Métodos Numéricos y Optimización
Preguntas Parcial 1

24 de febrero de 2020

1. Análisis numérico y cómputo científico.

1. ¿Cuáles son las características del análisis numérico y cómputo científico?

   Cómputo Científico: Es parte de lo que se conoce como ciencia de la computación y **se encarga del desarrollo de un algoritmo, su implementación en un software y simulaciones numéricas del fenómeno con el software**. Es decir, utiliza los algoritmos numéricos y software para aplicaciones tales como: análisis y ajuste de modelos a datos, optimización y machine learning.
   Uno de los aspectos más importantes del cómputo científico es: encontrar algoritmos iterativos que converjan rápidamente, así como dar una estimación de la exactitud de las aproximaciones calculadas.

   Análisis numérico: se enfoca al diseño y análisis de algoritmos para resolver problemas que surgen en la ciencia de la computación e ingeniería.
   Se distingue de otras áreas de la ciencia de la computación en primer lugar porque aunque consideran problemas con cantidades continuas como la velocidad y temperatura presentes naturalmente, en la práctica se debe trabajar con recursos finitos y cantidades discretas. Y en segundo lugar, porque analiza las aproximaciones calculadas y sus efectos.

2. ¿Qué propiedades debe tener un buen algoritmo?

   Son algoritmos iterativos que **convergen** rápidamente y proveen una estimación de la **exactitud** de las aproximaciones calculadas. En general, son algoritmos "buenos"los que sean:

   a) **Eficientes**: es decir, hacen más con menos, reduciendo el costo computacional (medido en términos del número de operaciones que realiza el algoritmo);

b) **Confiables**;
c) **Exactos** ante una serie de aproximaciones realizadas: **estables**: *i)* no amplifican errores generados por las aproximaciones o resultados calculados durante la ejecución del algoritmo; *ii)* ante perturbaciones en los datos de entrada, el algoritmo calcula soluciones cercanas a los datos no perturbados (*análisis de sensibilidad*).

3. ¿Cuáles fuentes de error principalmente son estudiadas por el análisis numérico y cómputo científico al resolver un problema?

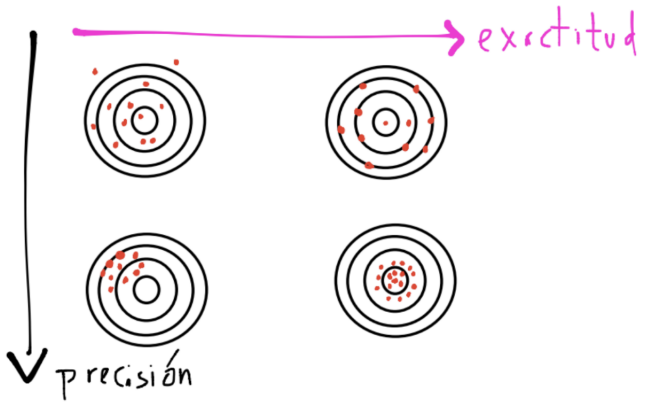
• **Truncamiento**: relacionado con el uso de procesos o cantidades finitas. Es la **diferencia** entre el **resultado verdadero y** el resultado que se obtiene **con un algortimo usando aritmética exacta**. Ejemplos: reemplazar una serie infinita (resultado verdadero) con una serie truncada (algoritmo: truncar serie infinita) y evaluar la serie truncada con aritmética exacta (por ejemplo usar en los cálculos $\frac{1}{3}$ en lugar de 0.3333)

• **Redondeo:** relacionado con la representación de los números y la aritmética realizada en una máquina. Es la **diferencia** entre el **resultado** obtenido por un algoritmo utilizando **aritmética exacta y** el resultado producido por el mismo algoritmo usando **aritmética de máquina**.

Estas, influyen en la exactitud de un cálculo y las perturbaciones que resulten de estas fuentes, serán amplificadas (o no) por la naturaleza del problema y el tipo de algoritmo utilizado. Al estudio de la exactitud y estabilidad de un algoritmo por fuentes de error se le llama análisis del error.

4. Menciona las diferencias entre los términos exactitud y precisión.

Los errores en los cálculos y las medidas se caracterizan con respecto a su **precisión** y **exactitud**.
La **exactitud** se refiere a qué tan cercano está el valor calculado o medido del valor verdadero; la precisión se refiere a la **dispersión** de los valores obtenidos de mediciones repetidas de una magnitud (es decir, qué tan cercanos se encuentran unos de otros diversos valores calculados o medidos).



2. Sistema de punto flotante.

1. ¿Cuáles componentes definen a un sistema de punto flotante?

   a) **Rango de un exponente** definido por un límite inferior y uno superior.

   b) **Base** del sistema.

   c) **Precisión**.

   Así, un número x en el SPF, $x \in \mathcal{F}l$, se representa de la forma:

   $$\pm 0.d_1 d_2 \dots d_k \times \beta^n.$$

   donde:

   - n es el exponente, $n \in [L, U] \cap \mathbb{Z}$ con L, U fijos.
   - k es la precisión.
   - β es la base.
   - $d_i \in \{0, 1, \dots, \beta - 1\} \forall i = 1, \dots, k$ son los dígitos.

   Obs: a la parte $\pm 0.d_1 d_2 \dots d_k$ se le llama mantisa. A la porción $d_1 \dots d_k$ se le llama fracción f.

2. Si un número tiene una representación exacta en la máquina, ¿qué nombre recibe?

   Los números *reales* que tienen una representación exacta en el $\mathcal{F}l$ reciben el nombre de **números de máquina**.

3. ¿Qué es un sistema de punto flotante normalizado?

   Formalmente es aquel SPF que además cumple que $d_1 \in \{1, 2, \dots, \beta - 1\}$ para los números distintos de cero. Donde d_1 es el primer dígito de la fracción que representa a un número x en el SPF:

   $$\pm 0.d_1 d_2 \dots d_k \times \beta^n.$$

$\beta = 10$, $k = 3$, rango del exponente en $[-3, 3] \cap \mathbb{Z}$, entonces algunos números en el SPFN:

Notación de punto flotante	Mantisa	Exponente	Valor de punto fijo
0.153×10^0	0.153	0	0.153
-0.990×10^2	-0.990	2	-99.0
0.343×10^{-3}	0.343	-3	0.000343

4. Menciona algunas propiedades de un sistema de punto flotante normalizado.

   a. El número cero es el único que tiene dígitos de la mantisa y exponente iguales a cero.
   b. Se utiliza $(.)_{10}$ para representar un número en base 10 pero típicamente se omite escribir paréntesis y la base.
   c. La representación normalizada permite una representación única de los números reales en el $\mathcal{F}l$. Por ejemplo: el número 0.343×10^{-3} está en $\mathcal{F}l$ pero ni 0.0343×10^{-2} ni 3.43×10^{-4} están en el $\mathcal{F}l$.

   d. Por lo anterior, en un SPFN con $\beta = 2$ no es necesario almacenar el primer bit pues siempre es 1 por lo que sólo se almacenará la parte fraccionaria de la forma $1.f$

5. ¿Cuántos bits se utilizan en el hardware de una máquina para almacenar un número en un sistema de doble precisión?

   En una máquina convencional, se utilizan 64 bits para representar un número real en un sistema punto flotante de doble precisión. Estos tienen la siguiente estructura:

   a) El primer bit es un indicador de signo denotado por s
   b) Le siguen 11 bits para construir al exponente c (también llamado **característica**).
   c) Finalmente, los últimos 52 bits construyen la parte fraccionaria f de la mantisa. La base es: $\beta = 2$

6. ¿Qué nombre reciben los errores que se generan por utilizar un sistema de punto flotante?

   En un sistema de punto flotante (SPF) existen huecos entre cada número de máquina. Lo anterior implica que al ingresar un número real x en la computadora, ésta realiza una aproximación a x que se encuentre en el SPF. Esta aproximación genera errores conocidos con el nombre de **errores por redondeo**.

7. ¿Cuáles reglas utiliza la máquina para dar aproximaciones a un número?

   La regla de corte y la regla de redondeo. Las cuales se pueden representar con funciones matemáticas como sigue:

Regla de corte: sea $fl_c : \mathbb{R} \to \mathcal{F}l$ una función cuya regla de correspondencia es: $x \in \mathbb{R}$ con x en el rango de valores del SPF, entonces: $x = \pm 0.d_1 d_2 \ldots d_k d_{k+1} \cdots \times \beta^n$ y la regla de corte a k dígitos es: $fl_c(x) = \pm 0.d_1 d_2 \ldots d_k$.

Regla de redondeo: sean $\beta = 10$ y $fl_r() : \mathbb{R} \to \mathcal{F}l$ una función cuya regla de correspondencia es: $x \in \mathbb{R}$ con x en el rango de valores del SPF, entonces: $x = \pm 0.d_1 d_2 \ldots d_k d_{k+1} \cdots \times \beta^n$ y la regla de redondeo es:

$$f_r(x) = \begin{cases} \text{sumar uno a } d_k & \text{si } d_{k+1} \geq 5, \\ f_c(x) & \text{en otro caso} \end{cases}$$

Observación: también la regla $fl_r(\cdot)$ se define como antes pero se añade la restricción entre si es par o impar el último dígito en caso de empate, entonces se almacena par.

8. Explica con palabras la diferencia entre el epsilon de la máquina y el nivel de underflow:

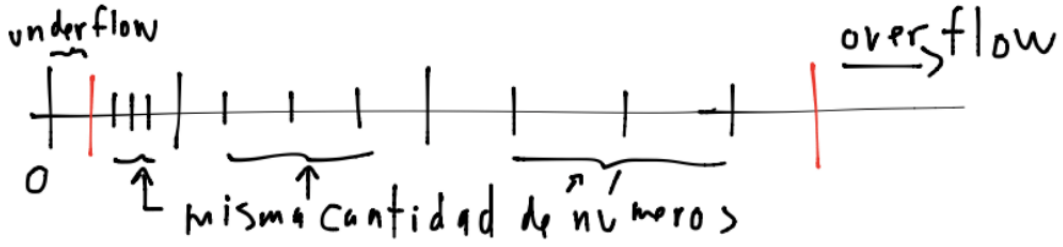
El número de máquina **normalizado** más pequeño positivo es:

$$2^{-1022}(1+0) \approx 2.2251 * 10^{-308}$$

Números con magnitud más chica que el valor anterior resultan en **underflow** y las computadoras les asignan el número cero; por su parte, el ϵ_{maq} nos dice el nivel de precisión o granularidad de un SPF.

Una definición puntual de ϵ_{maq}: el número más chico positivo que: $1 + \epsilon maq \neq 1$

En un sistema de punto flotante de doble precisión $\epsilon_{maq} \approx 10^{-16}$, indicándonos que tenemos aproximadamente 16 dígitos correctos al representar números reales en el SPF; en este sentido, el error relativo nos ayuda a decir aproximadamente el número de dígitos correctos de una aproximación.



a) ¿Cuál de ellos depende únicamente del número de dígitos de la mantisa? El epsilon de la máquina.

b) ¿Cuál de ellos depende únicamente del número de dígitos del exponente? El nivel de underflow.

c) ¿Cuál de ellos no depende de la reglas usadas que se preguntaron en la pregunta 7 (regla de corte y redondeo)? El nivel de underflow no depende de las reglas de corte ni de las reglas del redondeo, ya que los huecos entre cada número de máquina tienen que ver con el ϵ_{maq}. Cuando se ingresa un número real x en la computadora, ésta realiza una aproximación a x que se encuentre en el SPF. Esta aproximación genera errores conocidos con el nombre de errores por redondeo.

$$\epsilon_{maq} = \begin{cases} \beta^{1-k} & \text{si se utiliza regla de corte,} \\ \frac{1}{2}\beta^{1-k} & \text{si se utiliza la regla de redondeo} \end{cases}$$

9. Si calculamos un error relativo para una aproximación y resulta ser del orden de 10^{-8} ¿alrededor de cuántos dígitos correctos tengo en mi aproximación?

   Se tiene que si el ErrRel(aproximación)$\approx 10^{-k}$, entonces la aproximación al objetivo cuenta con alrededor de k dígitos correctos. Por ello, para este caso la aproximación tiene alrededor de 8 dígitos correctos.

10. Menciona algunos problemas típicos de la aritmética de máquina y algunas formas de resolverlos.

    Entre los **problemas** que se pueden encontrar se encuentran:

    - **Problema de cancelación**: pérdida de cifras significativas a partir de la resta de números similares.
    - Suma entre un número de magnitud grande y un número de magnitud pequeña.
    - Sumas con términos que involucren signos alternados.
    - Multiplicación por un número de magnitud grande.
    - División por un número de magnitud pequeña.

    Algunas de las **soluciones** posibles que se tienen para enfrentar estos problemas:

    a) Usar mayor precisión.
    b) Reordenar operaciones.
    c) Reescribir expresiones matemáticas para obtener expresiones equivalentes.
    d) Escalar las variables. También funciona estandarizarlas.

3. Condición, estabilidad y normas.

1. ¿Qué factores influyen en la falta de exactitud de un cálculo?

   La falta de exactitud se presenta por **problemas mal condicionados** (no importando si los algoritmos son estables o inestables) y **algoritmos inestables** (no importando si los problemas son mal o bien condicionados).

2. Menciona 5 propiedades que un conjunto debe cumplir para que sea considerado un espacio vectorial.

   Un conjunto $\mathcal{V}$ no vacío ($\emptyset$) en el que se han definido las operaciones ($+$, $\cdot$) se le llama espacio vectorial sobre $\mathbb{R}$ si satisface las siguientes propiedades $\forall\ x, y$ y $z \in V$, $\forall\ a, b \in \mathbb{R}$

   - $x + (y + z) = (x + y) + z\ \rightarrow$ **Propiedad asociativa de la suma**
   - $x + y = y + x\ \rightarrow$ **Propiedad conmutativa de la suma**
   - $\ni\ 0 \in \mathcal{V}\ t.q.\ x + 0 = x\ , \forall x \in \mathcal{V}$
   - $\forall x \in \mathcal{V}, \ni\ -x \in \mathcal{V}\ t.q.\ x + (-x) = 0$
   - $a(bx) = ab(x)\ \rightarrow$ **Propiedad asociativa de la multiplicación**
   - $1x = x, 1 \in \mathbb{R} \rightarrow$ **Neutro la multiplicación**
   - $a(x+y) = ax+ay\ \rightarrow$ **Propiedad distributiva respecto de la suma vectorial**
   - $(a + b)x = ax + bx$

3. Menciona las propiedades que debe cumplir una función para que se considere una norma.

   Sea g una función tal que g: $\mathrm{R}^n \rightarrow \mathbb{R}$ entonces para que g sea una norma, satisface:

   a) g es no negativa: $g(x) \geq 0, \forall x \in \mathbb{R}^n$.

   b) g es definida: $g(x) = 0\ \iff\ x = 0$.

   c) g satisface la desigualdad del triángulo: $g(x + y) \leq g(x) + g(y),\ \forall x, y \in \mathbb{R}^n$

   d) g es homogénea: $g(\alpha x) = |\alpha| g(x),\ \forall \alpha \in \mathbb{R}, \forall x \in \mathbb{R}^n$

   Notación: $g(x) = ||x||$

4. ¿Qué es una norma matricial inducida?, ¿qué mide una norma matricial inducida?

   Entre las normas matriciales más importantes se encuentran las inducidas por normas vectoriales. Estas normas matriciales se definen en términos de los vectores en $\mathbb{R}^n$ a los que se les aplica la multiplicación Ax.

   Dadas las normas vectoriales $||\cdot||_{(n)}, ||\cdot||_{(m)}$ en $\mathbb{R}^n$ y $\mathbb{R}^m$ respectivamente, la norma matricial inducida $||A||_{(m,n)}$ para $A \in \mathbb{R}^{m \times n}$ es

$$||A||_{(m,n)} = \sup_{x \in \mathbb{R}^n} \frac{||Ax||_{(m)}}{||x||_{(n)}}$$

   Esto es, el menor número C para el cual la desigualdad:

$$||Ax||_{(m)} \leq C||x||_{(n)}$$

se cumple $\forall x \in \mathbb{R}^n$.

Así definidas, la norma $||\cdot||_{(m,n)}$ es la **norma matricial inducida** por las normas vectoriales $||\cdot||_{(m)}, ||\cdot||_{(n)}$.

Comentarios:

La norma matricial inducida $||A||_{(m,n)}$ **representa el máximo factor por el cual** A **puede modificar el tamaño de** x **sobre todos los vectores** $x \in \mathbb{R}^n$, es una medida de un tipo de *worst case stretch factor*.

Son definiciones equivalentes:

$$||A||_{(m,n)} = \sup_{x\in\mathbb{R}^n} \frac{||Ax||_{(m)}}{||x||_{(n)}} = \sup_{||x||_{(n)}\leq 1} \frac{||Ax||_{(m)}}{||x||_{(n)}} = \sup_{||x||_{(n)}=1} ||Ax||_{(m)}$$

5. ¿La norma de Frobenius, es una norma matricial inducida?

   La norma de Frobenius para una matriz $A \in \mathbb{R}^{mxn}$ se define como: $||A||_F = \text{tr}(A^TA)^{1/2} = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2}$ ahora bien, una norma matricial inducida se encuentra dada por:

   $$||A||_{(n,n)} = \sup_{x\in\mathbb{R}^n} \frac{||Ax||_{(n)}}{||x||_{(n)}}$$

   Por lo que si tomamos $A \in \mathbb{R}^{2x2}$ la matriz identidad entonces para que fuese inducida:

   $$||A||_{(2,2)} = 1$$

   . Sin embargo,

   $$||A||_F = \sqrt{2}$$

   Por tanto, la norma de Frobenius no es una norma matricial inducida.

6. ¿A qué son iguales sup($\emptyset$), inf($\emptyset$) ? (el conjunto $\emptyset$ es el conjunto vacío)

   $$\sup(\emptyset) = -\infty$$

   $$\inf(\emptyset) = \infty$$

   Explicación: [Tomado literalmente de las notas de clase]

   Si $C \subseteq \mathbb{R}$ entonces $a \subseteq \mathbb{R}$ es una cota superior en C si:

   $$x \leq a, \forall x \in C$$

En $\mathbb{R}$ el conjunto de cotas superiores es $\emptyset, \mathbb{R}$ ó un intervalo de la forma $[b, \infty]$. En el último caso, b se llama mínima cota superior o supremo del conjunto C y se denota $\sup C$. **Por convención** $\sup \emptyset = -\infty$ **y** $\sup C = \infty$ **si** C **no es acotado por arriba.**

Análogamente, $a \in \mathbb{R}$ es una cota inferior en $C \subseteq \mathbb{R}$ si:

$$a \leq x, \forall x \in C$$

El ínfimo o máxima cota inferior de C es $\text{ínf}\, C = -\sup(-C)$. **Por convención** $\text{ínf}\, \emptyset = \infty$ **y si** C **no es acotado por debajo entonces** $\text{ínf}\, C = -\infty$**.**

7. Si f es un problema mal condicionado, ¿a qué nos referimos? Da ejemplos de problemas bien y mal condicionados.

   Un problema es mal condicionado si perturbaciones pequeñas en x conducen a grandes cambios en $f(x)$. Por ello, una manera de determinar si el problema está mal condicionado es calculando el número de condición de una matriz: $||A||||A^{-1}||$

   - Si el numero de condición es $< 10^1$ se considera un problema "bien condicionado"
   - Si el numero de condición está entre 10^1 y 10^2 es "medianamente condicionado"
   - Si el numero de condición es $> 10^3$ es "mal condicionado"

   a)Ejemplo problema mal condicionado:

$$a) \begin{array}{c} x_1 + x_2 = 2 \\ x_1 + 1.001x_2 = 2 \end{array}$$

   Si se perturbara de la siguiente manera:

$$a) \begin{array}{c} x_1 + x_2 = 2 \\ x_1 + 1.001x_2 = 2.001 \end{array}$$

   En este ejemplo el número de condición de la matriz referencia es 4002 (resuelto con el paquete numpy de phyton), por lo que es un problema mal condicionado.

   a)Ejemplo problema bien condicionado:

$$a) \begin{array}{c} x_1 + x_2 = 2 \\ .001x_1 + x_2 = 1 \end{array}$$

   Si se perturbara de la siguiente manera

$$a) \begin{array}{c} x_1 + x_2 = 2 \\ .999x_2 = .998 \end{array}$$

   En este ejemplo el número de condición de la matriz referencia es 2.62 (resuelto con el paquete numpy de phyton), por lo que es un problema bien condicionado.

8. Si f es un problema que resolvemos con un algoritmo g, ¿qué significa:

   a. que g sea estable?

   Se dice que un algoritmo g para un problema f es estable si:

$$\forall x \in \mathbb{X}, \frac{||g(x) - f(\hat{x})||}{||f(\hat{x})||} \leq K_1 \epsilon_{maq}, K_1 > 0$$

   para $\hat{x} \in \mathbb{X}$ tal que $\frac{||x-\hat{x}||}{||x||} \leq K_2 \epsilon_{maq}, K_2 > 0$.

   Esto es, g resuelve un problema cercano para datos cercanos (cercano en el sentido del ϵ_{maq}) independientemente de la elección de x.

   b. que g sea estable hacia atrás?

   Decimos que un algoritmo g para el problema f es estable hacia atrás si:

$$\forall x \in \mathbb{X}, g(x) = f(\hat{x})$$

   con $\hat{x} \in \mathbb{X}$ tal que $\frac{||x-\hat{x}||}{||x||} \leq K_2 \epsilon_{maq}, K_2 > 0$.

   Esto es, el algoritmo g da la solución exacta para datos cercanos (cercano en el sentido de ϵ_{maq}), independientemente de la elección de x.

   En otras palabras, el algoritmo g es estable hacia atrás sólo si la diferencia entre x y $\hat{x}$, en términos relativos, es menor a $K_2 \epsilon_{maq}$ con $K_2 > 0$.

   c. que g sea inestable?

   Que la respuesta f($\hat{x}$) que se obtiene a partir del algoritmo g tiene variaciones no pequeñas cuando los datos de entrada x tienen variaciones pequeñas.
   O bien que al aplicar g se amplifican errores de redondeo en los cálculos involucrados.

9. ¿Qué ventaja(s) se tiene(n) al calcular un error hacia atrás vs calcular un error hacia delante?

   Debemos enfatizar una vez más que si los errores con que conocemos los datos de entrada son grandes, por mucho que utilicemos un algoritmo perfecto no vamos a obtener un resultado con una precisión mejor. Por ello, es importante estudiar como se propagan los errores y cuáles son sus posibles fuentes durante la computación, con el objeto de minimizarlos o al menos acotarlos

   La principal ventaja es que el error hacia atrás no requiere resolver el problema f (para calcular $f(x)$) ni tener información sobre f.

   a) Error hacia adelante (forward error): $\hat{f}(x) - f(x)$, nos indica si $\hat{f}$ obtiene una respuesta **correcta** y **cercana** al valor de f.

   b) Error hacia atrás (backward error): $\hat{x} - x$, que nos indica si la respuesta que obtenemos con $\hat{f}$ es **correcta** para **datos ligeramente** perturbados. En otras palabras, nos dice qué problema el algoritmo realmente resolvió. [1]

[1] https://en.wikipedia.org/wiki/Numerical$_s tability$

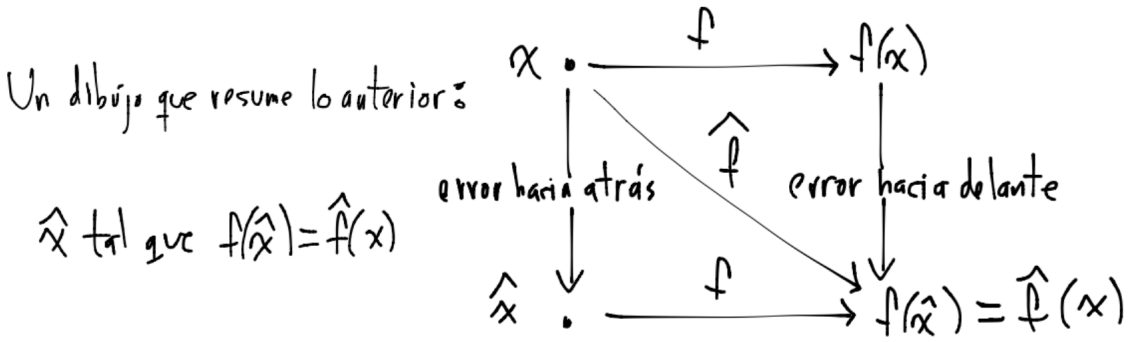


Figura 1: Diferencia entre error hacia adelante y hacia atrás