Clase 4: errores de redondeo y truncamiento

Hernán Mella

EIE PUCV

11 de septiembre de 2023

1. Introducción a Python
   - ¿Qué entendemos por error?
   - Error de redondeo
   - Error de truncamiento
     - Series de Taylor

1. Introducción a Python
   - ¿Qué entendemos por error?
   - Error de redondeo
   - Error de truncamiento
     - Series de Taylor

Objetivos de la clase

- Aprender la diferencia entre exactitud y precisión
- Aprender a cuantificar errores y sus fuentes
- Aprender cómo usar la estimación de errores para terminar un cálculo iterativo
- Reconocer que los errores de truncamiento ocurren cuando formulaciones matemáticas exactas son representadas con aproximaciones
- Conocer cómo usar series de Taylor para estimar errores de truncamiento

Recordemos el problema de caída libre

Anteriormente vimos que para resolver la ecuación diferencial tuvimos que aproximar la velocidad de caída como:

$$\frac{dv}{dt} \approx \frac{\Delta v}{\Delta t} = \frac{v(t_{k+1}) - v(t_k)}{t_{k+1} - t_k}$$

Lo que resulta en una aproximación que contiene errores.

El computador también comete errores representando los números.

Por lo tanto...

¿Cómo lidiamos con la incertidumbre de ambos errores?

Recordemos el problema de caída libre

Anteriormente vimos que para resolver la ecuación diferencial tuvimos que aproximar la velocidad de caída como:

$$\frac{dv}{dt} \approx \frac{\Delta v}{\Delta t} = \frac{v(t_{k+1}) - v(t_k)}{t_{k+1} - t_k}$$

Lo que resulta en una aproximación que contiene errores.
El computador también comete errores representando los números.

Por lo tanto...

¿Cómo lidiamos con la incertidumbre de ambos errores?

Recordemos el problema de caída libre

Anteriormente vimos que para resolver la ecuación diferencial tuvimos que aproximar la velocidad de caída como:

$$\frac{dv}{dt} \approx \frac{\Delta v}{\Delta t} = \frac{v(t_{k+1}) - v(t_k)}{t_{k+1} - t_k}$$

Lo que resulta en una aproximación que contiene errores.
El computador también comete errores representando los números.

Por lo tanto...

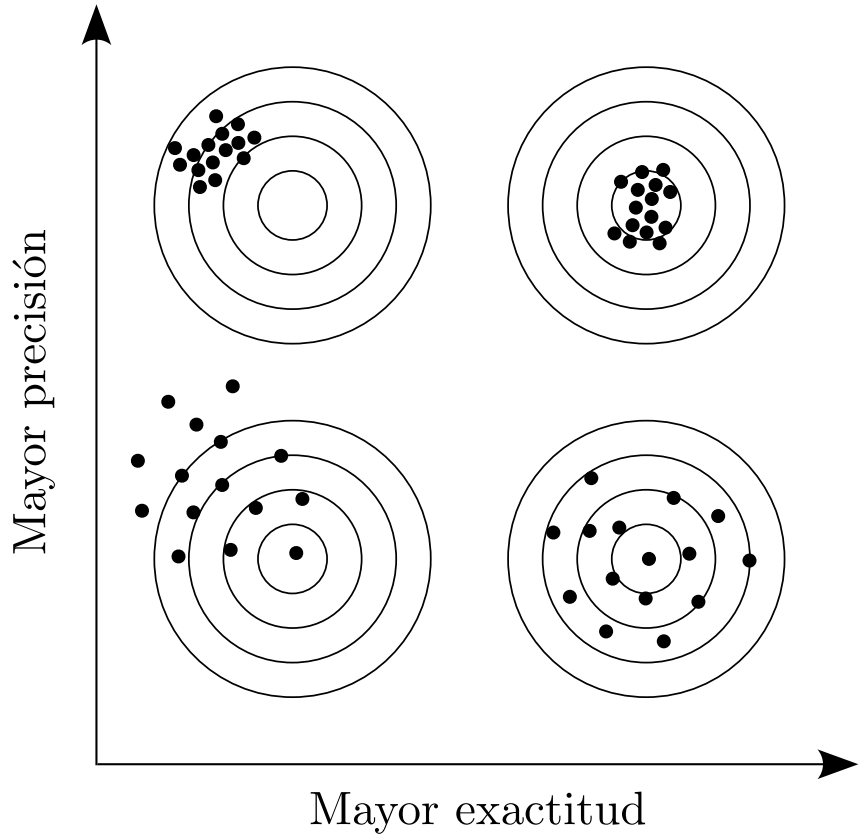
¿Cómo lidiamos con la incertidumbre de ambos errores?

Entendiendo la diferencia entre exactitud y precisión

Exactitud se refiere a qué tan cercano se encuentra un valor estimado/medido de su valor real

Precisión se refiere a qué tan cerca están las estimaciones/mediciones entre ellas

Una métrica de error se utiliza para representar la inexactitud e imprecisión de nuestras predicciones

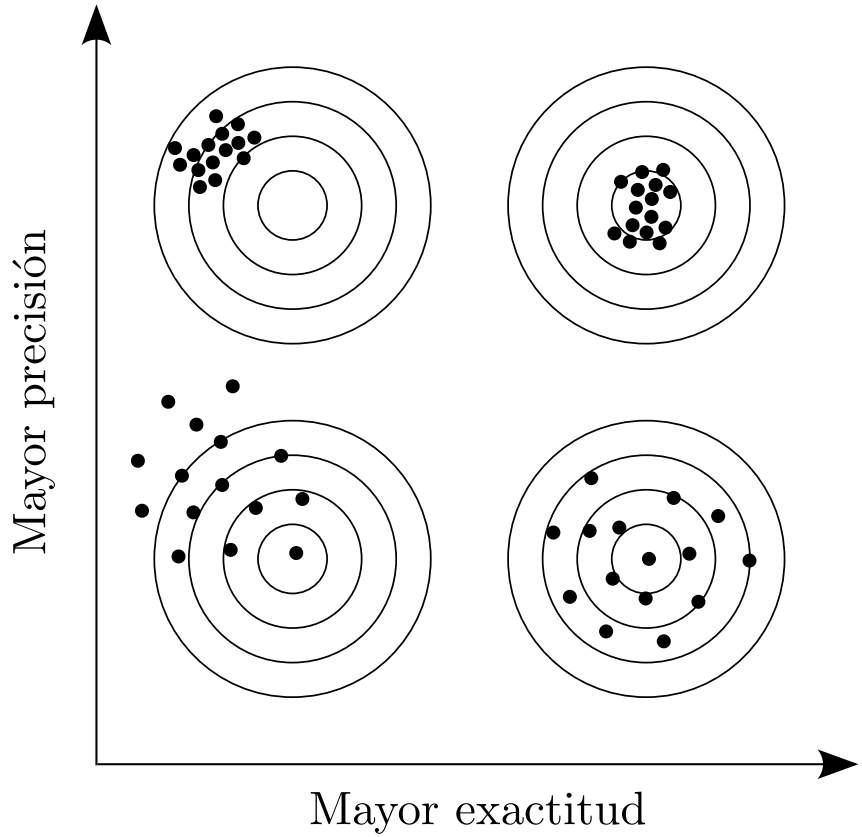


Entendiendo la diferencia entre exactitud y precisión

Exactitud se refiere a qué tan cercano se encuentra un valor estimado/medido de su valor real

Precisión se refiere a qué tan cerca están las estimaciones/mediciones entre ellas

Una métrica de error se utiliza para representar la inexactitud e imprecisión de nuestras predicciones

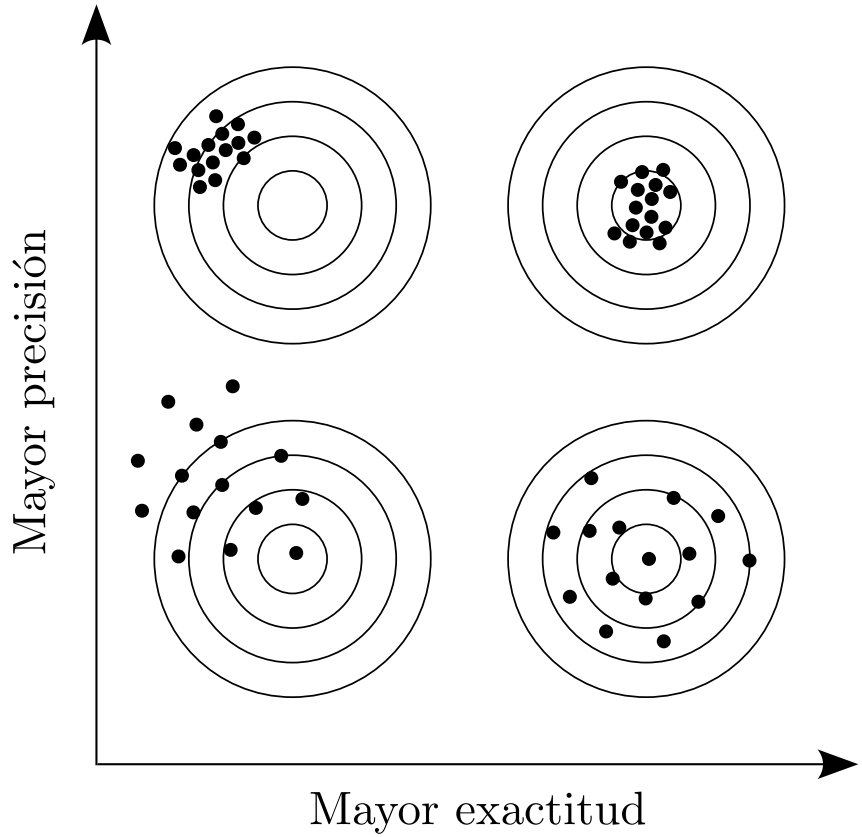


Entendiendo la diferencia entre exactitud y precisión

Exactitud se refiere a qué tan cercano se encuentra un valor estimado/medido de su valor real

Precisión se refiere a qué tan cerca están las estimaciones/mediciones entre ellas

Una métrica de error se utiliza para representar la inexactitud e imprecisión de nuestras predicciones



1. Introducción a Python
   - ¿Qué entendemos por error?
   - Error de redondeo
   - Error de truncamiento
     - Series de Taylor

Definiciones de error

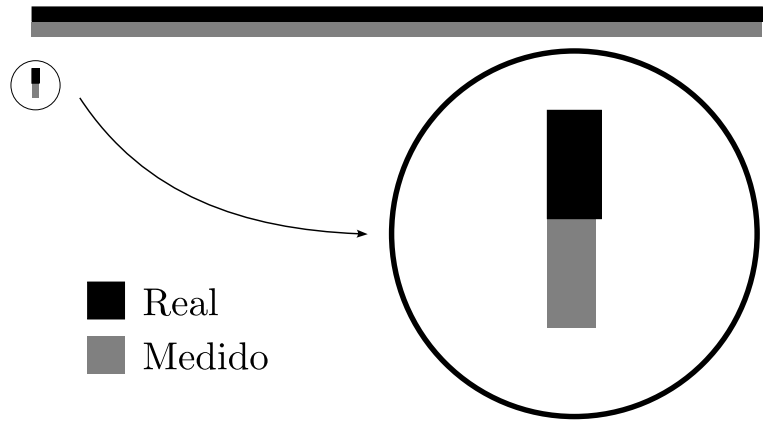
Error real[1]:

$$E_t = \text{valor real} - \text{aproximación}$$

Error relativo[2]:

$$\varepsilon_t = \frac{\text{valor real} - \text{aproximación}}{\text{valor real}}$$

Ejemplo: se le da la tarea de medir el ancho de la sala y el alto de una taza de café y ud. consigue las medidas de 999 y 9 cm. Las medidas reales son 1000 y 10 cm respectivamente. ¿Cuál es el error real y relativo entre las medidas?



[1] Cuando se expresa con valor absoluto se conoce como *error absoluto*

[2] También se puede expresar como porcentaje

Definiciones de error

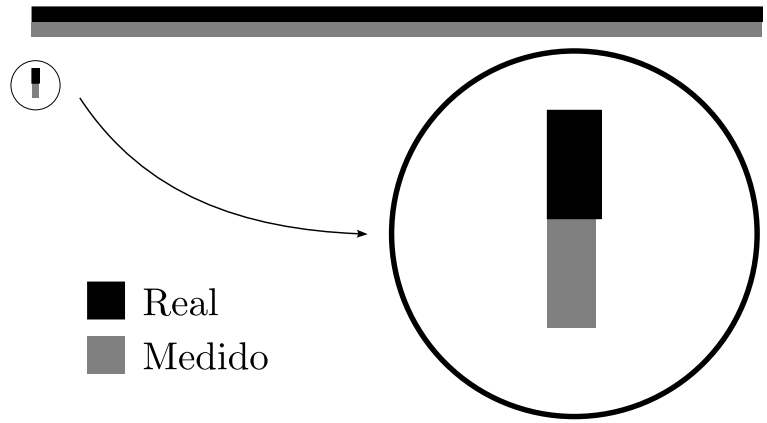
Error real[1]:

$$E_t = \text{valor real} - \text{aproximación}$$

Error relativo[2]:

$$\varepsilon_t = \frac{\text{valor real} - \text{aproximación}}{\text{valor real}}$$

Ejemplo: se le da la tarea de medir el ancho de la sala y el alto de una taza de café y ud. consigue las medidas de 999 y 9 cm. Las medidas reales son 1000 y 10 cm respectivamente. ¿Cuál es el error real y relativo entre las medidas?



[1] Cuando se expresa con valor absoluto se conoce como *error absoluto*

[2] También se puede expresar como porcentaje

Definiciones de error

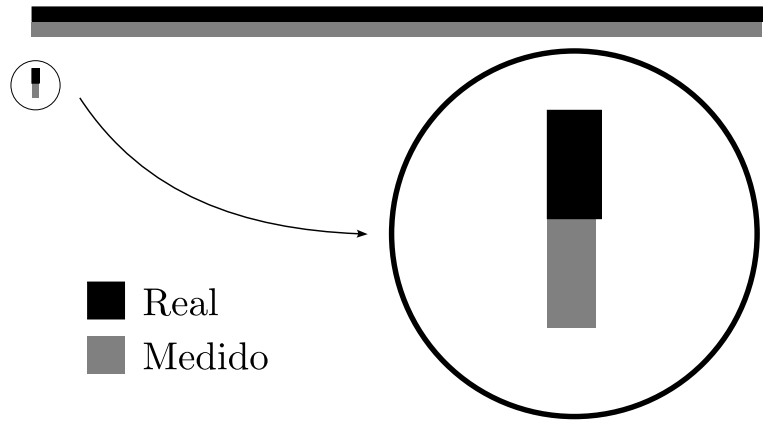
Error real[1]:

$$E_t = \text{valor real} - \text{aproximación}$$

Error relativo[2]:

$$\varepsilon_t = \frac{\text{valor real} - \text{aproximación}}{\text{valor real}}$$

Ejemplo: se le da la tarea de medir el ancho de la sala y el alto de una taza de café y ud. consigue las medidas de 999 y 9 cm. Las medidas reales son 1000 y 10 cm respectivamente. ¿Cuál es el error real y relativo entre las medidas?



[1] Cuando se expresa con valor absoluto se conoce como *error absoluto*

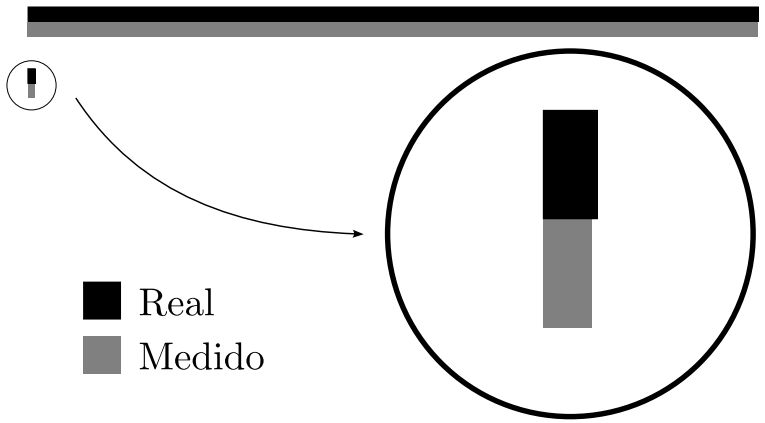
[2] También se puede expresar como porcentaje

Datos:

$$\text{valor real} = \{1000, 10\} \text{ cm}$$
$$\text{aproximación} = \{999, 9\} \text{ cm}$$

Primer caso

$$E_t = 1000 - 999 = 1 \text{ cm}$$
$$\varepsilon_r = \frac{1000 - 999}{1000} = 0.001 \quad (0.1\,\%)$$



Segundo caso

$$E_t = 10 - 9 = 1 \text{ cm}$$
$$\varepsilon_r = \frac{10 - 9}{10} = 0.1 \quad (10\,\%)$$

Cuando no conocemos el valor real, ¿cómo estimamos el error?

Se puede estimar el error de la siguiente forma:

$$\varepsilon_a = \frac{\text{error estimado}}{\text{estimación}}$$

Existen métodos que usan iteraciones para mejorar la estimación. En ese caso:

$$\varepsilon_a = \frac{\text{estimación actual} - \text{estimación previa}}{\text{estimación actual}}$$

Criterio de parada en métodos iterativos

Las iteraciones se repiten hasta que el error se encuentra por debajo de cierto umbral:

$$|\varepsilon_a| < \varepsilon_s, \qquad \text{para } \varepsilon_s \in \mathbb{R}^+ \text{ fijo}$$

Dicho umbral se puede relacionar con la cantidad de cifras significativas a través de:

$$\varepsilon_s = 0.5 \times 10^{2-n} \qquad \text{(Scarborough, 1966)}$$

Cuando no conocemos el valor real, ¿cómo estimamos el error?

Se puede estimar el error de la siguiente forma:

$$\varepsilon_a = \frac{\text{error estimado}}{\text{estimación}}$$

Existen métodos que usan iteraciones para mejorar la estimación. En ese caso:

$$\varepsilon_a = \frac{\text{estimación actual} - \text{estimación previa}}{\text{estimación actual}}$$

Criterio de parada en métodos iterativos

Las iteraciones se repiten hasta que el error se encuentra por debajo de cierto umbral:

$$|\varepsilon_a| < \varepsilon_s, \qquad \text{para } \varepsilon_s \in \mathbb{R}^+ \text{ fijo}$$

Dicho umbral se puede relacionar con la cantidad de cifras significativas a través de:

$$\varepsilon_s = 0.5 \times 10^{2-n} \qquad \text{(Scarborough, 1966)}$$

Calculemos una aproximación de la función exponencial

Ejemplo 1: la función exponencial se puede aproximar usando la siguiente serie de Mclaurin:

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

Mientras más términos se añaden a la serie, mejor es la aproximación. Partiendo de la aproximación $e^x \approx 1$, agregue términos de la serie para estimar el valor de $e^{0.5}$. Calcule el error aproximado para cada una de las estimaciones[1].
Defina un umbral de parada que asegure una buena aproximación de al menos 3 cifras significativas. Deje de añadir términos a la serie una vez que se cumpla dicho criterio.

[1] A modo de referencia, $e^{0.5} = 1.648721\ldots$

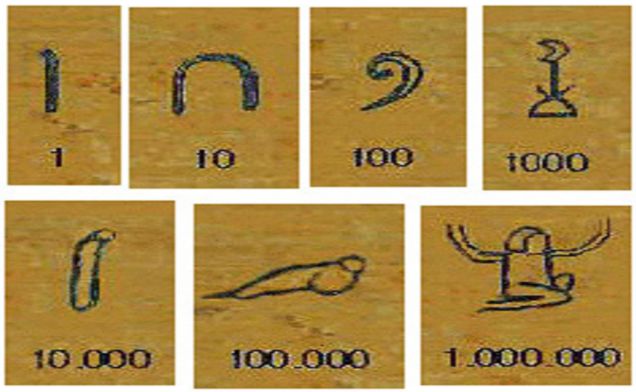
1. Introducción a Python
   - ¿Qué entendemos por error?
   - Error de redondeo
   - Error de truncamiento
     - Series de Taylor

Sistema de representación en base-10

Cotidianamente representamos los número en un sistema decimal (o base-10)

La base es el número que se usa como referencia para construir el sistema

Este sistema usa 10 dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) para representar números



Sistema de notación egipcio en base-10

Notación posicional

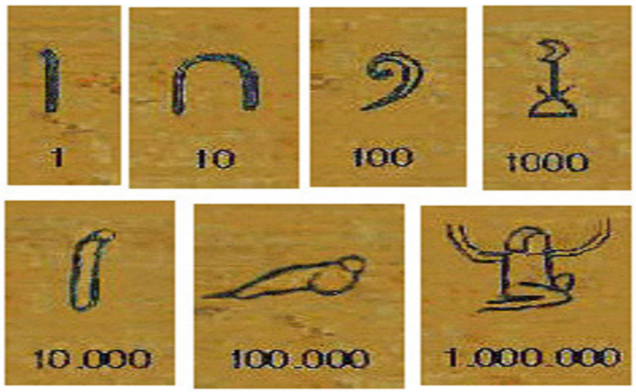
Para representar grandes números se utiliza una combinación con el valor posicional especificando su magnitud

Sistema de representación en base-10

Cotidianamente representamos los número en un sistema decimal (o base-10)

La base es el número que se usa como referencia para construir el sistema

Este sistema usa 10 dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) para representar números



Sistema de notación egipcio en base-10

Notación posicional

Para representar grandes números se utiliza una combinación con el valor posicional especificando su magnitud

Sistema de representación en base-10

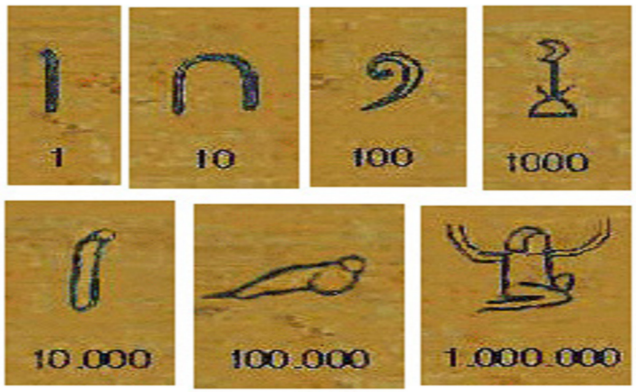
Cotidianamente representamos los número en un sistema decimal (o base-10)

La base es el número que se usa como referencia para construir el sistema

Este sistema usa 10 dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) para representar números

Notación posicional

Para representar grandes números se utiliza una combinación con el valor posicional especificando su magnitud



Sistema de notación egipcio en base-10

Representación de números en computadores: sistema base-2

Un computador se limita a dos estados: 0 y 1

Por esto se utiliza el sistema binario en computadores

Usa notación posicional para representar números

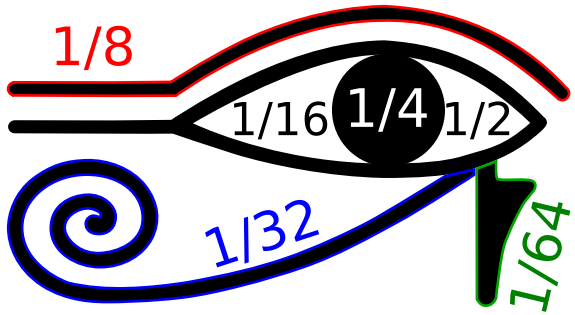
Ejemplos

El número 179 en binario se representa como:

$$10110011 = 2^7 + 2^5 + 2^4 + 2^1 + 2^0$$

El número binario 1101.1 en decimal representa:

$$2^4 + 2^3 + 2^2 + 2^0 + 2^{-1} = 29.5$$



Sistema de notación egipcio en base-2

Representación de números en computadores: sistema base-2

Un computador se limita a dos estados: 0 y 1

Por esto se utiliza el sistema binario en computadores

Usa notación posicional para representar números

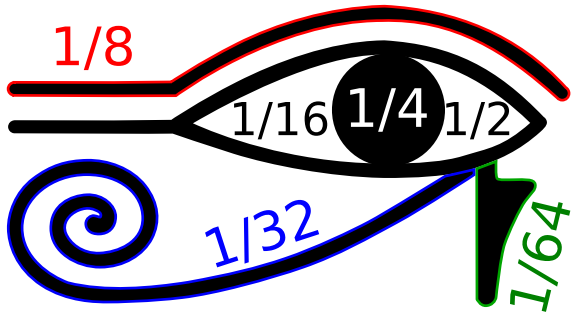
Ejemplos

El número 179 en binario se representa como:

$$10110011 = 2^7 + 2^5 + 2^4 + 2^1 + 2^0$$

El número binario 1101.1 en decimal representa:

$$2^4 + 2^3 + 2^2 + 2^0 + 2^{-1} = 29.5$$



Sistema de notación egipcio en base-2

Representación de números en computadores: sistema base-2

Un computador se limita a dos estados: 0 y 1

Por esto se utiliza el sistema binario en computadores

Usa notación posicional para representar números

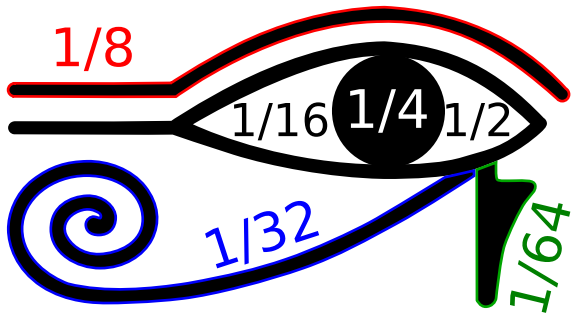
Ejemplos

El número 179 en binario se representa como:

$$10110011 = 2^7 + 2^5 + 2^4 + 2^1 + 2^0$$

El número binario 1101.1 en decimal representa:

$$2^4 + 2^3 + 2^2 + 2^0 + 2^{-1} = 29.5$$



Sistema de notación egipcio en base-2

Representación de números en computadores: sistema base-2

Un computador se limita a dos estados: 0 y 1

Por esto se utiliza el sistema binario en computadores

Usa notación posicional para representar números

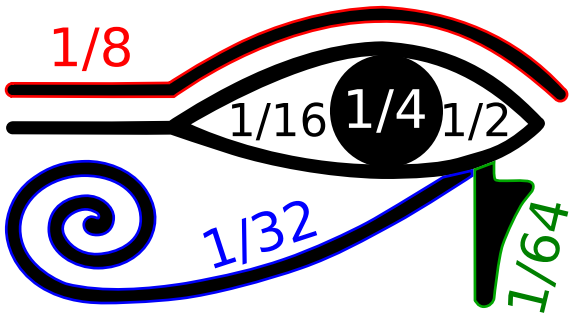
Ejemplos

El número 179 en binario se representa como:

$$10110011 = 2^7 + 2^5 + 2^4 + 2^1 + 2^0$$

El número binario 1101.1 en decimal representa:

$$2^4 + 2^3 + 2^2 + 2^0 + 2^{-1} = 29.5$$



Sistema de notación egipcio en base-2

Representación de números en computadores: sistema base-2

Un computador se limita a dos estados: 0 y 1

Por esto se utiliza el sistema binario en computadores

Usa notación posicional para representar números

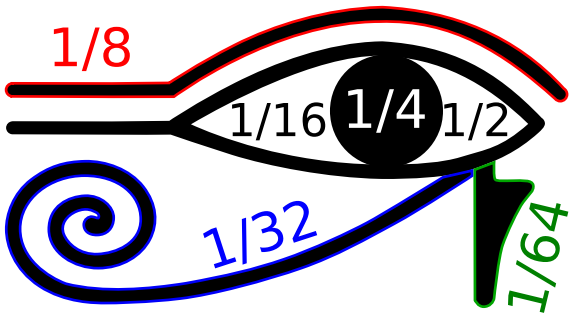
Ejemplos

El número 179 en binario se representa como:

$$10110011 = 2^7 + 2^5 + 2^4 + 2^1 + 2^0$$

El número binario 1101.1 en decimal representa:

$$2^4 + 2^3 + 2^2 + 2^0 + 2^{-1} = 29.5$$



Sistema de notación egipcio en base-2

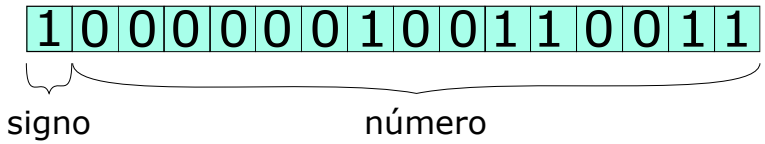
Sistema de representación de números enteros

Una forma de representar enteros en computadores es el método de *signed-magnitude*

En un número binario se utiliza el primer bit para indicar el signo (0: positivo y 1: negativo)

Los demás bits se usan para almacenar el número.

Por ejemplo, la representación binaria de 16 bits del número 179 sería:



El entero más grande representable con 16 bits es 32767 ($2^{15} - 1$) y el más pequeño -32768 (-2^{15}) (¿?)

En la práctica el método utilizado para almacenar enteros en un computador se llama *2s complement* (complemento a dos)

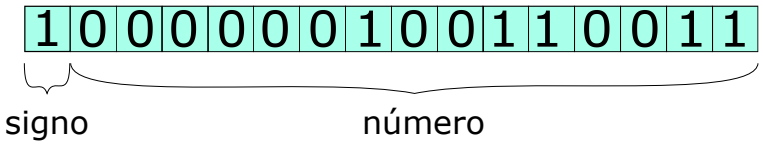
Sistema de representación de números enteros

Una forma de representar enteros en computadores es el método de *signed-magnitude*

En un número binario se utiliza el primer bit para indicar el signo (0: positivo y 1: negativo)

Los demás bits se usan para almacenar el número.

Por ejemplo, la representación binaria de 16 bits del número 179 sería:



El entero más grande representable con 16 bits es 32767 ($2^{15} - 1$) y el más pequeño -32768 (-2^{15}) (¿?)

En la práctica el método utilizado para almacenar enteros en un computador se llama *2s complement* (complemento a dos)

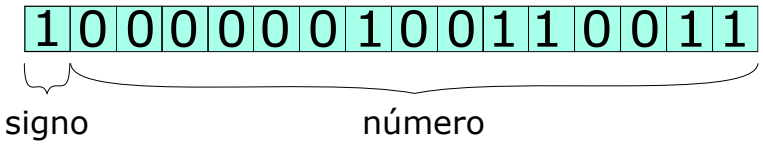
Sistema de representación de números enteros

Una forma de representar enteros en computadores es el método de *signed-magnitude*

En un número binario se utiliza el primer bit para indicar el signo (0: positivo y 1: negativo)

Los demás bits se usan para almacenar el número.

Por ejemplo, la representación binaria de 16 bits del número 179 sería:



El entero más grande representable con 16 bits es 32767 ($2^{15} - 1$) y el más pequeño -32768 (-2^{15}) (¿?)

En la práctica el método utilizado para almacenar enteros en un computador se llama *2s complement* (complemento a dos)

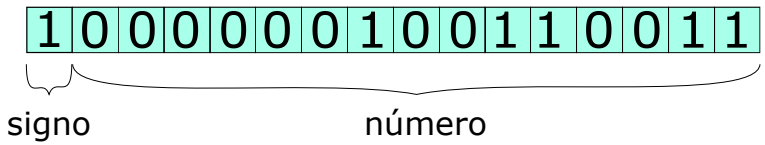
Sistema de representación de números enteros

Una forma de representar enteros en computadores es el método de *signed-magnitude*

En un número binario se utiliza el primer bit para indicar el signo (0: positivo y 1: negativo)

Los demás bits se usan para almacenar el número.

Por ejemplo, la representación binaria de 16 bits del número 179 sería:



El entero más grande representable con 16 bits es 32767 ($2^{15} - 1$) y el más pequeño -32768 (-2^{15}) (¿?)

En la práctica el método utilizado para almacenar enteros en un computador se llama *2s complement* (complemento a dos)

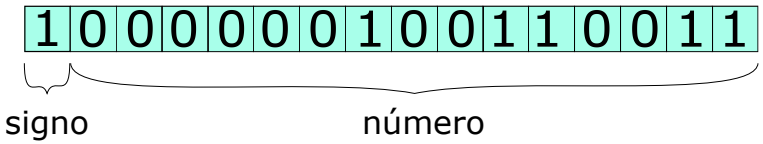
Sistema de representación de números enteros

Una forma de representar enteros en computadores es el método de *signed-magnitude*

En un número binario se utiliza el primer bit para indicar el signo (0: positivo y 1: negativo)

Los demás bits se usan para almacenar el número.

Por ejemplo, la representación binaria de 16 bits del número 179 sería:



El entero más grande representable con 16 bits es 32767 ($2^{15} - 1$) y el más pequeño -32768 (-2^{15}) (¿?)

En la práctica el método utilizado para almacenar enteros en un computador se llama *2s complement* (complemento a dos)

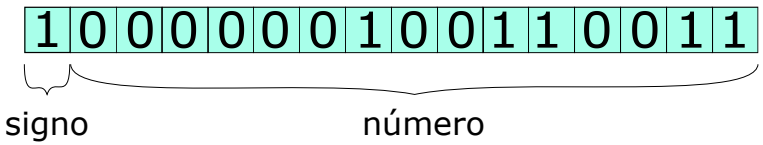
Sistema de representación de números enteros

Una forma de representar enteros en computadores es el método de *signed-magnitude*

En un número binario se utiliza el primer bit para indicar el signo (0: positivo y 1: negativo)

Los demás bits se usan para almacenar el número.

Por ejemplo, la representación binaria de 16 bits del número 179 sería:



El entero más grande representable con 16 bits es 32767 ($2^{15} - 1$) y el más pequeño -32768 (-2^{15}) (¿?)

En la práctica el método utilizado para almacenar enteros en un computador se llama *2s complement* (complemento a dos)

Sistema de representación de punto-flotante

En el formato punto-flotante el número se representa como

$$\pm c \times b^e$$

donde c es el coeficiente, b es la base del sistema numérico y e el exponente.

Como regla general, para expresar un número en punto flotante primero debe ser normalizado.

Un ejemplo en base-10

$$0.005678 = 5.678 \times 10^{-3}$$

Sistema de representación de punto-flotante

En el formato punto-flotante el número se representa como

$$\pm c \times b^e$$

donde c es el coeficiente, b es la base del sistema numérico y e el exponente.

Como regla general, para expresar un número en punto flotante primero debe ser normalizado.

Un ejemplo en base-10

$$0.005678 = 5.678 \times 10^{-3}$$

Sistema de representación de punto-flotante

En el formato punto-flotante el número se representa como

$$\pm c \times b^e$$

donde c es el coeficiente, b es la base del sistema numérico y e el exponente.

Como regla general, para expresar un número en punto flotante primero debe ser normalizado.

Un ejemplo en base-10

$$0.005678 = 5.678 \times 10^{-3}$$

Antes de pasar al sistema punto-flotante binario, revisemos un ejemplo

Supongamos que tenemos un computador en base-10 hipotético con una representación de punto flotante de 5 bits. Asuma que un digito se usa para el signo, dos para el exponente y dos para el coeficiente. Por simplicidad asuma que uno de los digitos del exponente se usa para almacenar su signo. Responda las siguientes preguntas:

1. ¿Cuáles son los números más grandes y más pequeños que se pueden representar en este sistema?
2. ¿Que pasaría si aumentamos la cantidad bits del exponente?
3. ¿Que pasaría si aumentamos la cantidad de bits del coeficiente?

Asuma que el número puede ser representado de la siguiente forma:

$$s_1 c_1 . c_2 \times 10^{s_0 e_0}$$

Antes de pasar al sistema punto-flotante binario, revisemos un ejemplo

1. ¿Cuáles son los números más grandes y más pequeños que se pueden representar en este sistema?

$$+1.0 \times 10^{-9}, \quad +9.9 \times 10^{+9}, \quad -9.9 \times 10^{+9}, \quad -1.0 \times 10^{-9}$$

2. ¿Que pasaría si aumentamos la cantidad bits del exponente?

$$+1.0 \times 10^{-99}, \quad +9.9 \times 10^{+99}, \quad -9.9 \times 10^{+99}, \quad -1.0 \times 10^{-99}$$

3. ¿Que pasaría si aumentamos la cantidad de bits del coeficiente?

$$+1.00 \times 10^{-9}, \quad +9.99 \times 10^{+9}, \quad -9.99 \times 10^{+9}, \quad -1.00 \times 10^{-9}$$

Antes de pasar al sistema punto-flotante binario, revisemos un ejemplo

1. ¿Cuáles son los números más grandes y más pequeños que se pueden representar en este sistema?

$$+1.0 \times 10^{-9}, \quad +9.9 \times 10^{+9}, \quad -9.9 \times 10^{+9}, \quad -1.0 \times 10^{-9}$$

2. ¿Que pasaría si aumentamos la cantidad bits del exponente?

$$+1.0 \times 10^{-99}, \quad +9.9 \times 10^{+99}, \quad -9.9 \times 10^{+99}, \quad -1.0 \times 10^{-99}$$

3. ¿Que pasaría si aumentamos la cantidad de bits del coeficiente?

$$+1.00 \times 10^{-9}, \quad +9.99 \times 10^{+9}, \quad -9.99 \times 10^{+9}, \quad -1.00 \times 10^{-9}$$

Antes de pasar al sistema punto-flotante binario, revisemos un ejemplo

1. ¿Cuáles son los números más grandes y más pequeños que se pueden representar en este sistema?

$$+1.0 \times 10^{-9}, \quad +9.9 \times 10^{+9}, \quad -9.9 \times 10^{+9}, \quad -1.0 \times 10^{-9}$$

2. ¿Que pasaría si aumentamos la cantidad bits del exponente?

$$+1.0 \times 10^{-99}, \quad +9.9 \times 10^{+99}, \quad -9.9 \times 10^{+99}, \quad -1.0 \times 10^{-99}$$

3. ¿Que pasaría si aumentamos la cantidad de bits del coeficiente?

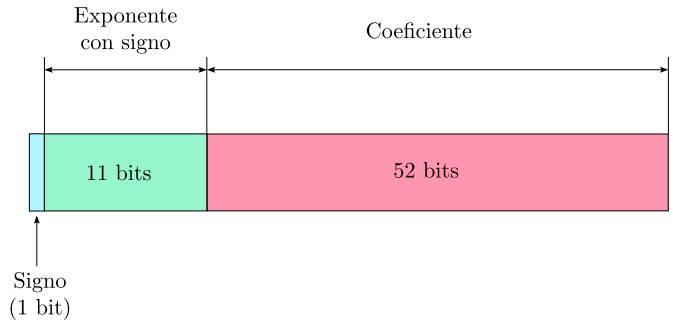
$$+1.00 \times 10^{-9}, \quad +9.99 \times 10^{+9}, \quad -9.99 \times 10^{+9}, \quad -1.00 \times 10^{-9}$$

Representación de punto-flotante en computadores: doble-precisión

En computadores se utiliza el formato *IEEE double-precision* de 64 bits dado por:

$$(-1)^{\text{sign}}(1+f)\times 2^{e-1023}$$

A la representación del exponente se le denomina *offset-binary representation*



IEEE double-precision

Ejemplo

El número en base-10 1.0000000000000002 se puede representar en doble-precisión como:

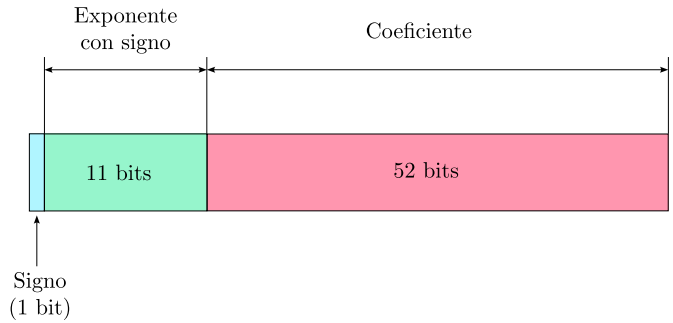
0 01111111111 0000000000000000000000000000000000000000000000000000

Representación de punto-flotante en computadores: doble-precisión

En computadores se utiliza el formato *IEEE double-precision* de 64 bits dado por:

$$(-1)^{\text{sign}}(1+f)\times 2^{e-1023}$$

A la representación del exponente se le denomina *offset-binary representation*



IEEE double-precision

Ejemplo

El número en base-10 1.0000000000000002 se puede representar en doble-precisión como:

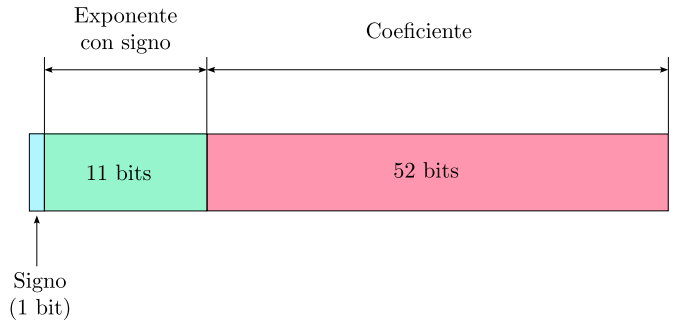
0 01111111111 0000000000000000000000000000000000000000000000000000

Representación de punto-flotante en computadores: doble-precisión

En computadores se utiliza el formato *IEEE double-precision* de 64 bits dado por:

$$(-1)^{\text{sign}}(1+f)\times 2^{e-1023}$$

A la representación del exponente se le denomina *offset-binary representation*



IEEE double-precision

Ejemplo

El número en base-10 1.0000000000000002 se puede representar en doble-precisión como:

$$0\ \ 01111111111\ \ 0000000000000000000000000000000000000000000000000000$$

Rango de la representación de doble-precisión

Los 11 bits utilizados para el exponente se traduce en un rango entre -1022 y 1023

Los exponentes -1023 y 1024 están reservados para representar números especiales

¿Cuáles es el mayor y menor número positivo que se puede representar?

$e = \texttt{00000000001}_2 = \texttt{001}_{16} = 1$: $\quad 2^{1-1023} = 2^{-1022}$

$e = \texttt{01111111111}_2 = \texttt{3ff}_{16} = 1023$: $\quad 2^{1023-1023} = 2^{0}$

$e = \texttt{10000000101}_2 = \texttt{405}_{16} = 1029$: $\quad 2^{1029-1023} = 2^{6}$

$e = \texttt{11111111110}_2 = \texttt{7fe}_{16} = 2046$: $\quad 2^{2046-1023} = 2^{1023}$

Rango de la representación de doble-precisión

Los 11 bits utilizados para el exponente se traduce en un rango entre -1022 y 1023

Los exponentes -1023 y 1024 están reservados para representar números especiales

¿Cuáles es el mayor y menor número positivo que se puede representar?

$e = \texttt{00000000001}_2 = \texttt{001}_{16} = 1$: $\quad 2^{1-1023} = 2^{-1022}$

$e = \texttt{01111111111}_2 = \texttt{3ff}_{16} = 1023$: $\quad 2^{1023-1023} = 2^{0}$

$e = \texttt{10000000101}_2 = \texttt{405}_{16} = 1029$: $\quad 2^{1029-1023} = 2^{6}$

$e = \texttt{11111111110}_2 = \texttt{7fe}_{16} = 2046$: $\quad 2^{2046-1023} = 2^{1023}$

Rango de la representación de doble-precisión

Los 11 bits utilizados para el exponente se traduce en un rango entre -1022 y 1023

Los exponentes -1023 y 1024 están reservados para representar números especiales

¿Cuáles es el mayor y menor número positivo que se puede representar?

$e = \texttt{00000000001}_2 = \texttt{001}_{16} = 1$: $2^{1-1023} = 2^{-1022}$

$e = \texttt{01111111111}_2 = \texttt{3ff}_{16} = 1023$: $2^{1023-1023} = 2^{0}$

$e = \texttt{10000000101}_2 = \texttt{405}_{16} = 1029$: $2^{1029-1023} = 2^{6}$

$e = \texttt{11111111110}_2 = \texttt{7fe}_{16} = 2046$: $2^{2046-1023} = 2^{1023}$

Respuesta:

Número mayor: $1.111\ldots111 \times 2^{+1023} \approx 2^{+1024}$

Número menor: $1.000\ldots000 \times 2^{-1022} = 2^{-1022}$

Rango de la representación de doble-precisión

Los 11 bits utilizados para el exponente se traduce en un rango entre -1022 y 1023

Los exponentes -1023 y 1024 están reservados para representar números especiales

¿Cuáles es el mayor y menor número positivo que se puede representar?

$e = \texttt{00000000001}_2 = \texttt{001}_{16}$ =1: $2^{1-1023} = 2^{-1022}$

$e = \texttt{01111111111}_2 = \texttt{3ff}_{16}$ =1023: $2^{1023-1023} = 2^{0}$

$e = \texttt{10000000101}_2 = \texttt{405}_{16}$ =1029: $2^{1029-1023} = 2^{6}$

$e = \texttt{11111111110}_2 = \texttt{7fe}_{16}$ =2046: $2^{2046-1023} = 2^{1023}$

Respuesta:

Número mayor: $1.111\ldots111 \times 2^{+1023} \approx 2^{+1024}$

Número menor: $1.000\ldots000 \times 2^{-1022} = 2^{-1022}$

Precisión de la representación de doble-precisión

La precisión del número a representar está dada por la cantidad de bits del coeficiente

Los 52 bits utilizados en doble-precisión entregan 15 o 16 dígitos de precisión en base-10

Revisemos qué pasa con lo siguiente:

```
format long
realmax % mayor numero positivo que se puede representar
realmin % menor numero positivo que se puede representar
eps     % precision de la maquina
```

[0] Código disponible en `C4_E2_maximos_valores.m`

Precisión de la representación de doble-precisión

La precisión del número a representar está dada por la cantidad de bits del coeficiente

Los 52 bits utilizados en doble-precisión entregan 15 o 16 dígitos de precisión en base-10

Revisemos qué pasa con lo siguiente:

```
format long
realmax % mayor numero positivo que se puede representar
realmin % menor numero positivo que se puede representar
eps     % precision de la maquina
```

[0] Código disponible en `C4_E2_maximos_valores.m`

Precisión de la representación de doble-precisión

La precisión del número a representar está dada por la cantidad de bits del coeficiente

Los 52 bits utilizados en doble-precisión entregan 15 o 16 dígitos de precisión en base-10

Revisemos qué pasa con lo siguiente:

```
format long
realmax % mayor numero positivo que se puede representar
realmin % menor numero positivo que se puede representar
eps     % precision de la maquina
```

[0] Código disponible en `C4_E2_maximos_valores.m`

Manipulaciones aritméticas con números de computadores

Ejemplo 3: cálculos largos

Sumemos 0.0001 diez mil veces.
¿Por qué el resultado no es exacto?

1. Introducción a Python
   - ¿Qué entendemos por error?
   - Error de redondeo
   - Error de truncamiento
     - Series de Taylor

Representación de una función usando series

Una serie permite representar una *cantidad complicada* como la suma infinita de una serie de *términos simples*.

Para aproximar la cantidad, tomamos los primeros términos de la serie y descartamos el resto.

Para aproximar $f(x)$ (cantidad complicada), ésta debe ser *analítica* en torno a $x = a$.

Usaremos *analítica* como sinónimo de $suave$.

Representación de una función usando series

Una serie permite representar una *cantidad complicada* como la suma infinita de una serie de *términos simples*.

Para aproximar la cantidad, tomamos los primeros términos de la serie y descartamos el resto.

Para aproximar $f(x)$ (cantidad complicada), ésta debe ser *analítica* en torno a $x = a$.

Usaremos *analítica* como sinónimo de $suave$.

Representación de una función usando series

Una serie permite representar una *cantidad complicada* como la suma infinita de una serie de *términos simples*.

Para aproximar la cantidad, tomamos los primeros términos de la serie y descartamos el resto.

Para aproximar $f(x)$ (cantidad complicada), ésta debe ser *analítica* en torno a $x = a$.

Usaremos *analítica* como sinónimo de *suave*.

Representación de una función usando series

Una serie permite representar una *cantidad complicada* como la suma infinita de una serie de *términos simples*.

Para aproximar la cantidad, tomamos los primeros términos de la serie y descartamos el resto.

Para aproximar $f(x)$ (cantidad complicada), ésta debe ser *analítica* en torno a $x = a$.

Usaremos *analítica* como sinónimo de $suave$.

Teorema de la serie de Taylor

Sea $f(x)$ una función suave en torno a $x = a$. Entonces, podemos escribir $f(x)$ como la siguiente serie infintia, llamada serie de Taylor de $f(x)$ en $x = a$:

$$f(x) = f(a) + f^{'}(a)(x-a) + \frac{f^{''}(a)}{2!}(x-a)^2 + \frac{f^{'''}(a)}{3!}(x-a)^3 + \cdots$$

La expresión anterior es válida para valores de x dentro de un radio de convergencia $|x - a| < R$ con $R > 0$.

Si escribimos la n-ésima derivada de $f(x)$ como $f^n(x)$, la expresión anterior queda:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Teorema de la serie de Taylor

Sea $f(x)$ una función suave en torno a $x = a$. Entonces, podemos escribir $f(x)$ como la siguiente serie infintia, llamada serie de Taylor de $f(x)$ en $x = a$:

$$f(x) = f(a) + f^{'}(a)(x-a) + \frac{f^{''}(a)}{2!}(x-a)^2 + \frac{f^{'''}(a)}{3!}(x-a)^3 + \cdots$$

La expresión anterior es válida para valores de x dentro de un radio de convergencia $|x - a| < R$ con $R > 0$.

Si escribimos la n-ésima derivada de $f(x)$ como $f^n(x)$, la expresión anterior queda:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Teorema de la serie de Taylor

Sea $f(x)$ una función suave en torno a $x = a$. Entonces, podemos escribir $f(x)$ como la siguiente serie infintia, llamada serie de Taylor de $f(x)$ en $x = a$:

$$f(x) = f(a) + f^{'}(a)(x-a) + \frac{f^{''}(a)}{2!}(x-a)^2 + \frac{f^{'''}(a)}{3!}(x-a)^3 + \cdots$$

La expresión anterior es válida para valores de x dentro de un radio de convergencia $|x - a| < R$ con $R > 0$.

Si escribimos la n-ésima derivada de $f(x)$ como $f^n(x)$, la expresión anterior queda:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Teorema de la serie de Taylor

Sea $f(x)$ una función suave en torno a $x = a$. Entonces, podemos escribir $f(x)$ como la siguiente serie infintia, llamada serie de Taylor de $f(x)$ en $x = a$:

$$f(x) = f(a) + f^{'}(a)(x-a) + \frac{f^{''}(a)}{2!}(x-a)^2 + \frac{f^{'''}(a)}{3!}(x-a)^3 + \cdots$$

La expresión anterior es válida para valores de x dentro de un radio de convergencia $|x - a| < R$ con $R > 0$.

Si escribimos la n-ésima derivada de $f(x)$ como $f^n(x)$, la expresión anterior queda:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Observación

A una serie de Taylor centrada en $x = 0$ se le llama serie de Maclaurin.

Calculemos la serie de Taylor de la función seno

Calcule la serie de Taylor de $f(x) = \sin(x)$ en torno a $x = 0$.

Recordatorio

Recuerde que la serie de Taylor de una función $f(x)$ en torno a $x = a$ se calcula como:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Calculemos la serie de Taylor de la función seno

Calcule la serie de Taylor de $f(x) = \sin(x)$ en torno a $x = 0$.

Recordatorio

Recuerde que la serie de Taylor de una función $f(x)$ en torno a $x = a$ se calcula como:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Solución

La serie de Taylor pedida es

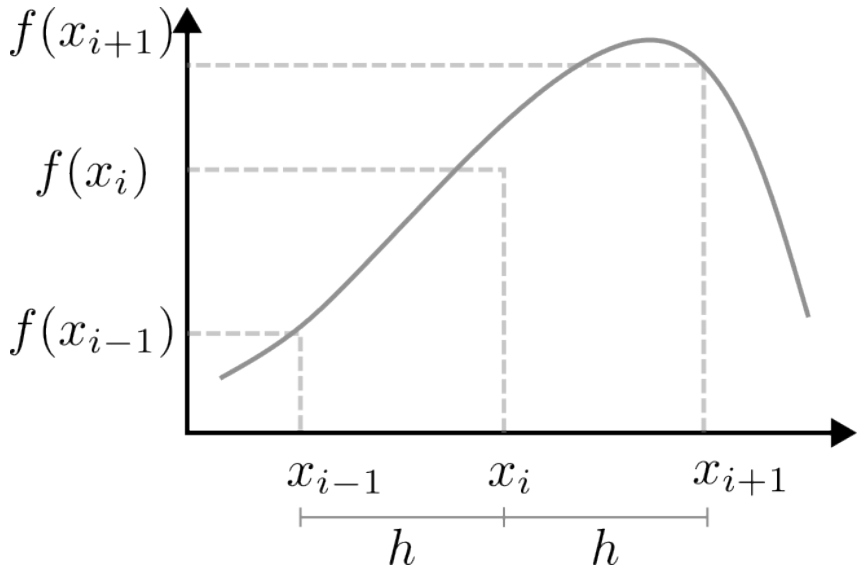
$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

Calculemos la aproximación de una función cualquiera

Calcule la serie de Taylor de $f(x)$ en torno a $x = x_i$. Evalúe su resultado en $x = x_{i+1}$ y $x = x_{i-1}$.

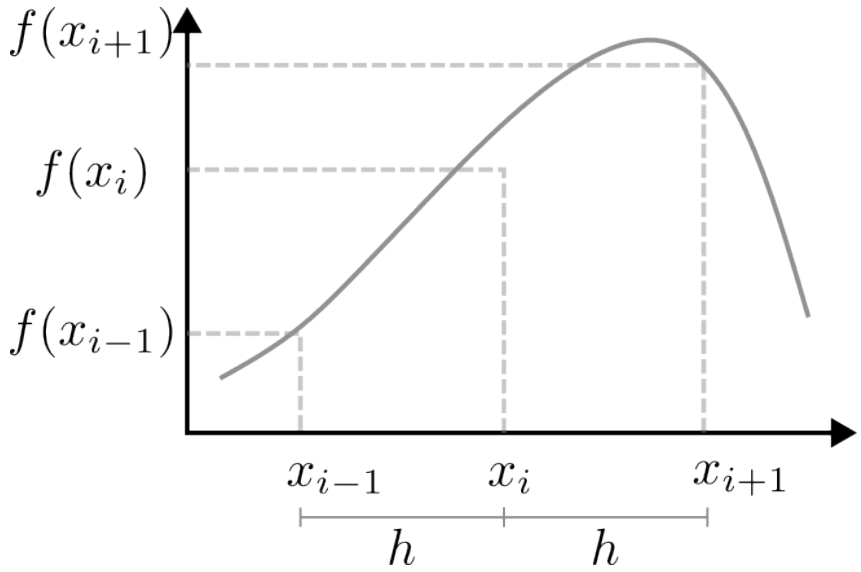
Recordatorio

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$



Calculemos la aproximación de una función cualquiera

Calcule la serie de Taylor de $f(x)$ en torno a $x = x_i$. Evalúe su resultado en $x = x_{i+1}$ y $x = x_{i-1}$.



Recordatorio

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Resultado

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots$$

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots$$

Las series de Taylor

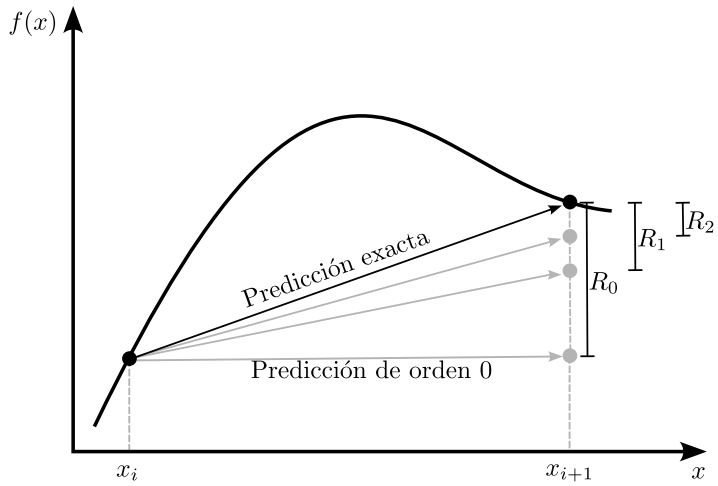
Cualquier fución suave puede ser aproximada con un polinomio

Considere las siguientes aproximaciones:

$$f(x_{i+1}) \cong f(x_i) + R_0$$

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + R_1$$

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + R_2$$



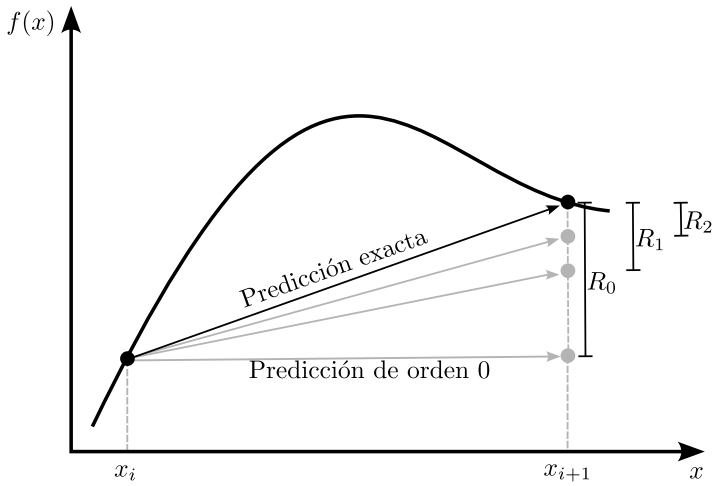
Cada aproximación comete un error dado por R_i que se puede usar al estimar derivadas para evaluar su comportamiento

Las series de Taylor

Cualquier fución suave puede ser aproximada con un polinomio

Considere las siguientes aproximaciones:

$$f(x_{i+1}) \cong f(x_i) + R_0$$
$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + R_1$$
$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + R_2$$



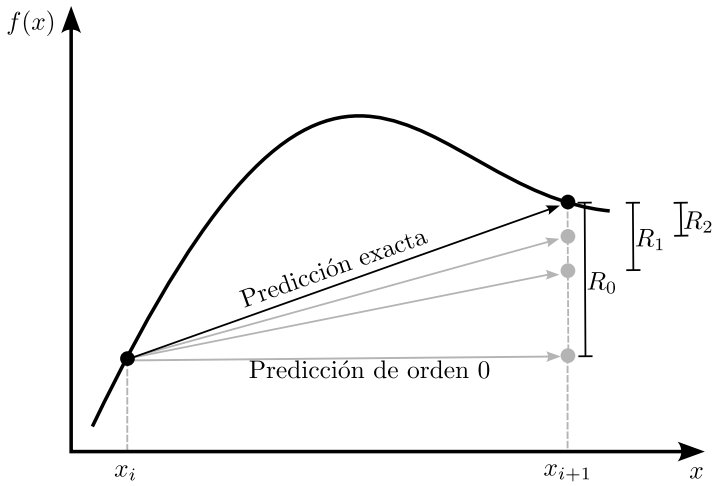
Cada aproximación comete un error dado por R_i que se puede usar al estimar derivadas para evaluar su comportamiento

Las series de Taylor

Cualquier fución suave puede ser aproximada con un polinomio

Considere las siguientes aproximaciones:

$$f(x_{i+1}) \cong f(x_i) + R_0$$
$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + R_1$$
$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + R_2$$



Cada aproximación comete un error dado por R_i que se puede usar al estimar derivadas para evaluar su comportamiento

Estimando errores de truncamiento con series de Taylor

Los errores de truncamiento ocurren cuando usamos una aproximación en lugar de un procedimiento matemático exacto

Las series de Taylor se pueden utilizar para aproximar funciones de distintas maneras:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots \quad (1)$$

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots \quad (2)$$

De (1), (2) y (1)−(2) se puede obtener lo siguiente:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \qquad \text{DF hacia adelante}$$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} - O(h) \qquad \text{DF hacia atrás}$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2) \qquad \text{DF centradas}$$

Estimando errores de truncamiento con series de Taylor

Los errores de truncamiento ocurren cuando usamos una aproximación en lugar de un procedimiento matemático exacto

Las series de Taylor se pueden utilizar para aproximar funciones de distintas maneras:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots \quad (1)$$

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots \quad (2)$$

De (1), (2) y (1)−(2) se puede obtener lo siguiente:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \quad \text{DF hacia adelante}$$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} - O(h) \quad \text{DF hacia atrás}$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2) \quad \text{DF centradas}$$

Estimando errores de truncamiento con series de Taylor

Los errores de truncamiento ocurren cuando usamos una aproximación en lugar de un procedimiento matemático exacto
Las series de Taylor se pueden utilizar para aproximar funciones de distintas maneras:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots \quad (1)$$

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots \quad (2)$$

De (1), (2) y (1)−(2) se puede obtener lo siguiente:

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \qquad \text{DF hacia adelante}$$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} - O(h) \qquad \text{DF hacia atrás}$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2) \qquad \text{DF centradas}$$

Estimación de errores usando series de Taylor

Si el error depende de $O(h)$, el método de aproximación mejora o empeora linealmente con h

Si el error depende de $O(h^2)$, el método de aproximación mejora o empeora cuadráticamente con h

Las series de Taylor dan una idea general del error que se está cometiendo

Ejemplo 4

Use diferencias finitas hacia adelante, hacia atrás y centradas para estimar la primera derivada de

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

en $x = 0.5$ usando un paso de $h = 0.5$. Repita el cálculo usando $h = 0.25$. Note que la derivada puede ser calculada directamente como

$$f'(x) = -0.4x^3 - 0.45x^2 - x - 0.25$$

Estimación de errores usando series de Taylor

Si el error depende de $O(h)$, el método de aproximación mejora o empeora linealmente con h

Si el error depende de $O(h^2)$, el método de aproximación mejora o empeora cuadráticamente con h

Las series de Taylor dan una idea general del error que se está cometiendo

Ejemplo 4

Use diferencias finitas hacia adelante, hacia atrás y centradas para estimar la primera derivada de

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

en $x = 0.5$ usando un paso de $h = 0.5$. Repita el cálculo usando $h = 0.25$. Note que la derivada puede ser calculada directamente como

$$f'(x) = -0.4x^3 - 0.45x^2 - x - 0.25$$

Estimación de errores usando series de Taylor

Si el error depende de $O(h)$, el método de aproximación mejora o empeora linealmente con h

Si el error depende de $O(h^2)$, el método de aproximación mejora o empeora cuadráticamente con h

Las series de Taylor dan una idea general del error que se está cometiendo

Ejemplo 4

Use diferencias finitas hacia adelante, hacia atrás y centradas para estimar la primera derivada de

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

en $x = 0.5$ usando un paso de $h = 0.5$. Repita el cálculo usando $h = 0.25$. Note que la derivada puede ser calculada directamente como

$$f'(x) = -0.4x^3 - 0.45x^2 - x - 0.25$$

Repasemos

- ¿Cuál es la diferencia entre exactitud y precisión?
- ¿De qué factores depende el error que estamos cometiendo en una aproximación?
- ¿Cuándo ocurren los errores de truncamiento?
- ¿Que información nos entregan las series de Taylor al estimar derivadas con diferencias finitas?

Repasemos

- ¿Cuál es la diferencia entre exactitud y precisión?
- ¿De qué factores depende el error que estamos cometiendo en una aproximación?
- ¿Cuándo ocurren los errores de truncamiento?
- ¿Que información nos entregan las series de Taylor al estimar derivadas con diferencias finitas?

Repasemos

- ¿Cuál es la diferencia entre exactitud y precisión?
- ¿De qué factores depende el error que estamos cometiendo en una aproximación?
- ¿Cuándo ocurren los errores de truncamiento?
- ¿Que información nos entregan las series de Taylor al estimar derivadas con diferencias finitas?

Repasemos

- ¿Cuál es la diferencia entre exactitud y precisión?
- ¿De qué factores depende el error que estamos cometiendo en una aproximación?
- ¿Cuándo ocurren los errores de truncamiento?
- ¿Que información nos entregan las series de Taylor al estimar derivadas con diferencias finitas?