Métodos Numéricos

Yuriy Kovalenko.

Dr. en Física y Matemática.

ForEstudiantes@gmail.com

Materiales del curso

https://www.courses.is-great.org

Organización del curso. 75 horas + tareas.

Asistencia<80% = examen extraordinario

Tareas < 80% = examen extraordinario

Examen parcial = 60%, Asistencia + tareas + preguntas=40%

Contenido del curso.

Tema 1. Introducción al curso. Entorno Scilab. Presentación de los números reales en las computadoras. Errores de calculo.

Tema 2. Resolución de sistemas de ecuaciones lineales. Métodos directos e iterativos.

Tema 3. Solución de ecuaciones y sistemas de ecuaciones no lineales.

Tema 4. Aproximación. Interpolación. Ajuste de curvas.

Tema 5. Diferenciación numérica.

Tema 6. Integración numérica.

Tema 7. Optimización.

Tema 8. Ecuaciones diferenciales ordinarias.

Tema 9. Ecuaciones diferenciales parciales.

Tema 1. Introducción al curso. Presentación de los números en las computadoras. Errores de cálculo.

1.1. ¿Por qué se necesitan los métodos numéricos?

1.2. Entorno de cálculos y lenguaje de programación SCILAB.

1.3. Los errores en los cálculos de ordenador.

1.4. Representación de los números reales en un ordenador

1.5. Errores de redondeo.

1.6. La pérdida de dígitos significativos en sustracción.

1.7. Errores de truncamiento.

1.8. Propagación de los errores.

¿Por qué se necesitan los métodos numéricos?

Porque **no todos los problemas se pueden resolver analíticamente**.

Sabemos que una ecuación cuadrática $x^2 + px + q = 0$ tiene una solución en forma analítica

$$x = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}$$

Dicen que **la ecuación tiene una solución analítica, si sus raíces se pueden expresar a través de los valores (miembros) de la ecuación mediante operaciones aritméticas, la extracción de raíces, exponenciales, logaritmos, funciones trigonométricas y trigonométricas inversas.**

También hay **una fórmula** (**solución analítica**) para la ecuación de grado 3

$$x^3 + px + q = 0$$

Pero es justificado que **para las ecuaciones de grado mayor que el quinto** $n \geq 5$, **es imposible expresar la solución utilizando sólo las operaciones de la aritmética y la extracción de raíces**.

¿Por qué se necesitan los métodos numéricos?

Afortunadamente, **para la mayoría de los propósitos** es suficiente encontrar **soluciones en forma de un número**, y no como una fórmula. Es decir, en lugar de soluciones analíticas (soluciones en forma de una fórmula) **es suficiente tener solución numérica del problema**.

En problemas como la **optimización** o el **control** nosotros debemos **examinar un sistema** con diversas combinaciones de las condiciones externas y parámetros internos. Sin embargo, **no todos los sistemas** del mundo real **permiten jugar con ellos**. Por ejemplo, jugar con una planta grande es un poco costoso, y jugar con la bomba atómica es un poco peligrosos. Por lo tanto, **usualmente jugamos solo con modelos matemáticos de sistemas reales**, o simulamos los sistemas reales con computadora.

El **análisis numérico** o **cálculo numérico** es la rama de las matemáticas que se encarga de **diseñar algoritmos para modelos matemáticos** que **simulan los procesos y sistemas del mundo real**.

Entorno de cálculos y lenguaje de programación SCILAB

Para probar los algoritmos que desarrollamos vamos a usar **Scilab.** Scilab es una alternativa gratuita de Matlab.

Preguntas: ¿Quien tiene experiencia en cualquier lenguaje de programación? ¿Quien tiene experiencia en Scilab? ¿O en Matlab?

Scilab es un entorno de cálculos y lenguaje de programación de alto nivel para cálculo científico, interactivo de libre uso y disponible en múltiples sistemas operativos (Mac OS X, GNU/Linux y Windows) desarrollado por INRIA (Institut National de Recherche en Informatique et Automatique) y la ENPC (École Nationale des Ponts et Chaussées) desde 1990.

Entorno de cálculos y lenguaje de programación SCILAB

Las ventajas del Scilab (y Matlab)

- el lenguaje de programación que utiliza es muy fácil y se enfoca en los problemas matemáticos;
- contiene un gran número de bibliotecas de funciones matemáticas y algoritmos;
- contiene herramientas de visualización de los resultados.

La **desventaja** es que en la realidad entornos de cálculos como Matlab o Scilab **no realizan cálculos efectivos**. Un programa bien escrito en C trabaja mucho más rápido que un programa bien escrito en Scilab. Sin embargo, un programa mal escrito en C se puede ejecutar más lento que un programa bien escrito en Scilab.

Entorno de cálculos y lenguaje de programación SCILAB

Las recomendaciones pueden ser las siguientes: si para resolver el problema el cálculo debe ser **ejecutado una vez, es mejor utilizar Scilab**. Si el problema involucra cálculos grandes y repetidos, tales como problemas de optimización, lo mejor es utilizar un lenguaje de programación de propósito general como C. No hay que empezar de cero, para C existen muchas bibliotecas de los métodos numéricos.

Tarea: Descarga e instala

1. Scilab www.scilab.org

2. wxMaxima http://andrejv.github.io/wxmaxima/download.html

//http://sourceforge.net/projects/wxmaxima/

3. LibreOffice http://www.libreoffice.org/

En nuestras clases vamos a **utilizar Scilab, pero no tenemos tiempo para estudiarlo**. Además, deliberadamente vamos **programar en** Scilab en forma **no efectiva**. Aquellos quienes quieran aprender a trabajar eficientemente en Scilab deben leer los libros pertinentes.

Literatura para el curso

Literatura de Scilab

1. Scilab Numerical Methods Tutorials. APAC-ANU Teaching Module.
2. Introduction to Scilab. Scilab Group, INRIA.
3. Eike Rietsch. An Introduction to Scilab from a Matlab User's Point of View. 2002
4. Stephen L. Campbell, Jean-Philippe Chancelier and Ramine Nikoukhah. Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4., 2009. Springer

Curso de Analisis numerico en web

http://numericalmethods.eng.usf.edu/

Literatura para el curso

Literatura de Métodos Numéricos

1. Chapra, S. C., & Canale, R. P. (2010). Numerical methods for engineers. New York: McGraw-Hill Higher Education.
2. Burden R. & Faires J., Análisis Numérico. Grupo Editorial Iberoamérica (1985)
3. J. Stoer, R. Bulirsch. Introduction to Numerical Analysis. 3ed. Springer. (1993)
4. John H. Mathews, Kurds D. Fink. Numerical Methods Using MATLAB. Prentice Hall.4th ed. (2004)
5. Numerical Recipes in C. Press, W. et al. Cambridge University Press. (1992)
6. David Goldberg, What Every Computer Scientist Should Know About Floating-Point Arithmetic, 1991, Computing Surveys. Association for Computing Machinery, Inc.

Los errores en los cálculos de ordenador.

La principal ventaja de los métodos numéricos es que con ellos puedes **resolver todo**. Por supuesto, cuando eliges el método adecuado para el problema. **Otra ventaja** de los métodos numéricos es que los **cálculos** necesarios pueden hacerse **por una computadora**. Por lo general la computadora lo hace **mucho más rápido que los humanos**. Aunque **la máquina casi no comete errores**, tradicionalmente casi todos los libros sobre los métodos numéricos comienzan con la sección "Errores de cálculos y su propagación". Nosotros también comenzaremos con el **análisis de los errores debidos a las limitaciones de las computadoras**.

Consejo: El entorno Scilab tiene un **sistema de ayuda** integrado. Para obtener ayuda para cualquier función, en Scilab se puede escribir en la línea de comandos la palabra **help** y el **nombre de la función** o sólo **help** para accesar al contenido del sistema de ayuda.

Además, existe un acceso al sistema de ayuda a través el menú de Scilab. Aquí pueden encontrar muchos ejemplos de Scilab y SciCos para resolver problemas diferentes.

Ejemplo (Scilab example01.01.sce): Sistema de ayuda de Scilab

Los errores en los cálculos de ordenador.

Ejemplo (Scilab example01.02.sce): Errores de redondeo

Se puede utilizar Scilab como una calculadora. Si escribimos una expresión en línea de comandos y pulsamos "Enter".

0.1-0.1+0.1-0.1+0.1-0.1

$\text{ans} = 0$

Obtenemos la respuesta (answer), la cual, como era esperado, en este caso es cero.

Naturalmente, se supone que en la siguiente expresión también obtendríamos cero.

(0.1+0.1+0.1)-0.3

Sin embargo, el valor calculado por Scilab aunque está cerca, pero no es igual a cero.

$\text{ans} = 5.551\text{D}-17$. Que significa $5.551 \cdot 10^{-17}$

Los errores en los cálculos de ordenador.

La siguiente expresión tampoco nos da cero

-0.1-0.1-0.1+0.1+0.1+0.1

$\text{ans} = -2.776\text{D}-17 \quad (-2.776 \cdot 10^{-17})$

Aquí el resultado es un número negativo pequeño.

A estos, se les llama errores de redondeo. Surgen porque **el tamaño de la memoria** del ordenador **para almacenar el número es limitada**. Sin embargo, **no todos los números reales se pueden escribir con el número de longitud limitada**. Por ejemplo, la proporción de 1 a 3 es el número de longitud ilimitada en notación decimal.

$$\frac{1}{3} = 0.333333(3)$$

Pregunta: ¿Quien sabe que es un **bit** y un **byte** y cual es deferencia entre ellos?

Representación de los números reales en un ordenador

Bit es el acrónimo de **Bi**nary digi**t**. (dígito binario). Un bit es un dígito del sistema de numeración binario. **El bit es la unidad mínima** de información. Con él, podemos representar dos valores 0 y 1. **Byte es un grupo de 8 bits**.

El Sistema de numeración binario se describe en cualquier libro sobre los métodos numéricos o informática.

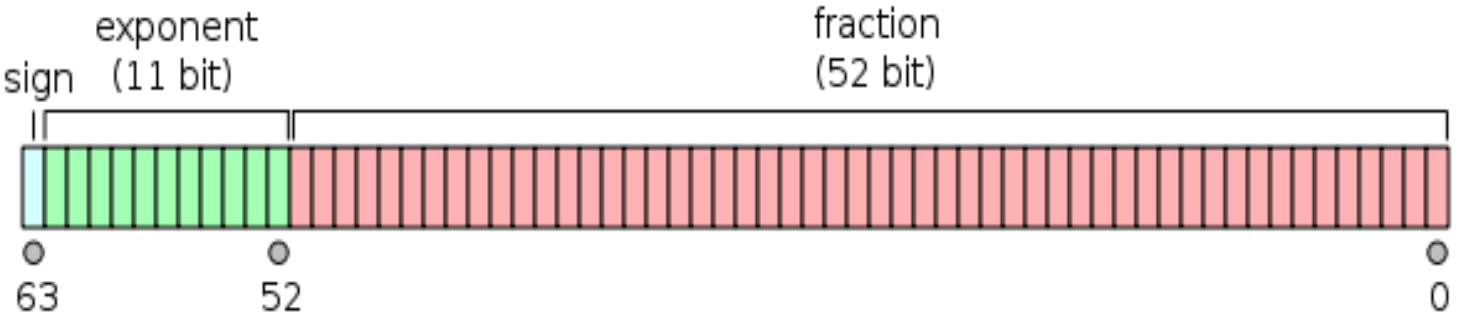
Más información sobre la presentación de los números reales en un ordenador se encuentra en el siguiente artículo.

David Goldberg, *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, 1991, Computing Surveys. Association for Computing Machinery, Inc.

Representación de los números reales en un ordenador

Entornos de computación como Matlab y Scilab **usan para almacenar los números reales** grupos de 64 bits de acuerdo a un estándar

IEEE 754 (Floating point, double precision=64 bits).



De estos 64 bits: Sign bit: 1 bit (signo de número)

Exponent width: 11 bits (exponente)

Significand precision: 52 bits (mantisa)

Para formar el número real se usa la siguiente fórmula

$$(-1)^S \cdot 2^{e-1023} \cdot (1-f)\ ,$$

donde S es el signo del número, e – el exponente y f – la mantisa.

Representación de los números reales en un ordenador

El rango de números reales en este formato es:

$5.562684646268003 \cdot 10^{-309}$ a $8.988465674311580 \cdot 10^{307}$

con **cerca de 16 dígitos decimales de precisión numérica**.

La **longitud limitada de una representación de los números reales en computadora** (64 bits) **limita la cantidad de números que una computadora puede mantener sin alteraciones**.

El espacio de los números en una computadora es discreto!

Punto flotante

Numero	**Representación con punto flotante**
123456=	0.123456*10^6
123.456=	0.123456*10^3
1.23456=	0.123456*10^1
0.0123456=	0.123456*10^-1

Representación de los números reales en un ordenador

La representación del numero 1_{10} en notación hexadecimal es

$$0x3ff0\,0000\,0000\,0000_{16} = 1_{10}$$

los siguientes números > 1.0 son

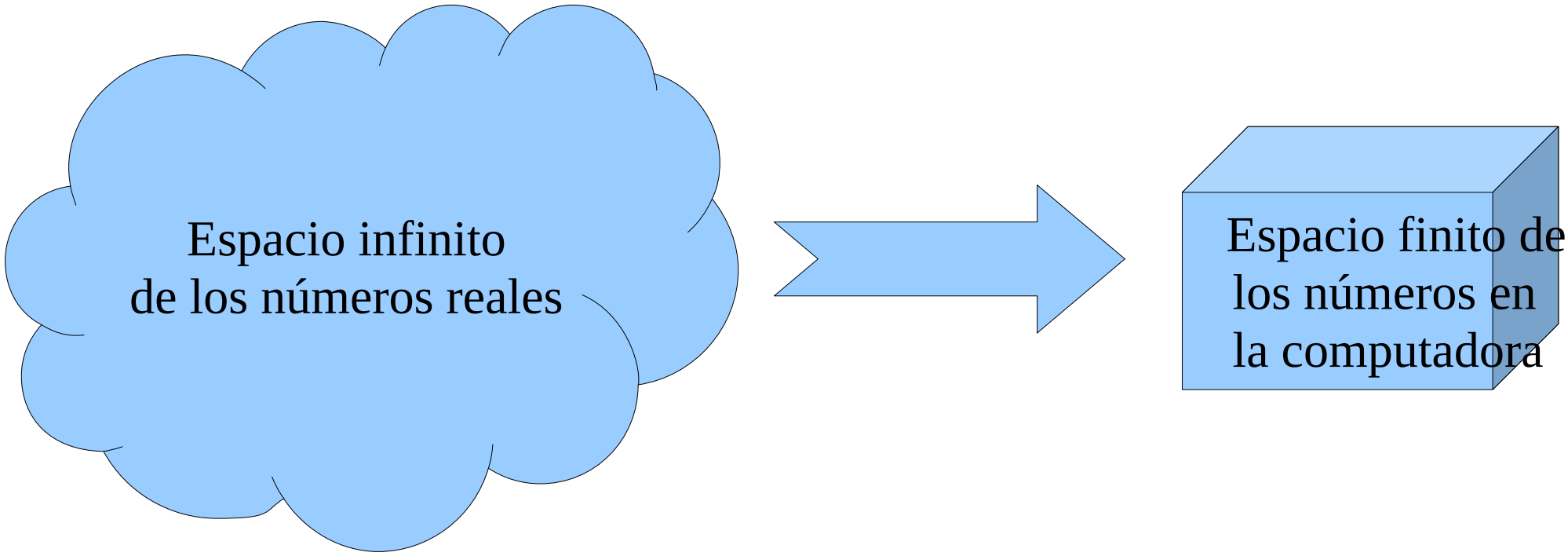
$$0x3ff0\,0000\,0000\,0001_{16} = 1.0000000000000002_{10}\ ,$$

$$0x3ff0\,0000\,0000\,0002_{16} = 1.0000000000000004_{10}$$

La computadora no puede representar el número situado entre los números 1.0000000000000002 y 1.0000000000000004 **sin alteraciones.**

Errores de redondeo

Es decir, el ordenador **debe utilizar un conjunto de números limitado para mostrar un conjunto ilimitado** de números reales.



Es evidente que esto no es posible. Por lo tanto, la computadora utiliza la aproximación de los números que no están incluidos en el espacio limitado por 64 bits. Esto se conoce como **errores de redondeo**.

Errores de redondeo

Los errores de redondeo son más notables si estamos trabajando con números grandes.

Ejemplo (Scilab example01.03.sce):

x = 9007199254740994.0

y = 1.0 - 1/65536.0

(x + y)-x

$\text{ans} = 0$.

Aquí la computadora en lugar de casi 1 nos da un valor de 0

Errores de redondeo

Ejemplo(Scilab example01.04.sce): En este ejemplo vemos que un equipo puede perder un número aún mayor.

$a = 2^{100}$

$\text{ans} = 1.268\text{D}+30$

$b = 2^{47}$

$\text{ans} = 1.407\text{D}+14$

$c = a + b$

$\text{ans} = 1.268\text{D}+30$

La pérdida de dígitos significativos en sustracción.

La longitud limitada de la mantisa conduce a los errores llamados **pérdida de dígitos significativos**.

Ejemplo:

Consideramos los dos números

$p = 3.1415926536$ y

$q = 3.1415957341$,

que son casi iguales **y ambos tienen 11 dígitos decimales de precisión**. Suponemos que la diferencia de ellos esta representada como: $p - q = -0.0000030805$. Como los primeros seis dígitos de p y q son los mismos, **su diferencia** $p - q$ **contiene sólo 5 dígitos decimales de precisión**. Esto es claro si escribimos el resultado en el formato de punto flotante $p - q = -0.30805 \cdot 10^{-5}$

Los errores de este tipo **son inevitables**. A veces se puede reducir el valor del error por el cambio del orden de operaciones del cálculo o por cambio de la forma de la expresión.

Errores de truncamiento

El concepto de error de truncamiento se refiere generalmente a los errores introducidos **cuando una expresión matemática más complicada es "reemplazada" con una fórmula más elemental**. Esta terminología se origina en la técnica de la **sustitución de una función complicada con una serie de Taylor truncada**. Por ejemplo, la infinita serie de Taylor

$$e^{x^2} = 1 + x^2 + \frac{x^4}{2!} + \frac{x^6}{3!} + \frac{x^8}{4!} + \ldots + \frac{x^{2n}}{n!}$$

podría sustituirse con sólo los primeros cinco términos

$$e^{x^2} \approx 1 + x^2 + \frac{x^4}{2!} + \frac{x^6}{3!} + \frac{x^8}{4!}$$

En contraste con los errores anteriores, **estos tipos de errores se pueden descubrir y corregir fácilmente**. Sólo se tiene que calcular el resultado con un gran número de términos de expansión.

Propagación de los errores

Pequeños errores de redondeo en el inicio de los cálculos pueden llevar a errores significativos en el resultado final en el cálculo de gran tamaño. A esto se le llama propagación de los errores.

La suma de dos variables $p=\hat{p}+e_p$ y $q=\hat{q}+e_q$ donde $\hat{p}$ y $\hat{q}$ son valores aproximados y e_p y e_q son errores de aproximación nos da la suma de errores en el resultado final.

$$p+q=(\hat{p}+\hat{q})+(e_p+e_q)$$

Su producto puede dar un valor de error más grande.

$$p\cdot q=(\hat{p}+e_p)\cdot(\hat{q}+e_q)=\hat{p}\,\hat{q}+\hat{p}\,e_q+\hat{q}\,e_p+e_p\,e_q$$

Al fin de un cálculo grande podemos recibir un resultado que esta muy diferente del resultado correcto.

Ejemplo

$p=10\pm0.1$. Calcular error de operación $p-2\,p+3\,p^2$

$$error=0.1+2*0.1+3*\left(0.1^2+10*0.1+10*0.1\right)=6.33$$

Preguntas de autoevaluación

1. Por qué se necesitan los métodos numéricos?
2. Cuando la ecuación tiene una solución analítica?
3. Representación los números en entornos de computación como Matlab y Scilab. Estandarte IEEE 754 (doble precisión).
4. El origen de los errores de redondeo?
5. La pérdida de dígitos significativos en sustracción.
6. Errores de truncamiento
7. Propagación de los errores