Pathologies

non-uniqueness

$$\frac{du}{dt} = \sqrt{u} \quad , \quad u(0) = 0$$

$$\rightarrow \quad u(t) = \frac{t^2}{4}$$

But: Euler method

$$t_0 = 0 \qquad u_0 = 0$$

$$t_1 = h \qquad u_1 = 0 + h\sqrt{0} = 0$$

$$t_2 = 2h \qquad u_2 = 0 + h\sqrt{0} = 0$$

$$\rightarrow \qquad \underline{u_n = 0}$$

$\frac{du}{dt} = \sqrt{u}$ is not differentiable at 0, so it does not satisfy uniqueness criteria! (in fact $u(t) = 0$ is a valid solution) finite - time blowup

$$\frac{du}{dt} = u^2 \qquad , u(0) = 1$$

$$\rightarrow \quad u(t) = \frac{1}{1-t}$$

$\rightarrow$ numerical scheme will blow up as well.

xact condition: $\frac{du}{dt} = f(t,u)$ has a unique solution iff f is Lipschitz (can look up in any good ODE Textbook or Numerical Analysis textbook (Süli & Mayers))

Machine Arithmetic

How do computers represent numbers and what are the side effects?

<u>Ex:</u> $\pi = 3.1415926535\ldots$

∞ ly many digits — have to approximate somehow!

many different methods. In practice:

- fixed point numbers
- floating point numbers

are most often used.

Finite collections of (binary) digits to represent numbers
(Here we will do everything with decimal digits, for simplicity)

Fixed point numbers

allocate N digits to represent numbers, and *fix* the decimal point (and maybe a sign)

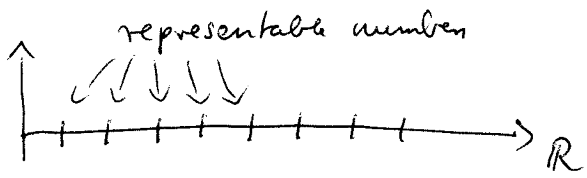
$$\pm \sqcup\sqcup\sqcup.\sqcup\sqcup\sqcup\sqcup$$

Ex: $12.34 \longrightarrow \boxed{+012.3400}$

Which numbers can we represent like this?

$$\left.\begin{array}{l} -999.9999 \\ -999.9998 \\ \vdots \\ -000.0001 \\ +000.0000 \\ +000.0001 \\ \vdots \\ +999.9999 \end{array}\right\} \text{representable set } \mathcal{F}^{rep}$$

Fixed point #s are <u>constant density</u>: here $|r_n - r_{n+1}| = 0.001$ for consecutive exactly representable numbers.



Rounding error

Let $r \in \mathbb{R}$ and $r \in [R_{min}, R_{max}]$ be arbitrary and real.

→ must be rounded to the nearest representable number to do computations.

$$fi(r) = \text{nearest fixed-pt representable number to } r.$$

then $fi(r) = r + \varepsilon$

with $|\varepsilon| < EPS\,ABS$: maximum error when rounding.

for our set of #s: $EPS\,ABS = 0.00005$

→ absolute rounding error is bounded for fixed-pt numbers.

alculation examples

Error free: $12.34 + 742.55$

$$\rightsquigarrow \begin{array}{r} 0012.3400 \\ +\ 0742.5500 \\ \hline 0754.8900 \end{array}$$

f all inputs and outputs are in the representable set,
e calculation is error free!

Ex: 24/7

$$\begin{array}{r} 024.0000 \\ /\ 007.0000 \\ \hline 003.4286 \end{array}$$

not an exact result : $24/7 \simeq 3.\overline{428571}$.

had to round to a representable number!

Ex: $412.24 + 742.55 = 1154.89$

$$\begin{array}{r} 412.2400 \\ + \ 742.5500 \\ \hline 154.7900 \end{array}$$

not enough space for the last digit!

-> "overflow" (some computers detect this and give you an error message)

Floating point numbers

(mostly) fix overflow issues, but at a cost!

ea: scientific notation: $12345.38 = 1.234538 \cdot 10^4$

$\rightarrow \quad \pm \underbrace{\sqcup . \sqcup \sqcup \sqcup \sqcup \sqcup}_{\text{mantissa}} \times 10^{\overbrace{\pm \sqcup \sqcup}^{\text{exponent}}}$

Ex: $12.34 = 1.2340 \cdot 10^{+01}$

$754.89 = 7.5489 \cdot 10^{+02}$

$\rightarrow$ huge dynamic range. Here: from $10^{-103} \sim 10^{99}$,

more than 200 orders of magnitude!

ost: Representable set is non-uniform

Same number of repres. numbers between $[0, 10]$ as there are between $[100, 1000]$

which numbers are exactly representable?

- integers between $(-I_{max}, I_{max})$, where I_{max} depends on mantissa. here: $I_{max} = 99{,}999$.
- integers divided by 10 (in decimal arithmetic)
- integers divided by 2 (in binary arithmetic)
- zero

rounding error bounds are relative:

$fl(r)$ = closest floating point number to $r \in \mathbb{R}$.

then:

$$fl(r) = r(1+\varepsilon),$$

$$|\varepsilon| \leq \underbrace{EPSREL}_{\text{max. relative error}}$$

here: $EPSREL \approx 10^{-5}$

for "double precision": $EPSREL \simeq 10^{-15}$.

Catastrophic loss of precision

> Never compute a small number as the difference between two large numbers

x:

US population (thousands)

Feb	311, 189
Mar	311, 356

$\rightarrow$ difference $311{,}356 - 311{,}189 = 167$ (reasonable)

in our floating point scheme:

$$fl(311{,}189) = 3.1119 \cdot 10^5$$
$$fl(311{,}356) = 3.1136 \cdot 10^5$$

$$\begin{array}{rl} \rightsquigarrow & 3.1136 \cdot 10^5 \\ - & 3.1119 \cdot 10^5 \\ \hline & 0.0017 \cdot 10^5 = 1.7000 \cdot 10^2 \end{array}$$

$\rightsquigarrow$ relative error of $\approx 2 \cdot 10^{-2}$ but EPSREL $\approx 10^{-5}$???

→ allmost all digits used up to store the "large part" of the numbers, which cancel out.

<u>generally</u>: if for N digits of precision, when the first M digits of two numbers agree, expect only <u>$N - M$</u> digits of precision in the difference.

$M \approx N \longrightarrow$ catastrophic precision loss x: Finite differences

$$f'_{FD}(x) = \frac{f(x+h) - f(x)}{h}$$

$f(x) = x$, want $f'(1) = 1$

x) $h = \frac{2}{3}$, $fl(h) = 0.66667$

$f(x+h) = 1.6667$

$f(x) = 1.0000$

$$\rightarrow \quad \frac{f(x+h) - f(x)}{h} = \frac{0.666\underbrace{70}_{}}{0.666\underbrace{67}_{\text{differ in 4th digit.}}}$$

$$\rightarrow \quad f'_{FD}(1) \approx 1 + O(10^{-4})$$

) $h = \frac{2}{30}$, $fl(h) = 0.0\underbrace{66667}_{\text{still 5 sign. digits!}}$

How to avoid floating point precision loss?

→ Rearrange things creatively to avoid the nasty differences!

Ex: $f(x, \Delta) = \sqrt{x+\Delta} - \sqrt{x}$, $\Delta \ll x$.

for instance: $x = 900$, $\Delta = 4 \cdot 10^{-3}$

$$\to \underbrace{30.0000}_{\text{6 wasted digits}}6667 - 30.0000\,0000$$

6 wasted ~~precision~~ digits

in our scheme: $fl(f(900, 4 \cdot 10^{-3})) = 0$

here we can use:

$$(\sqrt{x+\Delta} - \sqrt{x})(\sqrt{x+\Delta} + \sqrt{x}) = x + \Delta - x$$
$$= \Delta$$

$$\Rightarrow \sqrt{x+\Delta} - \sqrt{x} = \frac{\Delta}{\sqrt{x+\Delta} + \sqrt{x}}$$

$$\rightsquigarrow \frac{4 \cdot 10^{-3}}{30.00006667 - 30.00000000} \simeq \frac{4 \cdot 10^{-3}}{6.0000 \cdot 10^{2}} \simeq 6.6667 \cdot 10^{-5}$$

here we got all the precision we want!