Métodos Numéricos I

Tema 1: Introducción a los problemas del Análisis Numérico

Miguel A. Piñar
Departamento de Matemática Aplicada
Facultad de Ciencias
Universidad de Granada

23 de febrero de 2022

Contenidos

Sobre el concepto de Análisis Numérico

El Análisis Numérico

es la ciencia que trata del diseño de métodos para la obtención, de una manera eficiente, de soluciones numéricas aproximadas de problemas formulados matemáticamente.

Algoritmo

Es una secuencia finita de operaciones algebraicas y lógicas que producen la aproximación a la solución del problema matemático con una tolerancia o precisión predeterminada. Normalmente se diseña para implementarlo en una máquina

Ejemplo: El cálculo de $\sqrt{2}$

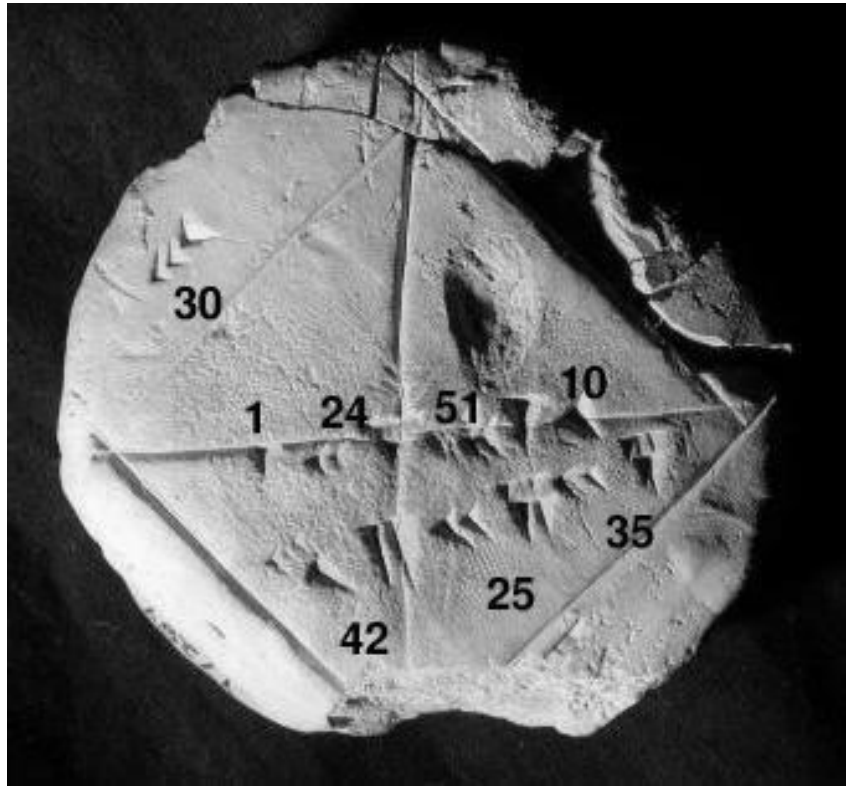
El método de Herón de Alejandría (siglo I a.C.).

$$x_0 = 2, x_1 = \frac{1}{2}\left(x_0 + \frac{2}{x_0}\right) = 1.5,$$

Repetimos el proceso

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{2}{x_n}\right),$$

n	x_n
0	2.0
1	1.5
2	1.41666666666666
3	1.414215686274 51
4	1.41421356237469
5	1.41421356237309 5


30
1
24
51
10
35
25
42

Ejemplo: El algoritmo de Horner

Polinomio:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0,$$

donde

- $a_i \in \mathbb{R}$, $0 \leq i \leq n$,
- $a_n \neq 0$, llamado coeficiente líder o conductor,
- n es el grado del polinomio,
- a_0 es el término independiente.

El algoritmo de Horner (o de la división sintética) se utiliza para evaluar un polinomio en un número real c, minimizando el número de operaciones para minimizar los errores. Se trata de sacar factor común, y disponer los cálculos de forma adecuada.

$$
\begin{aligned}
p(c) &= a_n c^n + a_{n-1} c^{n-1} + \cdots + a_2 c^2 + a_1 c + a_0 = \\
&= (a_n c^{n-1} + a_{n-1} c^{n-2} + \cdots + a_2 c + a_1) c + a_0 = \\
&= ((a_n c^{n-2} + a_{n-1} c^{n-3} + \cdots + a_2) c + a_1) c + a_0 = \\
&= \cdots \\
&= (\cdots (a_n c + a_{n-1}) c + a_{n-2}) c + \cdots + a_2) c + a_1) c + a_0
\end{aligned}
$$

$$
\begin{aligned}
b_n &= a_n, \\
b_{n-1} &= b_n c + a_{n-1}, \\
b_{n-2} &= b_{n-1} c + a_{n-2}, \\
&\dots \\
b_0 &= b_1 c + a_0 = p(c).
\end{aligned}
$$

Algoritmo de Horner

$$
\begin{aligned}
b_n &= a_n, \\
b_i &= b_{i+1} c + a_i, \quad i = n-1, \dots, 0, \\
p(c) &= b_0.
\end{aligned}
$$

Ejemplo Calcular $p(-2)$, donde

$$p(x) = 2\,x^5 - 7\,x^3 + 2\,x^2 - x + 1.$$

Aquí, $a_5 = 2$, $a_4 = 0$, $a_3 = -7$, $a_2 = 2$, $a_1 = -1$, $a_0 = 1$.

$$\begin{aligned}
b_5 &= a_5 = 2,\\
b_4 &= b_5 c + a_4 = 2 \times (-2) + 0 = -4,\\
b_3 &= b_4 c + a_3 = (-4) \times (-2) - 7 = 1,\\
b_2 &= b_3 c + a_2 = 1 \times (-2) + 2 = 0,\\
b_1 &= b_2 c + a_1 = 0 \times (-2) - 1 = -1,\\
b_0 &= b_1 c + a_0 = (-1) \times (-2) + 1 = 3 = p(-2).
\end{aligned}$$

Otra disposición de los cálculos

$$
\begin{array}{c|ccccccc}
 & a_n & a_{n-1} & a_{n-2} & \cdots & a_2 & a_1 & a_0 \\
c & & c\,b_n & c\,b_{n-1} & \cdots & c\,b_3 & c\,b_2 & c\,b_1 \\
\hline
 & b_n & b_{n-1} & b_{n-2} & \cdots & b_2 & b_1 & b_0 \quad = p(c)
\end{array}
$$

Así, el algoritmo de Horner no es más que la conocida regla de Ruffini para la evaluación de un polinomio.

Algoritmos

- Un algoritmo recoge las instrucciones que permiten resolver un problema. Los algoritmos se suelen ejecutar en un ordenador y, de hecho, en la mayoría de los casos, no puede ser una persona.
- Al desarrollar y analizar un algoritmo hemos de tener en cuenta los siguientes elementos: complejidad, precisión, estabilidad y efectos de la representación finita de los números reales.

Complejidad de un algoritmo

Medida del tiempo de ejecución y que suele expresarse en términos de un parámetro asociado al problema.

Estabilidad de un algoritmo

Medida de la sensibilidad del algoritmo a la variación de los datos

Error absoluto y relativo

Definición

Si p^* es una aproximación de p, se define el error absoluto mediante la expresión

$$|p - p^*|,$$

y el error relativo se define por

$$\frac{|p - p^*|}{|p|},$$

siempre y cuando $p \neq 0$.

a) Si $p = 0.3000 \times 10^1$ y $p^* = 0.3100 \times 10^1$, el error absoluto es 0.1 y el error relativo es $0.333\tilde{3} \times 10^{-1}$.

b) Si $p = 0.3000 \times 10^{-3}$ y $p^* = 0.3100 \times 10^{-3}$, el error absoluto es 0.1×10^{-4} y el error relativo es $0.333\tilde{3} \times 10^{-1}$.

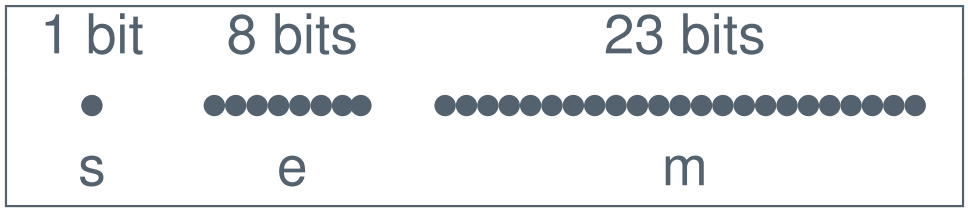
c) Si $p = 0.3000 \times 10^4$ y $p^* = 0.3100 \times 10^4$, el error absoluto es 0.1×10^3 y el error relativo es $0.333\tilde{3} \times 10^{-1}$.

Errores en los cálculos numéricos

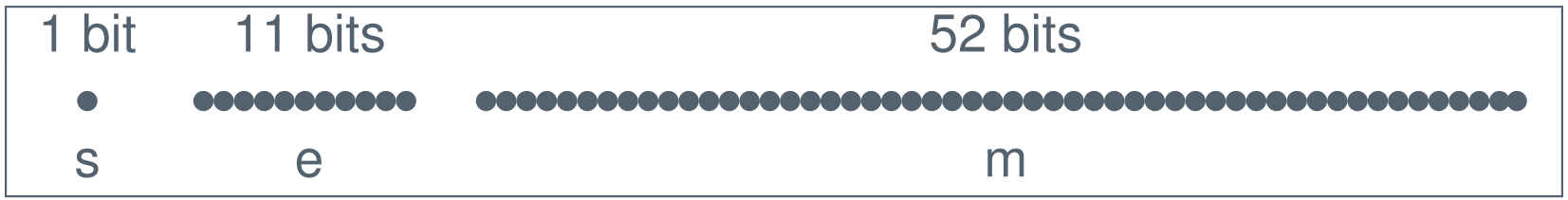
- Cuando se usa un ordenador para realizar cálculos numéricos, el error es inevitable.
- Este error se origina porque la aritmética realizada en una máquina utiliza números con sólo una cantidad finita de dígitos, con el resultado de que los cálculos se realizan con representaciones aproximadas de los números reales.

Representación binaria

Estándar ISO/IEC/IEEE60559:2011, representación binaria, N = 32 bits (binary digits), precisión simple



Estándar ISO/IEC/IEEE60559:2011, representación binaria, N = 64 bits, precisión doble



Sistemas posicionales de numeración

- Sea $b \in \mathbb{N}$ la base de un sistema de numeración (binaria, octal, decimal, hexadecimal, ...)
- Cualquier número real puede escribirse en la forma

$$x = (-1)^s \sum_{n=-\infty}^{N} x_n b^n$$

donde $s \in \{0, 1\}$ representa el signo, $N \in \mathbb{N} \cup \{0\}$ y x_k es la cifra en la posición k tal que $0 \leqslant x_k < b$, $\forall k = -\infty \ldots, N$.

- La representación posicional del número real x es:

$$x_b := (-1)^s \cdot (x_N \ldots x_1 x_0 . x_{-1} x_{-2} \ldots)_b$$

Ejemplo

$x = (101.11)_{10}$, es la representación posicional de $x = 1 \cdot 10^2 + 1 \cdot 10^0 + 1 \cdot 10^{-1} + 1 \cdot 10^{-2}$. $y = (101.11)_2$, es la representación posicional de $y = 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$.

Representación en punto flotante

- En un ordenador, fijada una base b, para representar los números reales se considera sólo un subconjunto finito de números racionales.
- Sea $t \in \mathbb{N}$, el número máximo de dígitos d_n tales que $0 \leqslant d_n \leqslant b-1$
- Sea $e \in \mathbb{Z}$ el exponente, tal que $L \leqslant e \leqslant U$ donde $L, U \in \mathbb{Z},\ L \leq U$ son, respectivamente, el valor mínimo y máximo de los exponentes.
- Sea $s \in \{0, 1\}$
- La representación de un número en punto flotante será

$$(-1)^s b^e \cdot \sum_{n=1}^{t} d_n b^{-n} \sim (-1)^s \cdot (0.d_1 \dots d_t) \cdot b^e$$

Ejemplo

Sea $x = -3.1415$ y la base $b = 10$, su representación con punto flotante con $t = 6$ y $e = 2$ será

$$\begin{aligned} x =& (-1) \cdot 10^2 \cdot \left(0 \cdot 10^{-1} + 3 \cdot 10^{-2} + 1 \cdot 10^{-3} + 4 \cdot 10^{-4} \right. \\ & \left. + 1 \cdot 10^{-5} + 5 \cdot 10^{-6}\right) \\ =& (-1)(0.031415) \cdot 10^2 \end{aligned}$$

Con esta definición un número puede tener varias representaciones.

Ejemplo

Tomemos $b = 2,\ t = 3,\ L = 1,\ U = 3$. Entonces:

$$1 = (0.100) \cdot 2^1 = (0.010) \cdot 2^2 = (0.001) \cdot 2^3$$

Representación normalizada en punto flotante

Para evitar este problema, usaremos la representación normalizada que se obtiene al hacer que la primera cifra de la mantisa sea no nula, es decir $d_1 \neq 0$.

Sistema normalizado de punto flotante

$$\mathbb{F}(b,t,L,U) := \{0\} \cup \left\{ (-1)^s b^e \sum_{n=1}^{t} d_n b^{-n} \right\}$$

donde $s \in \{0,1\}, 0 < d_1 \leq b-1, 0 \leq d_2, \ldots, d_t \leq b-1, L \leqslant e \leqslant U$

Cualquier número real positivo x puede ser normalizado para que adquiera la forma

$$x = (0.d_1 d_2 \ldots d_t d_{t+1} d_{t+2} \ldots) \times b^e.$$

Si suponemos que x está dentro del rango numérico de la máquina, esto es $L \leqslant e \leqslant U$, la representación en punto flotante se obtiene considerando la mantisa de x con t dígitos. Puede relizarse de dos formas distintas

- Mediante truncatura
- Mediante redondeo

Fijado un sistema de punto flotante concreto $\mathbb{F}(b,t,L,U)$ y sea $x \in \mathbb{R}$

$$x = (-1)^s b^e \sum_{n=1}^{\infty} d_n b^{-n} = (-1)^s (0.d_1 d_2 \ldots d_t d_{t+1} \ldots) b^e$$

- La representación por truncatura de x es el número de $\mathbb{F}(b,t,L,U)$ dado por

$$tr(x) := (-1)^s \cdot (0.d_1 \cdots d_t) \cdot b^e$$

- La representación por redondeo de x es el número de $\mathbb{F}(b,t,L,U)$ dado por

$$rd(x) := (-1)^s \cdot (0.d_1 \ldots d_{t-1} r_t) \cdot b^e$$

donde

$$r_t := \begin{cases} d_t & \text{si } d_{t+1} < \dfrac{b}{2} \\ d_t + 1 & \text{si } d_{t+1} \geq \dfrac{b}{2} \end{cases}$$

Ejemplo

El número π tiene un desarrollo decimal infinito

$$\pi = 3.14159265\ldots = (0.314159265\ldots)10^1.$$

Supongamos que $t = 5$ y que se emplea el truncamiento. Entonces la forma de punto flotante de π es

$$tr(\pi) := (0.31415)10^1$$

Como el sexto dígito del desarrollo decimal de π es un nueve, la forma de punto flotante de π usando redondeo a cinco dígitos es

$$rd(\pi) := (0.31416)10^1$$

El error de redondeo

Definición (Épsilon máquina)

Para un sistema de punto flotante $\mathbb{F}(b, t, L, U)$ con $L \leqslant 1 \leqslant U$, el épsilon máquina, que se nota como ϵ_M, es la distancia entre el menor número de $\mathbb{F}(b, t, L, U)$ mayor que 1 y la propia unidad, es decir,

$$\epsilon_M := b^{1-t}$$

Los errores de redondeo o truncatura están acotados.

Proposición

Consideremos el sistema de punto flotante $\mathbb{F}(b, t, L, U)$, con $L \leqslant e \leqslant U$ y sea $x = (-1)^s b^e \sum_{n=1}^{\infty} a_n b^{-n} \in \mathbb{R}$. Entonces:

(*i*) $|x - tr(x)| \leqslant b^{e-t}$

(*ii*) $\dfrac{|x - tr(x)|}{|x|} \leqslant \epsilon_M$

(*iii*) $|x - rd(x)| \leqslant \dfrac{1}{2} b^{e-t}$

(*iv*) $\dfrac{|x - rd(x)|}{|x|} \leqslant \dfrac{\epsilon_M}{2}$

Corolario

Dados $\mathbb{F}(b, t, L, U)$ y $x \in \mathbb{R}$, con $b^{L-1} \leqslant |x| \leqslant b^U(1 - b^{-t})$, se verifica que

$$rd(x) = (1 + \mu)x$$

para cierto $\mu \in \mathbb{R}$ tal que $|\mu| \leqslant \frac{\epsilon_M}{2}$

Definición

El error que resulta de reemplazar un número por su forma de punto flotante se denomina error de redondeo (independientemente de que se use el método de redondeo o el de truncamiento).

Implicaciones de la precisión finita

Proposición

Sean $t \in \mathbb{N},\ L,\ U \in \mathbb{Z}$ con $L \leqslant U$ y $x \in \mathbb{F}(b,t,L,U)$. Entonces:

(*i*) $b^{L-1} \leqslant |x| \leqslant b^U(1-b^{-t})$.

(*ii*) $\text{card}(\mathbb{F}(b,t,L,\ \text{U})) = 2(b-1)b^{t-1}(\text{U}-L+1)+1$.

1. Los números de máquina son discretos y finitos
2. Los números de máquina están acotados
   - Si $|x| < b^{L-1}$: *desbordamiento inferior* (underflow),
   - si $|x| > b^U(1-b^{-t})$: *desbordamiento superior* (overflow).
3. Los números de máquina no están uniformemente distribuidos

Aritmética finita

Las operaciones con números máquina en $\mathbb{F}(b, t, L, U)$ no son necesariamente internas.
En un sistema de cálculo bien diseñado se debería verificar

$$
\begin{aligned}
x \oplus y &= rd(rd(x) + rd(y)),\\
x \ominus y &= rd(rd(x) - rd(y)),\\
x \otimes y &= rd(rd(x) \times rd(y)),\\
x \oslash y &= rd(rd(x)/rd(y)).
\end{aligned}
$$

Las propiedades habituales (asociatividad, elemento neutro) para las operaciones suma, diferencia, multiplicación y división no se cumplen en general.

Ejemplo

Utilizando

$$rd(x) = (0.341) \cdot 10^4, \quad rd(y) = (0.487) \cdot 10^1, \quad rd(z) = (0.492) \cdot 10^1,$$

en un ordenador de tres dígitos con redondeo, probar que no se cumplen las propiedades asociativa y elemento neutro de la suma. Para el elemento neutro de la suma

$$x \oplus y = rd(3410 + 4.87) = rd(3414.87) = (0.341) \cdot 10^4 = rd(x)!!!$$

Para la propiedad asociativa se tiene:

$$x \oplus y = rd(3410 + 4.87) = rd(3414.87) = (0.341) \cdot 10^4$$
$$(x \oplus y) \oplus z = rd(3410 + 4.92) = rd(3414.92) = (0.341) \cdot 10^4$$

Por otro lado,

$$y \oplus z = rd(4.87 + 4.92) = rd(9.79) = (0.979) \cdot 10^1$$
$$x \oplus (y \oplus z) = rd(3410 + 9.79) = rd(3419.79) = (0.342) \cdot 10^4$$

El error en la suma (o en la diferencia)

Sean $x, y \in \mathbb{R}$ tales que $x + y \neq 0$ y sus valores aproximados $rd(x) = (1 + \mu_x)x,\ rd(y) = (1 + \mu_y)y$, entonces

$$(1 + \mu_x)x + (1 + \mu_y)y = x + y + \mu_x x + \mu_y y = (x + y)(1 + \frac{\mu_x x + \mu_y y}{x + y})$$

$$= (x + y)(1 + \frac{x}{x + y}\mu_x + \frac{y}{x + y}\mu_y)$$

Luego

$$\mu_{x+y} = \frac{x}{x + y}\mu_x + \frac{y}{x + y}\mu_y$$

Si x e y tienen el mismo signo se puede controlar el error relativo:

$$|\mu_{x+y}| \leqslant |\mu_x| + |\mu_y|$$

Aunque si x e y tienen signos opuestos, μ_{x+y} puede dispararse si $x + y \approx 0$, generándose un error relativo enorme, conocido como error de cancelación.

El error en el producto y el cociente

Multiplicación.
Los errores relativos de x e y son pequeños.

$$(1+\mu_x)x \cdot (1+\mu_y)y = (1+\mu_x+\mu_y+\mu_x\mu_y)xy \approx (1+\mu_x+\mu_y)xy$$

Luego $\mu_{xy} \approx \mu_x + \mu_y$
Y el error relativo del producto se puede acotar en la forma

$$|\mu_{xy}| \leqslant |\mu_x| + |\mu_y|$$

División.
Los errores relativos de x e y son pequeños, siendo $y \neq 0$.

$$\frac{(1+\mu_x)x}{(1+\mu_y)y} = \frac{x}{y}(1+\mu_x)(1-\mu_y+\mu_y^2-\mu_y^3+\cdots) \approx \frac{x}{y}(1+\mu_x-\mu_y)$$

Por lo que $\mu_{x/y} \approx \mu_x - \mu_y$
Y el error relativo del cociente se puede acotar en la forma

$$|\mu_{x/y}| \leqslant |\mu_x| + |\mu_y|$$

Control del error de cancelación

Las raíces de una ecuación de segundo grado $ax^2 + bx + c = 0$, cuando $a \neq 0$, vienen dadas por las expresiones

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{y} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Consideremos

$$x^2 + 62.10x + 1 = 0,$$

con raíces $x_1 = -0.01610723$ y $x_2 - 62.08390.$
Usando aritmética de redondeo a cuatro dígitos

$$\begin{aligned} \sqrt{b^2 - 4ac} &= \sqrt{(62.10)^2 - 4.000} \\ &= \sqrt{3856. - 4.000} \\ &= \sqrt{3852.} = 62.06. \end{aligned}$$

$$rd(x_1) = \frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{-62.10 + 62.06}{2.000}$$

$$= \frac{-0.04000}{2.000} = -0.02000,$$

Por otro lado

$$rd(x_2) = \frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{-62.10 - 62.06}{2.000}$$

$$= \frac{-124.2}{2.000} = -62.10.$$

Racionalizando el numerador:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \left(\frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} \right)$$

$$= \frac{-2c}{b + \sqrt{b^2 - 4ac}}$$

Se obtiene el resultado exacto:

$$rd(x_1) = \frac{-2.000}{62.10 + 62.06} = \frac{-2.000}{124.2} = -0.01610$$

Propagación del error

Aunque una única operación genere un error pequeño una sucesión finita de operaciones es susceptible de producir la llamada propagación del error, que puede ser considerable.

Dada la sucesión

$$x_n := \int_0^1 x^n e^x dx,$$

Integrando por partes obtenemos

$$\int_0^1 x^n e^x dx = [x^n e^x]_0^1 - n\int_0^1 x^{n-1} e^x dx = e - n\int_0^1 x^{n-1} e^x dx$$

Por lo que $x_n = e - nx_{n-1}$. Si $x \in [0,1]$, entonces $x^n e^x \leqslant x^{n-1} e^x \Rightarrow \{x_n\}_{n\geqslant 0}$ es decreciente y positiva, luego converge a un número real $\ell \geqslant 0$. Ahora bien

$$x_{n-1} = \frac{e - x_n}{n} \Rightarrow \ell = 0.$$

Sin embargo, redondeando en el sistema de punto flotante $b = 10$ y $t = 7$ obtenemos

$$
\begin{aligned}
x_0 &= 2.7182818 \cdot 10^0 \\
x_1 &= 1.0 \cdot 10^0 \\
x_2 &= 7.182818 \cdot 10^{-1} \\
x_3 &= 5.634364 \cdot 10^{-1} \\
x_4 &= 4.645362 \cdot 10^{-1} \\
x_5 &= 3.956006 \cdot 10^{-1} \\
&\vdots \\
x_{15} &= 1.103763 \cdot 10^4 \\
x_{16} &= -1.765994 \cdot 10^5 \\
x_{17} &= 3.002193 \cdot 10^6 \\
x_{18} &= -5.403947 \cdot 10^7
\end{aligned}
$$

El fallo de los misiles Patriot: Dharan, 25-02-91

Veamos de donde proviene el error. $1/10$ en binario no tiene una representación finita

$$\frac{1}{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{1}{2^{12}} + \frac{1}{2^{13}} + \dots$$

$$\frac{1}{10} \longrightarrow 0.0001100110011001100110011001100 \dots$$

Al truncar a 24 bits el error es

$$0.00000000000000000000000011001100_{2)} \dots \approx 0.000000095_{10)}$$

En 100 horas el error será

$$0.000000095 \times 100 \times 60 \times 60 \times 10 = 0.34$$

Y el espacio recorrido por un misil a 1702 m/s (mach 5) será 578.68 m.