

Tema 1 - Escalas, condición y estabilidad

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

Facultad de Ciencias - UNAM

14 de febrero de 2018



1. Tema 1
2. Física Computacional
3. Conceptos principales
4. Errores en los métodos numéricos
5. Los números en las computadoras
6. Tipos de errores

7. Contaminación en los cálculos

8. Algoritmo de Horner

Al concluir el Tema 1, el alumno:

- 1 En el diseño de algoritmos para la solución numérica de problemas de la física, aplicará los conceptos de: *Condición, Estabilidad y Eficiencia*, apoyándose en la teoría de representación de números en la computadora y de la teoría de propagación de errores.

¿Qué es la física computacional?

La física computacional es una nueva manera de hacer investigación en física, próxima al experimento y a la teoría.

En el laboratorio se realizan mediciones en sistemas físicos reales (restringida a la factibilidad de recursos técnicos), y que luego los físicos teóricos explican esas mediciones mediante las teorías.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.
- Validar aproximaciones y hacer efectivas las teorías propuestas.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.
- Validar aproximaciones y hacer efectivas las teorías propuestas.
- Comparar cuantitativamente teorías y mediciones experimentales.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.
- Validar aproximaciones y hacer efectivas las teorías propuestas.
- Comparar cuantitativamente teorías y mediciones experimentales.
- Visualizar conjuntos de datos complejos.

Áreas de investigación en la física

- Problemas que no tienen solución analítica.
- Validar aproximaciones y hacer efectivas las teorías propuestas.
- Comparar cuantitativamente teorías y mediciones experimentales.
- Visualizar conjuntos de datos complejos.
- Control y medición de experimentos.

- Predicción del clima

- Predicción del clima
- Superconductividad

- Predicción del clima
- Superconductividad
- Genoma Humano

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión

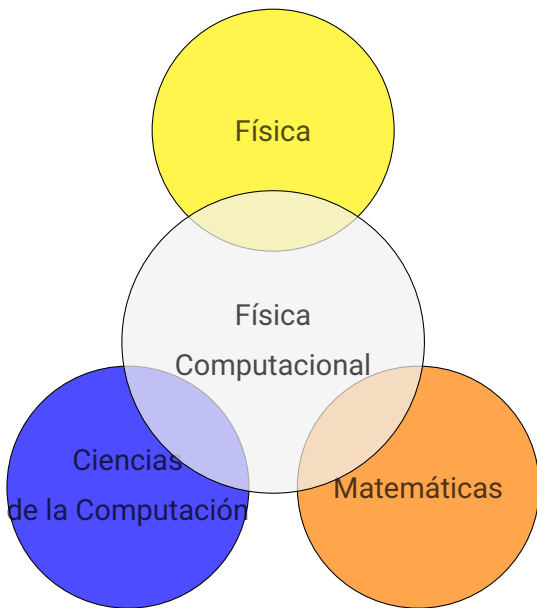
- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica
- Diseño de fármacos

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica
- Diseño de fármacos
- Turbulencia

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica
- Diseño de fármacos
- Turbulencia
- Recuperación de petróleo y gas

- Predicción del clima
- Superconductividad
- Genoma Humano
- Visión y lenguaje
- Fusión nuclear
- Oceanografía
- Ciencia de los materiales
- Diseño de semiconductores
- Astrofísica relativista
- Sistemas de combustión
- Estructura biológica
- Diseño de fármacos
- Turbulencia
- Recuperación de petróleo y gas
- Cromodinámica cuántica



Alcance del curso

El curso está diseñado para ofrecer una introducción a los métodos numéricos aplicados a la física.

Se da un punto de referencia para continuar profundizando de manera particular en temas específicos.

El desarrollo de las habilidades de programación, están en función del tiempo dedicado al trabajo fuera de la clase, pero consideramos que se abren un panorama diferente para abordar ya sea un servicio social, tesis o proyecto de trabajo para un posgrado.

Animación generada con un script de python

Ejemplo

Péndulo

1. Tema 1

2. Física Computacional

3. Conceptos principales

3.1 Métodos numéricos

3.2 Modelos y métodos numéricos

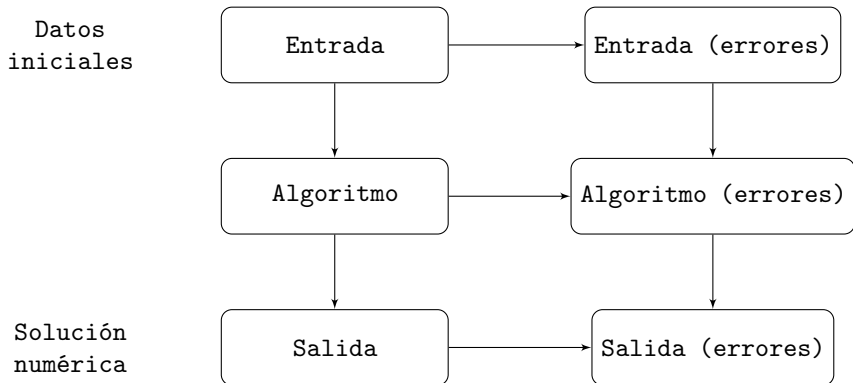
4. Errores en los métodos numéricos

5. Los números en las computadoras

6. Tipos de errores

Método numérico

Se puede representar como una cadena de algoritmos A_i con $(i = 1, 2, 3, \dots, N)$ en la entrada y salida.



La solución obtenida por un método numérico es **aproximada**, es decir, hay cierta diferencia entre la solución exacta y la solución numérica.

Principales causas de la diferencia

- Falta de correspondencia entre el problema (modelo) matemático y el fenómeno físico real.

Principales causas de la diferencia

- Falta de correspondencia entre el problema (modelo) matemático y el fenómeno físico real.
- Errores en los datos iniciales (parámetros de entrada).

Principales causas de la diferencia

- Falta de correspondencia entre el problema (modelo) matemático y el fenómeno físico real.
- Errores en los datos iniciales (parámetros de entrada).
- Errores en el método numérico usado para resolver el problema.

Principales causas de la diferencia

- Falta de correspondencia entre el problema (modelo) matemático y el fenómeno físico real.
- Errores en los datos iniciales (parámetros de entrada).
- Errores en el método numérico usado para resolver el problema.
- Errores de redondeo en las operaciones aritméticas.

- Aproximación.

Conceptos en métodos numéricos

- Aproximación.
- Estabilidad.

Conceptos en métodos numéricos

- Aproximación.
- Estabilidad.
- Convergencia.

Es la proximidad de un modelo numérico al modelo original (diferencial, integral, etc.) o el grado de aproximación, caracteriza el error que se introduce al hacer discreto el modelo continuo.

El grado de aproximación n se estima mediante un factor que tiene el error entre dos modelos.

- Caracteriza la manera de propagación de los errores iniciales dentro del algoritmo en el proceso del cálculo.

- Caracteriza la manera de propagación de los errores iniciales dentro del algoritmo en el proceso del cálculo.
- Si el incremento de errores iniciales es considerable y sin ningún control, entonces el método numérico se llama **inestable**.

- Caracteriza la manera de propagación de los errores iniciales dentro del algoritmo en el proceso del cálculo.
- Si el incremento de errores iniciales es considerable y sin ningún control, entonces el método numérico se llama **inestable**.
- Si los errores de cálculos dependen continuamente de los errores iniciales, entonces el método se llama **estable**.

- Significa que la solución numérica converge hacia la solución exacta cuando el tamaño de la malla h tiene a cero, o el número de truncación N tiende al infinito.

1. Tema 1

2. Física Computacional

3. Conceptos principales

4. Errores en los métodos numéricos

4.1 Tipos de errores

4.2 Series de Taylor

5. Los números en las computadoras

6. Tipos de errores

Tipos de errores en los métodos numéricos

- **Truncamiento:** se debe a las aproximaciones utilizadas en la fórmula matemática del modelo.

Tipos de errores en los métodos numéricos

- **Truncamiento**: se debe a las aproximaciones utilizadas en la fórmula matemática del modelo.
- **Redondeo**: se asocia al hecho de que la representación de un número en la computadora se hace con un conjunto limitado de dígitos.

- Las soluciones numéricas son en su mayoría, aproximaciones de las soluciones exactas.

- Las soluciones numéricas son en su mayoría, aproximaciones de las soluciones exactas.
- Gran parte de los métodos numéricos se basan en la aproximación de funciones por medio de polinomios.

El desarrollo de Taylor es una serie infinita de potencias, representa de manera exacta a una función dentro de un cierto radio alrededor de un punto dado.

Al comparar el desarrollo polinomial de la solución numérica con la serie de Taylor de la solución exacta, es posible evaluar el error, conocido como error de **truncamiento**.

Serie de Taylor truncada

Si se ignoran todos los términos de la serie de Taylor, excepto algunos cuantos, se puede obtener un polinomio que se aproxime a la función verdadera.

A éste polinomio se le llama *serie de Taylor truncado* y se usa como punto de partida para obtener métodos numéricos.

Definición de Serie de Taylor

Una función $f(x)$ es analítica en $x = a$ si $f(x)$ se puede representar por medio de una serie de potencias en términos de $h = x - a$ dentro de un radio de convergencia $D > |x - a| > 0$.

Definición de Serie de Taylor

Una función $f(x)$ es analítica en $x = a$ si $f(x)$ se puede representar por medio de una serie de potencias en términos de $h = x - a$ dentro de un radio de convergencia $D > |x - a| > 0$.

Una condición necesaria para que una función sea analítica es que todas sus derivadas sean continuas tanto en $x = a$ como en alguna vecindad alrededor de ese punto.

Un punto en donde una función $f(x)$ no es analítica recibe el nombre de *punto singular*.

Un punto en donde una función $f(x)$ no es analítica recibe el nombre de *punto singular*.

Si $f(x)$ es diferenciable en todas las partes de la vecindad de x_0 , excepto en x_0 , entonces x_0 es un punto singular.

Los polinomios son analíticos en todas partes.

Serie para una función analítica

Si f es analítica alrededor de $x = a$, se puede representar $f(x)$ de manera exacta en la vecindad de $x = a$ por medio de una serie de Taylor, dada por:

$$\begin{aligned} f(x) = f(a) + h f'(a) + \frac{h^2}{2} f''(a) + \frac{h^3}{6} f'''(a) + \dots \\ + \dots + \frac{h^m}{m!} f^m(a) + \dots \end{aligned}$$

Unicidad de la serie

La serie de Taylor es única, esto quiere decir que no existe otra serie de potencias en $h = x - a$ para representar $f(x)$

El desarrollo de Taylor de una función alrededor de $x = 0$ recibe el nombre de serie de Maclaurin.

Truncamiento necesario

En aplicaciones prácticas, se debe de truncar la serie de Taylor después cierto orden, ya que es imposible incluir un número infinito de términos. Si la serie se trunca después del término N , se expresa por:

$$\begin{aligned} f(x) &= f(a) + hf'(a) + \frac{h^2}{2}f''(a) + \frac{h^3}{6}f'''(a) + \dots \\ &= + \dots + \frac{h^N}{N!}f^N(a) + O(h^{N+1}) \end{aligned}$$

Truncamiento necesario

$$\begin{aligned}f(x) &= f(a) + hf'(a) + \frac{h^2}{2}f''(a) + \frac{h^3}{6}f'''(a) + \dots \\&= + \dots + \frac{h^N}{N!}f^N(a) + O(h^{N+1})\end{aligned}$$

Donde $h = x - a$ y $O(h^{N+1})$ representa el error por el truncamiento de los términos de orden $N + 1$

Error por el truncamiento

El error global se puede representar como:

$$O(h^{N+1}) = f^{N+1}(a + \xi h) \frac{h^{N+1}}{(N+1)!} \quad 0 \leq \xi \leq 1$$

Dado que ξ no puede calcularse con exactitud, se aproxima el término del error, haciendo $\xi = 0$

$$O(h^{N+1}) \simeq f^{N+1}(a) \frac{h^{N+1}}{(N+1)!}$$

Error por el truncamiento

Si $N = 1$ la serie de Taylor truncada es:

$$f(x) \simeq f(a) + f'(a)h \quad h = x - a$$

incluyendo el efecto del error, tenemos que

$$f(x) \simeq f(a) + f'(a)h + O(h^2)$$

donde

$$O(h^2) \simeq f''(a + \xi h) \frac{h^2}{2} \quad 0 \leq \xi \leq 1$$

1. Tema 1

2. Física Computacional

3. Conceptos principales

4. Errores en los métodos numéricos

5. Los números en las computadoras

5.1 Base decimal

5.2 Epsilon de la máquina

6. Tipos de errores

Representación en una computadora

Antes de lanzarnos a la revisión de estrategias de solución para un problema real, se requiere entender las limitaciones que tendremos para representar los números en una computadora.

De lo continuo a la discreto

Hemos recibido una formación matemática en donde lo común es pensar en intervalos continuos.

De lo continuo a la discreto

Hemos recibido una formación matemática en donde lo común es pensar en intervalos continuos.

Donde hemos logrado demostrar que existe un conjunto infinito de valores dentro de un intervalo finito.

De lo continuo a lo discreto

Pero en la práctica con las computadoras, no podremos representar esa continuidad.

De lo continuo a lo discreto

Pero en la práctica con las computadoras, no podremos representar esa continuidad.

Tendremos que “truncar” nuestra representación, ya que el hardware que usamos no nos permite contar con esa propiedad de continuidad.

El valor decimal de un número de base r es

$$(abcdefg.hijk)_r$$

que se calcula como:

$$\begin{aligned} ar^6 + br^5 + cr^4 + dr^3 + er^2 + fr + g + \\ + hr^{-1} + ir^{-2} + jr^{-3} + kr^{-4} \end{aligned}$$

Un número real en la computadora

La menor y mayor magnitud de un número real que se pueden representar en una computadora, varían de acuerdo con el diseño tanto de hardware como de software.

Los números reales en una computadora no son continuos. Si nos fijamos en número cercano a cero, el número positivo más pequeño en una IBM es 2.9×10^{-39}

Un número real en la computadora

Por tanto, no se pueden representar números entre 0 y 2.9×10^{-39}

A éste intervalo se le conoce como **épsilon de la máquina**.

Epsilon de la máquina

Hay dos maneras de definir el ϵ de la máquina: un ϵ absoluto y un ϵ relativo.

Este último es el más usado, ya que el conjunto de números en la computadora es finito.

Definición de épsilon relativo:

$$\epsilon_{maq} = \epsilon = \min[t > 0 : 1 + t > 1]$$

Epsilon absoluto de la máquina

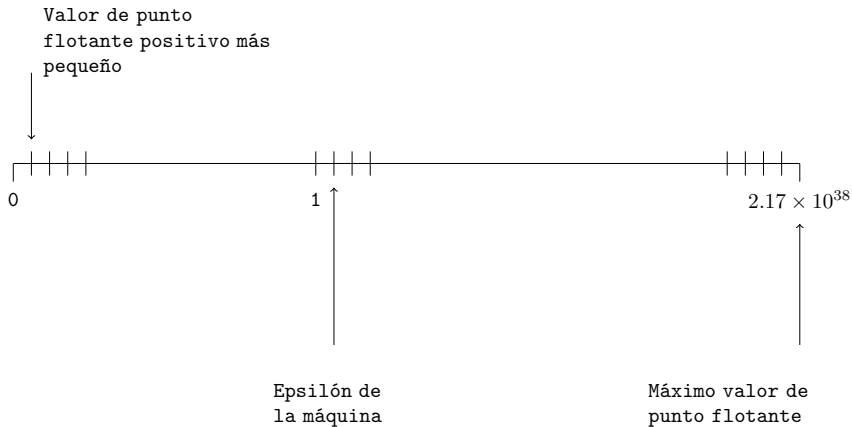
El épsilon absoluto se define comparando con cero:

$$\epsilon_{abs} = \min[t > 0 : t \neq 0]$$

En realidad el épsilon depende de la máquina pero también del sistema operativo, del compilador y del tipo de números utilizados.

Números reales en un computadora

Para una computadora IBM PC tenemos:



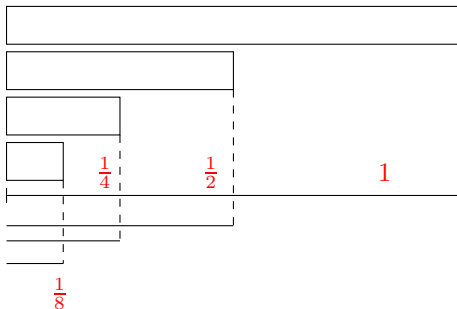
Calculemos el ϵ con `python`

Una manera sencilla de calcular el ϵ del equipo que estamos usando, es reducir a la mitad un intervalo inicial, hasta cierto punto.

Calculemos el épsilon con python

Una manera sencilla de calcular el épsilon del equipo que estamos usando, es reducir a la mitad un intervalo inicial, hasta cierto punto.

Empecemos dividiendo a la mitad la unidad.



Calculamos el epsilon con python

Código 1: Código para el epsilon

```
1 t = 1.0;
2 while 1+t != 1:
3     eps = t
4     t = t/2
5 print ('el epsilon de la maquina es: ', eps)
```

1. Tema 1

2. Física Computacional

3. Conceptos principales

4. Errores en los métodos numéricos

5. Los números en las computadoras

6. Tipos de errores

6.1 Errores en programación

6.2 Error absoluto verdadero

Como ya hemos mencionado que un programa numérico ejecutado en la computadora, nos va a devolver un resultado que es una aproximación, entonces hay que considerar que existe un **error** debido al proceso.

En métodos numéricos es importante identificar el tipo de error que se obtiene, ya que nos permitirá entonces establecer no sólo un valor, sino la magnitud del error comparada con el valor esperado o “exacto”.

El error entendido como la suma o consecuencia por un truncamiento o redondeo, se clasifica en:

- **Error absoluto verdadero.**

El error entendido como la suma o consecuencia por un truncamiento o redondeo, se clasifica en:

- Error absoluto verdadero.
- Error relativo verdadero.

El error entendido como la suma o consecuencia por un truncamiento o redondeo, se clasifica en:

- Error absoluto verdadero.
- Error relativo verdadero.
- Error relativo aproximado.

Supóngase que \hat{p} es una aproximación a p .

El error absoluto verdadero se define con la siguiente expresión:

$$E_v = |p - \hat{p}|$$

Esta definición de error, lo cuantifica en términos brutos.

No obstante, una medida que puede describir con mayor detalle o proporción el error, es aquella que lo expresa en términos porcentuales.

Para ello se emplea el error verdadero relativo.

Supóngase que \hat{p} es una aproximación a p . El error relativo verdadero se calcula con la siguiente expresión:

$$e_v = \frac{|p - \hat{p}|}{p}$$

El resultado suele expresarse en términos porcentuales.

Error relativo aproximado

El error relativo aproximado, mide el error de un método numérico, determinando el error de la iteración actual respecto el error surgido en la iteración anterior:

$$e_a = \frac{|\hat{x}_i - \hat{x}_{i-1}|}{\hat{x}_i}$$

Donde x_i es la aproximación actual a x y x_{i-1} es la aproximación anterior a x .

En métodos numéricos suele establecerse una tolerancia porcentual como criterio de paro, tal que el error relativo aproximado de un método, no exceda dicha tolerancia.

$$e_a < t$$

donde t , es tolerancia fijada de antemano.

A menor tolerancia se tiene mayor precisión en la aproximación al valor verdadero, sin embargo esto implica un aumento en el número de iteraciones requeridas para detener el método.

Contaminación en los cálculos.

Un error en un cálculo numérico “contamina” las sucesivas evaluaciones.

Esta propagación puede describirse en términos de dos conceptos relacionados: los de estabilidad y condición.

La condición de una función $f(x)$ mide la sensibilidad de los valores de $f(x)$ a pequeños cambios de x , se define como:

$$C = \left| \frac{E_{rel}(f(x))}{E_{rel}(x)} \right|$$

Definición de condición

Del teorema del valor medio en cálculo, podemos expresar

$$\begin{aligned} f(x_T) - f(x_A) &\approx f'(x_t)(x_T - x_A) \rightarrow E_{rel}(f(x)) \approx \\ &\approx \frac{f'(x_T)}{f(x_T)}(x_T - x_A) \end{aligned}$$

luego

$$C \approx \left| x_T \frac{f'(x_T)}{f(x_T)} \right|$$

Definición de condición

Se utilizará ésta definición como definición de condición para funciones $f(x)$ de una variable real.

Entonces los números de condición serán

$$C = \left| x \frac{f'(x)}{f(x)} \right|$$

Definición de condición

$$C = \left| x \frac{f'(x)}{f(x)} \right|$$

- ❶ Para un x dado $0 < C(x) < 1$ se dirá que el problema está bien condicionado, y cuanto menor sea C , mejor condicionado.

Definición de condición

$$C = \left| x \frac{f'(x)}{f(x)} \right|$$

- 1 Para un x dado $0 < C(x) < 1$ se dirá que el problema está bien condicionado, y cuanto menor sea C , mejor condicionado.
- 2 Si $C(x) > 1$, el problema estará mal condicionado.

Definición de condición

$$C = \left| x \frac{f'(x)}{f(x)} \right|$$

- 1 Para un x dado $0 < C(x) < 1$ se dirá que el problema está bien condicionado, y cuanto menor sea C , mejor condicionado.
- 2 Si $C(x) > 1$, el problema estará mal condicionado.
- 3 Si $C(x) = 1$, el error relativo se mantiene.

Las siguientes funciones están bien condicionadas?

1 $f(x) = \sqrt{x}$ $C(x) = ?$

2 $g(x) = x^2 - 1$ $C(x) = ?$

La estabilidad de un algoritmo describe la sensibilidad de un método numérico específico respecto a los inevitables errores de redondeo cometidos durante su ejecución en aritmética de precisión finita.

Consideremos la siguiente función:

$$f(x) = \sqrt{x+1} - \sqrt{x}$$

Su número de condición es:

$$C(x) = \left| x \frac{f'(x)}{f(x)} \right| = \frac{x}{2\sqrt{x}\sqrt{x+1}}$$

Vemos que $C(x) < \frac{1}{2}$ para $x > 0$, por lo que la función está bien condicionada pero ...

El pseudo-código para calcular x de tal forma que se vayan realizando las operaciones, es:

- 1 obtener x

El pseudo-código para calcular x de tal forma que se vayan realizando las operaciones, es:

- 1 obtener x
- 2 $y = x + 1$

El pseudo-código para calcular x de tal forma que se vayan realizando las operaciones, es:

- 1 obtener x
- 2 $y = x + 1$
- 3 $f = \text{sqrt}(y)$

El pseudo-código para calcular x de tal forma que se vayan realizando las operaciones, es:

① obtener x

② $y = x + 1$

③ $f = \text{sqrt}(y)$

④ $g = \text{sqrt}(x)$

El pseudo-código para calcular x de tal forma que se vayan realizando las operaciones, es:

① obtener x

② $y = x + 1$

③ $f = \text{sqrt}(y)$

④ $g = \text{sqrt}(x)$

⑤ $h = f - g$

El pseudo-código para calcular x de tal forma que se vayan realizando las operaciones, es:

① obtener x

② $y = x + 1$

③ $f = \text{sqrt}(y)$

④ $g = \text{sqrt}(x)$

⑤ $h = f - g$

El pseudo-código para calcular x de tal forma que se vayan realizando las operaciones, es:

- 1 obtener x
- 2 $y = x + 1$
- 3 $f = \text{sqrt}(y)$
- 4 $g = \text{sqrt}(x)$
- 5 $h = f - g$

es inestable para x grandes, dado el paso 5, por lo que debemos de re-estructurar la función.

Debemos evitar que todo algoritmo sea inestable.

Si existieran varios métodos para evaluar una misma función, entonces conviene utilizar aquel que sea más eficiente, es decir, más rápido.

Hay que aprovechar al máximo los recursos: hardware, software, algoritmos, para resolver problemas más complejos y no para resolver peor problemas simples.

Ejemplo de eficiencia

Por ejemplo, para calcular $x * 4$ para un x dado, no es buena idea calcular $x * 4.0$ (exponente en punto flotante).

Ejemplo de eficiencia

Por ejemplo, para calcular $x * *4$ para un x dado, no es buena idea calcular $x * *4.0$ (exponente en punto flotante).

La mejor idea consiste en economizar el cálculo en dos pasos:

$$x^2 = x * x$$

$$x^4 = x^2 * x^2$$

y no un producto

$$x^4 = x * x * x * x$$

Ejemplo: Evaluación de polinomios

Supongamos que queremos evaluar el polinomio:

$$P(x) = 2 + 4x - 5x^2 + 2x^3 - 6x^4 + 8x^5 + 10x^6$$

Ejemplo: Evaluación de polinomios

Supongamos que queremos evaluar el polinomio:

$$P(x) = 2 + 4x - 5x^2 + 2x^3 - 6x^4 + 8x^5 + 10x^6$$

Contando con que cada potencia de exponente k entero como $k - 1$ productos, tendríamos que el total de productos para evaluar en forma directa es:

$$1 + 2 + 3 + 4 + 5 + 6 = 21$$

Además de seis sumas.

Ejemplo: Evaluación de polinomios

Una mejora en el algoritmo , es calcular primero las potencias de forma sucesiva:

$$x^2 = x * x$$

$$x^3 = x * x^2$$

$$x^4 = x * x^3$$

$$x^5 = x * x^4$$

$$x^6 = x * x^5$$

De tal forma que se añade un producto por cada potencia, para un total de productos:

$$1 + 2 + 2 + 2 + 2 + 2 = 11$$

Ejemplo: Evaluación de polinomios

Con el polinomio

$$P(x) = 2 + 4x - 5x^2 + 2x^3 - 6x^4 + 8x^5 + 10x^6$$

podemos mejorar el algoritmo de la siguiente manera

$$P(x) = 2 + x \{4 + x (-5 + x [2 + x (-6 + x \{8 + x * 10\})])\}$$

Ejemplo: Evaluación de polinomios

Para evaluar un polinomio de grado n en el que ninguno de los coeficientes es cero, se necesitan

Ejemplo: Evaluación de polinomios

Para evaluar un polinomio de grado n en el que ninguno de los coeficientes es cero, se necesitan

$\frac{n(n+1)}{2}$ Productos para el primer método

$2n - 1$ para el segundo métodos

n para el tercero

Ejemplo: Evaluación de polinomios

Para evaluar un polinomio de grado n en el que ninguno de los coeficientes es cero, se necesitan

$\frac{n(n+1)}{2}$ Productos para el primer método

$2n - 1$ para el segundo métodos

n para el tercero

Antes de escribir una línea de código, hay que revisar la manera en que podemos optimizar la solución del problema.

Algoritmo de Horner

Dado el polinomio

$$P(x) = a_0 + a_1x + \dots + a_nx^n \quad a_n \neq 0$$

Algoritmo de Horner

Dado el polinomio

$$P(x) = a_0 + a_1x + \dots + a_nx^n \quad a_n \neq 0$$

La evaluación de $P(x)$ para cierto valor de $x = z$ se puede realizar en n pasos mediante:

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + z * b_n \\ b_{n-2} &= a_{n-2} + z * b_{n-1} \\ &\vdots \\ b_0 &= a_0 + z * b_1 \end{aligned}$$

Pseudocódigo

$$\textcircled{1} \ b_n = a_n$$

Pseudocódigo

- 1 $b_n = a_n$
- 2 repetir mientras $n > 0$

Pseudocódigo

- 1 $b_n = a_n$
- 2 repetir mientras $n > 0$
- 3 $n = n - 1$

Pseudocódigo

- ➊ $b_n = a_n$
- ➋ repetir mientras $n > 0$
- ➌ $n = n - 1$
- ➍ $b = a_n + z * b$

Pseudocódigo

- 1 $b_n = a_n$
- 2 repetir mientras $n > 0$
- 3 $n = n - 1$
- 4 $b = a_n + z * b$
- 5 volver al paso 2

Pseudocódigo

- 1 $b_n = a_n$
- 2 repetir mientras $n > 0$
- 3 $n = n - 1$
- 4 $b = a_n + z * b$
- 5 volver al paso 2
- 6 $p(z) = b$

Evalúa el polinomio $P(x)$

$$P(x) = 2 + 4x - 5x^2 + 2x^3 - 6x^4 + 8x^5 + 10x^6$$

x	$P(x)$
-1.5	
-0.65	
0.1	
1.4	
2.87	

Resultado

El polinomio $P(x)$ evaluado con el método de Horner:

—1.5	0.781 25
—0.65	—4.506 83
0.1	2.351 49
1.4	98.559 68
2.87	6758.702 45

Extendiendo la respuesta al problema

¿Cómo resolver el problema usando una función?
¿mostrando una tabla con formato de salida?
usando una gráfica que muestre $P(x)$ y un
conjunto de datos evaluados? ¿Evaluar el error
relativo?

Extendiendo la respuesta al problema

Ya contamos con las herramientas necesarias para extender la respuesta al problema, en nuestro código podemos agregar funciones, ajustar formatos de salida en los resultados, graficar el polinomio y los puntos (o un conjunto diferente de puntos), y obtener el error relativo.

Pasos a resolver

- 1 Conviene definir una función que resuelva la evaluación del método de Horner.

Pasos a resolver

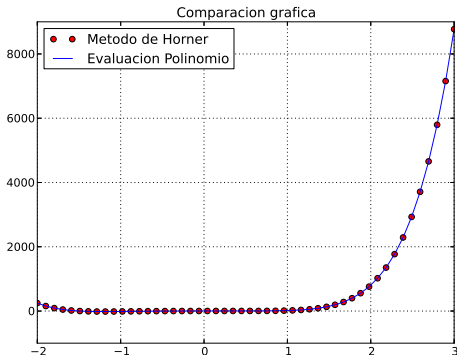
- 1 Conviene definir una función que resuelva la evaluación del método de Horner.
- 2 Para obtener el error relativo, se debe de evaluar el polinomio y considerar que los valores obtenidos, son los valores exactos.

Pasos a resolver

- ❶ Conviene definir una función que resuelva la evaluación del método de Horner.
- ❷ Para obtener el error relativo, se debe de evaluar el polinomio y considerar que los valores obtenidos, son los valores exactos.
- ❸ Comparamos los resultados mediante una gráfica que represente los dos resultados de la evaluación.

Resultado gráfico más completo

En la gráfica se muestra un conjunto de datos que se evalúan y posteriormente con la función (en línea continua) se compara, podemos ver que los resultados son prácticamente los mismos.



Estructuramos el código en `python`.

Definimos mediante dos listas:

- 1 Los valores donde queremos evaluar el polinomio $P(x)$.
- 2 Los coeficientes de $P(x)$.

Función de Horner.

Código 2: Código para la función de Horner

```
1 # Metodo de Horner
2
3 def P_Horner(x):
4     P_Hor=0
5     for n in range(len(a)-1,-1,-1):
6         P_Hor=a[n]+P_Hor*x
7     return P_Hor
```


Función que evalúa directamente $P(x)$.

Código 3: Evaluación directa de la función

```
1 # Evaluacion directa
2
3 def P_Directo(x):
4     return 2 + 4 * x - 5 * x**2 + 2 * x**3 - 6 *
           x**4 + 8 * x**5 + 10 * x**6
```

Calculamos el error relativo.

Código 4: Evaluación del error relativo

```
1 # Calculo de error relativo
2
3 def Err_Rel(p,p_): return (p-p_)/p*100
```

Uso de valores conocidos

Código 5: Valores conocidos

```
1 # Valores de x0 para evaluar P(x0)
2 x0 = [-1.5, -0.65, 0.1, 1.4, 2.87]
3
4 # Coeficientes a de P(x)
5 a = [2, 4, -5, 2, -6, 8, 10]
```

Mostramos el error relativo de los puntos a evaluar.

Código 6: Error relativo calculado

```
1 # Evaluacion de valores de P(x0)
2
3 for i in range(len(x0)):
4     print ("P(%.2f) =" %x0[i], P_Horner(x0[i]),
           "; Error rel. =", Err_Rel(P_Directo(x0[i]),
           P_Horner(x0[i])))
```

Los resultados en una gráfica. I

Código 7: Elaboración de la gráfica

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 x = np.linspace(-2., 3.)
5
6 plt.plot(x, P_Horner(x), 'ro', label='Metodo de
    Horner')
7
8 plt.plot(x, P_Directo(x), label='Evaluacion
    Polinomio')
```

Los resultados en una gráfica. II

```
9  
10 plt.title('Comparacion grafica')  
11 plt.legend(loc='upper left')  
12  
13 plt.grid(True)  
14 plt.show()
```

Función para generar secuencia de reales

Ya hemos mencionado que la librería `numpy` extiende el conjunto de funciones matemáticas que se tienen dentro de la otra librería `math`.

Veamos la sintaxis de la función `linspace` que aparece en la rutina de graficación.

La función `linspace`

Sintaxis mínima:

```
linspace(inicio, fin, num=50)
```

Donde:

- `inicio`: es el valor a partir del cual queremos generar la secuencia de puntos.
- `fin`: es el valor en donde se detiene la secuencia de puntos.
- `num`: es el número de puntos que se genera en el intervalo, por defecto se generan 50 puntos.

La secuencia que se genera

Lo que obtenemos con `linspace`, es una secuencia de puntos distribuidos uniformemente en el intervalo `[inicio, fin]`.