강 명주 mkang@snu.ac.kr

25-104

조교: 정혜원 jhw4277@snu.ac.kr

H.W.: 마감시간까지 eTL에 제출

NA_HW#_이름_학번.zip (or pdf)

성적: 과제 (20%) + 중간 (30%) + 기말 (30%)

+출석 (20%) [결석 2%, 지각 2번 (1%)]

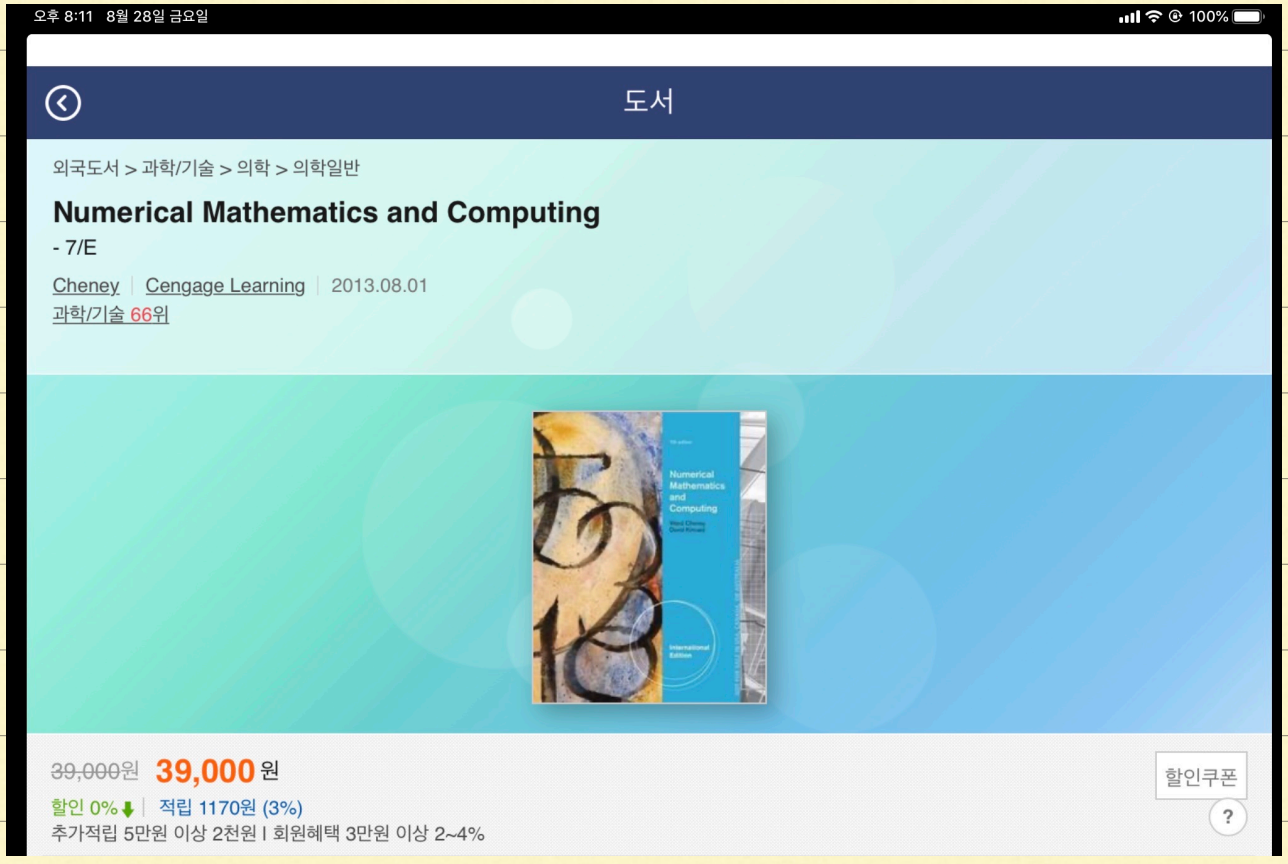
지각: 수업시작후 30분이내, 그 이후는 결석

시험(대면): 중간 4/24(목) 9:30 ~ 10:45

기말 6/10 (화) 9:30 ~ 10:45

시험장소: 25-104

교재



Chap 1. Mathematical Preliminaries and Floating-Point Representation.

Carl Erik Fröberg once remarked
"Never in the history of mankind has it been possible to produce so many wrong answers so quickly"

Primary issues :

- nature of numerical errors
- the propagation of errors
- the efficiency of the computations involved
- the number of operations

Chap1

- Error :

  α : exact , β : approximation

  absolute error : $|\alpha - \beta|$

  relative error : $\dfrac{|\alpha - \beta|}{|\alpha|}$

- Rounding, Chopping

- Floating-Point Representation

  Normalized Floating-Point representation

  $$x = \pm 0.d_1 d_2 d_3 \cdots \times 10^n \quad \leftarrow \text{decimal form}$$

  $d_1 \neq 0$, n integer

  $$= \pm r \times 10^n \quad \left(\frac{1}{10} \le r < 1\right)$$

  r : normalized mantissa

  n : exponent

  binary system

  $$x = \pm q \times 2^n \quad , \left(\frac{1}{2} \le q < 1\right)$$

  $$q = (0.b_1 b_2 \cdots)_2 \quad , \quad b_1 \neq 0$$

  machine number : the real numbers that are representable in a computer

EXAMPLE 1 List all the floating-point numbers that can be expressed in the form

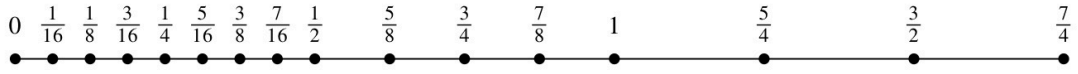
$$x = \pm(0.b_1b_2b_3)_2 \times 2^{\pm k} \qquad (k, b_i \in \{0, 1\})$$

Solution There are two choices for the $\pm$, two choices for b_1, two choices for b_2, two choices for b_3, and three choices for the exponent. Thus, at first, one would expect $2 \times 2 \times 2 \times 2 \times 3 = 48$ different numbers. However, there is some duplication. For example, the nonnegative numbers in this system are as follows:

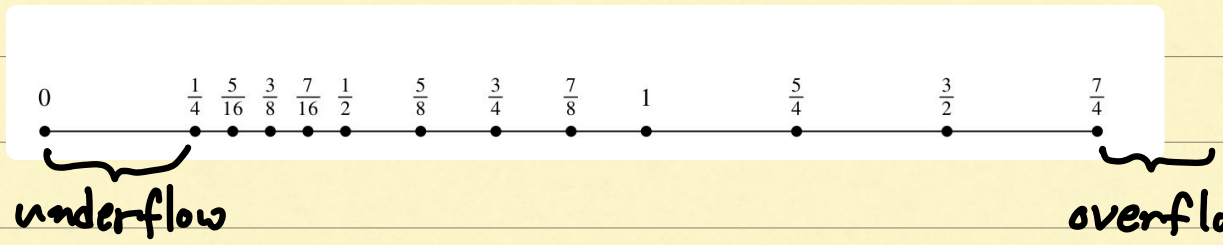
$$\begin{array}{lll}
0.000 \times 2^0 = 0 & 0.000 \times 2^1 = 0 & 0.000 \times 2^{-1} = 0 \\
0.001 \times 2^0 = \frac{1}{8} & 0.001 \times 2^1 = \frac{1}{4} & 0.001 \times 2^{-1} = \frac{1}{16} \\
0.010 \times 2^0 = \frac{2}{8} & 0.010 \times 2^1 = \frac{2}{4} & 0.010 \times 2^{-1} = \frac{2}{16} \\
0.011 \times 2^0 = \frac{3}{8} & 0.011 \times 2^1 = \frac{3}{4} & 0.011 \times 2^{-1} = \frac{3}{16} \\
0.100 \times 2^0 = \frac{4}{8} & 0.100 \times 2^1 = \frac{4}{4} & 0.100 \times 2^{-1} = \frac{4}{16} \\
0.101 \times 2^0 = \frac{5}{8} & 0.101 \times 2^1 = \frac{5}{4} & 0.101 \times 2^{-1} = \frac{5}{16} \\
0.110 \times 2^0 = \frac{6}{8} & 0.110 \times 2^1 = \frac{6}{4} & 0.110 \times 2^{-1} = \frac{6}{16} \\
0.111 \times 2^0 = \frac{7}{8} & 0.111 \times 2^1 = \frac{7}{4} & 0.111 \times 2^{-1} = \frac{7}{16}
\end{array}$$

Altogether there are 31 distinct numbers in the system. The positive numbers obtained are shown on a line in Figure 2.1. Observe that the numbers are symmetrically but unevenly distributed about zero. ■

FIGURE 2.1 Positive machine numbers in Example 1



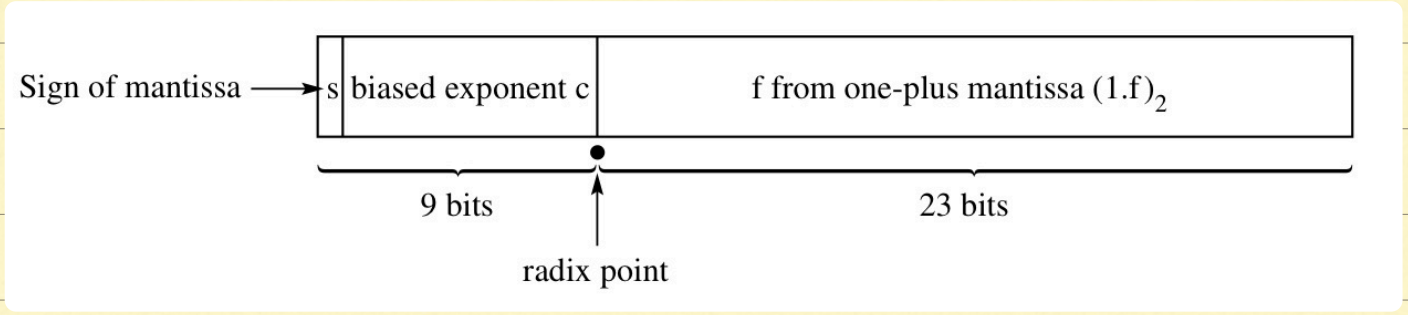
Normalized floating pt. numbers ($b_1 = 1$)



Standard form for Floating point Arithmetic (IEEE-754)

precision	Bits	Sign	Exponent	Mantissa
single	32	1	8	23
Double	64	1	11	52
Long Double	80	1	15	64

Single precision Floating point Form



$$(-1)^s \times 2^{c-127} \times (1.f)_2$$

$s = 0, 1$, $\quad 0 < c < (11\,111\,111)_2 = 255$

$$-126 \leq c-127 \leq 127$$

$0, 255$ reserved for $\pm 0, \pm\infty$

$$1 \leq (1.f)_2 \leq (1.11\cdots1)_2 = 2 - 2^{-23}$$

largest number : $(-1)^0 \times 2^{254-127} \times (2-2^{-23})$

$$\approx 2^{128} \approx 3.4 \times 10^{38}$$

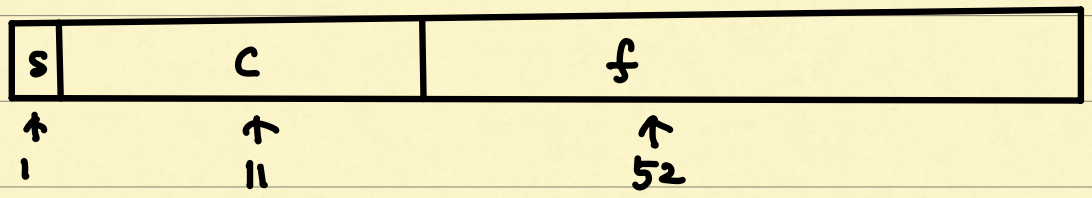
smallest positive number : $(-1)^0 \times 2^{1-127} \times 1$

$$\approx 1.2 \times 10^{-38}$$

machine epsilon $\varepsilon = 2^{-24}$

↳ the smallest positive machine number ε s.t. $1+\varepsilon \neq 1$

Double precision Floating point Form

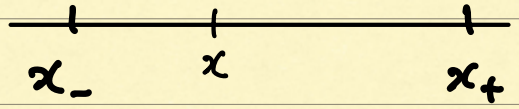


Computer Errors in Representing Numbers

Consider a 32-bit computer.

normalized floating-point form

$$x = q \times 2^m, \quad \left(\frac{1}{2} \le q < 1, \ -127 \le m \le 128\right)$$



rounding
roundoff error

↳ $x = (0.1\, b_2 b_3 \cdots b_{24} \cdots)_2 \times 2^m$

$x_- = (0.1\, b_2 b_3 \cdots b_{24})_2 \times 2^m$

$x_+ = [(0.1\, b_2 b_3 \cdots b_{24})_2 + 2^{-24}] \times 2^m$

x_+ or x_- is a nearest machine number.

$$|x - x_-| \le \frac{1}{2}|x_+ - x_-| = 2^{-25+m}$$

relative error

$$\left|\frac{x - x_-}{x}\right| \le \frac{2^{-25+m}}{(0.1 b_2 b_3 \cdots)_2 \times 2^m} \le \frac{2^{-25}}{1/2} = 2^{-24} = u$$

where $u = 2^{-24}$ is the unit roundoff error for a 32-bit binary computer with standard floating-point arithmetic

machine epsilon (ε)
unit roundoff error (u)

Notation fl(x) and Backward Error Analysis

fl(x): floating pt. machine number

$$fl(x) = x(1+\delta), \quad |\delta| \le 2^{-24}$$ for a single precision

For x, y machine numbers,

$$fl(x \odot y) = (x \odot y)(1+\delta)$$

$$fl(x+y) = (x+y)(1+\delta) = x(1+\delta) + y(1+\delta)$$

- backward error analysis

  It attempts to determine what perturbation of the original data would cause computer results to be the exact results for a perturbed problem

- direct error analysis

  attempts to determine how computed answers differ from exact answers based on the same data.

example 4

x, y, z : machine numbers in a 32-bit computer

$$\begin{aligned} fl(z(x+y)) &= fl[z\, fl(x+y)] \\ &= fl[z(x+y)(1+\delta_2)] \qquad (|\delta_2| \le 2^{-24}) \\ &= z(x+y)(1+\delta_1)(1+\delta_2) \\ &= z(x+y)(1+\delta_1+\delta_2+\delta_1\delta_2) \\ &\simeq z(x+y)(1+\delta_1+\delta_2) \end{aligned}$$

$= z(x+y)(1+\delta) \quad (|\delta| \le 2^{-23})$

examples

x, y : two real numbers , $x+y = ?$

$z = fl(fl(x) + fl(y))$

$= [x(1+\delta_1) + y(1+\delta_2)](1+\delta_3)$

$= (x+y) + \delta_1 x + \delta_2 y + \delta_3 x + \delta_3 y + \delta_1\delta_3 x + \delta_2\delta_3 y$

$\approx (x+y) + x(\delta_1+\delta_3) + y(\delta_2+\delta_3)$

$\Rightarrow$ relative roundoff error :

$$\left|\frac{(x+y)-z}{x+y}\right| = \left|\frac{x(\delta_1+\delta_3)+y(\delta_2+\delta_3)}{(x+y)}\right|$$

$$= \left|\delta_3 + \frac{x\delta_1 + y\delta_2}{x+y}\right|$$

1.4. Loss of Significance

To see the sort of situation in which a large relative error can occur, we consider a subtraction of two numbers that are close to each other. For example,

$$x = 0.37214\,78693$$
$$y = 0.37202\,30572$$
$$x - y = 0.00012\,48121$$

If this calculation were to be performed in a decimal computer having a five-digit mantissa, we would see

$$\text{fl}(x) = 0.37215$$
$$\text{fl}(y) = 0.37202$$
$$\text{fl}(x) - \text{fl}(y) = 0.00013$$

The relative error is then very large:

$$\left|\frac{x - y - [\text{fl}(x) - \text{fl}(y)]}{x - y}\right| = \left|\frac{0.00012\,48121 - 0.00013}{0.00012\,48121}\right| \approx 4\%$$

Whenever the computer must shift the digits in the mantissa to achieve a *normalized* floating-point number, additional 0's are supplied on the right. These 0's are spurious and do not represent additional accuracy. Thus, $\text{fl}(x) - \text{fl}(y)$ is represented in the computer as 0.13000×10^{-3}, but the 0's in the mantissa serve only as placeholders.

Thm 1 **Loss of Precision Theorem**

x, y : normalized floating pt machine numbers $x > y > 0$. If $2^{-p} \leq 1 - (y/x) \leq 2^{-q}$, then at most p and at least q significant binary bits are lost in the subtraction $x - y$.

Example 2

$x = 37.593621$, $y = 37.584216$

$$2^{-12} \leq 1 - \frac{y}{x} = 0.00025\,01754 \leq 2^{-11}$$

By thm, at least 11 but not more than 12 bits are lost.

Example 3

pf of thm

$x = r \times 2^m$, $y = s \times 2^m$, $\frac{1}{2} \leq r, s < 1$, $y < x$

$x - y = (r - s\,2^{m-n}) \times 2^n$

$$r - s\,2^{m-n} = r\left(1 - \frac{s\,2^m}{r\,2^m}\right) = r\left(1 - \frac{y}{x}\right) < 2^{-q}$$

Hence, to normalize the representation of $x - y$, a shift of at least q bits to the left is necessary. Then at least q zeros are supplied on the right-hand end of the mantissa. This means that at least q bits of precision have been lost. //

Avoiding Loss of Significance in Subtraction

Example 4) **Rationalizing**

$$f(x) = \sqrt{x^2+1} - 1 \Rightarrow \frac{x^2}{\sqrt{x^2+1}+1}$$

x near 0

Example 5) **Taylor Series**

$$f(x) = x - \sin x, \quad x \text{ near } 0$$

$$= x - \left(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots\right)$$

$$= \frac{x^3}{3!} - \frac{x^5}{5!} + \cdots$$

Example 7) **formula**

$$y = \cos^2 x - \sin^2 x = \cos(2x)$$

Example 9) **range reduction**

$$\sin(12532.14) = \sin(12532.14 - 2(1994)\pi)$$

$$= \sin(3.49)$$

↗ 7 significant figures

↳ 3 significant

Example (0)

For $\sin x$, how many binary bits of significance are lost in range reduction to the interval $[0, 2\pi)$?

sol) $x > 2\pi$

Need to determine n s.t. $0 \le x - 2n\pi < 2\pi$

By the thm on Loss of Precision, at least g bits are lost if

$$1 - \frac{2n\pi}{x} \le 2^{-g}$$

Since $1 - \frac{2n\pi}{x} = \frac{x - 2n\pi}{x} < \frac{2\pi}{x}$,

$$\frac{2\pi}{x} \le 2^{-g}$$

i.e. we conclude that at least g bits are lost if $2^g \le x/2\pi$. //

오후 8:21 8월 28일 금요일 96%

numerical disaster - Bing | The sinking of the Sleipner A offshore... | www-users.math.umn.edu/~arnold//disasters/sleipner.html

The sinking of the Sleipner A offshore platform

Excerpted from a report of SINTEF, Civil and Environmental Engineering:

The Sleipner A platform produces oil and gas in the North Sea and is supported on the seabed at a water depth of 82 m. It is a Condeep type platform with a concrete gravity base structure consisting of 24 cells and with a total base area of 16 000 m^2. Four cells are elongated to shafts supporting the platform deck. The first concrete base structure for Sleipner A sprang a leak and sank under a controlled ballasting operation during preparation for deck mating in Gandsfjorden outside Stavanger, Norway on 23 August 1991.

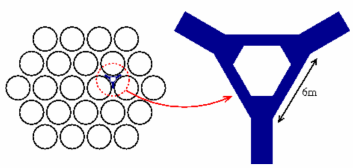
Immediately after the accident, the owner of the platform, Statoil, a Norwegian oil company appointed an investigation group, and SINTEF was contracted to be the technical advisor for this group.

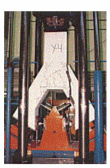
The investigation into the accident is described in 16 reports...

The conclusion of the investigation was that the loss was caused by a failure in a cell wall, resulting in a serious crack and a leakage that the pumps were not able to cope with. The wall failed as a result of a combination of a serious error in the finite element analysis and insufficient anchorage of the reinforcement in a critical zone.

A better idea of what was involved can be obtained from this photo and sketch of the platform. The top deck weighs 57,000 tons, and provides accommodation for about 200 people and support for drilling equipment weighing about 40,000 tons. When the first model sank in August 1991, the crash caused a seismic event registering 3.0 on the Richter scale, and left nothing but a pile of debris at 220m of depth.
The failure involved a total economic loss of about $700 million.

The 24 cells and 4 shafts referred to above are shown to the left while at the sea surface. The cells are 12m in diameter. The cell wall failure was traced to a tricell, a triangular concrete frame placed where the cells meet, as indicated in the diagram below. To the right of the diagram is pictured a portion of tricell undergoing failure testing.





The post accident investigation traced the error to inaccurate finite element approximation of the linear elastic model of the tricell (using the popular finite element program NASTRAN). The shear stresses were underestimated by 47%, leading to insufficient design. In particular, certain concrete walls were not thick enough. More careful finite element analysis, made after the accident, predicted that failure would occur with

---

오후 8:21 8월 28일 금요일 97%

numerical disaster - Bing | The Explosion of the Ariane 5 | www-users.math.umn.edu/~arnold//disasters/ariane.html

The Explosion of the Ariane 5

On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer. The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

The following paragraphs are extracted from the report of the Inquiry Board. An interesting article on the accident and its implications by James Gleick appeared in The New York Times Magazine of 1 December 1996. The CNN article reporting the explosion, from which the above graphics were taken, is also available.

On 4 June 1996, the maiden flight of the Ariane 5 launcher ended in a failure. Only about 40 seconds after initiation of the flight sequence, at an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded.

The failure of the Ariane 501 was caused by the complete loss of guidance and altitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system.

The internal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer.*

*SRI stands for Système de Référence Inertielle or Inertial Reference System.

More disasters attributable to bad numerics

Last modified August 23, 2000 by Douglas N. Arnold

이 페이지를 한국어(으)로 번역할까요? 아니요 번역

The Patriot Missile Failure

On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Dharan, Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people. A report of the General Accounting office, GAO/IMTEC-92-26, entitled *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia* reported on the cause of the failure. It turns out that the cause was an inaccurate calculation of the time since boot due to computer arithmetic errors. Specifically, the time in tenths of second as measured by the system's internal clock was multiplied by 1/10 to produce the time in seconds. This calculation was performed using a 24 bit fixed point register. In particular, the value 1/10, which has a non-terminating binary expansion, was chopped at 24 bits after the radix point. The small chopping error, when multiplied by the large number giving the time in tenths of a second, led to a significant error. Indeed, the Patriot battery had been up around 100 hours, and an easy calculation shows that the resulting time error due to the magnified chopping error was about 0.34 seconds. (The number 1/10 equals $1/2^4+1/2^5+1/2^8+1/2^9+1/2^{12}+1/2^{13}+\ldots$ In other words, the binary expansion of 1/10 is 0.0001100110011001100110011001100.... Now the 24 bit register in the Patriot stored instead 0.00011001100110011001100 introducing an error of 0.0000000000000000000000011001100... binary, or about 0.000000095 decimal. Multiplying by the number of tenths of a second in 100 hours gives $0.000000095\times100\times60\times60\times10=0.34$.) A Scud travels at about 1,676 meters per second, and so travels more than half a kilometer in this time. This was far enough that the incoming Scud was outside the "range gate" that the Patriot tracked. Ironically, the fact that the bad time calculation had been improved in some parts of the code, but not all, contributed to the problem, since it meant that the inaccuracies did not cancel, as discussed here and here.

The following paragraph is excerpted from the GAO report.

The range gate's prediction of where the Scud will next appear is a function of the Scud's known velocity and the time of the last radar detection. Velocity is a real number that can be expressed as a whole number and a decimal (e.g., 3750.2563...miles per hour). Time is kept continuously by the system's internal clock in tenths of seconds but is expressed as an integer or whole number (e.g., 32, 33, 34...). The longer the system has been running, the larger the number representing time. To predict where the Scud will next appear, both time and velocity must be expressed as real numbers. Because of the way the Patriot computer performs its calculations and the fact that its registers are only 24 bits long, the conversion of time from an integer to a real number cannot be any more precise than 24 bits. This conversion results in a loss of precision causing a less accurate time calculation. The effect of this inaccuracy on the range gate's calculation is directly proportional to the target's velocity and the length of the the system has been running. Consequently, performing the conversion after the Patriot has been running continuously for extended periods causes the range gate to shift away from the center of the target, making it less likely that the target, in this case a Scud, will be successfully intercepted.

More disasters attributable to bad numerics

Last modified August 23, 2000 by *Douglas N. Arnold*

Numerical disaster