

Pitfalls in Floating Point Computation —and How to Avoid Them

Nick Higham
Department of Mathematics
University of Manchester

`higham@ma.man.ac.uk`

<http://www.vortex.ma.man.ac.uk/Higham.html>

Book in preparation:
Accuracy and Stability of Numerical Algorithms
(SIAM, 1995)



THE UNIVERSITY
of MANCHESTER

Floating Point Numbers

Floating point number system $F \subset \mathbb{R}$:

$$\begin{aligned} y &= \pm \beta^e \times .d_1 d_2 \dots d_t, & 0 \leq d_i \leq \beta - 1, \\ &= \pm \beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \dots + \frac{d_t}{\beta^t} \right). \end{aligned}$$

Parameters: *base* β , *precision* t , *exponent range* $e_{\min} \leq e \leq e_{\max}$.

Assume $d_1 \neq 0$ if $y \neq 0$, so system is *normalized*.

Alternative:

$$y = \pm m \times \beta^{e-t}, \quad 0 \leq m \leq \beta^t - 1.$$

Assume $m \geq \beta^{t-1}$, so the system is normalized.

To form $y = \text{round}(x)$ ($x > 0$), write

$$x = \mu \times \beta^{e-t}, \quad \beta^{t-1} \leq \mu \leq \beta^t - 1.$$

Then

$$y = \lfloor \mu \rfloor \beta^{e-t} \quad \text{or} \quad y = \lceil \mu \rceil \beta^{e-t}.$$

Three Misconceptions of
Floating Point Arithmetic

1. An Innocuous Calculation?

For any $x \geq 0$ consider

for $i = 1:60$

$$x = \sqrt{x}$$

end

for $i = 1:60$

$$x = x^2$$

end

For any x the 12-digit HP 48G calculator computes, in place of $f(x) = x$,

$$\hat{f}(x) = \begin{cases} 0, & 0 \leq x < 1, \\ 1, & x \geq 1. \end{cases}$$

Explanation

The positive numbers x representable on the HP 48G satisfy $10^{-499} \leq x \leq 10^{500}$.

Defining $r(x) = x^{1/2^{60}}$, for any machine number $x \geq 1$,

$$\begin{aligned} 1 &\leq r(x) < r(10^{500}) = 10^{500/2^{60}} \\ &= e^{500 \cdot 2^{-60} \cdot \log 10} < e^{10^{-15}} \\ &= 1 + 10^{-15} + \frac{1}{2} \cdot 10^{-30} + \dots, \end{aligned}$$

which rounds to 1.

Similar argument for $0 < x < 1$.

Conclusion: nothing wrong with the calculator. This calculation exhausts its precision and range.

Misconception 1 *A short computation free from cancellation, underflow and overflow must be accurate.*

2. Increasing the Precision

“Increasing the precision increases the accuracy”.

For a computation where error comes from rounding usually have a bound of the form

$$\text{error} \leq f(\text{data})u.$$

$y = e^{\pi\sqrt{163}}$ evaluated at t digit precision:

t	y
20	262537412640768744.00
25	262537412640768744.0000000
30	262537412640768743.999999999999

Is the last digit before the decimal point 4?

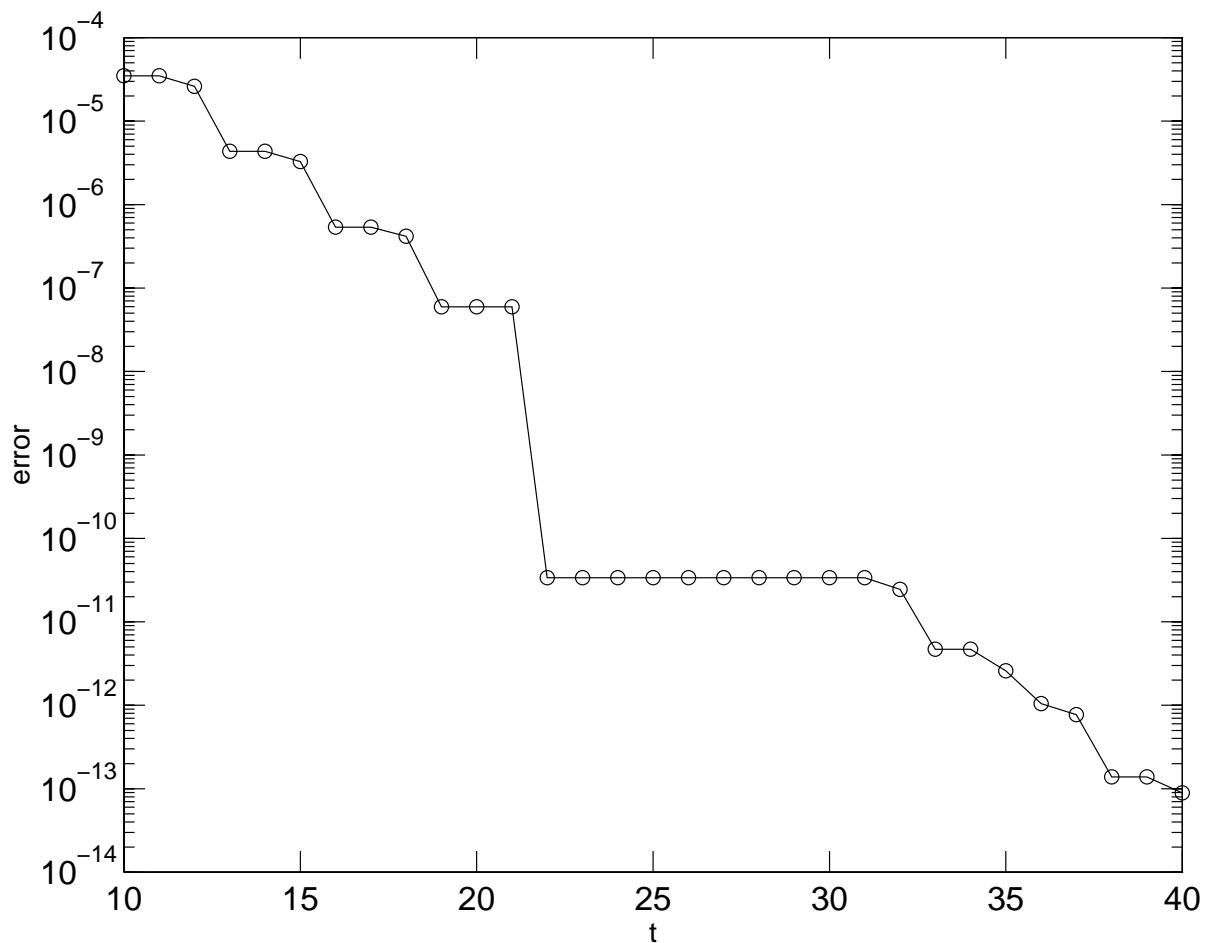
t	y
35	262537412640768743.99999999999925007
40	262537412640768743.9999999999992500725972

So no, it's 3!

Another Example

Consider the evaluation in precision $u = 2^{-t}$ of

$$y = x + a \sin(bx), \quad x = \frac{1}{7}, \quad a = 10^{-8}, \quad b = 2^{24}.$$



Misconception 2 *Increasing the precision at which a computation is performed increases the accuracy of the answer.*

3. Cancellation of Rounding Errors

Let

$$f(x) = \frac{e^x - 1}{x} = \sum_{i=0}^{\infty} \frac{x^i}{(i+1)!}.$$

The obvious way to evaluate f :

```
% Algorithm 1.  
if  $x = 0$   
     $f = 1$   
else  
     $f = (e^x - 1)/x$   
end
```

Alternative:

```
% Algorithm 2.  
 $y = e^x$   
if  $y = 1$   
     $f = 1$   
else  
     $f = (y - 1)/\log y$   
end
```

Some Results

Computing $f(x) = \frac{e^x - 1}{x}$.

x	Algorithm 1	Algorithm 2
10^{-5}	1.0000050000 <u>06965</u>	1.000005000016667
10^{-6}	1.000000 <u>499962184</u>	1.000000500000167
10^{-7}	1.0000000 <u>49433680</u>	1.000000050000002
10^{-8}	<u>0.999999993922529</u>	1.000000005000000
10^{-9}	1.0000000 <u>82740371</u>	1.000000000500000
10^{-10}	1.0000000 <u>82740371</u>	1.000000000050000
10^{-11}	1.0000000 <u>82740371</u>	1.000000000005000
10^{-12}	1.0000 <u>88900582341</u>	1.000000000000500
10^{-13}	<u>0.999200722162641</u>	1.000000000000050
10^{-14}	<u>0.999200722162641</u>	1.000000000000005
10^{-15}	<u>1.110223024625156</u>	1.00000000000000 <u>0</u>
10^{-16}	<u>0</u>	1

A Closer Look

Consider $x = 9 \times 10^{-8}$ and $u \approx 6 \times 10^{-8}$, for which $f(x) = 1.00000005$ to the significant digits shown.

Algorithm 1 gives

$$fl\left(\frac{e^x - 1}{x}\right) \equiv fl\left(\frac{1.19209290 \times 10^{-7}}{9.00000000 \times 10^{-8}}\right) = 1.32454766.$$

Algorithm 2:

$$fl\left(\frac{e^x - 1}{\log e^x}\right) \equiv fl\left(\frac{1.19209290 \times 10^{-7}}{1.19209282 \times 10^{-7}}\right) = 1.00000006.$$

Algorithm 2 in exact arithmetic would give

$$\frac{e^x - 1}{\log e^x} \equiv \frac{9.00000041 \times 10^{-8}}{9.00000001 \times 10^{-8}} = 1.00000005.$$

Misconception 3 *The final computed answer from an algorithm cannot be more accurate than any of the intermediate quantities, that is, errors cannot cancel.*

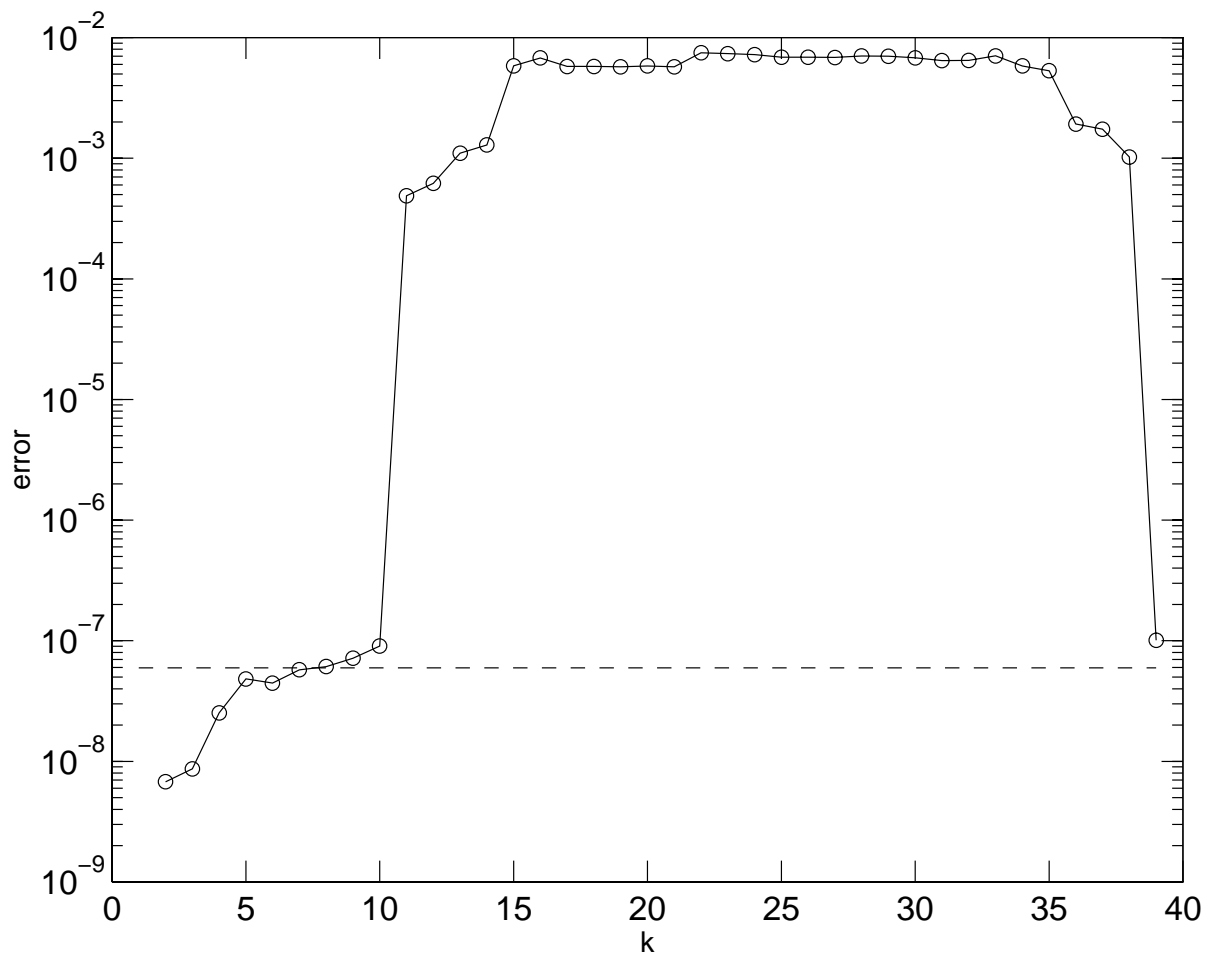
Givens QR Factorization

$$G_p G_{p-1} \dots G_1 A = R.$$

$A \in \mathbb{R}^{10 \times 6}$, single precision ($u \approx 6 \times 10^{-8}$).

$$\|A\|_2 = 1, \kappa_2(A) \approx 24.$$

Look at $\|A_k - \hat{A}_k\|_2 / \|A\|_2$:



Computing the Sample Variance

Sample variance of x_1, \dots, x_n defined as

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2,$$

where the sample mean

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i.$$

This is a *two-pass* formula.

Alternative *one-pass* formula:

$$s_n^2 = \frac{1}{n-1} \left(\sum_{i=1}^n x_i^2 - \frac{1}{n} \left(\sum_{i=1}^n x_i \right)^2 \right).$$

If $x = [10000, 10001, 10002]^T$ then, in single precision arithmetic ($u \approx 6 \times 10^{-8}$), computed sample variances are

two-pass formula: 1.0,

one-pass formula: 0.0.

Accurate One-Pass Formula

Instead of accumulating $\sum_i x_i$ and $\sum_i x_i^2$,
accumulate

$$M_k := \frac{1}{k} \sum_{i=1}^k x_i$$

and

$$Q_k := \sum_{i=1}^k (x_i - M_k)^2 = \sum_{i=1}^k x_i^2 - \frac{1}{k} \left(\sum_{i=1}^k x_i \right)^2,$$

which can be done via the updating formulae

$$M_1 = x_1, \quad M_k = M_{k-1} + \frac{x_k - M_{k-1}}{k},$$

$$Q_1 = 0, \quad Q_k = Q_{k-1} + \frac{(k-1)(x_k - M_{k-1})^2}{k},$$

after which $s_n^2 = Q_n / (n - 1)$.

Cray Peculiarity

Carter (1991, NASA Ames Lab.) used Cholesky factorization on a Cray Y-MP to solve a linear system of order 16146.

Bits	u	Computer	Displacement
128	1e-29	Cray 2	.447440341
64	1e-17	IBM 3090	.44744034 <u>4</u>
64	1e-16	IRIS	.44744033 <u>9</u>
64	1e-15	Cray 2	.4474403 <u>03</u>
64	1e-15	Cray Y-MP	.44743 <u>6106</u>

Why do the two Crays give different results?

- ▷ Cray Y-MP subtraction produces biased results:
 $fl(x - y)$ is too big if $x > y > 0$.
- ▷ Cray 2 subtraction more nearly unbiased.

Inner loop of the Cholesky algorithm:

$$a_{ii} = a_{ii} - a_{ij}^2.$$

Cure: iterative refinement in fixed precision.

Patriot Missile Software Problem

Official Report from United States:

“On February 25, 1991, a Patriot missile defense system operating at Dhahran, Saudi Arabia, during Operation Desert Storm failed to track and intercept an incoming Scud. This Scud subsequently hit an Army barracks, killing 28 Americans”

- Patriot missile’s computer: 1970s design, 24 bit arithmetic.
- Patriot tracks targets by measuring the time for radar pulses to reflect back.
- Time recorded by system clock in tenths of a second, but stored as an integer.
- For calculations, time converted to a 24 bit fl. pt. number.

Patriot: More Detail

Hours	Calculated Time (secs)	Inaccuracy (secs)	Approx. shift in range gate (m)
0	0	0	0
1	3599.9966	.0034	7
8	28799.9725	.0275	55
20	71999.9313	.0687	137
48	172799.8352	.1648	330
72	259199.7528	.2472	494
100	359999.6567	.3433	687

- Target outside range gate after 20 hours.
- On February 25, 1991, Alpha Battery, which was protecting Dhahran Air Base, had been in continuous operation for over 100 hours.
- On February 26, modified software to correct the problem arrived in Dhahran.
- Table consistent with relative error in 24-bit fixed point representation of 0.1.

Model of Fl. Pt. Arithmetic

STANDARD MODEL

$$\begin{aligned} fl(x \text{ op } y) &= (x \text{ op } y)(1 + \delta), \\ |\delta| &\leq u, \quad \text{op} = +, -, *, /. \end{aligned}$$

Normal to assume the model holds also for the square root.

Here, u is the unit roundoff. In IEEE double precision arithmetic $u = 2^{-53} \approx 1.1 \times 10^{-16}$.

This modification can also be used:

$$fl(x \text{ op } y) = \frac{x \text{ op } y}{1 + \delta}, \quad |\delta| \leq u.$$

Both models are valid for most current computers, but not for most Cray machines (Cray 1, 2, X-MP, Y-MP and C90).

Summation

$$S_n = \sum_{i=1}^n x_i$$

Recursive summation:

```

s = 0
for i = 1:n
    s = s + x_i
end

```

Increasing order: $|x_1| \leq |x_2| \leq \cdots \leq |x_n|,$

Decreasing order: $|x_1| \geq |x_2| \geq \cdots \geq |x_n|.$

Easy to show that $E_n = S_n - \widehat{S}_n$ satisfies

$$|E_n| \leq (|x_1| + |x_2|)\gamma_{n-1} + \sum_{i=3}^n |x_i|\gamma_{n-i+1},$$

where

$$\gamma_n := \frac{nu}{1 - nu}.$$

Two More Methods

Pairwise summation:

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

Insertion method: (assume $|x_1| \leq \dots \leq |x_n|$)

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

General Algorithm

Algorithm for $S_n = \sum_{i=1}^n x_i$.

Let $\mathcal{S} = \{x_1, \dots, x_n\}$.

repeat while \mathcal{S} contains more than one element

Remove two numbers x and y from \mathcal{S} ,
and add their sum $x + y$ to \mathcal{S} .

end

Assign the remaining element of \mathcal{S} to S_n .

Error Analysis

Express i th execution of the repeat loop as

$T_i = x_{i_1} + y_{i_1}$. The computed sums satisfy

$$\widehat{T}_i = \frac{x_{i_1} + y_{i_1}}{1 + \delta_i}, \quad |\delta_i| \leq u, \quad i = 1:n - 1.$$

Local error is

$$\delta_i \widehat{T}_i.$$

Error Bound

Total error is sum of local errors:

$$E_n := S_n - \widehat{S}_n = \sum_{i=1}^{n-1} \delta_i \widehat{T}_i.$$

Smallest possible error bound:

$$|E_n| \leq u \sum_{i=1}^{n-1} |\widehat{T}_i|.$$

Since $|\widehat{T}_i| \leq \sum_{j=1}^n |x_j| + O(u)$, have also the weaker bound

$$|E_n| \leq (n-1)u \sum_{i=1}^n |x_i| + O(u^2).$$

In designing or choosing a summation method to achieve high accuracy, the aim should be to minimize the absolute values of the intermediate sums T_i .

Application

Recursive Summation

- Here,

$$T_{i-1} = S_i := \sum_{j=1}^i x_j.$$

Ideally, choose ordering to minimize $\sum_{i=2}^n |\widehat{S}_i|$.

- Compromise (Psum): minimize, in turn, $|x_1|$, $|\widehat{S}_2|$, \dots , $|\widehat{S}_{n-1}|$. ($O(n \log n)$ comparisons.)
- Further compromise: increasing ordering.
- If $x_i \geq 0$, increasing ordering is optimal.

“ \pm ” method

- Compute sum of the positive numbers, S_+ , and the nonpositive numbers, S_- , separately, then form $S_n = S_- + S_+$.
- Maximizes $\max_i |T_i|$.

IEEE Arithmetic

IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985.

Design principles included

- Encourage experts to develop robust, efficient and portable numerical programs.
- Enable handling of arithmetic exceptions.
- Provide for development of transcendental functions and very high precision arithmetic.

The standard specifies

- floating point number formats,
- results of the basic floating point operations and comparisons,
- rounding modes,
- floating point exceptions and their handling.

NB: Says nothing about exponentiation or transcendental functions.

Floating Point Formats

Type	Size	Mantissa	Exponent	Range
single	32 bits	23+1 bits	8 bits	$10^{\pm 38}$
double	64 bits	52+1 bits	11 bits	$10^{\pm 308}$

single precision: $u = 2^{-24} \approx 5.96 \times 10^{-8}$,
double precision: $u = 2^{-53} \approx 1.11 \times 10^{-16}$.

All arithmetic operations to be performed *as if* first calculated to infinite precision, then rounded according to one of four modes.

Default is round to nearest, with rounding to even in case of a tie.

IEEE arithmetic implemented on, e.g.,

- ▷ Intel 80287, 80387, 80486 and Pentium,
- ▷ Motorola 680x0,
- ▷ DEC Alpha,
- ▷ Sun SPARCstation,
- ▷ Hewlett Packard Precision Architecture,
- ▷ IBM RS/6000.

Exceptional Results

IEEE arithmetic is a closed system: every arithmetic operation produces a result.

Default results are:

Exception type	Default result
Invalid operation	NaN (Not a Number)
Overflow	$\pm\infty$
Divide by zero	$\pm\infty$
Underflow	Subnormal numbers
Inexact	Correctly rounded result

A NaN is generated by operations such as $0/0$, $0 \times \infty$, ∞/∞ , $(+\infty) + (-\infty)$ and $\sqrt{-1}$.

The infinity symbol satisfies $\infty + \infty = \infty$, $(-1) \times \infty = -\infty$ and $(\text{finite})/\infty = 0$.

Advantages of IEEE—1

Problem: find biggest element of an array $a(1:n)$.

```
max = -inf
for j = 1:n
    if a_j > max
        max = a_j
    end
end
```

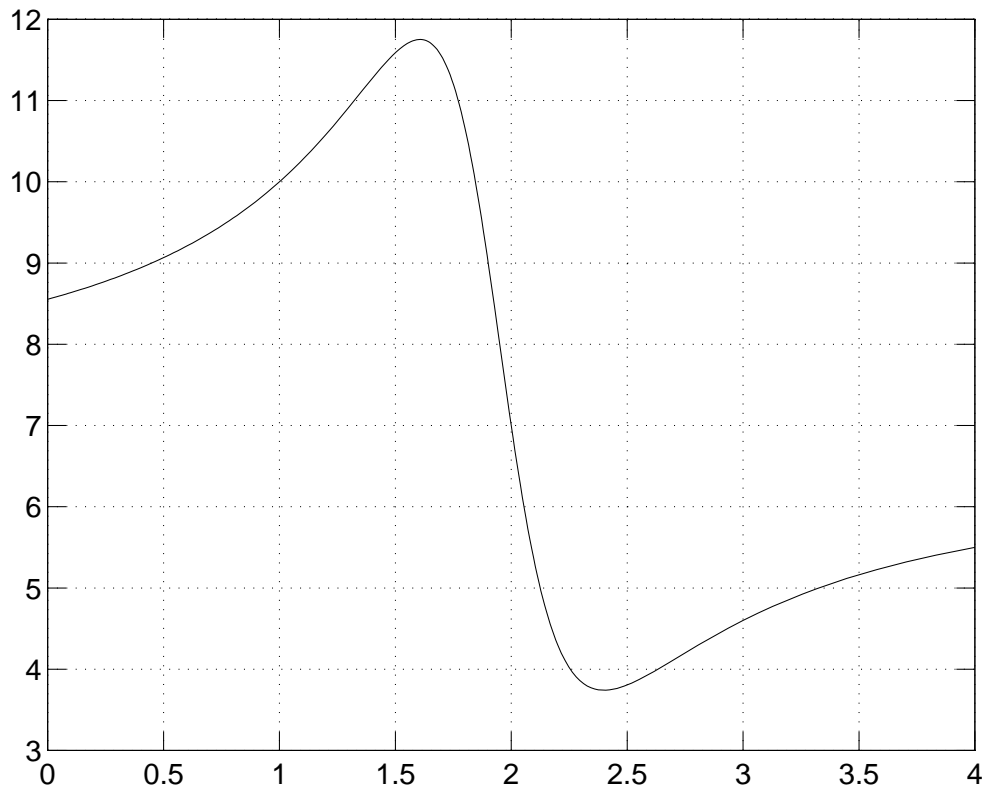
Unknown or missing elements of the array \mathbf{a} could be assigned NaNs: $\mathbf{a}(j) = \text{NaN}$.

A NaN compares as unordered with everything.

Advantages of IEEE—2

Rational function

$$r(x) = \frac{(((7x - 101)x + 540)x - 1204)x + 958}{(((x - 14)x + 72)x - 151)x + 112}$$

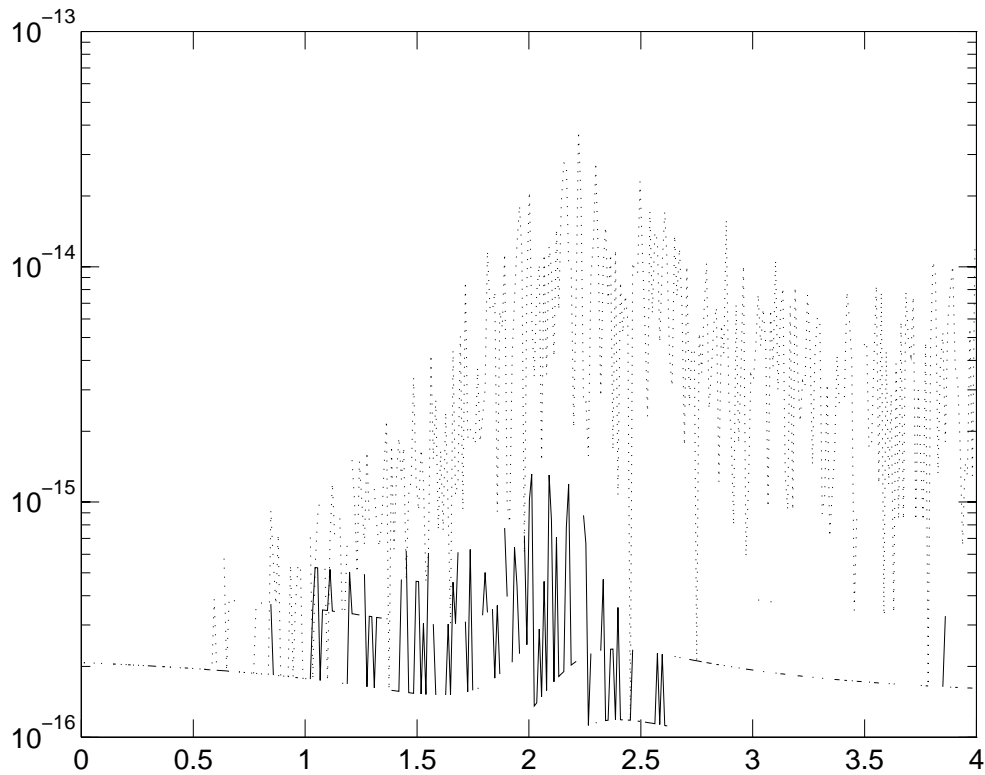


Cont. Frac. versus Rational

Continued fraction

$$r(x) = 7 - \frac{3}{x - 2 - \frac{1}{x - 7 + \frac{10}{x - 2 - \frac{2}{x - 3}}}}$$

Division by zero at $x = 1, 2, 3, 4$, but r evaluates correctly in IEEE arithmetic!



Summary

▷ Recommend the representation of fl. pt. numbers

$$y = \pm m \times \beta^{e-t}, \quad 0 \leq m \leq \beta^t - 1.$$

▷ Misconceptions

- *Short computations free from cancellation, underflow and overflow must be accurate.*
- *Increasing the precision increases the accuracy of an answer.*
- *Final answer can't be more accurate than the intermediate quantities.*

▷ Practical examples: computing variance, Cray discrepancy, Patriot missile.

▷ IEEE arithmetic: carefully defined, closed system, exception handling, widely used.

▷ A simple error analysis can be informative (summation).