# Métodos Numéricos

Trabajo Práctico 1

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

## Errores



| Integrante | LU | Correo electrónico |
|---|---|---|
| Aronson, Alex | 443/08 | alexaronson@gmail.com |
| Nahabedian, Leandro | 250/08 | leanahabedian@hotmail.com |
| Ravasi, Nicolás | 53/08 | nravasi@gmail.com |

**Resumen:** Se puede aproximar una función trigonométrica utilizando una sumatoria finita de términos, pero eso supone una aproximación. Además, los números con infinitos decimales necesitan ser truncados o redondeados por la máquina. Por lo tanto, intentar calcular el coseno de un número con una serie finita de McLaurin nos da una aproximación del resultado, buena o mala, pero aproximación al fin y al cabo. Analizaremos los errores involucrados, la propagación de los mismos, y las formas de aproximar y qué tan efectivas son a lo largo de este TP

**Palabras clave: McLaurin coseno aproximación mantisa**

# Contents

**1 Introducción** **4**

**2 Desarrollo** **5**

2.1 Análisis Teórico . . . . . 5

2.1.1 Análisis sobre la finitud de los términos . . . . . 5

2.1.2 Análisis del error en la sumatoria . . . . . 6

2.1.3 Cálculo del error en el término $t_i$ . . . . . 7

2.2 Análisis Empírico . . . . . 9

2.2.1 Cálculo del coseno . . . . . 9

2.2.2 Mantisa variable . . . . . 9

**3 Resultados** **11**

3.1 Análisis con cantidad de términos variable . . . . . 11

3.1.1 Comparación del error absoluto en función de la entrada, para varias cantidades de términos fijas . . . . . 11

3.1.2 Gráficos del error en función de la cantidad de términos, para varias entradas fijas . . . . 12

3.1.3 Comportamiento de la función para una cantidad de términos fija y un espectro de entrada más grande . . . . . 13

3.2 Análisis con tamaños de mantisa variable . . . . . 14

3.2.1 Comparación del comportamiento en función de la cantidad de términos . . . . . 14

3.2.2 Comparación del comportamiento en función del tamaño de la mantisa . . . . . 15

3.2.3 Comparación del comportamiento en función del valor de la entrada . . . . . 17

3.2.4 Comparación de errores obtenidos por la sumatoria con los teóricos . . . . . 19

**4 Discusión** **20**

**5 Conclusión** **21**

**6 Apéndices** **22**

6.1 Apéndice A . . . . . 22

6.2 Apéndice B . . . . . 23

6.2.1 Código fuente de la serie de McLaurin . . . . . 23

6.2.2 Código fuente de la máscara . . . . . 24

**7 Referencias** **26**

# 1 Introducción

Si bien las computadoras nos permiten agilizar mucho el cálculo de operaciones aritméticas, siempre se debe tener en cuenta que poseen memoria finita. Por lo tanto, siempre que se quiera hacer una cuenta que involucre un operando infinito o una cantidad de operaciones infinitas, el resultado no va a poder ser calculado con exactitud y se va a tener que recurrir a una aproximación. Por supuesto, el avance tecnológico hace que esa aproximación sea cada vez más y más cercana al número exacto, pero aún así existe un error en esa aproximación. La serie de Taylor es una forma conocida de calcular una función cuyo cálculo sea difícil (por ejemplo, una función trigonométrica) con operaciones más simples. Esta serie involucra una sumatoria de una cantidad de términos infinita, por lo tanto, si se quiere calcular una serie de Taylor con una computadora, se deberá acotar la cantidad de términos a usar, con el consiguiente error que trae aparejado. La función coseno es una función que tiene su conjunto imagen en los reales (por supuesto, acotado entre -1 y 1), por lo tanto, existen valores que no vamos a poder calcular con exactitud y debremos aproximar. Para este TP, vamos a analizar el comportamiento de la serie de Taylor alrededor del 0 (también llamada serie de McLaurin) para el coseno, la manera de aproximarla con una cantidad finita de términos, diferentes formas de calcularla y el error conllevado con cada una de ellas

# 2 Desarrollo

## 2.1 Análisis Teórico

### 2.1.1 Análisis sobre la finitud de los términos

Nuestro análisis se centró en los errores causados por aproximar la función coseno con la serie de McLaurin, primero nos centramos en el error cometido en función de $x$, al reemplazar la serie por la sumatoria según la cantidad de términos $n$ y hallar una cota para el mismo.

Analizando la serie de McLaurin para el coseno, notamos que el resultado obtenido va a depender de dos parámetros, como ser el valor para el que vamos a aplicar la serie y la cantidad de términos que vamos a usar para evaluar la serie y llegar a la aproximación del resultado del coseno para dicha entrada.

Observando el comportamiento de la serie, notamos que se alternan términos positivos y negativos. Esto significa que cada término de la serie se "pasa" del valor real, ya sea por arriba o por abajo, y cada término subsiguiente que se agrega contribuye a hacer que se "pase" en un valor absoluto menor, por lo tanto, con cada término conseguimos que el resultado obtenido esté más cerca del real. Esta descripción informal de lo que sucede en cada paso puede resumirse a decir que nuestra serie converge cuando la cantidad de términos $n$ tiende a infinito.

Esto nos lleva a analizar el error cometido por la aproximación, algo inevitable considerando que estamos limitando una serie infinita a una cantidad finita de términos.

El error de la serie es, obviamente, la sumatoria del resto de los términos que no sumamos en la serie, o sea, si la serie tiene $n$ términos, o sea, si la función es la sumatoria

$$a = \sum_{i=0}^{n-1} \frac{(-1)^i}{(2i)!} x^{2i} \tag{1}$$

El error de esa cuenta entonces es la sumatoria desde $n$ hasta $\infty$, o sea

$$\varepsilon_{f_n} = \sum_{i=n}^{\infty} \frac{(-1)^i}{(2i)!} x^{2i} \tag{2}$$

Ahora bien, ¿cómo podemos estimar ese error? Tomando lo que dijimos antes, y sabiendo fuertemente que la serie de McLaurin es un ejemplo particular de la serie de Taylor afirmamos de que esta serie conlleva un error. Por el comportamiento de esta, tambien sabemos que a medida que agregamos téminos a nuestra forma de representar la funcion $cosx$ el error se reduce.

O sea, el error de estimar la serie con $n$ términos es siempre menor en valor absoluto al valor del término siguiente, por lo tanto, podemos dar una cota para el error causado por acotar la función, y este resulta

$$\varepsilon_{f_n} < |t_n| \tag{3}$$

Este es el error que sale de acotar la cantidad de términos, pero también existe otro error y es el que sale del hecho de que las máquinas utilizan una aritmética finita, mientras que los números reales son infinitos para cualquier intervalo de estos que tome. Es nuestra elección determinar que números reales representar con mi cantidad de bits disponibles para usar, y al hacer esto dejamos números reales fuera de nuestra representacion

aritmética, causando así un error por aproximar dichos reales no representables por el más cercano en mi aritmética.

### 2.1.2 Análisis del error en la sumatoria

Luego de esto vamos a analizar la propagación de errores en el término general de la sumatoria, en función de la precisión aritmética utilizada.

Para analizar la propagación del error voy a tener que calcular $\varepsilon_{gn}$. Por como esta compuesto $g_n$ digo que:

$$\varepsilon_{g_n} = \varepsilon(...(((t_1 + t_2) + t_3) + t_4)...t_n) \tag{4}$$

Como el cálculo de la propagación del error para la suma es:

$$y = f(a,b) = a + b$$
$$|\varepsilon_y| \leq \frac{|a|}{|a+b|} * |\varepsilon_a| + \frac{|b|}{|a+b|} * |\varepsilon_b| + |\varepsilon_+|$$

Entonces, en la sumatoria la cota del error del primer paréntesis es:

$$|\varepsilon_{t_1+t_2}| \leq \frac{|t_1|}{|t_1+t_2|} * |\varepsilon_{t_1}| + \frac{|t_2|}{|t_1+t_2|} * |\epsilon_{t_2}| + |\varepsilon_+|$$

La cota del error del segundo paréntesis es:

$$|\varepsilon_{t_1+t_2+t_3}| \leq \frac{|t_1+t_2|}{|t_1+t_2+t_3|} * |\varepsilon_{t_1+t_2}| + \frac{|t_3|}{|t_1+t_2+t_3|} * |\varepsilon_{t_3}| + |\varepsilon_+|$$
$$|\varepsilon_{t_1+t_2+t_3}| \leq \frac{|t_1+t_2|}{|t_1+t_2+t_3|} * (\frac{|t_1|}{|t_1+t_2|} * |\varepsilon_{t_1}| + \frac{|t_2|}{|t_1+t_2|} * |\varepsilon_{t_2}| + |\varepsilon_+|) + \frac{|t_3|}{|t_1+t_2+t_3|} * |\varepsilon_{t_3}| + |\varepsilon_+|$$

Si distribuímos la multiplicación nos queda:

$$|\varepsilon_{t_1+t_2+t_3}| \leq \frac{|t_1+t_2|}{|t_1+t_2+t_3|} * \frac{|t_1|}{|t_1+t_2|} * |\varepsilon_{t_1}| + \frac{|t_1+t_2|}{|t_1+t_2+t_3|} \frac{|t_2|}{|t_1+t_2|} * |\epsilon_{t_2}| + \frac{|t_1+t_2|}{|t_1+t_2+t_3|} |\varepsilon_+| + \frac{|t_3|}{|t_1+t_2+t_3|} * |\varepsilon_{t_3}| + |\varepsilon_+|$$
$$|\varepsilon_{t_1+t_2+t_3}| \leq \frac{|t_1|}{|t_1+t_2+t_3|} * |\varepsilon_{t_1}| + \frac{|t_2|}{|t_1+t_2+t_3|} * |\varepsilon_{t_2}| + \frac{|t_3|}{|t_1+t_2+t_3|} * |\varepsilon_{t_3}| + (\frac{|t_1+t_2|}{|t_1+t_2+t_3|} + 1) * |\varepsilon_+|$$

Veamos un caso más:

$$|\varepsilon_{t_1+t_2+t_3+t_4}| \leq \frac{|t_1+t_2+t_3|}{|t_1+t_2+t_3+t_4|} * |\varepsilon_{t_1+t_2+t_3}| + \frac{|t_4|}{|t_1+t_2+t_3+t_4|} * |\varepsilon_{t_4}| + |\varepsilon_+| \leq$$
$$\frac{|t_1+t_2+t_3|}{|t_1+t_2+t_3+t_4|} * (\frac{|t_1|}{|t_1+t_2+t_3|} * |\varepsilon_{t_1}| + \frac{|t_2|}{|t_1+t_2+t_3|} * |\varepsilon_{t_2}| + \frac{|t_3|}{|t_1+t_2+t_3|} * |\varepsilon_{t_3}| + (\frac{|t_1+t_2|}{|t_1+t_2+t_3|} + 1) * |\varepsilon_+|) + \frac{|t_4|}{|t_1+t_2+t_3+t_4|} * |\varepsilon_{t_4}| + |\varepsilon_+| =$$
$$= \frac{|t_1|}{|t_1+t_2+t_3+t_4|} * |\varepsilon_{t_1}| + \frac{|t_2|}{|t_1+t_2+t_3+t_4|} * |\varepsilon_{t_2}| + \frac{|t_3|}{|t_1+t_2+t_3+t_4|} * |\varepsilon_{t_3}| + \frac{|t_4|}{|t_1+t_2+t_3+t_4|} * |\varepsilon_{t_4}| +$$
$$(\frac{|t_1+t_2|}{|t_1+t_2+t_3+t_4|} + \frac{|t_1+t_2+t_3|}{|t_1+t_2+t_3+t_4|} + 1) * |\varepsilon_+|$$

Luego, podemos escribir esto generalizando este concepto obteniendo la fórmula de la propagación del error de $g_n$:

$$|\varepsilon_{g_n}| \leq \frac{|t_1 + t_2 + ... + t_{n-1}|}{|t_1 + t_2 + ... + t_n|} * |\varepsilon_{t_1+t_2...t_{n-1}}| + \frac{|t_n|}{|t_1 + t_{2+}...t_n|} * |\varepsilon_{t_n}| + |\varepsilon_+| \tag{5}$$

y si reemplazamos $|\varepsilon_{t_1+t_2...t_{n-1}}|$ obtendremos la fórmula cerrada de la pinta de lo que venía formándose en $|\varepsilon_{t_1+t_2+t_3+t_4}|$:

$$|\varepsilon_{g_n}| \leq (\sum_{i=1}^{n} |\varepsilon_{t_i}| * \frac{|t_i|}{|t_1 + ... + t_n|}) + |\varepsilon_+| * (1 + \frac{1}{|t_1 + ... + t_n|} * \sum_{i=2}^{n} |t_1 + ... + t_i|) \quad (6)$$

Notar que para facilitar la escritura del producto que esta multiplicando a $|\varepsilon_+|$ colocamos 1+ y luego sacamos $\frac{1}{|t_1+...+t_n|}$ como factor común para poder tener la sumatoria más legible.

### 2.1.3 Cálculo del error en el término $t_i$

Luego de finalizar la escritura de la sumatoria pasemos entonces a hallar la propagación del error de $\varepsilon_{ti}$

Veamos que $t_i = \frac{x^{2i}}{(2i)!}$

Como el cálculo de la propagación del error para la división es:

$$y = f(a, b) = \frac{a}{b}$$
$$|\varepsilon_y| \leq |\varepsilon_a| + |\varepsilon_b| + |\varepsilon_\div|$$

Entonces obtenemos que:

$$|\varepsilon_{t_i}| = \varepsilon_{x^{2i}} + \varepsilon_{(2i)!} + |\varepsilon_\div|$$

Ahora veamos qué pasa en 1) $\varepsilon_{x^{2i}}$ y qué pasa en 2) $\varepsilon_{(2i)!}$

1) Sea

$$x^{2i} = (...((x * x) * x)... *x)$$

Llamemos a $a = (x * x)$ y $b = ((x * x) * x)$ para mostrar el calculo del error en las primeras operaciones "*"

Para esto vamos a utilizar que el error de propagación del producto es:

$$y = f(\alpha, \beta) = \alpha * \beta \qquad \forall\ \alpha,\beta \in \mathbb{R}$$
$$|\varepsilon_y| \leq |\varepsilon_\alpha| + |\varepsilon_\beta| + |\varepsilon_*|$$

El cálculo del error de $|\varepsilon_a|$ y $|\varepsilon_b|$ puede acotarse de la siguiente manera:

$$|\varepsilon_a| \leq |\varepsilon_x| + |\varepsilon_x| + |\varepsilon_*|$$
$$|\varepsilon_a| \leq 2 * |\varepsilon_x| + |\varepsilon_*|$$

Luego

$$|\varepsilon_b| \leq |\varepsilon_a| + |\varepsilon_x| + |\varepsilon_*|$$
$$|\varepsilon_b| \leq 3|\varepsilon_x| + 2|\varepsilon_*|$$

Si extendemos esta idea llegamos a que:

$$|\varepsilon_{x^{2i}}| \leq 2i|\varepsilon_x| + (2i - 1)|\varepsilon_*|$$

2) Ahora sea $y_i = f(i) = (2i)!$

$$|\varepsilon_{y_1}| = |\varepsilon_2|$$
$$|\varepsilon_{y_2}| \leq |\varepsilon_2| + |\varepsilon_3| + |\varepsilon_4| + 2|\varepsilon_*|$$
$$|\varepsilon_{y_3}| \leq |\varepsilon_{y_2}| + |\varepsilon_5| + |\varepsilon_6| + 3|\varepsilon_*|$$

Siguiendo así:

$$|\varepsilon_{y_i}| \leq |\varepsilon_{y_{i-1}}| + |\varepsilon_{2i-2}| + |\varepsilon_{2i-1}| + i|\varepsilon_*|$$
$$|\varepsilon_{y_i}| \leq |\varepsilon_{2i}| + |\varepsilon_{2i-1}| + ... + |\varepsilon_2| + (2i-1)|\varepsilon_*|$$

Ahora, usamos que sea $j$ cualquier número natural vale que el error de calcularlo es igual a todos sin importar que número natural es. O sea $|\varepsilon_j| = h$ donde $j \in \mathbb{N}$ y $h \in \mathbb{R}$

$$|\varepsilon_{y_i}| \leq 2ih + (2i-1)|\varepsilon_*|$$

Finalmente teniendo las dos fórmulas generales, reemplazo:

$$|\varepsilon_{t_i}| = |\varepsilon_{x^{2i}}| + |\varepsilon_{(2i)!}| + |\varepsilon_\div|$$
$$|\varepsilon_{t_i}| \leq 2i|\varepsilon_x| + (2i-1)|\varepsilon_*| + 2ih + (2i-1)|\varepsilon_*| + |\varepsilon_\div|$$
$$|\varepsilon_{t_i}| \leq 2i|\varepsilon_x| + 2(2i-1)|\varepsilon_*| + 2ih + |\varepsilon_\div|$$
$$|\varepsilon_{t_i}| \leq 2i|\varepsilon_x| + 2(2i-1)|\varepsilon_*| + 2ih + \varepsilon_\div|$$

Ahora vamos a utilizar el siguiente teorema[1]:

*Supongamos que los números de punto flotante en una máquina tienen base $\beta$ y una mantisa con $t$ digitos (el digito binario que le da el signo al numero no se cuenta.) Entonces, todo número real en el rango de los puntos flotantes que una máquina puede representar con un error relativo que no excede la unidad de la máquina (**round-off unit**) $u$ esta definido por:*

$$u = \begin{cases} \frac{1}{2} * \beta^{1-t} & \text{si se usa redondeo,} \\ \beta^{1-t} & \text{si se usa truncamiento.} \end{cases}$$

Luego, como todos los números reales pueden ser representados con $t$ dígitos de mantisa con un error relativo que no supera el error de truncamiento, entonces

$$|\varepsilon_l| \leq \beta^{1-t}$$

Donde $\beta = 2$ porque $\beta$ representa la base de nuestra máquina (binaria), $l$ representa cualquier valor, ya sea una operación $(*, \div)$ o una variable $(x, j)$.

Luego la función de $\varepsilon_{t_i}$ al reemplazar $|\varepsilon_l|$ por $\beta^{1-t}$ y $\beta$ por 2, queda acotada por: (recordemos que $h = |\varepsilon_j|$ entonces $h$ tambien puede ser reemplazado por $2^{1-t}$)

$$|\varepsilon_{t_i}| \leq 2i|\varepsilon_x| + 2(2i-1)|\varepsilon_*| + 2ih + \varepsilon_\div|$$
$$|\varepsilon_{t_i}| \leq 2i * 2^{1-t} + 2(2i-1) * 2^{1-t} + 2i * 2^{1-t} + 2^{1-t}$$
$$|\varepsilon_{t_i}| \leq (2^{1-t}) * (2i + 2(2i-1) + 2i + 1)$$
$$|\varepsilon_{t_i}| \leq (2^{1-t}) * (8i-1)$$

[1] Ver referencias en sección 7

## 2.2 Análisis Empírico

### 2.2.1 Cálculo del coseno

En primer lugar se buscó implementar la ecuación de McLaurin tal como nos venía dada en el enunciado. Analizando más profundamente, notamos que para poder propagar los errores de la misma manera que lo habíamos hecho en el análisis teórico, debíamos asegurarnos que las funciones se calcularan de la misma manera, por lo tanto, no podíamos recurrir a la función *pow* de $Math.h$ sino que debíamos programar la función nosotros mismos. Por eso, primero programamos las funciones auxiliares potencia y factorial, que en el primer caso toman un double y un int, que multiplica al double contra sí mismo tantas veces como dice el int; en la segunda, se inicializa dos variables en 1 y se incrementa una hasta llegar al parámetro mientras se multiplica a esa misma sobre la otra en sucesivas iteraciones.

Hecho esto, se decidió que la función main tomaría dos parámetros que serían pasados por linea de comandos como argumentos del programa, el número a evaluar y la cantidad de términos sobre los que se iba a hacerlo. El programa entonces iteraría sobre la cantidad de términos, sumando el resultado del $t_i$ a un acumulado que al terminar la iteración daría el resultado de la función.

Dado que la sumatoria alterna entre términos positivos y negativos, era preciso alternar entre iteración e iteración un booleano que decidiera si se sumaba o se restaba, para eso declaramos la variable signo, y en cada iteración se pregunta su valor y se entra a una u otra rama del if dependiendo de ésta, para sumar o restar según corresponda.

Para mostrar el resultado, pensamos en hacerlo de una manera que sea fácil de mostrar en una tabla pero que a su vez fuera precisa para percibir aún las diferencias más chicas. Notamos que si queríamos imprimir por pantalla con *cout*, la precisión por default que se imprime no era la más adecuada, por lo tanto recurrimos a la función *setprecision* en la que seteamos al principio de la ejecución un número arbitrariamente alto para no perder datos en la impresión.

Hecho esto, imprimimos en pantalla, separados por comas, el número a evaluar, la cantidad de términos usados, el resultado de la función y el resultado de evaluar $\cos(x)$ con $Math.h$ a manera de comparación.

Para el análisis decidimos hacer dos gráficos que ilustran el comportamiento observado. Uno, que graficara los $x$ y $f(x)$ obtenidos para las diferentes cantidades de términos y también para la curva $\cos(x)$, en el rango de valores entre 0 y 3.14, a intervalos de 0.1. Esto nos permite observar qué tanto difiere cada función evaluada sobre $n$ términos contra la función teórica,

En segundo lugar, decidimos tomar una curva lo suficientemente "precisa", para comparar cómo se comportaba contra la curva teórica en un rango no tan acotado. Para esto, elegimos calcular 10 términos de la serie desde -10 a 10, con intervalos de 0.5, así podíamos observar si seguían o no una curva parecida.

### 2.2.2 Mantisa variable

Hecho esto, pasamos a considerar cuánto los valores diferían del resultado empírico del coseno dado por $Math.h$ si modificábamos la estructura del *double* para poder modificar el tamaño de la mantisa por parámetro. Para esto, recurrimos a una máscara, al recibir el tamaño de la mantisa deseado, se crean dos *int* que los inicializamos de la cantidad de unos que va a tener la mantisa, para poder hacer un $AND$ con los resultados que se obtengan y así de esa manera filtrar los valores que no queremos que aparezcan por estar afuera de la mantisa deseada.

La manera que encontramos de crear la máscara fue recurrir a dos *int*, uno va a funcionar como la parte baja de la máscara y otro como la parte alta; ambos son inicializados como 0xFFFFFFFF, o sea, como si no hubiera máscara, ya que vamos a ir agregando ceros. Al recibir como parámetro el tamaño de la mantisa deseado, calculamos cuántos shifts se requerirán (sabiendo que el tamaño máximo de la mantisa es 52), y luego procedemos a shiftear, primero la parte baja, si necesitamos hacer más de 32 shifts, lo hacemos con la parte

alta luego.

Una vez creada la máscara, tenemos la función *domask* que filtra el número para que tenga la mantisa deseada; esta función debe ser llamada cada vez que se hace una operación sobre dicho numero para poder truncarlo (para poder hacerlo en algunas secciones del código recurrimos a variables temporarias para almacenar los resultados parciales) y así hacer que se comporte como si la mantisa tuviera sólo la cantidad de dígitos deseada en lugar de 52.

El resultado fue impreso por pantalla de la misma manera que con el programa anterior, seteando la precisión a un valor fijo arbitrariamente alto, pasra luego imprimir en pantalla, separados por comas, el número a evaluar, la cantidad de términos usados, el tamaño de la mantisa, el resultado de la función y el resultado de evaluar $\cos(x)$ con $Math.h$ a manera de comparación

Para realizar el análisis, decidimos usar un número de términos de la serie fijo, que sería 10, para poder trabajar con una sola variable. De esta manera, calculamos los valores obtenidos para la serie con mantisas variables yendo desde 2 a 52 de 5 en 5, y tomando como parámetro $x$ valores desde 0 hasta 3.14 en intervalos de 0.1 y luego seleccionamos las curvas que nos parecían más significativas.

# 3 Resultados

## 3.1 Análisis con cantidad de términos variable

### 3.1.1 Comparación del error absoluto en función de la entrada, para varias cantidades de términos fijas

En los primeros gráficos comparamos los errores absolutos al hacer $|f(x) - cos(x)|$, usando la función $cos(x)$ de C++, para diferentes cantidades de términos T. En primer lugar, graficamos con T tomando valores entre 2 y 5, para $x$ entre 0 y 3 evaluando en intervalos de 0.2

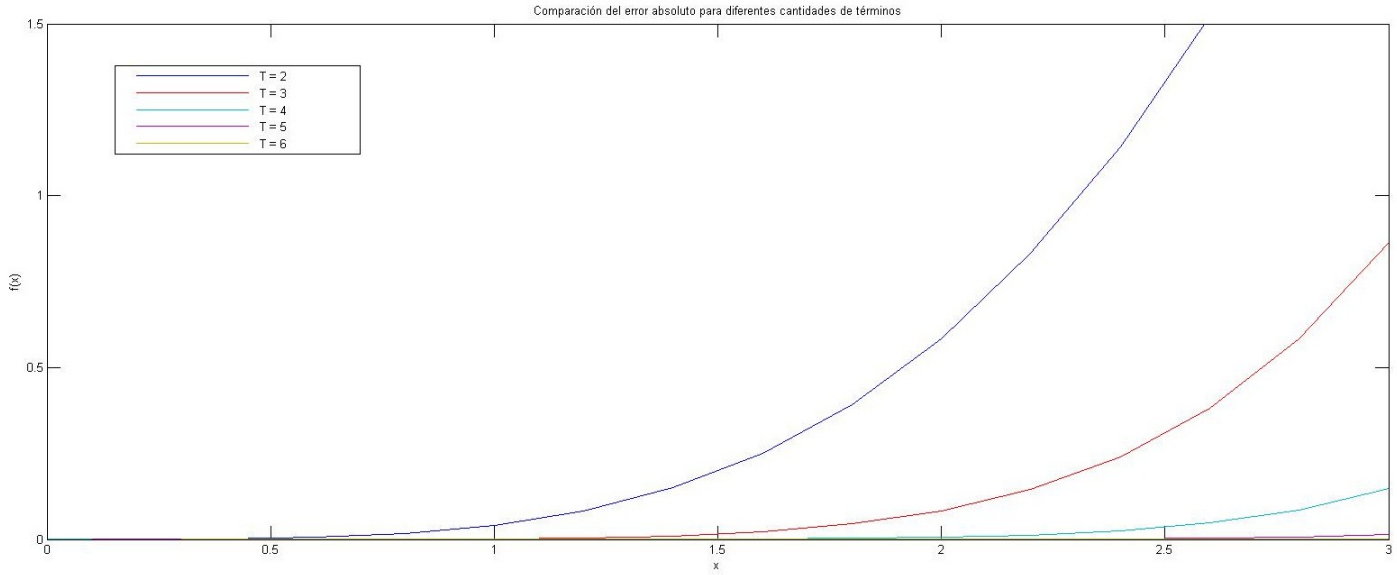


**Gráfico 1.1, comparación del error absoluto para diferentes cantidades de términos T de la serie**

Luego hicimos el mismo gráfico pero para T con valores entre 6 y 8. Nótese que la escala en el eje $y$ está en todos casos multiplicada por $10^{-3}$.

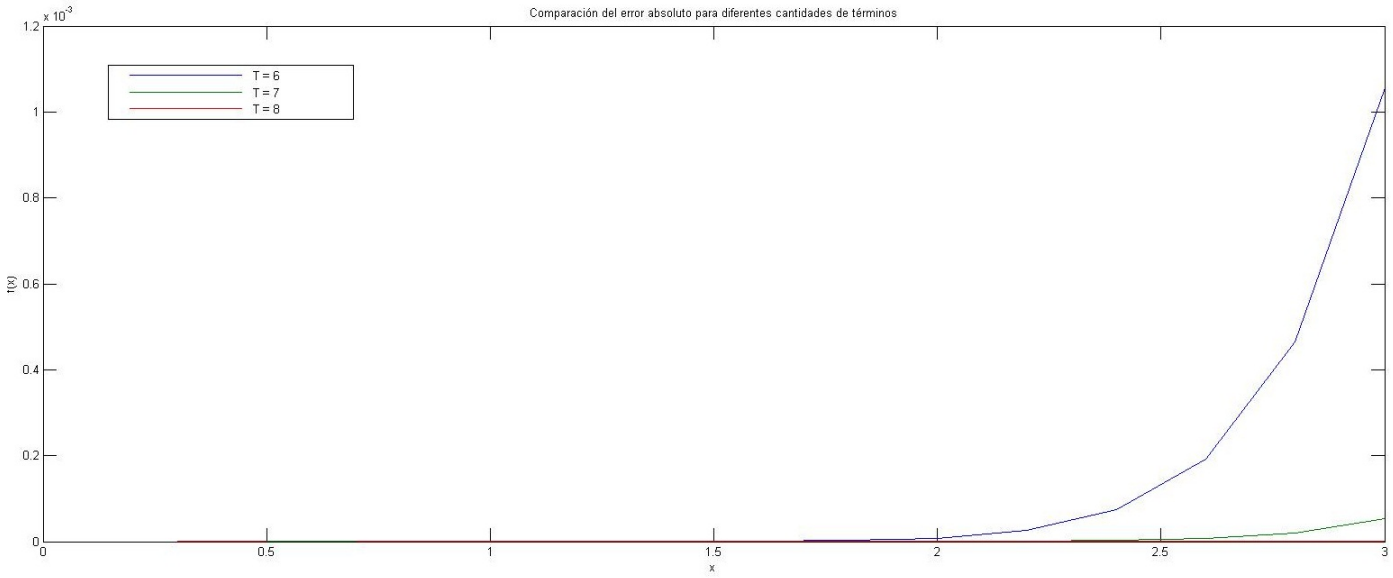


**Gráfico 1.2, comparación del error absoluto para diferentes cantidades de términos T de la serie**

### 3.1.2 Gráficos del error en función de la cantidad de términos, para varias entradas fijas

Una vez terminado de comparar diferentes T, nos interesó comparar el comportamiento del error para diferentes $x$, en función de T. Para esto, obtuvimos el cálculo de $f(x)$ para todas las cantidades de términos de 2 a 8, para diferentes $x$, y obtuvimos el error absoluto de la misma manera que antes.

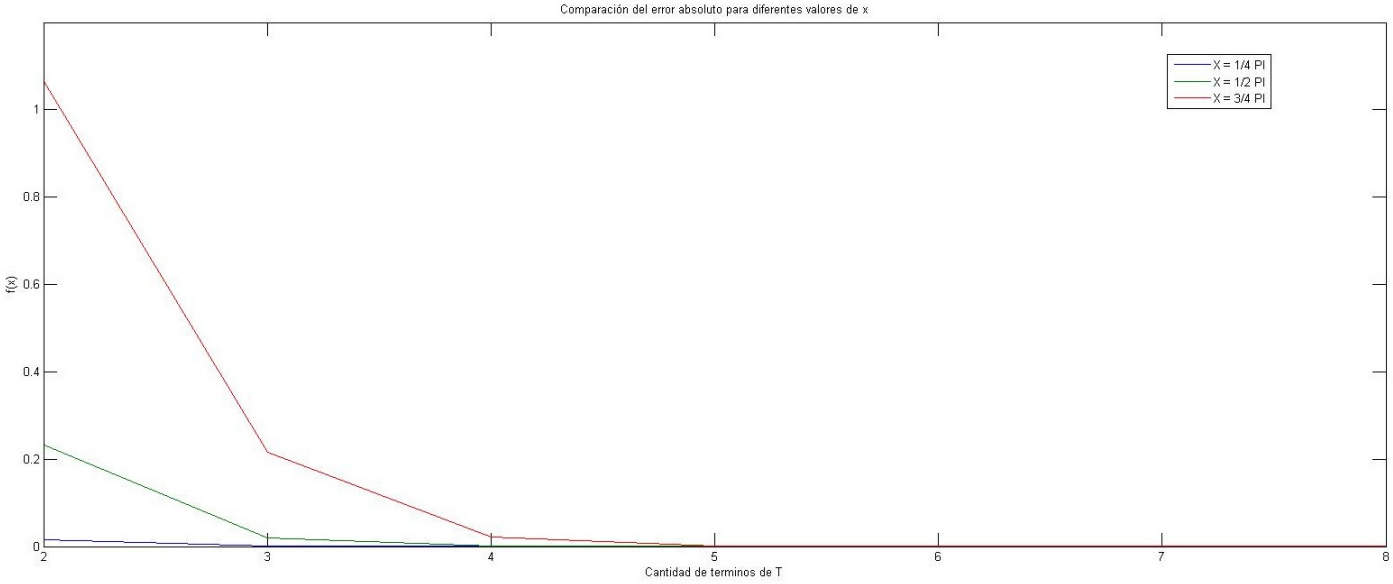


**Gráfico 2.1, comparación del error absoluto para diferentes entradas $x$**

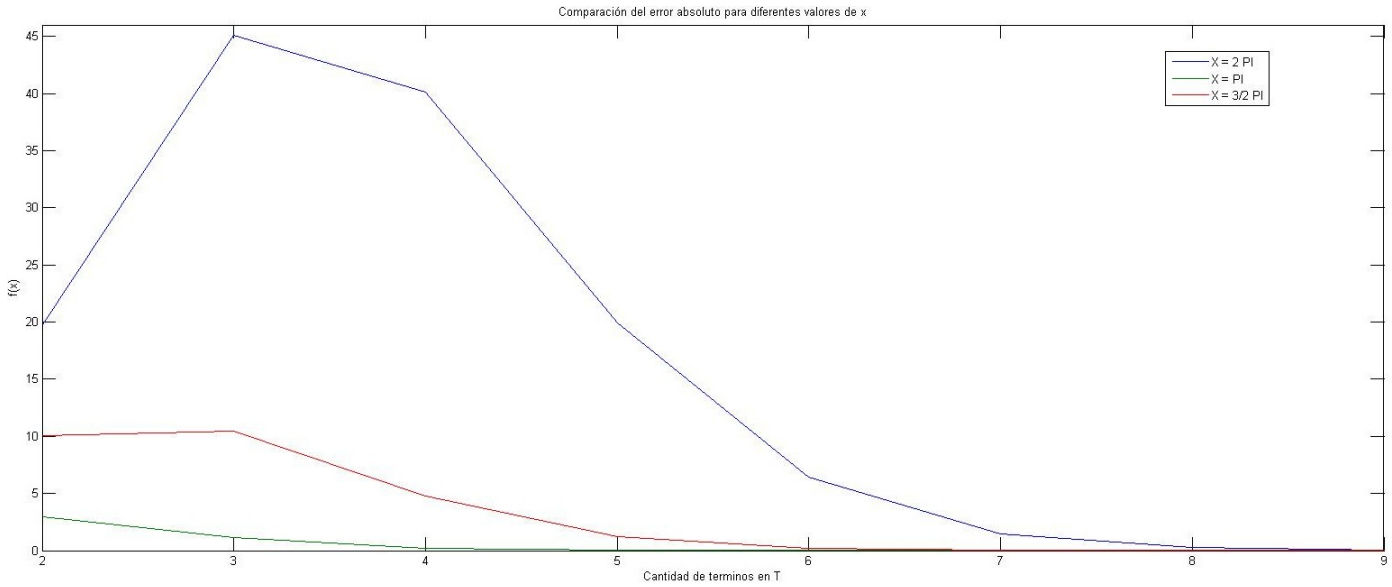


**Gráfico 2.2, comparación del error absoluto para diferentes entradas $x$**

### 3.1.3 Comportamiento de la función para una cantidad de términos fija y un espectro de entrada más grande

Seguidamente quisimos analizar el comportamiento para un T más grande, para esto tomamos T=10 y comparamos los resultados obtenidos entre $f(x)$ y $cos(x)$ para valores entre 0 y 10

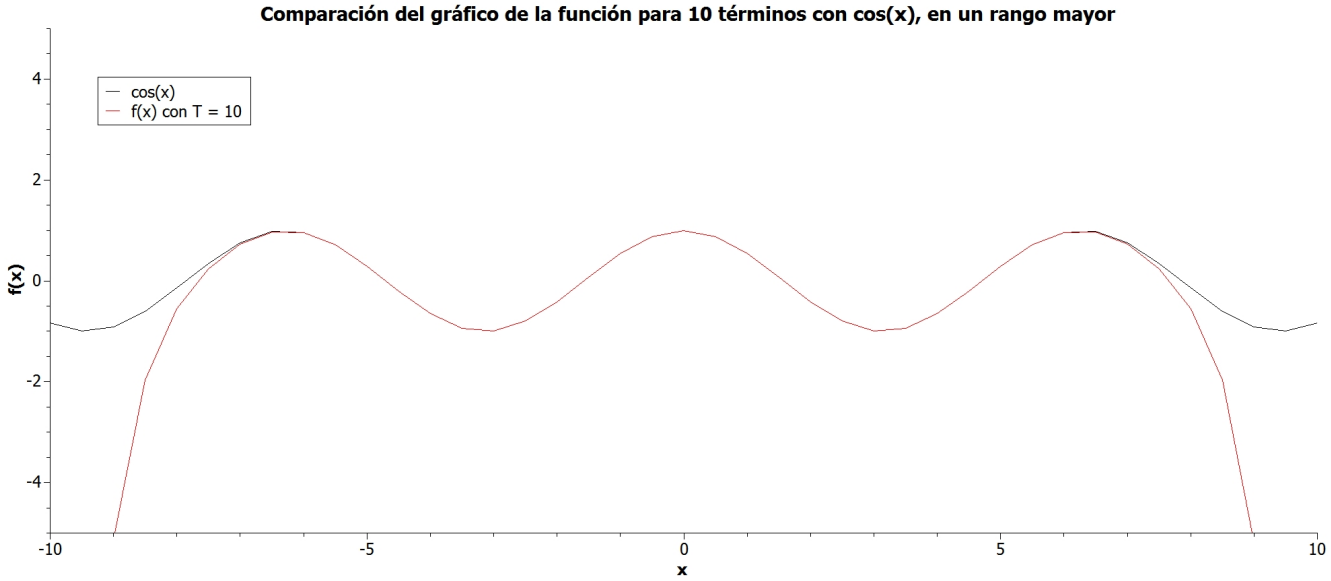


**Gráfico 1.3, comparación de $f(x)$ con un número de términos fijo = 10 contra la curva $\cos(x)$, para un rango mayor de números**

## 3.2 Análisis con tamaños de mantisa variable

Para esta sección, comparamos el comportamiento de la aproximación en función de la cantidad de términos, de la entrada y del tamaño de la mantisa elegida

### 3.2.1 Comparación del comportamiento en función de la cantidad de términos

En primer lugar, decidimos observar el comportamiento en función de la cantidad de términos, para esto, elegimos tres entradas $x$, como ser $\pi$, $2\pi$ y $pi$ y tres tamaños de mantisa diferentes, 2, 15 y 50

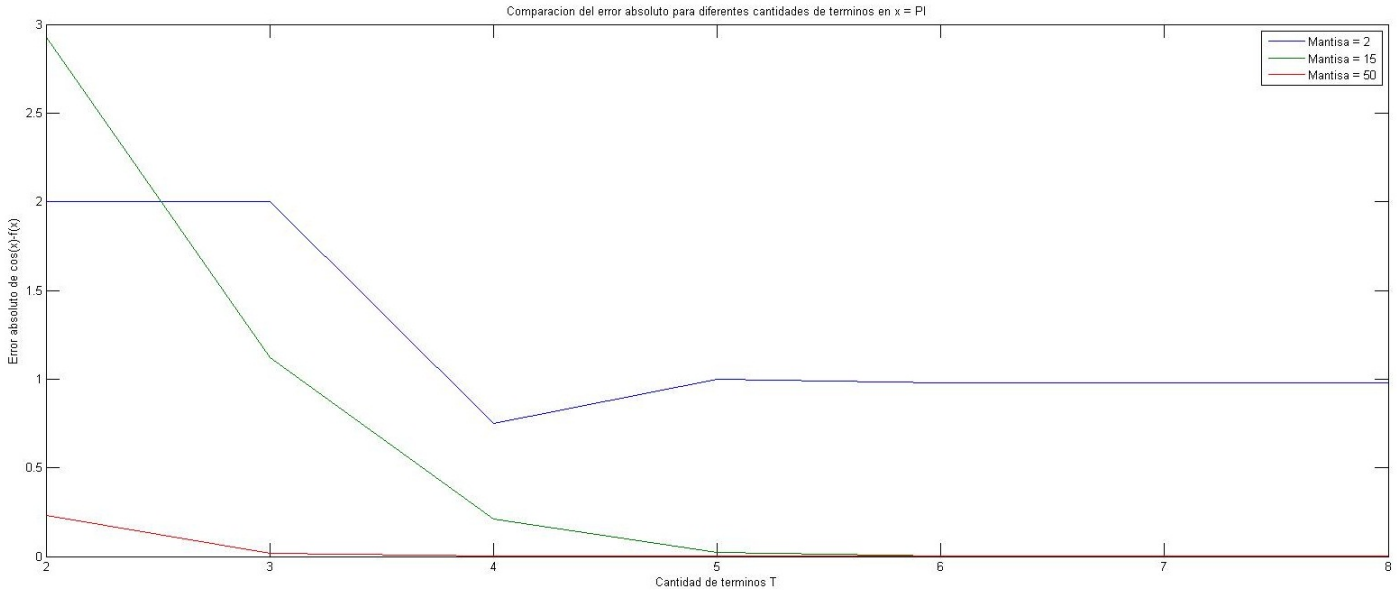


**Gráfico 3.1, comparación del error absoluto para diferentes cantidades de términos, con x=$\pi$ y tres tamaños de mantisa diferentes**

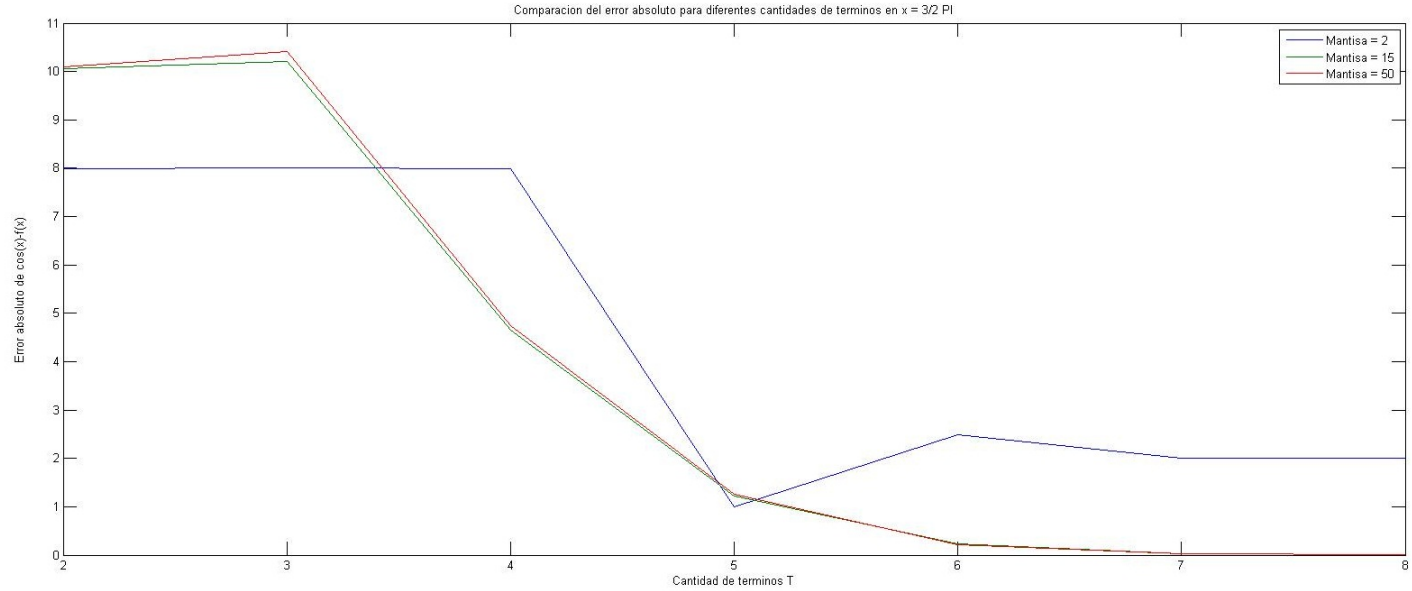


**Gráfico 3.2, comparación del error absoluto para diferentes cantidades de términos, con x=$\frac{3}{2}\pi$ y tres tamaños de mantisa diferentes**

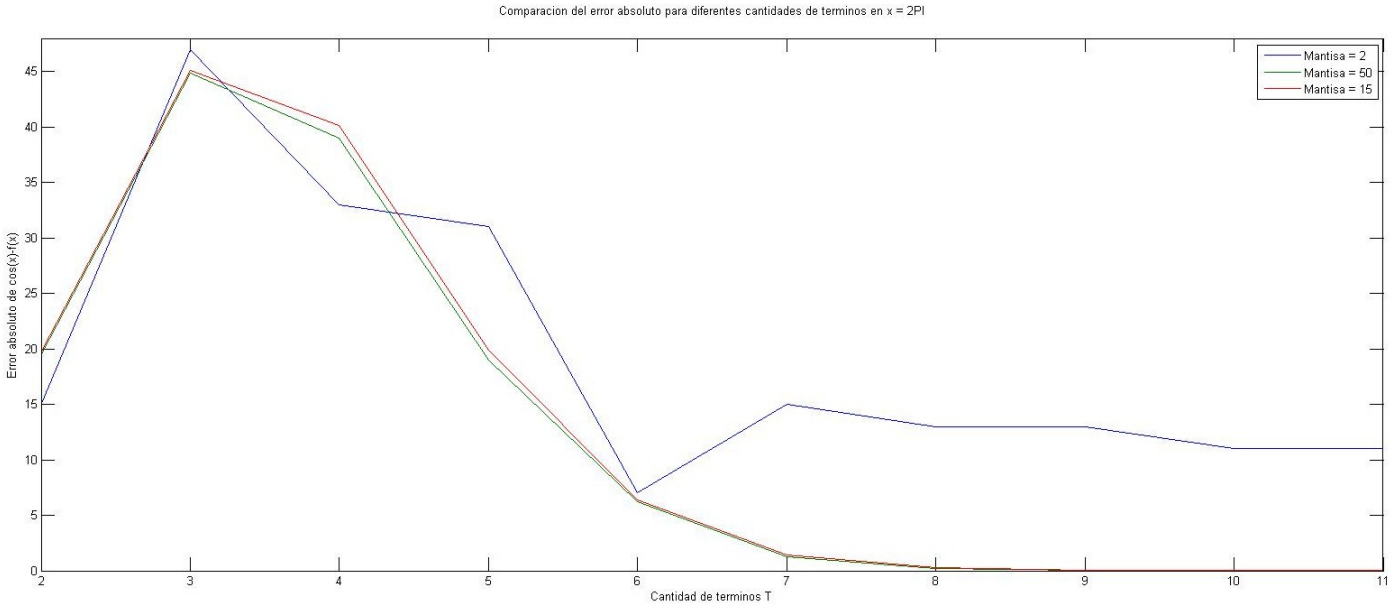


**Gráfico 3.3, comparación del error absoluto para diferentes cantidades de términos, con x=$2\pi$ y tres tamaños de mantisa diferentes**

### 3.2.2 Comparación del comportamiento en función del tamaño de la mantisa

Luego estudiamos el comportamiento en función del tamaño de la mantisa, para esto, elegimos tres entradas $x$, como ser $\frac{1}{2}\pi$, $\pi$, y $\frac{3}{2}\pi$ y tres cantidades de términos diferentes, 2, 4 y 6

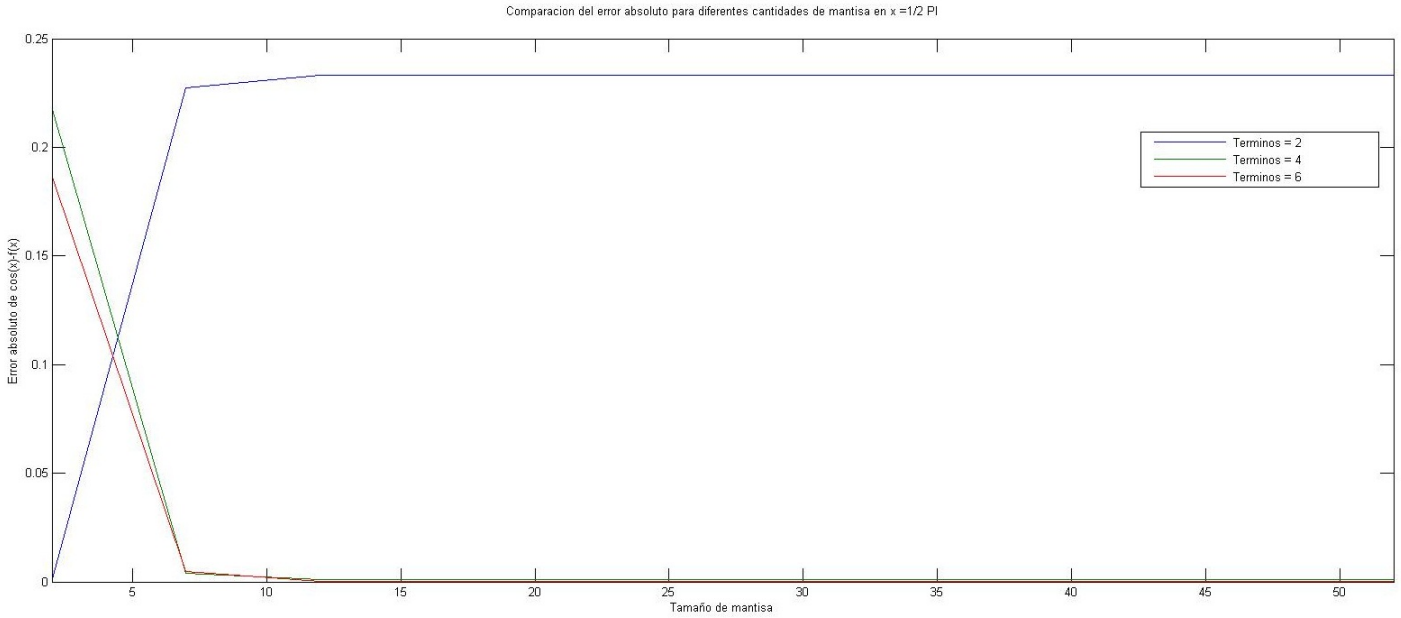


**Gráfico 4.1, comparación del error absoluto para diferentes tamaños de mantisa, con x=$\frac{1}{2}\pi$ y tres cantidades de términos diferentes**

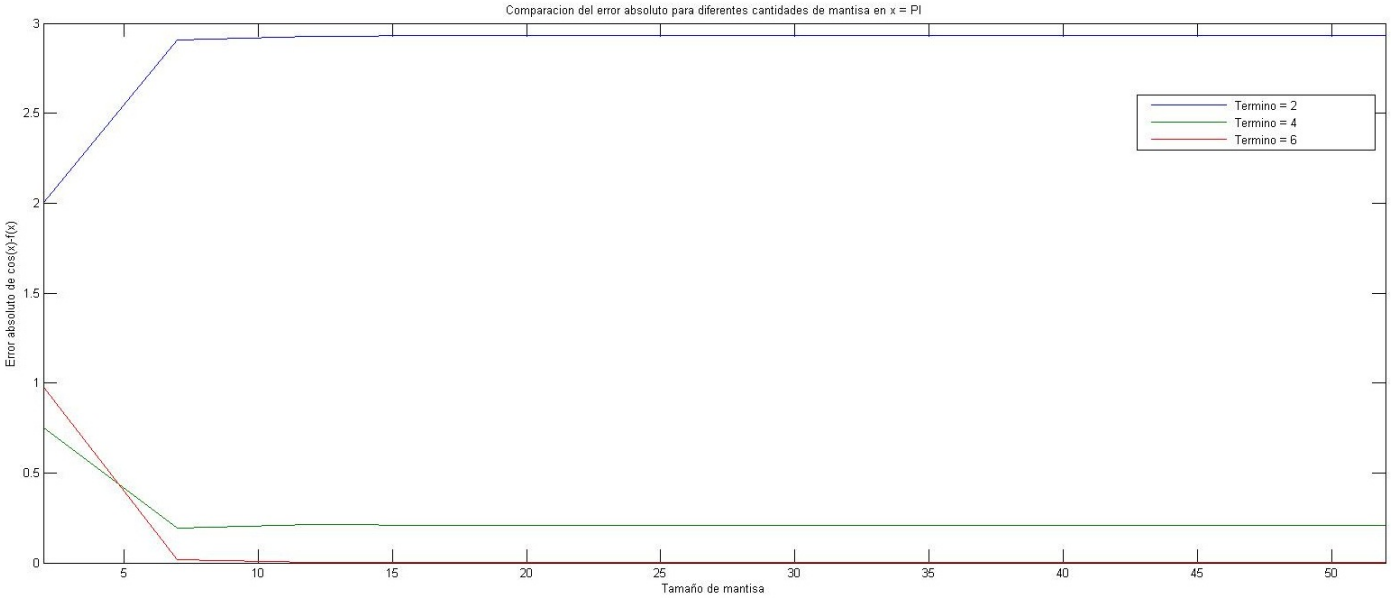


**Gráfico 4.2, comparación del error absoluto para diferentes tamaños de mantisa, con x=$\pi$ y tres cantidades de términos diferentes**

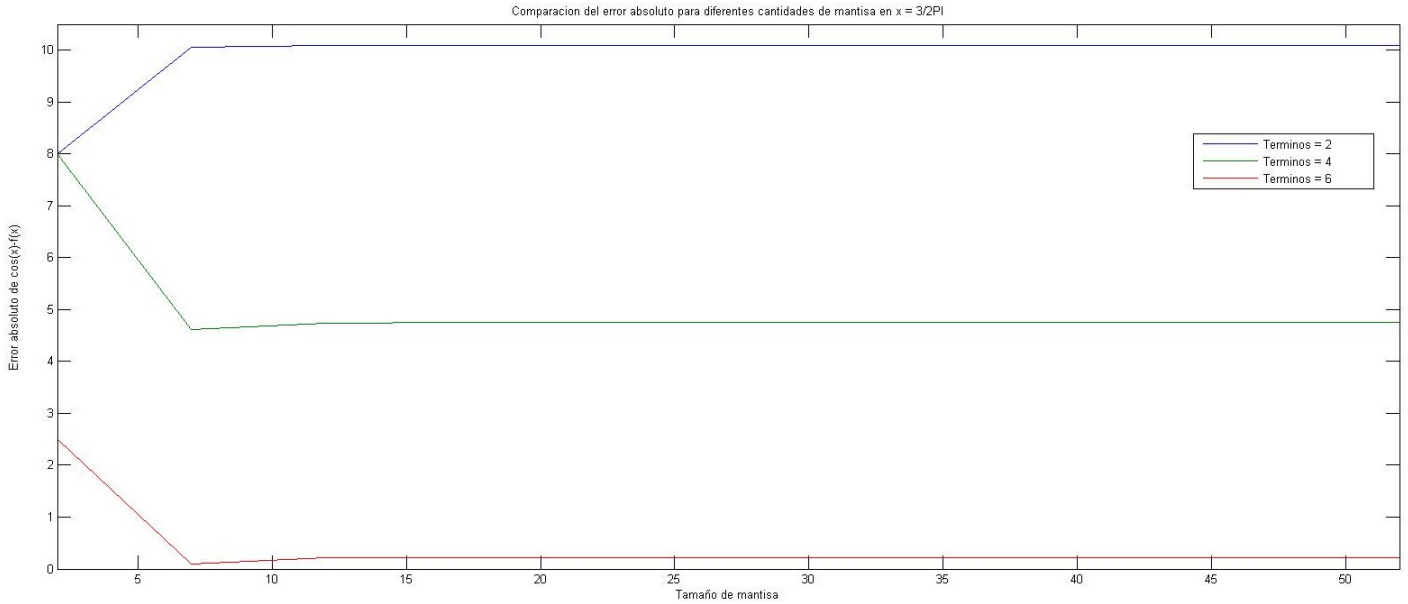


**Gráfico 4.3, comparación del error absoluto para diferentes tamaños de mantisa, con x=$\frac{3}{2}\pi$ y tres cantidades de términos diferentes**

### 3.2.3 Comparación del comportamiento en función del valor de la entrada

También analizamos el comportamiento en función del valor de la entrada, para esto, elegimos tres cantidades de términos diferentes, 2, 4 y 6 y tres tamaños de mantisa diferentes, 2, 15 y 50

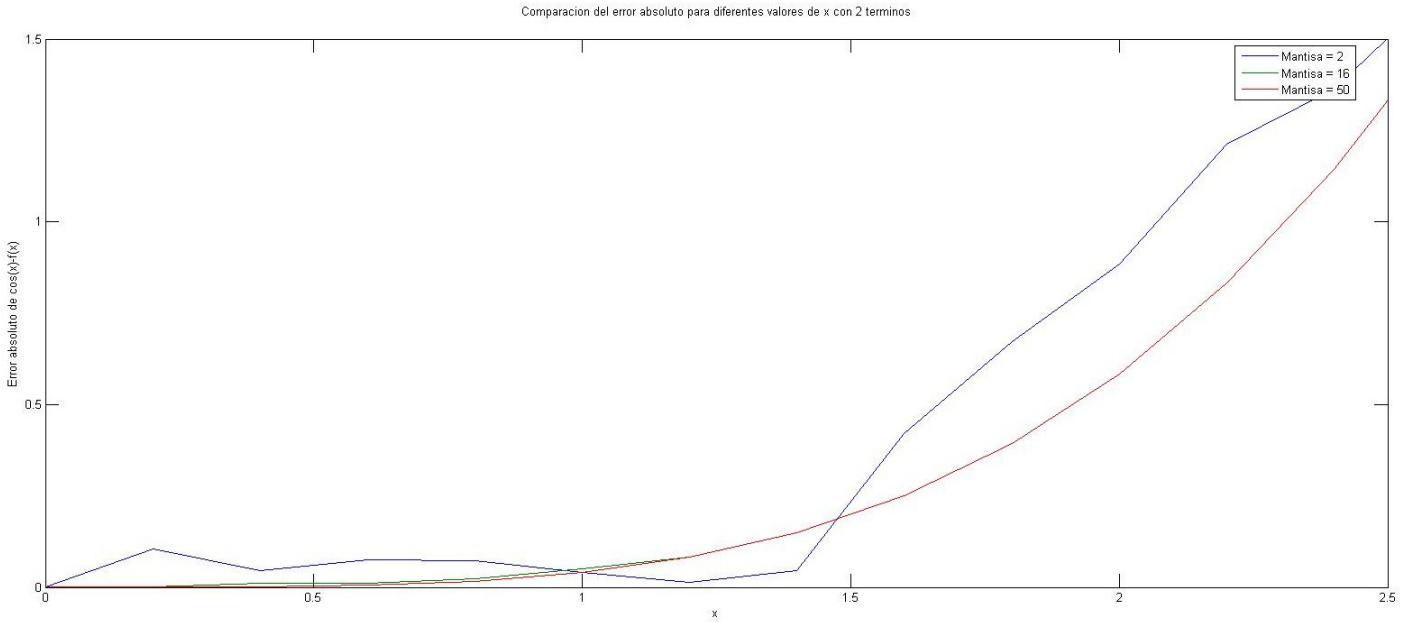


**Gráfico 5.1, comparación del error absoluto para diferentes entradas, con T=2 y tres tamaños de mantisa diferentes**

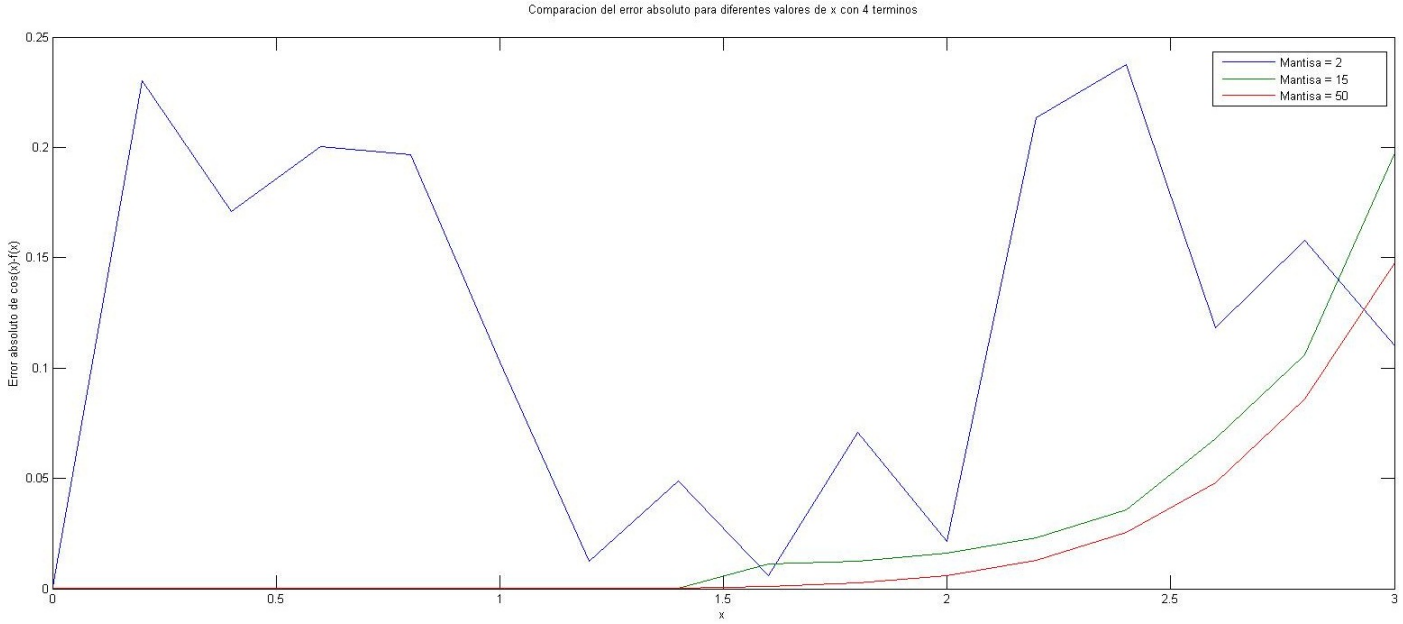


**Gráfico 5.2, comparación del error absoluto para diferentes entradas, con T=4 y tres tamaños de mantisa diferentes**

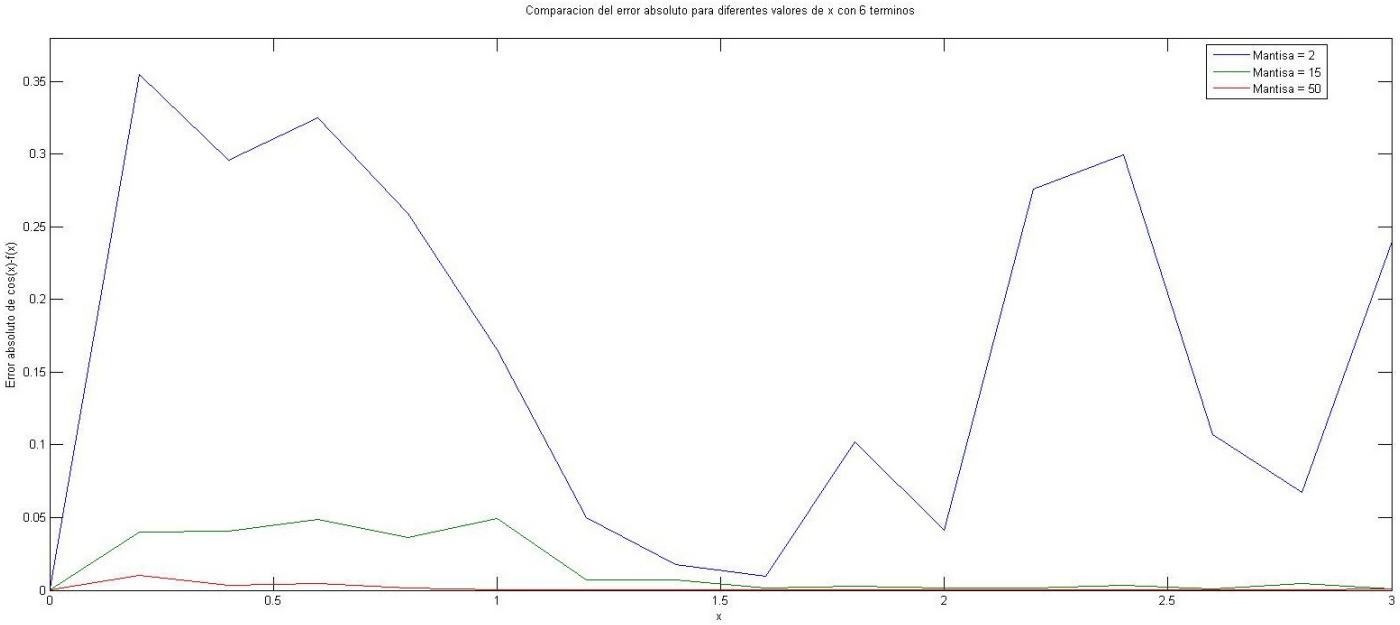


**Gráfico 5.3, comparación del error absoluto para diferentes entradas, con T=6 y tres tamaños de mantisa diferentes**

### 3.2.4 Comparación de errores obtenidos por la sumatoria con los teóricos

Finalmente analizamos como el error teórico hace una curva similar a la curva dada por el error que produce nuestro algorítmo usando un $x$ fijo ($x = 2,4$) y variando la cantidad de términos utilizados de la serie de McLaurin.

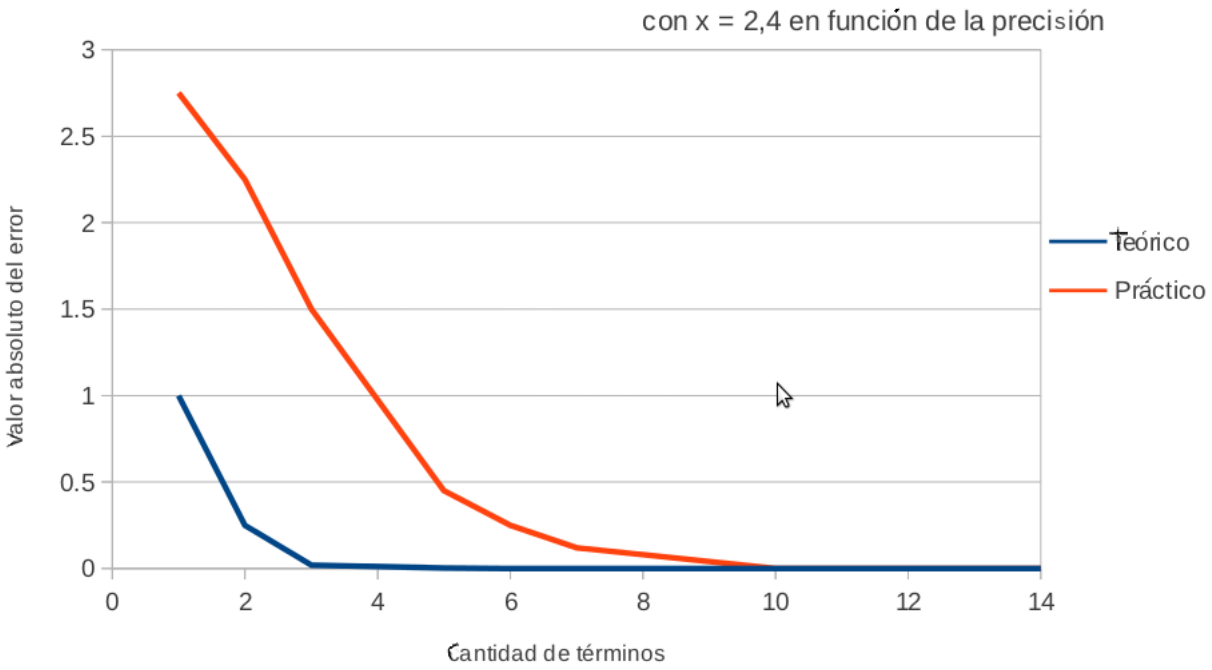


**Gráfico 6.1, comparación de errores teóricos con los errores prácticos a medida que aumenta la cantidad de términos utilizados**

# 4 Discusión

Se puede observar en los gráficos 1.1 y 1.2 cómo una cantidad mayor de términos en la serie significa una mejor aproximación de $cos(x)$, eso significa que cuando aumenta el $n$ la aproximación es mucho mejor, ya con 5 términos el error absoluto está en el orden de $10^{-3}$ para $x < \pi$ y se hace exponencialmente más pequeño a medida que se aumentan los términos.

De la misma manera, en el gráfico 1.3, se puede observar el comportamiento de la función con una cantidad de términos que consideramos suficientemente preciso, podemos ver que mientras cerca del 0 las funciones son idénticas, existe un punto de quiebre desde donde la función aproximación empieza a alejarse cada vez más de la función coseno, es más, $f(x)$ toma valores < -1, lo que es imposible en una función coseno. Analizando más profundamente, ese punto de quiebre se produce entre el 8 y el 8.5, este es el valor para el cual $x^{2i} > 2i!$ para i = 10, por lo tanto, la función empieza a diverger en ese punto. Obviamente, cuanto más grande sea el $n$, este punto se hallará más lejos del 0, pero al ser finito, irremediablemente existirá, por lo tanto, la aproximación se separará de la función coseno tarde o temprano.

En cuanto a cómo afecta el tamaño de la mantisa al error obtenido en el cálculo, la primera conclusión que obtenemos es que, si bien una mantisa muy chica obviamente causa un error, en los gráficos 4.1, 4.2 y 4.3 podemos ver que variar la mantisa más alla de un espectro (en este caso, alrededor de los 10 bits) no produce ningún cambio visible en el resultado obtenido, por lo tanto, el error que se produce al limitar la mantisa es en comparación menor que el que se produce al acotar la cantidad de términos o al alejarse mucho del 0 como entrada de la función. El comportamiento que por momentos parece casi aleatorio del error cuando se usan 2 bits de mantisa lo podemos atribuir al redondeo que está haciendo, puede ser que por redondear mucho se termine disminuyendo un error en vez de arrastrarlo, si bien sabemos que 2 bits de mantisa nunca serían usados seriamente, nos pareció que su comportamiento era interesante para el análisis.

Finalmente analizamos la relación entre el error teórico con el empírico. Cabe decir que cuando tuvimos la matriz de datos obtenidos y de errores teóricos, hicimos la resta para ver en qué valores era mayor el teórico que el empírico. La respuesta fue que en casi todos, salvo algunos casos particulares; en general con $x$ grande y $n$ chico. Fue por esto que decidimos graficar este caso particular donde la cota del error teórica queda por debajo del error empírico.

Igualmente, cabe notar que la comparación de errores teórico y práctico la estamos haciendo calculando el error teórico mediante un script, eso invariablemente trae aparejado los mismos errores que tenemos al aproximar el coseno (redondeo, etc), por lo tanto, siempre la comparación que hagamos no va a ser la perfecta.

# 5 Conclusión

Este TP consistió de dos partes, una teórica, donde analizamos la propagación de los errores teóricos, y una práctica, donde observamos eso en los programas que escribimos. Primero, obtuvimos una cota para el error que surge de cortar la sumatoria en un término, que fue menor al valor absoluto del término siguiente. Al momento de realizar el cálculo del error de propagación de la serie de McLaurin obtuvimos funciones recursivas de la propagación del error, lo que no nos permitía expresar la misma de manera clara. La solucion fue desarrollar esa función hasta lograr la formula cerrada. Al final pudimos observar la magnitud del error que puede producirse al realizar dicha cuenta.

Al analizar la función de forma empírica, comenzamos calculando el coseno con precisión nativa y términos fijos. Lo que primero fue detectado fue la forma en que iban intercambiándose los resultados de la aplicación desarrollada por nosotros, en que según cada término el resultado se pasaba del valor real ya sea por arriba o por abajo. Comparando los valores con el coseno implementado en la clase math notamos que el error siempre disminuía y que llegaba un punto en que la precisión de la impresión del resultado y de la herramienta que usamos para graficar era menor que la magnitud del error cometido en el cálculo, con lo cual consideramos que para ese punto el cálculo es suficientemente preciso.

Otra de las cosas que pudimos observar tiene que ver en la diferencia que tenia la serie de MacLaurin cuando trabaja con valores cercanos al 0 y cuando lo hace con valores alejados del mismo. Observamos que mientras en valores cercanos al 0 con una cantidad de términos $n$ los resultados que se obtienen son muy precisos, en valores mas alejados los resultados obtenidos son poco acertados aún aumentando en gran cantidad la cantidad de terminos. Esto pudimos observar que comenzaba a ocurrir cuando cuando $x^{2i} > 2i!$, concluimos que esto ocurre porque cada termino es mayor al anterior por lo que a medida que vamos incrementandolos en vez de acercarse al resultado esperado se va alejando, un mayor $n$ causa que este punto de quiebre se alcance más lentamente, si $n = \infty$, ese punto de quiebre no existiría nunca y la función sería exactamente igual a $\cos(x)$, pero como $n$ es finito, la aproximación no es exacta.

Por otro lado, teníamos que aproximar la función coseno de la misma manera pero con un tamaño de mantisa variable para el double, este tamaño debía poder ser pasado como parámetro al programa. Podemos ver que se obtiene una menor precisión si se limita mucho la cantidad de dígitos de la mantisa, porque se propagan errores mayores porque $\beta^{1-t}$ es más grande, al ser $t$ más pequeño, esto causa que cada operación tenga un error aparejado más grande y por ende el resultado final diste más del teórico. Pero a su vez, el limitar la mantisa (en un número razonable) trae aparejado menor error que el limitar la cantidad de términos, esto es algo a tener en cuenta, porque si tuviéramos problemas de performance y quisiéramos sacrificar un poco de exactitud en aras de obtener un cálculo más rápido, podemos observar que acortar el tamaño de la mantisa causa menor error absoluto que quitar términos de la serie.

Por último, hemos notado que los valores que obtuvimos como resultados luego de la ejecución de nuestro programa, en muchos casos han logrado errores pequeños entre el resultado de nuestra función y la función a calcular Sin embargo al realizar los mismos cálculos simulando un programa que representa el análisis teórico, concluimos que en la mayor cantidad de corridas realizadas el error obtenido fue mucho más chicos, lo que la hace a esta forma más fiel para obtener mejores resultados.

# 6 Apéndices

## 6.1 Apéndice A

Muchas funciones, por ejemplo trigonométricas o trascendentes, se pueden aproximar por medio de series. Un ejemplo de esto es la serie de Maclaurin (la serie de Taylor alrededor del cero) de la función coseno:

$$f(x) = cosx = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i)!} x^{2i} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ...\forall x \in \mathbb{R}$$

En la práctica estas series se reemplazan por sumatorias con una cantidad finita de términos, por ejemplo:

$$g_n(x) = \sum_{i=0}^{n} \frac{(-1)^i}{(2i)!} x^{2i} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ... \tag{7}$$

A diferencia de la primera ecuacion $g_n$ tiene $n + 1$ terminos en total.

Cuando la cantidad de términos n utilizados en el cálculo es sufientemente grande, la función $g_n(x)$ se parece a la función $f(x)$, es decir:

$$g_n(x) \xrightarrow[n\to\infty]{} f(x)$$

El objetivo del trabajo práctico es analizar el comportamiento numérico de la serie de Maclaurin de la función coseno. En particular, se estudiará la implementación por medio de una aritmética finita de $t$ dígitos, de la sumatoria de $n$ términos de la serie (ecuación 1).

El trabajo práctico consta de dos partes:

**1. Análisis teórico**

a) Analizar el error cometido en función de x, al reemplazar la serie por la sumatoria según la cantidad de términos $n$ y hallar una cota para el mismo. b) Analizar la propagación de errores en el término general de la sumatoria, en función de la precisión aritmética utilizada.

**2. Análisis empírico**

a) Implementar el cálculo de la sumatoria en precisión nativa del compilador y comparar el resultado en función de $x$, obtenido para diferente cantidad de términos $n$, con el valor dado por la función coseno de la biblioteca math.h. Comparar los errores obtenidos con las cotas del punto 1a.

b) Implementar el cálculo de la sumatoria 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 con las cotas de error del punto 1b. Realizar al menos los siguientes experimentos numéricos:

1. Reportar/Graficar el error de considerar la sumatoria con cantidad finita de términos,

a) en función de la variable $x$, para algunos valores fijos de la cantidad de términos $n$ (al menos 5);

b) en función de la cantidad de términos $n$, para algunos valores fijos de $x$ (al menos 5).

2. Reportar/Graficar el error de la propagación de errores debidos al uso de la aritmética finita,

a) en función de la variable $x$, para algunos valores fijos de la cantidad de dígitos $t$ de precisión y de la cantidad de términos $n$ de la sumatoria (al menos 3 de cada uno);

b) en función de la cantidad de dígitos t de precisión en la mantisa, para algunos valores fijos de $x$ y de la cantidad de términos $n$ (al menos 3 de cada uno);

c) en función de la cantidad de términos $n$, para algunos valores fijos de $x$ y de la cantidad de dígitos t de precisión (al menos 3 de cada uno).

3. Comparar los resultados obtenidos teórica y empíricamente.

4. (Opcional) Explorar distintas formas de implementar la fórmula (realizar las cuentas) del 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, qué conclusiones puede extraer?

## 6.2 Apéndice B

### 6.2.1 Código fuente de la serie de McLaurin

Incluimos a continuación el código (simplificado) relevante de las funciones usadas para calcular la serie de McLaurin para $\cos(x)$

```
int terms;
double x;
bool signo = true;
double acum = 0;

cin >> terms >> x;

for (int i = 0; i < terms; i++){
    if (signo){
        acum +=  (potencia (x, (2*i)) / (factorial (2*i)));
    }else{
        acum -=  (potencia (x, (2*i)) / (factorial (2*i)));
    }
    signo = !signo;
    cout << acum << endl;
}
```

```
double factorial (int x){

    double res = 1;

    for (int i = 1; i<=x; i++){
        res *= i;
    }

    return res;
}

double potencia(double num, int exp){

    double res = 1;

    for (int i = 0; i < exp; i++){
        res *= num;
    }

    return res;
}
```

### 6.2.2 Código fuente de la máscara

Incluimos a continuación el código relevante de la inicialización de la máscara y del filtrado por medio de la misma de un double

```
#define MAX_MANTISA 52


    int mantisa_size = atoi(argv[3]);
    unsigned int cantshifts = MAX_MANTISA - mantisa_size;
    upper = 0xFFFFFFFF;
    lower = 0xFFFFFFFF;

    if (cantshifts > 32){
        for (unsigned int i = 0; i < max (cantshifts - 32, (unsigned int)0); i++){
            upper = upper << 1;
        }
    }

    for(unsigned int i = 0; i < min ((unsigned int)32, cantshifts); i++){
        lower = lower << 1;
    }


double domask (double x){
```

```
    int* b = (int*) &x;

    *b = *b & lower;
    *(b+1) = *(b+1) & upper;

    return *((double *) b);
}
```

# 7 Referencias

1) Germund Dahlquist & Åke Björck, Numerical Methods, 1974. Theorem 2.3.1