Trabajo Práctico I

Matías Pérez (2/05)
Alberto Bertogli (763/06)
Rodrigo Campos Catelin (561/06)

September 10, 2010

Abstract

El siguiente trabajo trata sobre los problemas de moderlar los números reales mediante una aritmética finita. Con tal fin, se usará la aritmética del computador (base 2) y se analizará el comportamiento de distintos algoritmos iterativos variando la precisión de los números involucrados.

Palabras clave: aritmética finita, error numérico, precisión arbitraria, algoritmos iterativos.

1 Introducción teórica

Existen varios métodos iterativos para aproximar$\sqrt{2}$. Para la realización de este trabajo consideramos los siguientes métodos:

- Método de la serie binomial:

$$(1+x)^n = 1 + n\,x + \frac{n\,(n-1)}{2!}x^2 + \frac{n\,(n-1)\,(n-2)}{3!}x^3 + \dots$$

  - Para estimar $\sqrt{2}$ se suma una cantidad finita k de términos de la serie, con $n = 1/2$ y $x = 1$.

- Método de las fracciones contínuas

$$\sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \dots}}}$$

  - Este método consiste en sumar una cantidad finita k de términos de la fracción continua presentada.

- Método babilonio

$$x_0 = A$$
$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{2}{x_n}\right)$$

  - Este método es una adaptación del método de Newton-Raphson para este caso particular, pero es conocido desde por lo menos el siglo II A.C. Comenzando desde una aproximación A de $\sqrt{2}$, la sucesión converge al valor buscado.

2 Desarrollo

2.1 Artimética de precisión arbitraria

Para realizar los experimentos se observó que es necesario contar con una aritmética de precisión arbitraria con las operaciones de suma, resta multiplicación y división.

Primero se pensó en hacer una clase que represente números de punto flotante, a la cual se le pueda fijar la presicion de manera dinámica. Para realizar las operaciones de la misma, se evaluó la posibilidad de implementar "manualmente" dichas operaciones.

Luego se optó por un enfoque más simple: representar internamente el número con un double (*IEEE 754 binary64*). De esta forma, ya que el objetivo es lograr que la precisión sea t < 52, y sabiendo que un double tiene 52 bits en la mantisa (lo que define la presicion), el mismo alcanza para representar cualquier numero con presicion t. Lo que nos permite realizar las operaciones de suma, resta, multiplicación y división utilizando como base las provistas por el procesador.

Otro aspecto relevante es la cantidad de bits del exponente, es decir, cuántos bits asignarle. Luego de discutir el tema y considerar que el exponente sólo define el alcance de la representación (cuál será el mayor y el menor numero representable), se optó por utilizar la misma cantidad de bits para el exponente que la que se utiliza para representar un double. De esta forma solo tendremos que ocuparnos de la mantisa de dicha representación. Cabe mencionar que para esta decisión se tomo en consideración que las operaciónes aritméticas a realizar son todas con el fín de calcular $\sqrt{2}$.

La aritmética finalmente funciona de la siguiente manera:

Cuando se genera un numero con presicion t, lo que se hacer es guardar internamente el mismo como un double, pero se le quita presicion hasta llegar a la presicion t buscada.

Luego, para realizar alguna operacion, se toman los dos operandos (ya con la presicion correcta) y se opera utilizano los doubles de la representacion interna. Una vez obtenido este resultado se genera nuevamente un numero con presicion t a partir de este.

2.1.1 Métodos de aproximación de la mantisa

Para adecuar la precisión de un número a la presición t buscada, se implementaron dos métodos: truncamiento y redondeo.

El primero consiste en poner en cero todos los bits de la mantisa que estén en posiciones mayores a t. Para esto se utilizó una máscara de bits y se realizó un and lógico (de bits) sobre los bits de la mantisa y la máscara.

Para implementar el método de redondeo es necesario considerar el bit de la posición t+1 de la mantisa. Lo que se traduce en sumarle al numero original el valor que representa ese bit para luego truncar el resulado a precisión t. Para realizar esto se realiza un truncamiento del numero a presicion $t + 1$ y otro con

presicion t, al restar estos dos numeros se obtiene el numero que representa el bit $t+1$. Por ultimo se suma al numero original este valor y se lo trunca a presicion t, como se explicó anteriormente.

2.2 Método binomial

Para este método se realizaron distintas implementaciones teniendo en cuenta ciertas particularidades que se notaron de la fórmula:

- que el numerador y denominador de cada término es un producto
- que cada término es igual al anterior multiplicado por una constante que depende de la iteración actual
- que la fórmula se puede expresar como una sumatoria

La primera implementación, *binomial simple*, calcula para cada término el numerador y el denominador y, luego, el resultado de la division. Mediante este proceso se calculan todos los terminos (hasta cierto n variable). Y a medida que se genera cada término, se lo suma en una variable que finalmente tendrá el resultado buscado.

La segunda implementación, *binomial incremental*, consta en dividir cada termino en productos de fracciones, esto es posible gracias a que:

$$(1+x)^n = 1 + n\,x + \frac{n\,(n-1)}{2!}x^2 + \frac{n\,(n-1)\,(n-2)}{3!}x^3 + \ldots$$

$$(1+x)^n = 1 + \sum_{i=0}^{\infty} a_i$$

Con,$a_i = \prod_{j=0}^{i} b_j$ y $b_j = \frac{n-j}{j+1}$.

Se puede ver tambien que $a_{i+1} = a_i.b_{i+1}$. Sabiendo esto, lo que se hizo en cada paso fue, en vez de recalcular todo el termino se utiliza el termino anterior y a partir de este (multiplicando por el valor de b_j) se obtiene el termino siguiente.

La tercera y ultima implementacion, *binomial decremetal*, usa la propiedad conmutativa de la suma, ya que suma los terminos en el orden inverso. Con este fin se calcularon los terminos de manera similar a la implementacion anterior, pero en vez de sumarlos, se los guarda en una lista. Finalmente se recorre la lista de atras haca adelante realizando, ahora sí, la sumatoria de los terminos.

2.3 Método de las fracciones continuas

La primera implementacion planteada para este metodo consiste en realizar el calculo de la fraccion partiendo desde la más interior hasta llegar a la exterior. La implementacion, de este modo, no tiene mayores complicaciones.

Otro enfoque que se usó fué el de no realizar las divisiones de las fracciones hasta la última iteración, esto es más sencillo de ver de la siguiente manera:

$$\sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2+\dots}}}$$

$$\sqrt{2} = 1 + \lim_{n \to \infty} a_n$$

Siendo:

$$\begin{cases} a_1 = \frac{1}{2} \\ a_{i+1} = \frac{1}{2+a_i} \end{cases}$$

A partir de esto, si se considera: $a_i = \frac{b_i}{c_i}$, se obtiene:

$$a_{i+1} = \frac{1}{2 + a_i} = \frac{1}{2 + \frac{b_i}{c_i}} = \frac{1}{\frac{2c_i + b_i}{c_i}} = \frac{c_i}{2c_i + b_i} = \frac{b_{i+1}}{c_{i+1}}$$

Finalmente se obtiene:

$$\sqrt{2} = 1 + \lim_{n \to \infty} \frac{b_n}{c_n}$$

Con:

$$\begin{cases} b_1 = 1 \\ b_{i+1} = c_i \end{cases}$$

$$\begin{cases} c_1 = 2 \\ c_{i+1} = 2c_i + b_i \end{cases}$$

Utilizando esto es posible no realizar ninguna division hasta la ultima iteración.

2.4 Metodo Babilonio

Para este método sólo se realizó una implementación, la misma es una traduccion bastante literal de la formula que el método presenta. La aproximacion inicial A se fijó como 2, pero se estableció también como parámetro opcional de entrada. De esta forma es posible ver cómo el mismo varía según el A elegido.

3 Resultados

Para analizar el comportamiento de los algoritmos y los metodos implementados, se realizaron distintos experimentos. A continuacion se incluyen gráficos sobre los comportamientos más relevantes.

3.1 Comparativa general

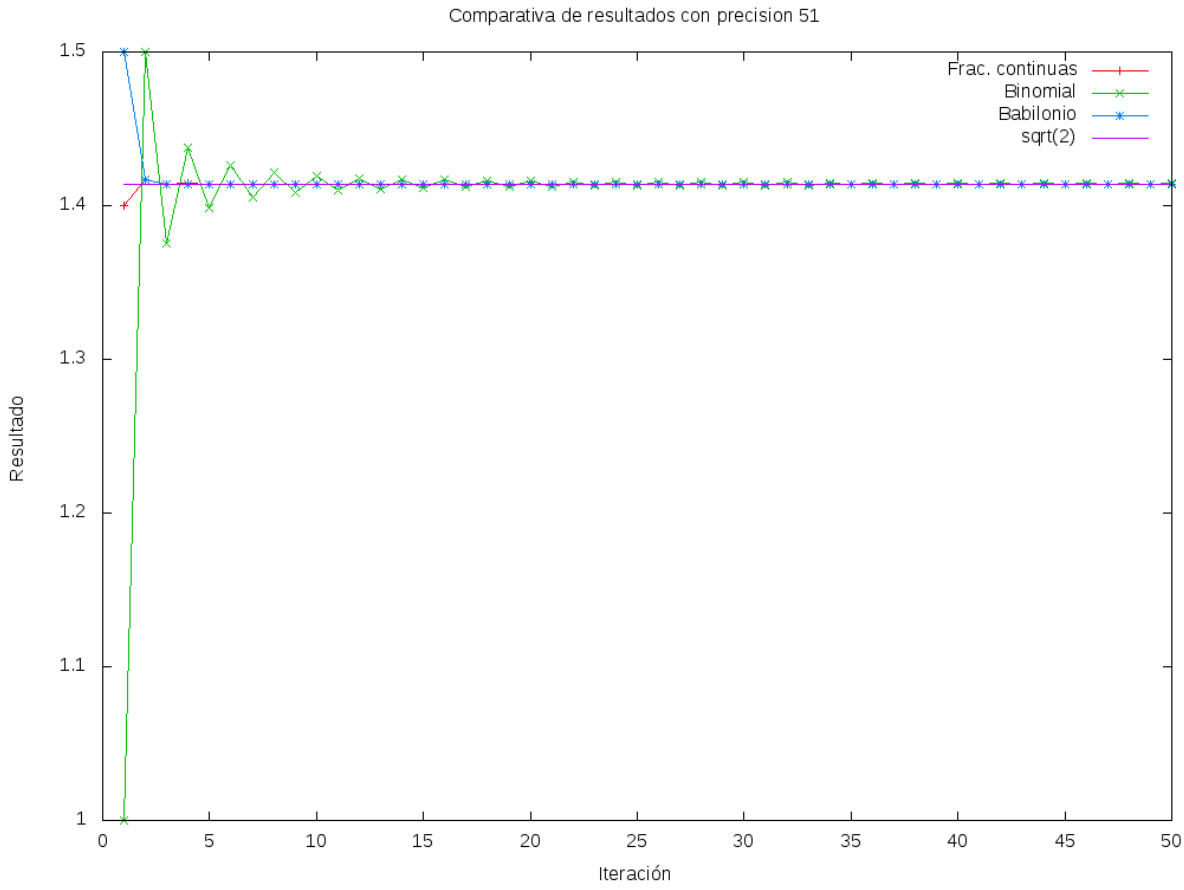

Figura 1: Comparativa de resultados de distintos métodos, con precisión 51

Nuestro primer gráfico muestra el resultado obtenido por cada método conforme aumenta la cantidad de iteraciones, en la máxima precisión. Se grafican solo las implementaciones mas básicas por claridad, ya que el objetivo es presentar visualmente el funcionamiento de cada método, y las variantes no presentan comportamientos significativamente distinto, como se verá en gráficos subsecuentes.

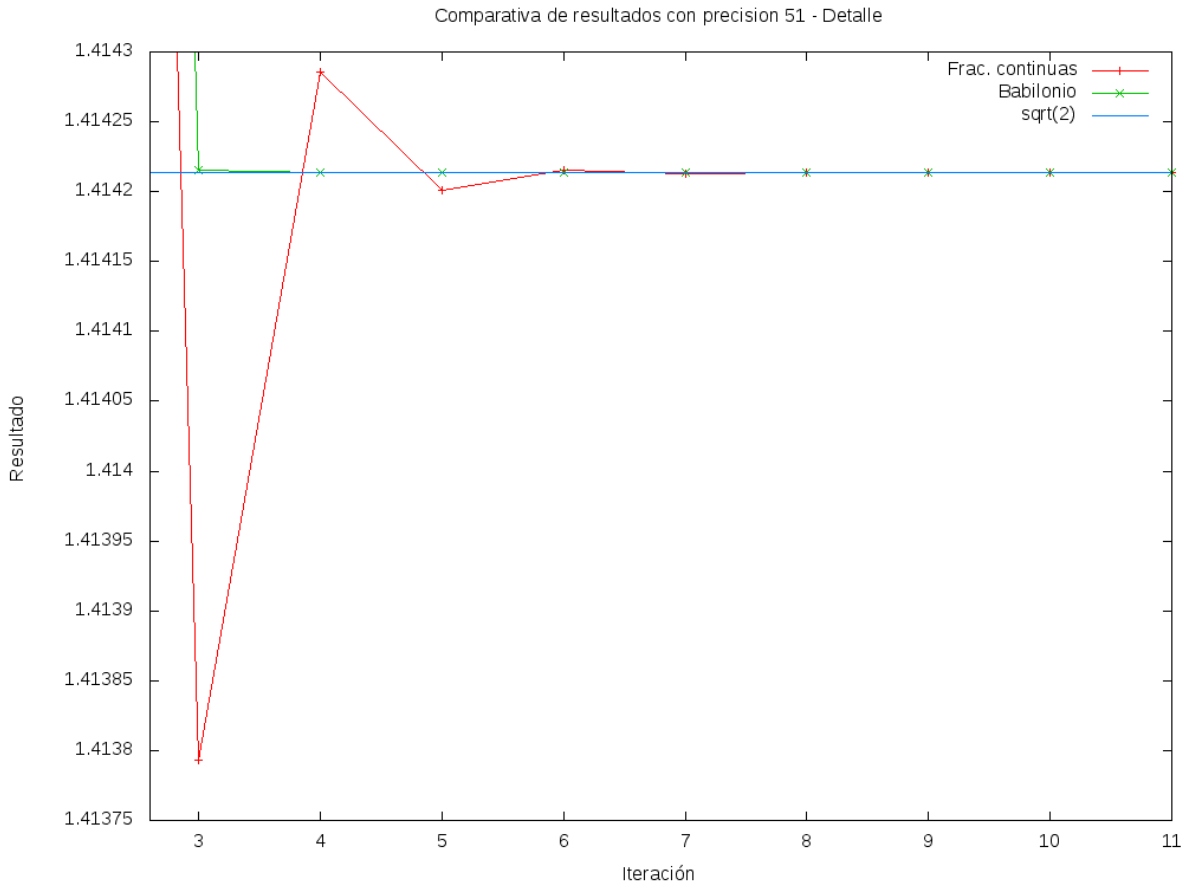

Figura 2: Detalle de la figura 1

Este detalle de la figura anterior permite apreciar con mas detalle como convergen los métodos de fracciones continuas y babilonio, aun con la máxima precisión. Como se verá en otros gráficos, fue casi una constante que los métodos tardaban mas iteraciones en obtener su mejor resultado segun aumentaba la precision.

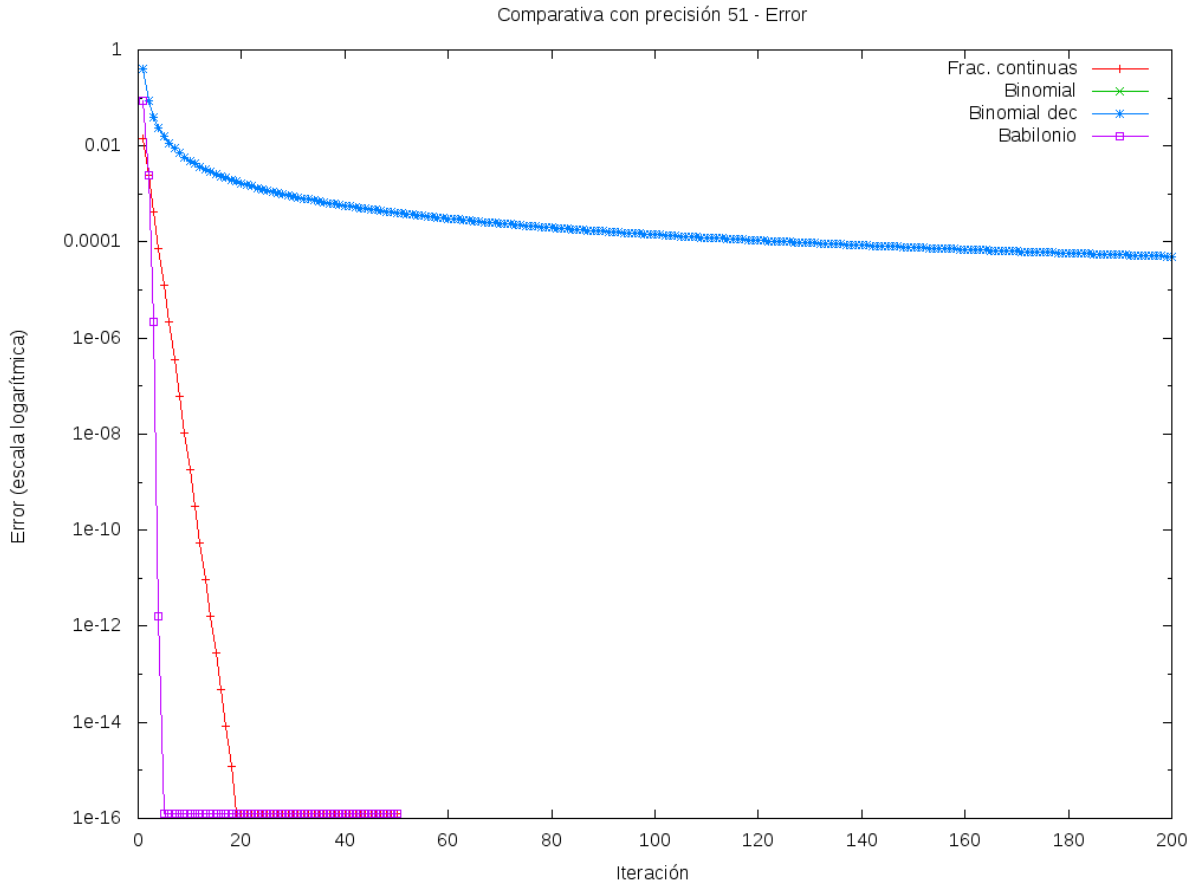

Figura 3: Error en los distintos métodos segun la iteración, con precisión fija en 51

Aquí nuevamente se utiliza la máxima precisión para ilustrar la convergencia de los distintos métodos conforme aumenta la cantidad de iteraciónes. En este caso, nos concentramos en el error del resultado obtenido en cada caso. Utilizamos una escala logarítmica para el mismo, con el propósito de incrementar la legibilidad del gráfico.

De ahora en mas graficaremos casi exclusivamente el error, por considerarlo mas representativo a fines de la realización de un análisis comparativo de los métodos.

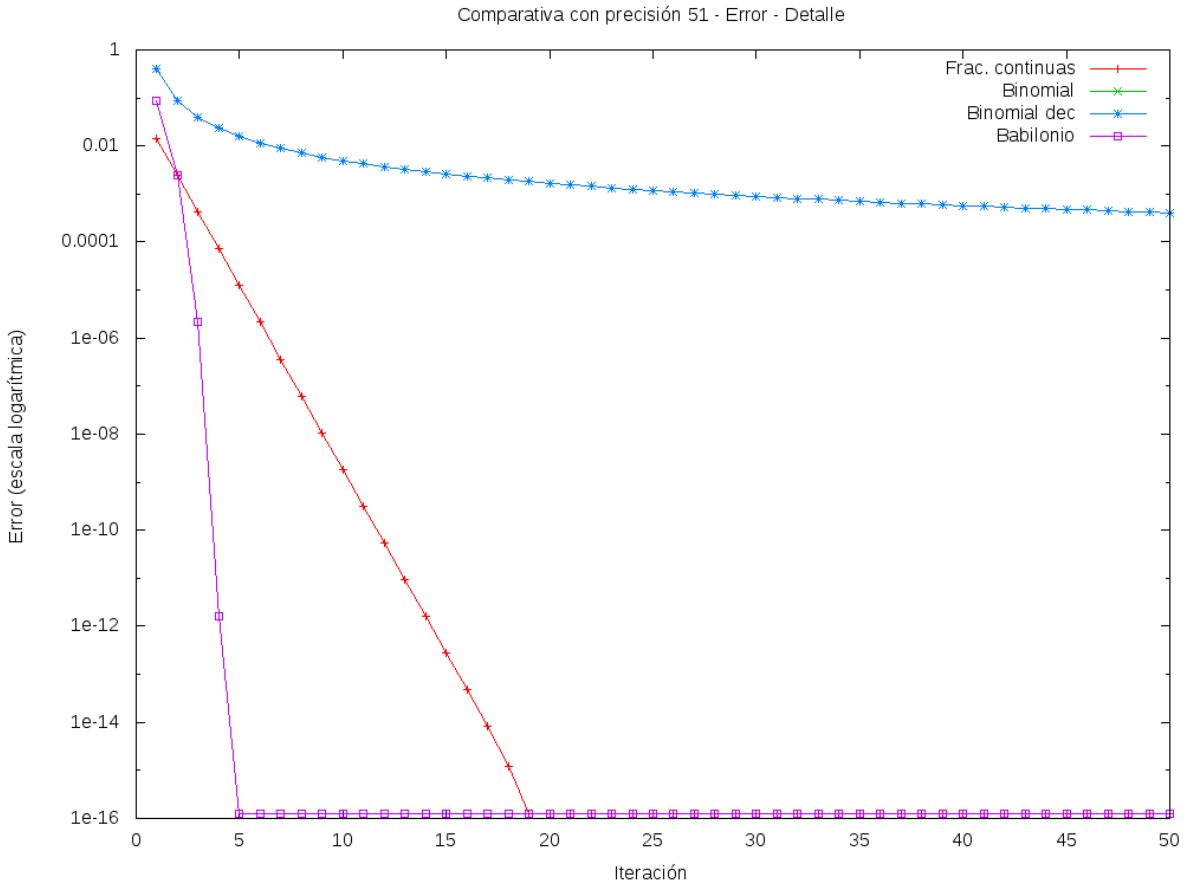

Figura 4: Detalle de la figura 3

Este detalle de la figura anterior nos permite ver claramente la fuerte diferencia en la convergencia de los tres métodos. Mientras que el babilonio alcanza el mejor resultado en la 5ª iteración, y el de fracciónes continuas en la 19ª, ambos estabilizandose, el método binomial (en su variante decreciente, aunque como veremos en la sección correspondiente, no hay diferencia significativa en este caso en ninguno de las variantes) aproxima de forma extremadamente lenta.

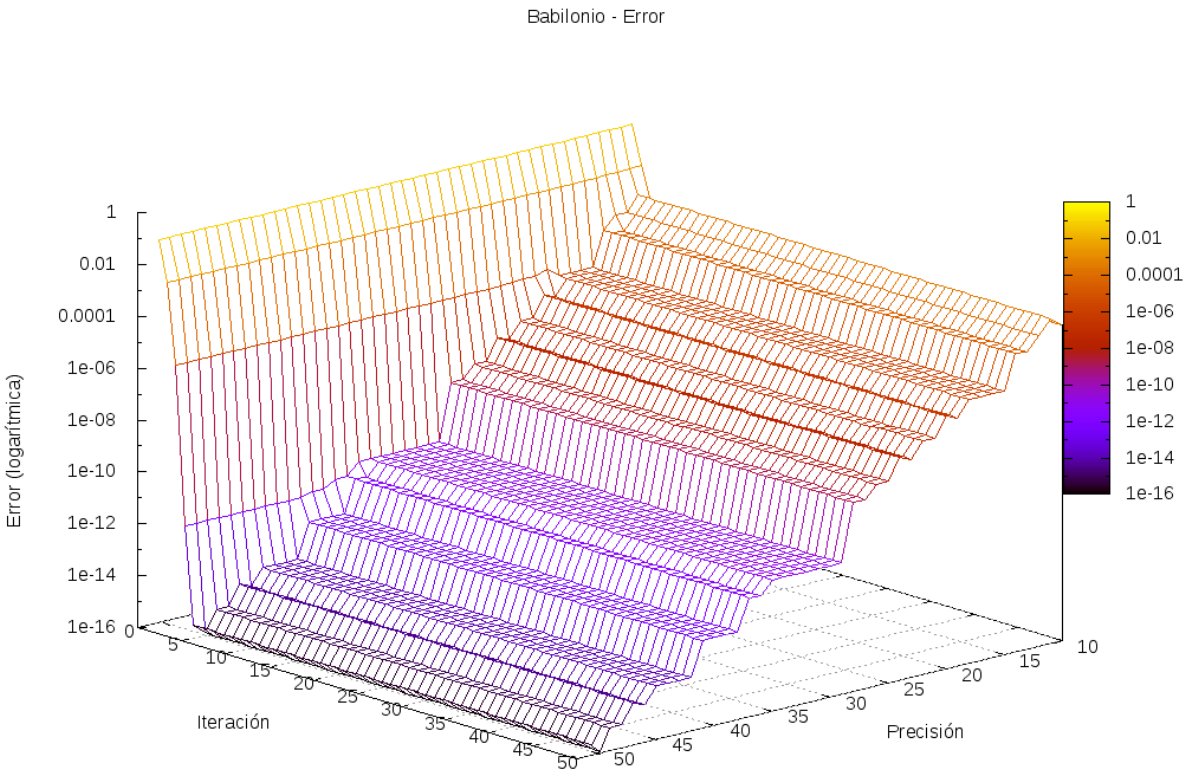

Figura 5: Método babilonio con redondeo

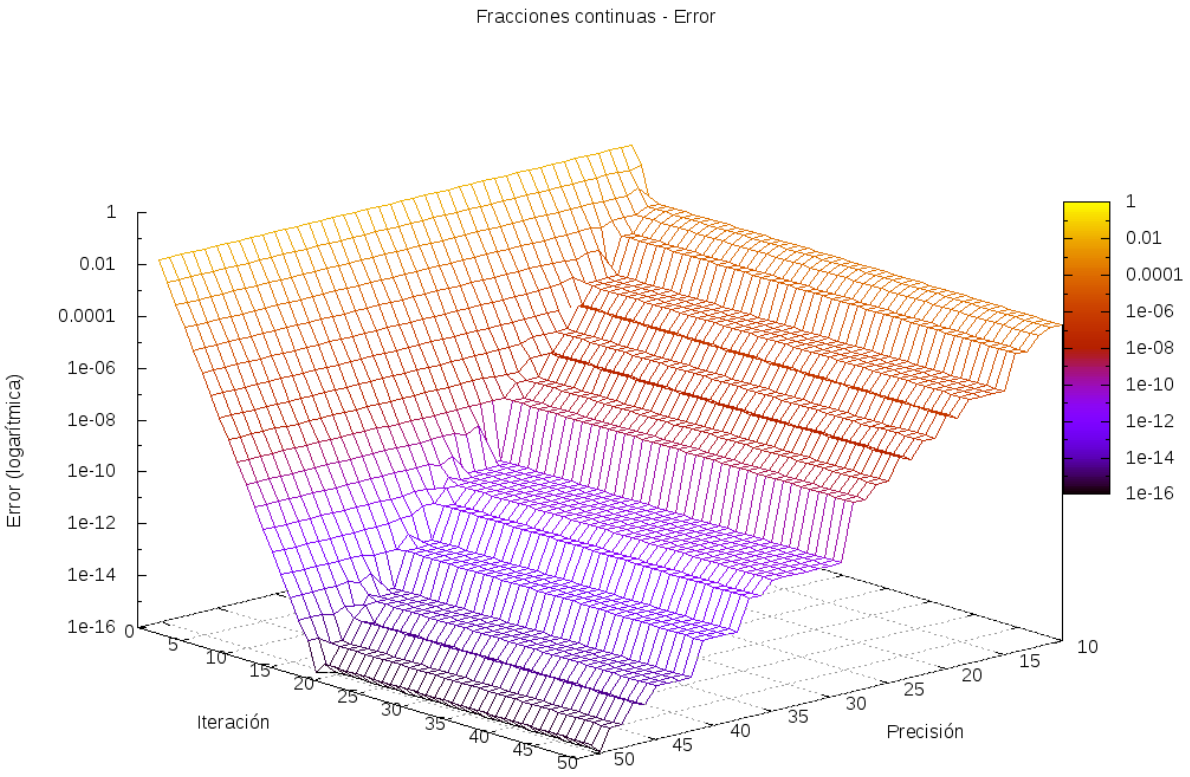

Figura 6: Método de fracciones continuas con redondeo

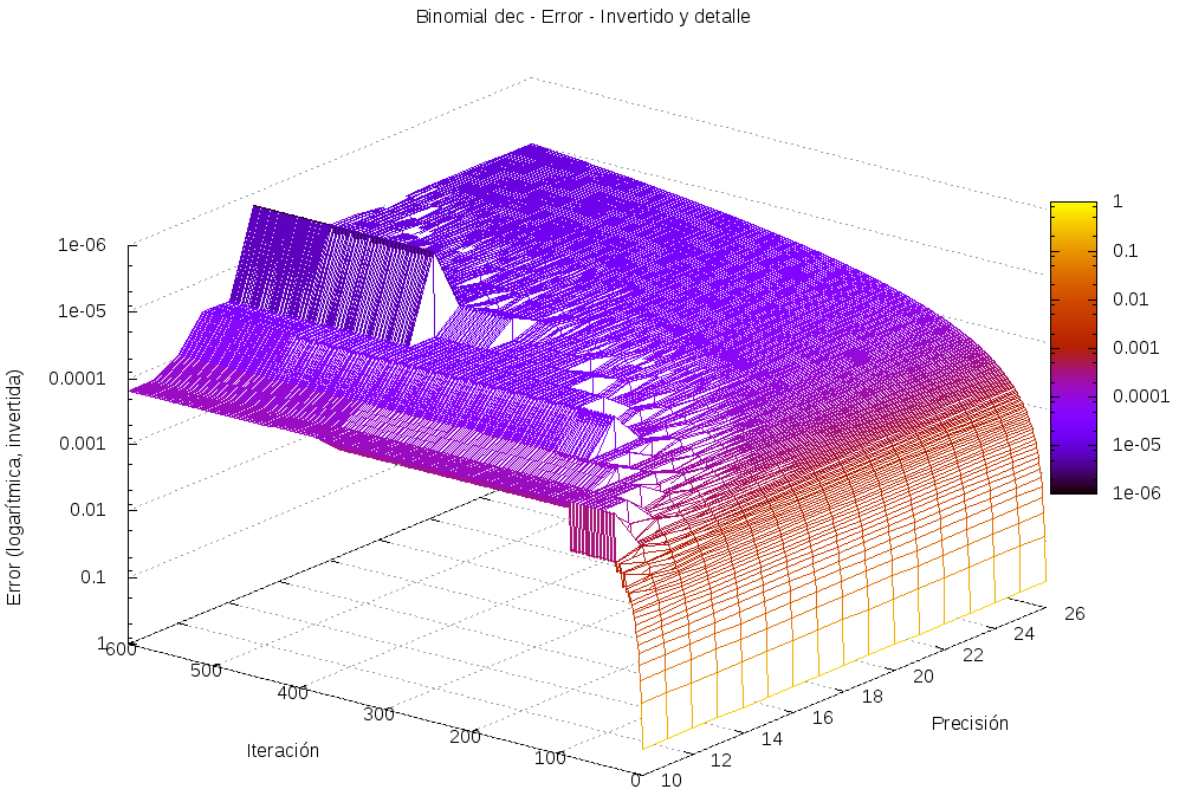

Figura 7: Método binomial decreciente con redondeo

3.2 Diferentes metodos de aproximación de la mantisa

3.3 Método Babilonio

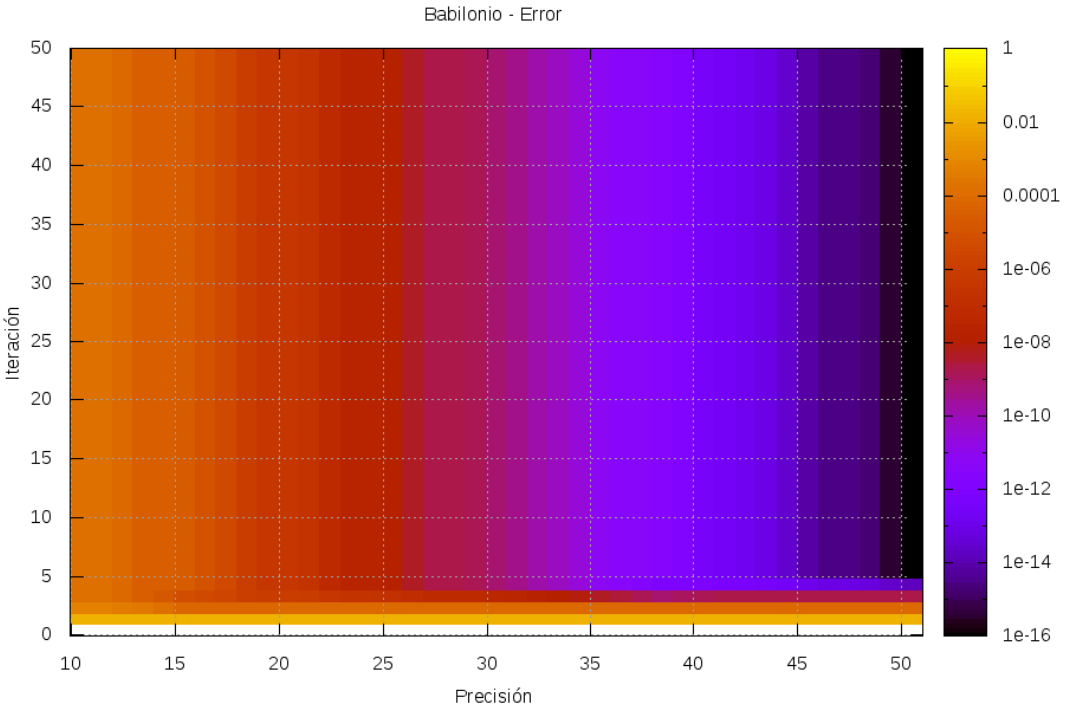

Figura 8: Error en el método babilonio, con distinta cantidad de iteraciones y distinta precisión. Presentación en forma de mapa de colores.

Este primer gráfico focalizado en el método babilonio podemos observar distintas caracteristicas interesantes del mismo, muchas de las cuales tambien se podran observar en gráficos subsecuentes.

Como primer resultado interesante podemos ver que a partir de la iteración 5, para *todas las precisiones* el error se estabiliza, lo cual se manifiesta como una linea vertical del mismo color.

Tambien podemos ir viendo como al incrementar la precision se va reduciendo el error, hasta llegar a un error del orden de $1e-16$ con precisión 51.

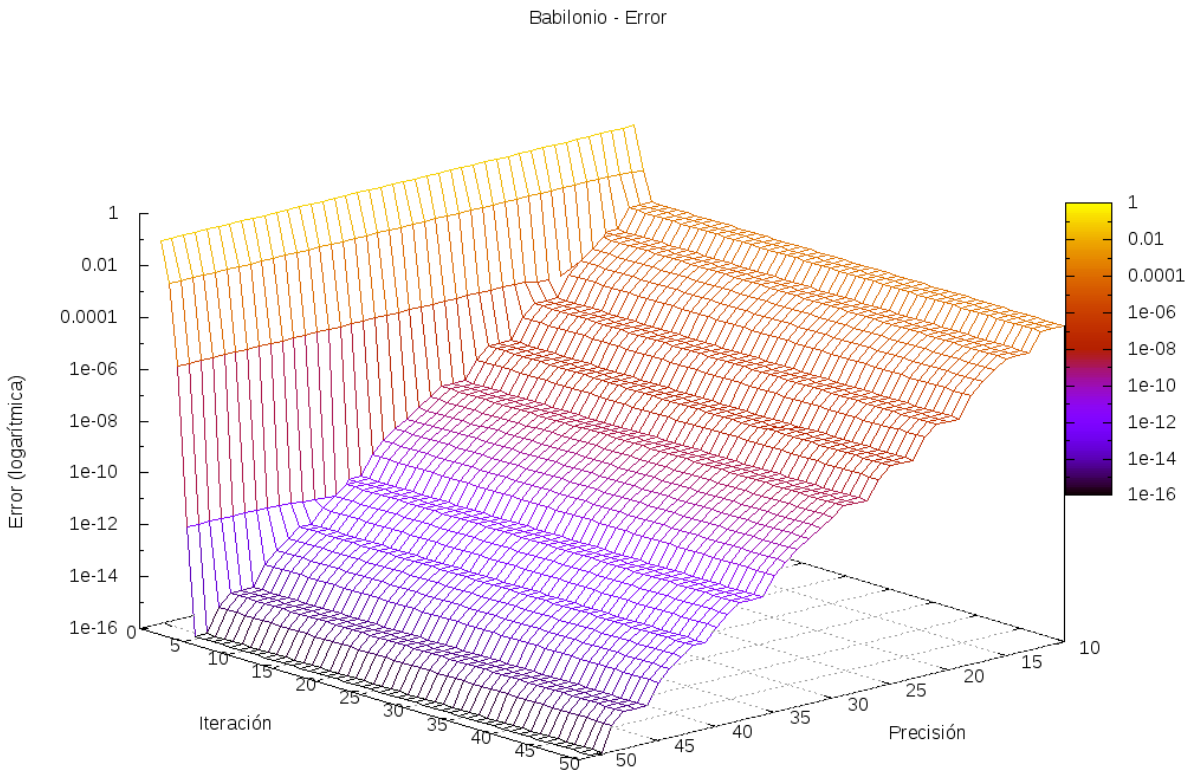

Figura 9: Error en el método babilonio, con distinta cantidad de iteraciones y distinta precisión (ídem figura 8). Presentación en forma de gráfico 3D coloreado en el eje z.

Esta es una representación alternativa de la figura anterior, en donde se grafica en el eje z el error del resultado obtenido en la precisión e iteración dadas.

Se pueden realizar las mismas observaciones, y es de principal interes la "pared" vertical en donde se aprecia la velocidad en la que el método se estabiliza, y como el resultado al que llega tiene un error estable conforme aumentan las iteraciónes, y dependiente de la precisión.

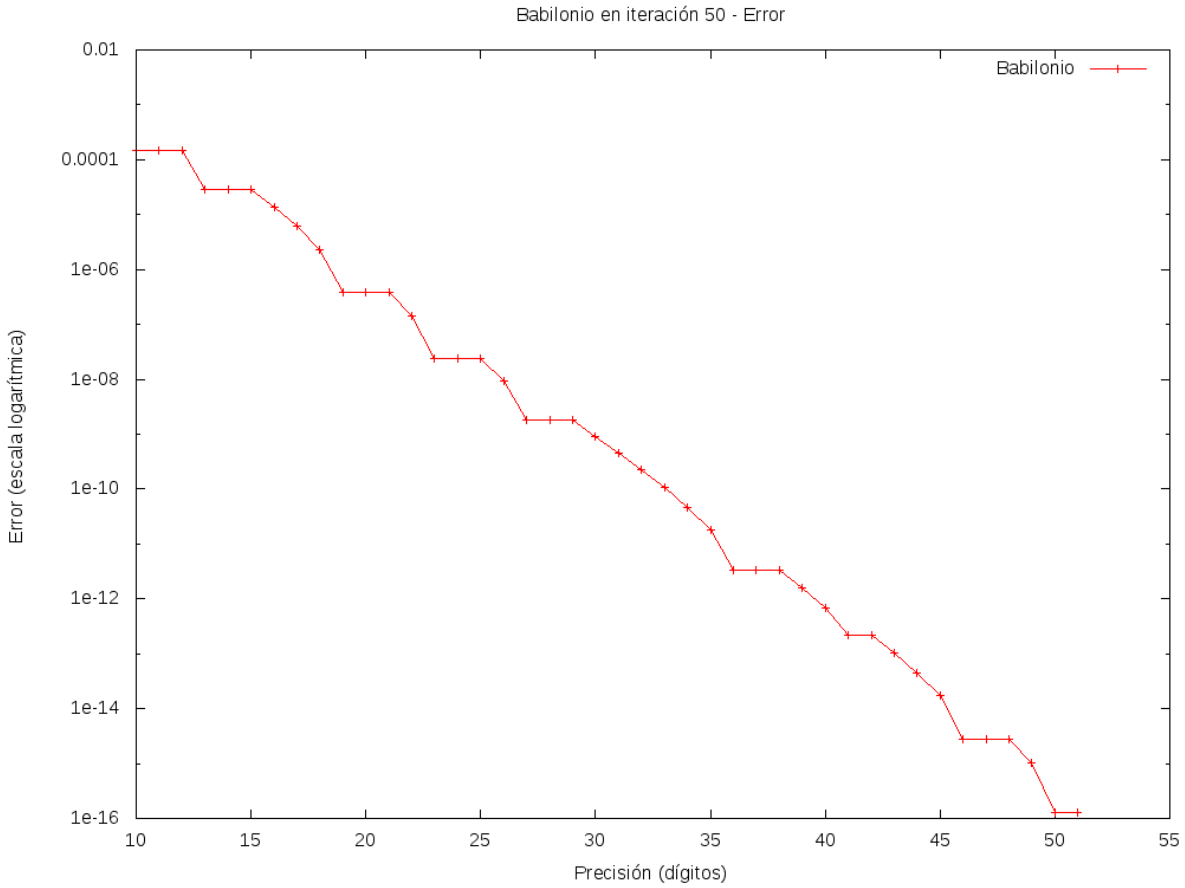

Figura 10: Error en los resultados del método babilonio en la iteración 100, variando la precisión

Este corte del gráfico anterior sirve para poder concentrarnos en como, en este método, el error va mejorando de forma exponencial (se aprecia lineal pero recordar que la escala del error es logarítmica) conforme aumentamos la precision. Se eligió la iteración 100 debido a que en ella el método ya se habia estabilizado en su mejor resultado, en todas las precisiones probadas.

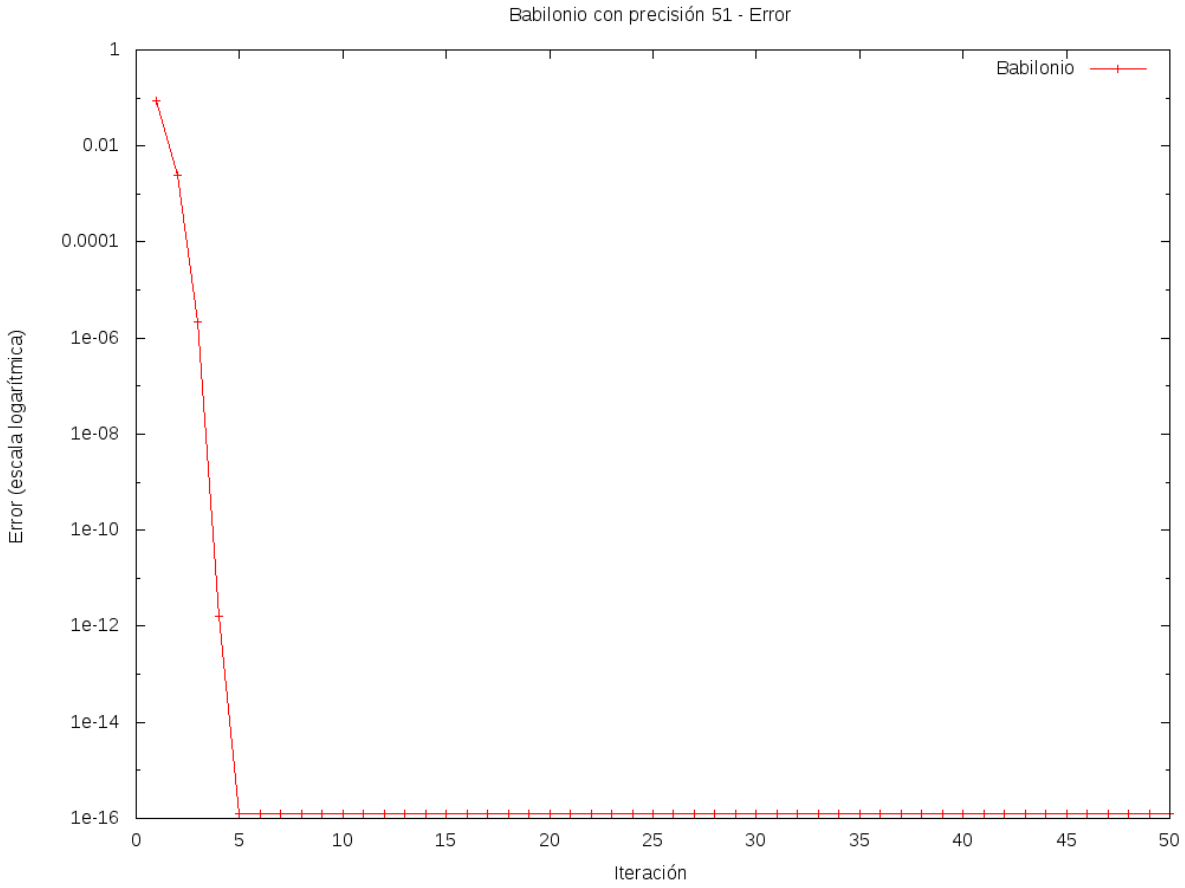

Figura 11: Detalle del error obtenido con precisión fija en 51 y variando la cantidad de iteraciones

En esta figura apreciamos en detalle como aun para la máxima precisión considerada se obtiene la mejor aproximación en la 5^{a} iteración, y permanece estable de ahi en adelante.

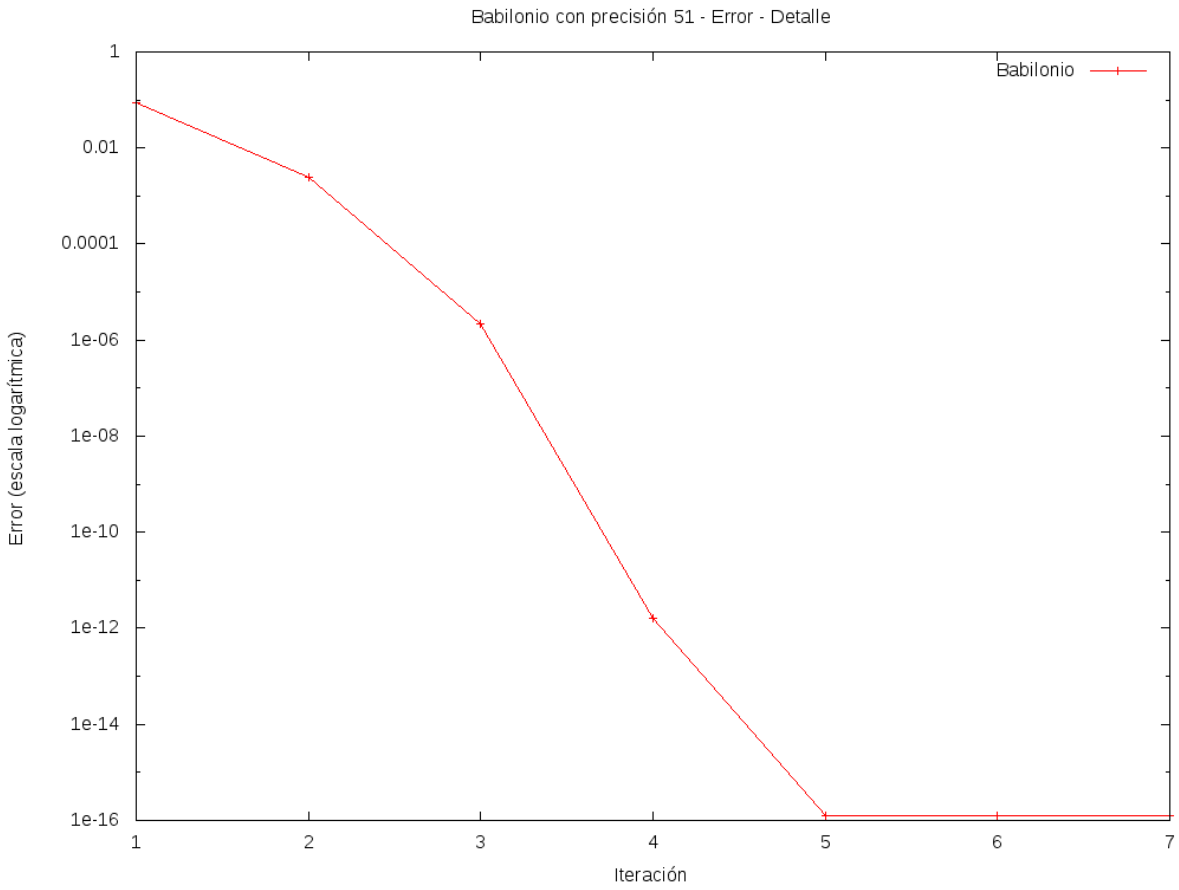

Figura 12: Detalle de la figura anterior, mostrando solo las primeras 7 iteraciones

Al ver solo las primeras 7 iteraciones, podemos apreciar mejor la forma en la que va reduciendo el error con cada iteración hasta llegar al mejor resultado.

3.4 Método Binomial

Para el análisis de resultados del método binomial, comenzaremos primero por unos gráficos que comparan los tres métodos implementados y que dan una idea general del comportamiento de los mismos asi como tambien de sus diferencias, para luego proceder a analizarlos en mas detalle de forma particular.

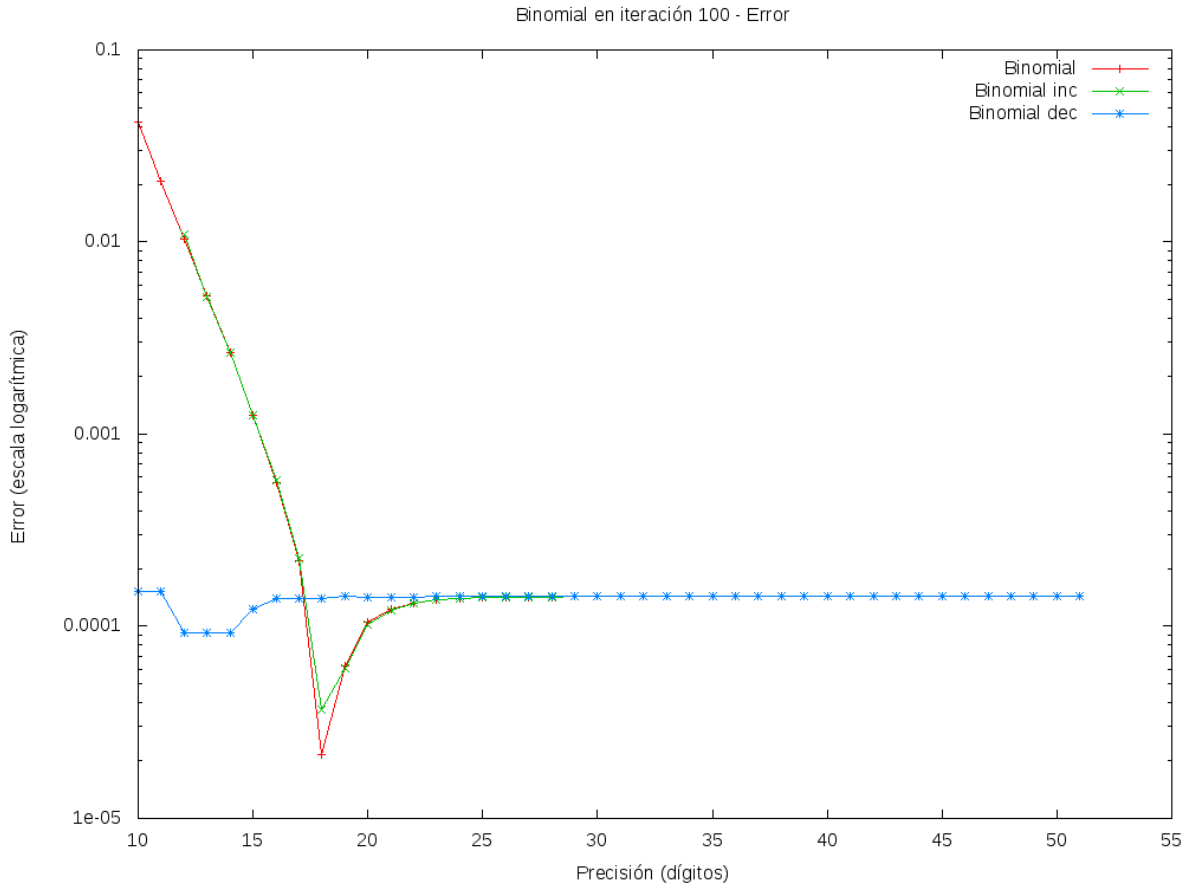

Figura 13: Error en los resultados obtenidos de los distintos métodos segun la precisión, en la iteración 100

En esta primer figura vemos como en la iteración 100 los distintos métodos van obteniendo distintos resultados con bastante distintos errores. Se aprecia claramente como los métodos convergen en precisiones altas, pero difieren en las bajas. Mas especificamente, el binomial decremental se comporta de forma notablemente distinta de los otros dos, que a su vez tienen un comportamiento similar.

Un resultado temprano interesante es que el menor error obtenido no se encuentra cuando la precisión es mayor, sino que se ubica entre la 15 y la 20 para los métodos simple e incremental, y 10 y 15 para la decremental. Esto marca una diferencia muy importante entre este método y los otros, que es que el incremento de precisión no redunda en un menor error. Mas adelante se analizará este fenómeno en detalle, con gráficos orientados a tal efecto.

Tambien vemos el principio de un resultado fuerte que utilizaremos mas ade-

lante, que es la fuerte similitud en el comportamiento de las implementaciones simple e incremental. Este, de hecho es uno de los pocos casos en donde es visible una pequeña diferencia entre ellos.

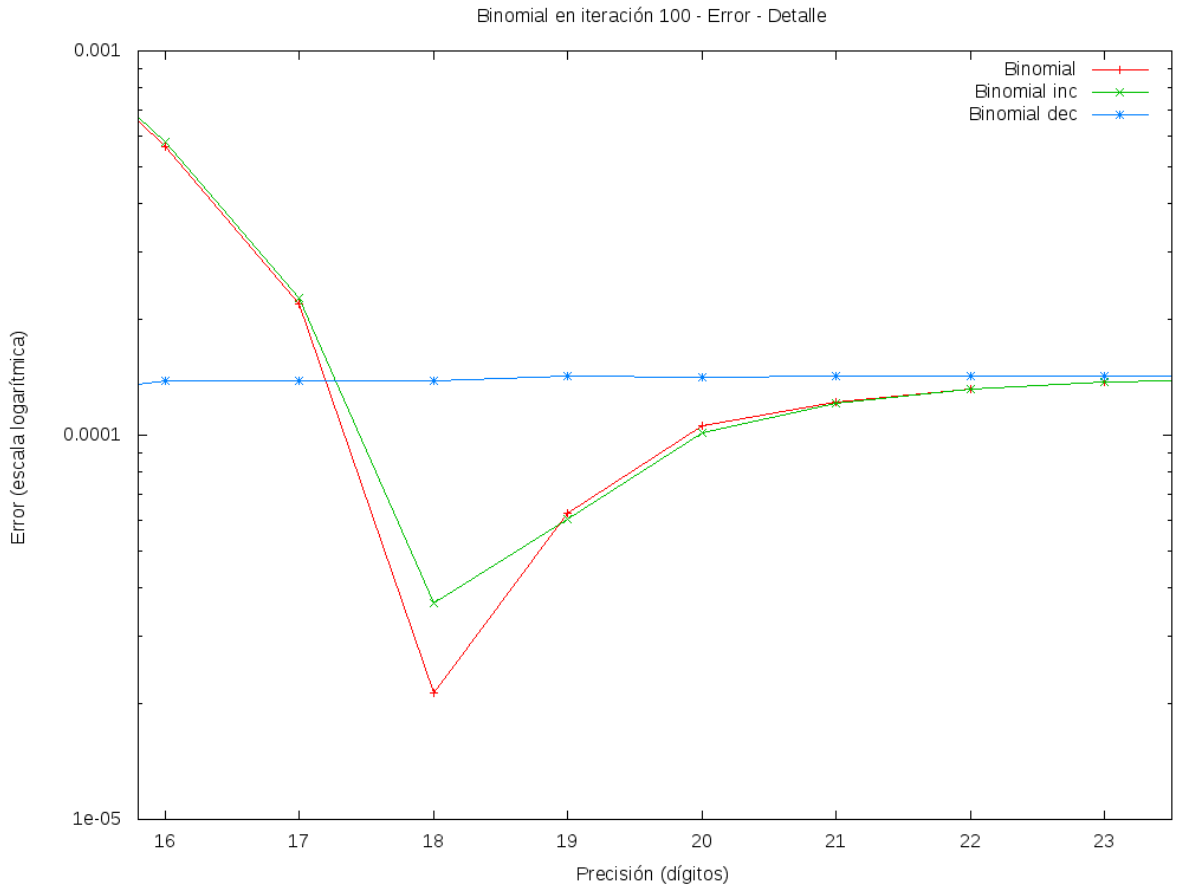

Figura 14: Detalle de la figura anterior, con precisión reducida

Este detalle del gráfico anterior permite concentrarnos en la diferencia entre las implementaciones simple e incremental, y en como se observa una ligera diferencia en su comportamiento en la precisión 18, pero velozmente vuelven a comportarse de forma similar.

Tambien se observa el efecto de que incrementar la precisión esta resultando en una peor aproximación, pasada la precisión 18, peculiaridad descripta anteriormente.

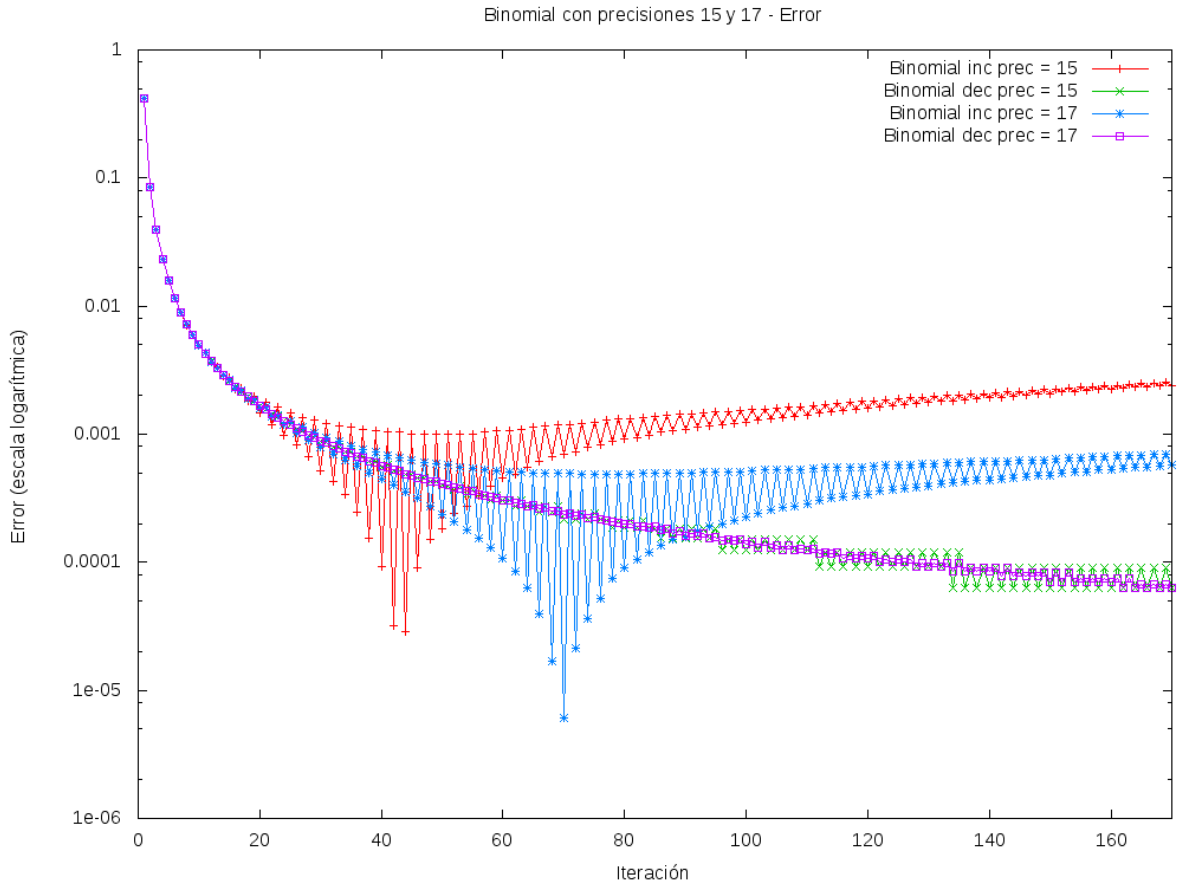

Figura 15: Error en los resultados de los métodos incremental y decremental para dos iteraciones dadas, variando la cantidad de iteraciones

Dejando fija la precisión, podemos observar como para ciertas precisiones el método incremental no mejora conforme se aumentan las iteraciones, sino que oscila, mejorando lentamente y aumentando la amplitud hasta llegar a un pico de mínimo error, luego del cual comienza a reducir la amplitud y, aun mas llamativamente, a empeorar.

En cambio, en la implementación decremental no se observa este fenómeno en este gráfico, aunque veremos que tambien tiene un comportamiento peculiar y no tan distinto mas adelante.

La implementación simple se omitió debido a que se comportaba de igual manera que la incremental, pero cargaba notablemente el gráfico, dificultando la presentación y legibilidad de los resultados, y por consiguiente su análisis.

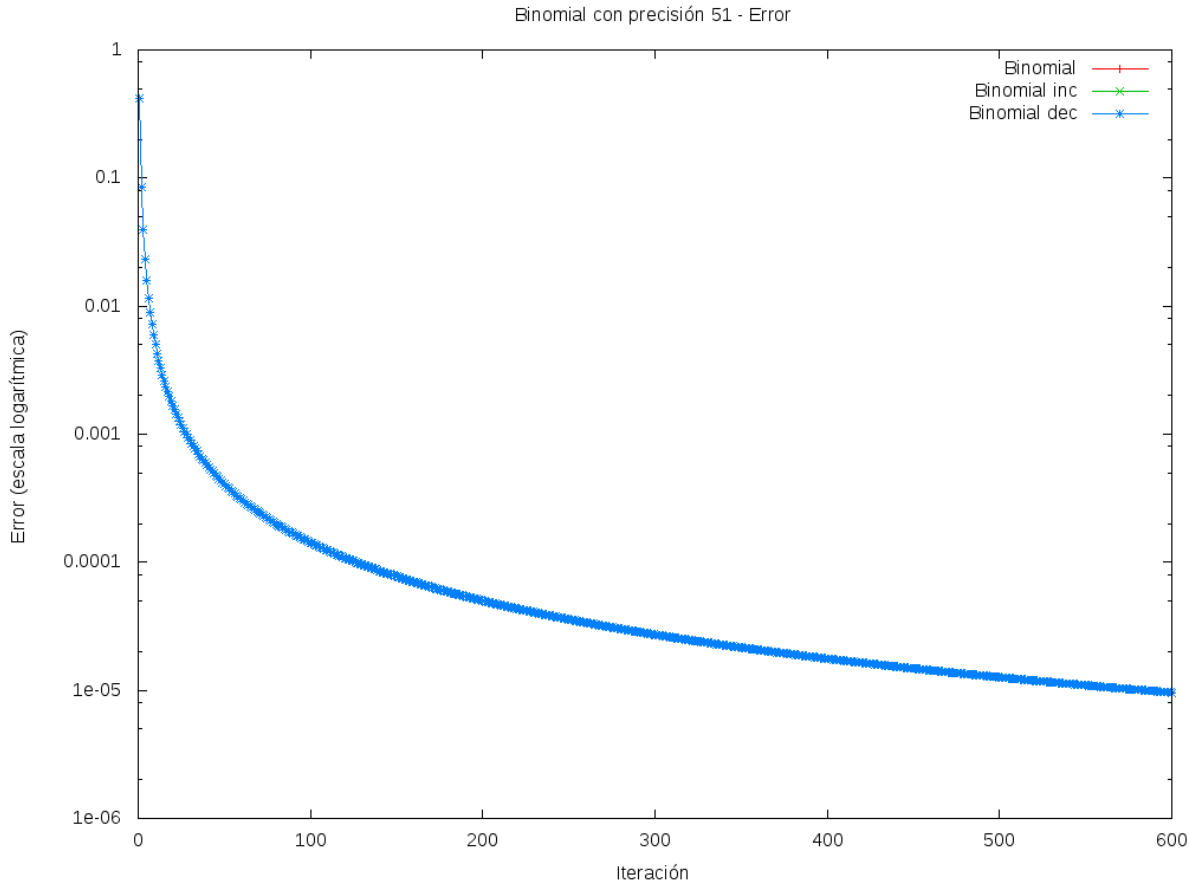

Figura 16: Comportamiento de las tres implementaciónes con precisión 51, variando la cantidad de iteraciones

Aqui vemos como todos los fenómenos de los otros gráficos no se manifiestan con la precisión 51, y como no parece haber diferencia apreciable en el comportamiento de los tres métodos. Mas adelante, al analizar las implementaciones en particular, se verá el patron de este comportamiento.

Notar que, como ya se menciono, las implementaciones simple e incremental se detienen en la iteración 171 a partir de la cual se obtiene un NaN. No se observa en este gráfico por un detalle de representación, dado que el gráfico de la implementación decremental oculta los otros.

3.4.1 Binomial simple e incremental

Dado que obtuvimos resultados prácticamente identicos en los resultados de las implementaciónes simple e incremental del método binomial, para realizar una presentación mas clara nos concentraremos solo en la implementación simple.

Salvo donde se nota lo contrario, los resultados aplican a ambas implementaciones.

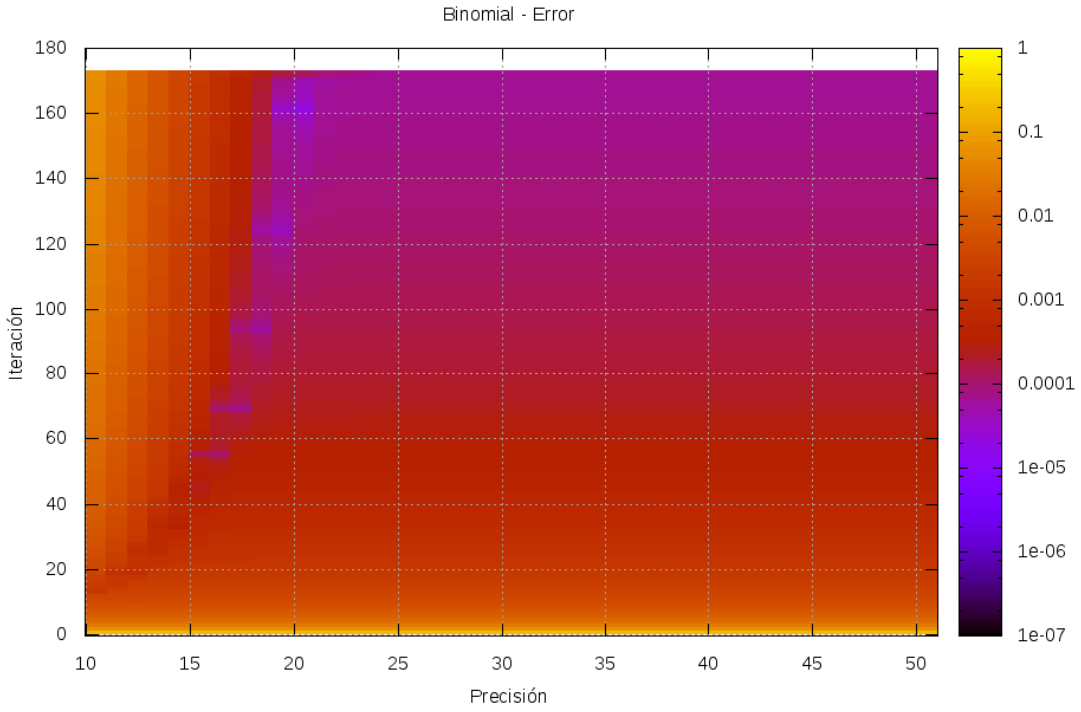

Figura 17: Error en la implementación binomial simple, con distinta cantidad de iteraciones y distinta precisión. Presentación en forma de mapa de colores.

Este gráfico muestra varias peculiaridades propias de esta implementación. Por un lado, se aprecia como no hay una clara mejora al aumentar la precisión, sino que mas bien parece estable (luego veremos en otros gráficos que en realidad empeora muy lentamente). Tambien se aprecia una curva creciente del lado izquierdo, en donde se obtienen picos de bajo error.

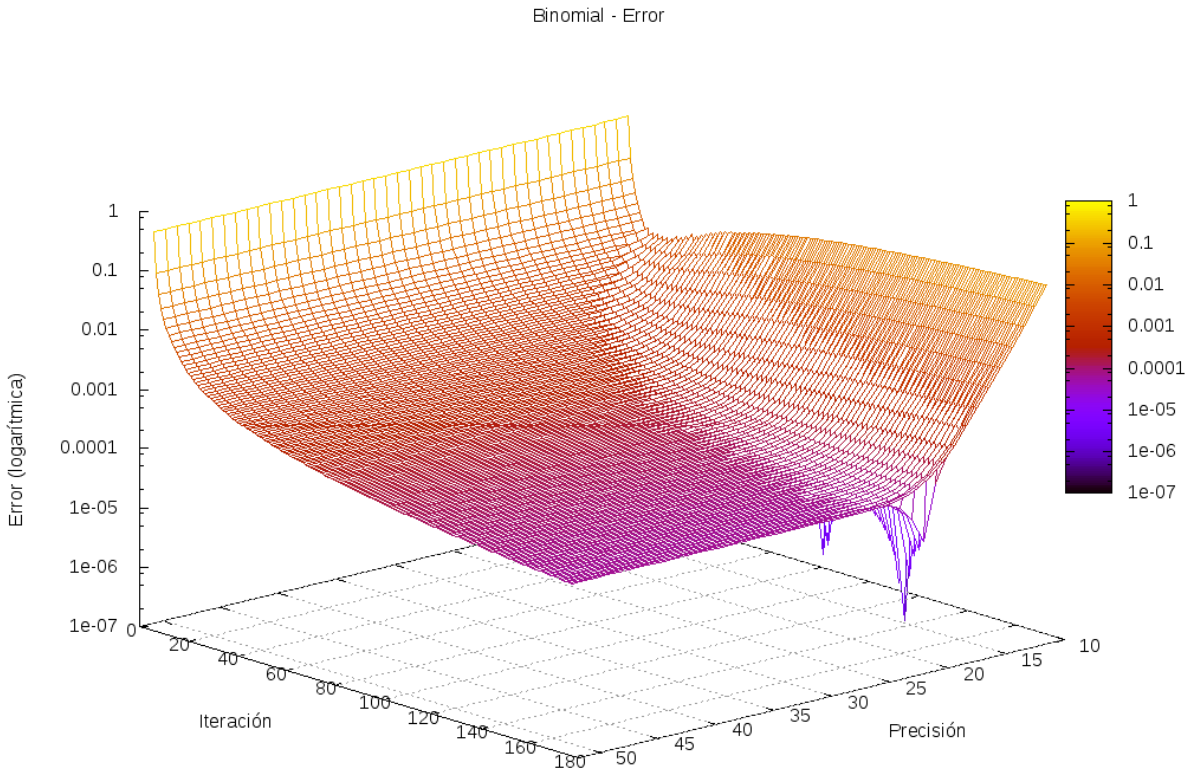

Figura 18: Error en la implementación binomial simple, con distinta cantidad de iteraciones y distinta precisión. Presentación en forma de gráfico 3D coloreado en el eje z.

Representación alternativa del gráfico anterior. En esta vista no se llegan a apreciar varios de los detalles mencionados previamente, debido a que los picos de bajo error quedan ocultos debajo del "manto" principal, solo se alcanza ver el perfil sobre el final del gráfico.

Si se puede observar el comportamiento general una vez pasados los picos, en donde se observa una superficie muy suave con una aparente tendencia a mejorar el error según aumentan las iteraciones.

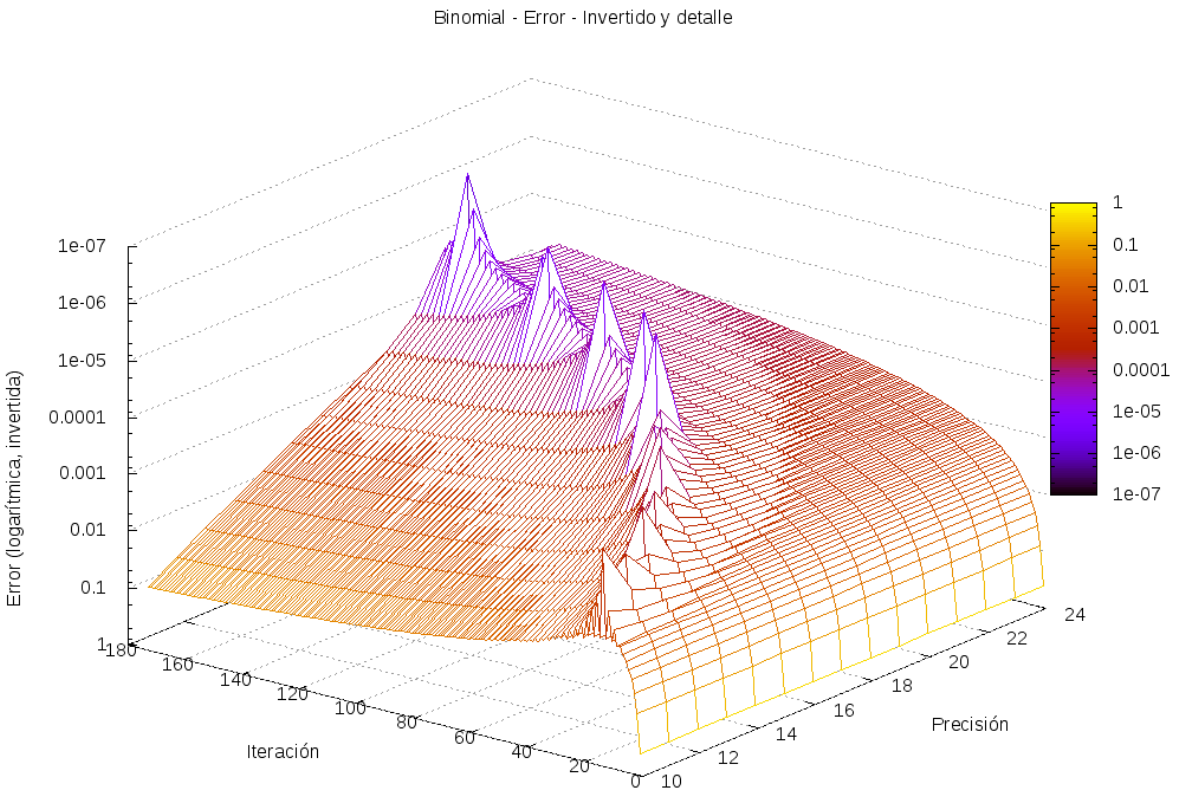

Figura 19: Detalle del gráfico anterior, en donde se lo muestra invertido (el mejor error se ubica mas alto sobre el eje z) y con un rango reducido en la precisión

Este gráfico nos permite apreciar en muy buen detalle el comportamiento general de los picos observados en el mapa y ocultos en su versión 3D. Como se ve, forman una "cadena" que crece de forma aparentemente logarítmica conforme aumentan las iteraciónes y la precision. La forma de cadena es interesante, ya que implica que no necesariamente al incrementar las iteraciones para una precisión dada, ni la precisión para una iteración dada, se obtiene un mejor resultado.

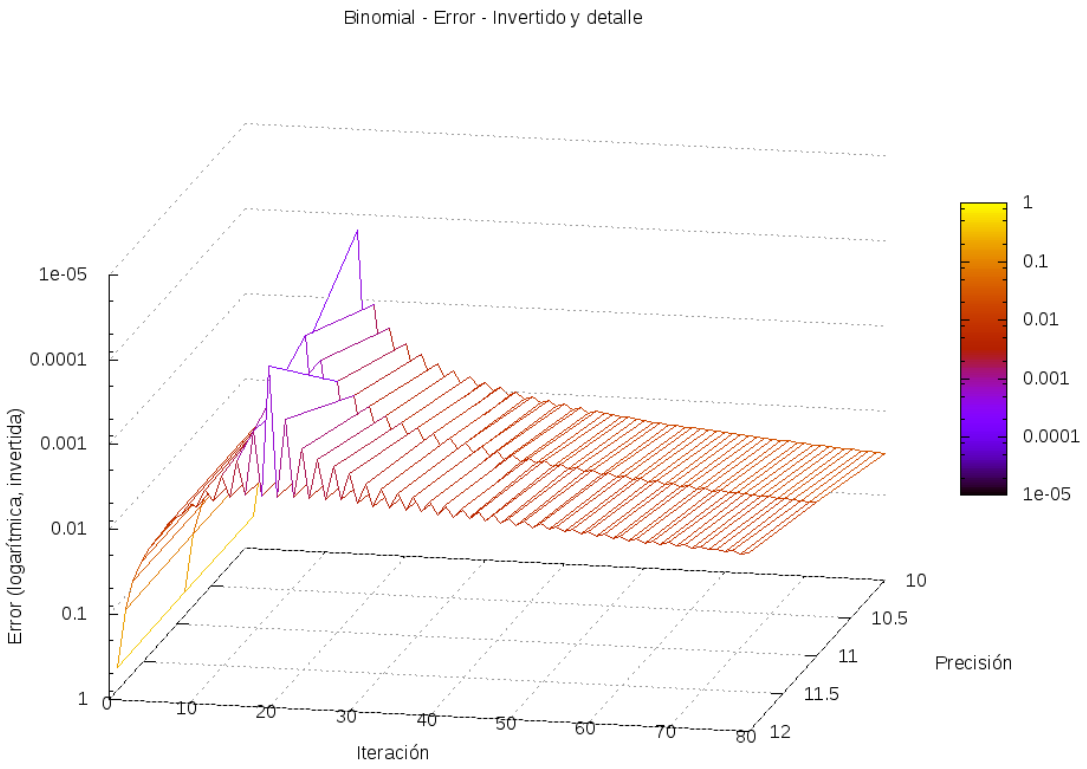

Figura 20: Detalle de uno de los picos

En este gráfico vemos un detalle de los picos, con un muy bajo rango de precisión para permitir apreciar los fenómenos anteriormente mencionados. También vemos que los picos tienen una relación con las oscilaciones, y como hay dos curvas claramente diferenciadas para las iteraciones pares e impares.

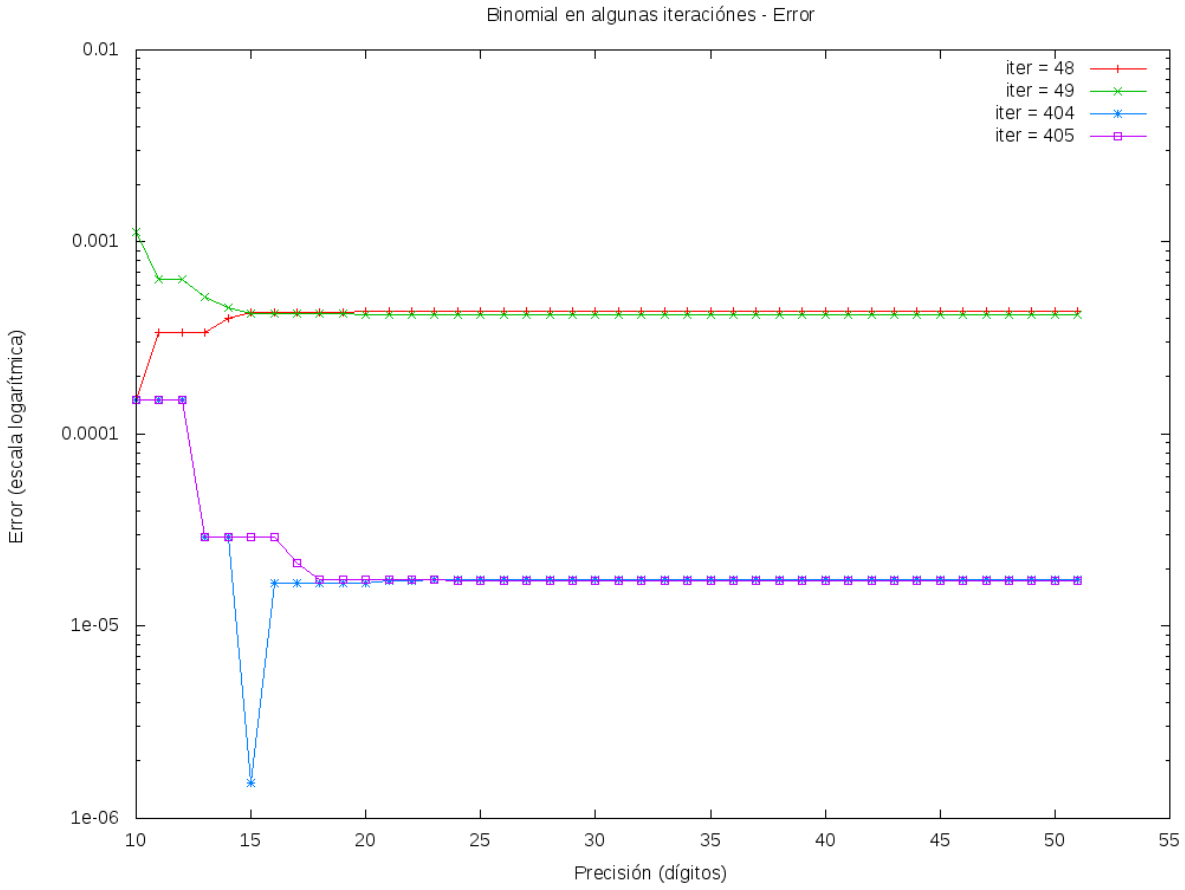

Figura 21: Error al varias la precisión, con iteraciónes seleccionadas

Este gráfico permite visualizar dos casos puntuales de los fenómenos mencionados, puntualmente que una iteración posterior no necesariamente obtiene un mejor resultado que una anterior, y que aún dentro de la misma iteración al aumentar la precisión puede empeorar el resultado obtenido.

3.4.2 Binomial decremental

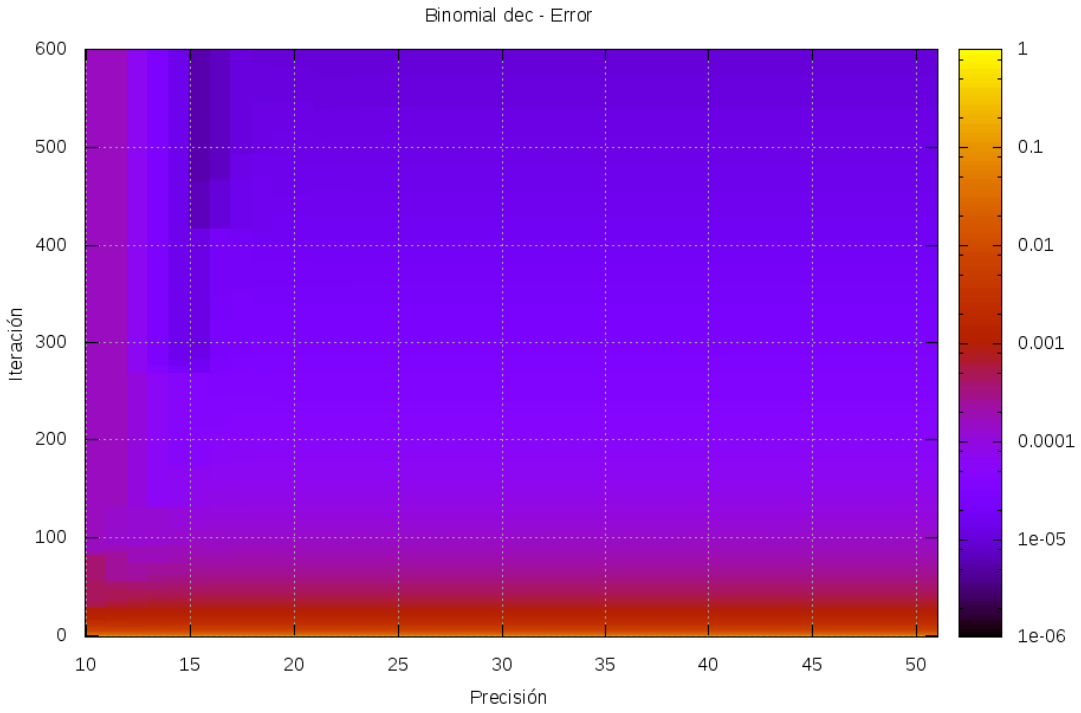

Figura 22: Error según varian la precisión y la cantidad de iteraciónes, en forma de mapa de colores

Este gráfico nos permite ver como el comportamiento de la implementación decremental es notablemente distinta que la simple e incremental, pero que reproduce los mismos fenómenos solo que a menor escala.

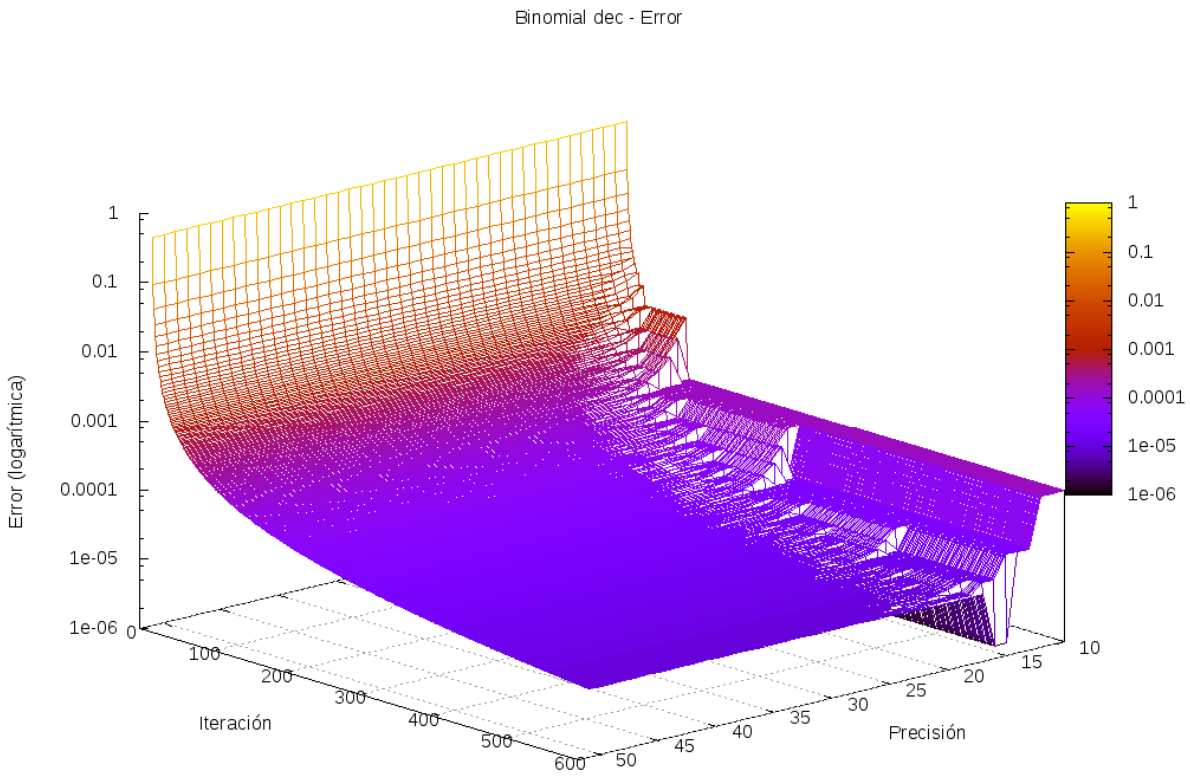

Figura 23: Error según varian la precisión y la cantidad de iteraciónes, en forma de gráfico 3D coloreado en el eje z.

Representación alternativa del mapa de colores de la figura anterior. Se aprecia como, a diferencia del método incremental, el "manto" es menos suave en bajas precisiones, y aparecen indicios de picos.

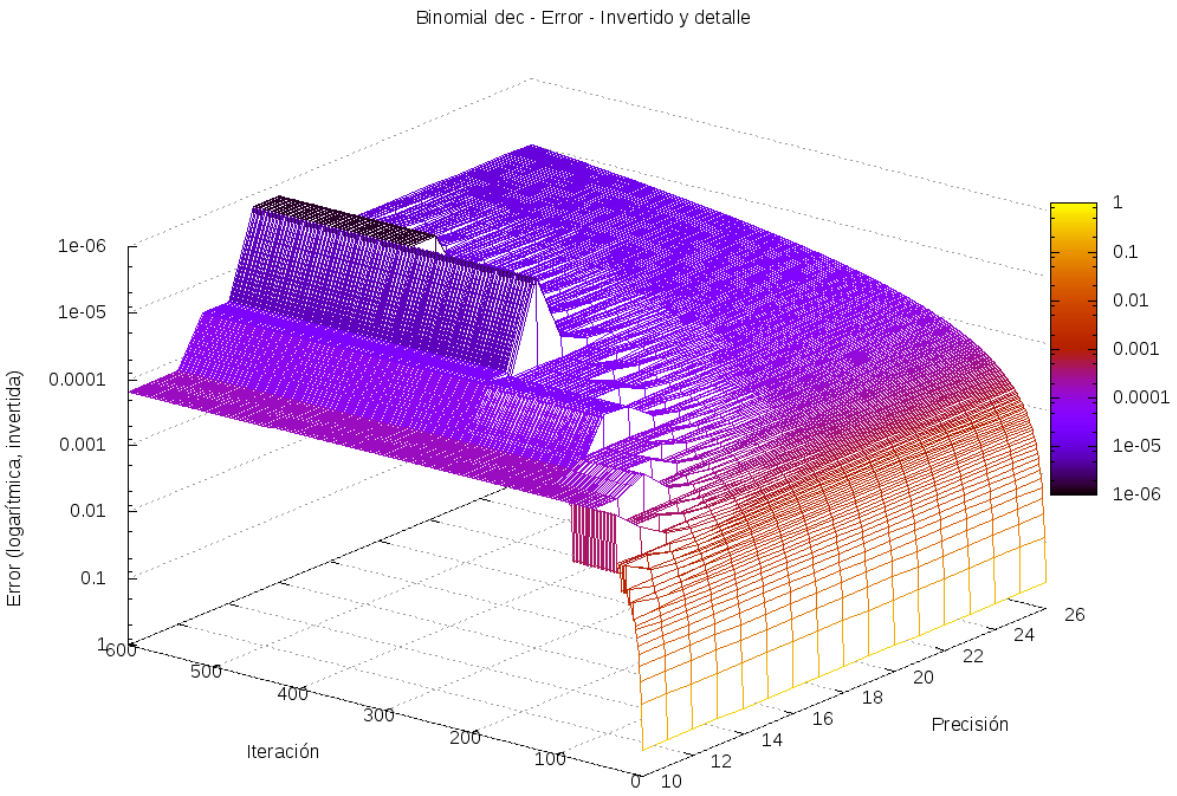

Figura 24: Visualización invertida de la figura anterior, con rango reducido de precisión

Esta visualización invertida nos permite concentrarnos en los picos, que como se ve aparecen crecientes, a diferencia de la naturaleza oscilante de los otros métodos. Sin embargo tambien se aprecia un crecimiento exponencial, aunque mas lento.

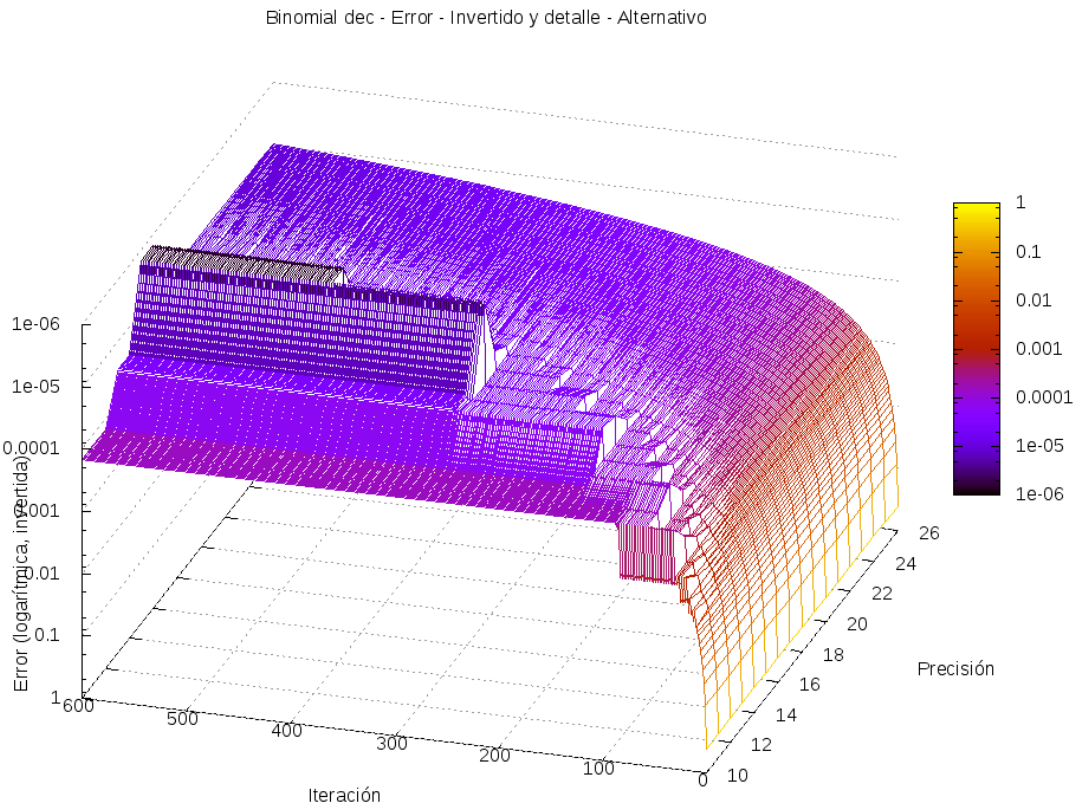

Figura 25: Binomial decreciente

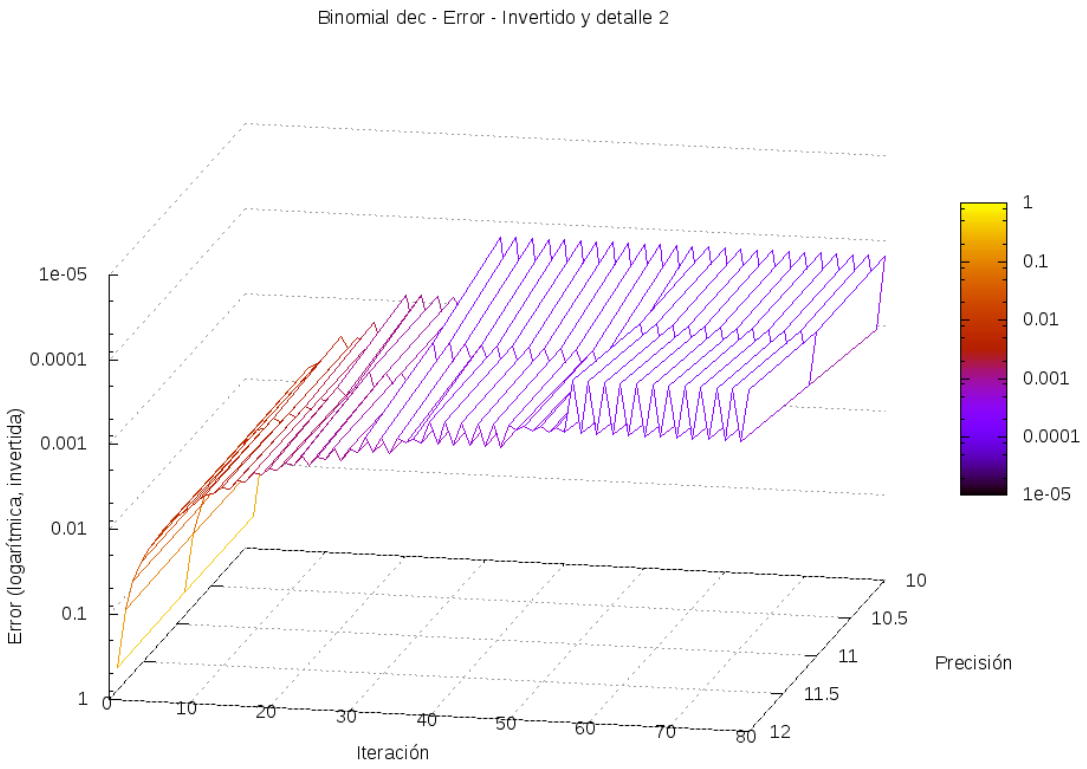

Figura 26: Binomial decreciente

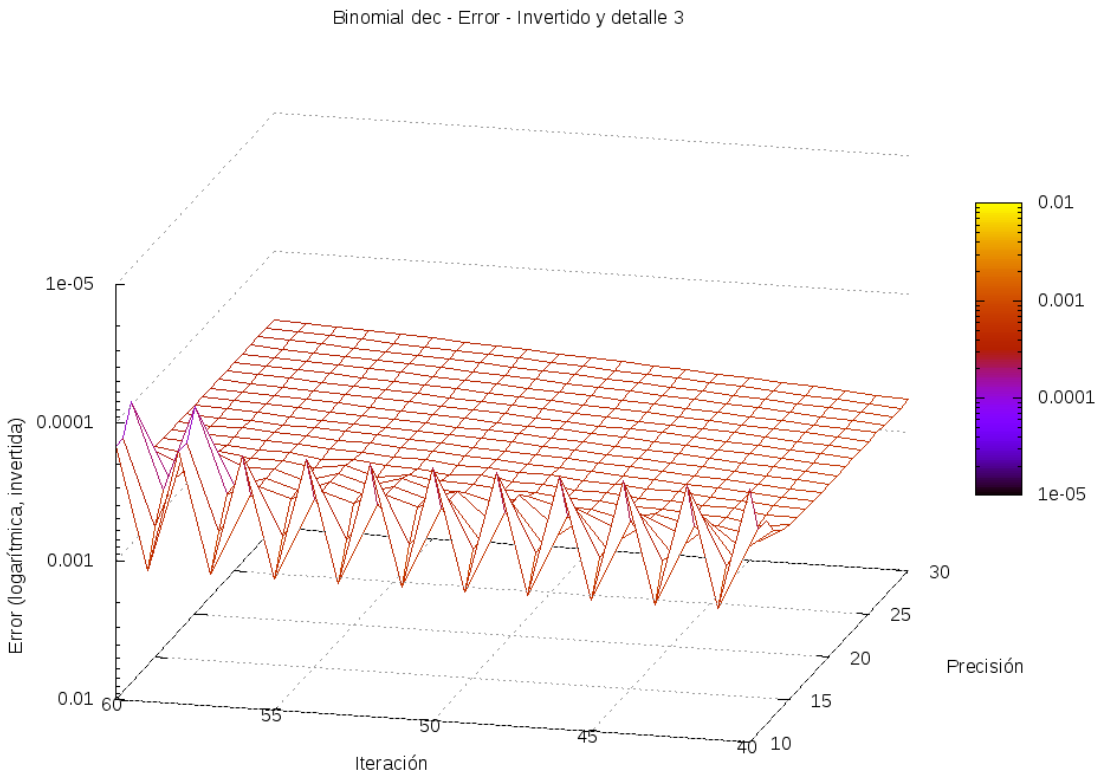

Figura 27: Binomial decreciente

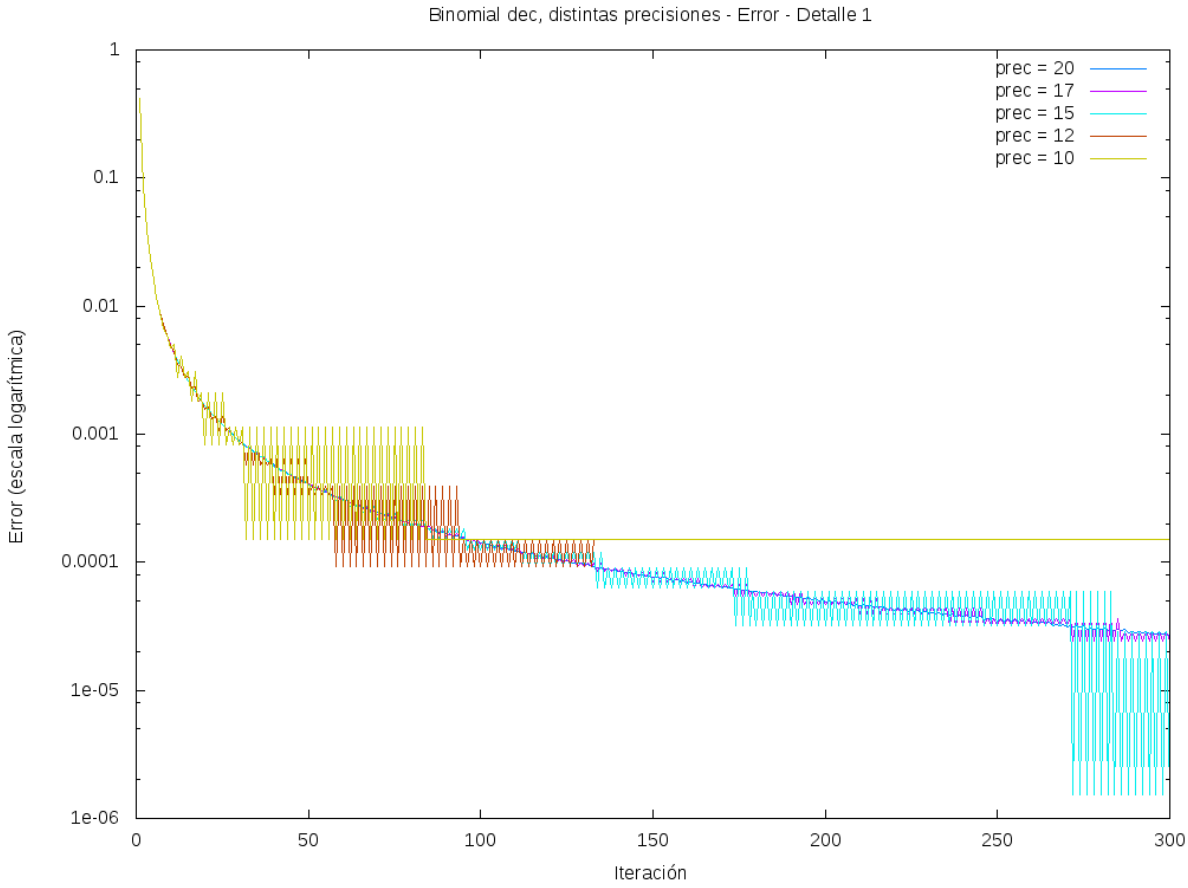

Figura 28: Binomial decreciente

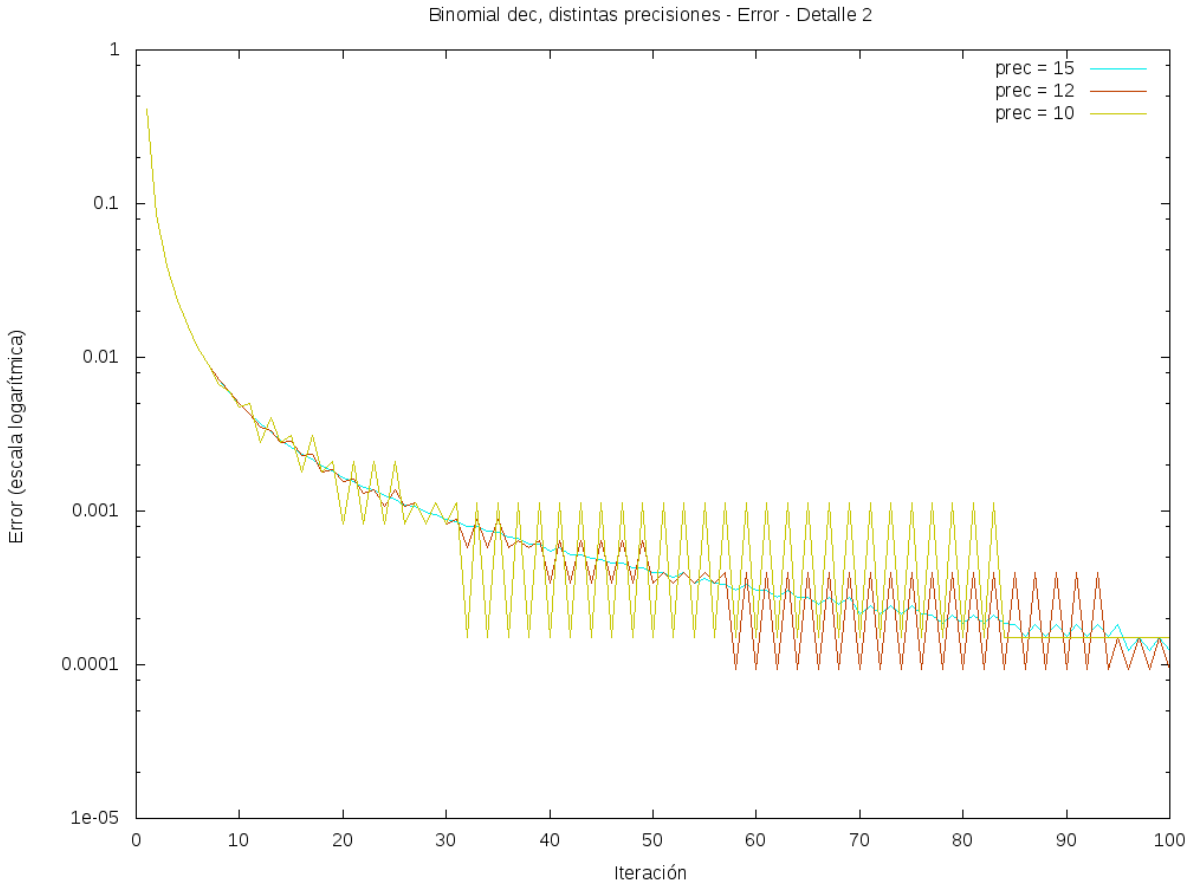

Figura 29: Binomial decreciente

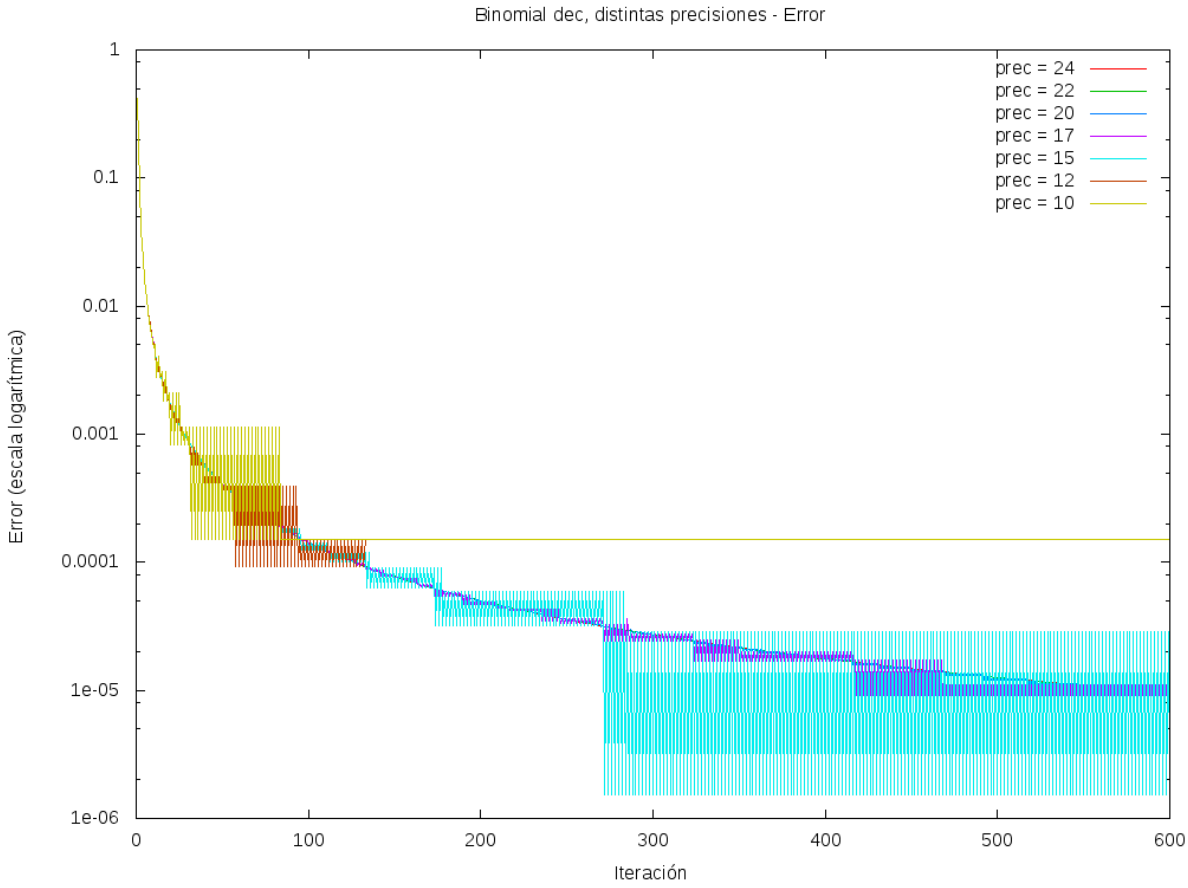

Figura 30: dsa

3.5 Método de fracciones contínuas

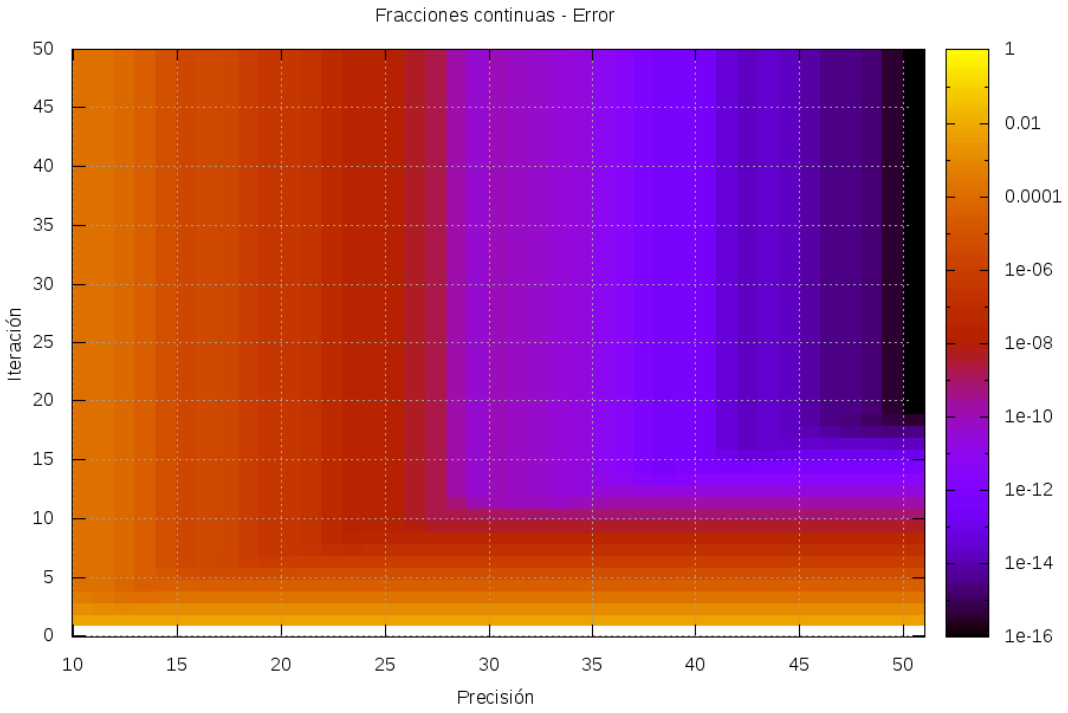

Figura 31: Fracciones Continuas

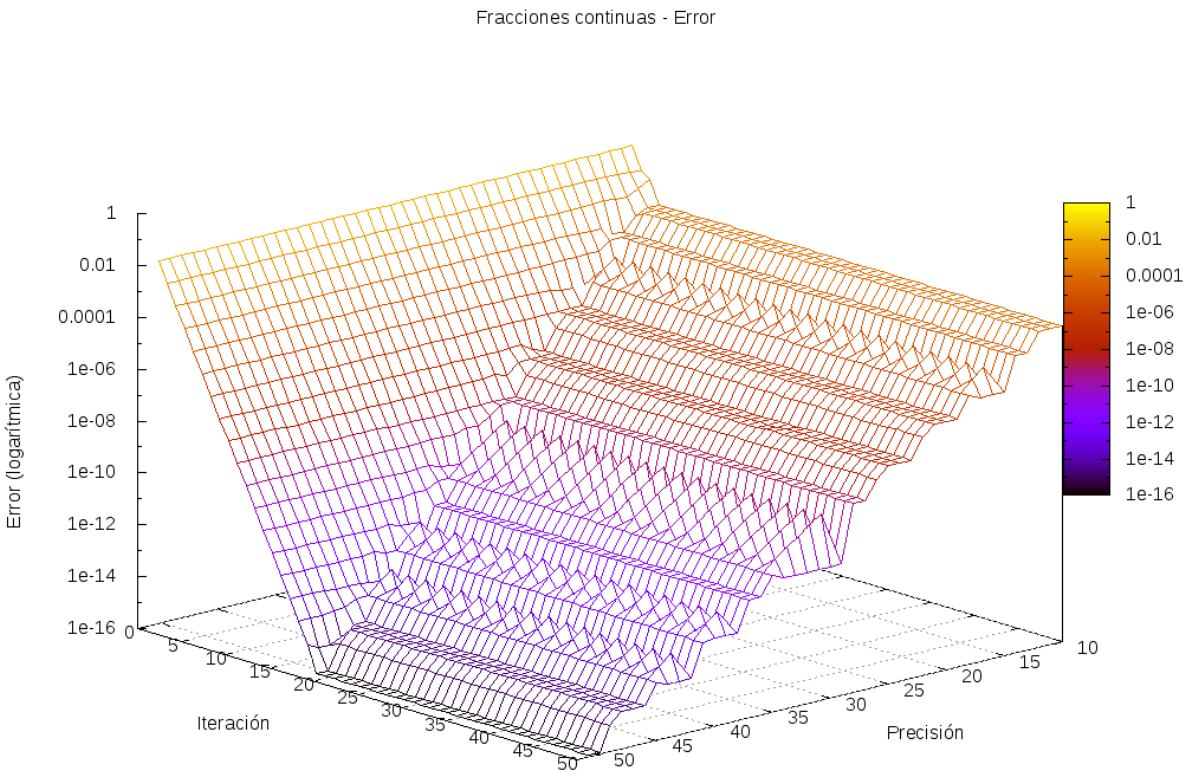

Figura 32: Fracciones Continuas

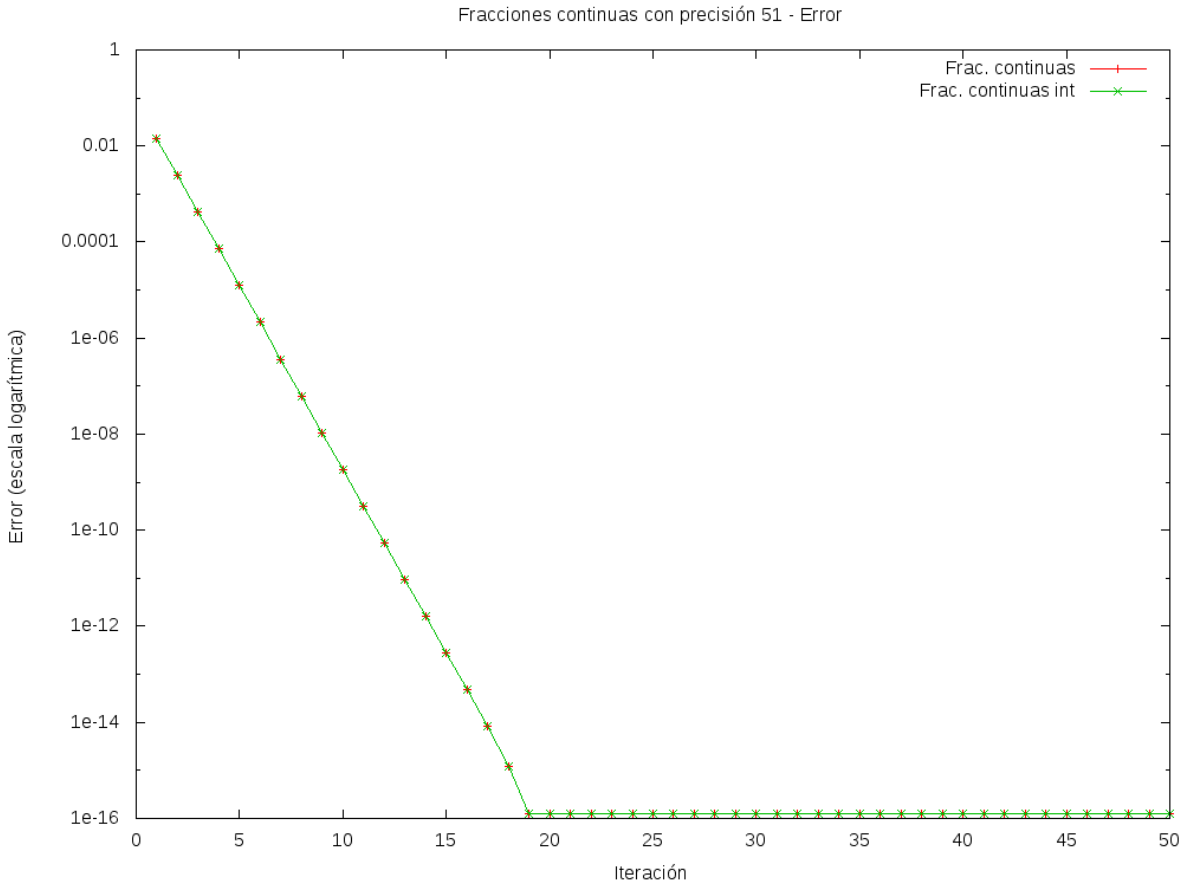

Figura 33: Fracciones Continuas

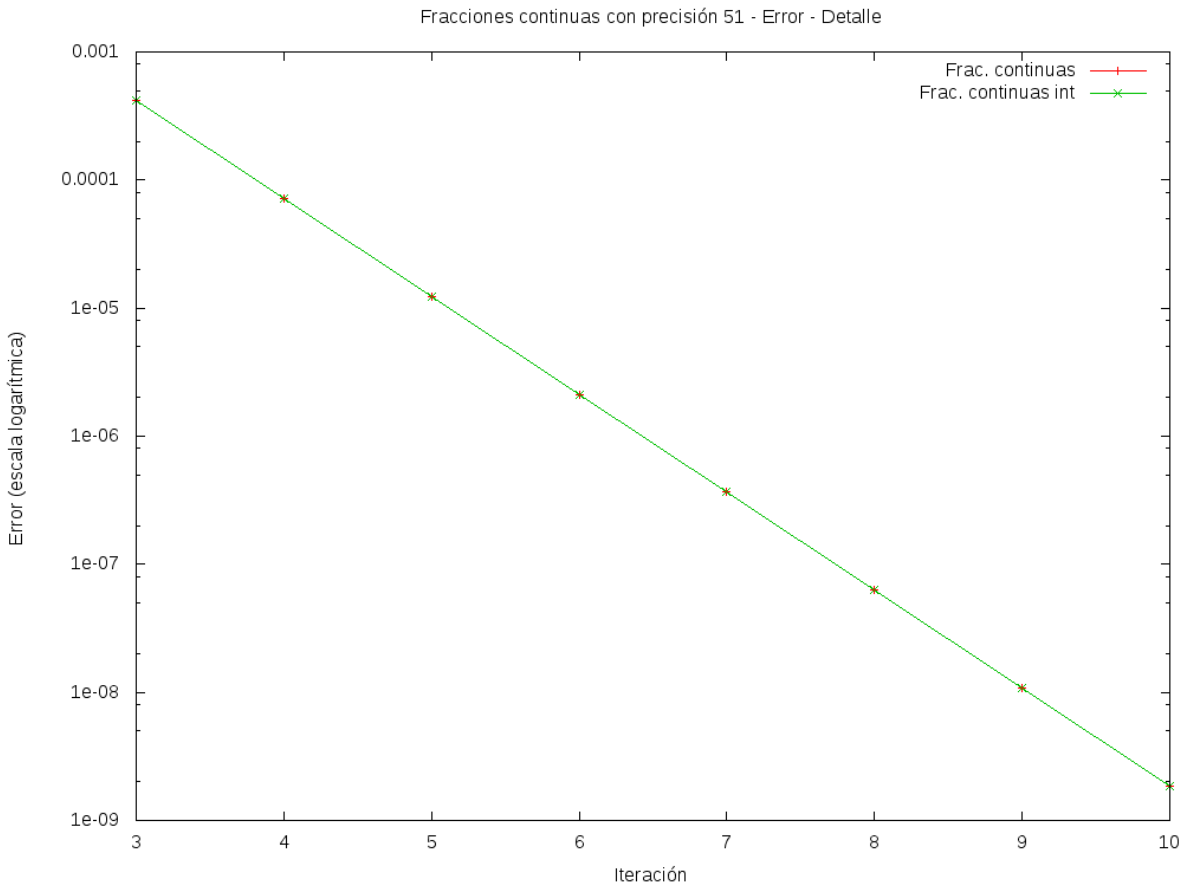

Figura 34: Fracciones Continuas

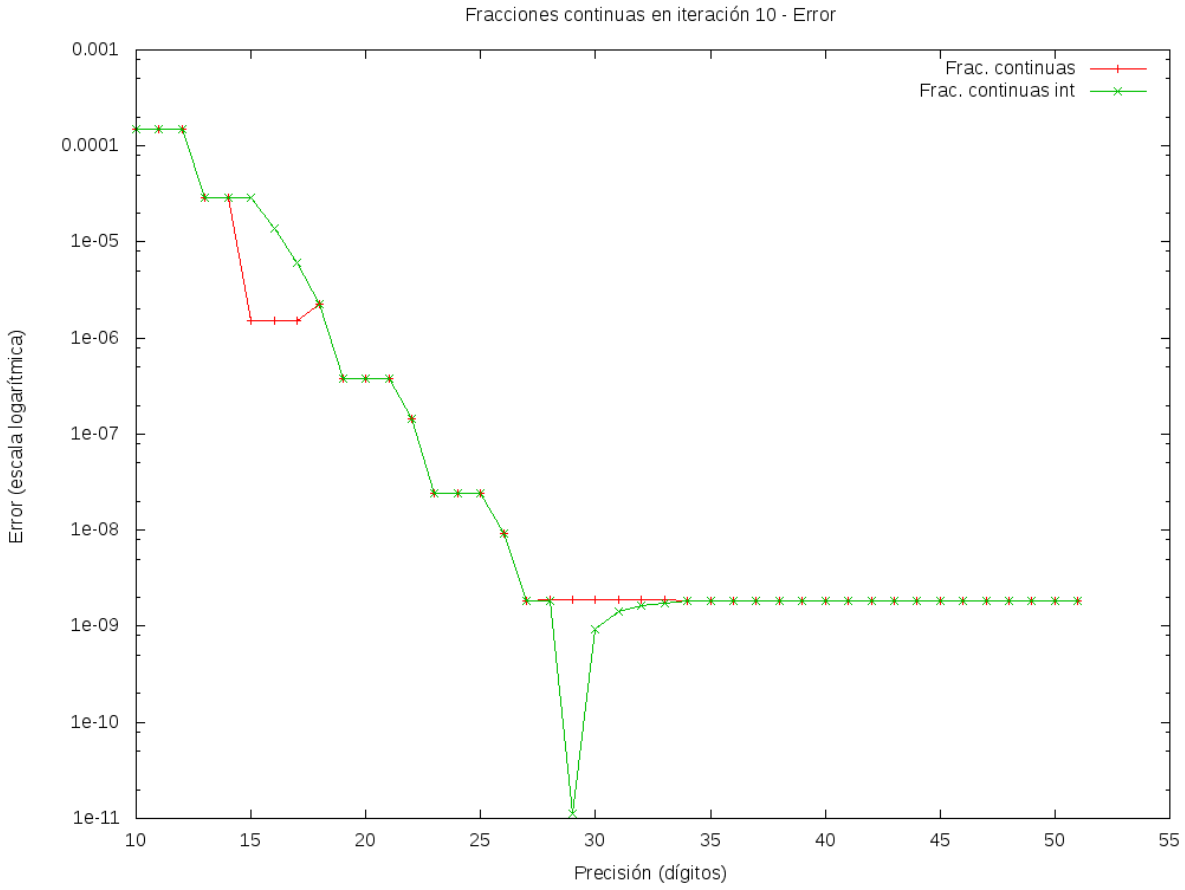

Figura 35: Fracciones Continuas

4 Discusión

Luego de obtener los resultados presentados en la sección anterior, se pueden obsevar distintos aspectos tanto de los algoritmos como de los métodos. Los mismos serán discutidos a continuación.

4.1 Métodos de aproximación de la mantisa

Se pudo observar a partir de las pruebas empíricas como cambia el resultado al cambiar el algoritmo que ajusta la precisión de los numeros, tanto al usar truncamiento como al usar redondeo.

Se puede observar que el comportamiento en sí de los algoritmos no se ve modificado al cambiar los métodos. Es decir, los resultados en sí difieren al usar un método u otro, pero la tendencia de los mismos no se ve afectada. Esto se puede ver en los graficos. Al compararlos podemos ver las distintas perturbaciones al usar un método o el otro, pero ambos decaen en manera similar al aumentar la cantidad de términos usados. También la cantidad de iteraciones hasta lograr que el algoritmo se estabilice en un resultado resulta del mismo orden.

Comparando el método babilonio con redondeo y el método babilonio con truncamiento, se puede ver que para ciertas precisiones y usando redondeo se llega a un valor más preciso que para la misma precisión usando truncamiento. Además se puede observar que, para ciertas precisiones, al utilizar el método de truncamiento en vez del de redondeo, el algoritmo comete un error mayor. En este caso, dicho algoritmo no mejora ni subsana dicho error con más iteraciones. Lo que hace que en estos casos se aproxime con más error.

Si comparamos también el método de fracciones continuas simple usando redondeo con él mismo usando truncamiento, se puede observar un comportamiento similar a lo que ocurre con el babilonio. Se puede ver que usando redondeo el error se estabiliza en ciertos entornos con más de 20 iteraciones. En cambio, al usar truncamiento, en los mismos lugares donde se estabiliza con redondeo, se ve una fluctuación en el error acotada inferiormente por donde se estabiliza con redondeo.

Al comparar el método binomial decreciente usando truncamiento con su variante usando redondeo, se puede ver como estos errores son menores.

4.2 Método Babilonio

Este método es el que mejores resultados obtiene de los tres propuestos. El mismo encuentra el mejor resultado para una presicion t dada ($t < 52$) al termino de 5 iteraciones. Se puede observar en los gráficos como el error decae abruptamente hasta estabilizarse en dicho valor.

Es decir, al aumentar la cantidad de iteraciones no siempre mejora la aproximación.

4.3 Método Binomial

Este método resultó ser el más interesante para analizar. El mismo no logra aproximar $\sqrt{2}$ con menor error que el resto de los métodos, pero es el que más interesante nos resultó a nivel numérico.

A partir del análisis de los gráficos se puede observar como se diferencia el comportamiento del método al implementarlo de manera creciente o decreciente. Al comenzar la sumatoria con un 1, los errores que comete el algoritmo son más notables. En cambio, cuando se comienza la sumatoria desde el término menos significativo, el error no es tan notable. Esto se debe a que al ser $\sqrt{2}$ un numero entre 1 y 2. Cuando se comienza con un uno, se obliga a que el exponente, del número que acumula el resultado, quede fijo. Haciendo imposible que los números más pequeños tengan influencia en él. En cambio, cuando se comienza a sumar desde los números más pequeños, el exponente se establece de forma tal de adecuarse al orden de magnitud de los sumandos, haciendo que todos los números sean considerados en el resultado. Esto se traduce en un error menor en los experimentos.

También, es importante notar que al se posible expresar la fórmula como una sumatoria, a partir de cierta iteración (dependiente de la precisión) los sumandos son despreciables para la precisión dada. A partir de este punto no tiene sentido continuar realizando más iteraciones.

Otro aspecto interesante es ver como, para una cantidad de iteraciones dada, a partir de cierta precisión (dependiente de la cantidad de iteraciones) el resultado final no varía. Por ejemplo, en el gráfico <ref>{bin-err-i:100.png} se puede observar que para 100 iteraciones, a partir de 25 bits de precisión el resultado se mantiene. Se supone que esto se debe a que los términos de la sumatoria van decreciendo hasta que no pueden ser representados con la cota de 52bits que se tomó.

También se puede observar en los gráficos la presencia de ciertos "picos". Suponemos que éstos son errores numéricos. En particular, se puede ver el pico donde el error es menor a 10^{-6}, esto no significa un mejor cálculo, sino que es producto del error que se comete.

4.4 Método de fracciones continuas

Luego de implementar este método de la manera más simple, se trató de mejorar dicha implementación como se explicó en la seccion de desarrollo. Esta segunda implementación resltó ser peor que la inicial, ya que no mostró un resultado superior y al alcanzar cierta cantidad de iteraciones, los números que maneja el algoritmo son tan grandes que quedan por afuera del rango representable. Por esta razón, las pruebas anteriores se realizaron con la implementacion inicial.

Este método logra establizarse luego de pocas iteraciones, como se puede observar. Es decir, luego de esa cantidad de iteraciones, no mejora la aproximación. En particular, se puede ver que para el rango de precisiones del trabajo, no tiene sentido hacer más de 25 iteraciones.

También se puede observar como el algoritmo se acerca de igual manera al

resultado, independentemente de la presicion con la que se trabaje. Pero la precisión termina definiendo cuánto se acercará al valor buscado. Es decir, con más precisión se acercará a un valor más próximo al buscado.

5 Conclusiones

Luego de analizar todos los resultados se llegó a la conclusión de que los errores númericos que se generan al trabajar con una aritmética finita son muy variados. Los mismos pueden aparecer tanto por particularidades del método que se está implementando como por particularidades de la implementacion. Las diferentes decisiones de implementación afectan los resultados de diferentes formas, y el efecto que tienen sobre los errores dependen también de los algoritmos con los que se trabaja.

Esto se pudo ver, por ejemplo, al analizar cómo se ven afectados los resultados al elegir truncamiento o redondeo al ajustar la precisión de un número. Por ejemplo, para el método babilonio, esta decisión tiene una influencia más que significativa.

También se vio cómo decisiones de la implementación pueden lograr que no se cometan ciertos errores. Esto se puede ver en el caso del método binomial. En este caso, cuando se realiza una implementación que acumula los valores desde el más grande hasta el más chico, contra una que lo hace en el sentido contrario, los errores cometidos son menores en el caso de la segunda opción.

Además, se vio que no necesariamente un método mejora si se aumentan la cantidad de términos sumados (de hecho, a veces empeora), por lo que a veces tiene sentido no correr demasiadas iteraciones de un método. También, no siempre tener mayor precisión asegura una mejor aproximación al resultado.

6 Apéndice A

7 Apéndice B

Para compilar todos los métodos se debe ejecutar el comando "make". Se necesita tener el compilador gcc. Este comando generará los siguientes archivos ejecutables:

- babilonio
- binomial
- binomial_dec
- binomial_inc
- fcont
- fcont_int

Al ejecutarlos[1], los mismos darán un detalle en pantalla sobre los parámetros de entrada, estos son *<prec>*, *<niter>* en el caso de los algoritmos binomiales y babilonio. Estos son la prescion elegida y la cantidad de iteraciones deseadas. En el caso de fracciones continuas, además tiene el parámetro opcional *[aprox]*, este parámetro sirve como aproximación inicial de $\sqrt{2}$ para el algoritmo.

Los algoritmos imprimen en pantalla el resultado final obtenido para los parámetros dados.

[1] En linux se ejecuta: .\<nombre>