rutinas *gather/scatter* y por otro lado, que destaca el uso de cómputo paralelo en la disminución del tiempo de ejecución secuencial.

Tabla 5.43: Tiempos de ejecución (segs) en paralelo T_p del método MMJ en la plataforma Kefren

p	**GATHER/SCATTER**	**SEND/RECV**
1	138.08	138.08
2	89.33	**88.27**
4	54.90	**53.07**
6	40.51	**38.05**
8	33.86	**31.30**
12	25.47	**22.60**

La realización y aplicación de los distintos métodos multipaso básicos (MMJ y MMGS), así como los métodos multipaso propuestos en este trabajo de tesis (MMGS2 y MMGS2A) en PETSc da origen a los tiempos de ejecución paralelos representados en la tabla 5.44, donde se ha tomado como valor del mejor tiempo secuencial (T_s) el tiempo de ejecución del programa paralelo registrado con p=1 procesador.

Tabla 5.44: Tiempos de ejecución (segs) en paralelo T_p de los métodos multipaso en la plataforma Kefren

p	**MMJ**	**MMGS**	**MMGS2**	**MMGS2A**
1	138.08	63.98	36.29	27.31
2	88.27	36.83	21.11	15.83
4	53.07	28.78	12.01	8.90
6	38.05	20.26	8.68	6.51
8	31.30	11.94	6.91	5.17
12	22.60	10.41	5.23	3.90

De la tabla 5.44 destaca el hecho de que el uso de computación de alto desempeño ha disminuido los tiempos de ejecución secuencial para todos y cada uno de los métodos estudiados. Por ejemplo, para los métodos MMJ y MMGS, los tiempos de ejecución secuencial han sido reducidos hasta en un 84 % de su valor cuando usamos $p = 12$ procesadores; en tanto que para el caso de los métodos MMGS2 y MMGS2A, el tiempo de ejecución se ha reducido en un 86 % usando el mismo número de procesadores en la plataforma Kefren. En general, el método MMGS2A ofrece los mejores tiempos de ejecución en paralelo.

5.3.4.3. Speedup y Eficiencia

Se han calculado los coeficientes de speedup y eficiencia para todos los métodos obtenidos de las pruebas paralelas en Kefren y se reunen en la tabla 5.45.

Tabla 5.45: Coeficientes de speedup y eficiencia en la plataforma Kefren

p	SPEEDUP				EFICIENCIA(%)			
	MMJ	MMGS	MMGS2	MMGS2A	MMJ	MMGS	MMGS2	MMGS2A
1	1.00	1.00	1.00	1.00	100.00	100.00	100.00	100.00
2	1.56	1.74	1.72	1.73	78.21	86.86	85.95	86.26
4	2.60	2.22	3.02	3.07	65.05	55.58	75.54	76.71
6	3.63	3.16	4.18	4.20	60.48	52.63	69.68	69.92
8	4.41	5.36	5.25	5.28	55.14	66.98	65.65	66.03
12	6.11	6.15	6.94	7.00	50.91	51.22	57.82	58.35

En esta tabla se ve que los coeficientes de speedup y eficiencia registrados por todos los métodos son buenos cuando se usa un número entre 2 y 8 procesadores y aceptables con p =12 procesadores. También es interesante notar que los métodos MMGS2 y MMGS2A ofrecen coeficientes de speedup mejores que MMJ y MMGS para el caso de estudio, además que los porcentajes de eficiencia permanecen por arriba del 50 % para el caso de usar $p \leq 12$ procesadores. Las figuras 5.13 y 5.14 muestran gráficas de los coeficientes de speedup y eficiencia que ilustran mejor lo antes dicho.

Todos estos tiempos fueron calculados sin el uso de precondicionadores de PETSc, no obstante, no desmerece, la eficiencia registrada por los métodos.

5.3.5. Análisis en las Librerías de Hypre y PETSc. Plataforma Jacquard

Esta parte del trabajo describe las pruebas experimentales de los métodos MMJ, MMGS, MMGS2 y MMGS2A con las librerías numéricas de Hypre y PETSc. Las pruebas se han realizado en un subgrupo del cluster de altas prestaciones llamado Jacquard (ver sección 4.2.3) de los laboratorios ubicados en el Centro Nacional de Computación Científica en Investigación de la Energía (NERSC), dependiente del Departamento de Energía de los Estados Unidos de Norteamérica y los tiempos de ejecución se han registrado utilizando las rutinas `MPI_Wtime` en el caso de Hypre y `PetscGetTime` en el caso de PETSc. Parte de los estudios aquí presentados han dado origen a [33].

Hypre es una librería numérica de precondicionadores de alto desempeño para desarrollar algoritmos escalables que resuelven SELD de gran dimensión sobre computadoras paralelas (ver 4). Estos experimentos permitirán realizar una comparativa de desempeño paralelo entre las librerías de Hypre y PETSc. Se empezará con el estudio secuencial que se describe a continuación.

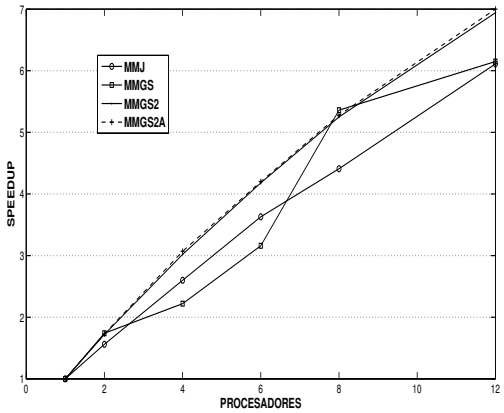

Figura 5.13: Gráfica de speedup en la plataforma Kefren

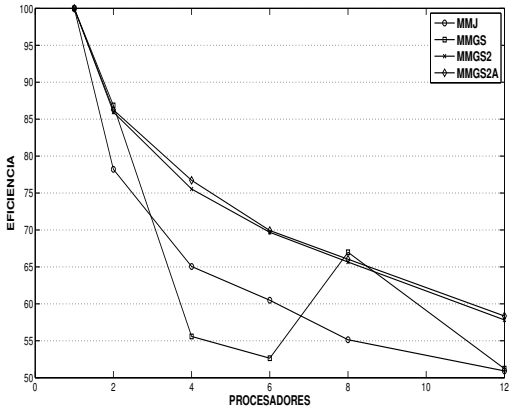

Figura 5.14: Gráfica de eficiencia en la plataforma Kefren

5.3.5.1. Estudio Secuencial

En las distintas pruebas con la librería Hypre, se han usado los siguientes precondicionadores combinados con el método del Gradiente Conjugado:

- *Diagonal Scaling(DS)*,
- *BoomerAMG*,
- *ParaSails*,
- *Euclid*.

El precondicionador DS es el equivalente al precondicionador de Jacobi. En el caso del precondicionador BoomerAMG, que es una implementación paralela de un método multimalla algebraico, las pruebas se han realizado con valor de *nlevel*=1 y valor *threshold*=0.1. En el caso de ParaSails, que es un precondicionador inverso aproximado disperso, se usaron valores de *nlevel* y *threshold* idénticos al caso BoomerAMG. Por otra parte, Euclid, que es un precondicionador paralelo tipo ILU que puede usarse con nivel de relleno k y *threshold* (umbral), se usó en su forma equivalente a Jacobi a bloques, es decir, con k =0 y valor de *threshold*=0.0. Experimentos en el cluster Jacquard sin y con los precondicionadores mencionados anteriormente han dado origen a los tiempos de ejecución mostrados en la tabla 5.46.

Tabla 5.46: Tiempos de ejecución (segs) con la librería Hypre en la plataforma Jacquard

Método	**PRECONDICIONADOR**				
	Sin PC	**DS**	**BoomerAMG**	**ParaSails**	**Euclid**
MMJ	403.12	310.43	1955.17	290.23	**236.66**
MMGS	192.18	147.81	885.01	133.50	**111.73**
MMGS2	112.48	86.57	433.25	77.87	**63.60**
MMGS2A	79.08	61.45	434.69	56.04	**48.27**

De esta tabla pueden hacerse los siguientes comentarios. Primero, todos los métodos muestran prestaciones similares a las observadas en los experimentos con el cluster Kefren, donde la eficiencia de los métodos MMGS2 y MMGS2A es mejor que la presentada en los métodos MMJ y MMGS. La figura 5.15 ilustra mejor lo afirmado.

Por otra parte, la aplicación de las técnicas de precondicionamiento han acelerado la tasa de convergencia de los métodos multipaso; por ejemplo, el método MMGS2A combinado con el precondicionador *Euclid*, ha reducido el tiempo del caso sin precondicionar en un 60 %. Esto confirma la influencia del uso de precondicionamiento para mejorar el desempeño de los métodos multipaso. Por otro lado, la aplicación de un precondicionador disperso aproximado inverso (*ParaSails*) ha sido ligeramente más eficiente comparado con el precondicionador tipo escalado diagonal (*DS*) para las matrices del caso de prueba. Pruebas experimentales con el método BoomerAMG presentan tiempos no tan eficientes debido al uso de las técnicas multinivel, que para el caso de estudio y por las distintas pruebas experimentales llevadas a cabo, se ha encontrado que no han sido apropiadas.

A fin de hacer una comparación entre el desempeño obtenido por las librerías numéricas de PETSc e Hypre, y dado que los experimentos con la librería Hypre se han realizado en el cluster de Jacquard, se hizo necesario recalcular los resultados experimentales de los métodos de la librería PETSC en la plataforma Jacquard. Los tiempos secuenciales

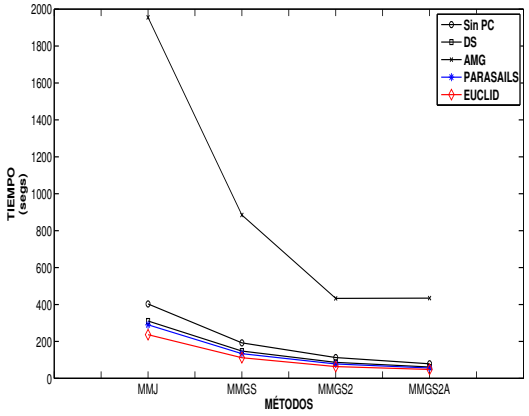

Figura 5.15: Gráfica de tiempos de ejecución de los métodos con la librería Hypre en la plataforma Jacquard

de la librería PETSc se muestran en la tabla 5.47.

Tabla 5.47: Tiempos de ejecución (segs) con la librería PETSc en la plataforma Jacquard

Método	**PRECONDICIONADOR**		
	Sin PC	**JACOBI**	**JACB**
MMJ	290.17	218.00	**176.28**
MMGS	148.72	104.71	**84.23**
MMGS2	85.45	62.19	**47.09**
MMGS2A	61.81	45.21	**36.39**

Como se puede observar en la tabla 5.47 y al igual que en resultados obtenidos de aplicar los métodos a las matrices del caso de estudio en Hypre, se puede ver que los tiempos secuenciales mejores son aquellos reportados por los métodos propuestos en este trabajo de tesis: MMGS2 y MMGS2A. También al usar técnicas de precondicionamiento, se logra reducir el tiempo de ejecución secuencial de los casos sin precondicionador (ver figura 5.16). Por ejemplo, para el caso del método MMGS2A se logra reducir el tiempo del caso sin precondicionar hasta en casi la mitad del tiempo cuando se combina con un precondicionador de tipo Jacobi a bloques.

5.3.5.2. Estudio Paralelo

Para el caso de los experimentos paralelos con la librería Hypre, se muestran aquellos resultados que tienen que ver con el precondicionador Euclid, puesto que presentó los

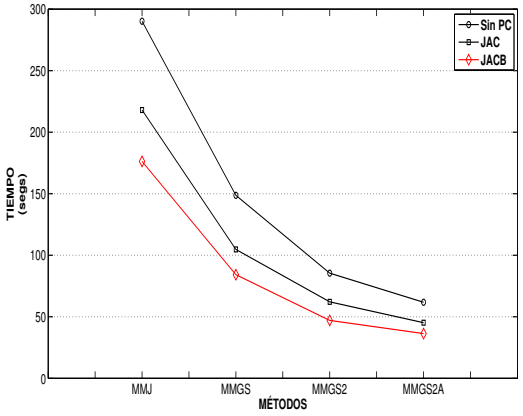

Figura 5.16: Gráfica de tiempos de ejecución de los métodos con la librería de PETSc en la plataforma Jacquard

menores tiempos en las pruebas realizadas en secuencial. Los tiempos paralelos se recogen en la tabla 5.48, donde nuevamente el uso de cómputo paralelo muestra sus ventajas al reducir el tiempo secuencial para todos los métodos.

Tabla 5.48: Tiempos de ejecución (segs) en paralelo T_p en la librería Hypre con el precondicionador Euclid

p	**MMJ**	**MMGS**	**MMGS2**	**MMGS2A**
1	236.66	111.73	63.60	48.27
2	128.22	62.82	35.39	26.02
4	71.94	33.54	19.62	14.64
6	49.23	23.27	13.46	9.84
8	38.53	18.16	10.28	7.89
10	31.68	14.95	8.70	6.39
14	24.11	11.38	6.84	4.88
16	22.06	10.51	6.12	4.45

Los tiempos de la tabla 5.48 muestran que en la librería numérica de Hypre y para el caso de estudio abordado, se ha logrado reducir el tiempo cuando se usa p=1 procesador en casi un 10 % que cuando se usan $p = 16$ procesadores en todos los métodos. También se observa que el método MMGS2A permanece registrando los tiempos de ejecución menores para cualquier valor de p. La figura 5.17 muestra más claramente lo antes afirmado.

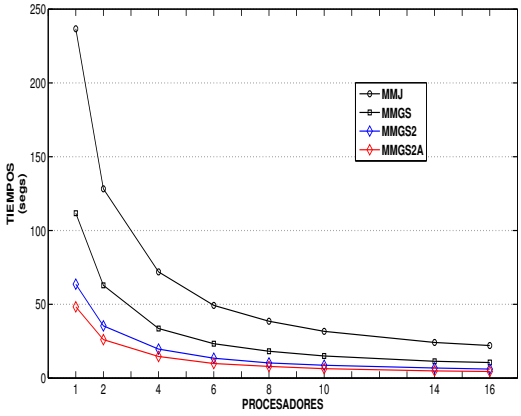

Figura 5.17: Gráfica de tiempos de ejecución en paralelo de los métodos con la librería Hypre en la plataforma Jacquard

A continuación, se presentan los mejores tiempos paralelos de ejecución obtenidos en la librería PETSc, los cuales corresponden al precondicionador de Jacobi a bloques (ver tabla 5.49). En este caso el uso de cómputo paralelo ha logrado reducir el tiempo registrado con p =1 procesador en un 89 % de su valor en todos los métodos cuando se usa un valor de p igual a 16 procesadores.

Tabla 5.49: Tiempos de ejecución (segs) en paralelo T_p en la librería PETSc con el precondicionador JACB

p	**MMJ**	**MMGS**	**MMGS2**	**MMGS2A**
1	176.28	84.23	47.09	36.39
2	102.18	48.91	28.69	21.37
4	58.27	27.51	16.31	12.04
6	42.49	19.95	11.65	8.74
8	33.70	15.87	9.19	6.92
10	29.34	13.85	8.04	6.06
14	23.25	10.88	6.50	4.77
16	21.93	10.03	5.95	4.60

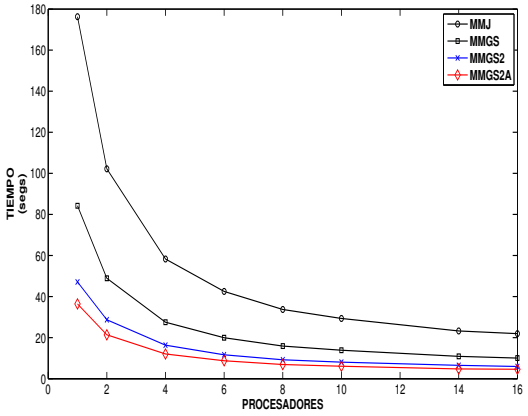

Figura 5.18: Gráfica de tiempos de ejecución en paralelo de los métodos con la librería PETSc en la plataforma Jacquard

5.3.5.3. Speedup y Eficiencia. Una Comparativa entre las Librerías Hypre y PETSc

Las distintas pruebas secuenciales y paralelas de los métodos en las librerías numéricas de Hypre y PETSc, permiten realizar una comparativa desde el punto de vista de las métricas de desempeño paralelo, tales como speedup y eficiencia, de los mejores métodos obtenidos que fueron GC con Jacobi a bloques en PETSc, y GC con Euclid en Hypre. En esta parte, los coeficientes de speedup y eficiencia paralela se han calculado tomando como valor de mejor tiempo secuencial (T_s), el tiempo de ejecución registrado por el programa paralelo con p=1 procesador para cada una de las librerías y que se presentan en la tabla 5.50.

En general, puede decirse que el desempeño paralelo registrado por ambas librerías numéricas en la plataforma Jaquard, no se degrada por debajo de 16 procesadores. Sin embargo, para el caso de la librería Hypre, aun cuando registra tiempos de ejecución ligeramente más altos que PETSc, muestra mejores coeficientes de speedup y eficiencia. Por ejemplo, para el caso de $p = 16$ procesadores, la librería de Hypre arroja un speedup de 10.8 y una eficiencia en el uso de la plataforma paralela de 68 %, contra valores de speedup y eficiencia de 7.9 y 49 % registrados por el uso de la librería PETSc, respectivamente. Los coeficientes de desempeño paralelo registrados por la librería Hypre, indican el nivel de eficiencia en el uso de los recursos del cluster Jacquard (ver figura 5.20).

Respecto de los tiempos paralelos en ambas librerías, se observa que en términos de desempeño, la librería PETSc es ligeramente mejor que Hypre (ver la figura 5.19). No obstante y en general, ambas librerías muestran un desempeño aceptable para el caso

Tabla 5.50: Coeficientes de speedup y eficiencia obtenidos con las librerías de Hypre y PETSc en la plataforma Jacquard

p	**Hypre**			**PETSc**		
	T_p	S_p	E_p	T_p	S_p	E_p
1	48.27	1.0	100 %	36.39	1.0	100 %
2	26.02	1.9	93 %	21.37	1.7	85 %
4	14.64	3.3	82 %	12.04	3.0	76 %
6	9.84	4.9	82 %	8.74	4.2	69 %
8	7.89	6.1	76 %	6.92	5.3	66 %
10	6.39	7.6	76 %	6.06	6.0	60 %
14	4.88	9.9	71 %	4.77	7.6	54 %
16	4.45	10.8	68 %	4.60	7.9	49 %

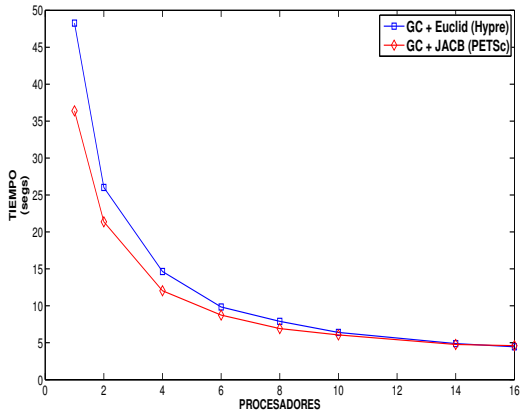

Figura 5.19: Gráfica de tiempos de ejecución en paralelo en las librerías Hypre y PETSc en la plataforma Jacquard

de estudio.

5.3.6. Conclusiones del Caso Dinámico

La solución eficiente de los sistemas lineales dispersos que surgen de la discretización de la EDN 3D Multigrupo dependiente del tiempo, juega un papel central en los estudios de estabilidad y seguridad de los reactores nucleares. Por tal motivo, esta memoria ha mostrado los resultados de dos métodos iterativos multipaso, uno basado en una partición de Jacobi (MMJ) y otro basado en una partición de Gauss–Seidel (MMGS), aplicados

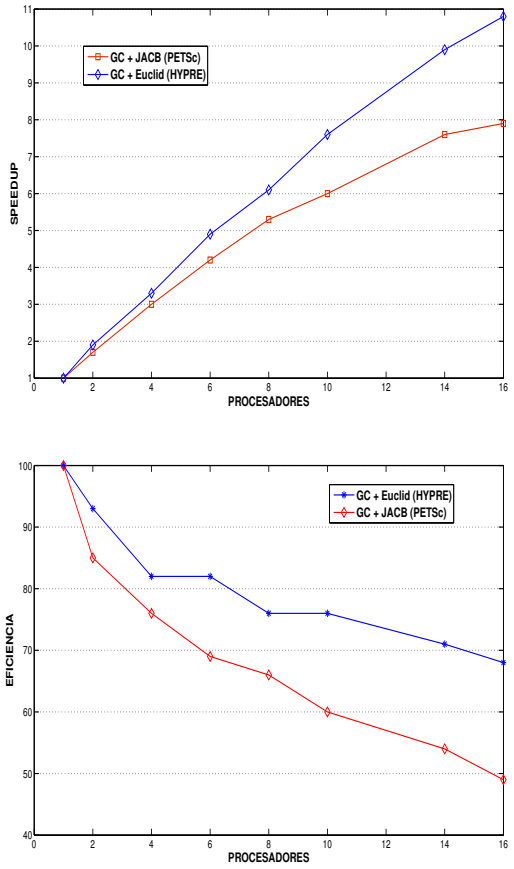

Figura 5.20: Gráfica de speedup y eficiencia con las librerías de Hypre y PETSc en la plataforma Jacquard

a un conjunto de matrices de un reactor nuclear real (Leibstadt) utilizando las librerías numéricas paralelas de PETSc e Hypre y realizando pruebas experimentales en dos clusters de alto desempeño: Kefren y Jacquard.

Adicionalmente, y como aportaciones originales de este trabajo de tesis, se han modificado los métodos multipaso MMJ y MMGS con el objetivo de acelerar su velocidad de convergencia. Para esto, se han implementado dos versiones: la primera de ellas, está basada en dos diferentes parámetros de relajación (ω_1,ω_2) para cada grupo de energía, a la cual se le ha nombrado método MMGS2. La segunda de ellas, que ha dado lugar al llamado método MMGS2A, está basado en una técnica *adaptable* que mejora aun más el desempeño con respecto a otros métodos, como ha podido verse a lo largo de las distintas pruebas realizadas, mismas que han sido apoyados por tablas de tiempos y gráficas

comparativas.

Para mejores resultados, se ha realizado un estudio heurístico en el ambiente de desarrollo de Matlab, de los parámetros óptimos de relajación para cada uno de los métodos en el caso de estudio abordado. Estos parámetros han ayudado a acelerar su convergencia, especialmente para MMGS2 y MMGS2A.

También, todos los métodos multipaso han sido combinados con un conjunto de precondicionadores contenidos en dos librerías de libre distribución: PETSc e Hypre. Para el caso de PETSc, los precondicionadores aplicados han sido Jacobi y Jacobi a bloques. Para el caso de la librería Hypre, los precondicionadores aplicados fueron Escalado Diagonal (DS), BoomerAMG, ParaSails y Euclid. Todos los precondicionadores fueron combinados con el método del Gradiente Conjugado.

De las pruebas realizadas, los precondicionadores más eficientes han sido Jacobi a bloques en el caso de PETSc y Euclid en el caso de Hypre. Se ha encontrado también que los precondicionadores de PETSc presentan menores tiempos de ejecución que los de Hypre.

Se ha logrado decrementar el tiempo secuencial con la ayuda del cómputo paralelo, alcanzándose coeficientes aceptables de speedup y eficiencia para ambas librerías. Sin embargo, a pesar que la librería paralela Hypre registra tiempos de ejecución menos eficientes que los presentados por la librería PETSc, en contraste presenta índices más altos de speedup y eficiencia, lo que significa un alto nivel de optimización de los algoritmos de Hypre en el uso de los recursos computacionales de la plataforma Jacquard.

La principal ventaja de los métodos de segundo grado presentados en esta memoria, es que la matriz T del sistema a resolver no necesita ser formada explícitamente, por lo que la simulación con más de dos grupos de energía es factible.

En general, y de acuerdo a lo ya expuesto, el método multipaso más adecuado para resolver los sistemas lineales dispersos asociados a la EDN en su caso dinámico para el caso de estudio ha sido uno de los propuestos por esta memoria: MMGS2A; por otra parte, la herramienta software que ha resultado más adecuada para implementar el MMGS2A ha sido PETSc.

Capítulo 6

Extensión de los Métodos y Algoritmos con más de 2 Grupos de Energía

6.1. Introducción

Los métodos, pruebas y resultados experimentales mostrados en el capítulo 5 han sido aplicados al caso de la modelización de la ecuación de difusión neutrónica (EDN) (tanto caso estacionario como dinámico) usando G =2 grupos de energía (flujo rápido y térmico). Sin embargo, aunque el grupo de investigación no cuenta actualmente con las matrices originadas por discretizaciones utilizando más de dos grupos de energía, es posible que en un futuro próximo se cuente con dichos datos, ya que existe un fuerte interés por realizar simulaciones que involucren un mayor número de grupos de energía [23], para lo cual sería interesante describir algoritmos que puedan resolver los problemas asociados con la EDN multigrupo.

Por tal motivo, este capítulo se ha organizado de la siguiente manera: la sección 6.2 plantea, bajo ciertas suposiciones de los flujos neutrónicos, tres escenarios distintos para la EDN con $G > 2$ grupos de energía. Por otra parte, la sección 6.3 presenta la posible estructura de las matrices que surgirían de la discretización usando $G > 2$ grupos de energía para el caso de la ecuación de los modos Lambda (caso estacionario de la EDN) y presenta algunas formas de paralelización que pueden sacar ventaja de las estructuras de las matrices con la nueva modelización. La sección 6.4 por su parte, realiza el mismo estudio para el caso dinámico (estudio de transitorios), además de presentar la forma en la que un método multipaso basado en la partición Gauss–Seidel se adaptaría

para resolver el sistema de ecuaciones lineales disperso cuando se tienen más grupos de energía. Por último, la sección 6.5 enuncia algunas conclusiones sobre lo mostrado en este capítulo.

6.2. Planteamientos Hipotéticos

Antes de comenzar con la descripción de los algoritmos que resuelvan la EDN para más de dos grupos de energía, es importante establecer ciertas suposiciones generales acerca de las condiciones de los flujos neutrónicos que se estudiarán. Para esto se han planteado tres escenarios distintos que se exponen a continuación a través de tres hipótesis, en donde G denota el total de grupos de energía a estudiar y g_i denota el grupo i–ésimo de energía.

H1 Existen procesos de dispersión (o trasvase de energía) del grupo g_i a los grupos $g_{i+1}, \ldots, g_G$, para cada $i = 1, 2, ..., G$.

H2 Existen procesos de dispersión del grupo g_1 al resto de los grupos g_i $(i = 2, 3, \ldots, G)$. Además, el trasvase de energía existe del grupo g_i única y exclusivamente al grupo g_{i+1}, para $i = 2, 3, ..., G - 1$.

H3 Se presenta trasvase de energía del grupo g_i única y exclusivamente al grupo g_{i+1}, para $i = 1, 2, ..., G - 1$.

Suponiendo que la discretización de la EDN para G grupos de energía en su caso estacionario deriva en un sistema de la forma

$$L\psi = \frac{1}{k_n} M\psi,$$

análogamente al sistema (2.55) del capítulo 2 para $G = 2$, se contempla que en M existen tantas submatrices $M_{1i}, i = 1, \ldots, G$ como número G de grupos de energía se estén manejando, es decir, la matriz M tendrá la forma de la expresión (6.1)

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1,G} \\ 0 & \cdots & \cdots & 0 \\ 0 & & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}. \tag{6.1}$$

Establecidas las condiciones bajo las cuales se tratará el problema, a continuación se abordan los distintos estudios propuestos.

6.3. Estudio del Caso Estacionario. Ecuación de los Modos Lambda

Del capítulo 2 se recordará que el problema a resolver es la ecuación de los modos Lambda representada por

$$\mathcal{L}\Phi = \lambda \mathcal{M}\Phi,$$

la cual, después de aplicar el método de discretización nodal, adquiere la forma de un problema de valores propios generalizado, donde el problema es encontrar los valores propios y vectores propios de:

$$L\psi_n = \frac{1}{k_n} M\psi_n, \tag{6.2}$$

donde L y M son matrices de dimensión $2N$ para el caso de dos grupos de energía ($G = 2$) y que puede expresarse como un problema algebraico de autovalores generalizado con la siguiente estructura a bloques:

$$\begin{bmatrix} L_{11} & 0 \\ -L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \psi_{1_n} \\ \psi_{2_n} \end{bmatrix} = \frac{1}{k_n} \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi_{1_n} \\ \psi_{2_n} \end{bmatrix}. \tag{6.3}$$

Como ya se ha visto en el capítulo 2, una de las estrategias ha consistido en reducir el problema de valores propios generalizado a uno estándar, en donde la matriz A adquiere la siguiente forma,

$$A = L_{11}^{-1}(M_{11} + M_{12}L_{22}^{-1}L_{21}).$$

En función de las suposiciones realizadas anteriormente (hipótesis H1, H2 y H3), la matriz A adquiere formas distintas en el caso de manejar $G > 2$ grupos de energía.

En el caso del problema de los modos Lambda, la forma general de la matriz A, estará definida por la siguiente expresión

$$A = L_{11}^{-1}[M_{11}f(1) + M_{12}f(2) + M_{13}f(3) + \ldots + M_{1G}f(G)] \tag{6.4}$$

en donde la forma de la expresión $f(i)$, dependerá de las hipotésis establecidas y que para el caso estacionario se explican a continuación.

6.3.1. Análisis del Planteamiento con la Hipótesis H1

Considerando que existe trasvase de energía del grupo g_i al $g_{i+1}, \ldots, g_G$ ($i = 1, 2, ..., G$), se tendría que para un número G arbitrario de grupos, el esquema general a bloques L en la expresión (6.2) sería el mostrado por (6.5)

$$L = \begin{bmatrix} L_{11} & 0 & \cdots & 0 \\ -L_{21} & L_{22} & \cdots & 0 \\ -L_{31} & -L_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -L_{G,1} & -L_{G,2} & \cdots & L_{G,G} \end{bmatrix}, \tag{6.5}$$

en tanto que, la matriz M tendría la forma de la expresión en (6.1), ya comentada anteriormente, es decir,

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1G} \\ 0 & \cdots & \cdots & 0 \\ 0 & & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}.$$

Para este caso, la expresión $f(i)$ en la ecuación (6.4) puede definirse recursivamente así

$$f(i) = \begin{cases} f(i=1) = I \\ f(i>1) = L_{ii}^{-1}(\sum_{j=1}^{i-1} L_{ij}(f(j))). \end{cases} \tag{6.6}$$

La obtención de la matriz A para el problema estándar de valores propios con G grupos de energía, puede ser entonces derivada a partir de las expresiones (6.4) y (6.6).

Por ejemplo, la modelización con $G = 4$ grupos de energía, da origen a que la forma de la matriz A sea la siguiente

$$A = L_{11}^{-1}(M_{11}f(1) + M_{12}f(2) + M_{13}f(3) + M_{14}f(4)), \tag{6.7}$$

en donde los valores de las expresiones $f(i)$, de acuerdo a las formulaciones establecidas en (6.6), estarán dados por

$$\begin{aligned} f(1) =& I, \\ f(2) =& L_{22}^{-1}L_{21}, \\ f(3) =& L_{33}^{-1}(L_{31} + L_{32}(L_{22}^{-1}L_{21})), \\ f(4) =& L_{44}^{-1}(L_{41}f(1) + L_{42}f(2) + L_{43}f(3)) \\ =& L_{44}^{-1}(L_{41} + L_{42}(L_{22}^{-1}L_{21}) + L_{43}(L_{33}^{-1}(L_{31} + L_{32}(L_{22}^{-1}L_{21})))). \end{aligned}$$

Por lo que, la matriz A adoptaría la forma siguiente:

$$
\begin{aligned}
A = L_{11}^{-1}(&M_{11} \\
&+ M_{12}(L_{22}^{-1}L_{21}) \\
&+ M_{13}(L_{33}^{-1}(L_{31} + L_{32}L_{22}^{-1}L_{21})) \\
&+ M_{14}(L_{44}^{-1}(L_{41} + L_{42}(L_{22}^{-1}L_{21}) + L_{43}(L_{33}^{-1}(L_{31} + L_{32}L_{22}^{-1}L_{21}))))).
\end{aligned}
$$

6.3.2. Análisis del Planteamiento con la Hipótesis H2

Para el caso en donde se considera un trasvase de energía del grupo específico g_1 con $g_i, i = 2, \ldots, G$; y donde además se cumple que, para el resto de grupos con $g_{i\neq 1}$, este trasvase de energía está presente con el grupo inmediato siguiente g_{i+1} $(i = 2, 3, ..., G-1)$, se tendría que para un valor arbitrario G de grupos, la forma a bloques de L en la expresión (6.2) sería

$$
\begin{bmatrix}
L_{11} & 0 & 0 & \cdots & 0 & 0 \\
-L_{21} & L_{22} & 0 & \cdots & 0 & 0 \\
-L_{31} & -L_{32} & L_{33} & \ddots & \vdots & \vdots \\
-L_{41} & 0 & \ddots & \ddots & 0 & 0 \\
\vdots & \vdots & \ddots & -L_{G-1,G-2} & L_{G-1,G-1} & 0 \\
-L_{G,1} & 0 & \cdots & 0 & -L_{G,G-1} & L_{G,G}
\end{bmatrix}
$$

Con la forma a bloques de este planteamiento, las definiciones de las funciones $f(i)$ serían como sigue

$$
f(i) = \begin{cases} f(i=1) = I \\ f(i=2) = L_{ii}^{-1}(L_{i,1}f(i-1)) \\ f(i>2) = L_{ii}^{-1}(L_{i,1} + L_{i,i-1}f(i-1)). \end{cases} \tag{6.8}
$$

Así, en el caso de modelar con $G = 4$ grupos de energía, y tomando en cuenta las siguientes equivalencias para $f(i)$

$$
\begin{aligned}
f(1) =& I \\
f(2) =& L_{22}^{-1}(L_{21}) \\
f(3) =& L_{33}^{-1}(L_{31} + L_{32}(L_{22}^{-1}L_{21})) \\
f(4) =& L_{44}^{-1}(L_{41} + L_{43}(L_{33}^{-1}(L_{31} + L_{32}(L_{22}^{-1}L_{21})))),
\end{aligned}
$$

la obtención de la matriz A puede ser entonces derivada a partir de las expresiones (6.4) y (6.8) como sigue

$$
\begin{aligned}
A = L_{11}^{-1}(&M_{11} \\
&+ M_{12}(L_{22}^{-1}L_{21}) \\
&+ M_{13}(L_{33}^{-1}(L_{31} + L_{32}(L_{22}^{-1}L_{21}))) \\
&+ M_{14}(L_{44}^{-1}(L_{41} + L_{43}(L_{33}^{-1}(L_{31} + L_{32}(L_{22}^{-1}L_{21})))))).
\end{aligned}
$$

6.3.3. Análisis del Planteamiento con la Hipótesis H3

Cuando el trasvase de energía existe única y exclusivamente de un grupo cualquiera g_i, con el grupo inmediato siguiente g_{i+1} con $i = 1, 2, ..., G-1$, la forma a bloques de L en el problema de valores propios estándar con G grupos de energía representado en la expresión (6.2) sería de la siguiente forma

$$
\begin{bmatrix}
L_{11} & 0 & 0 & \cdots & 0 & 0 \\
-L_{21} & L_{22} & 0 & \cdots & 0 & 0 \\
0 & -L_{32} & L_{33} & \ddots & \vdots & \vdots \\
0 & 0 & \ddots & \ddots & 0 & 0 \\
\vdots & \vdots & \ddots & -L_{G-1,G-2} & \ddots & 0 \\
0 & 0 & \cdots & 0 & -L_{G,G-1} & L_{G,G}
\end{bmatrix}
$$

y las expresiones $f(i)$ se definirían así:

$$
f(i) = \begin{cases} f(i=1) = I \\ f(i>1) = L_{ii}^{-1}(L_{i,i-1}f(i-1)). \end{cases} \tag{6.9}
$$

En el caso de $G = 4$ grupos de energía y tomando en cuenta las definiciones de $f(i)$ en la expresión (6.9), se tendrían las siguientes equivalencias

$$
\begin{aligned}
f(1) &= I \\
f(2) &= L_{22}^{-1}L_{21} \\
f(3) &= L_{33}^{-1}(L_{32}(L_{22}^{-1}L_{21})) \\
f(4) &= L_{44}^{-1}(L_{43}(L_{33}^{-1}(L_{32}(L_{22}^{-1}L_{21}))))
\end{aligned}
$$

las cuales, una vez sustituidas en la ecuación (6.4) darían origen a

$$
\begin{aligned}
A = L_{11}^{-1}(&M_{11} \\
&+ M_{12}(L_{22}^{-1}L_{21}) \\
&+ M_{13}(L_{33}^{-1}(L_{32}(L_{22}^{-1}L_{21})) \\
&+ M_{14}(L_{44}^{-1}(L_{43}(L_{33}^{-1}(L_{32}(L_{22}^{-1}L_{21})))))).
\end{aligned} \tag{6.10}
$$

Las hipótesis planteadas en la sección 6.2 acerca de la estructura de las matrices en la expresión (6.2), han dado origen a que, en la descripción de la forma de la matriz A del problema estándar de valores propios, existan relaciones tanto de recurrencia como de dependencia en los cálculos. No obstante lo anterior, cabe recordar que para el caso estacionario, la resolución de la ecuación de los modos Lambda representada por la expresión (6.2) puede calcularse, por ejemplo, con el método de Arnoldi, cuya operación más costosa esta representada por un producto matriz–vector, y es sobre este producto que se revisarán algunas estrategias de paralelismo que a continuación se explican para la hipótesis H3.

6.3.4. Estrategias de Paralelización bajo la Hipótesis H3

Este apartado desarrolla una serie de análisis que podrían utilizarse para aplicar paralelismo sobre la multiplicación matriz–vector, operación medular para resolver el planteamiento hipotético H3 utilizando $G = 4$ grupos de energía. Análisis parecidos pueden realizarse para las hipótesis H1 y H2.

6.3.4.1. Identificación de Operaciones

Partiendo del caso con $G = 4$ grupos de energía representado por la expresión (6.10) y eliminando los paréntesis, se obtiene la expresión siguiente

$$A = L_{11}^{-1}(M_{11} + M_{12}L_{22}^{-1}L_{21} + M_{13}L_{33}^{-1}L_{32}L_{22}^{-1}L_{21} + M_{14}L_{44}^{-1}L_{43}L_{33}^{-1}L_{32}L_{22}^{-1}L_{21}),$$

misma que al factorizar términos comunes se expresa como

$$A = L_{11}^{-1}(M_{11} + (M_{12} + (M_{13} + M_{14}L_{44}^{-1}L_{43})L_{33}^{-1}L_{32})L_{22}^{-1}L_{21}). \tag{6.11}$$

Puesto que se busca calcular el producto $Ax = y$, las primeras operaciones que se pueden realizar al multiplicar por x en (6.11) son el cálculo de los vectores w_1, w_2 y w_3 como se ve en la siguiente expresión

$$Ax = L_{11}^{-1}(\underbrace{M_{11}x}_{w_3} + (M_{12} + (M_{13} + M_{14}L_{44}^{-1}L_{43})L_{33}^{-1}L_{32})\underbrace{L_{22}^{-1}\underbrace{L_{21}x}_{w_1}}_{w2}), \tag{6.12}$$

en donde los vectores w_1 y w_3 se calculan con una multiplicación matriz diagonal por vector y el vector w_2 con la resolución del sistema $L_{22}w_2 = w_1$. Una vez calculado el vector w_2, la expresión (6.12) se puede reescribir como

$$Ax = L_{11}^{-1}(w_3 + (M_{12}w_2 + (M_{13} + M_{14}L_{44}^{-1}L_{43})L_{33}^{-1}L_{32}w_2)) \tag{6.13}$$

con lo que puede calcularse w_6 (con un producto matriz diagonal por vector) y w_5 (con la resolución de un sistema de ecuaciones), como se ve a continuación

$$Ax = L_{11}^{-1}(w_3 + (\underbrace{M_{12}w_2}_{w_6} + (M_{13} + M_{14}L_{44}^{-1}L_{43})\underbrace{L_{33}^{-1}\underbrace{L_{32}w_2}_{w_4}}_{w_5})). \tag{6.14}$$

Procediendo de igual forma y habiendo calculado w_5, se pueden calcular w_{10} y w_9 escribiendo (6.18) como

$$Ax = L_{11}^{-1}(w_3 + (w_6 + (\underbrace{M_{13}w_5}_{w_{10}} + \underbrace{M_{14}\underbrace{L_{44}^{-1}\underbrace{L_{43}w_5}_{w_7}}_{w_8}}_{w_9}))). \tag{6.15}$$

Así, las últimas operaciones por realizar en (6.15) son: 3 sumas de vectores y la resolución de un sistema con L_{11} como se indica a continuación:

$$\begin{aligned} Ax &= L_{11}^{-1}(w_3 + (w_6 + (w_{10} + w_9))) = y & (6.16)\\ &= L_{11}^{-1}w_\sigma = y. & (6.17) \end{aligned}$$

El siguiente apartado presenta un estudio de la búsqueda de operaciones que puedan realizarse en paralelo.

6.3.4.2. Grafo de Dependencias

De acuerdo al desarrollo visto de (6.11) a (6.17), puede describirse un grafo de dependencias y distribución de datos como el que se muestra en la figura 6.1, en donde se tienen los siguientes elementos:

Grupos de procesadores. Esta columna representa la existencia de dos grupos de procesadores G_0 y G_1, donde G_0 está formado por p_{G0} procesadores y G_1 por p_{G1} procesadores.

Distribución de Datos. Esta columna muestra la forma en la que las distintas matrices involucradas en el problema deben estar distribuidas en cada uno de los grupos. Por ejemplo, en la figura 6.1, el grupo de procesadores G_0 necesitará operar con los bloques de matrices M_{11}, M_{12} y M_{13}; en tanto que el grupo G_1 necesitará operar con los bloques $L_{22}, L_{21}, L_{33}, L_{32}, L_{44}, L_{43}, M_{14}$. Por otra parte, el bloque L_{11} necesitará estar distribuido en ambos grupos G_0 y G_1.

Etapas de paralelismo. En cada etapa marcada se indican las operaciones que pueden realizarse en paralelo (que en el caso particular que se está analizando: H3 y $G = 4$, se

tienen 7 etapas). Las etapas se corresponden con los niveles del grafo. Por ejemplo, en la fila que corresponde a la etapa 1, se realizan simultáneamente una operación matriz diagonal por vector en G_0 ($M_{11}x$), y una operación matriz diagonal por vector y la resolución de un sistema de ecuaciones lineales ($L_{22}^{-1}L_{21}x$) en G_1.

Grafo de dependencias. Las etapas muestran el grafo de dependencias de las operaciones, indicando cuáles se realizan en cada grupo de procesadores y los resultados obtenidos después de operar (etiquetas de los arcos del grafo). Para cada nodo del grafo se indica, en la parte superior, el grupo de procesadores que realiza la operación; mientras que en la parte inferior, la operación propiamente dicha. Con los arcos del grafo se pueden detectar comunicaciones de datos entre grupos de procesadores. Por ejemplo, en el nivel 2 del grafo, el grupo G_1 debe enviar al grupo G_0 el vector w_2 calculado en el nivel 1.

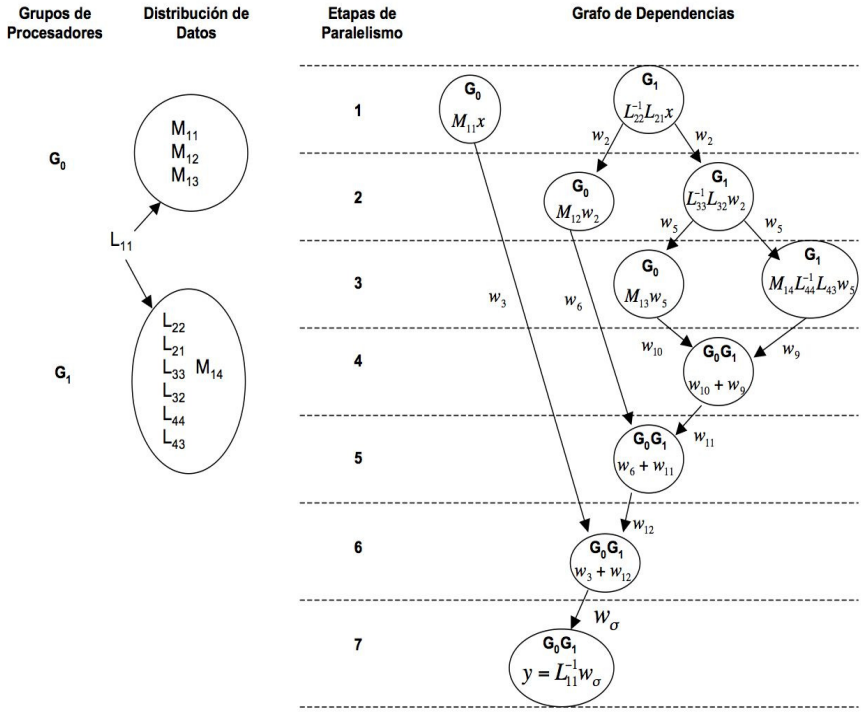

Figura 6.1: Grafo de dependencias y distribución de datos con la hipótesis H3 (Alternativa 1).

Si se considera que las operaciones del grafo de la figura 6.1 se ejecutarán en un computador paralelo de memoria distribuida homogéneo (en donde las unidades de procesamiento tienen las mismas características), podrían presentarse tiempos ociosos de

procesamiento. Por ejemplo, para la etapa paralela 1, el grupo G_0 realizaría una operación matriz diagonal por vector, mientras que el grupo G_1 realizaría también una operación matriz diagonal vector, pero emplearía tiempo de ejecución adicional para resolver un sistema de ecuaciones lineales disperso, este tiempo adicional representaría tiempo ocioso para el grupo G_0, ya que no podría continuar con sus cálculos de la etapa paralela 2 por la dependencia de los resultados del grupo G_1. En cambio, si se utilizara un computador paralelo heterogéneo (donde las características de arquitectura y velocidad de las unidades de procesamiento pueden variar), se podrían agrupar los procesadores más lentos en G_0 (que de acuerdo al grafo es el grupo que hace las operaciones menos costosas y el que debe esperar resultados del otro grupo) y se podrían agrupar los procesadores más rápidos en G_1, con lo que podría disminuirse los tiempos ociosos. Esta última opción sería más adecuada si se busca eficiencia en el paralelismo.

Además de las comunicaciones de los vectores w_2 y w_5 de G_1 a G_0, en los niveles 4, 5, 6 y 7 del grafo se especifica que ambos grupos de procesadores, G_0 y G_1, trabajen en las operaciones de suma de vectores (para evitar ociosidad), pero esto conlleva la redistribución de los vectores w_9, w_{10}, w_6 y w_3. En una plataforma paralela heterogénea, un estudio de la parametrización de ésta permitiría una elección más juiciosa entre no permitir la redistribución de los vectores en los niveles 4, 5, 6 y 7 (dejando a G_0 ocioso y a G_1 la carga completa de la suma de ellos) e involucrar a todos los procesadores en estos cómputos.

Si no se contemplara la posibilidad del paralelismo de la figura 6.1, entonces tendrían que realizarse las operaciones en un orden secuencial, sin embargo, no hay que olvidar que cada una de las operaciones matriz diagonal por vector y resolución de un sistema lineal disperso se realizan en paralelo, tal y como ha sucedido en las pruebas experimentales del caso estacionario con $G = 2$ grupos de energía y cuyo estudio se ha presentado en esta tesis. Entonces, ejecutar secuencialmente todas las operaciones para calcular $Ax = y$ llevaría a tener 10 etapas de procesamiento (que es el número de nodos del grafo), mientras que con la paralelización propuesta en la figura 6.1 el número de etapas se reducirían a un valor de 7, lo cual representa un ahorro de 3 pasos.

6.3.4.3. Grafo de Dependencias Alternativo

Existe un grafo alternativo al presentado en la figura 6.1 que surge a partir de reescribir la expresión (6.16), al ser la suma de vectores conmutativa y asociativa, como sigue

$$Ax = L_{11}^{-1}(\underbrace{\underbrace{\underbrace{(\underbrace{w_3 + w_6}_{w_a}) + w_{10}}_{w_b}) + w_9}_{w_c}) = y.$$

Este reordenamiento de la suma de vectores da origen al grafo de dependencias de la figura 6.2.

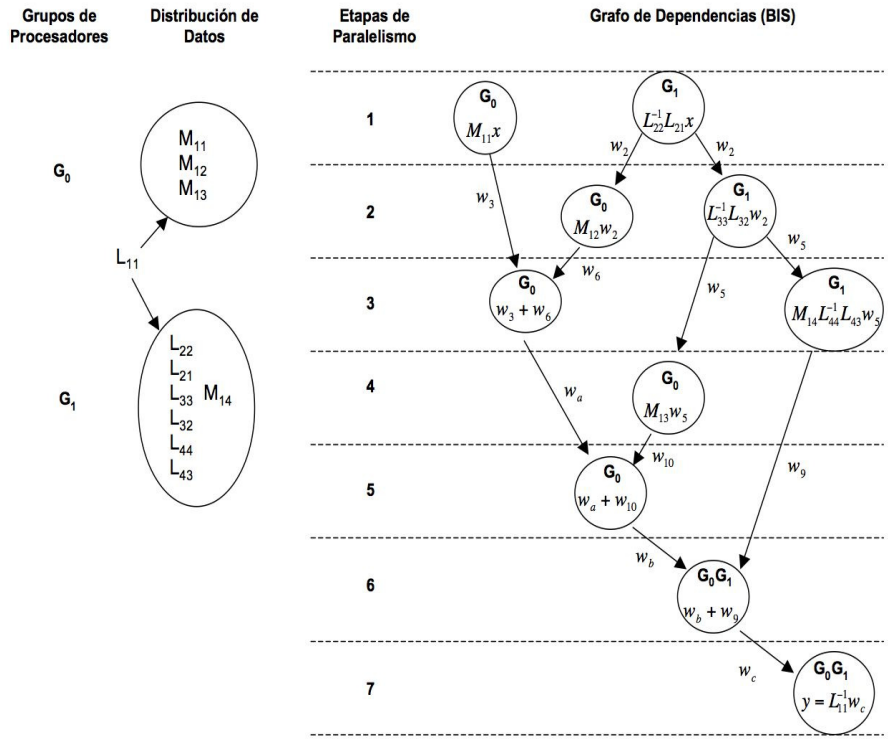

Figura 6.2: Grafo de dependencias y distribución de datos con la hipótesis H3 (Alternativa 2).

Comparando este nuevo grafo con el de la figura 6.1, se observa que, con el nuevo grafo se ahorran comunicaciones, pues aunque también se deben comunicar dos vectores (w_2 y w_5 de G_1 a G_0) sólo hace falta redistribuir dos vectores (w_b y w_9) para que ambos grupos de procesadores realicen la suma de vectores del nivel 6 y la resolución del sistema del nivel 7 (mientras que en grafo de la figura 6.1 se necesita redistribuir cuatro vectores).

6.4. Estudio del Caso Dinámico. Transitorios

En el capítulo 2 se ha establecido que para el estudio de transitorios en simulaciones de reactores nucleares, es necesario resolver, para cada paso de tiempo, un sistema de ecuaciones de la forma

$$T\psi = E, \tag{6.18}$$

en donde, para el caso de $G = 2$ grupos de energía se expresa como

$$\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}, \tag{6.19}$$

y donde el lado derecho de la ecuación depende de la solución en el paso de tiempo previo y del método de diferencias hacia atrás utilizado. También, como se ha establecido anteriormente, la matriz de coeficientes T del sistema comparte propiedades parecidas a las matrices L y M del caso estacionario, por lo que, debido a las condiciones de continuidad de los flujos y las corrientes neutrónicas interpuestas entre las celdas, los bloques T_{ii} $(i = 1, 2)$ son matrices simétricas definidas positivas, en tanto que el resto de los bloques son matrices diagonales que pueden o no, ser matrices singulares debido a las condiciones de los flujos neutrónicos impuestos entre las celdas en que se ha dividido el reactor.

Esta sección tiene como objetivo presentar la posible forma de la matriz T para el caso de la simulación de transitorios con G grupos de energía ($G > 2$), al igual que como se ha hecho para el caso de la ecuación de los modos Lambda (EDN caso estacionario). La forma de la matriz T se deducirá a partir de la forma que ha tomado la matriz L bajo las suposiciones H1, H2 y H3 presentadas en el apartado 6.2.

6.4.1. Análisis del Planteamiento con la Hipótesis H1

Partiendo del planteamiento H1 en donde se considera que existe trasvase de energía de un grupo g_i cualquiera con $g_{i+1}, \ldots, g_G$ $(i = 1, 2, ..., G-1)$ se tendría que, para un número G arbitrario de grupos, el esquema general a bloques de la matriz T en la expresión 6.18, sería

$$T = \begin{bmatrix} T_{11} & T_{12} & T_{13} & \cdots & T_{1n} \\ T_{21} & T_{22} & T_{23} & \cdots & T_{2n} \\ T_{31} & T_{32} & T_{33} & \cdots & T_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ T_{n1} & T_{n2} & T_{n3} & \cdots & T_{G,G} \end{bmatrix}. \tag{6.20}$$

En el bloque anterior representado por (6.20), las matrices diagonales T_{ii} $(i = 1, 2, ..., n)$ serían simétricas definidas positivas y el resto de las matrices son matrices diagonales que pueden ser o no singulares dependiendo de las condiciones de continuidad de los flujos neutrónicos impuestos al reactor.

Como se puede observar, bajo esta primera hipótesis, la matriz resultante T, sería una matriz dispersa a bloques de gran dimensión (ver patrón de dispersión de casos de estudio en capítulo 5), por lo que tratarla de manera total, como puede ser bajo una sola estructura de almacenamiento (tipo CSR por ejemplo), acarrearía muchos problemas en almacenamiento primario. Por otro lado, al ser la matriz T no simétrica, puede

traer como consecuencia problemas de convergencia para su resolución. Un algoritmo multipaso, como el expuesto en este trabajo de tesis, sería de gran ayuda para hacer frente a ambos problemas.

6.4.2. Análisis del Planteamiento con la Hipótesis H2

Para el caso en donde se considera un trasvase de energía del grupo g_1 con $g_i, i = 2, \ldots, G$; y de los grupos $g_{i\neq 1}$ con el grupo inmediato siguiente g_{i+1} $(i = 2, 3, ..., G-1)$, se tendría que la forma a bloques de la matriz T en (6.18) sería

$$T = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} & \cdots & T_{1n} \\ T_{21} & T_{22} & T_{23} & 0 & \cdots & 0 \\ T_{31} & T_{32} & T_{33} & \ddots & \ddots & \vdots \\ T_{41} & 0 & \ddots & \ddots & T_{G-2,G-1} & 0 \\ \vdots & \vdots & \ddots & T_{G-1,G-2} & \ddots & T_{G-1,G} \\ T_{n1} & 0 & \cdots & 0 & T_{G,G-1} & T_{G,G} \end{bmatrix}.$$

En este caso, T resulta ser una matriz dispersa de gran dimensión a bloques y en forma de flecha apuntando hacia arriba y a la izquierda, en donde, al igual que en el caso anterior, las matrices T_{ii} $(i = 1, 2, ..., G)$ son matrices simétricas definidas positivas y el resto de las matrices son diagonales, que pueden o no, ser singulares.

6.4.3. Análisis del Planteamiento con la Hipótesis H3

Cuando el trasvase de energía que existe de un grupo cualquiera g_i única y exclusivamente con el grupo inmediato siguiente g_{i+1} $(i = 1, 2, ..., G-1)$, la forma a bloques de T sería

$$T = \begin{bmatrix} T_{11} & T_{12} & 0 & 0 & \cdots & 0 \\ T_{21} & T_{22} & T_{23} & 0 & \cdots & 0 \\ 0 & T_{32} & T_{33} & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & T_{G-2,G-1} & 0 \\ \vdots & \vdots & \ddots & T_{G-1,G-2} & \ddots & T_{G-1,G} \\ 0 & 0 & \cdots & 0 & T_{G,G-1} & T_{G,G} \end{bmatrix}. \tag{6.21}$$

Para esta hipótesis, la forma de la matriz T es tridiagonal a bloques, con las mismas características de los bloques con las hipótesis H1 y H2.

El siguiente apartado muestra el diseño de un algoritmo multipaso para el caso de la matriz T obtenida en (6.21).

6.4.4. Diseño de un Método Multipaso en la Hipótesis H3

En este apartado se muestran algoritmos iterativos multipaso basados en la iteración de Jacobi y Gauss–Seidel acelerado para el caso de $G = 4$ grupos de energía bajo la hipótesis H3, así como también algunas estrategias de la paralelización con dichos métodos analizando tanto sus ventajas como desventajas.

6.4.4.1. Método Multipaso en la Hipótesis H3 basado en Jacobi

El algoritmo 23 muestra un método multipaso basado en la iteración de Jacobi para el planteamiento H3 del caso dinámico de la EDN con $G = 4$ grupos de energía.

Algoritmo 23 Método Multipaso Jacobi para el caso de la Hipótesis H3 con 4 Parámetros de Relajación (MMJ4H3)

(* ψ_i^* soluciones de un paso previo *)
01 $\psi_1^0 := \psi_1^*$;
02 $\psi_2^0 := \psi_2^*$;
03 $\psi_3^0 := \psi_3^*$;
04 $\psi_4^0 := \psi_4^*$;

(* Resolver para $\psi_i^1, i = 1, \ldots, 4$ *)
05 $T_{11}\psi_1^1 = e_1 - T_{12}\psi_2^0$
06 $T_{22}\psi_2^1 = e_2 - (T_{21}\psi_1^0 + T_{23}\psi_3^0)$
07 $T_{33}\psi_3^1 = e_3 - (T_{32}\psi_2^0 + T_{34}\psi_4^0)$
08 $T_{44}\psi_4^1 = e_4 - T_{43}\psi_3^0$

(* Ciclo externo *)
09 **do** l=1, 2, 3, ...
10 $T_{11}\psi_1^{l+1} = e_1 - T_{12}(\omega_1\psi_2^l + (1-\omega_1)\psi_2^{l-1})$;
11 $T_{22}\psi_2^{l+1} = e_2 - (T_{21}(\omega_2\psi_1^l + (1-\omega_2)\psi_1^{l-1}) + T_{23}(\omega_2\psi_3^l + (1-\omega_2)\psi_3^{l-1}))$;
12 $T_{33}\psi_3^{l+1} = e_3 - (T_{32}(\omega_3\psi_2^l + (1-\omega_3)\psi_2^{l-1}) + T_{34}(\omega_3\psi_4^l + (1-\omega_3)\psi_4^{l-1}))$;
13 $T_{44}\psi_4^{l+1} = e_4 - T_{43}(\omega_4\psi_3^l + (1-\omega_4)\psi_3^{l-1})$;

(* Detectar convergencia *)
14 **until** $\|\psi_i^{l+1} - \psi_i^l\| < tol$ **for** $i = 1, \ldots, 4$

Al igual que los algoritmos multipaso propuestos en esta tesis (ver capítulo 5), se hace necesario disponer de dos soluciones consecutivas del flujo neutrónico ψ, es decir ψ^0 y ψ^1 (esta operación se indica en las líneas 1 a 8 del algoritmo). Después, en el ciclo externo señalado por las líneas 9 a 14, se realiza una serie de cómputos que involucran flujos de la etapa l y $l-1$, seguidas de la resolución de los sistemas con los bloques T_{ii} $(i = 1, \ldots, 4)$. Por último, se lleva a cabo una prueba de convergencia (línea 14).

Como se puede observar, las líneas 9 a 14 involucran operaciones extras que no

existen en los métodos multipaso para el caso con $G = 2$ grupos de energía ya descritos en el capítulo 5, esto se debe a la estructura a bloques de la matriz T bajo la hipótesis H3 cuando se tienen más grupos de energía. En otras palabras, en los algoritmos para $G = 2$ grupos de energía, solamente se tenía una suma de vectores y un producto matriz diagonal por vector (como la operación del lado derecho en la línea 10 del algoritmo 23); con la aparición de más grupos de energía, ahora se tienen operaciones que contemplan 2 sumas de vectores y 2 productos matriz diagonal por vector (como la operación de la línea 11 del algoritmo 23), lo que significa el doble de operaciones de punto flotante.

La paralelización del algoritmo 23 resulta directa ya que no existen dependencias en los cálculos de las líneas 10 a 13. Entonces, considerando dos grupos de procesadores G_0 y G_1, se pueden realizar las siguientes operaciones:

1. En paralelo [Etapa de cálculo]:
   - El grupo de procesadores G_0 realizará las líneas 10 y 11.
   - El grupo de procesadores G_1 realizará las líneas 12 y 13.
2. En paralelo [Etapa de comunicación]
   - Los grupos de procesadores envían y reciben los resultados de la etapa $l+1$ recién calculada.

Con la estrategia anterior se logra una carga equilibrada tanto de datos como de operaciones. Como se observa, esta estrategia es completamente eficiente desde el punto de vista del paralelismo.

6.4.4.2. Método Multipaso en la Hipótesis H3 basado en Gauss–Seidel

El algoritmo 24 muestra un método multipaso tipo Gauss–Seidel acelerado para el planteamiento H3 del caso dinámico de la EDN con $G = 4$ grupos de energía.

Para el algoritmo 24 pueden hacerse los mismos comentarios que para el algoritmo multipaso basado en Jacobi (algoritmo 23) con respecto a las etapas de inicialización de los flujos neutrónicos, y de la cantidad de cálculos que se debe realizar, excepto que ahora se presentan dependencias en los cálculos de las líneas 10 a 13.

Para reducir el número de cálculos del algoritmo 24 se pueden combinar técnicas multipaso con técnicas de un sólo paso para reescribir las líneas 11 y 12 como sigue:

$$11 \qquad T_{22}\psi_2^{l+1} = e_2 - (T_{21}(\underbrace{\omega_2\psi_1^{l+1} + (1-\omega_2)\psi_1^l}_{\text{2 pasos}}) + \underbrace{T_{23}\psi_3^l}_{\text{1 paso}});$$

$$12 \qquad T_{33}\psi_3^{l+1} = e_3 - (T_{32}(\underbrace{\omega_3\psi_2^{l+1} + (1-\omega_3)\psi_2^l}_{\text{2 pasos}}) + \underbrace{T_{34}\psi_4^l}_{\text{1 paso}}),$$

Algoritmo 24 Método Multipaso Gauss–Seidel Acelerado para el caso de la Hipótesis H3 con 4 Parámetros de Relajación (MMGS4H3)

(* ψ_i^* soluciones de un paso previo *)
01 $\psi_1^0 := \psi_1^*$;
02 $\psi_2^0 := \psi_2^*$;
03 $\psi_3^0 := \psi_3^*$;
04 $\psi_4^0 := \psi_4^*$;

(* Resolver para $\psi_i^1, i = 1, \ldots, 4$ *)
05 $T_{11}\psi_1^1 = e_1 - T_{12}\psi_2^0$
06 $T_{22}\psi_2^1 = e_2 - (T_{21}\psi_1^0 + T_{23}\psi_3^0)$
07 $T_{33}\psi_3^1 = e_3 - (T_{32}\psi_2^0 + T_{34}\psi_4^0)$
08 $T_{44}\psi_4^1 = e_4 - T_{43}\psi_3^0$

(* Ciclo externo *)
09 **do** l=1, 2, 3, ...
10 $T_{11}\psi_1^{l+1} = e_1 - T_{12}(\omega_1\psi_2^l + (1-\omega_1)\psi_2^{l-1})$;
11 $T_{22}\psi_2^{l+1} = e_2 - (T_{21}(\omega_2\psi_1^{l+1} + (1-\omega_2)\psi_1^l) + T_{23}(\omega_2\psi_3^l + (1-\omega_2)\psi_3^{l-1}))$;
12 $T_{33}\psi_3^{l+1} = e_3 - (T_{32}(\omega_3\psi_2^{l+1} + (1-\omega_3)\psi_2^l) + T_{34}(\omega_3\psi_4^l + (1-\omega_3)\psi_4^{l-1}))$;
13 $T_{44}\psi_4^{l+1} = e_4 - T_{43}(\omega_4\psi_3^{l+1} + (1-\omega_4)\psi_3^l)$;

(* Detectar convergencia *)
14 **until** $\|\psi_i^{l+1} - \psi_i^l\| < tol$ **for** $i = 1, \ldots, 4$

con esto, se consigue eliminar 2 sumas de vectores escalados.

Respecto de la paralelización del algoritmo 24 con las líneas 11 y 12 reescritas, una primera propuesta (presentada en la figura 6.3) consiste en utilizar dos grupos de procesadores (G_0 y G_1) donde la carga de los datos está equilibrada, con lo que se evita que un grupo de procesadores se vea limitado en sus recursos de almacenamiento primario. Esta distribución es necesaria, puesto que se propone que el grupo G_0 realice las operaciones de las líneas 10 y 11 del algoritmo 24, por lo que necesitará los datos indicados por la figura 6.3. En tanto que el grupo de procesadores G_1 tendrá que realizar las operaciones de las líneas 12 y 13 del algoritmo, indicadas también en la misma figura.

Bajo este esquema, en primera instancia se resuelven en paralelo los sistemas para ψ_1^{l+1} y ψ_3^{l+1}, donde el cálculo de ψ_3^{l+1} utiliza ψ_2^{l+1} y ψ_2^l, así como ψ_4^l, de acuerdo al algoritmo 24 y las nuevas líneas 11 y 12; pero esto ya no será posible, dado que aún no se cuenta con ψ_2^{l+1}, por lo que el cálculo de ψ_3^{l+1} debe utilizar ψ_2^l y ψ_2^{l-1}, además de ψ_4^l. Sin embargo, esto se ve compensado con la resolución del sistema para ψ_2^{l+1}, ya que podrá utilizar el último valor calculado de ψ_3^{l+1}, a diferencia de como se tenía en el

algoritmo 24. Entonces, las líneas 10 a 13 del algoritmo 24 quedarían como sigue

Calcular en paralelo líneas 10 (G_0) y 12 (G_1)

10 $T_{11}\psi_1^{l+1} = e_1 - T_{12}(\omega_1\psi_2^l + (1-\omega_1)\psi_2^{l-1})$

12 $T_{33}\psi_3^{l+1} = e_3 - (T_{32}(\omega_3\psi_2^l + (1-\omega_3)\psi_2^{l-1}) + T_{34}\psi_4^l)$

Comunicar ψ_3 de G_1 a G_0

Calcular en paralelo líneas 11 (G_0) y 13 (G_1)

11 $T_{22}\psi_2^{l+1} = e_2 - (T_{21}(\omega_2\psi_1^{l+1} + (1-\omega_2)\psi_1^l) + T_{23}\psi_3^{l+1})$

13 $T_{44}\psi_4^{l+1} = e_4 - T_{43}(\omega_4\psi_3^{l+1} + (1-\omega_4)\psi_3^l)$

Comunicar ψ_2 de G_0 a G_1.

G_0	G_1
Datos $T_{11}, T_{22}, T_{12}, T_{21}, T_{23}$ ψ_1^{l+1}, ψ_1^l ψ_2^{l+1}, ψ_2^l $\psi_3^{l+1}, \psi_3^l, e_1, e_2, \omega_1, \omega_2$	**Datos** $T_{33}, T_{44}, T_{32}, T_{34}, T_{43}$ ψ_3^{l+1}, ψ_3^l ψ_2^{l+1}, ψ_2^l $\psi_4^{l+1}, \psi_4^l, e_3, e_4, \omega_3, \omega_4$
Operaciones 1. Calcula ψ_1^{l+1} (línea 10) 2. Recibe ψ_3^{l+1} 3. Calcula ψ_2^{l+1} (línea 11) 4. Envía ψ_2^{l+1}	**Operaciones** 1. Calcula ψ_3^{l+1} (línea 12) 2. Envía ψ_3^{l+1} 3. Calcula ψ_4^{l+1} (línea 13) 4. Recibe ψ_2^{l+1}

Figura 6.3: Distribución de datos y procesamiento paralelo en MMGS4H3 versión 1

Analizando esta propuesta en la figura 6.3, el grupo de procesadores en G_0 terminará de ejecutar la línea 10 del algoritmo 24 antes de que G_1 termine la línea 12 (pues la línea 12 involucra más operaciones que la línea 10). Entonces, G_0 podría tener un tiempo ocioso mientras espera que G_1 le envíe datos necesarios para ejecutar la línea 11. Pasada la etapa de comunicaciones se podrían ejecutar en paralelo las líneas 11 y 13 del algoritmo, las que utilizarían las ψ_1 y ψ_3 recientemente calculadas. Un problema de ociosidad podría darse también después del cálculo de las líneas 11 y 13 del algoritmo, ya que la línea 11 contiene más operaciones que la 13. Sin embargo, bajo este esquema

se preserva la ventaja de calcular ψ_2 y ψ_4 con los valores más actualizados de ψ_1 y ψ_3, favoreciendo la aceleración de la convergencia del algoritmo.

A continuación se presenta otro posible esquema paralelo en la figura 6.4, en donde las operaciones se realizan en otro orden.

G_0	G_1
Datos $T_{11}, T_{22}, T_{12}, T_{21}, T_{23}$ ψ_1^{l+1}, ψ_1^l ψ_2^{l+1}, ψ_2^l $\psi_3^{l+1}, \psi_3^l, e_1, e_2, \omega_1, \omega_2$	**Datos** $T_{33}, T_{44}, T_{32}, T_{34}, T_{43}$ ψ_3^{l+1}, ψ_3^l ψ_2^{l+1}, ψ_2^l $\psi_4^{l+1}, \psi_4^l, e_3, e_4, \omega_3, \omega_4$
Operaciones	**Operaciones**
1. Calcula ψ_1^{l+1} (línea 10)	1. Calcula ψ_4^{l+1} (línea 13)
2. Calcula ψ_2^{l+1} (línea 11)	2. Calcula ψ_3^{l+1} (línea 12)
3. Recibe ψ_3^{l+1}	3. Envía ψ_3^{l+1}
4. Envía ψ_2^{l+1}	4. Recibe ψ_2^{l+1}

Figura 6.4: Distribución de datos y procesamiento paralelo en MMGS4H3 versión 2

En este nuevo esquema, las operaciones de las líneas 10 y 13 del algoritmo 24 se realizan simultáneamente en cada grupo de procesadores G_0 y G_1, respectivamente. Posteriormente, en paralelo se calculan las líneas 11 y 12 del algoritmo. Por último, se tiene una etapa de intercambio de ψ_2 y ψ_3. Respecto del esquema anterior (figura 6.3) este esquema tiene la ventaja de que el tiempo de ocio de los procesadores puede disminuir, ya que en cada etapa paralela el número de operaciones que debe realizar cada grupo de procesadores es el mismo. En este nuevo esquema ψ_4^{l+1} no puede calcularse con el valor más reciente de ψ_3^{l+1}, ni ψ_3^{l+1} con el valor más reciente de ψ_2^{l+1}, como se tenía en el algoritmo 24; sin embargo, a diferencia también del mismo algoritmo, ψ_3 puede echar mano de ψ_4^{l+1}, además de que ψ_2^{l+1} utiliza ψ_1^{l+1}, favoreciendo, en este sentido, la convergencia del algoritmo. Entonces, las líneas 10 a 12 del algoritmo quedarían como

sigue

Calcular en paralelo líneas 10 (G_0) y 13 (G_1)

10 $T_{11}\psi_1^{l+1} = e_1 - T_{12}(\omega_1\psi_2^l + (1-\omega_1)\psi_2^{l-1})$

13 $T_{44}\psi_4^{l+1} = e_4 - T_{43}(\omega_4\psi_3^l + (1-\omega_4)\psi_3^{l-1})$

Calcular en paralelo líneas 11 (G_0) y 12 (G_1)

11 $T_{22}\psi_2^{l+1} = e_2 - (T_{21}(\omega_2\psi_1^{l+1} + (1-\omega_2)\psi_1^l) + T_{23}\psi_3^l)$

12 $T_{33}\psi_3^{l+1} = e_3 - (T_{32}(\omega_3\psi_2^l + (1-\omega_3)\psi_2^{l-1}) + T_{34}\psi_4^{l+1})$

Comunicar ψ_3 de G_1 a G_0

Comunicar ψ_2 de G_0 a G_1.

6.5. Conclusiones

Como se ha podido observar, existen diversas alternativas de paralelización para cada uno de los problemas computacionales planteados por los problemas estáticos y dinámicos en la EDN modelada con más de dos grupos de energía. En particular, se han presentado 3 escenarios distintos bajo los cuales se puede predecir la estructura que tendrán las matrices en la modelización de la EDN multigrupo. Se han analizado las estrategias de paralelización para uno de los 3 planteamientos, tanto para el caso estacionario como dinámico, manejando G =4 grupos de energía, mencionando sus ventajas y desventajas. Se ha visto también que para el caso del problema estacionario (ecuación de los modos Lambda) modelado con varios grupos de energía, existe ciertas recurrencias y dependencias que hacen difícil la obtención de paralelismo para resolver el problema; sin embargo, puede obtenerse cierto nivel de paralelismo a partir de algunos reordenamientos de operaciones. Por otro lado, en lo que respecta al problema dinámico de la EDN multigrupo, existen distintas posibilidades que pueden considerarse a la hora de diseñar un método iterativo multipaso. En este capítulo se han delineado algunas estrategias de paralelización con una de las tres hipótesis expuestas y con un enfoque basado en la iteración de Jacobi y en Gauss–Seidel acelerado con distintos parámetros de relajación. Como se ha podido observar, la versión basada en Jacobi presenta un grado de paralelismo mayor que la versión basada en Gauss–Seidel.

Por otro lado, los estudios expuestos en este capítulo pueden extenderse para construir métodos iterativos multipaso como los presentados en el capítulo 5, mismos que presentarán tanto ventajas como desventajas, como ha podido verse para los ejemplos mostrados. El establecimiento de distintas estrategias es fundamental para identificar aquellas áreas en donde pudieran aplicarse técnicas paralelas.

Capítulo 7

Conclusiones Finales y Trabajos Futuros

La resolución de la ecuación de difusión neutrónica multigrupo caso 3D es crucial para el estudio del comportamiento de un reactor nuclear. Se ha visto que la discretización de esta ecuación da origen a dos problemas fundamentales.

Primero, en el caso estacionario se tiene el problema de resolver la ecuación de los modos Lambda, para lo cual existen métodos como Arnoldi, cuya operación nuclear es la multiplicación matriz–vector. Dado que se han considerado dos grupos de energía y por la cantidad de nodos en que se ha llevado a cabo la discretización del caso de estudio presentado en esta tesis, la matriz con la que se debe realizar esta multiplicación matriz–vector no está en forma explícita, por lo que un producto matriz–vector debe calcularse mediante una sucesión de productos matriz diagonal por vector y la resolución de sistemas de ecuaciones lineales dispersos de gran dimensión. La resolución eficiente de estos sistemas es uno de los problemas fundamentales que se han resuelto en esta tesis.

Segundo, en el caso dinámico, el resultado de la discretización temporal, da lugar a un sistema de ecuaciones lineales disperso de gran dimensión no simétrico, que hay que resolver para cada paso de tiempo durante la simulación de un transitorio en un reactor nuclear. La resolución eficiente de este sistema es el segundo problema fundamental que se ha resuelto en esta tesis.

Los resultados de las investigaciones realizadas en esta tesis para resolver los problemas planteados dan lugar a las siguientes conclusiones.

⋆ Se han aplicado diversas librerías numéricas de libre distribución, tanto secuenciales como paralelas para resolver los sistemas de ecuaciones lineales dispersos de gran dimensión que surgen de la discretización espacial y temporal de la ecuación de difusión

neutrónica multigrupo caso 3D.

⋆ Todos los algoritmos mostrados en esta memoria se han programado y aplicado a una batería de matrices provenientes de dos casos de estudio reales, que corresponden a los reactores nucleares comerciales de Ringhals-I y Leibstadt, en donde los resultados experimentales muestran las ventajas del uso de la computación paralela y distribuida para reducir los tiempos de ejecución en la resolución de los sistemas de ecuaciones lineales dispersos, asociados al problema estacionario y dinámico de la ecuación de la difusión neutrónica multigrupo.

⋆ En el problema estacionario (representado por la ecuación de los modos Lambda) de la EDN para el reactor Ringhals I, se ha encontrado que los métodos más eficientes para resolver los sistemas de ecuaciones lineales dispersos están basados en subespacios de Krylov como el Gradiente Conjugado (GC) y Gradiente Biconjugado estabilizado (BCGSTAB) para el caso de la librería SPARSKIT. En el caso paralelo, respecto de la librería PETSc, se ha encontrado que el método del Gradiente Conjugado ha sido el más eficiente. Por otro lado, los distintos experimentos numéricos con la librería pARMS, han demostrado no ser competitivos para el caso de estudio. Así mismo, el uso de métodos directos para resolver sistemas de ecuaciones lineales dispersos de gran dimensión en librerías como SuperLU, ha demostrado experimentalmente que a pesar de la alta precisión obtenida al resolver los sistemas dispersos, el tiempo de cómputo continua siendo prohibitivo para el caso de estudio abordado en este trabajo de tesis.

⋆ Se han aplicado distintos precondicionadores a los métodos de resolución de los sistemas de ecuaciones lineales dispersos de las matrices del problema estacionario abordado. En el caso secuencial con la librería SPARSKIT, el precondicionador más eficiente ha sido ILU0 sobre otros tipos de precondicionadores basados en LU incompleto como ILUT e ILUK. En los experimentos paralelos, se han destacado en desempeño los precondicionadores de Jacobi y Jacobi a bloques de la librería PETSc, sobre los distintos precondicionadores existentes en pARMS. Así mismo, para el caso de los experimentos paralelos con la librería numérica de PETSc, se han obtenido coeficientes aceptables de speedup y eficiencia.

⋆ Para el caso dinámico, la resolución eficiente de los sistemas lineales dispersos relacionados con la EDN 3D multigrupo juega un papel central en los estudios de estabilidad y seguridad de los reactores nucleares. Por tal motivo, esta memoria ha mostrado los resultados experimentales de dos métodos iterativos multipaso de grado $\hat{s} = 2$, basados en una partición de Jacobi (método MMJ) y en una partición de Gauss–Seidel acelerado (método MMGS), que se han aplicado al conjunto de matrices que surgen de un estudio de estabilidad del reactor nuclear Leibstadt. Para esto, se han utilizando las librerías

numéricas paralelas de PETSc e Hypre y se han realizado pruebas experimentales en dos clusters de alto desempeño: Kefren y Jacquard.

⋆ Se han propuesto modificaciones a los métodos multipaso MMJ y MMGS con el objetivo de acelerar la convergencia. Para lograr esto, se han diseñado dos métodos, aportes de esta tesis: el primero de ellos está basado en el uso de dos diferentes parámetros de relajación (ω_1 y ω_2) para cada grupo de energía, al cual se le ha nombrado método MMGS2. El segundo de ellos, llamado método MMGS2A, está basado en una técnica *adaptativa*, que consiste en ir variando la precisión de la solución conforme avanzan las iteraciones del método. Se ha comprobado que los métodos propuestos MMGS2 y MMGS2A resuelven los sistemas del caso de estudio en un tiempo de ejecución significativamente menor que los métodos MMJ y MMGS, alcanzando con ello el objetivo propuesto. De los distintos experimentos numéricos con los métodos MMGS2 y MMGS2A, este último es el que ha presentado mejor desempeño con respecto al resto de los métodos multipaso probados.

⋆ Al igual que en el problema estacionario, los métodos multipaso implementados han sido combinados con un conjunto de precondicionadores contenidos en las librerías paralelas de alto desempeño PETSc e Hypre, con el fin de identificar aquellos precondicionadores que aceleran la convergencia. En el caso de la librería PETSc, los precondicionadores aplicados han sido Jacobi y Jacobi a bloques. Para el caso de la librería Hypre, los precondicionadores probados han sido escalado diagonal (DS), BoomerAMG, ParaSails y Euclid. Del estudio realizado se ha encontrado que los precondicionadores de PETSc han presentado un mejor desempeño con respecto a los precondicionadores de Hypre, todos ellos combinados con el método Gradiente Conjugado. Otro resultado importante ha sido identificar a los mejores precondicionadores para el caso de estudio y que para el caso de la librería PETSc, el precondicionador Jacobi a bloques ha resultado ser el más eficiente; en el caso de la librería Hypre, el precondicionador Euclid ha sido el más eficiente.

⋆ La aplicación de los métodos y precondicionadores en ambas librerías (PETSc e Hypre) sobre la plataforma paralela de Jacquard han proporcionado valores aceptables de speedup y eficiencia en general. Sin embargo, Hypre registra los índices más altos speedup y eficiencia, lo que puede significar que los algoritmos de Hypre usan los recursos computacionales de una manera más eficiente y equilibrada.

⋆ La principal ventaja de los métodos multipaso presentados en esta tesis para el problema dinámico de la EDN, es en primer lugar, que la matriz de coeficientes del sistema a resolver no necesita manejarse como una única estructura de datos, lo que ocasionaría problemas de administración de almacenamiento primario y problemas al operar con dicha matriz; en segundo lugar, se evitaría tener que resolver una matriz no simétrica,

dispersa y de gran dimensión, la cual puede o no presentar problemas de convergencia. Por tal motivo, la aplicación de los métodos multipaso tratados por esta memoria, son convenientes y apropiados para la simulación con más de dos grupos de energía, como se ha puesto de manifiesto en el capítulo 6 de esta tesis, donde se describen las estructuras de los sistemas de ecuaciones a resolver, y se analizan y proponen esquemas paralelos de resolución. La posibilidad de utilizar más de dos grupos de energía en la modelización de la EDN, permitiría la realización de estudios más completos de seguridad y estabilidad en los reactores nucleares.

⋆ Por último, es necesario continuar investigando, aplicando y desarrollando nuevos métodos y probando librerías numéricas que permitan resolver eficientemente los distintos problemas que plantea la ingeniería. Esta tesis ha abordado en concreto sólo una parte de dos problemas fundamentales asociados a la EDN del campo de la ingeniería química nuclear, presentando por un lado, un estudio que permite distinguir la eficiencia obtenida por distintas librería numéricas del álgebra lineal dispersa, aplicadas al caso estacionario de la EDN; por otro lado, ha propuesto modificaciones a dos métodos iterativos multipaso que han logrado mayor eficiencia en el proceso de resolución de los sistemas dispersos del caso dinámico de la EDN.

Trabajos futuros

Los trabajos futuros que se desprenden de la presente memoria son:

⋆ Estudio, evaluación y aplicación a los casos de estudio de otras librerías numéricas dispersas secuenciales y paralelas como PARDISO (PARallel DIrect SOlver) [104] [52] [103], MUMPS (MUltifrontal Massively Parallel sparse direct Solver) [5] [3] [4], SPOOLES (SParse Object Oriented Linear Equations Solver), por ejemplo.

⋆ El estudio y aplicación de nuevos tipos de precondicionadores a los casos de estudio puede ser de interés particular, como por ejemplo los precondicionadores paralelos basados en inversas aproximadas dispersas como SPAI (SParse Approximate Inverse) [55]. Estas técnicas de precondicionamiento han dado buenos resultados como en el caso del precondicionador inverso ParaSails de la librería Hypre y que ha sido aplicado al problema dinámico de la EDN.

⋆ Están surgiendo nuevos métodos de discretización para la EDN como el que aparece en [51] que se carateriza por usar métodos pseudoespectrales para calcular los modos Lambda dominantes del núcleo de un reactor bidimensional. En esta discretización se usa un mallado triangular, en donde los flujos neutrónicos de cada nodo se expresan como polinomios modificados de Dubiner, lo que trae consigo nuevas formas del patrón disperso

resultante, por lo que los estudios realizados en esta memoria pueden extrapolarse a este nuevo tipo de discretización.

⋆ Aunque no se cuenta actualmente con matrices de estudio que se originen de casos de métodos de discretización de colocación nodal con más grupos de energía, sería interesante implementar y evaluar los algoritmos y estrategias paralelas mencionadas en el capítulo 6 a la luz de nuevas técnicas de paralelización como pueden ser la tecnología multihilo (multithread) en arquitecturas multicore o arquitecturas heterogéneas.

⋆ Por otro lado, sería interesante también diseñar, implementar y evaluar los algoritmos iterativos multipaso delineados en el capítulo 6 para el caso de dos grupos de energía en arquitecturas paralelas heterogéneas, así como evaluar y diseñar algoritmos que combinen cálculos con precisión mixta (combinación de precisión sencilla y doble) y técnicas de precisión adaptativa como las presentadas en el capítulo 5 de esta tesis.

Publicaciones

Parte de los resultados experimentales obtenidos de las investigaciones realizadas en esta tesis han dado lugar a las siguientes publicaciones:

Publicaciones Internacionales

- *Evaluation of Parallel High-Performance Libraries Applied to Neutron Diffusion Equation Problems* enviado a Mathematical Problems in Engineering [1] en noviembre del 2008. En revisión.

- *Sequential and Parallel Resolution of the Two-Group Transient Neutron Diffusion Equation using Second-Degree Iterative Methods*, (versión extendida). High-Performance Computing for Computational Science-VECPAR'2006, Lecture Notes in Computer Science, Vol. 4395, pp. 426-438, 2007.

- *Free Distribution Parallel Sparse Solvers. Application to Lambda Modes Equation*, (versión extendida). IADIS International Journal on Computer Science and Information System. Vol. 1/2, pp. 76-87, 2006.

- *Factorización LU a bloques en Arquitecturas con Memoria Distribuida.* Congreso Internacional de Computación Paralela, Distribuida y Aplicaciones. Actas del Congreso. Vol. 1, pp. 50-55, 2003.

[1] http://www.hindawi.com/journals/mpe

Comunicaciones Internacionales

- High-Performance Preconditioning Techniques for the Solution of Two-Group Transient Neutron Diffusion Equation. High-Performance Computing for Computational Science-VECPAR'08, 8th International Meeting, Toulouse, France, June 2008.
- Some Experiments using Preconditioning Techniques and Second-Order Iterative Methods for the Resolution of the 3D Transient Neutron Diffusion Equation. Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (MC&SNA+2007), Monterey, California, USA, April 2007.
- Sequential Numerical Study of some variations of a Second-Degree Iterative Method: Application to the Neutron Diffusion Equation. The Fifth International Conference on Engineering Computational Technology, Las Palmas de Gran Canaria, Spain, Septiembre 2006.
- Sequential and Parallel Resolution of the Two-Group Transient Neutron Diffusion Equation using Second-Degree Iterative Methods. High-Performance Computing for Computational Science-VECPAR'06, 7th International Meeting, Rio de Janeiro, Brazil, July 2006.
- Free Distribution Parallel Sparse Solvers. Application to Lambda Modes Equation. IADIS International Conference on Applied Computing, San Sebastián, Spain, Febrero 2006.

Comunicaciones Nacionales

- Estudio Comparativo de la Aplicación de Precondicionadores Paralelos para resolver la Ecuación de la Difusión Neutrónica Multigrupo 3D. Caso de estudio: PETSc e Hypre. XXXIV Reunión Anual SNE, Murcia, Octubre 2008.
- Métodos Iterativos de Segundo Grado Precondicionados para resolver la Ecuación de Difusión Neutrónica Multigrupo 3D. XXXIII Reunión Anual SNE, Segovia, Septiembre 2007.
- Implementación de Métodos Iterativos de Segundo Grado utilizando Computación de Altas Prestaciones para resolver la Ecuación de Difusión Neutrónica Multigrupo 3D. XXXII Reunión Anual SNE, Tarragona 2006.
- Estudio de la Eficiencia de Librerás Numéricas de Libre Distribución (PETSc, SPARSKIT) a la Resolución de los Sistemas Lineales Dispersos relacionados con la Ecuación de los modos Lambda. XXXI Reunión Anual SNE, Logroño, Octubre 2005.

Reportes de investigación

- O. Flores-Sánchez. *Aplicación de las Librerías Paralelas de PETSc y pARMS a la resolución de los Sistemas de Ecuaciones Lineales relacionados con la Ecuación de los Modos Lambda*. DSIC-II/14/05, Departamento de Sistemas Informáticos y Computación–DSIC, Universidad Politécnica de Valencia, 2005.

- O. Flores-Sánchez. *Estudio de la Eficiencia de Librerías Numéricas de Libre Distribución (PETSc,SPARSKIT) a la resolución de los Sistemas de Ecuaciones Lineales relacionados con la Ecuación de los Modos Lambda*. DSIC-II/15/05, Departamento de Sistemas Informáticos y Computación–DSIC, Universidad Politécnica de Valencia, 2005.

Proyectos de investigación

- Los trabajos derivados de esta tesis, se enmarcan dentro del proyecto de investigación *Computación de Altas prestaciones y Optimización aplicados a Modelos Neutrónicos–Termohidráulicos 3D* con clave ENE2005-09219-C02-02 auspiciado por el Ministerio de Educación y Ciencia (MEC) de España.

Bibliografía

[1] C.H. Adams. Currents trends in methods for neutron diffusion calculations. Technical Report CONF-770304-12, Argonne National Lab., IL, USA, 1997.

[2] D.U. Altiparmakov and D. Tomasevic. Variational formulation of a higher order nodal diffusion method. *Nuclear Science and Engineering*, 105(3):256–270, 1990.

[3] P. R. Amestoy, I. S. Duff, Koster J., and J.Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[4] P. R. Amestoy, I. S. Duff, and J.Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods Appl. Mech. Eng.*, 184:501–520, 2000.

[5] P.R. Amestoy, A. Guermouche, J.Y. L'Excellent, and Pralet S. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

[6] Josep Arnal García. *Algoritmos Iterativos Paralelos para la Resolución de Sistemas No Lineales*. PhD thesis, Departamento de Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante, 2000.

[7] W.E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29, 1951.

[8] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[9] G.I. Bell and S. Glasstone. *Nuclear Reactor Theory*. Robert E. Krieger Publishing Company, 1970.

[10] M. Benzi, J.C. Haws, and M. Tůma. Preconditioning highly indefinite and nonsymmetric matrices. *SIAM J. Sci. Comput.*, 22(4):1333–1353, 2000.

[11] J. Blomstrand. The KKL core stability test, conducted in september 1990. Technical Report Report BR91-245, ABB, 1992.

[12] W.L. Briggs. *A Multigrid Tutorial*. SIAM, Philadelphia, 1987.

[13] P.N. Brown, A.C. Hindmarsh, and L.R. Petzold. Using Krylov methods in the solution of large scale differential-algebraic systems. *SIAM J. Sci. Comput.*, 15(6):1467–1488, 1994.

[14] R. Bru, Ginestar D., Marín J., Verdú G., Mas J., and Manteuffel T. Iterative schemes for the neutron diffusion equation. *Computers and Mathematics with Applications*, 44:1307–1323, 2002.

[15] R.L. Burden and J.D. Faires. *Numerical Analysis*. Brooks/Cole, 2004.

[16] M. de J. Castel de Haro. *Métodos Iterativos Paralelos para la Resolución de Sistemas Lineales Hermíticos y Definidos Positivos*. PhD thesis, Universidad de Alicante, Valencia, España, 2000.

[17] G.S. Chen, J.M. Christenson, and D.Y. Yang. Application of the preconditioned transpose-free quasi-minimal residual method for two group reactor kinetics. *Annals of Nuclear Energy*, 24(5):339–35, 1996.

[18] E. Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21(5):1804–1822, 2000.

[19] E. Chow. Parallel implementation and practical use of sparse approximate inverses with a priori sparsity patterns. *Int'l J. High Perf. Comput. Appl.*, 15:56–74, 2001.

[20] H.N. Cofer and M.A. Stadtherr. Reliability of iterative linear equations solvers in chemical process simulation. *Computers Chem. Engrg.*, 20:1123–1132, 1996.

[21] J.W. Demmel, J.R. Gilbert, and S.L. Xiaoye. SuperLU Users' guide. Technical Report ANL-44289, Lawrence Berkeley National Laboratory, 2003.

[22] Jack J. Dongarra, I.S. Duff, Danny C. Sorensen, and Henk A. Van der Vorst. *Numerical Linear Algebra for High–Performance Computers*. SIAM–Society for Industrial and Applied Mathematics, Philadelphia, 1998.

[23] T. Downar and E. Lewis. Adaptive nodal transport methods for reactor transient analysis. Technical Report DE-FG07-01ID14106, Purdue University, IN, USA, 2005.

[24] I.S. Duff, A.M. Erisman, and J.K. Reid. *Direct Methods for Sparse Matrices*. Clarendon Press, Oxford, 1986.

[25] I.S. Duff, R.G. Grimes, and J.G. Lewis. User's guide for the Harwell–Boeing sparse matrix collection (Release-I). Technical Report RAL 92-086, Rutherford Appleton Laboratory, 1992.

[26] I.S. Duff, R.G. Grimes, and J.G. Lewis. The Rutherford-Boeing sparse matrix collection. Technical Report RAL-TR-97-031,ISSTECH-97-017, TR/PA/97/36, Rutherford Appleton Laboratory, Boeing Information and Support Services, CERFACS, 1997.

[27] I.S. Duff and Scott J.A. Computing selected eigenvalues of sparse unsymmetric matrices using Subspace iteration. *ACM Trans. on Mathematical Software*, 19(2):137–159, 1993.

[28] D.G. Duffy. *Advanced Engineering Mathematics with Matlab*. Chapman & Hall, Boca Ratón, 2003.

[29] Robert D. Falgout, Jim E. Jones, and Ulrike M. Yang. Domain decomposition methods for partial differential equations on parallel computers. *Int. J. Supercomputing Applications*, 2:5–12, 1988.

[30] Robert D. Falgout, Jim E. Jones, and Ulrike M. Yang. Conceptual interfaces in Hypre. *Future Generation Computer Systems*, 22:239–251, 2006.

[31] R. Fletcher. Conjugate gradient methods for indefinite systems. *Proceedings of the Dundee Bienal Conference on Numerical Analysis 1974*, pages 73–89, 1975.

[32] O. Flores-Sánchez and V.E. Vidal. Free distribution parallel sparse solvers. Application to Lambda Modes equation. *IADIS International Journal on Computer Science and Information Systems*, pages 76–87, 2006.

[33] O. Flores-Sánchez, V.E. Vidal, L.A. Drummond, and G. Verdú. High performance preconditioning techniques for the solution of two-group transient neutron diffusion equation. In *High–Performance Computing for Computational Science–VECPAR'08, 8th International Meeting*. Toulouse, France, 2008.

[34] O. Flores-Sánchez, V.E. Vidal, V.M. García, and P. Flores-Sánchez. Free distribution parallel sparse solvers. Application to Lambda Modes equation. In *IADIS International Conference on Applied Computing*. San Sebastián, Spain, 2006.

[35] O. Flores-Sánchez, V.E. Vidal, V.M. García, and P. Flores-Sánchez. Sequential and parallel resolution of the two-group transient neutron diffusion equation using

second-degree iterative methods. In *High–Performance Computing for Computational Science–VECPAR'06, 7th International Meeting*. Rio de Janeiro, Brazil, 2006.

[36] O. Flores-Sánchez, V.E. Vidal, G. Verdú, J. Garayoa, and P. Flores-Sánchez. Sequential numerical study of some variations of a second-degree iterative method. Application to the neutron diffusion equation. In *The Fifth International Conference on Engineering Computational Technology*. Las Palmas de Gran Canaria, Spain, 2006.

[37] O. Flores-Sánchez, V.E. Vidal, G. Verdú, and R. Miró. Some experiments using preconditioning techniques and second-order iterative methods for the resolution of the 3D transient neutron diffusion equation. In *Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)*. California, USA, 2007.

[38] R.W. Freund. A transpose-free quasi-minimal residual algorithm for non–hermitian linear systems. *SIAM J. Sci. Comput.*, 14:470–482, 1993.

[39] R.W. Freund and N.M. Nachtigal. QMR: A quasi-minimal residual method for non–hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[40] V.M. García, V. Vidal, J. Garayoa, G. Verdú, and R. Gómez. Fast resolution of the neutron diffusion equation through public domain ODE codes. *International Conference on SuperComputing in Nuclear Applications, SNA 2003*, 5:146, 2003.

[41] V.M. García, V. Vidal, G. Verdú, J. Garayoa, and R. Miró. Parallel resolution of the two-group time dependent neutron diffusion equation with public domain ODE codes. *VECPAR'04*, 2004.

[42] V.M. García, V. Vidal, G. Verdú, and R. Miró. Sequential and parallel resolution of the 3D transient neutron diffusion equation. *Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, ANS 2005*, on CD-ROM, 2005.

[43] A. George. Nested dissection of a regular finite element mesh. *SIAM J. on Numerical Analysis*, 10:345–363, 1973.

[44] A. George and N. Esmond. Symbolic factorization for sparse Gaussian elimination with partial pivoting. *SIAM J. Sci. Stat. Comput.*, 8(6):877–898, 1987.

[45] A. George and Joseph W.H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1–19, 1989.

[46] J.A. George and J.W. Liu. *Computer Solution of Large Sparse Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[47] D. Ginestar. *Integración de la Ecuación de la Difusión Neutrónica en Geometrías Multidimensionales. Aplicación a Reactores Nucleares. Cálculos de los Modos Lambda*. PhD thesis, Universidad Politécnica de Valencia, 1995.

[48] D. Ginestar, J. Marín, and G. Verdú. Multilevel methods to solve the neutron diffusion equation. *Applied Mathematical Modelling*, 25:463–477, 2001.

[49] D. Ginestar, G. Verdú, V. Vidal, R. Bru, J.M. Mateos, and Muñoz-Cobo J.L. High order backward discretization of the neutron diffusion equation. *Ann. Nucl. Energy*, 25(1–3):47–64, 1998.

[50] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[51] S. González-Pintor, D. Ginestar, and G. Verdú. Método pseudoespectral continuo para el cálculo de los Modos Lambda dominantes en reactores con geometría hexagonal. In *Actas en CD–ROM de la 34 Reunión anual de la Sociedad Nuclear Española*. Sociedad Nuclear Española, Murcia, España, 2008.

[52] N.I.M Gould, Y. Hu, and J.A. Scott. A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations. Technical Report RAL-TR-2005-005, Council for the Central Laboratory of the Research Councils Rutherford Appleton Laboratory, Oxfordshire, (UK), 2005.

[53] A. Greenbaum. Iterative methods for solving linear systems. *Frontiers in Applied Mathematics. SIAM*, 17, 1997.

[54] L. Grigori, J.W. Demmel, and X.S. Li. Parallel symbolic factorization for sparse LU with static pivoting. *SIAM J. Scientific Computing*, 29(3):1289–1314, 2007.

[55] M.J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18(3):838–853, 1997.

[56] W. Groupp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with Message Passing Interface*. MIT Press, 1994.

[57] A. Gupta. Recent advances in direct methods for solving unsymmetric sparse systems of linear equations. *ACM Trans. Math. Softw.*, 28(3):301–324, 2002.

[58] W. Hackbusch and U. Trottenberg. *Multigrid Methods*. Springer–Verlag, Berlin, New York, 1982.

[59] A. Hébert. Application of the Hermite method for finite element reactor calculations. *Nuclear Science and Engineering*, 91:34–58, 1985.

[60] A. Hébert. Development of the nodal collocation method for solving the neutron diffusion equation. *Ann. of Nucl. Energy*, 14(10):527–541, 1987.

[61] V.E. Henson and U.M. Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41:155–177, 2002.

[62] V. Hernández, J.E. Román, A.M. Vidal, and V. Vidal. Calculation of Lambda Modes of a nuclear reactor: a parallel implementation using the Implicitly Restarted Arnoldi method. In Springer, editor, *VECPAR'98 - 3rd International Conference on Vector and Parallel Processing*, volume 1573 of *Lecture Notes in Computer Science*, pages 43–57, 1999.

[63] V. Hernández, J.E. Román, A.M. Vidal, and V. Vidal. Diversos planteamientos del problema de valores propios generalizado: Aplicación a la ecuación de los Modos Lambda. *XVI CEDYA / VI CMA*, pages 1173–1180, 1999.

[64] V. Hernández, J.E. Román, and V. Vidal. Computing the Lambda Modes of a nuclear reactor with SLEPc and PETSc. In *International Conference on Computational Methods for Mathematical Science and Engineering*, pages 183–192, 2002.

[65] V. Hernández, J.E. Román, V. Vidal, and G. Verdú. Resolución de problemas de valores propios perturbados por el método de homotopía. In *Actas del XVII CEDYA/VII CMA*, pages 427–428, Salamanca, 2001.

[66] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–435, 1952.

[67] A.C. Hindmarsh and A.G. Taylor. PVODE and KINSOL: Parallel software for differential and nonlinear systems. Technical Report UCRL-ID-129739, Lawrence Livermore National Laboratory, 1998.

[68] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, New York, 1985.

[69] D. Hysom and A. Pothen. Efficient parallel computation of ILU(k) preconditioners. In ACM, editor, *Proceedings of Supercomputing '99*, pages 73–130. IEEE Computer Society Press, November,, 1999.

[70] D. Hysom and A. Pothen. A scalable parallel algorithm for incomplete factor preconditioning. *SIAM J. on Scientific Computing*, 22(6):2194–2215, 2001.

[71] Duff I.S., Grimes R.G., and Lewis J.G. Sparse matrix test problems. *ACM Trans. Math. Softw.*, 15(1):1–14, 1989.

[72] W.S. Jae and K. Jong-Kyung. *Nuclear Technology*, 1993.

[73] D. Jespersen. Multigrid methods for partial differential equations. *Studies in Numerical Analysis*, 24 of Studies in Mathematics, Mathematical Association of America, 1984.

[74] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to parallel computing: design and analysis of parallel algorithms.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.

[75] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Stand.*, 45:255–282, 1952.

[76] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Stand.*, 49:33–53, 1952.

[77] S. Langenbuch, W. Maurer, and W. Werner. Coarse-mesh flux-expansion method for the analysis of space-time effects in large lightwater reactor cores. *Nuclear Science and Engineering*, 63:437–456, 1977.

[78] Z. Li, Y. Saad, and M. Sosonkina. pARMS: A parallel version of the algebraic recursive multilevel solver. Technical Report UMSI-2001-100, Minnesota Supercomputer Institute, University of Minnesota, 2001.

[79] Z. Li, Y. Saad, and M. Sosonkina. pARMS: A parallel version of the algebraic recursive multilevel solver. *Numerical Linear Algebra with Applications*, 10:485–509, 2003.

[80] R.J. Lipton, D.J. Rose, and R.E. Tarjan. Generalized nested dissection. *SIAM J. on Numerical Analysis*, 16:346–358, 1979.

[81] J.M. Mateos Aparicio. *Métodos Numéricos para la Resolución de la Ecuación de la Difusión Neutrónica Dependiente del Tiempo.* PhD thesis, Universidad Politécnica de Valencia, Valencia, España, 2000.

[82] M.V. Migallón. *Modelos Iterativos Caóticos Síncronos y Asíncronos para la Resolución de Sistemas Lineales.* PhD thesis, Departamento de Tecnología Informática y Computación, Universidad de Alicante, 1993.

[83] Rafael Miró Herrero. *Métodos Modales para el Estudio de Inestabilidades en Reactores Nucleares BWR.* PhD thesis, Universidad Politécnica de Valencia, 2002.

[84] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. on Numerical Analysis*, 11:197–209, 1974.

[85] B. Pershagen. *Light Water Reactor Safety.* Pergamon Press, Oxford, 1989.

[86] J.W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S.F. McCormick, editor, *Multigrid methods*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[87] Y. Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Mathematics of Computation*, 42(166):567–588, 1984.

[88] Y. Saad. SPARSKIT: A basic toolkit for sparse matrix computations. Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, 1990.

[89] Y. Saad. *Numerical Methods for Large-Eigenvalue Problems: Theory and Algorithms.* Wiley, New York, 1992.

[90] Y. Saad. SPARSKIT: A basic toolkit for sparse matrix computations. Version 2, manual. Technical report, Computer Science Dept., University of Minnesota, Minneapolis, MN 55455., 1994.

[91] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, 2003.

[92] Y. Saad and A. Malevsky. PSPARSLIB: A portable library of distributed memory sparse iterative solvers. In V.E.M. et al., editor, *Proceedings of Parallel Computing Technologies (PaCT–95), 3–rd International Conference*, St. Petersburg, Russia,September, 1995.

[93] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statis. Comput.*, 7:856–869, 1986.

[94] Y. Saad and M. Sosonkina. Solution of distributed sparse linear systems using PSPARSLIB. In B. Kågström et al., editor, *Applied Parallel Computing (PARA–98), 3–rd international conference*, pages 503–509. No. 1541, Springer–Verlag, Berlin, 1998.

[95] Y. Saad and M. Sosonkina. pARMS: A package for the parallel iterative solution of general large sparse linear systems: User's guide. Technical report, Minnesota Supercomputer Institute, University of Minnesota, 2003.

[96] Y. Saad and B. Suchomel. ARMS: An algebraic recursive multilevel solver for general sparse linear systems. Technical Report UMSI-99-107, Minnesota Supercomputer Institute, University of Minnesota, 1999.

[97] Y. Saad and J. Zhang. BILUM: Block versions of multi–elimination and multi–level ilu preconditioner for general sparse linear systems. *SIAM J. on Scientific Computing*, 21, 2000.

[98] A.T. Sanz and A.T. Onrubia. *Diccionario Inglés–Español sobre Tecnología Nuclear*. TECNATOM S.A. y Foro de la Industria Nuclear Española, 2008.

[99] Balay Satish, D. Gropp William, C. McInnes Lois, and F. Smith Barry. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A.M. Bruaset, and H.P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Nirkhauser Press, 1997.

[100] Balay Satish, D. Gropp William, C. McInnes Lois, and F. Smith Barry. Petsc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 1997.

[101] Balay Satish, D. Gropp William, C. McInnes Lois, and F. Smith Barry. PETSc home page. http://www.mcs.anl.gov/petsc, 2001.

[102] R. Scheichl. Parallel solvers for the transient multigroup neutron diffusion equation. *International Journal for Numerical Methods in Engineering*, 47:1751–1771, 2000.

[103] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems–FGCS*, 20:475–487, 2004.

[104] O. Schenk and K. Gärtner. On fast factorization pivoting methods for symmetric indefinite systems. *Elec. Trans. Numer. Anal.*, 23:158–179, 2006.

[105] R.A. Shober, R.N. Sims, and A.F. Henry. *Nuclear Science and Engineering*, 1977.

[106] Gerard L.G. Sleijpen and Henk A. Van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM J. on Matrix Analysis and Applications*, 17(2):401–425, 1996.

[107] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10:36–52, 1989.

[108] D.C. Sorensen. Implicit application of polynomial filters in a k–step Arnoldi method. *SIAM J. on Matrix Analysis and Applications*, 13(1):357–385, 1992.

[109] M. Sosonkina, Y. Saad, and X. Cai. Using the parallel algebraic recursive multilevel solver in modern physical applications. *Future Generation Computer Systems*, 20:489–500, 2004.

[110] Hypre Team. Hypre reference manual. Technical Report UCRL-CODE-222953 - Software Version 2.2.0, Center for Applied Scientific Computing Lawrence Livermore National Laboratory, 2006.

[111] Hypre Team. Hypre user's manual. Technical Report UCRL-CODE-222953 - Software Version 2.2.0, Center for Applied Scientific Computing Lawrence Livermore National Laboratory, 2007.

[112] H.A. Van der Vorst. Bi-CGstab: A fast and smoothly converging variant of bi–cg for the solution of non–symmetric linear systems. *SIAM J. on Scientific and Statistical Computing*, 12(6):631–644, 1992.

[113] R.S. Varga. *Matrix Iterative Analysis.* Prentice–Hall, Englewood Cliffs, NJ, 1962.

[114] G. Verdú. *Teoría General del Transporte Estocástico de Neutrones y sus Aplicaciones a la Medida de Reactividad en Conjuntos Subcríticos.* PhD thesis, Universidad Politécnica de Valencia, 1984.

[115] G. Verdú, D. Ginestar, R. Miró, and V. Vidal. Using the Jacobi-Davidson method to obtain the dominant Lambda Modes of a nuclear power reactor. *Ann. of Nuclear Energy*, 32:1274–1296, 2005.

[116] G. Verdú, D. Ginestar, V. Vidal, and Muñoz-Cobo J.L. 3D λ-modes of the neutron diffusion equation. *Ann. of Nucl. Energy*, 21(7):405–421, 1994.

[117] G. Verdú, D. Ginestar, V. Vidal, and J.L. Muñoz-Cobo. A consistent multidimensional nodal method for transient calculation. *Ann. Nucl. Energy*, 22(6):395–411, 1995.

[118] G. Verdú, J.L. Muñoz Cobo, C. Pereira, and D. Ginestar. Lambda modes of the neutron diffusion equation: Application to BWR's Out-of-Phase instabilities. *Ann. of Nucl. Energy*, (7):477–501, 1993.

[119] V. Vidal, J. Garayoa, G. Verdú, and D. Ginestar. Optimization of the Subspace Iteration method for the Lambda Modes determination of a nuclear power reactor. *Journal of Nuclear Science and Technology*, 34(9):929–947, 1997.

[120] V. Vidal, G. Verdú, D. Ginestar, and J.L. Muñoz-Cobo. Variational acceleration for Subspace Iteration method. Application to nuclear power reactors. *International Journal for Numerical Methods in Engineering*, 41:391–436, 1998.

[121] Vicente Vidal. *Métodos Numéricos para la obtención de los Modos Lambda de un Reactor Nuclear. Técnicas de Aceleración y Paralelización.* PhD thesis, Universidad Politécnica de Valencia, Valencia, España, 1997.

[122] H.F. Walker. Implementation of the GMRES method using Householder transformations. *SIAM J. Sci. Statist. Comput.*, 9:152–163, 1988.

[123] J.R. Weston and M. Stacey. *Space–Time Nuclear Reactor Theory.* Robert E. Krieger Publishing Company, 1970.

[124] S. Li Xiaoye. An overview of *SuperLU*: Algorithms, implementation, and user interface. *ACM Trans. Math. Softw.*, 31(3):302–325, 2005.

[125] S. Li Xiaoye and J.W. Demmel. SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. on Mathematical Software*, 29(2):110–140, June 2003.

[126] D.M. Young. *Iterative Solution of Large Linear Systems.* Academic Press Inc., 111 Fifth Avenue, New York, 1971.

[127] T. Yutaka, T. Yukito, U. Hitoshi, E. Shiegeo, J.C. Shau, and S. Bharat. *Nuclear Technology*, 1994.

[128] V.G. Zimin and H. Ninokata. Acceleration of the outer iterations of the space–dependent neutron kinetics equations solution. *Annals of Nuclear Energy*, 23(17):1407–1424, 1996.