\[ K \approx \left| \frac{x \cdot f'(x)}{f(x)} \right| \]

\[ f(x) = \sin x \Rightarrow \frac{x \cdot \cos(x)}{\sin(x)} \]

\[ \text{biggest near 0} \Rightarrow \frac{x \cdot 1}{x} \approx 1 \]
Stability and Accuracy

- When is a method stable?
  
  If it sensitivity to variation in input is no (or not much) greater than that of the underlying problem.

- When is a method accurate?
  
  Closeness of method output to the actual answer for completely accurate input

- How can I produce inaccurate results?
1.2 Floating Point
Wanted: Real Numbers... in a computer

- Computers can represent integers, using bits:

$$23 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (10111)_2$$

How would we represent fractions?

$$23.625 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

$$+ \frac{1}{2} \cdot 2^{-1} + \frac{0}{2} \cdot 2^{-2} + \frac{1}{2} \cdot 2^{-3}$$

Fixed-point arithmetic
Fixed-Point Numbers

- Suppose we use units of 64 bits, with 32 bits for exponents $\geq 0$ and 32 bits for exponents $< 0$. What numbers can we represent?

- How many ‘digits’ of relative accuracy (think relative rounding error) are available for the smallest vs. the largest number?

Largest: 64 bits: $2^{64} \approx 10^{19}$ $\sim$ 19 digits

Smallest: 0 bits $\sim$ 0 digits
Floating Point numbers

- Convert $13 = (1101)_2$ into floating point representation.

$$13 = (1.101)_2 \cdot 2^3$$

- What pieces do you need to store an FP number?

$$\begin{align*}
(1, 101)_2 & \rightarrow \text{"significand" stored: } -1 \cdot 2^3 + 1026 \\
3 & \rightarrow \text{"exponent"}
\end{align*}$$

$\pm$ \rightarrow sign
\[ 2.25 = \frac{1100.01}{1.10001 \cdot 2^3} = 2^{-3} \]
In-class activity: Floating Point
Unrepresentable numbers?

○ Can you think of a somewhat central number that we cannot represent as

\[ x = (1.\_\_\_\_\_\_\_\_\_)_2 \cdot 2^{-p}? \]

\[ ^n \text{special exponent} : -1023 \]
Demo: Picking apart a floating point number