Centro de Investigación en Matemáticas, A.C.

Algoritmo paralelo para solución de ecuaciones diferenciales

TESIS

Que para obtener el grado de

Maestro en Ciencias

con Especialidad en

Computación y Matemáticas Industriales

P R E S E N T A:

Maria Trinidad Pimentel Villegas

Directores de Tesis:

Dr. Miguel Ángel Moreles Vázquez

Dr. Joaquín Peña Acevedo

Diciembre del 2015 *Guanajuato, Gto. México*

Centro de Investigación en Matemáticas, A.C.

Algoritmo paralelo para solución de ecuaciones diferenciales

TESIS

Que para obtener el grado de

Maestro en Ciencias

con Especialidad en

Computación y Matemáticas Industriales

P R E S E N T A:

Maria Trinidad Pimentel Villegas

Comité de evaluación

Dr. Salvador Botello Rionda

Dr. José Antonio Muñoz Gómez

Director de Tesis *Co-Director de Tesis*

Dr. Miguel Ángel Moreles Vázquez

Dr. Joaquín Peña Acevedo

Diciembre del 2015 *Guanajuato, Gto. México*

A mis padres
Luis y Maria Elena

A mis hermanas
Elena, Mary y Lucy

Dios no pudo darme mayor bendición
que pertenecer a esta familia.

Agradecimientos

Quiero agradecer antes que nada a mis asesores el **Dr. Joaquín Peña Acevedo** y el **Dr. Miguel Ángel Moreles Vázquez** de verdad aprecio su paciencia y su disposición al siempre apoyarme en todo lo que se me dificultaba, gracias por permitir aportar y aprender un poco de lo mucho que ustedes hacen. Toda mi admiración para ustedes que realizan con tanto gusto su trabajo. De igual forma agradezco a mis sinodales el Dr. Salvador Botello Rionda y el Dr. José Antonio Muñoz Gómez por aceptar ser parte del comité evaluador de mi tesis.

Agradecer además a **CIMAT** por tener el mejor ambiente de trabajo en el que yo he estado y siempre brindarme todo lo necesario para cursar mi Maestría. Gracias también a todos mis profesores(as)/investigadores por poner tanto empeño en transmitir conocimientos, así mismo agradecer a los ayudantes de las materias que curse por brindarnos su tiempo y orientarnos cuando los necesitamos. Agradezco en especial al **Dr. Johan Van Horebeek** por permitirme ser parte del club de ciencias de CIMAT, que ha dejado en mi una grata experiencia, mucho aprendizaje y una gran motivación.

Todo este trabajo no hubiera sido posible sin el valioso apoyo del Consejo Nacional de Ciencia y Tecnología **CONACYT** cuyo apoyo economico fue mi sustento durante mi estancia en CIMAT.

Sin duda alguna agradezco inmensamente a mi familia, a mis padres **Luis Pimentel** y **Maria Elena Villegas**, a mis hermanas **Elena, Mary** y **Lucy** ustedes son lo máximo, gracias por todo su amor y apoyo incondicional.

Y no pueden faltar por ningún motivo mis compañeros de Maestría: **Rosy, Rubén, Orlando, Xavi, Odin, Iván, Ernesto** y **Jony**, gracias por su valioso apoyo en todo y por todos los momentos divertidos y amenos que pasamos, nada hubiera sido lo mismo sin ustedes.

Gracias a todos los amigos y personas valiosas que conocí en mi estancia en CIMAT, todos contribuyeron de alguna manera a este trabajo.

Finalmente agradezco a Dios por permitirme concluir esta etapa de mi vida que ha marcado tantas cosas en mi y a dejado la mas gratas experiencias.

...

Resumen

Algoritmo paralelo para solución de ecuaciones diferenciales.

Maria Trinidad Pimentel Villegas

El presente trabajo de tesis tiene como objetivo explicar y mostrar las ventajas de la implementación de un algoritmo del tipo predictor-corrector o de corrección integral diferida denominado RIDC (Revisionist integral deferred correction), que combina la eficiencia de este tipo de algoritmos para encontrar soluciones numéricas precisas con el hecho de utilizar un esquema de cómputo paralelo planteado de forma natural para el algoritmo, el cual esta integrado con el esquema de solución de la ecuación diferencial. A diferencia de otros algoritmos donde la paralelización es agregada como una mejora y esta puede o no estar presente para la solución del problema, es decir, la paralelización solo es utilizada para acelerar alguna cálculo propio del algoritmo (por ejemplo, la solución de un sistema de ecuaciones).

Este algoritmo permite realizar cálculos numéricos en tiempos de cómputo más cortos debido a que utiliza tamaños de paso grandes para calcular la solución numérica, de tal forma que tenemos precisión y rapidez.

Analizaremos el comportamiento y las ventajas del algoritmo para encontrar soluciones tanto en ecuaciones diferenciales ordinarias con discretización en el tiempo, como en ecuaciones diferenciales parciales con discretizaciones en el tiempo y en el espacio.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Preliminares

1.1. Introducción

El presente trabajo de tesis tiene como objetivo explicar y mostrar las ventajas de la implementación de un algoritmo del tipo predictor-corrector o de corrección diferida planteado en [1], que combina las noblezas de este tipo de algoritmos como el hecho de encontrar soluciones precisas y utilizar un esquema de cómputo paralelo que está planteado de forma natural para el algoritmo, es decir, esta integrada con el esquema de solución de la ecuación diferencial. A diferencia de otros algoritmos en donde la paralelización es agregada como una mejora y esta puede o no estar presente para la solución del problema, es decir, la paralelización solo es utilizada para acelerar algún cálculo propio del algoritmo (por ejemplo, la solución de un sistema de ecuaciones). Este esquema permite realizar los cálculos numéricos en un tiempo de cómputo más corto debido a que se utilizan tamaños de paso (Δt) más grandes para calcular la solución, de tal forma que tenemos precisión y rapidez.
Analizaremos el comportamiento y las ventajas del algoritmo para encontrar soluciones tanto en ecuaciones diferenciales ordinarias con discretización en el tiempo, como en ecuaciones diferenciales parciales con discretizaciones en el tiempo y en el espacio.

1.2. Motivación

En general el estudio y solución numérica de modelos con ecuaciones diferenciales siempre ha sido un reto sobre todo en lo que a recursos computacionales se refiere ya que este tipo de cálculos requieren un tiempo de cómputo bastante considerable con la finalidad de obtener buenas aproximaciones a la solución. Si bien existen métodos que nos permiten llegar a soluciones muy precisas para resolver problemas que requieren de una gran cantidad de cálculos, estas soluciones pueden tener un alto costo computacional tanto en recursos como en tiempo de ejecución, tomando como referencia que cuando hablamos de tiempo de ejecución nos podemos referir a horas, días, meses e incluso más, dependiendo de la complejidad que tenga el problema, los recursos (hardware) que se tengan o de la cantidad de datos que se procesa.
El algoritmo planteado en esta tesis combina un método del estado del arte que por su esquema de operación permite de manera natural utilizar técnicas de programación en paralelo para agilizar el tiempo de cálculo de soluciones a ecuaciones diferenciales.

1.3. Ecuaciones Diferenciales

Una ecuación diferencial es una ecuación que relaciona a una función de una o varias variables con sus derivadas. La derivada de mayor orden determina el orden de la ecuación diferencial.

1.3.1. Ecuaciones Diferenciales Ordinarias

Si la solución de una ecuación diferencial es una función de una sola variable, entonces decimos que la ecuación es ordinaria. De manera general, una ecuación ordinaria es de la forma

$$\frac{d^n y}{dx^n} = f(x,\, y(x),\, y'(x), ...,\, y^{(n-1)}(x)\,). \tag{1.1}$$

También podemos tener un sistema de ecuaciones diferenciales ordinarias. En este caso tenemos n funciones $y_1(x), y_2(x), ..., y_n(x)$ tales que

$$\begin{aligned}
\frac{dy_1}{dx} &= f_1(x,\, y_1(x),\, y_2(x),\, \dots, y_n(x)\,) \\
\frac{dy_2}{dx} &= f_2(x,\, y_1(x),\, y_2(x),\, \dots, y_n(x)\,) \\
\vdots \;&\; \vdots \;\; \vdots \\
\frac{dy_n}{dx} &= f_n(x,\, y_1(x),\, y_2(x),\, \dots, y_n(x)\,)
\end{aligned}$$

Escrito de manera más compacta,

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(x, \boldsymbol{y}(x)), \quad \text{con} \quad \boldsymbol{y}(x) = (y_1(x),\, y_2(x),\, \dots, y_n(x)),$$

Así, $\boldsymbol{y} : \mathbb{R} \to \mathbb{R}^n$, y $\boldsymbol{f} : \mathbb{R}^{n+1} \to \mathbb{R}^n$.

Hay que especificar el intervalo en que queremos encontrar la solución. También necesitamos una condición auxiliar para encontrar la solución del problema. Un *problema de valor inicial* (PVI) para una ecuación diferencial de primer orden:

$$\begin{aligned}
\tfrac{d\boldsymbol{y}}{dx} &= \boldsymbol{f}(x, \boldsymbol{y}(x)), \qquad x \in [x_0, x_1] \\
\boldsymbol{y}(x_0) &= \boldsymbol{y}_0.
\end{aligned} \tag{1.2}$$

Una ecuación de orden n la podemos transformar en un sistema de ecuaciones de primer orden introduciendo variables al problema:

$$y_1(x) = y(x), \qquad y_{i+1}(x) = y^{(i)}(x), \quad i = 1, ..., n-1.$$

$$y_1(a) = y(a), \quad \cdots \quad , y_{n-1}(a) = y^{(n-1)}(a)$$

Un problema de valor inicial (o problema de Cauchy) es de la forma

$$\begin{cases}
\frac{dy}{dx} &= f(x,y) \qquad x \in (a,b], \\
y(a) &= y_0.
\end{cases}$$

La solución numérica de este tipo de problemas se puede obtener haciendo una discretización del intervalo y el valor de la aproximación en un nodo se calcula a partir de los valores en nodos anteriores.

Otro tipo de problemas de interés son los problemas de valores en la frontera:

$$\begin{cases} \frac{dy}{dx} &= f(x,y) \qquad x \in (a,b), \\ y(a) &= \alpha, \\ y(b) &= \beta. \end{cases}$$

Para resolver numéricamente este tipo de problemas hay que establecer relaciones entre los valores en los nodos y resolver un sistema de ecuaciones.

1.3.2. Ecuaciones Diferenciales Parciales

Una ecuación diferencial parcial (EDP) es una ecuación que relaciona las derivadas parciales de una función. La variable dependiente (función desconocida) en la EDP es función de al menos dos variables independientes. El orden de la EDP es igual al orden de la derivada parcial de mayor orden que aparece en la ecuación. Son ampliamente utilizadas para modelar fenómenos físicos.

Ejemplo.

Sea $u : \Omega^2 \to \mathbb{R}$ una función de dos variables. La forma general de una EDP de segundo orden es

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + F = 0. \tag{1.3}$$

donde $A, B, C, D, E, F \in \mathbb{R}$. Se dice que esta ecuación es:

- *Elíptica* si $AC - B^2 > 0$.
  Ejemplo: la ecuación de Poisson

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = g(x,y)$$

- *Parabólica* si $AC - B^2 = 0$.
  Ejemplo: la ecuación de calor

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + g(x)$$

- *Hiperbólica* si $AC - B^2 < 0$.
  Ejemplo: la ecuación de onda

$$\frac{\partial^2 u}{\partial t^2} = a^2\frac{\partial^2 u}{\partial x^2} + g(x)$$

Falta agregar condiciones de frontera a los problemas, y para el caso de los problemas parabólicos e hiperbólicos hay que dar una condición inicial.

1.3.3. Condiciones Iniciales y de Frontera

Un aspecto importante que debemos considerar al querer resolver cualquier modelo de ecuaciones diferenciales es tener bien especificado cuales serán las condiciones iniciales y/o las condiciones de frontera, ya que el resultado de nuestra solución depende totalmente de estos valores.
Establecer de manera adecuada las condiciones tanto iniciales y de frontera determina la solución, como mencionamos anteriormente la solución calculada con base en cualquier método debe satisfacer tanto a las condiciones (iniciales y/o de frontera) como a la ecuación diferencial planteada.

En el caso de las EDP y de acuerdo a [7] si se especifican muchas condiciones (iniciales y de frontera) puede que el problema no tenga solución, en cambio si se especifican muy pocas la solución puede no ser única. Si las condiciones son especificadas en el lugar incorrecto o en el tiempo incorrecto entonces la solución puede no depender de las condiciones y podemos obtener grandes cambios en la solución, a esto se le conoce como un problema *mal condicionado*.

1.3.3.1. Condiciones de frontera Dirichlet

Este tipo de condiciones se establecen cuando en una ecuación diferencial ordinaria o parcial, se le especifican los valores de solución que necesita la frontera del dominio.

En caso de una EDO como (1.1), sobre un intervalo de $[a, b]$ las condiciones de frontera Dirichlet serian:

$$\begin{cases} y(a) = & \alpha_a \\ y(b) = & \alpha_b, \end{cases} \tag{1.4}$$

donde α_a y α_b son valores conocidos.

Mientras que para una EDP como (1.3) serian:

$$u(x,t) = f(x,t) \qquad \forall x \in \partial\Omega \qquad t \in (0,T] \tag{1.5}$$

donde f es una función conocida definida en $\partial\Omega$.

1.3.3.2. Condiciones de frontera Neumann

Este tipo de condiciones se establecen cuando en una ecuación diferencial ordinaria o parcial, se le especifican los valores de la derivada de una solución tomada sobre la frontera.

En caso de una EDO como (1.1), sobre un intervalo de $[a, b]$ las condiciones de frontera Dirichlet serian:

$$\frac{dy}{dx}(a) = \alpha_a \tag{1.6}$$

donde α_a es un valor conocido.

Mientras que para una EDP como (1.3) sobre un dominio $\Omega \subset \mathbb{R}^n$ serian:

$$\frac{\partial u}{\partial n}(x,0) = f(x) \qquad \forall x \in \partial\Omega \tag{1.7}$$

donde n es la normal a la frontera $\partial\Omega$ y f es una función escalar.

1.3.4. Solución de ecuaciones diferenciales

Cuando hablamos de solución de ecuaciones diferenciales en general nos referimos a encontrar el valor de la función U que satisface la ecuación diferencial (Ordinaria o Parcial) y que además satisface la condiciones iniciales y/o de frontera. Existen métodos que nos permiten encontrar soluciones analíticas a algunos modelos de ecuaciones diferenciales, pero desafortunadamente no todos los modelos tienen una

solución exacta por lo que debemos recurrir a soluciones numéricas cuyo resultado es una aproximación a la solución exacta.
El trabajo que realizamos en esta tesis utilizamos un esquema de diferencias finitas que nos permite obtener las aproximaciones de las soluciones a las ecuaciones diferenciales que queremos resolver.
La idea de este tipo de esquemas es realizar una discretización en la variable independiente de tal manera que se pueda resolver con el uso de una computadora.

1.3.5. Estimación del error en la solución aproximada

Una forma de validar que el algoritmo encuentre la solución al modelo que estamos resolviendo, es haciendo comparaciones de la solución numérica aproximada (que es el resultado una vez que se aplicó el método) con la solución analítica en caso de ser conocida, usaremos la diferencia entre las dos soluciones (aproximada y exacta) para determinar el *error numérico*. En particular para los ejemplos mostrados en esta tesis usaremos las siguientes métricas: Error Relativo, error absoluto y error cuadrático medio (*véase apéndice C*), los cuales nos permiten saber que tan cerca esta la solución numérica de la solución analítica del problema que queremos resolver.

1.4. Aproximación numérica de derivadas

Sea $F(x)$ una función con n derivadas continuas en el intervalo $[a, b]$. Entonces para $a < x_0, x_0 + h < b$.

$$F(x_0 + h) = F(x_0) + hF'(x_0) + h^2\frac{F''(x_0)}{2!} + ... + h^{n-1}\frac{F^{n-1}(x_0)}{(n-1)!} + O(h^n) \quad (1.8)$$

donde $F(x_0)$ representa el valor de la función evaluada en x_0, $F'(x_0)$ representa el valor de la derivada evaluada en x_0 y $O(h^n)$ representa términos de orden h^n o superior.

La interpretación del teorema de Taylor dice que si nosotros conocemos los valores de F y los valores de sus derivadas en el punto x_0 entonces podemos usar la ecuación (1.8) para aproximar el valor de F en el punto $x_0 + h$, cometiendo un error de orden $O(h^n)$.

Por ejemplo, si tomamos únicamente los términos hasta la primer derivada tendríamos

$$F(x_0 + h) = F(x_0) + hF'(x_0) + O(h^2). \tag{1.9}$$

Si despejamos de tal forma que nuestra incógnita sea la primera derivada obtenemos lo siguiente

$$F'(x_0) = \frac{F(x_0 + h) - F(x_0)}{h} - O(h). \tag{1.10}$$

Sin considerar el error tenemos la aproximación a la primer derivada

$$F'(x_0) \approx \frac{F(x_0 + h) - F(x_0)}{h}. \tag{1.11}$$

La ecuación (1.11) es una aproximación por diferencias finitas hacia adelante de *primer orden* a la derivada $F'(x_0)$.

Es posible desarrollar versiones similares para las formulas hacia atrás y centradas, así como para las aproximaciones de derivadas de orden superior.

Formulas de diferencias finitas hacia adelante

Primera derivada	*Error*
$F'(x_0) \approx \frac{F(x_0+h)-F(x_0)}{h}$	$O(h)$
$F'(x_0) \approx \frac{-F(x_0+2h)+4F(x_0+h)-3F(x_0)}{2h}$	$O(h^2)$
Segunda derivada	
$F''(x_0) \approx \frac{F(x_0+2h)-F(x_0+h)+F(x_0)}{h^2}$	$O(h)$
$F''(x_0) \approx \frac{-F(x_0+3h)+4F(x_0+2h)-5F(x_0+h)+2F(x_0)}{h^2}$	$O(h^2)$

Formulas de diferencias finitas hacia atrás

Primera derivada	*Error*
$F'(x_0) \approx \frac{F(x_0)-F(x_0-h)}{h}$	$O(h)$
$F'(x_0) \approx \frac{3F(x_0)-4F(x_0-h)+F(x_0-2h)}{2h}$	$O(h^2)$
Segunda derivada	
$F''(x_0) \approx \frac{F(x_0)-2F(x_0-h)+F(x_0-2h)}{h^2}$	$O(h)$
$F''(x_0) \approx \frac{2F(x_0)-5F(x_0-h)+4F(x_0-2h)-F(x_0-3h)}{h^2}$	$O(h^2)$

Formulas de diferencias finitas centradas

Primera derivada	*Error*
$F'(x_0) \approx \frac{F(x_0+h)-F(x_0-h)}{2h}$	$O(h^2)$
$F'(x_0) \approx \frac{-F(x_0+2h)+8F(x_0+h)-8F(x_0-h)+F(x_0-2h)}{12h}$	$O(h^4)$
Segunda derivada	
$F''(x_0) \approx \frac{F(x_0+h)-2F(x_0)+F(x_0-h)}{h^2}$	$O(h^2)$
$F''(x_0) \approx \frac{-F(x_0+2h)+16F(x_0+h)-30F(x_0)+16F(x_0-h)-F(x_0-2h)}{12h^2}$	$O(h^4)$

Para todos los casos la aproximación de la derivada es mas exacta cuanto mayor sea el orden del error $O(h^n)$ [6].

Otra forma de obtener aproximaciones de la derivada de la función en un punto es mediante interpolación. Supongamos que tenemos $x_0 < x_1 < ... < x_n$ puntos en donde conocemos los valores $f(x_i)$ de la función. Entonces

$$f(x) \approx \sum_{j=0}^{n} f(x_j) L_{n,j}(x),$$

donde $L_{n,j}(x)$ es el polinomio de Lagrange de grado n

$$L_{n,j}(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i}.$$

Entonces una aproximación de la derivada de f en el punto x_k es

$$f'(x_k) = \sum_{j=0}^{n} f(x_j) L'_{n,j}(x_k)$$

A esta aproximación se le conoce como la aproximación de la derivada de f con $n+1$ puntos. En particular se puede escoger los nodos x_k equiespacios, con h como distancia entre ellos, y dependiendo de la elección del punto x_k se puede tener fórmulas de derivadas hacia adelante, hacia atrás, centrales, etc.

1.5. Método de Diferencias Finitas

La idea principal de trabajar en un esquema por diferencias finitas se basa en hacer una discretización del dominio de la ecuación y los valores de la derivada de una función. Cada punto de la discretización es calculado con los valores de sus vecinos.

Si tenemos el siguiente problema de valor inicial y valores en la frontera de [15]

$$\begin{aligned} u_t &= v u_{xx}, & x \in (0,1), t > 0 \\ u(x,0) &= f(x), & x \in [0,1] \\ u(0,t) &= a(t), & u(1,t) = b(t), t \geq 0, \end{aligned} \tag{1.12}$$

donde $f(0) = a(0)$ y $f(1) = b(0)$.

Queremos encontrar la solución numérica a este problema reduciéndolo a su forma discreta. Iniciaremos discretizando el dominio en el espacio de esta forma si queremos referirnos a un punto en el espacio, utilizaremos el valor de x_k donde $k = 0, ..., M$ para $x_k = k\Delta x$. De esta misma forma discretizamos el dominio en el tiempo con t_n donde $n = 0, ..., N$ para $t_n = n\Delta t$, así tendremos un mallado donde un eje representa la discretización en el espacio (x) y el otro representa la discretización en el tiempo (t). Como se muestra en la *figura* (1.1).

Podemos aproximar la solución al problema de interés en cada punto de la malla. Podemos definir a u_k^n como incógnita en el punto $(k\Delta x,\ n\Delta t)$ o (k,n) de la malla. La variable u_k^n será nuestra aproximación a la solución del problema (1.12) en el punto $(k\Delta x,\ n\Delta t)$.

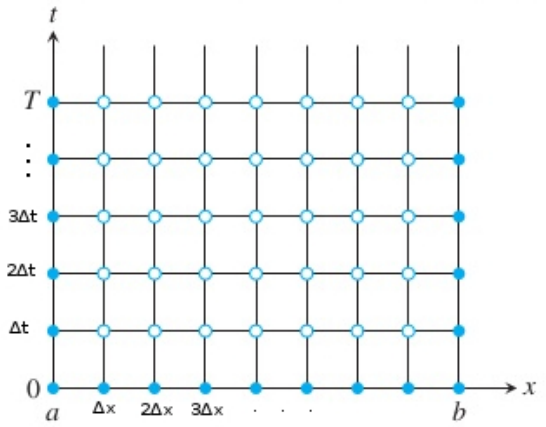


FIGURA 1.1: Representación de la discretización en tiempo y espacio.

Como ya tenemos un mallado que aproxima nuestro dominio, el siguiente paso es utilizarlo para aproximar la solución a (1.12). Notemos que necesitamos calcular el valor de la derivadas u_t y u_{xx}, utilizaremos las aproximaciones del tema anterior para definirlas, comenzaremos utilizando

$$u_t(k\Delta x, n\Delta t) = \frac{u_k^{n+1} - u_k^n}{\Delta t}. \tag{1.13}$$

Que como vimos anteriormente se trata de la aproximación por diferencias finitas hacia adelante de la primera derivada, mientras que para

$$u_{xx}(k\Delta x, n\Delta t) = \frac{u_{k+1}^n - 2u_k^n + u_{k-1}^n}{\Delta t^2}, \tag{1.14}$$

tomamos la aproximación de la segunda derivada por diferencias finitas centradas. Tanto (1.13) como (1.14) serán utilizadas para aproximar la ecuación diferencial parcial en el punto $(k\Delta x, n\Delta t)$ por

$$\frac{u_k^{n+1} - u_k^n}{\Delta t} = v\frac{u_{k+1}^n - 2u_k^n + u_{k-1}^n}{\Delta t^2} \tag{1.15}$$

Y finalmente vemos que la condición inicial y las condiciones de frontera son aproximadas por

$$u_k^0 = f(k\Delta x), k = 0, ..., M \tag{1.16}$$

$$u_0^{n+1} = a((n+1)\Delta t), n = 0 \tag{1.17}$$

$$u_M^{n+1} = b((n+1)\Delta t), n = 0 \tag{1.18}$$

Podemos obtener una aproximación al problema (1.12) resolviendo el problema discreto (1.15)-(1.18). Debemos escoger un Δx y Δt apropiados. Vemos que la ecuación (1.16) da u_k^0 para $k = 0, ..., M$, la ecuación (1.15) con $n = 0$ puede utilizarse para determinar u_k^1 para $k = 1, ..., M-1$ y finalmente las ecuaciones (1.17) y (1.18) pueden usarse para determinar u_0^1 y u_M^1. Entonces las ecuaciones (1.16), (1.17) y (1.18) utilizan la información en $n = 0$ para determinar u en el primer paso en el tiempo. Ahora que u_k, $k = 0, ..., M$ es conocido, las ecuaciones (1.16), (1.17) y (1.18) pueden usarse para determinar u en $n = 2$ y por supuesto este proceso puede continuarse para determinar u_k, $k = 0, ..., M$ en cualquier nivel n de tiempo.

Cabe señalar que no es posible calcular u_0^1 y u_M^1 puesto que $k+1$ o $k-1$ no estarían dentro de nuestro intervalo por lo tanto debemos de tener algún valor en nuestras fronteras. Inclusive como muestra la *figura* (1.1) los nodos de la malla que están vacíos son los que podemos calcular, mientras que los que están rellenos son nodos con valores que ya conocemos (ya sean valores iniciales y/o de frontera).

Ahora ya tenemos un esquema para aproximar la solución a un problema de valor inicial y valores en la frontera (1.15). Llamaremos a este *esquema explicito* debido a que resolvemos para la variable en nivel de tiempo $n + 1$, explícitamente.

1.6. Cómputo paralelo

El cómputo paralelo ha tenido un gran auge en las ultimas décadas, sin duda alguna el hecho que aprovechar de manera eficiente el hardware de las computadoras propicia que cada vez se puedan resolver problemas más y más complejos, a veces pareciera que es una competencia entre problema y solución, es decir, cuando se tiene una solución a un problema ya tenemos otro más complejo esperando por una nueva.

Y ciertamente el resolver ecuaciones diferenciales numéricamente siempre ha sido un problema de importante costo computacional, de tal forma que se implementan algoritmos que puedan trabajar en paralelo resolviendo dichas ecuaciones, estos algoritmos son utilizados generalmente para resolver los sistemas de ecuaciones que se generan al querer resolver la ED, esto con la finalidad de reducir los tiempos de cómputo.

El cómputo paralelo implementado de manera adecuada puede traer múltiples beneficios a los programas en general. Aunque su implementación no necesariamente es sencilla como se menciona en [17] "no se puede tener código en paralelo eficiente sin antes haber optimizado el código en serie", y optimizar un código en serie tiene sus propias complicaciones principalmente por que debemos tener muy claro como accesar de manera eficiente a la memoria, esto representa un reto para toda persona que quiera paralelizar algoritmos, aunque existen algunas soluciones a este tipo de problemas como buscar algoritmos que sean Cache-Oblivious, los cuales propiamente están diseñados para aprovechar de manera eficiente la memoria cache agilizando el acceso a la memoria Ram.

Además de optimizar el código en serie como se mencionó con anterioridad existen esquemas para paralelizar programas; uno de ellos es el *esquema de OpenMP*: se trata de un esquema con memoria compartida, es utilizado para paralelizar programas escritos en C, C++ o fortran en computadoras multicore, utiliza códigos sencillos para paralelizar algoritmos y tiene soporte en compiladores GNU, IBM, Oracle, Intel, PGI, Texas Instruments, entre otros. La idea principal de este esquema es el uso de hilos (threads) que realizan tareas simultáneas aprovechando en mayor medida la capacidad de hardware del equipo de cómputo. En esta tesis trabajamos de manera particular con este esquema.

Otro esquema es *MPI* ó Interfaz de paso de mensajes (por sus siglas en ingles Message Passing Interface) este esquema utiliza memoria compartida y su comunicación puede ser a través de la red , este esquema es utilizado en clusters de computadoras aprovechando en mayor medida la capacidad de cómputo del mismo.

Existen trabajos diversos trabajos previos donde se utiliza el cómputo paralelo en la solución de ecuaciones diferenciales como [18], que resuelve ecuaciones diferenciales ordinarias ó [16] que presenta toda una serie de técnicas de paralelización para ecuaciones diferenciales parciales. Por otro lado existen iniciativas de proyectos universitarios enfocados a utilizar técnicas con cómputo paralelo para la solución de ecuaciones diferenciales como el *Parallel Solution of Systems of Ordinary Differential Equations by Adaptive Techniques* impulsado por la universidad de Bayreuth en alemania apartir del 2010, por mencionar algunos.

Capítulo 2

Método RIDC

2.1. Introducción

Este método es una modificación(revisión) de la familia de métodos **IDC** (*Integral Deferred Correction*), el método RIDC (Revisionist Integral Deferred Correction) utiliza un esquema *predictor-corrector*, donde la idea principal es *corregir* una solución poco precisa que denominamos *predicción*, la cuál es calculada utilizando un método de menor orden (Euler implícito o explícito), mientras que una *corrección* se realiza resolviendo una ecuación de error que obtenemos partiendo del hecho de que la solución que hemos obtenido en el nivel de predicción tiene un error con respecto a la solución exacta, resolviendo esta ecuación de error obtenemos una nueva solución que llamaremos primera corrección. Volvemos a suponer que tenemos un error en esta primera corrección y se aplicamos nuevamente una ecuación de error cuyo resultado es una solución en el segundo nivel de corrección y así sucesivamente continuamos haciendo correcciones a soluciones previamente calculadas.

Todo lo anterior no solo se hace con la finalidad de incrementar la precisión en la solución, si no además aprovechar el esquema de niveles de predicción-corrección para avanzar con tamaños de paso más grandes, mientras que en paralelo se van realizando las correcciones de los valores calculados en la predicción [1].

2.2. Planteamiento del problema

Sea $u(t,x)$ la solución analítica del problema

$$\begin{aligned} u_t &= f(t,u), && (x,t) \in \Omega \times [0,T], \\ u(t,x) &= h(t,x), && (x,t) \in \partial\Omega \times [0,T], \\ u(0,x) &= g(x), && x \in \Omega, \end{aligned} \tag{2.1}$$

donde Ω es un dominio en $\mathbb{R}^n$, $T > 0$, $\partial\Omega$ se refiere a la frontera de Ω. El problema de interés es encontrar la solución aproximada $\eta(t,x)$ de $u(t,x)$.

2.3. Funcionamiento

En general el método **IDC** consiste, de acuerdo a [3] en suponer que tenemos un dominio en el tiempo $[0,T]$ que está discretizado uniformemente en N intervalos y los nodos resultantes son numerados y agrupados de tal forma que se generan J grupos de M intervalos. Es decir, si $\Delta t = \frac{T}{N}$, entonces los nodos

$$t_n = n\Delta t, \qquad n = 0, ..., N$$

son numerados

$$t_{j,m} = (jM + m)\Delta t, \quad m = 0, ..., M, \quad j = 0, ..., J-1$$

de tal forma que cada grupo contiene $p = M + 1$ nodos,

$$I_j = \{t_{j,0}, t_{j,1}, ..., t_{j,M}\}, \quad j = 0, ..., J-1$$

Este tipo de métodos iteran completamente en cada grupo de intervalos I_j secuencialmente, iniciando con j = 0 y el procedimiento general es como sigue

1. Suponemos que la solución es conocida en $t_{j,0}$. Esto es cierto para $j = 0$ donde $\eta(t_0, x) = g(x)$.

2. Utilizando algún método numérico resolvemos el problema $\eta_t = f(t,\eta)$ para una solución provisional en todos los nodos que contiene I_j. Dicha solución provisional la identificamos como $\eta_j^{m,[0]}$, donde $m = 0, ..., M$.

3. Resolvemos la ecuación de error para obtener una solución corregida en todos los nodos contenidos en I_j. La solución de la primera corrección la identificaremos como $\eta_j^{m,[1]}$, donde $m = 0, ..., M$.

4. Repetimos el paso 3 $(M-1)$ veces, utilizando la solución corregida mas reciente como una aproximación a la solución exacta. La solución después del p-ésimo ciclo de corrección como , $\eta_j^{m,[p]}$, donde $m = 0, ..., M$.

5. Utilizamos la solución actual en $t_{j,M}$ repitiendo el proceso para el siguiente número de intervalos, I_{j+1}. Específicamente hacer $\eta(t_{j+1,0}) = \eta_{j,M}^{[M]}$, $j \leftarrow j+1$ y regresar al paso 1.

La propuesta **RIDC** en [3] no requiere de la iteración de M ciclos de corrección para cada grupo de M intervalos, en lugar de esto cada grupo contiene K intervalos, donde $K >> M$, pero aún se itera solo M veces utilizando una cierta cantidad de puntos para aproximar la cuadratura del residual.

2.4. Ecuación de Error

Como mencionamos con anterioridad existe una ecuación de error que obtenemos calculando una aproximación a la solución $\eta(t, x)$ para la ecuación que queremos resolver. Denotamos a la solución exacta (no necesariamente conocida) como $u(t, x)$. Entonces el error de la solución aproximada es

$$e(t, x) = u(t, x) - \eta(t, x) \tag{2.2}$$

Definimos el residual como $\epsilon(t, x) = \eta_t(t, x) - f(t, \eta(t, x))$, entonces la derivada parcial respecto al tiempo del error satisface

$$e_t(t, x) = u_t(t, x) - \eta_t(t, x) = f(t, u(t, x)) - f(t, \eta(t, x)) - \epsilon(t, x) \tag{2.3}$$

Reescribiendo la ecuación de error obtenemos

$$\frac{\partial}{\partial t}\left[e(t, x) + \int_0^t \epsilon(\tau, x)d\tau\right] = f(t, \eta(t, x) + e(t, x)) - f(t, \eta(t, x)) \tag{2.4}$$

Si definimos la solución aproximada como $\eta^{[p]}(t, x)$, donde p es el nivel de corrección y podemos calcular una corrección $e^{[p]}(t, x)$, la solución corregida en el nivel $p + 1$

quedaría como $\eta^{p+1}(t,x) = \eta^{[p]}(t,x) + e^{[p]}(t,x)$. Sustituyendo en (2.4) la definición del residual obtenemos

$$\frac{\partial}{\partial t}\left[\eta^{[p+1]}(t,x) - \eta^{[p]}(t,x) + \int_0^t [\eta_t^{[p]}(\tau,x) - f(\tau,\eta^{[p]}(\tau,x))]d\tau\right] = f(t,\eta^{[p+1]}(t,x)) - f(t,\eta^{[p]}(t,x)). \tag{2.5}$$

Integrando $\eta_t^{[p]}(\tau,x)$ y simplificando términos

$$\frac{\partial}{\partial t}\left[\eta^{[p+1]}(t,x) - \int_0^t f(\tau,\eta^{[p]}(\tau,x))d\tau\right] = f(t,\eta^{[p+1]}(t,x)) - f(t,\eta^{[p]}(t,x)). \tag{2.6}$$

Definimos

$$q(t,x) = \eta^{[p+1]}(t,x) - \int_0^t f(\tau,\eta^{[p]}(\tau,x))d\tau. \tag{2.7}$$

Sustituyendo en (2.6)

$$q_t = f\left(t, q(t,x) + \int_0^t f(\tau,\eta^{[p]}(\tau,x))d\tau\right) - f(t,\eta^{[p]}(t,x)).$$

Usamos la discretización en el tiempo $0 = t_0 < t_1 < ... < t_N = T$, con $t_m = m\Delta t$, para $m = 0, 1, 2...N$. El valor de la aproximación en el nodo t_m lo denotamos como $\eta^m = \eta(t_m, x)$, y $q^m = q(t_m, x)$. Aplicando el esquema de Euler implícito

$$\frac{q^{m+1} - q^m}{\Delta t} = f\left(t^{m+1}, q^{m+1} + \int_0^{t^{m+1}} [f(\tau,\eta^{m+1,[p]})]d\tau\right) - f(t^{m+1},\eta^{m+1,[p]}) \tag{2.8}$$

$$q^{m+1} - q^m = \Delta t\left[f\left(t^{m+1}, q^{m+1} + \int_0^{t^{m+1}} [f(\tau,\eta^{m+1,[p]})]d\tau\right) - f(t^{m+1},\eta^{m+1,[p]})\right]$$

despejamos y sustituimos el valor de $q(t,x)$ de la ecuación (2.7)

$$\eta^{m+1,[p+1]} - \int_0^{t^{m+1}} [f(\tau,u(\tau,x))]d\tau - \eta^{m,[p+1]} + \int_0^{t^m} [f(\tau,\eta(\tau,x))]d\tau = \Delta t f(t^{m+1},\eta^{m+1,[p+1]}) - \Delta t f(t^{m+1},u^{m+1,[p]}). \tag{2.9}$$

Agrupando términos obtenemos

$$\eta^{m+1,[p+1]} - \int_{t^m}^{t^{m+1}} [f(\tau,u(\tau,x))]d\tau - \eta^{m,[p+1]} = \Delta t f(t^{m+1},\eta^{m+1,[p+1]}) -$$

$$\Delta t f(t^{m+1}, u^{m+1,[p]}), \tag{2.10}$$

donde el super-índice p hace referencia al nivel de predicción ($p = 0$) o corrección ($p = 1, 2, ...M$), para M = número de correcciones. Por lo tanto la solución aproximada en el nivel $p+1$ es la solución corregida y la obtenemos despejando $\eta^{m+1,[p+1]}$ de (2.10)

$$\eta^{m+1,[p+1]} = \eta^{m,[p+1]} + \Delta t f(t^{m+1}, \eta^{m+1,[p+1]}) - \Delta t f(t^{m+1}, u^{m+1,[p]}) + \int_{t^m}^{t^{m+1}} [f(\tau, u(\tau, x))] d\tau \tag{2.11}$$

la integral $\int_{t^m}^{t^{m+1}} [f(\tau, u(\tau, x))] d\tau$ se calcula mediante algún método de cuadratura propuesto más adelante.

2.4.1. Nivel de predicción

Iniciamos discretizando el problema en el tiempo. Denotamos a $\eta^{[p]}(t_{m+1})$ como el valor de la aproximación $\eta^{m+1,[p]}$ obtenida por una discretización de Euler implícito de primer orden para (2.11). Los pasos en el tiempo en la predicción están dados por $m = 1, 2, ...N$, como

$$\frac{\eta^{m+1,[0]} - \eta^{m,[0]}}{\Delta t} = f(t^{m+1}, \eta^{m+1,[0]}) \quad \rightarrow \quad \eta^{m+1,[0]} = \eta^{m,[0]} + \Delta t (f(t^{m+1}, \eta^{m+1,[0]})) \tag{2.12}$$

2.4.2. Nivel de Corrección

Una vez que se ha calculado el nivel de predicción se realizaran las M correcciones para determinar una mejor aproximación de $\eta(t, x)$.

$$\eta^{m+1,[p]} = \eta^{m,[p]} + \Delta t (f(t^m, \eta^{m,[p]}) - f(t^m, \eta^{m,[p-1]})) + \int_{t^m}^{t^{m+1}} f(t, \eta^{[p-1]}(t)) dt \tag{2.13}$$

donde $\int_{t^m}^{t^{m+1}} f(t, \eta^{[p-1]}(t)) dt$, es calculada mediante

1. Regla de Trapecio (*ver apéndice* A)
2. Extrapolación de Richarson (*ver apéndice* B)

3. Aproximación por polinomio cubico (*ver apéndice C*)

2.5. Algoritmo

Algorithm RIDC

Require: Intervalo $\{a, b\}$; condición inicial α; cantidad de intervalos N; orden del método p; número de correcciones $M = p - 1$; K intervalos por grupo (Nota: Se requieren N intervalos que sean divisibles en K para obtener J grupos de K intervalos).

1: $\eta - 1 \leftarrow \alpha, \quad \Delta t \leftarrow \frac{b-a}{N}, \quad J = \frac{N}{K}$
2: **for** j = 0 **to** (J-1) **do**
3: (*Ciclo de Predicción*)
4: $\eta_{j,0}^{[0]} \leftarrow \eta_{j-1}$
5: **for** m = 0 **to** (M-1) **do**
6: $t_{j,m} \leftarrow (jM + m)\Delta t$
7: $\eta_j^{m+1,[0]} \leftarrow \eta_j^{m,[0]} + \Delta t f(t_{j,m}, \eta_j^{m,[0]})$
8: **end for**
9: (*Ciclo de Corrección*)
10: **for** p = 1 **to** M **do**
11: $\eta_j^{0,[p]} \leftarrow \eta_j^{0,[p-1]}$
12: **for** m = 0 **to** (M-1) **do**
13: $\eta_j^{m+1,[p]} \leftarrow \eta_j^{m,[p]} + \Delta t(f(t_{j,m}, \eta_j^{m,[p]} - f(t_{j,m}, \eta_j^{m,[p-1]})) + \int_{t_m}^{t_{m+1}} f(t, \eta^{[p-1]}(t))dt$
14: **end for**
15: **end for**
16: $\eta_j^{m+1,[p]} \leftarrow \eta_{j,M}^{[M]}$
17: **end for**

El algoritmo puede resolver la cuadratura de (2.11) mediante distintas técnicas como son interpoladores de Lagrange, regla del trapecio, spline, extrapolación de Richardson, entre otras las cuales se implementan en el paso **13**: del algoritmo, e inclusive pueden ser calculadas antes de los ciclos de predicción-corrección (según la cuadratura que se elija).

2.6. Esquema de paralelización

Debido a su naturaleza en nivel el método RIDC puede ser calculado de manera eficiente en ambientes multicore, multi-cpu o procesamiento gráfico (GPU). Básicamente el uso de múltiples procesos se realiza de la siguiente manera:

1. El primer proceso es encargado de calcular el valor del nivel de predicción $\eta_m^{[0]}, \quad m = 1, ..., K$

2. Un segundo proceso es encargado de calcular el primer nivel de corrección $\eta_m^{[1]}, \quad m = 1, ..., K$ una vez que existe suficiente información del ciclo de predicción. Específicamente, el segundo proceso calcula $\eta_m^{[1]}$ después de que $\eta_z^{[0]}, \eta_{z+1}^{[0]}, ..., \eta_{z+M}^{[0]}$ han sido calculados por el primer proceso, donde $z = max(m - M, 0)$.

3. Un tercer proceso calcula la segunda corrección $\eta_m^{[2]}, \quad m = 1, ..., K$, una vez que existe suficiente información de la primera corrección, y así sucesivamente las siguientes correcciones serán calculadas por procesos diferentes una vez que se tenga la suficiente información de la corrección anterior hasta terminar los p niveles de corrección deseados.

Para un uso óptimo de memoria, el p-ésimo proceso no debería estar muy lejos del proceso anterior $(p-1)$, de tal forma que se puedan descartar valores de $\eta_m^{[p]}$ que no son necesarios.

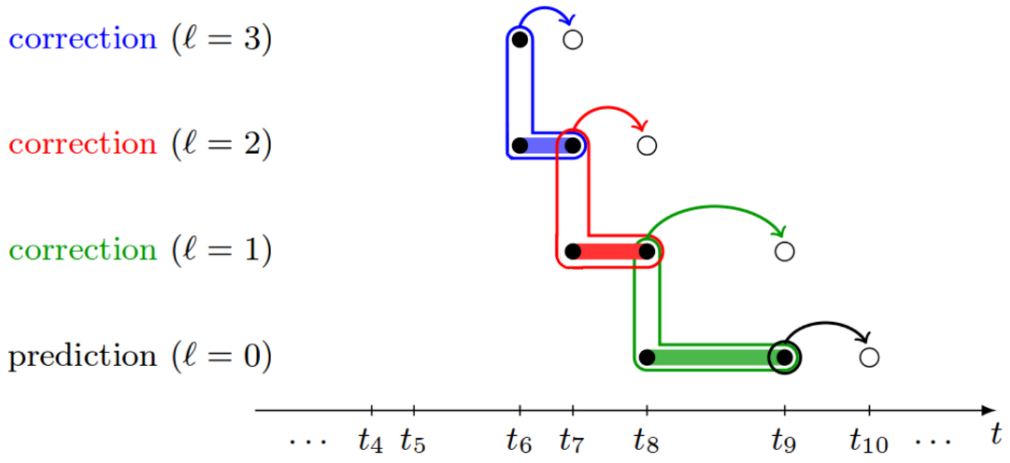


FIGURA 2.1: Se muestran los nodos necesarios (círculo negro) para realizar el cálculo de la nueva solución en el nivel de corrección correspondiente (círculo blanco)

Múltiples niveles de solución son calculados en paralelo utilizando M procesos (nodos y/o hilos) donde $M = 1, 2, 3, ...$ el cálculo en paralelo no puede iniciar de manera simultánea, cada proceso debe esperar a que en el nivel anterior se calculen suficientes $\eta^{m,[p]}$ valores antes de que todos los procesos estén trabajando al mismo tiempo.

El $j-$ *ésimo* proceso (ejecutando la corrección $j-$ *ésima*) debe iniciar hasta una vez que se han finalizado $\frac{j(j+1)}{2}$ pasos.
El calculo termina cuando el nivel más alto (nivel de mas precisión) alcanza el tiempo final $t_N = b$.

2.6.1. Análisis de tiempo de ejecución

Los procesos (cores y/o hilos) no pueden iniciar de manera simultánea: cada proceso debe esperar hasta tener suficientes valores de la solución aproximada η. En teoría una implementación de p-procesos construida usando J grupos de K intervalos utilizando el método de Euler hacia delante puede ser calculado en $N + JM^2$ pasos. Esto puede ser comparado con el método Euler ejecutado en un solo proceso el cuál toma N pasos. Por lo cual podemos considerar la proporción como $\gamma = \frac{N+JM^2}{N}$. La convergencia del método se basa en el hecho de que no solo las soluciones son calculadas con un método de orden menor, sino que también el cálculo en paralelo de la solución aproximada en cada nivel permite una eficiencia notable en el tiempo total de cómputo.

2.6.2. Convergencia

Para ecuaciones diferenciales ordinarias el método RIDC establece que

Theorem 2.1. *Sea $f(t, y)$ y $y(t)$ un problema de valor inicial suficientemente suave. Entonces, para un método RIDC constituido usando $K > p$ con nodos uniformemente distribuidos ($t_k = kh, k = 0, ..., K$), una predicción Euler y $M = p - 1$ ciclos de corrección, el error de truncamiento local es $O(h^{p+1})$.*

La comprobación puede ser verificada en [3]

Capítulo 3

Ecuaciones Diferenciales Ordinarias

3.1. Introducción

Decidimos evaluar la eficiencia del método en Ecuaciones Diferenciales ordinarias y parciales con la finalidad de conocer y corroborar ventajas y/o desventajas del método, además claro de potencializar las fortalezas del mismo. Comenzaremos hablando de ecuaciones diferenciales ordinarias. Recordemos que éstas se caracterizan por tener una sola variable independiente, pueden clasificarse de acuerdo a su orden y se utilizan para describir muchos de los fenómenos físicos de la naturaleza, además es posible resolver este tipo de ecuaciones por una buena cantidad de métodos numéricos uno de los más utilizados y más exactos, aunque no el mas sencillo, es el método de Runge-Kutta (con distintos órdenes de solución) el cual utilizaremos refiriéndonos a él como RK con la finalidad de comparar soluciones con las planteadas en el algoritmo al que hacemos referencia en esta tesis, y aun cuando el algoritmo no está específicamente planteado para la solución de este tipo de ecuaciones diferenciales, es importante llevar al lector desde los casos mas sencillos hasta los más complejos así el algoritmo puede entenderse de una forma mas simple, cuando solo planteamos una discritización, en este caso la discretización en el tiempo.

Para ello nos dimos a la tarea de elegir algunos modelos que presentaremos enseguida donde podremos evaluar el comportamiento y la eficiencia del algoritmo

dependiendo del modelo en el que se utiliza, en general el algoritmo como tal debe mejorar la solución aproximada calculando la mejor solución en los niveles de corrección, tomando como base la solución en el nivel de predicción, podremos observar además el comportamiento de los errores numéricos para cada uno de los niveles de corrección así como comparaciones en la precisión de la solución aproximada con la exacta (en caso de conocerla) y el tiempo de ejecución comparado el método de Runge Kutta de cuarto orden (RK4).

Como nuestras soluciones son a partir de métodos numéricos, es importante comentar que estamos considerando dos tipos de métricas para evaluar los errores cometidos en el cálculo de la solución aproximada, estos son: Error relativo y Error cuadrático medio.
Cada modelo se presentará con la siguiente estructura: planteamiento del problema, solución analítica (si existe), solución numérica, análisis del error en la solución numérica, análisis de tiempo de ejecución y comparación con el método RK4.

3.2. Nivel de Predicción

En el caso de las EDOs únicamente necesitamos discretizar nuestro problema respecto al tiempo (t) y utilizar el esquema propuesto en 2.12, para cada ejemplo necesitaremos especificar quien es nuestra función $f(t, u(t))$ así como las condiciones iniciales del problema. Es importante mencionar que en todos los modelos utilizamos un esquema de solución implícito para encontrar la solución numérica. Los ejemplos propuestos en el presente trabajo siguen la forma :

$$f(t, u) \equiv A(t^{n+1})u(t^{n+1}) + b(t^{n+1}) \tag{3.1}$$

por lo que el esquema de predicción queda como sigue:

$$[I - \Delta t\ A(t^{n+1})]u^{n+1,[0]} = u^{n,[0]} + b(t^{n+1})$$

$$u^{n+1,[0]} = [I - \Delta t\ A(t^{n+1})]^{-1}[u^{n,[0]} + b(t^{n+1})] \tag{3.2}$$

donde $u^{n+1,[0]}$ es el valor de la solución aproximada en el predictor ($[0]$), en el tiempo $n + 1$.

3.3. Nivel de Corrección

Cada nivel de corrección lo realizamos resolviendo la ecuación propuesta en (2.13), tomando como función lo propuesto en (3.1) obtenemos

$$[I - \Delta t\ A(t^{n+1})]u^{n+1,[p+1]} = \Delta t\ b(t^{n+1}) + u^{n,[p+1]} - \Delta t[A(t^{n+1})u^{n+1,[p]} + b(t^{n+1})] + \int_{t_n}^{t_{n+1}} [A(t)u^{[p]} + b(t)]dt$$

$$u^{n+1,[p+1]} = [I - \Delta t\ A(t^{n+1})]^{-1}[\Delta t\ b(t^{n+1}) + u^{n,[p+1]} - \Delta t[A(t^{n+1})u^{n+1,[p]} + b(t^{n+1})] + \int_{t_n}^{t_{n+1}} A(t)u^{[p]} + b(t)dt] \tag{3.3}$$

3.4. Modelos y Resultados

3.4.1. Modelo 1. EDO de primer orden

Para el problema de valor inicial

$$\begin{cases} u'(t) = & \lambda u \\ u(0) = & u_0 \end{cases} \tag{3.4}$$

De (3.1) podemos notar que para el modelo actual $f(t,u) \equiv \lambda u(t)$ si vemos nuestro problema desde un enfoque matricial tenemos que $f(t,u) \equiv A(t)u(t) + b(t)$, donde $A(t) = \lambda$ y $b(t) = 0$ entonces la función de predicción ($p = 0$) quedaría como (*véase* (3.2)) mientras que los p niveles de corrección se calculan con (véase (3.3)).

Las condiciones iniciales están dadas por $\alpha = u_0$. Los parámetros que se utilizaron para la evaluación de la ecuación son los siguientes:

1. $t \in [0, 5]$, $\lambda = 5$, $u_0 = 1$
2. $N = K \in \{250,\ 500,\ 1000,\ 5000\}$
3. $\Delta t \in \{0.02,\ 0.01,\ 0.005,\ 0.001\}$,

la solución analítica del problema se observa en la siguiente figura

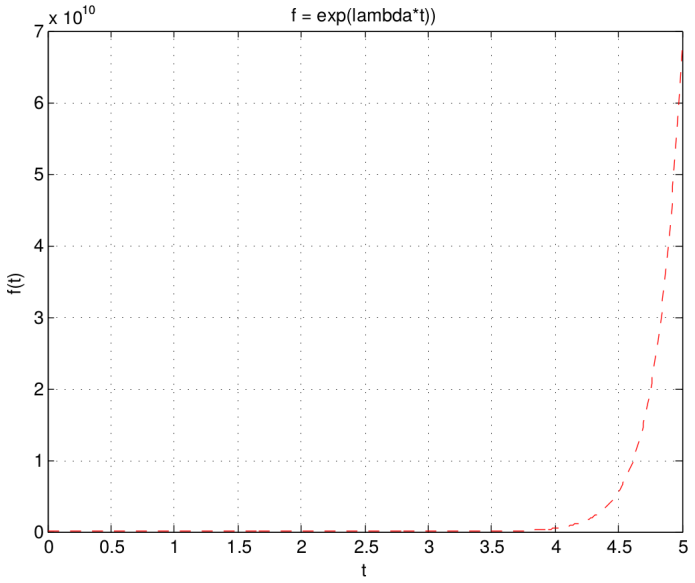


FIGURA 3.1: Solución analítica para el modelo 3.4

Sabemos que de acuerdo a lo establecido por el método RIDC las correcciones mejoran de manera significativa la solución obtenida en el nivel de predicción, reduciendo así el error relativo (véase *cuadro 3.1*).

	Δt	N	ε_0	ε_1	ε_2	ε_3
a)	0.02	250	2.70954	2.33159	1.36118	0.56804
b)	0.01	500	0.88351	0.32991	0.09225	0.01266
c)	0.005	1000	0.36521	0.06344	0.00931	0.00049
d)	0.001	5000	0.06338	0.00208	0.00010	0.00005

CUADRO 3.1: Error relativo de la solución aproximada en el nivel de predicción (ε_0) y tres correcciones (ε_1, ε_2, ε_3) evaluado para distintos tamaños de paso (Δt).

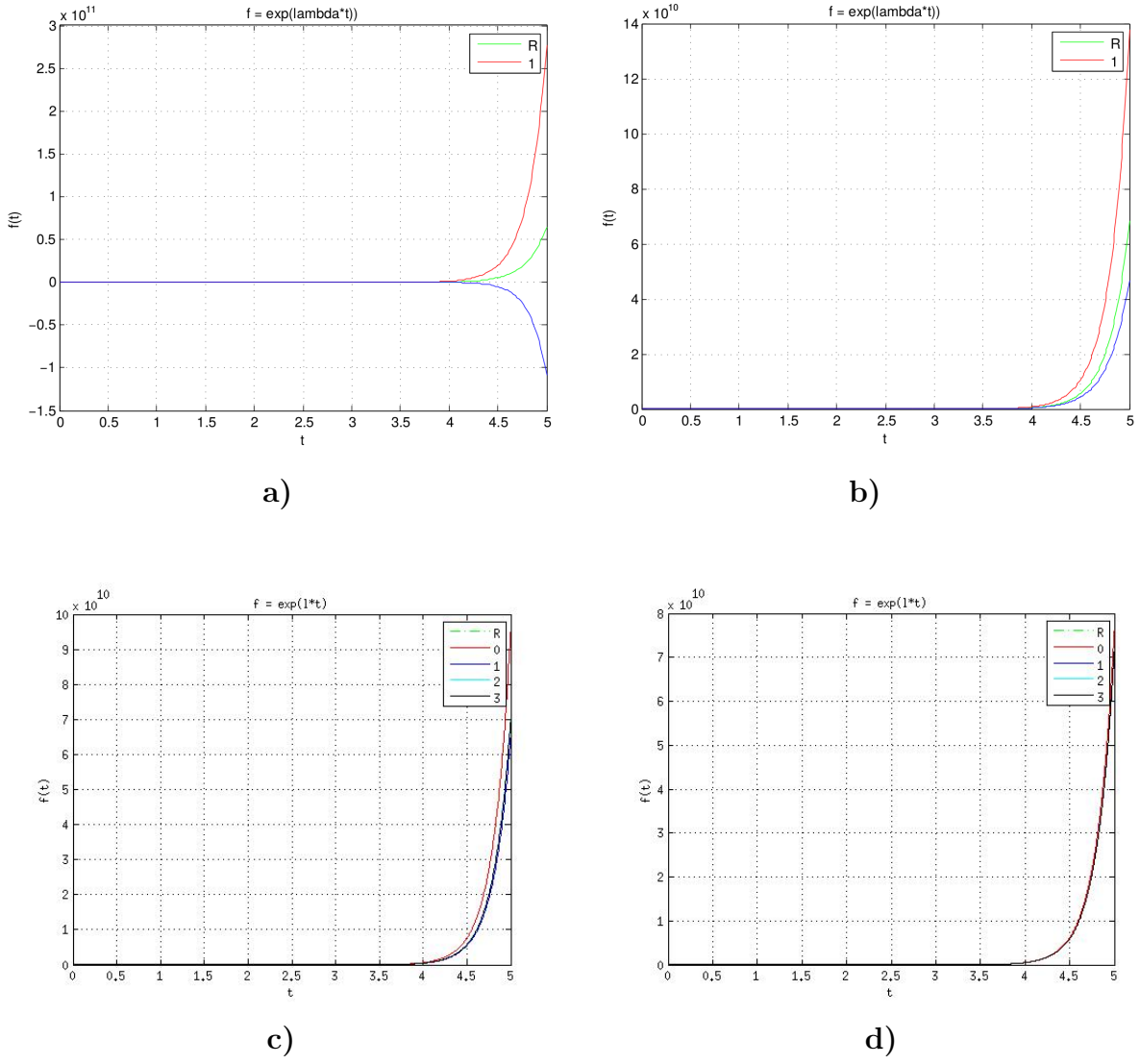


FIGURA 3.2: En la figura se muestra el resultado de la solución aproximada en cada nivel de predicción-corrección para Δt a) 0.02, b) 0.01, c) 0.005, d) 0.001. Se muestra la solución exacta en color verde y las correcciones aparecen en distintos colores cada una indicando un número el cual se refiere a la predicción (0), o alguna corrección(1,2,3,...).

Para la solución de esta ecuación se ha observado que la convergencia depende directamente del tamaño de paso, debido a que para $\Delta t > 0.01$ las correcciones no realizan una buena aproximación a la solución (véase *figura 3.2a*), este comportamiento se observar para $\Delta t = 0.02$, mientras que para $\Delta t \leq 0.01$ los resultados de la aproximación son estables y convergen a la solución (véase *figura 3.2b,3.2c,3.2d*).

En el cuadro (3.2) podemos notar ventajas considerables en los valores de los errores relativos entre el método RIDC y el método RK4, aunque en tiempo de

Δt	T_{ridc}	T_{rk4}	ε_{ridc}	ε_{rk4}
0.02	0.007	0.004	0.12763	0.22137
0.01	0.013	0.005	0.05567	0.10517
0.005	0.015	0.005	0.02613	0.05127
0.001	0.017	0.006	0.00496	0.00516

CUADRO 3.2: Tiempo de ejecución comparado con el método RK4

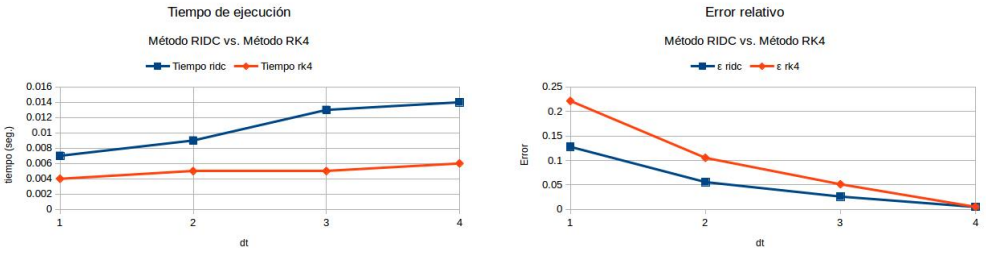


FIGURA 3.3: En la figura se muestra graficamente lo que ocurre con el tiempo de ejecución (*izquierda*) y el valor de error relativo (*derecha*) en ambos métodos.

ejecución el método RK4 es más rápido manteniendo tiempo promedio de 0.005 *segundos* contra 0.013 *segundos* del método RIDC.

Es importante mencionar que los tiempos de ejecución fueron tomados de un promedio de treinta ejecuciones de cada uno de los algoritmos para cada Δt y que a diferencia de los resultados del cuadro (3.1) se utilizaron 8 procesos de ejecución para las correcciones del método, se observo que después de la octava corrección la solución no tenia una mejora significativa en el error relativo.

3.4.2. Modelo 2. EDO de segundo orden

Se desea resolver la siguiente EDO de segundo orden

$$\begin{cases} \ddot{u} + \omega u = & sin(\omega_0 t) \\ u(0) = & u_0 \\ u'(0) = & 0 \end{cases} \tag{3.5}$$

Antes de resolver la ecuación tenemos que realizar algunos cambios algebraicos. Comenzaremos realizando un cambio de variable

$$\begin{aligned} u_1 &= u \\ u_2 &= u' = u_1' \\ u_2' + w^2 u_1 &= sin(w_0 t) \end{aligned} \tag{3.6}$$

con lo cual tenemos

$$\begin{aligned} u_1' &= u_2 \\ u_2' &= sin(\omega_0 t) - \omega^2 u_1 \end{aligned}$$

con condiciones iniciales

$$\begin{aligned} u_1(0) &= u_0 \\ u_2(0) &= 0 \end{aligned}$$

si tomamos

$$v = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

entonces tenemos que $v' = Av + b$ y se define como sigue

$$v' = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ sin(w_0 t) \end{bmatrix} \tag{3.7}$$

por lo tanto el modelo que queremos resolver nos queda en función de vectores y matrices y debido al cambio de variable, podemos reducir hasta llegar a lo siguiente:

$$\begin{cases} v' = & Av + b \\ v(0) = & [u_0, 0] \end{cases} \tag{3.8}$$

utilizando (3.2) tenemos que el valor en el nivel de predicción se calcula como sigue

$$[I - \Delta t\ A(t^{n+1})]v^{n+1,[0]} = v^{n,[0]} + \Delta t\ b(t^{n+1}) \tag{3.9}$$

mientras que los (p) niveles de corrección (2.11) se calculan como siguen:

$$[I - \Delta t\ A(t^{n+1})]v^{n+1,[p+1]} = \Delta t\ b(t^{n+1}) + v^{n,[p+1]} - \Delta t[A(t^{n+1})v^{n+1,[p]} + b(t^{n+1})]$$

$$+ \int_{t_n}^{t_{n+1}} A(t)u^{[p]} + b(t)dt \tag{3.10}$$

La ecuación se resolvió tanto para la forma homogénea como para la no homogénea, los resultados se describen a continuación.

3.4.2.1. Ecuación homogénea

Conocemos la solución analítica al sistema cuando la ecuación es homogénea, el cual es:

$$\begin{aligned} u(t) &= u_0 cos(\omega t) \\ u'(t) &= -\omega u_0 sin(\omega t) \\ u''(t) &= -u_0 \omega^2 cos(\omega t) \end{aligned} \tag{3.11}$$

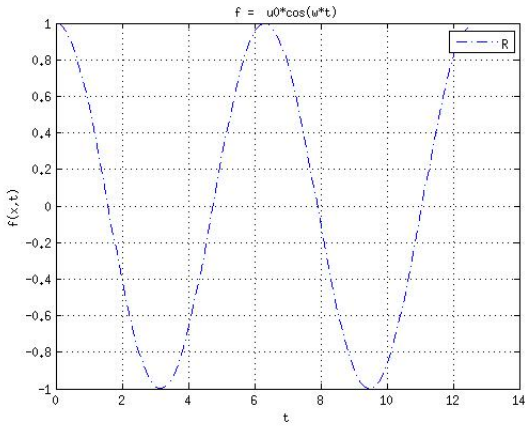


FIGURA 3.4: Gráfica de la solución analítica para la ecuación homogénea

Si resolvemos la ecuación para la forma homogénea, obtenemos resultados tanto de predicción como de corrección para aproximar la función $u(x,t)$, tomando como **predicción** el indicativo 0 y las **correcciones** como $1, 2, 3...(M-1)$, donde M es el número de procesos. Además se tienen los siguientes parámetros:

1. Intervalo de evaluación $t \in [0, 4\pi]$
2. $\Delta t \in \{0.5,\ 0.1,\ 0.01\}$
3. $N = K \in \{$ 25,126,251,1257 $\}$
4. M : 4 (predictor, 3 correcciones)
5. Errores $\{\varepsilon_0,\ \varepsilon_1,\ \varepsilon_2,\ \varepsilon_3\}$

	Δt	N	ε_0	ε_1	ε_2	ε_3
a)	0.5	25	0.64004	0.38540	0.21028	0.11392
b)	0.1	126	0.25451	0.04875	0.00748	0.00317
c)	0.05	251	0.14046	0.01404	0.00122	0.00081
d)	0.01	1257	0.03075	0.00064	0.00003	0.00003

CUADRO 3.3: Error relativo de la solución aproximada en el nivel de predicción (ε_0) y sus respectivas correcciones ($\varepsilon_1, \varepsilon_2, \varepsilon_3$) comparados con el resultado exacto de la función.

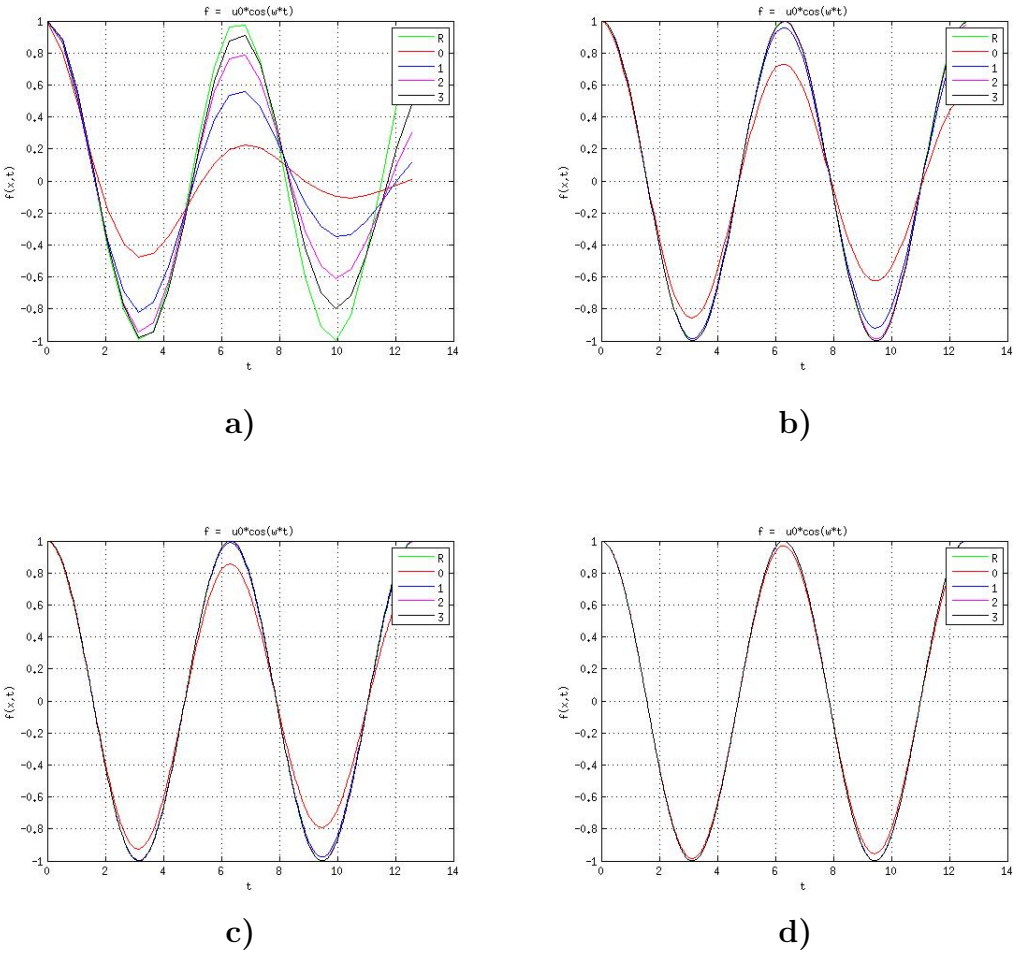


FIGURA 3.5: En la figura se muestra el resultado de la solución aproximada en cada nivel de corrección, para tamaños de paso(Δt) a)0.5, b) 0.1, c)0.05, d)0.01, se puede observar que al reducir Δt la predicción y las correcciones son cada vez más precisas. Aunque también se observa que a tamaños de paso mayores las correcciones son más significativas mejorando así el valor de la predicción.

En cuanto a los tiempos de ejecución del algoritmo, recordemos que se debe mantener una estabilidad en el tiempo de ejecución al momento de aumentar el número de procesos utilizados para realizar las correcciones, a continuación mostraremos

los resultados de tiempos de ejecución en el intervalo $[0, 50]$ con 500 y 5000 nodos en la driscetización en el tiempo para los tamaños de paso $\Delta t = \{0.1, 0.01\}$ elegidos por no ser muy pequeños y ofrecer buenos resultados en la solución numérica de la ecuación evaluada.

	Δt	$M = 2$	$M = 4$	$M = 6$	$M = 8$
a)	0.1	0.0048	0.0072	0.0054	0.0054
b)	0.01	0.0133	0.0172	0.0162	0.0122

CUADRO 3.4: Tiempo de ejecución para M cantidad de procesos en dos tamaños de paso distintos.

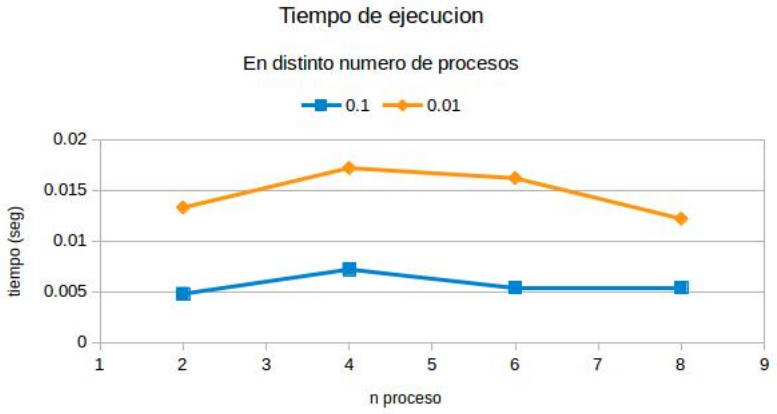


FIGURA 3.6: En la figura se muestra el tiempo de ejecución del método RIDC para 2,4,6,8 procesos, respectivamente.

Como mencionamos con anterioridad, uno de los objetivos del método es mantener el tiempo de ejecución aun cuando se vayan aumentando la cantidad de procesos y como se puede observar en la *figura* (3.6) el tiempo de ejecución aumenta ligeramente cuando se está ejecutando por 4 procesos pero se vuelve a estabilizar para 6 y ocho, y así sucesivamente para mayor cantidad de procesos.

Como ocurre en el modelo 1, presentado con anterioridad los valores del error para los tamaños de paso propuestos comparados con RK4, son mejores para la solución que estamos aproximando, aunque este método sigue estando por encima del RIDC en el análisis de tiempo de ejecución.

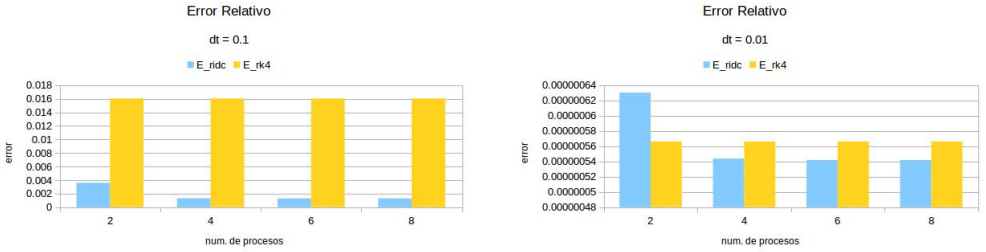


FIGURA 3.7: Gráficas de los errores relativos comparados con el método RK4 para $\Delta t = 0.1$ (*izquierda*) y $\Delta t = 0.01$ (*derecha*).

3.4.2.2. Ecuación no homogénea

La solución analítica para el caso no homogéneo se muestra a continuación

$$\begin{aligned} u(t) &= \frac{-sin(t\omega_0)}{\omega_0^2-\omega^2} + \frac{\omega_0 sin(t\omega)}{w(\omega_0^2-\omega^2)} + u_0\ cos(t\omega) \\ u'(t) &= \frac{-\omega_0 cos(t\omega_0)}{\omega_0^2-\omega^2} + \frac{\omega_0\omega cos(t\omega)}{w(\omega_0^2-\omega^2)} - u_0\omega\ sin(t\omega) \\ u''(t) &= \frac{\omega_0^2 sin(t\omega_0)}{\omega_0^2-\omega^2} + \frac{\omega^2\omega_0 sin(t\omega)}{w(\omega_0^2-\omega^2)} + \omega^2\ u_0\ cos(t\omega) \end{aligned} \tag{3.12}$$

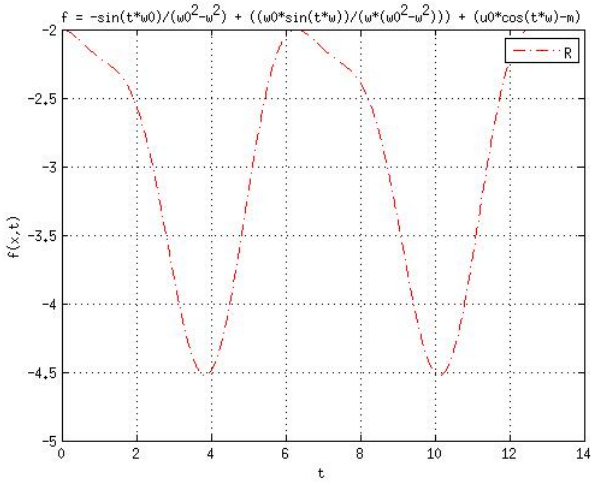


FIGURA 3.8: Gráfica de la solución analítica para la ecuación homogénea

Considerando los mismos parámetros utilizados en la solución de la ecuación homogénea tenemos los siguientes resultados para la ecuación no homogénea

	Δt	N	ε_0	ε_1	ε_2	ε_3
a)	0.5	25	0.16147	0.09940	0.05814	0.03721
b)	0.1	126	0.05547	0.01248	0.00879	0.00865
c)	0.05	251	0.02983	0.00492	0.00450	0.00447
d)	0.01	1257	0.00638	0.00091	0.00092	0.00092

CUADRO 3.5: Resultados del error relativo de la solución aproximada en el nivel de predicción (ε_0) y sus respectivas correcciones ($\varepsilon_1, \varepsilon_2, \varepsilon_3$) comparados con el resultados de la función analítica.

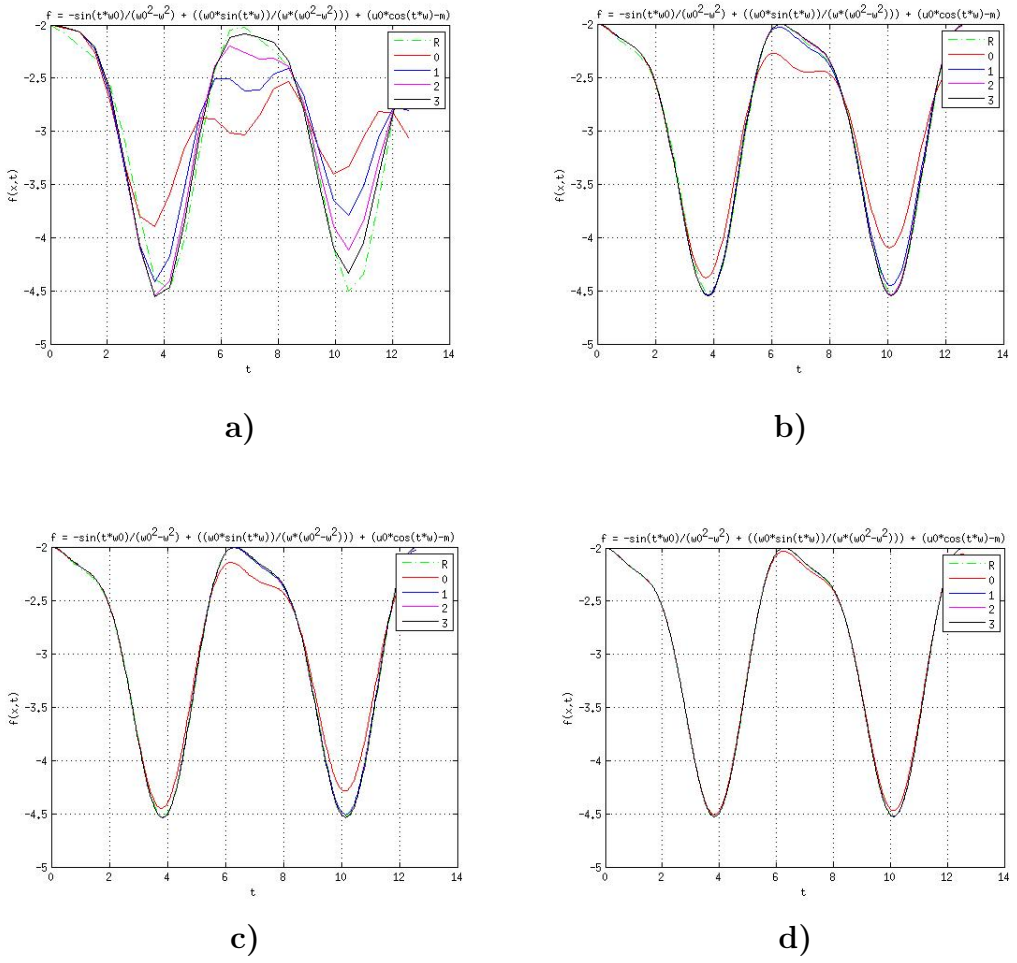


FIGURA 3.9: En la figura se muestra el resultado de la solución aproximada en cada nivel de corrección, para tamaños de paso(Δt) a) 0,5, b) 0,1, c) 0,05, d) 0,01.

se puede observar que al reducir Δt la predicción y las correcciones son cada vez más precisas. Aunque también se observa que a tamaños de paso mayores las correcciones realizar una buena mejora de la predicción (*figura a*), lo que ocurre de manera más natural cuando el tamaño de paso es pequeño ya que si la predicción es muy buena los correctores no tienen mucho que corregir (*figura d*).

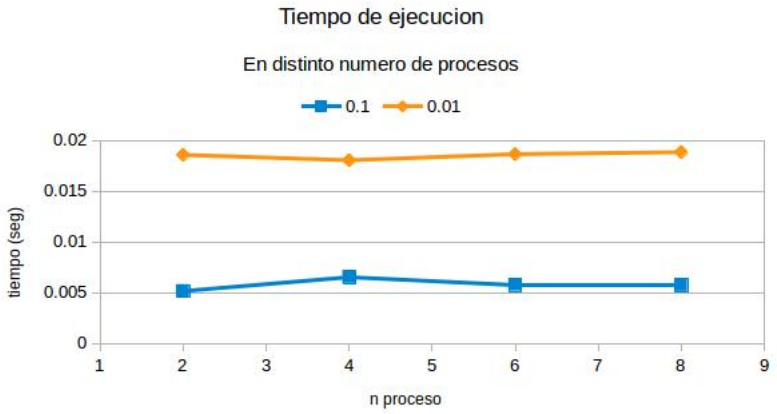


FIGURA 3.10: En la figura se muestra el tiempo de ejecución del método RIDC para 2,4,6,8 procesos, respectivamente.

El tiempo de ejecución cuando calculamos la solución a este problema se mantiene más estable que el de la solución homogénea, aunque aumenta ligeramente la cantidad de tiempo invertido en esta solución, en cuanto a los resultados de los errores relativos el método RIDC sigue obteniendo un menor error que el presentado por RK4, para esta solución los resultados son del orden de los que se presentan en la *figura* (3.14).

En el caso de esta solución no homogénea el método RK4 sigue presentando tiempos de ejecución más cortos, incluso con intervalos de tiempo más prolongados pero la mejor precisión ocurre con tamaños de paso pequeños, lo cual no necesariamente ocurre con el método RIDC, el cual para tamaños de paso considerables mantiene una precisión competitiva.

3.4.3. Modelo 3. Rígido

Un sistema de ODE se denomina rígido si su solución numérica por algunos métodos requiere una disminución significativa de la longitud de paso para evitar inestabilidad.
Se desea resolver el siguiente sistema de ecuaciones

$$\begin{cases} u' = \quad 998u + 1998v \\ v' = -999u - 1999v \end{cases} \tag{3.13}$$

Consideramos importante evaluar este tipo de sistemas con el algoritmo propuesto con la finalidad de conocer su efectividad en sistemas con características de rigidez. con condiciones de frontera $u(0) = 1$ y $v(0) = 0$. Si consideramos a u y v como un vector de valores $w = [u, v]'$ nuestro sistema quedará expresado como $w' = Aw + b$ y se define como sigue

$$w' = \begin{bmatrix} 998 & 1998 \\ -999 & -1999 \end{bmatrix} w + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.14}$$

considerando que la solución analítica es

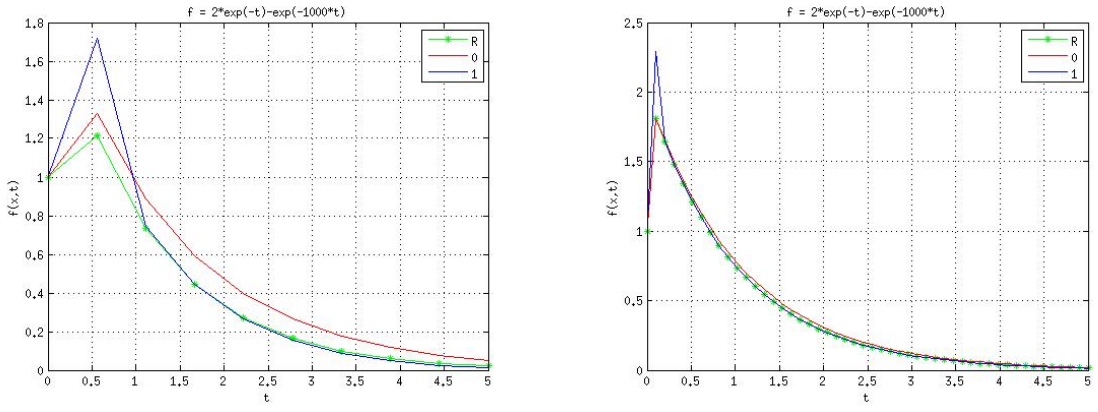
$$u = 2e^{-x} - e^{-1000x}$$

$$v = -e^{-x} + e^{-1000x}$$

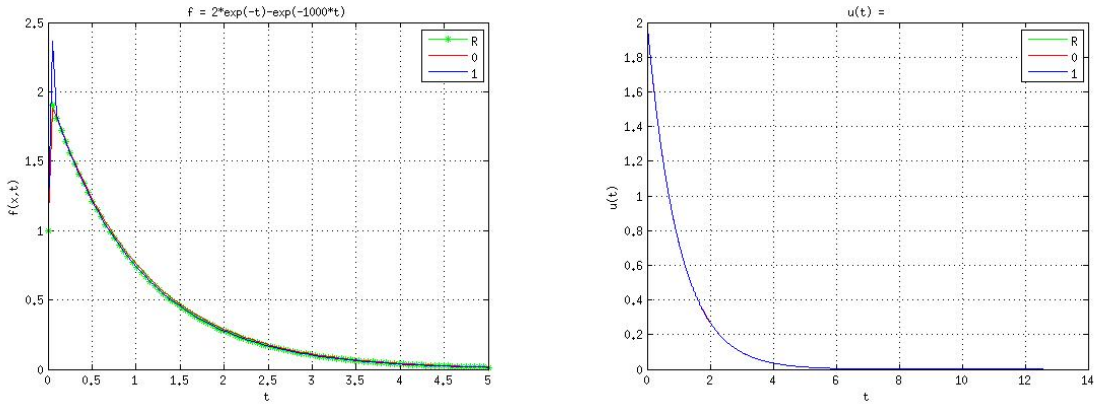
Evaluando para distintos tamaños de paso obtenemos las aproximaciones a la solución de la ecuación (*véase figura* (3.11)), así como sus respectivos errores relativos con respecto a la solución analítica (*véase cuadro* (3.6)).

	Δt	N	ε_0	ε_1
a)	0.5	10	0.11557	0.18779
b)	0.1	50	0.02441	0.04680
c)	0.01	500	0.00348	0.00398
d)	0.001	500	0.00063	0.00009

CUADRO 3.6: Resultados del error relativo de la solución aproximada en el nivel de predicción (ε_0) y una única corrección (ε_1) comparados con el resultados de la función analítica, para distintos tamaños de paso (Δt).



a) **b)**



c) **d)**

FIGURA 3.11: En la figura se muestra el resultado de la solución aproximada en cada nivel de predicción-corrección,respectivamente, para tamaños de paso Δt para a) 0,5, b) 0,1, c) 0,05, d) 0,01.

Si observamos con detenimiento los resultados obtenidos en este modelo en particular para los primeros tamaños de paso ($\Delta t \in 0.5{,}0.1$), los valores obtenidos en comparación con lo observado hasta ahora del algoritmo no corrigen la predicción, pero conforme el tamaño de paso disminuye la solución encuentra un momento de estabilidad en el cual la predicción realiza un buen cálculo y las correcciones efectivamente corrigen el valor de la solución de tal forma que se puedan obtener valores cercanos a la solución.

Recordemos que una de las bondades que queremos aprovechar del método RIDC es el hecho de que no necesariamente tener que elegir un tamaño pequeño, por el contrario queremos tener tamaños de paso considerablemente 'grandes' con los cuales la solución pueda ser corregida de tal forma que los resultados fueran muy parecidos a los obtenidos con tamaños de paso pequeños, ahorrando así tiempo de computo.

Los tiempos de ejecución son presentados en la siguiente figura calculados para un total de 500 nodos en la discretización espacial.

Aún cuando parece que el tiempo de procesamiento aumenta al aumentar el numero de procesos, en realidad se estabiliza al llegar a 6 y 8 procesos respectivamente, como se observa en la *figura* (3.12), el comportamiento es muy similar a lo ocurrido en el modelo 2.

FIGURA 3.12: En la figura se muestra el tiempo de ejecución del método RIDC para 2,4,6,8 procesos, respectivamente.

3.4.4. Modelo 4. Sistema Autónomo

Se desea encontrar una solución para el siguiente sistema.

$$\begin{cases} y_1'(x) = -2(y_2(x) - 2{,}4) \\ y_2'(x) = 3(y_1(x) - 1{,}2) \end{cases} \tag{3.15}$$

si lo representamos de la forma

$$Y' = F(Y(x)) \qquad F(Y) = \begin{bmatrix} 0 & -2 \\ 3 & 0 \end{bmatrix} Y + \begin{bmatrix} 4{,}8 \\ -3{,}6 \end{bmatrix} \tag{3.16}$$

con condición inicial

$$Y(0) = \begin{bmatrix} 1{,}5 \\ 1{,}5 \end{bmatrix}, \tag{3.17}$$

entonces la solución analítica es

$$Y(x) = \begin{bmatrix} \sqrt{2} r cos(\sqrt{6}x + \delta) \\ \sqrt{3} r sin(\sqrt{6}x + \delta) \end{bmatrix} + \begin{bmatrix} 1{,}2 \\ 2{,}4 \end{bmatrix} \tag{3.18}$$

donde $\delta = arctan(-\sqrt{6})$ y $r = \frac{0{,}3}{\sqrt{2} * cos\delta}$

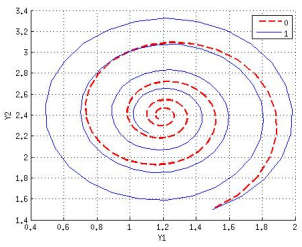
A continuación se muestran los resultados para la solución aproximada del sistema (3.15), donde se observan los errores calculados de acuerdo al error cuadrático medio (RMS *ver apéndice* D) para distintos tamaños de paso (Δt) en el intervalo $[0, 10]$.

	Δt	N	ε_0	ε_1	ε_2	ε_3
a)	0.1	100	6.48562e-01	4.37308e-01	2.63472e-01	1.43871e-01
b)	0.01	1000	1.08065e-01	1.58056e-02	1.35846e-03	5.78042e-04
c)	0.001	10000	1.51602e-02	1.75239e-04	6.08430e-06	6.28854e-06

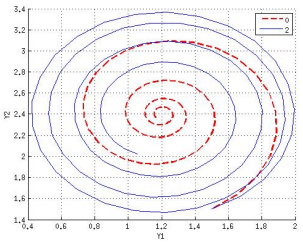
CUADRO 3.7: Resultados del error RMS de la solución aproximada en el nivel de predicción (ε_0) y en los niveles de corrección (ε_1, ε_2, ε_3) comparados con el resultados de la función analítica, para distintos tamaños de paso (Δt).

En el *cuadro* (3.7) ocurre una situación muy interesante con los valores de la fila b donde los valores en los errores disminuyen en un orden de magnitud respecto a su antecesor, con lo anterior es notable que las correcciones están obteniendo un aproximación muy buena de la solución esperada, así mismo en las filas a y c los errores disminuyen considerablemente conforme se van realizando las respectivas correcciones.

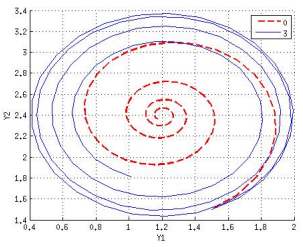
Una forma más visual de darnos cuenta como interpretar los resultados anteriores es mediante las gráficas de los resultados del modelo, las cuales se presentan a continuación.



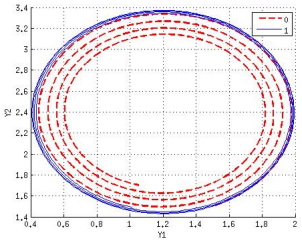
a1)



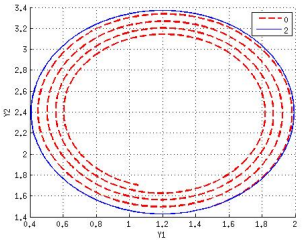
a2)



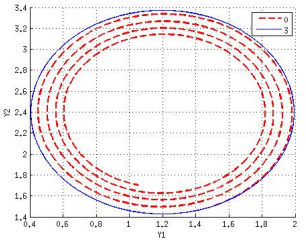
a3)



b1)

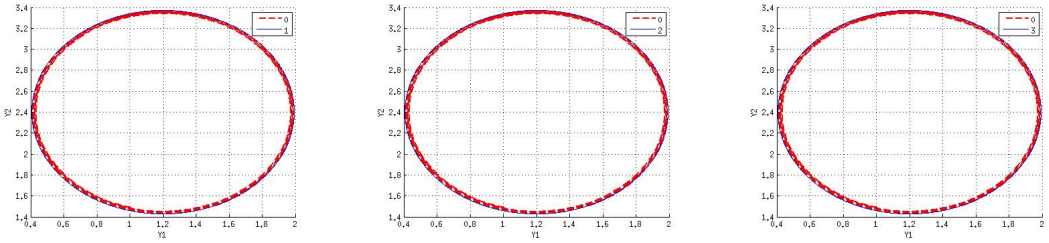


b2)



b3)

En este modelo en particular se observan muy claramente las bondades del método, tomando en cuenta la *figura* (3.13) observemos que para tamaños de paso grandes



c1) **c2)** **c3)**

FIGURA 3.13: En la figura se muestra el resultado de la solución aproximada vemos los valores de la corrección representados por la linea punteada mientras la linea continua representa alguna corrección (*de izquierda a derecha*) 1,2,3, para Δt a) *0.1*, b) *0.01*, c) *0.001*.

(**a**) la predicción tiene un valor muy diferente a la solución esperada, pero conforme se van realizando las corrección es notable la mejora y una aproximación a la solución exacta mucho más certera, en el caso de la figura solo se muestran las 3 primeras correcciones pero al aumentar el número de correcciones incluso para una mala predicción se obtienen resultados con errores pequeños.

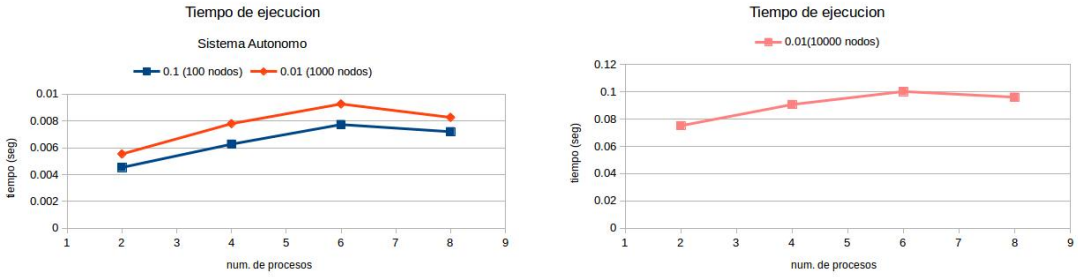


FIGURA 3.14: Tiempo de ejecución para $\Delta t \in \{\ 0.1,\ 0.01\}$ en un intervalo de $[0, 10]$ (*izquierda*) y $\Delta t = 0.01$ en un intervalo de [0,100] (*derecha*).

Los tiempos de ejecución para la solución de este modelo aumentan ligeramente conforme se van aumentando el numero se procesos aunque en el proceso 8 se nota una estabilización en el tiempo de ejecución ya que este disminuye.

Mientras los tiempos de ejecución son considerablemente buenos con respecto al RK4, éste último sigue siendo más rápido que el RIDC en cuanto a tiempo.

Lo que sí debemos notar es que RIDC tiene resultados más precisos, eso lo podemos observar en la *figura* (3.15) en la cual se observa que para tamaños de paso grandes (0.1) ocurre algo curioso mostrando más error cuando se tiene solo dos procesos, esto debido a que solo está corrigiendo la predicción una sola vez (un proceso se

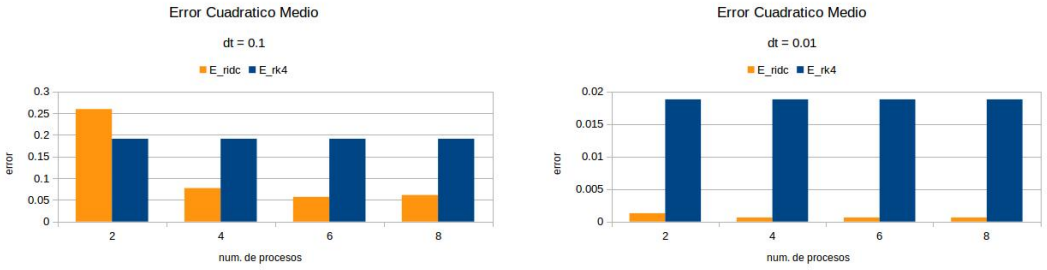


FIGURA 3.15: Tiempo de ejecución para $\Delta t \in \{\ 0.1,\ 0.01\}$ en un intervalo de $[0, 10]$ (*izquierda*) y $\Delta t = 0.01$ en un intervalo de [0,100] (*derecha*).

encarga de la predicción y el otro de la corrección), pero para más de tres procesos disminuye significativamente el error.

Mientras que el error aumenta y luego disminuye logrando ser mejor la aproximación calculada con RIDC que con RK4 para un $\Delta = 0.1$ para un $\Delta = 0.01$, la precisión prevalece para cualquier cantidad de procesos con respecto al valor calculado con RK4.

Como se mencionó al principio de este capítulo el método RIDC no está pensado propiamente para ecuaciones diferenciales ordinarias, pero como ya vimos en los modelos planteados anteriormente el algoritmo funciona con mejoras importantes y resultados muy competentes, existen casos muy particulares como es el caso del ejemplo rígido, planteado aquí, donde al algoritmo le cuesta más trabajo llegar a una solución aproximada válida sobre todo con tamaños de paso grandes. Esto debido en gran medida a la complejidad propia del problema. Sin embargo también pudimos observar resultados muy positivos en el modelo cuatro, donde los resultados nos sorprendieron con errores pequeños que disminuían en gran medida conforme se aumenta el número de correcciones.

Ya en la comparación con el método RK4, que se caracteriza por ser de los mejores en la solución de EDO, observamos que sigue siendo mas eficiente este ultimo en cuanto a tiempo de ejecución pero también vemos que no solo es cuestión de tiempo, sino también de precisión, por lo cual logramos mejorar en su mayoría los resultados proporcionados por RK4.

Capítulo 4

Ecuaciones Diferenciales Parciales

4.1. Introducción

Las ecuaciones diferenciales parciales o EDP se caracterizan por tener dos o más variables independientes y en general se clasifican en elípticas, parabólicas e hiperbólicas. Para encontrar la solución a una EDP, tenemos que encontrar una función que a lo largo de este capítulo denominaremos $u(x,t)$ que satisfaga la EDP y que además cumpla con las condiciones de frontera establecidas. La mayoría de la EDP no tienen una solución analítica por lo que se utilizan múltiples esquemas numéricos que encuentren soluciones aproximadas.

Recordemos que este tipo de ecuaciones tienen que ser discretizadas tanto en tiempo (t) como en espacio (x), lo cual ocasiona que presenten tiempos de computo más largos para la realización de los cálculos que llevan a la solución de la ecuación. El algoritmo presentado en esta tesis encuentra soluciones a EDP de manera eficiente por lo tanto se plantean expectativas muy buenas para el planteamiento de este capítulo, a lo largo del mismo se expondrá el esquema general para resolver este tipo de ecuaciones además por supuesto de la presentación de modelos de EDP cuya solución aproximada fue calculada con el método RIDC, con la misma secuencia que utilizamos en el capítulo anterior: Planteamiento del problema, solución analítica (si existe), solución numérica, análisis de error, análisis del tiempo de ejecución.

4.2. Esquema de Predicción y Corrección

Si tomamos como base una EDP de segundo orden y la plateamos de manera general como sigue:

$$\begin{array}{rcll} \frac{\partial u}{\partial t} & = & a(x)\ \frac{\partial^2 u}{\partial x^2} + b(x)\ \frac{\partial u}{\partial x} + c(x)\ u + d(x) & (x,t) \in (a,b) \times [0,T] \\ u(t,a) & = & h_a(t) & t \in [0,T] \\ u(t,b) & = & h_b(t) & t \in [0,T] \\ u(0,x) & = & g(x) & x \in [a,b]. \end{array} \tag{4.1}$$

Aproximamos la primera y segunda derivada en el nodo x_i con el método de diferencias finitas (diferencias centrales) obtenemos:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1} - 2\ u_i + u_{i+1}}{(\Delta x)^2} \tag{4.2}$$

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x} \tag{4.3}$$

Sustituyendo (4.2) y (4.3) en (4.1) obtenemos la siguiente ecuación

$$a(x)\ \left[\frac{u_{i-1} - 2\ u_i + u_{i+1}}{(\Delta x)^2}\right] + b(x)\ \left[\frac{u_{i+1} - u_{i-1}}{2\Delta x}\right] + c(x)\ u_i + d(x) = \frac{\partial u}{\partial t} \tag{4.4}$$

Multiplicando y agrupando términos semejantes obtenemos

$$\left[\frac{a(x)}{(\Delta x)^2} - \frac{b(x)}{2\Delta x}\right] u_{i-1} + \left[c(x) - \frac{2a(x)}{(\Delta x)^2}\right] u_i + \left[\frac{a(x)}{(\Delta x)^2} + \frac{b(x)}{2\Delta x}\right] u_{i+1} + d(x) = \frac{\partial u}{\partial t} \tag{4.5}$$

de tal forma que podemos representarlo como un sistema de ecuaciones

$$\frac{\partial u}{\partial t} = Au + d(x) \tag{4.6}$$

donde A es una matriz tridiagonal, cuyas diagonales contienen los valores de α, β y γ

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ \alpha & \beta & \gamma & 0 & 0 & 0 & \dots \\ 0 & \alpha & \beta & \gamma & 0 & 0 & \dots \\ 0 & 0 & \alpha & \beta & \gamma & 0 & \dots \\ 0 & 0 & 0 & \alpha & \beta & \gamma & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

donde

$$\alpha = \frac{a(x)}{(\Delta x)^2} - \frac{b(x)}{2\Delta x} \qquad \beta = c(x) - \frac{2a(x)}{(\Delta x)^2} \quad y \quad \gamma = \frac{a(x)}{(\Delta x)^2} + \frac{b(x)}{2\Delta x} \tag{4.8}$$

en este caso A tiene condiciones de frontera Dirichlet por ese motivo la primera y la ultima fila tienen un 1 al inicio y final respectivamente.

El número de renglones de A está determinado por el tamaño de la discretización que se haga en el espacio (Δ x) y para encontrar los valores de u se debe resolver el sistema de ecuaciones de (4.6) para cada instante de tiempo.

4.2.1. Nivel de predicción

Identificando el valor de $f(t, u)$ propuesto en [1] y cuya explicación obtuvimos en el *capítulo 2* como

$$f(t, u) = Au + b \tag{4.9}$$

donde $b = d(x)$ de acuerdo a (4.6) podemos obtener la ecuación de predicción utilizando un esquema de Euler implícito para $\frac{\partial u}{\partial t}$ de la siguiente forma:

$$\frac{u_i^{m+1,[0]} - u_i^{m,[0]}}{\Delta t} = f(t^{m+1}, u_i^{m+1,[0]}) \tag{4.10}$$

despejando y sustituyendo el valor de $f(t^{m+1}, u_i^{m+1,[0]})$ obtenemos

$$u_i^{m+1,[0]} - u_i^{m,[0]} = \Delta t[A(t^{m+1})u_i^{m+1,[0]} + b(t^{m+1})] \tag{4.11}$$

simplificando y agrupando términos semejantes tenemos

$$[I - \Delta t\ A(t^{m+1})]u_i^{m+1,[0]} = u_i^{m,[0]} + \Delta t\ b(t^{m+1}) \tag{4.12}$$

de tal forma que el valor de predicción lo encontramos de la siguiente forma

$$u_i^{m+1,[0]} = [I - \Delta t\ A(t^{m+1})]^{-1}[u_i^{m,[0]} + \Delta t\ b^{m+1}] \tag{4.13}$$

donde u es la aproximación de la solución, m representa en instante de tiempo ($m+1$ seria el nuevo valor) y [0] indica que estamos en el nivel de predicción $p = 0$ y el subíndice i representa la discretización en el espacio.

4.2.2. Nivel de corrección

Tomando como punto de partida el valor de (4.9) y la función de corrección propuesta en [1] obtenemos lo siguiente

$$u_i^{m+1,[p+1]} - u_i^{m,[p]} - \Delta t\ f(t^{m+1}, u_i^{m+1,[p+1]}) + \Delta t\ f(t^{m+1}, u_i^{m+1,[p]}) = \int_{t^m}^{t^{m+1}} f(t, u_i^{[p]}(t))dt \tag{4.14}$$

sustituyendo el valor de $f(t^{m+1}, n_i^{m+1,[p+1]})$ obtenemos

$$u_i^{m+1,[p+1]} - u_i^{m,[p]} - \Delta t\ [A(t^{m+1})u_i^{m+1,[p+1]} + b(t^{m+1})] + \Delta t\ [A(t^{m+1})u_i^{m+1,[p]} + b(t^{m+1})] =$$

$$\int_{t^m}^{t^{m+1}} [A(t)u_i^{m,[p]} + b(t^m)]dt \tag{4.15}$$

multiplicando y agrupando términos obtenemos

$$[I - \Delta t\ A(t^{m+1})]u_i^{m+1,[p+1]} = u_i^{m,[p]} - \Delta t\ A(t^{m+1})u_i^{m+1,[p]} +$$

$$\int_{t^m}^{t^{m+1}} [A(t^m)u_i^{m,[p]} + b(t^m)]dt \tag{4.16}$$

de esta forma el nivel de corrección se calcula como

$$u_i^{m+1,[p+1]} = \left[I - \Delta t\ A(t^{m+1})\right]^{-1} \left[u_i^{m,[p]} - \Delta t\ A(t^{m+1})u_i^{m+1,[p]}\right]$$

$$+ \left[\int_{t^m}^{t^{m+1}} [A(t^m)u_i^{m,[p]} + b(t^m)]dt\right] \tag{4.17}$$

donde $p \in \{1, 2, 3, ...M\}$ según la cantidad de correcciones que se requieran para encontrar la solución.

4.3. Modelos y Resultados

4.3.1. Modelo 1. EDP de segundo grado

Se desea resolver la ecuación de calor en una dimensión

$$\begin{cases} U_t = & U_{xx} + f \\ U(a) = & 0 \\ U(b) = & 0 \end{cases} \tag{4.18}$$

donde

$$f = (x-a)(x-b)(cos(t)) - 2sin(t) - 4 \tag{4.19}$$

es un término fuente y las condiciones iniciales para $t_0 = 0$ son

$$U(t_0) = (x-a)(x-b)(sin(t_0) + 2) \tag{4.20}$$

Utilizando el esquema de Euler implícito como lo vimos en (4.10) obtenemos

$$\frac{u_i^{n+1,[0]} - u_i^{n,[0]}}{\Delta t} = \frac{u_{i-1}^{n+1,[0]} - 2u_i^{n+1,[0]} + u_{i+1}^{n+1,[0]}}{(\Delta x)^2} + f_i^{n+1} \tag{4.21}$$

factorizando términos y despejando

$$u_i^{n+1,[0]} - u_i^{n,[0]} = \frac{\Delta t}{(\Delta x)^2}\left(u_{i-1}^{n+1,[0]} - 2u_i^{n+1,[0]} + u_{i+1}^{n+1,[0]}\right) + \Delta t\ f_i^{n+1} \tag{4.22}$$

simplificando obtenemos

$$-\frac{\Delta t}{(\Delta x)^2}u_{i-1}^{n+1,[0]} + \left(1 + \frac{2\Delta t}{(\Delta x)^2}\right)u_i^{n+1,[0]} - \frac{\Delta t}{(\Delta x)^2}u_{i+1}^{n+1,[0]} = u_i^{n,[0]} + \Delta t\ f_i^{n+1} \tag{4.23}$$

por lo tanto

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots \\ \alpha & \beta & \gamma & 0 & 0 & 0 & \dots \\ 0 & \alpha & \beta & \gamma & 0 & 0 & \dots \\ 0 & 0 & \alpha & \beta & \gamma & 0 & \dots \\ 0 & 0 & 0 & \alpha & \beta & \gamma & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.24}$$

donde

$$\alpha = \frac{-\Delta t}{(\Delta x)^2} \qquad \beta = 1 + \frac{2\Delta t}{(\Delta x)^2} \quad y \quad \gamma = \frac{\Delta t}{(\Delta x)^2} \tag{4.25}$$

y b para (4.17) y (4.13) seria

$$b = u_i^n + \Delta t \ f_i^{n+1} \tag{4.26}$$

La solución analítica del problema es la siguiente

$$U_t = (x-a)(x-b)(cos(t))$$

$$U_x = (sin(t)+2)[(x-a)+(x-b)]$$

$$U_{xx} = 2(sin(t)+2)$$

la solución analítica en el ultimo instante de tiempo tiene la siguiente *figura.*

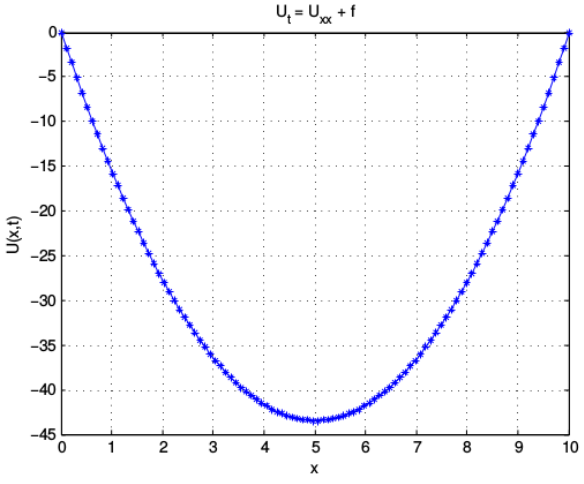


FIGURA 4.1: Gráfica de la solución analítica para la ecuación

Para realizar el cálculo numérico usaremos los siguientes parámetros:

1. Intervalo de evaluación en el tiempo $t \in [0, 50]$
2. Intervalo de evaluación en el espacio $x \in [0, 10]$
3. $\Delta t \in \{0.4,\ 0.005\}$
4. $\Delta x = 0.1$
5. $N \in \{100, 1000\}$
6. $M : 4$ (predictor, 3 correcciones)
7. Errores $\{\varepsilon_0,\ \varepsilon_1,\ \varepsilon_2,\ \varepsilon_3\}$

De la solución numérica obtenemos lo siguiente.

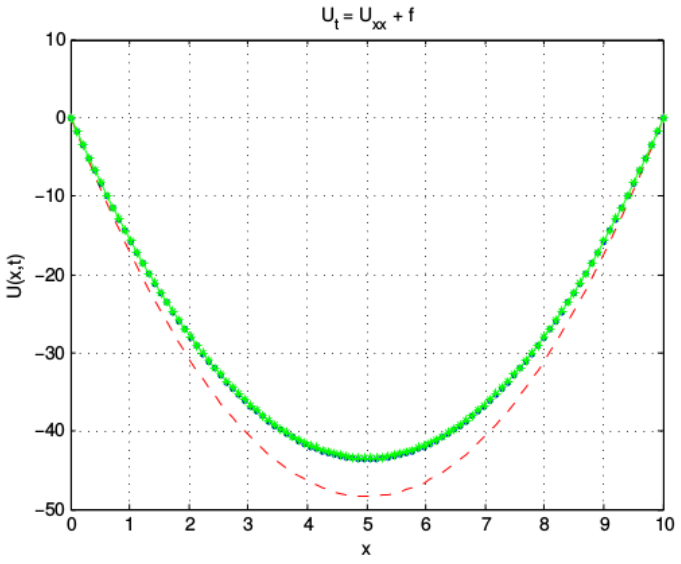


FIGURA 4.2: Gráfica de la solución numérica para la ecuación para $\Delta t = 0{,}4$. La línea verde hace referencia a la función exacta, mientras que la línea punteada en rojo representa el valor aproximado calculado en el nivel de predicción, mientras que la línea azul que se encuentra justo por debajo de la verde es la tercera corrección.

En las figuras (4.2) y (4.3) se muestran los resultados para distintos tamaños de paso, lo interesante de las figuras es que el valor de la predicción con el tamaño de paso mas pequeño ($\Delta t = 0.005$) es muy cercano al valor de la tercera corrección en $\Delta t = 0.4$.

La ventaja de lo mencionado es que el cálculo se realiza mucho más rápido para el tamaño de paso más grande y los resultados con muy parecidos, de esta forma podemos notar las enormes ventajas del método (*ver cuadro* 4.1).

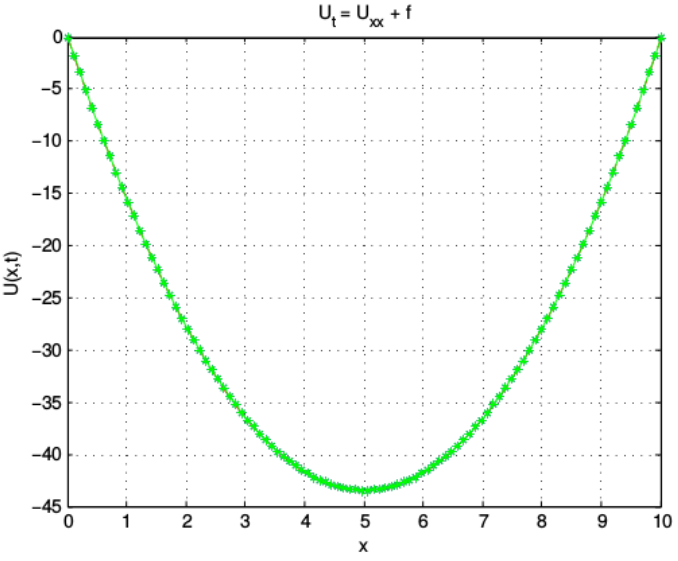


FIGURA 4.3: Gráfica de la solución numérica para la ecuación para $\Delta t = 0{,}005$. La línea verde hace referencia a la función exacta, la línea punteada en rojo representa el valor aproximado calculado en el nivel de predicción, mientras que la linea azul que se encuentra justo por debajo de la verde es la tercera corrección.

	Δt	N	ε_0	ε_1	ε_2	ε_3
a)	0.4	125	1.0968e-01	3.4620e-03	1.1795e-03	1.1493e-03
b)	0.005	10,000	1.3742e-03	5.3539e-07	1.7523e-07	1.7521e-07

CUADRO 4.1: Resultados del error RMS de la solución aproximada en el nivel de predicción (ε_0) y en los niveles de corrección (ε_1, ε_2, ε_3) comparados con el resultado de la función analítica, para distintos tamaños de paso (Δt) con una discretización en el espacio de 100 nodos y $\Delta x = 0.1$

	Δt	N	ε_0	ε_1	ε_2	ε_3
c)	0.005	10,000	1.3730e-03	5.3461e-07	1.7197e-07	1.7191e-07

CUADRO 4.2: Resultados del error RMS de la solución aproximada en el nivel de predicción (ε_0) y en los niveles de corrección (ε_1, ε_2, ε_3) comparados con el resultado de la función analítica, para distintos tamaños de paso (Δt).

En el *cuadro* (4.2) se realizó el mismo análisis que en el *cuadro* (4.1) con la diferencia de que la discretización en el espacio aumento de 100 a 1000 nodos, observemos que este cambio no altera el resultado de los errores ya que se mantienen en el mismo orden de magnitud, de esta forma sabemos también podemos notar que el cálculo se realiza más rápidamente con una discretización más grande en el espacio.

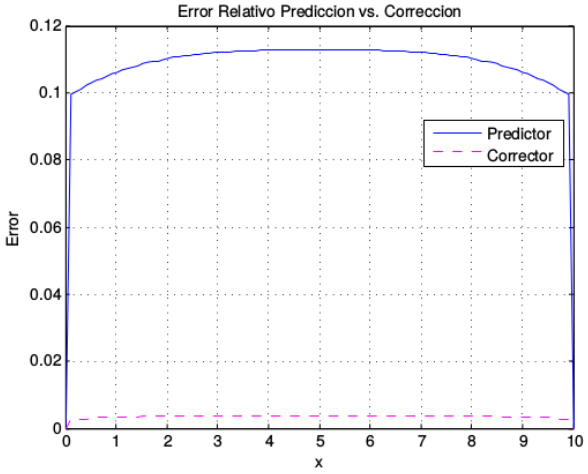


FIGURA 4.4: Error numérico de la predicción y la ultima corrección ϵ_3

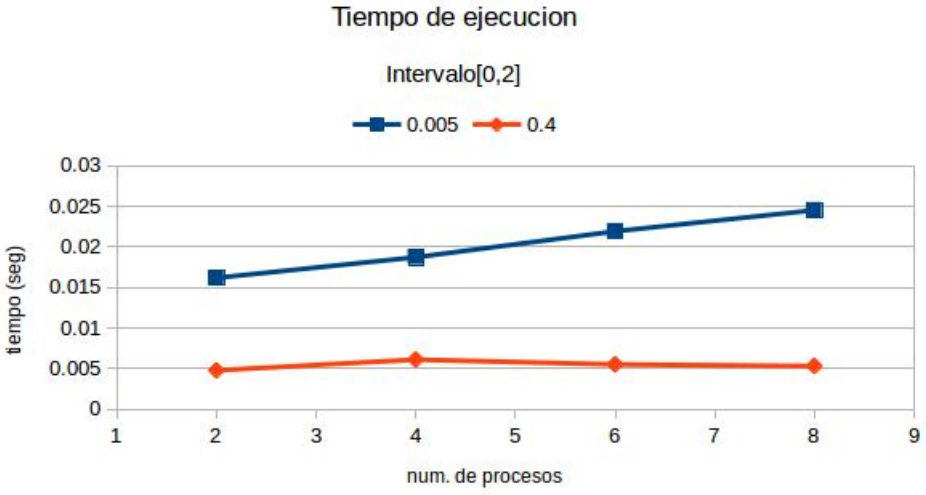


FIGURA 4.5: Tiempo de ejecución para distintos tamaños de Δt, con pocos nodos en la discretización en el tiempo

En cuanto a los tiempos de ejecución podemos notar que en discretizaciones con mayor o menos cantidad de nodos el tiempo de ejecución para un $\Delta t = 0.4$ son mucho menores a los tiempo de un $\Delta t = 0.005$.

Ese comportamiento parece muy lógico debido a que es grande la diferencia entre ellos y la intuición nos dice que es lo que debe ocurrir, lo interesante de ambas gráficas es que aun cuando aumentamos el intervalo de evaluación los tiempos para $\Delta t = 0.4$ se mantuvieron cercanos mientras que los tiempos de ejecución para $\Delta t = 0.005$ aumentaron casi al doble de los que se tienen en el intervalo más corto.

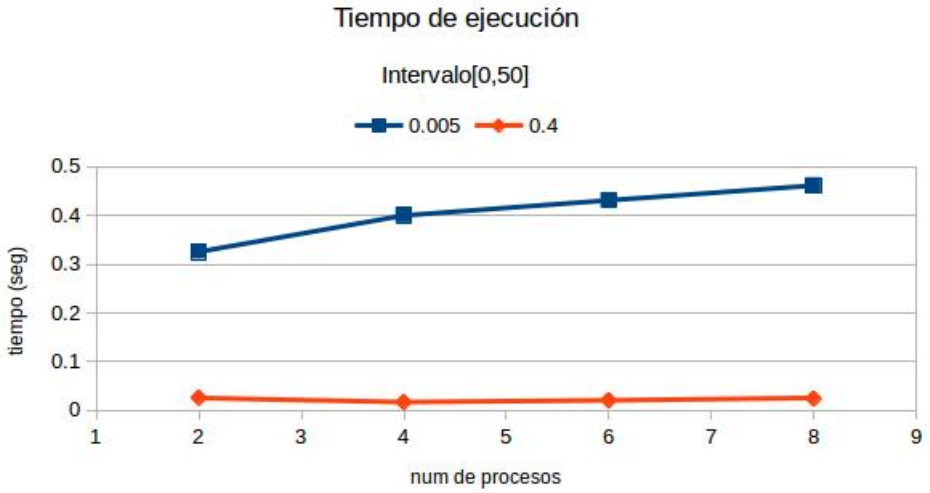


FIGURA 4.6: Tiempo de ejecución para distintos tamaños de Δt, con muchos nodos en la discretización en el tiempo

Capítulo 5

Conclusiones y trabajo a futuro

Como en todo trabajo de investigación, se plantean objetivos y existe una motivación principal sobre la que encaminamos nuestro trabajo. Conforme indagamos en el tema que estamos estudiando y obtenemos resultados, en ocasiones favorables y en otras no tanto como quisieramos, siempre se aprende en el proceso.

En el caso de esta tesis nuestro principal motivante fue la solución de ED con algoritmos naturalmente paralelizables y como resultados vimos en el capítulo III y IV que el método propuesto ofrece ventajas competitivas al momento de determinar aproximaciones a la solución en el caso de EDOs, mientras que para EDPs tenemos tanto resultados muy favorables en tiempo de ejecución como aproximaciones a la solución muy precisas.

Es importante mencionar que en un principio uno de nuestros principales retos fue resolver la integral planteada en la ecuación de corrección (2.11) que parecía variar mucho el resultado de la solución aproximada que buscábamos, para al final darnos cuenta que con métodos muy simples como la regla del trapecio obteníamos resultados muy competitivos. De acuerdo con esto aun podemos indagar en métodos mas sofisticados para conocer nuevos y muy probablemente mejores resultados.

Si bien estos experimentos nos demuestran que el método tiene bondades aprovechables existe un amplio trabajo todavía por hacer.

Uno de los principales y que se plantea en el artículo principal para la elaboración de esta tesis [1], es el uso de descomposición de dominios para trabajar la parte espacial de los problemas en EDP, hasta el momento podemos observar claramente

las ventajas del uso de los algoritmos predictor-corrector, pero aun con esto valdría la pena implementar la parte de descomposición de dominios.

Otro elemento que ya esta incluido en el algoritmo que presentamos en esta tesis, pero que podemos experimentar más a fondo es la creación de subgrupos tanto en la parte de la predicción como en la corrección del método (*veáse Capitulo II*), donde aun no podemos concluir que tan grandes o pequeños deben ser estos subgrupos y como puede llegar a afectar esto a la solución, haría falta trabajar en pruebas que nos permitan profundizar este tipo de información.

Además dentro de los trabajos que en un futuro se pueden realizar con información como esta, es llevar el método a su aplicación. En un principio una de las motivaciones para trabajar con este tipo de algoritmos y con este tema, fue su aplicación en la en la industria petrolera resolviendo la denominada *ecuación del pozo* con la finalidad principal de medir la caída de presión en un pozo petrolero así como determinar la producción del mismo en periodos de tiempo largos, donde métodos como el visto en esta tesis tienen amplias posibilidades de actuar, principalmente por que una de sus ventajas es poder dar pasos grandes en el tiempo manteniendo una aproximaciones de orden alto. Cabe mencionar que este tipo de cálculos es muy costoso computacionalmente y los cálculos pueden realizarse en tiempos (de ejecución) muy extensos, por lo tanto el trabajar con algoritmos que aminoren estos costos y sigan manteniendo una buena aproximación a al solución resulta en grandes beneficios. Si bien es un problema que todavía no atacamos de manera directa, ahora tenemos más herramientas para trabajarlo en un futuro cercano.

Además de la aplicación en pozos petroleros, existen otras diversas aplicaciones en las que podemos utilizar este tipo de métodos como es en la estimación de parámetros en algoritmos de optimización, así como en problemas donde el calculo de EDP sea un problema difícil de atacar.

Apéndice A

Regla del Trapecio

Se representa mediante la formula

$$\int_a^b f(x)dx = \frac{h}{2}[f(x_0) + f(x_1)] - \frac{h^3}{12}f''(\xi) \tag{A.1}$$

es conocida como la regla del trapecio debido a que cuando f es una función con valores positivos, aproximamos $\int_a^b f(x)dx$ por el área de un trapecio como se muestra en la figura (A.1). como el término de error de la regla del trapecio

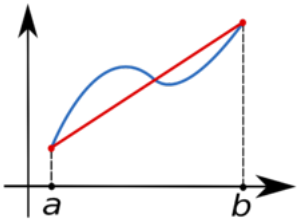


FIGURA A.1: Se ilustra el área del trapecio.

contiene f'', la regla da el resultado exacto cuando se aplica a una función cuya segunda derivada sea cero, es decir, con cualquier polinomio de grado 1 o menos. [6]

Apéndice B

Integración de Romberg

En esta integración se utiliza la regla compuesta del trapecio para obtener aproximaciones preliminares y luego el proceso de extrapolación de Richardson para mejorar las aproximaciones.[6] El Método de Romberg genera una matriz triangular cuyos elementos son estimaciones numéricas de la integral definida. El método se define como:

$$R(0,0) = \frac{1}{2}(b-a)(f(a)+f(b))$$

$$R(n,0) = \frac{1}{2}R(n-1,0) + h_n \sum_{k=1}^{2^{n-1}} f(a+(2k-1)h_n)$$

$$R(n,m) = \frac{1}{4^m-1}(4^m R(n,m-1) - R(n-1,m-1))$$

donde $n \geq 1$, $m \geq 1$, $h_n = \frac{b-a}{2^n}$, la regla en $R(0,0)$ es equivalente a la regla del trapecio.

El método de Romberg de que permite calcular íntegramente un nuevo renglón en la tabla con solo hacer una aplicación mas de la regla compuesta del trapecio, y luego ponderar los valores previamente calculados para obtener los elementos sucesivos del renglón.

El método con el que se construye una tabla de este tipo calcula los elementos renglón por renglón , es decir en el orden $R_{1,1}$, $R_{2,1}$, $R_{2,2}$, $R_{3,1}$, $R_{3,2}$, $R_{3,3}$, etc. Se puede utilizar el siguiente algoritmo de [6].

B.1. Algoritmo

Algorithm Integración de Romberg

Require: Para aproximar la integral $I = int_a^b f(x)dx$, seleccione $n > 0$. Se requiere a y b, así como n.

Tome $h = b - a$; $R_{1,1} = \frac{h}{2}(f(a) + f(b))$
salida $R_{1,1}$
for l = 2 **to** n **do**
  Tome $R_{2,1} = \frac{1}{2}\left[R_{1,1} + h\sum_{k-1}^{2^{i-1}} f(a + (k - 0{,}5)h)\right]$.
  for j = 2 **to** i **do**
    $R_{2,j} = R_{2,j-1} + \frac{R_{2,j-1} - R_{1,j-1}}{4^{j-1}-1}$.
  end for
  Salida $R_{2,j}, para j = 1, 2, ..., i$
  Tome $h = h/2$
  Para $j = 1, 2, ..., i$ tome $R_{1,j} = R_{2,j}$ (Actualiza el renglón 1 de R)
end for

Apéndice C

Interpolación de Polinomio

Aunque el polinomio de Newton y el de Lagrange son adecuados para determinar valores intermedios entre puntos, no ofrecen un polinomio adecuado de la forma convencional.

$$f(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n \tag{C.1}$$

Un método directo para calcular los coeficientes de este polinomio se basa en el hecho de que se requieren n + 1 puntos para determinar los n + 1 coeficientes. Así, se utiliza un sistema de ecuaciones algebraicas lineales simultáneas para calcular los coeficientes a_i. Por ejemplo, si se desea calcular los coeficientes de la parábola

$$f(x) = a_0 + a_1x + a_2x^2 \tag{C.2}$$

Se requiere de tres puntos: $[x_0, f(x_0)], [x_1, f(x_1)]y[x_2, f(x_2)]$ Cada uno se sustituye en la ecuación anterior.

$$f(x_0) = a_0 + a_1x_0 + a_2x_0^2$$

$$f(x_1) = a_0 + a_1x_1 + a_2x_1^2$$

$$f(x_2) = a_0 + a_1x_2 + a_2x_2^2 \tag{C.3}$$

De esta manera, las x son los puntos conocidos, y las a las incógnitas. Como hay el mismo número de ecuaciones que de incógnitas, la ecuación se podría resolver con algún método de eliminación. En resumen, si usted se interesa en determinar un punto intermedio, emplee la interpolación de Newton o de Lagrange. Si tiene que determinar una ecuación de la forma anterior, limítese a polinomios de grado menor y verifique cuidadosamente sus resultados

Apéndice D

Error Cuadrático Medio

$$RMSError = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{D.1}$$

Este error mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima. El ECM ó RMS es una función de riesgo, correspondiente al valor esperado de la pérdida del error al cuadrado o pérdida cuadrática. La diferencia se produce debido a la aleatoriedad o porque el estimador no tiene en cuenta la información que podría producir una estimación más precisa.

Bibliografía

[1] Christlieb, A.J., Haynes, R.D. y Ong B.W. , *A parallel space-time algorithm*, SIAM Journal SCI. Computing, 2012.

[2] Christlieb, A.J., Morton, M. y Ong B.W. ,*Semi-implicit integral deferred correction constructed with additive Runge–Kutta methods*, Commun. Math. Sci., 9, 2011. pp. 879-902.

[3] Christlieb, A.J., Macdonald, C.B. y Ong B.W. ,*Parallel high-order integrators*, SIAM J. Sci. Comput., 2010. pp. 818–835.

[4] Christlieb, A.J. y Ong B.W. ,*Implicit parallel time integrators*, J. Sci. Comput., 49, 2011. pp. 167–179.

[5] Dutt, A., Greegard, L. y Rokhlin V. , *Spectral deferred correction methods for ordinary differential equations*, BIT, 40 (2000) pp. 241-266.

[6] Burden, R.L. y Douglas J.,*Análisis Numérico*, septima edición, Thomson Learning, 2002.

[7] Professor Causon,D.M., Professor Mingham C.G., *Introductory Finite Difference Methods for PDEs*, Ventus Publishing ApS, (2010).

[8] Braun, Martin,*Differential Equations and Their Applications. An Introduction to Applied Mathematics*, tercera edición, Springer-Verlag, 1983.

[9] Stoer, J. y Bulirsch R., *Introduction to Numerical Analysis*, tercera edición, Springer, 2002.

[10] Drozdowski, Maciej,*Scheduling for Parallel Processing*, primera edición, Springer, 2009.

[11] Evans, L.C., *Partial Differential Equations*, primera edición, American Math Society, 2010.

[12] Christlieb, A.J., Ong B.W. y Qiu J.M. , *Comments on high order integrators embedded within integral deferred correction methods*,Comm. Appl. Math. Comput. Sci., 2009.

[13] Christlieb, A.J., Ong B.W. y Qiu J.M. , *Integral deferred correction methods constructed with high order Runge-Kutta integrators*, Math. Comp., 2009.

[14] Gander, M. y Vandewalle S., *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 2007.pp. 556–578.

[15] Thomas, J.W., *Numerical Partial Differential Equations (Finite Difference Methods)*, Springer, 1995.

[16] Jianping, Zhu, *Solving Partial Differential Equations on Parallel Computers* , Mississippi State Univ., 1994.

[17] Vargas, M., *Optimización de código* , Notas Cimat., 2010.

[18] Nievergelt, J., *Parallel methods for integrating ordinary differential equations* ,Commun. ACM, 7 (1964), pp. 731–733.