Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Métodos Numéricos

Trabajo Práctico Nº1

Errores en serie

Nombre	LU	Mail
Carla Livorno	424/08	carlalivorno@hotmail.com
Mariano De Sousa Bispo	389/08	marian_sabianaa@hotmail.com

Abstract

El siguiente trabajo consiste en analizar teóricamente la propagación de errores de cada una de las series propuestas, en función de la cantidad de dígitos de precisión de la aritmética utilizada para luego contrastar dicho análisis con los resultados empíricos. El trabajo pretende mostrar en los resultados lo que pueden producir pequeños errores de cálculo.

Aritmética finita Errores Pi

Índice

1. Introducción teórica

Cuando trabajamos con números reales en una computadora, nos encontramos con algunos factores limitantes para nuestros cálculos como lo son la aritmética finita y también el tipo de representación que se nos provee para trabajar.

Esto surge como una consecuencia de la necesidad de una memoria física finita de la computadora en contraste con la precisión infinita que requieren la mayoría de los números reales.

Luego, al intentar representar números en una computadora, cuya precisión va mas allá de la otorgada por la misma, surgen en la representación del número pequeños errores. Tambien se propagan los errores de redondeo y truncamiento debidos al almacenamiento a lo largo de las diferentes operaciones a que sometemos a los números. Estos errores, como muestra este trabajo, no son despreciables al realizarse ciertos tipos de cálculos.

Por otro lado, el álgebra de precisión finita es diferente al álgebra normal, la propiedad asociativa de los números reales no se cumple en general para los datos en punto flotante debido al error de redondeo. Si los valores a sumar son de muy diferente orden de magnitud se producirá un resultado más aproximado al correcto si se suman ordenadamente de menor a mayor. Esto nos demuestra que el orden en que se ejecutan las operaciones es importante para un computador.

2. Desarrollo

2.1. Analisis teórico

3. Serie de Gregory

Definimos primero el error de un término de la serie de Gregory.

Dado $n \in \mathbb{N}$, sea `f(x,y)` $= \frac{x}{y}$, donde $x = 1$ e $y = 2n + 1$.

El error de `f`(x, y) es el error de un término de la serie de Gregory ya que x e y son de tipo entero (exactos).

Analicemos el error de `f`(x, y):

$$\varepsilon_f(x, y, :) = \frac{\frac{x}{y}\varepsilon_x}{f} + \frac{\frac{-xy}{y^2}\varepsilon_y}{f} + \varepsilon_{(:)}^{(x,y)} = \frac{\frac{x}{y}\varepsilon_x}{\frac{x}{y}} + \frac{\frac{-x}{y}\varepsilon_y}{\frac{x}{y}} + \varepsilon_{(:)}^{(x,y)} =$$

$$\varepsilon_x - \varepsilon_y + \varepsilon_{(:)}^{(x,y)}$$

Teniendo esta información, queremos realizar el análisis teórico sobre la serie de Gregory para los primeros tres términos. Para poder realizarlo, llamamos a al primer término de la serie, b al segundo y c al tercero (llamamos $k = a + c$).

Sea `g(a,c)` $= a + c$

$$\varepsilon_g(a, c, +) = \frac{a}{a+c}\varepsilon_a + \frac{c}{a+c}\varepsilon_c + \varepsilon_{(+)}^{(a,c)}$$

Sea `g'(k,b)` $= k - b$

$\varepsilon_{g'}(k,b,-) = \frac{k}{k-b}\varepsilon_k - \frac{b}{k-b}\varepsilon_b + \varepsilon^{(k,b)}_{(-)}$

$\varepsilon_g(a+c,b,+) = \frac{a+c}{a+c-b}(\frac{a}{a+c}\varepsilon_a + \frac{c}{a+c}\varepsilon_c + \varepsilon^{(a,c)}_{(+)}) - \frac{b}{a+c-b}\varepsilon_b + \varepsilon^{(a+c,b)}_{(-)} =$

$\frac{a\varepsilon_a + c\varepsilon_c - b\varepsilon_b + (a+c)\varepsilon^{(a,c)}_{(+)}}{a-b+c} + \varepsilon^{(a+c,b)}_{(-)} = \varepsilon_{a+c-b} = \varepsilon_{\frac{\pi}{4}}$

Entonces el error total (ε_π) es el siguiente:

$\varepsilon_4 + \varepsilon_{a+c-b} + \varepsilon^{(4,a+c-b)}_{(*)} =$

$\varepsilon_4 + \frac{a\varepsilon_a + c\varepsilon_c - b\varepsilon_b + (a+c)\varepsilon^{(a,c)}_{(+)}}{a-b+c} + \varepsilon^{(a+c,b)}_{(-)} + \varepsilon^{(4,a+c-b)}_{(*)}$

Dado que $a = 1$, $b = \frac{1}{3}$ y $c = \frac{1}{5}$. Reemplazando por los términos correspondientes obtenemos:

$$\varepsilon_4 + \frac{\varepsilon_1 - \varepsilon_1 + \varepsilon^{(1,1)}_{(:)} - (\frac{1}{3})(\varepsilon_1 - \varepsilon_3 + \varepsilon^{(1,3)}_{(:)}) + (\frac{1}{5})(\varepsilon_1 - \varepsilon_5 + \varepsilon^{(1,5)}_{(:)}) + \frac{6}{5}\varepsilon^{(1,\frac{1}{5})}_{(+)}}{\frac{13}{15}} + \varepsilon^{(\frac{6}{5},\frac{1}{3})}_{(-)} + \varepsilon^{(4,\frac{13}{15})}_{(*)}$$

Si bien no conocemos cuál es el error exacto que posee cada operación, podemos acotarlas por el error máximo de todas ellas, llamémoslo k. Por lo tanto el resultado obtenido es menor igual a:

$k + \frac{k - (\frac{1}{3})k + (\frac{1}{5})k + \frac{6}{5}k}{\frac{13}{15}} + 2k = k + \frac{\frac{17}{15}k}{\frac{13}{15}} + 2k = k + \frac{17}{13}k + 2k = 3k + \frac{17}{13}k = \frac{56}{13}k < 5k$

Sabiendo que la implementación de estos algoritmos será realizada en una máquina con arítmetica binarias, podemos inferir que el error en los cálculos se producirá a partir de cierto bit, al que llamaremos t. El error entonces depende de la conversión de los valores enteros a números con coma, así como también

la presición que estemos utilizando. Tomamos de esta forma a $k = 2^{1-t}$.

Acotando el error de Gregory, tenemos que $\varepsilon_\pi < 52^{1-t}$

NOTA: Este último resultado (el valor de k), aplica tanto como al análisis previamente realizado, como a los dos subsiguientes (algoritmo de Machin y Ramanujan). Para simplificar los cálculos, y disminuir la probabilidad de arrastrar errores de cuentas, se acotará por k el que luego será reemplazado por 2^{1-t}.

4. Fórmula de Machin

Definimos primero el error de calcular $arctan(k)$ con $|k| < 1$. Para ello definimos el error de cada uno de los términos de la sumatoria.

Dado $n \in \mathbb{N}$, sea $\texttt{f(x,y)} = \frac{x}{y}$, donde $x = k^{2n+1}$ e $y = 2n + 1$.

Analicemos el error de $\texttt{f}(x, y)$:

$$\varepsilon_f(x, y, :) = \frac{\frac{x}{y}\varepsilon_x}{f} + \frac{\frac{-xy}{y^2}\varepsilon_y}{f} + \varepsilon_{(:)}^{(x,y)} = \frac{\frac{x}{y}\varepsilon_x}{\frac{x}{y}} + \frac{\frac{-x}{y}\varepsilon_y}{\frac{x}{y}} + \varepsilon_{(:)}^{(x,y)} =$$

$$\varepsilon_x - \varepsilon_y + \varepsilon_{(:)}^{(x,y)} \quad (1)$$

Como y es de tipo entero (exacto) resta calcular el error de x.

Analicemos el error de $x = k^y$:

$$\varepsilon_x = \frac{yk^{y-1}y}{k^y}\varepsilon_y + \varepsilon_{(POT)}^{(k,y)} = \frac{y^2}{k}\varepsilon_y + \varepsilon_{(POT)}^{(k,y)} \quad (2)$$

$$\Rightarrow_{Reemplazando\ (2)\ en\ (1)} \frac{y^2}{k}\varepsilon_y + \varepsilon_{(POT)}^{(k,y)} - \varepsilon_y + \varepsilon_{(:)}^{(x,y)}$$

Teniendo esta información, queremos realizar el análisis teórico para los primeros tres términos de la sumatoria. Para poder realizarlo, llamamos a al primer término, b al segundo y c al tercero (llamamos $k = a + c$).

Sea $\texttt{g(a,c)} = a + c$

$$\varepsilon_g(a, c, +) = \frac{a}{a+c}\varepsilon_a + \frac{c}{a+c}\varepsilon_c + \varepsilon_{(+)}^{(a,c)}$$

Sea `g'(k,b)` $= k - b$

$\varepsilon_{g'}(k, b, -) = \frac{k}{k-b}\varepsilon_k - \frac{b}{k-b}\varepsilon_b + \varepsilon_{(-)}^{(k,b)}$

$\varepsilon_g(a + c, b, +) = \frac{a+c}{a+c-b}(\frac{a}{a+c}\varepsilon_a + \frac{c}{a+c}\varepsilon_c + \varepsilon_{(+)}^{(a,c)}) - \frac{b}{a+c-b}\varepsilon_b + \varepsilon_{(-)}^{(a+c,b)} =$

$\frac{a\varepsilon_a + c\varepsilon_c - b\varepsilon_b + (a+c)\varepsilon_{(+)}^{(a,c)}}{a-b+c} + \varepsilon_{(-)}^{(a+c,b)} = \varepsilon_{a+c-b}$

Para la fórmula de Machin se necesita calcular $arctan(\frac{1}{5})$ y $arctan(\frac{1}{239})$.

$\Rightarrow \varepsilon_{arctan}(\frac{1}{5}) =$

$\frac{\frac{1}{5}(5\varepsilon_1 + \varepsilon_{(POT)}^{(\frac{1}{5},1)} - \varepsilon_1 + \varepsilon_{(:)}^{(\frac{1}{5},1)}) + \frac{1}{5}^6(5^3\varepsilon_5 + \varepsilon_{(POT)}^{(\frac{1}{5},5)} - \varepsilon_5 + \varepsilon_{(:)}^{(\frac{1}{5}^5,5)}) - \frac{1}{5}^4(3^2 5\varepsilon_3 + \varepsilon_{(POT)}^{(\frac{1}{5},3)} - \varepsilon_3 + \varepsilon_{(:)}^{(\frac{1}{5}^3,3)}) + (\frac{1}{5} + \frac{1}{5}^6)\varepsilon_{(+)}^{(\frac{1}{5},c)}}{\frac{1}{5} - \frac{1}{5}^4 + \frac{1}{5}^6} +$

$\varepsilon_{(-)}^{(\frac{1}{5}+\frac{1}{5}^6, \frac{1}{5}^4)}$

$\Rightarrow \varepsilon_{arctan}(\frac{1}{239}) =$

$\frac{\frac{1}{239}(239\varepsilon_y + \varepsilon_{(POT)}^{(\frac{1}{239},1)} - \varepsilon_1 + \varepsilon_{(:)}^{(\frac{1}{239},1)}) + \frac{\frac{1}{239}^5}{5}(239 * 5^2\varepsilon_5 + \varepsilon_{(POT)}^{(\frac{1}{239},5)} - \varepsilon_5 + \varepsilon_{(:)}^{(\frac{1}{239}^5,5)}) - \frac{\frac{1}{239}^3}{3}(3^2 239\varepsilon_3 + \varepsilon_{(POT)}^{(\frac{1}{239},3)} - \varepsilon_3 + \varepsilon_{(:)}^{(\frac{1}{239}^3,3)})}{\frac{1}{239} - \frac{\frac{1}{239}^3}{3} + \frac{\frac{1}{239}^5}{5}} +$

$\frac{(\frac{1}{239} + \frac{\frac{1}{239}^5}{5})\varepsilon_{(+)}^{(\frac{1}{239}, \frac{\frac{1}{239}^5}{5})}}{\frac{1}{239} - \frac{\frac{1}{239}^3}{3} + \frac{\frac{1}{239}^5}{5}} + \varepsilon_{(-)}^{(\frac{1}{239}+\frac{\frac{1}{239}^5}{5}, \frac{1}{239}^3)}$

Finalmente calculamos el error de la fórmula de Machin.

Sea `f'` $= z * 4$,

$\varepsilon_{f'}(z, 4, *) = \varepsilon_z + \varepsilon_4 + \varepsilon_{(*)}^{(z,4)}$ (3)

Por otro lado calculamos el error de z instanciando g' con

$k = 4 * arctan(\frac{1}{5})$ y $b = arctan(\frac{1}{239})$

$\varepsilon_z = \frac{4*arctan(\frac{1}{5})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}\varepsilon_{4*arctan(\frac{1}{5})} - \frac{arctan(\frac{1}{239})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}\varepsilon_{arctan(\frac{1}{239})} + \varepsilon_{(-)}^{(\frac{1}{5},\frac{1}{239})}(4)$

$\Rightarrow$ *Reemplazando* (4) *en* (3)

$\frac{4*arctan(\frac{1}{5})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}\varepsilon_{4*arctan(\frac{1}{5})} - \frac{arctan(\frac{1}{239})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}\varepsilon_{arctan(\frac{1}{239})} + \varepsilon_{(-)}^{(\frac{1}{5}),\frac{1}{239}))} + \varepsilon_4 + \varepsilon_{(*)}^{(4*arctan(\frac{1}{5}-arctan(\frac{1}{239}),4)}$

$\Rightarrow$

$\frac{4*arctan(\frac{1}{5})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}(\varepsilon_{arctan(\frac{1}{5})}+\varepsilon_4+\varepsilon_{(*)}^{(arctan(\frac{1}{5}),4)}) - \frac{arctan(\frac{1}{239})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}\varepsilon_{arctan(\frac{1}{239})} + \varepsilon_{(-)}^{(\frac{1}{5}),\frac{1}{239}))} + \varepsilon_4 + \varepsilon_{(*)}^{(z,4)} <$

$\Rightarrow$ *Acotando los errores por* k, *es menor a*:

$< \frac{4*arctan(\frac{1}{5})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}3k - \frac{arctan(\frac{1}{239})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}k + 3k$

$\Rightarrow$

$< \frac{4*arctan(\frac{1}{5})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}3k + \frac{arctan(\frac{1}{239})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})}3k + 3k$

$\Rightarrow$

$< (\frac{4*arctan(\frac{1}{5})+arctan(\frac{1}{239})}{4*arctan(\frac{1}{5})-arctan(\frac{1}{239})} + 1)3k < (\frac{5*arctan(\frac{1}{5})}{3*arctan(\frac{1}{239})} + 1)3k <$

DESDE ACA

$< 487k = 336k = 2^4$,3,7

Reemplazando $k = 2^{1-t}$ nos queda que el error relativo del algoritmo de Machin:

$\varepsilon_\pi < 2^4 x 3 x 7 x 2^{1-t} = 3 x 7 x 2^{5-t}$

5. Serie de Ramanujan

Definimos primero el error de un término de la serie de Ramanujan.

Dado $n \in \mathbb{N}$, sea `f(x,y)` $= \frac{x}{y}$, donde $x = (4n)!(1103+26390n)$ e $y = (n!)^4 396^{4n}$.

El error de $\mathtt{f}(x, y)$ es el error de un término de la serie de Gregory ya que x e y son de tipo entero (exactos).

Analicemos el error de $\mathtt{f}(x, y)$:

$$\varepsilon_f(x, y, :) = \frac{\frac{x}{y}\varepsilon_x}{f} + \frac{\frac{-xy}{y^2}\varepsilon_y}{f} + \varepsilon_{(:)}^{(x,y)} = \frac{\frac{x}{y}\varepsilon_x}{\frac{x}{y}} + \frac{\frac{-x}{y}\varepsilon_y}{\frac{x}{y}} + \varepsilon_{(:)}^{(x,y)} =$$

$$\varepsilon_x - \varepsilon_y + \varepsilon_{(:)}^{(x,y)}$$

Queremos realizar el análisis teórico sobre la serie de Ramanujan para los primeros tres términos de la sumatoria. Para poder realizarlo, llamamos a al primer término de la serie, b al segundo y c al tercero (llamamos $k = a + b$).

Sea `g(k,c)` $= k + c$

$$\varepsilon_g(k, c, +) = \frac{k}{k+c}\varepsilon_k + \frac{c}{k+c}\varepsilon_c + \varepsilon_{(+)}^{(k,c)}$$

$$\varepsilon_g(a + b, c, +) = \frac{a+b}{a+b+c}\left(\frac{a}{a+b}\varepsilon_a + \frac{b}{a+b}\varepsilon_b + \varepsilon_{(+)}^{(a,b)}\right) + \frac{c}{a+b+c}\varepsilon_c + \varepsilon_{(+)}^{(a+b,c)} =$$

$$= \frac{a\varepsilon_a + b\varepsilon_b + c\varepsilon_c + (a+b)\varepsilon_{(+)}^{(a,b)}}{a+b+c} + \varepsilon_{(+)}^{(a+b,c)}$$

Dado $a = 0$, $b = \frac{659832}{396^4}$ y $c = \frac{8!*53883}{2^4*396^8}$ el error de los tres primeros términos es el siguiente:

$$\frac{\frac{659832}{396^4}(\varepsilon_{659832}-\varepsilon_{396^4}+\varepsilon_{(:)}^{(659832,396^4)})+\frac{8!*53883}{2^4*396^8}(\varepsilon_{8!*53883}-\varepsilon_{2^4*396^8}+\varepsilon_{(:)}^{(8!*53883,2^4*396^8)})+\frac{659832}{396^4}\varepsilon_{(+)}^{(0,\frac{659832}{396^4})}}{\frac{659832}{396^4}+\frac{8!*53883}{2^4*396^8}}+$$
$$\varepsilon_{(+)}^{(\frac{659832}{396^4},\frac{8!*53883}{2^4*396^8})}$$

Por otro lado calculamos el error de `h(z)` $= \sqrt{z}$.

$$\varepsilon_h(z,\surd) = \frac{\frac{1}{2\sqrt{z}}}{\sqrt{z}}z\varepsilon_z + \varepsilon_{(\surd)}^{(z)} = \frac{1}{2\sqrt{z}}\frac{1}{\sqrt{z}}z\varepsilon_z + \varepsilon_{(\surd)}^{(z)} = \frac{\varepsilon_z}{2} + \varepsilon_{(\surd)}^{(z)}$$

Conociendo el error de $\mathtt{f}(x,y)$ calculamos el error de $\frac{\sqrt{8}}{9801}$ instanciando $x = \sqrt{z}$ e $y = 9801$.

$$\Rightarrow \frac{\varepsilon_z}{2} + \varepsilon_{(\surd)}^{(z)} - \varepsilon_{9801} + \varepsilon_{(:)}^{(\sqrt{8},9801)}$$

Sea `h'(x,y)` $= x*y$, donde $x = \frac{\sqrt{8}}{9801}$ e $y = \sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}}$.

sabemos que: $\varepsilon_{h'}(x,y,*) = \varepsilon_x + \varepsilon_y + \varepsilon_{(*)}^{(x,y)}$

$$\Rightarrow \varepsilon_{h'}(x,y,*) = \varepsilon_{\frac{\sqrt{8}}{9801}} + \varepsilon_{\sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}}} + \varepsilon_{(*)}^{(\frac{\sqrt{8}}{9801},\sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}})} =$$
$$\frac{\frac{659832}{396^4}(\varepsilon_{659832}-\varepsilon_{396^4}+\varepsilon_{(:)}^{(659832,396^4)})+\frac{8!*53883}{2^4*396^8}(\varepsilon_{8!*53883}-\varepsilon_{2^4*396^8}+\varepsilon_{(:)}^{(8!*53883,2^4*396^8)})+\frac{659832}{396^4}\varepsilon_{(+)}^{(0,\frac{659832}{396^4})}}{\frac{659832}{396^4}+\frac{8!*53883}{2^4*396^8}}+$$
$$\varepsilon_{(+)}^{(\frac{659832}{396^4},\frac{8!*53883}{2^4*396^8})} + \frac{\varepsilon_z}{2} + \varepsilon_{(\surd)}^{(z)} + \varepsilon_{(*)}^{(\frac{\sqrt{8}}{9801},\sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}})}$$

Conociendo el error de $\mathtt{f}(x,y)$ calculamos el error de $\frac{1}{\frac{\sqrt{8}}{9801}\sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}}}$ instanciando $x = 1$ e $y = \frac{\sqrt{8}}{9801}\sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}}$.

$$\Rightarrow \varepsilon_1 - \frac{\frac{659832}{396^4}(\varepsilon_{659832} - \varepsilon_{396^4} + \varepsilon_{(:)}^{(659832,396^4)}) + \frac{8!*53883}{2^4*396^8}(\varepsilon_{8!*53883} - \varepsilon_{2^4*396^8} + \varepsilon_{(:)}^{(8!*53883,2^4*396^8)}) + \frac{659832}{396^4}\varepsilon_{(+)}^{(0,\frac{659832}{396^4})}}{\frac{659832}{396^4} + \frac{8!*53883}{2^4*396^8}}$$

$$+\varepsilon_{(+)}^{(\frac{659832}{396^4},\frac{8!*53883}{2^4*396^8})} + \frac{\varepsilon_z}{2} + \varepsilon_{(\surd)}^{(z)} + \varepsilon_{(*)}^{(\frac{\sqrt{8}}{9801},\sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}})} + \varepsilon_{(:)}^{(1,\frac{\sqrt{8}}{9801}\sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4 396^{4n}})}$$

5.1. Implementación

Al necesitar manejar presición arbitraria los tipos de datos nativos del lenguaje C++ no satisfacían nuestras necesidades.

Una solución encontrada a este problema, fue implementar un tipo de datos (clase *Real*) con esta funcionalidad (presición variable). La clase *Real* implementa los operadores básicos: suma, resta, multiplicación. división y asignación, los observadores sobre la presición y truncamiento, y funciones para visualización de la información del *Real* en stdout. Se exporta además dos métodos con funcionalidad importante, uno de ellos para convertir el *Real* en *double* (tipo nativo de C++) y otro método que dado un *double* refresca esta información en la instancia *Real* correspondiente.

Como precondición de este trabajo práctico la presición máxima de digitos (en base 2) del cálculo de π debe ser a lo sumo 51 bits. Sabiendo que C++ cumple con el standar IEEE-754, el cual establece 52 bits de mantisa para un tipo de datos *double*, utilizamos a este como cimiento de nuestra clase *Real*. Las operaciones aritméticas sobre *Real* consisten en convertirlos en *double*, realizar la operación correspondiente y transformar el resultado nuevamente en un *Real*, aplicándole previamente el algoritmo de truncamiento acorde a los parámetros de entrada del programa (cantidad de dígitos).

Fuera de la clase implementamos el cálculo de raíz cuadrada, arco tangente y exponenciación de *Real*. Se tomó esta decisión ya que consideramos que estas no son operaciones básicas, por lo que el uso de estas sólo tiene sentido en instancias de *Real* en situaciones complejas (por ejemplo el cálculo de π). Tomamos como modelo la implementación de estas operaciones en C++ que requiere la inclusión de una biblioteca (cmath).

La implementación de todos los algoritmos del cálculo de π maximizan el uso de variables de tipo entero ya que las cuentas en número flotante poseen error mientras que enteros no. Cuando los cálculos dejan de tener sentido en el mundo de los enteros o su presición no es suficiente, se crea una instancia de *Real* que represente al valor entero y se continúa operando sobre este tipo.

Por ese motivo una instancia de *Real* se puede generar sólo a partir de un entero de 64 bits con signo.

A continuación detallamos las optimizaciones de los algoritmos:

- Gregory: Refactorizamos nuestra implementación original de la serie que consistía en respetar el orden de los sumandos según el enunciado como las operaciones internas de cada uno de ellos. A continuación se detalla en lenguaje matemático los cambios realizados.

  $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \sum_{n\in\mathbb{N}/n\ mod(2)=0}^{\infty} \frac{1}{2n+1} - \sum_{n\in\mathbb{N}/n\ mod(2)=1}^{\infty} \frac{1}{2n+1}$

  Al hacerlo de esta manera en la ejecución de cada iteración del algoritmo evitamos el cálculo de una potencia al costo de almacenar dos números que acumulan las sumas.

  Por otro lado, decidimos calcular $2n + 1$ en un tipo de dato entero de 64 bits sin signo ya que de esta forma podemos evitar el error en el cálculo (a diferencia de si se realizara con números flotantes) y sería necesario un n muy grande (cantidad de sumandos de la serie) para producir $overflow$.

- Machin: Este algoritmo ejecuta la función *arctan* la cual realiza la optimización antes mencionada.

- Ramanujan: Este algoritmo fue rediseñado en varias oportunidades al observar diferentes problemas en cada implementación que realizamos. Entre sus operaciones encontramos la función factorial la cual en pocas iteraciones (en enteros sin signo de 64 bits) produce $overflow$. Sabiendo que el rango de valores representable en *double*, y por consiguiente en *Real* es mayor al de cualquier entero de 64 bits, optamos por realizar la operación factorial en nuestro tipo de datos (*Real*).

  - NOTA: No olvidar que la cantidad de valores representable en *double* y enteros de 64 bits es la misma, ya que las permutaciones binarias posibles de ambos es igual a $2^{64} - 1$.

Nuestra primer aproximación fue definir la función factorial, la cual dado un n calculaba la $\prod_{k=1}^{n} k$. Observamos que la cantidad de operaciones elementales que factorial realiza en cada llamada es lineal en función de n. El uso que se le da en el algoritmo de *Ramanujan* es calcular el factorial de números consecutivos pudiéndose entonces, reutilizar el resultado de la iteración previa.

A continuación se muestra un breve pseudcódigo del uso de la función factorial en el algoritmo de *Ramanujan* (primera implementación).

```
Ramanujan(n)
    for j=0  to n
        factorial(4 * j)
            .
            .
            .
    end
```

Sabiendo que factorial es O(n) y la cantidad de iteraciones del ciclo for es n y suponiendo que el resto de las operaciones del ciclo son constantes el algoritmo tiene complejidad cuadrática en función de n.

A continuación se muestra un breve pseudocódigo del cálculo del factorial acumulando en una variable el resultado de la iteración anterior.

```
Ramanujan(n)
    acum_fact = 1
    for j=1  to n
        i = 4 * j
        acum_fact = acum_fact * (i - 1) * (i - 2) * (i - 3) * i
            .
```

.
.

end

De esta forma, se realiza una cantidad constante de multiplicaciones en cada iteración del ciclo for para el cálculo del factorial (suponiendo el resto de las operaciones del ciclo de costo constante) el algoritmo tiene complejidad lineal en función de n.

La complejidad final del algoritmo de *Ramanujan* no es esta ya que entre las operaciones del ciclo for se encuentra la operación potencia con costo lineal en función del exponente $(4 * j)$. La complejidad es de $O(n^2)$.

Cabe mencionar que el primer término de la serie es un valor constante y entero, por lo que decidimos excluirlo del cálculo de la sumatoria y adicionarlo al final (por ese motivo la implementación final itera de 1 a n).

Para facilitar el desarrollo del programa se generaron varias funciones auxiliares de impresión de información. A la hora de hacer los test no era suficiente la visualización provista por la salida standard de C++. Estas visualizaciones como por ejemplo mostrar el *double* bit a bit facilitaron la comprensión y analisis de errores programáticos.

6. Resultados

Para analizar las diferencias y/o similitudes en la aplicación práctica de los tres algoritmos implementados se generaron gráficos que pasan a detallarse a continuación.

El primero de ellos consiste en gráficar el error relativo en función de la cantidad de iteraciones para cada uno de los algoritmos. Se utilizó precisión fija de 51 bits ya que por precondición la cantidad de dígitos de la mantisa debe ser menor a 52, es decir, con 51 dígitos minimizamos el error. Conseguimos la precisión deseada mediante truncamiento. El eje del gráfico que corresponde al *error relativo* está en escala logarítmica para poder apreciar mejor los valores correspondientes dado que estos decrecen exponencialmente.

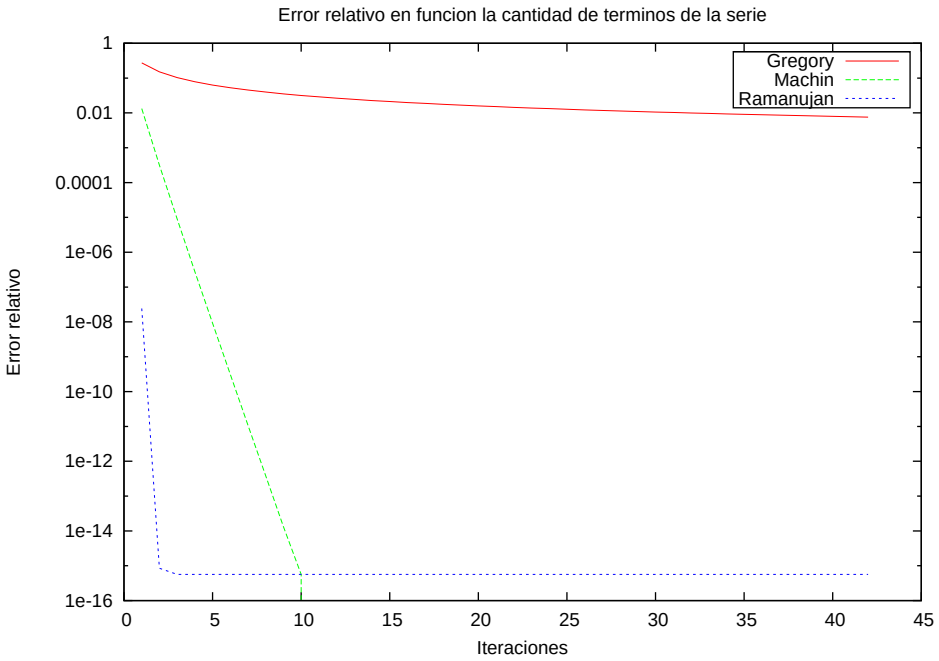

Figura 1: Comparación del error relativo de las tres series variando de 1 a 42 la cantidad de términos calculados con presición de 51 bits en la mantisa.

Como esperabamos observarmos en el gráfico que a medida que la cantidad de términos de la serie (cualquiera de ellas) aumenta el error cometido en el cálculo de π disminuye.

Observamos que el error introducido al aproximar π con la serie de $Gregory$ es logaritmico (en escala logarítmica) en función de la cantidad de términos de la serie calculados. Lo que significa que el error de la serie de $Greory$ decrece linealmente conforme aumenta la cantidad de iteraciones.

Creemos a partir del gráfico que el error relativo cometido al aproximar π con la fórmula de $Machin$ en función de la cantidad de iteraciones es una recta con pendiente negativa. Como la escala utilizada para representar dicho error es la logarítmica, podemos decir que el error decrece exponencialmente.

Además observamos que a partir de la décima iteración el gráfico no muestra de forma clara lo que ocurre con el error en la fórmula de $Machin$. Suponemos que se debe al hecho de que los valores que representan a $Machin$ se superponen con los valores correspondientes a $Ramanujan$ para esa cantidad de iteraciones.

Por otro lado, podemos apreciar que con pocas iteraciones la serie de $Ramanujan$ obtiene un error constante, lo que creemos que ocurre con $Machin$ recien en la iteración diez.

Para poder observar el comportamiento del algoritmo de $Machin$ vs. el de $Ramanujan$ a continuación se adjunta el gráfico con los mismos parámetros (51 digitos de presición), el cual muestra el error relativo entre la iteración 3 y 12.

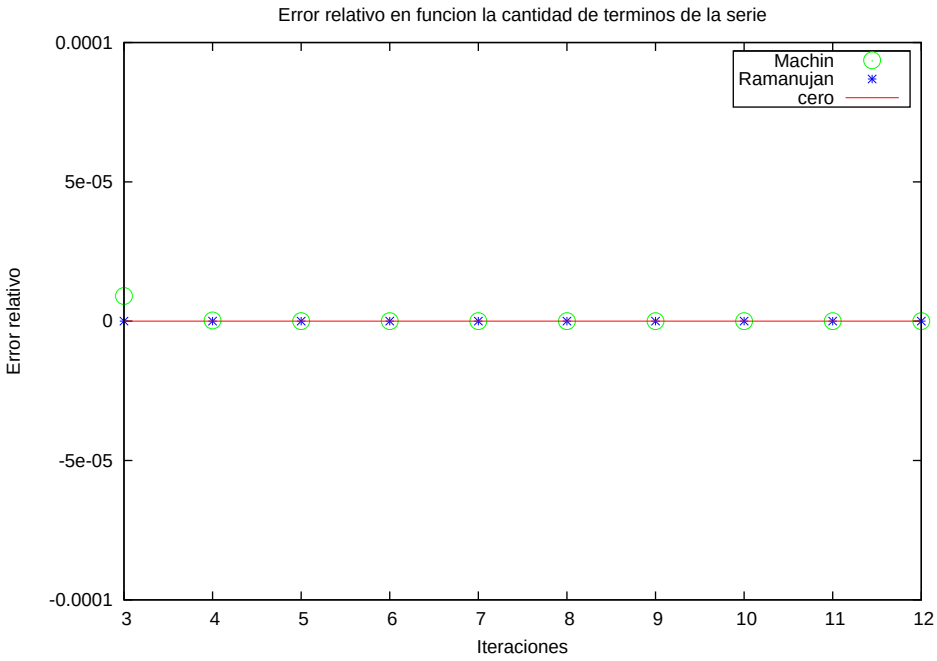

Figura 2: Comparación del error relativo la serie de Machin y de Ramanujan variando de 3 y 12 la cantidad de términos calculados con presición de 51 bits en la mantisa.

Con este gráfico nos vemos tentados a validar lo supuesto en un principio, ya que se puede apreciar que entre estas iteraciones el error cometido por ambos métodos es el mismo. Más aun se podria concluir a partir del gráfico que el error es cero lo cual no es cierto, pero al ser un error tan chico el gráfico ayuda a la malinterpretación de los datos. Igualmente a pesar de seguir iterando no se podria obtener una mejor aproximación a π. El impedimento viene dado por la cantidad de digitos de presición utilizada. A partir de esto, podemos concluir que al tratarse de magnitudes tan pequeñas los gráficos son informativos en rasgos generales y no en detalles.

El siguiente gráfico detalla el error relativo en función de la cantidad de bits de precisión (variando este parámetro de 1 a 51).

Se corrieron las pruebas hasta 42 iteraciones debido a que la serie de *Ramanujan* es informativa hasta esa iteración (a partir de 43 iteraciones devuelve *nan*), es decir, el error relativo presentado corresponde al error cometido al calcular 42 términos de la serie.

El gráfico se presenta bajo una escala logarítmica en y.

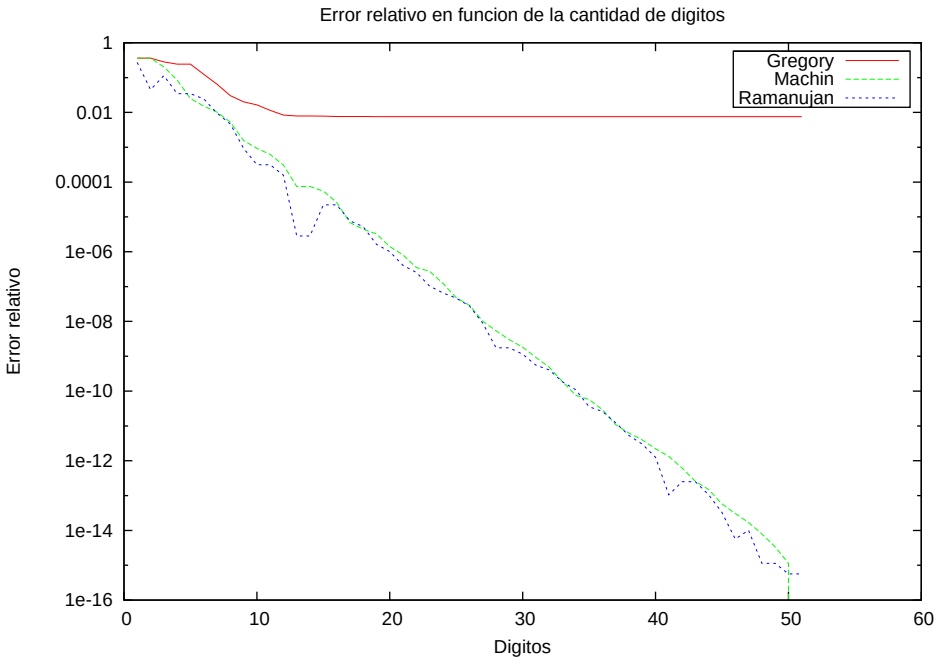

Figura 3: Comparación del error relativo de las tres series en el términos 42 variando de 1 y 51 la cantidad de bits en la mantisa.

La apreciación (lineal, exponencial, etc) de como decrecen los errores en función de la cantidad de dígitos es la misma que en función de la cantidad de iteraciones para *Gregory* y *Machin*.

Ramanujan a diferencia de tener un 'error constante' alcanzado en pocas iteraciones como pasa en el gráfico anterior (51 digitos de presición), parece reducir de manera exponencial el error cometido conforme aumenta la presición utilizada.

Observamos al igual que en el gráfico anterior que el error cometido por *Machin* con 51 dígitos de presición no aparece graficado mientras que con el resto de las precisiones si. Concluimos a partir de esto, que a pesar que el error cometido por ambas series (*Machin* y *Ramanujan*) es similar con 42 iteraciones, se necesita la máxima presicion brindada para que se haga totalmente despreciable la diferencia, si existe.

Por otra parte, vemos que cuando la presición utilizada es baja (con la cantidad de iteraciones usada) las distintas series se comportan de manera similar. Realizamos otro gŕafico para ver que sucede con estas presiciones si la cantidad de terminos calculados es menor. Elegimos arbitrariamente correr las pruebas con 5 iteraciones.

Figura 4: Comparación del error relativo de las tres series en el términos 5 variando de 1 y 51 la cantidad de bits en la mantisa.

Vemos que con poca presición los errores siguen siendo similares.

Comparando este gráfico con el anterior podemos decir que la serie de *Gregory* necesita de muchas iteraciones para poder alcanzar una 'buena' aproximación a π ya que a pesar de ir aumentando la cantidad de dígitos utilizados la mejora en calidad de la aproximación empieza a ser despreciable cuando se alcanza una cantidad de dígitos. Como con menos iteraciones el error se 'estabiliza' (la diferencia en el error al aumentar los dígitos es despreciable) en una cantidad de dígitos menor, dicho error es mayor. Lo mismo ocurre con la fórmula de *Machin* pero a mayor cantidad de dígitos. Creemos que esto ocurre porque como *Machin* consigue mejores aproximaciones que

Gregory fijando una cantidad de iteraciones, cuando la presición es baja pierde mayor información al truncar lo que se ve reflejado en la diferencia de errores al aumentar dicha presición (necesita mas dígitos para estabilizarse).

NOTA: No se utilizaron polinomios interpoladores para aproximar por la curva o recta que pase por todos los puntos correspondiente a una misma fuente de datos, estas conclusiones fueron sacadas utilizando experimentación empírica, suponiendo que el comportamiento cuando los valores del eje x tienden a infinito se corresponden a los de los valores graficados.

7. Discusión

lalala

8. Conclusiones

lalala

9. Apéndices

9.1. Apéndice A: Enunciado

Laboratorio de Métodos Numéricos - Primer cuatrimestre 2011
Trabajo Práctico Número 1: Errores en serie ...

El objetivo del trabajo práctico es analizar el comportamiento numérico de varios métodos para aproximar π cuando cada método se implementa por medio de una aritmética finita de t dígitos. Los métodos consisten en evaluar las siguientes series hasta un cierto término:

Serie de Gregory (1671):

$$\frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

Fórmula de Machin (1706):

$$\frac{\pi}{4} = 4\arctan(1/5) - \arctan(1/239), \qquad \text{con } \arctan(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1} \quad \text{cuando } |x| < 1$$

Serie de Ramanujan (1914):

$$\frac{1}{\pi} = \frac{\sqrt{8}}{9801} \sum_{n=0}^{\infty} \frac{(4n)!\,(1103 + 26390n)}{(n!)^4\, 396^{4n}}$$

El trabajo práctico consta de dos partes:

1. **Análisis teórico**

   Analizar la propagación de errores en el tercer término (en función de los errores de los dos primeros) de cada una de las series propuestas, en función de la precisión aritmética utilizada. Emplear estos resultados en la argumentación de sus conclusiones luego de realizar el análisis empírico del punto 2.

2. **Análisis empírico**

   Implementar los tres métodos con aritmética binaria de punto flotante con t dígitos de precisión en la mantisa (el valor t debe ser un parámetro de la implementación, $t < 52$) y comparar los errores relativos de los resultados obtenidos en los tres casos y con las cotas de error del punto anterior. Realizar al menos los siguientes experimentos numéricos:

   a) Reportar el error relativo de cada método en función de la cantidad de dígitos t de precisión en la mantisa, para cantidades fijas de términos n de la serie.

   b) Reportar el error relativo de cada método en función de la cantidad de términos n de la serie correspondiente, para cantidad de dígitos t de precisión.

   c) Comparar los resultados obtenidos en *a*) y *b*) con las cotas de error calculadas en el punto 1.

   d) (Opcional) Explorar distintas formas de implementar las fórmulas (realizar las cuentas) de cada cada método.

   Se deben presentar los resultados de estas pruebas en un formato conveniente para su visualización y análisis. Sobre la base de los resultados obtenidos, ¿se pueden extraer conclusiones sobre la conveniencia de utilizar uno u otro método?

9.2. Apéndice B: Códigos fuente

10. Referencias

lalala