Mecánica Computacional Vol. XXIV
A. Larreteguy (Editor)
Buenos Aires, Argentina, Noviembre 2005

MEJORA DEL DESEMPEÑO COMPUTACIONAL DEL RMA 10

Pablo Ezzatti [*] **e Ismael Piedra-Cueva** [†]

* Instituto de Computación, Facultad de Ingeniería,
Universidad de la República, Montevideo, Uruguay
e-mail: pezzatti@fing.edu.uy

† Instituto de Mecánica de los Fluidos e Ingeniería Ambiental, Facultad de Ingeniería,
Universidad de la República, Montevideo, Uruguay
e-mail: ismaelp@fing.edu.uy

Key words: Desempeño computacional, RMA 10, elementos finitos.

Abstract. *Uno de los modelos más utilizados para la simulación de mareas es el RMA 10 desarrollado por I. P. King (1993) y sus colaboradores. Entre las principales características se puede destacar sus cualidades de descripción, su capacidad de trabajar con grilla irregular y con distintos tipos de elementos y el tratamiento que realiza de la viscosidad turbulenta. Sin embargo en muchos casos, principalmente cuando se trabaja en 3 dimensiones, los excesivos costos computacionales, tiempo de cálculo, del modelo lo transforman en una elección prohibitiva. El trabajo presenta una modificación de la rutina encargada del cálculo y posterior resolución de las matrices involucradas en la aplicación del método de Newton-Raphson para la resolución de sistemas no lineales. Los resultados del desempeño computacional de la propuesta son sumamente promisorios, también se exponen diversos caminos interesante a seguir.*

1 INTRODUCCIÓN

En las últimas décadas, la simulación computacional de fluidos ha emergido como un área de intenso trabajo. A medida que las técnicas computacionales se fueron perfeccionando, la ambición de los científicos fue en aumento, tratando de modelar cada vez con mayor precisión el comportamiento de los fluidos, intención que exige importantes inversiones en equipamiento informático. De este modo se genera un círculo vicioso que frecuentemente coloca a la simulación de alta precisión como una técnica económicamente prohibitiva.

Uno de los modelos de simulación computacional de flujo a superficie libre baroclino tri-dimensional frecuentemente utilizado es el modelo RMA 10, propuesto por I. King en 1993, y desarrollado bajo el paradigma de elementos finitos utilizando la formulación de Galerkin residual. Si bien sus cualidades de descripción, su capacidad de trabajar con grilla irregular y con distintos tipos de elementos y el tratamiento que realiza de la viscosidad turbulenta lo sitúan como una de las mejores opciones para el modelado de estuarios, el tiempo de cálculo que insume al incrementar el número de elementos utilizados lo perjudica notoriamente al compararlo con otras opciones. Como ejemplo, la simulación hidrodinámica tridimensional baróclina del Río de la Plata de 8 días, con pasos de tiempo de 20 minutos, utilizando una malla de 1630 elementos con 5113 nodos, implica un tiempo de cómputo del orden de 24 horas sobre un computador Dell con dos procesadores Pentium IV de 1.8 GHz y 1 GB de memoria RAM. Estos tiempos de cómputo pueden hacer al modelo poco útil en escenarios de simulaciones de largo plazo, en especial en etapas de estudio paramétrico del modelo.

Este trabajo expone un estudio de los tiempos de ejecución insumidos por las distintas subrutinas del RMA 10 y presenta una propuesta para mejorar el desempeño computacional del modelo. La mejora se concentra en la rutina encargada del cálculo de los coeficientes y posterior factorización de los sistemas lineales que se resuelven en cada paso del método de Newton-Raphson utilizado por el RMA 10 para resolver los sistemas no lineales derivados del modelo de elementos finitos. La modificación propuesta consiste en separar las etapas de cálculo, generación y factorización de la matriz de cada sistema lineal. En una primera etapa se calculan todos los coeficientes de la matriz, almacenándolos en una estructura auxiliar de tipo tabla de dispersión[1]. Luego, se genera una matriz dispersa en algún formato preestablecido. Por último, se factoriza y se resuelve el sistema utilizando la biblioteca de uso público MUltifrontal Massively Parallel sparse direct Solver versión 4.3 (MUMPS) [2,3], en su modalidad secuencial.

Durante sus diferentes etapas, la biblioteca MUMPS necesita de las rutinas de los Basic Linear Algebra Subprograms (BLAS), en nuestro caso se utilizó la versión 0.97 multithreads de la implementación de Kazushige Goto para procesadores pentium IV con 512 Kb de Cache bajo sistemas operativos linux de dicha biblioteca.

La propuesta se evalúa sobre una simulación hidrodinámica tridimensional baróclina del Río de la Plata, como la mencionada anteriormente como ejemplo. Se reportan mejoras significativas en los tiempos de cómputo para una simulación de 8 días, reduciendo los tiempos a menos de una tercera parte de su valor original.

La sección 2 presenta una breve descripción del modelo RMA 10, sus prestaciones y diversas características de su implementación. La sección 3 expone el estudio de los tiempos de cómputos de las diversas secciones del RMA 10 aplicado al modelado numérico del Río de la Plata. La sección 4 explica los cambios propuestos y las distintas técnicas utilizadas. Los resultados experimentales sobre las instancias de prueba consideradas se presentan y comparan en la sección 5. Por último, se ofrecen las conclusiones y propuestas de trabajo futuro.

2 EL MODELO RMA 10

El modelo RMA-10 , desarrollado por el Dr. Ian P. King [11] y su equipo en el Resource Management Associates, fue diseñado para simular la circulación hidrodinámica cuando las velocidades verticales son importantes y la estratificación por densidad es un factor significativo. El modelo describe las variables de estado presión y velocidad en tres dimensiones resolviendo un conjunto de ecuaciones que derivan de la combinación de las ecuaciones de Navier-Stokes, continuidad volumétrica, advección-difusión, y una ecuación de estado que relaciona la densidad con la salinidad, temperatura y sedimento suspendido. En este modelo las fuerzas de fricción, el efecto de Coriolis y la tensión del viento en la superficie también se representan. Puede usarse para simular situaciones dependientes del tiempo o estacionarias.

Las ecuaciones básicas, junto con las condiciones de borde e iniciales apropiadas, son resueltas numéricamente utilizando la formulación estándar de elementos finitos, Galerkin residual. La técnica de los elementos finitos, si bien es computacionalmente más costosa que los métodos clásicos de las diferencias finitas, tiene la enorme ventaja de poder representar adecuadamente la forma de contornos muy irregulares, utilizando celdas de distinto tamaño según la necesidad de mayor resolución en las zonas de interés. Esto evita la implementación de modelos encajados, técnica usualmente aplicada en los modelos basados en diferencias finitas, donde secuencialmente se incrementa el nivel de resolución en diferentes etapas.

Otro elemento de importancia es que la implementación resuelve las ecuaciones dinámicas en un esquema puramente implícito, el cual tiene la ventaja práctica de admitir pasos de tiempo relativamente grandes, aún cuando se trabaja dinámicamente con mallas de alta resolución. Esta cualidad permite modelar ciclos de marea y variaciones rápidas de las condiciones de contorno utilizando pasos de tiempo relativamente grandes.

Para la generación de la grilla, el modelo permite definir en el plano elementos isoparamétricos lineales, triangulares y cuadriláteros. Dentro de un elemento, el modelo utiliza aproximaciones cuadráticas para representar las componentes de la velocidad y las variaciones en la salinidad, temperatura o sedimento suspendido, en contraposición con los métodos de diferencias finitas que lo hace linealmente. Para la profundidad, el modelo utiliza una aproximación lineal.

Las condiciones de borde se pueden especificar en términos de descarga, velocidad o de nivel del agua, con una estructura vertical de la densidad en caso de ser necesario.

Otro aspecto relevante y diferencial del modelo es el tratamiento que realiza de la viscosidad turbulenta. Mientras que la mayoría de los modelos bidimensionales integrados en

profundidad trabajan con un único valor del coeficiente de viscosidad turbulenta, el modelo RMA 10 trabaja con diferentes opciones. La opción más simple es trabajar con un único valor del coeficiente de viscosidad turbulenta, el cual es constante. Una segunda opción es trabajar con el modelo de Smagorinsky. La incorporación de este tipo de modelo de turbulencia permite una mejor modelación de las zonas donde existen vórtices de gran escala, como los generados aguas debajo de pilas de puente, islas, o en zonas de separación del flujo.

El RMA 10 ha sido aplicado con importante éxito, entre otras oportunidades, en la Bahía de San Franciso, en New York Harbor, en la Bahía Galveston y en el Río St. Lawrence[13], así mismo su aplicación en el Río de la Plata[6] arrojo insipientes resultados especialmente en lo referido a resultados numéricos.

En cuanto a la implementación computacional del modelo, el sistema consta de diversos módulos codificados en su mayoría en FORTRAN 77. En la Figura 1 se puede observar el pseudo-código simplificado del funcionamiento del RMA 10.

```
0 Inicialización
1 Desde 1 hasta cantPasosSimulación
2         Calcular estructuras auxiliares
3         Desde 1 hasta cantPasosNewton-Raphson
4                 Resolver sistema lineal para calcular aporte
5                 Desafectar ecuaciones que cumplen condición de convergencia
6         Fin
7     Modificar las condiciones iniciales
8     Registrar resultados
9 Fin
```

Figura 1. Pseudo-código simplificado del RMA 10.

La primera etapa de la rutina, paso 0 en la Figura 1, se puede ver en forma general como una componente de inicialización, la cual se encarga de la lectura de la grilla (generada por los programas RMA-1 o RMAGEN [15]), la lectura de los valores iniciales y las condiciones de borde, realiza diversas transformaciones y genera distintas estructuras auxiliares necesarias para las siguientes etapas.

También se dispone en esta etapa, de una rutina capaz de renumerar los nodos (sin necesidad de renumerar la grilla) de los distintos elementos de la grilla. Para esto, se utiliza el algoritmo Reverse Cuthill McKee (RCM) de forma tal de minimizar el ancho de la banda de las matrices que se forman, el tiempo de cálculo y las necesidades de almacenamiento del RMA 10 depende en forma directa de los logros en el reordenamiento.

Luego el sistema trabaja en forma de bucle, desde el paso 1 al paso 9, iterando en los intervalos de tiempo que se desea simular. En cada paso de la iteración se resuelve un sistema de elementos finitos no lineal que utiliza de condición inicial el resultado del paso anterior de la iteración, exceptuando el primer paso en el cual se utilizan los valores ingresados.

Para resolver las ecuaciones no lineales se utiliza el método de Newton-Raphson, pasos 3,4,5 y 6, método basado en una iteración de resoluciones de sistemas lineales. El RMA 10 permite establecer el criterio de convergencia y la cantidad máxima de iteraciones del método, siendo parámetros de entrada. Debido al gran costo computacional que involucra la resolución de grandes sistemas lineales, el RMA 10 posee la capacidad de desafectar, en pasos intermedios de la iteración de Newton-Rahson, aquellas ecuaciones que cumplan el criterio de convergencia.

El cálculo de los distintos coeficientes de la matriz y su posterior resolución, paso 4 del pseudo-código, es realizado por la rutina FRONT del sistema, utilizando diversas sub-rutinas para el cálculo de los coeficientes. La estrategia de resolución utilizada es el método frontal presentado por Irons en 1970 [10] y extendido para matrices no simétricas por Hood en 1976 [9].

Los métodos frontales y sus sucesores los métodos multi-frontales presentados por Duff y Reid en 1983 [5] son versiones de la eliminación Gaussiana, relacionadas directamente con los elementos finitos y diseñadas para disminuir las necesidades de memoria. Incluso son capaces de lograr la resolución de grandes matrices en las máquinas disponibles en los años 70, ya que trabajan cargando a memoria para su resolución la matriz de a trozos (los distintos frentes).

3 COSTO COMPUTACIONAL

Si bien cualidades del RMA 10, tales como, capacidad de trabajar con una grilla irregular, posibilidad de utilizar al mismo tiempo distintos tipos de elementos y el tratamiento que realiza de la viscosidad turbulenta lo sitúan como una de las mejores opciones para el modelado de estuarios, el tiempo de cálculo que insume al incrementar el número de elementos utilizados lo perjudica notoriamente al compararlo con otras opciones. Como ejemplo, la simulación hidrodinámica tridimensional baróclina del Río de la Plata de 8 días, con pasos de tiempo de 20 minutos (576 pasos), utilizando una malla de 5635 nodos implica un tiempo de cómputo del orden de 24 horas sobre un computador Dell con dos procesadores Pentium IV de 1.8 GHz y 1 GB de memoria RAM. Estos tiempos de cómputo pueden hacer al modelo poco útil en escenarios de simulaciones de largo plazo, en especial en etapas de estudio paramétrico del modelo.

Por lo visto anteriormente, mejorar el desempeño computacional del RMA 10 es una condicionante importante para su utilización. Para lograr este objetivo es necesario saber si existen secciones del código criticas, es decir, secciones del programa que su ejecución implica un alto porcentaje del total del tiempo de cómputo y de existir lograr identificarlas en forma precisa. En dicho sentido, se realizó un pequeño análisis de los tiempos de ejecución de las distintas etapas del RMA 10 aplicado al modelado del Río de la Plata (en al sección 4 se detallan la instancia de prueba y la plataforma de ejecución).

En la Tabla 1, se resumen los resultados arrojados por el estudio de los tiempos de ejecución de las etapas más costosas del RMA 10. Es de notar que la etapa de inicialización es la única independiente de la cantidad de pasos a simular. El cálculo del tiempo de resolución se realizó en base al promedio de los tiempo de la resolución de los sistemas de ecuaciones de mayor tamaño, es decir sin desacoplar ecuaciones. De los resultados se puede concluir en

forma determinante que la mayor fuente de consumo de tiempo de cómputo es la etapa de cálculo y factorización de las matrices en cada paso del método Newton-Raphson para la resolución de los sistema no lineal realizado por la rutina FRONT.

Etapa	Tiempo (seg)
Inicialización	0.59
Calcular estructuras auxiliares	0.02
Resolver sistema lineal para calcular aporte	34.32
Desafectar ecuaciones que cumplen condición de convergencia	0.10
Modificar las condiciones iniciales y Registrar resultados	0.06

Tabla 1. Tiempos de las distintas etapas del RMA 10.

Debido a que la rutina FRONT es la más costosa y de forma de afinar el estudio, se evaluó el costo de las distintas etapas de dicha rutina. Como se mencionó, la rutina FRONT no calcula la matriz en forma completa en una primera etapa y luego la factoriza, sino que su modalidad de trabajo es ir calculando los coeficientes y factorizar las distintas partes de la matriz (frentes), de forma de utilizar menos cantidad de memoria. Sin embargo, para intentar identificar de forma clara las zonas criticas del código, se delimitaron y calcularon la sumatoria de los tiempos de tres fragmentos lógicos que a priori posee la rutina: el cálculo de los coeficientes, la factorización de las distintas partes de la matriz dispersa y la sustitución para atrás para obtener el vector resultado.

Etapa	Tiempo (seg)
Cálculo de coeficientes	2.99
Sustitución hacia atrás	0.56
Factorización	30.77
Total	34.32

Tabla 2. Tiempos de las distintas etapas lógicas de la rutina FRONT.

En la Tabla 2 se puede observar la relación de tiempos de cómputo de las tres partes de la rutina FRONT. Pudiéndose observar que la factorización es en forma categórica la etapa más costosa de la rutina.

4 PROPUESTA

Como se presentó en la sección anterior el costo computacional del RMA 10 es debido en un alto porcentaje a los cálculos involucrados por la rutina FRONT, por esta razón es que la propuesta se centra en mejorar dicha rutina y en particular la fase de factorización.

La nueva rutina propuesta se pudo dividir, a diferencia de la anterior, en tres fases independientes: una primera etapa de cálculo de los coeficientes de la matriz, una segunda de rearmado, filtrado y generación de la matriz dispersa y por último la factorización y resolución del sistema disperso.

En la primera etapa de cálculo de los coeficientes, se itera sobre todos los elementos de la malla. Para cada elemento, se calcula el aporte a los coeficientes de la matriz, de cada uno de los nodos para cada una de las variables libres definidas. Para esta tarea se utilizan las distintas rutinas originales del sistema.

Una vez obtenido los coeficientes de un elemento se almacenan en una estructura auxiliar general de tipo tabla de dispersión abierta[1], en la cual la función de dispersión utilizada queda definida por la fila del coeficiente al cual se aplica (*f_dispersión(coefij) = i*). Una vez en la cubeta *i* de la tabla, se busca si ya existe algún valor que posea columna j, en caso afirmativo se le suma al valor de la cubeta el valor calculado *coefij*, en caso contrario, se agrega al final de la lista. Al finalizar esta etapa se cuenta con todos los coeficientes de la matriz en un formato disperso, sin embargo por la forma de construcción pueden existir entradas de fila y columna que posean valores nulo.

La segunda etapa consiste en filtrar los coeficientes nulos que pudiesen haberse generado. Ordenar, de ser necesario los coeficientes de la matriz y generar las estructuras auxiliares necesarias de forma de disponer de la matriz en el formato y almacenamiento necesario para la siguiente etapa. En nuestro caso, y debido a que la biblioteca utilizada en la etapa de factorización no lo requiere, no se establece ningún orden más que el dado por la agrupación de los coeficientes de las mismas filas.

La última etapa es la que se encarga de la factorización y resolución del sistema de ecuaciones propiamente dicho. En nuestro caso se utilizó la versión serial de la biblioteca de uso libre MUltifrontal Massively Parallel sparse direct Solver versión 4.3 (MUMPS) [2,3] , que utiliza el paradigma de los métodos multi-frontales[5]. La biblioteca permite optar entre diferentes estrategias de ordenamiento de los pivotes de forma de mejorar el tiempo de cálculo y de minimizar los errores numéricos.

Encontrar un ordenamiento óptimo de los pivotes es un problema NP-difícil, por lo que las distintas estrategias se basan en heurísticas que intentan lograr una solución de compromiso entre el tiempo de cálculo de la heurística, los posibles errores numéricos y la minimización del fill-in, es decir la aparición de nuevos coeficientes no nulos en la matriz durante su factorización. El tema es sumamente amplio y en nuestro trabajo no se profundiza, existe gran cantidad de literatura disponible sobre el tema [4,7].

Durante sus diferentes etapas, la biblioteca MUMPS utiliza las rutinas de los Basic Linear Algebra Subprograms (BLAS), en nuestro caso se utilizó la implementación de Kazushige Goto [14], versión 0.97, multithreads para procesadores pentium IV con 512 Kb de Cache bajo sistemas operativos Linux.

5 CASO DE PRUEBA Y RESULTADOS

Las pruebas se realizaron sobre el modelado hidrodinámico tridimensional baróclino del Río de la Plata utilizando dos grillas distintas para la ejecución del RMA 10 en su modo integrado en vertical. Se calcula el campo de velocidades y el campo de salinidad. El dominio modelado se extiende desde la plataforma oceánica hasta la parte interior del Río de la Plata.

El primer sistema (S1) se define por 1630 elementos y 5113 nodos, que implica un máximo

de 30787 ecuaciones presentes en los sistemas lineales. Mientras que el sistema dos (S2) utiliza 4578 elementos y 9950 nodos con 39759 ecuaciones. En la Figura 2 se puede observar la grilla que define el problema S1. En todos los casos se configuró el sistema para que realice como máximo 6 pasos en la iteración de Newton-Raphson .

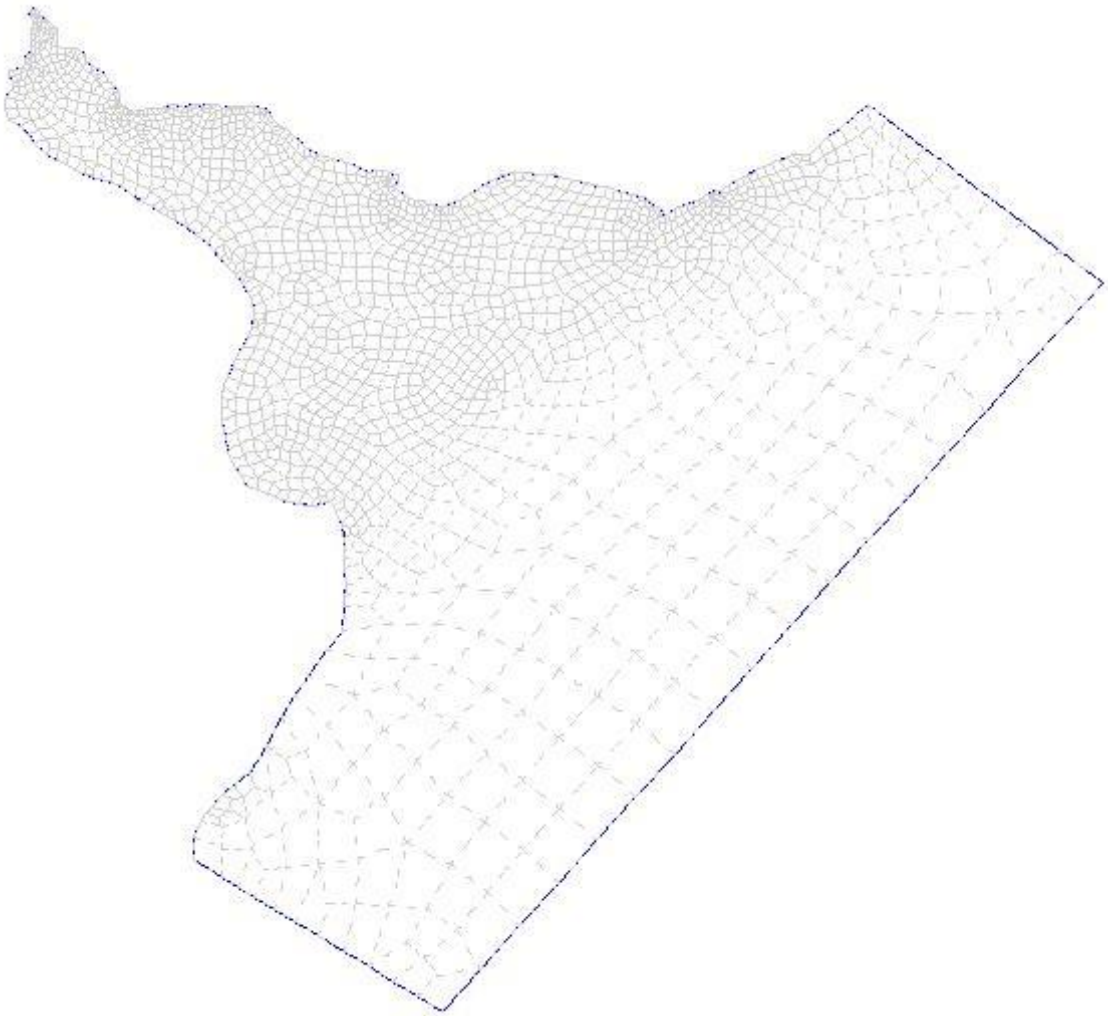

Figura 2. Imagen de la grilla utilizada.

Todos los cálculos se realizaron en una máquina Dell con dos procesadores Pentium IV de 1.8 GHz, que posee 1 Gb de memoria RAM. Se ejecutó sobre sistema operativo Linux. El compilador utilizado fue el Intel FORTRAN 7.0.

En la Tabla 3 se pueden observar los tiempos de ejecución, para el sistema S1 de la versión original y la versión utilizando la rutina FRONT propuesta, para una simulación de 576 pasos de tiempo que equivalen a 192 horas (8 días) de simulación. Mientras que las diferencias entre los distintos resultados de las simulaciones de cinco puntos distribuidos en distintas zonas de la

grilla se puede observar en la Tabla 4. Es de alta importancia obtener resultados numéricos similares dado que las matrices presentes en el modelo son altamente mal condicionadas.

Versión	Tiempo (minutos)
Original	1578,6
Propuesta	450,7

Tabla 3. Comparativo de tiempos de la versión original y la propuesta.

Nodo -Versión	Vel. Total	Elevación	Salinidad
Nodo 1519 – Original	0.111316	0.784859	33.9156
Nodo 1519 – Propuesto	0.111293	0.784851	33.9156
Nodo 2421 – Original	-0.395645	0.885763	33.6970
Nodo 2421 – Propuesto	-0.395656	0.885757	33.6970
Nodo 3021 – Original	0.365118	1.19102	23.4713
Nodo 3021 – Propuesto	0.365115	1.19102	23.4713
Nodo 3215 – Original	-1.09168	0.888560	22.4547
Nodo 3215 – Propuesto	-1.09166	0.888556	22.4547
Nodo 4226 – Original	0.567655	2.08858	0.179923E-04
Nodo 4226 – Propuesto	0.567654	2.08858	0.179923E-04

Tabla 4. Resultados numéricos de la versión original y propuesta.

El estudio de la Tabla 4 nos permite observar que ambas versiones obtuvieron resultados sumamente similares, ya que las diferencias nunca superaron el 0,01 %. Mientras que la versión propuesta, como se puede observar en la Tabla 3 presenta, en cuanto a tiempo de cómputo, un desempeño sumamente mejor logrando efectuar la misma simulación en casi un cuarto del tiempo necesario en la versión original.

Parte	Tiempo (seg)
Cálculo de coeficientes	3.53
Reordenamiento	0.22
Resolución	4.88
Total	8.63

Tabla 5. Tiempo de cálculo de las distintas etapas de la rutina FRONT propuesta.

La Tabla 5 presenta el estudio de los tiempos de cómputos de la nueva rutina FRONT. Se presentan los costos de cada una de las tres etapas de la nueva propuesta sobre el sistema S1.

Versión original	**Versión propuesta**

Ecuaciones	Tiempo (seg)	Ecuaciones	Tiempo (seg)
1774	0.550	1774	0.590
4485	0.960	4485	1.030
6460	1.060	6460	1.070
13666	4.030	13666	2.650
13690	4.020	13690	2.680
14933	4.090	14933	2.760
16464	5.410	16264	3.990
18586	8.570	18003	4.470
21010	10.800	21020	5.820
24759	16.180	24241	6.590
26290	29.410	26342	7.900
27102	32.110	27108	8.210
27415	32.840	27400	8.140
29370	33.620	29462	8.430
29761	34.130	29806	8.460
30787	34.320	30787	8.630

Tabla 6. Tiempo de cómputo al aumentar la cantidad de ecuaciones de la rutina FRONT original y propuesta

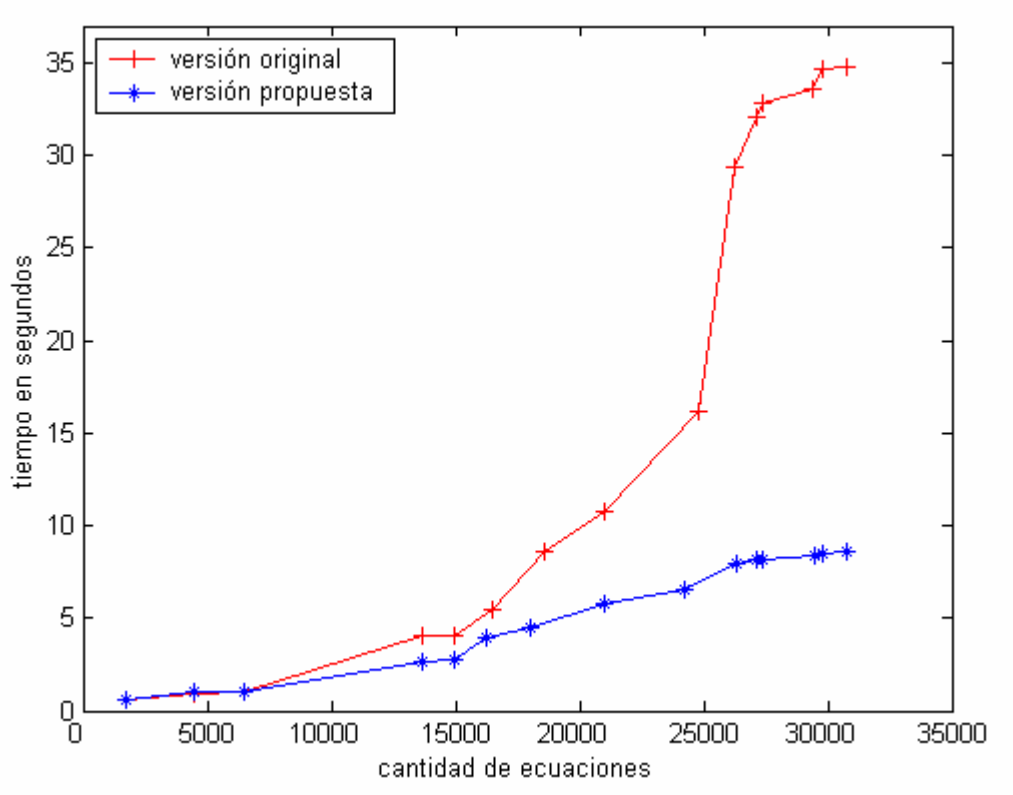

Figura 3. Tiempo de cómputo al aumentar la cantidad de ecuaciones de la rutina FRONT original y propuesta.

La Tabla 6 y la Figura 3 muestran la evolución del tiempo de ejecución de la rutina FRONT original y la propuesta con respecto al número de ecuaciones a resolver sobre el sistema S1. El resultado que se obtiene es sumamente alentador con respecto a la rutina propuesta, pues cuando se resuelven pocas ecuaciones, menos de 7000, el desempeño de la versión original es igual o superior al de la propuesta, sin embargo, a medida que aumentan las ecuaciones a resolver, las diferencias de tiempo de ejecución a favor de la rutina propuesta aumentan hasta llegar a su punto máximo para el caso del mayor sistema evaluado, de 30787 ecuaciones. Siendo en éste último caso la versión original más de 4 veces más lenta con respecto a la versión propuesta.

Versión	Tiempo (seg)
Un-thread	8.63
Dos-threads	8.82

Tabla 7 . Comparativo de tiempos de la versión la propuesta utilizando uno y dos threads.

Debido a que la computadora utilizada dispone de dos procesadores y que la biblioteca BLAS utilizada permite trabajar con multi-threads, se realizó un pequeño estudio de los tiempos de cómputo utilizando la biblioteca BLAS con dos threads. La tabla 7 presenta los tiempos de ejecución de la rutina FRONT propuesta para el sistema S1, en su mayor expresión (30787 ecuaciones) utilizando uno y dos threads. Como se puede observar en la tabla 7, los resultados en cuanto a tiempo de cómputo no presentan mejora alguna, por lo que intentar explotar el paralelismo a este nivel no parece ser un camino interesante. Posiblemente esta situación sea debido a la forma de trabajo de los métodos multifrontales que van resolviendo matrices completas “pequeñas”, por lo que el tiempo ganado en paralelizar dichas tareas se pierde en comunicaciones y sincronizaciones.

La resolución del sistema S2 implica la presencia de matrices con mayor número de ecuaciones que las involucradas en la resolución del sistema S1, sin embargo, la densidad de las matrices es mucho menor. Como ejemplo podemos ver que las matrices de mayor tamaño en el sistema S1 son de 30787 ecuaciones y tienen 2853756 coeficientes no nulos, mientras que las mayores matrices en el sistema S2 son de 39759 ecuaciones y 2332752 coeficientes no nulos. Las cualidades de ambos sistemas nos permiten estudiar la dependencia de los tiempos de cómputos de la rutina FRONT original y de la versión propuesta con respecto al numero de ecuaciones, así como, la dependencia con respecto a la cantidad de coeficientes no nulos presentes en las matrices. A modo de resumen, la tabla 8 presenta dichos resultados.

Sistema-ecuaciones-coef no nulos	Versión	Tiempo (seg)
S1-30787-2853756	Original	34.32
	Propuesta	8.63
S2-39759-2332752	Original	53.93
	Propuesta	6.10

Tabla 8 . Comparativo de tiempos de la rutina FRONT original y la propuesta para los sistemas S1 y S2.

Al observar los resultados de la tabla 8 podemos vislumbrar dos hechos sumamente importantes. El primero, que las diferencias de tiempo de cómputo entre la versión original y la propuesta en el caso del sistema S2 son mucho mayores que las arrojadas para el sistema S1, ya que para el sistema S2 la versión propuesta fue casi nueve veces más rápida que la original. El segundo, es que la versión propuesta obtiene un mejor desempeño en el sistema S2 que en el S1, aún teniendo este segundo un número mayor de ecuaciones. Esto nos permite suponer que el tiempo de cómputo de la versión propuesta no depende fuertemente de la cantidad de ecuaciones sino que se encuentra muy ligada a la densidad de las matrices a resolver, densidad que en general en las matrices vinculadas a los elementos finitos, es baja.

6 CONCLUSIONES

El artículo presenta el estudio de la modificación de la estrategia de resolución de los sistemas lineales presentes en la resolución de sistemas no lineales, mediante el método de Newton-Raphson para el modelo numérico RMA 10, aplicado al modelo del Río de la Plata.

El estudio propone, a diferencia de la versión original, separar la generación de la matriz de rigidez de la resolución propiamente dicha del sistema. Para llevar adelante dicho objetivo se expone la utilización de una estructura auxiliar de tipo tabla de dispersión, de forma de almacenar temporalmente los coeficientes y luego invocar una etapa de reordenamiento y consolidación de coeficientes para generar la matriz en algún formato disperso preestablecido. Por último, se resuelve el sistema de ecuaciones mediante el uso de la librería MUMPS 4.3 en su versión secuencial, utilizando la implementación de la librería BLAS de Kazushige Goto optimizada para la maquina que se utilizó.

Se compararon los resultados numéricos de la simulación en cinco puntos distribuidos en distintas zonas de la grilla obtenidos por ambas versiones. Mostrando la nueva versión comportarse en cuanto a la calidad de los resultados numéricos de igual forma que la versión original ya que las diferencias entre los resultados de ambas nunca superaron el 0.01 %.

En cuanto al desempeño computacional, evaluado en tiempo de ejecución, la versión propuesta mostró una sorprendente superioridad con respecto a la original, pues las pruebas efectuadas mostraron que la versión original necesita casi cuatro veces más de tiempo para efectuar la misma simulación. Mientras que la resolución de sistemas en los cuales la cantidades de coeficientes no nulos son menores, las diferencias de tiempo de cómputo son aún mayores.

El trabajo presenta una mejora del desempeño computacional del RMA 10. Dicha mejora permite ampliar su utilización, posibilitando en tiempos de ejecución razonables, mejorar la definición de las grillas y hacer viables simulaciones de períodos de tiempos que antes eran prohibitivos.

7 TRABAJO FUTURO

Una serie de aspectos sobre la mejora del desempeño computacional del modelo RMA 10 admiten ser explorados con mayor detalle. En la actualidad, dentro del grupo de trabajo se

están investigando algunos de estos aspectos, mientras que se prevé abordar otros en el futuro cercano.

El proceso de generación de la matriz de rigidez puede mejorarse significativamente, entre otras opciones, se ve como interesante la re-utilización de información de pasos anteriores para la ubicación de los coeficientes en la matriz. Notar que, más allá de la eliminación de ecuaciones desacopladas por criterios de convergencia del sistema no lineal, la estructura de las matrices no sufre mayores cambios.

Un tema sumamente promisorio a abordar, en nuestro concepto, es el criterio de ordenamiento de los pivotes al factorizar la matriz. Entre otras opciones, se destacan: la evaluación de distintas técnicas de ordenamiento (minimun degree, nested disection, híbridas), sacar partido de la igualdad de la estructura simbólica de las matrices para calcular el ordenamiento en una única ocasión y permitiendo efectuar un cálculo más riguroso y costoso pero que arroje mejores ordenamientos.

El área de resolución de matrices dispersa es sumamente dinámica y parece interesante el estudio de otras bibliotecas disponibles para el uso público, entre la que resalta la WSMP propuesta por Gupta[8].

Un camino que no hemos explorado es el del cambio de estrategia de resolución de sistema no lineal, ya que si bien es cierto que la mayor parte del tiempo de ejecución es debido a la resolución de los sistemas lineales, no menos cierto es que dichas resoluciones son necesarias por la utilización del método de Newton-Raphson.

Por último, pero no menos interesante, es la aplicación de otras técnicas de paralelismo. Se dispone de diversas opciones de paralelismo: al generar las matrices, a nivel de la resolución de los sistemas lineales, de los sistemas no lineales y descomposición de dominio de los elementos finitos.

8 BIBLIOGRAFÍA

[1] Aho A., Hopcroft J., Ullman J., *Data Structures and Algorithms*, Addison-Wesley (1983).

[2] Amestoy P. R., Duff I. S. and. L'Excellent J-Y, Multifrontal parallel distributed symmetric and unsymmetric solvers, in Comp. Meth. in Appl. Mech. Eng., 184,pp 501-520, (2000).

[3] Amestoy P. R., Duff I. S., Koster J. and L'Excellent J.-Y. A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM Journal of Matrix Analysis and Applications, Vol 23, No 1, pp. 15-41, (2001).

[4] Brainman I. and Toledo S., Nested-Dissection Orderings for Sparse LU with Partial Pivoting SIAM Journal on Matrix Analysis and Applications Volume 23 , Issue 4, pp. 998 – 1012, (2001).

[5] Duff I.and Reid J. K. , The multifrontal solution of indefinite sparse symmetric linear equations. ACM Transactions on Mathematical Software, 9 pp. 302-325, (1983).

[6] Fossati, M. and Piedra-Cueva, I., Modelación numérica del frente salino en el Río de la Plata. XXI Congreso Latinoamericano de Hidráulica, San Pablo, Brasil, (2004).

[7] George A. and Liu J., The evolution of the Minimum Degree Ordering Algorithm, SIAM Review, 31, pp. 1-19, (1989).

[8] Gupta A., Recent advances in direct methods for solving unsymmetric sparse systems of linear equations, ACM Transactions on Mathematical Software, Volume 28 , Issue 3 pp. 301 – 324, (2002).

[9] Hood P., Frontal solution program for unsymmetric matrices. Int. J. Num. Methods Eng. 10, pp. 379-400, (1976).

[10] Irons, B.M., A frontal solution program for finite element analysis, International Journal for Numerical Methods in Engineering, 2, pp. 5-32, (1970).

[11] King, I.P., RMA-10, A Finite Element Model for Three-Dimensional Density Stratified Flow. Report prepared in co-operation with Australian Water and Coastal Studies for the Sydney Deepwater Outfalls Environmental Monitoring Program Post Commissioning Phase, (1993).

[12] Rueda, F. J. and Schladow, S. G., Quantitative comparison of models for the barotropic response of homogeneous basins. ASCE, J. Hyd. Engineering, 128(2), pp. 201-213, (2002).

[13] Warner, J.C., Barotropic and baroclinic convergence zones in tidal channels: Dissertation, Department of Civil and Environmental Engineering, University of California, Davis, Calif, (2000).

[14] www.cs.utexas.edu/users/flame/goto/, consultada agosto 2005.

[15] www.bossintl.com/html/rmagen-features.html, consultada agosto 2005.