Floating-point operations I

- The science of floating-point arithmetics
- IEEE standard
- Reference

  What every computer scientist should know about floating-point arithmetic, ACM computing survey, 1991

Why learn more about floating-point operations I

Example:

- A one-variable problem

$$\min_{x} f(x)$$
$$x \geq 0$$

- In your program, should you set an upper bound of x?
- x in your program may be wrongly increased to ∞

Why learn more about floating-point operations II

- What is the largest representable number in the computer?
- Is there anything called infinity?

Example:

- A ten-variable problem

$$\min f(x)$$
$$0 \leq x_i, i = 1, \ldots, 10$$

Why learn more about floating-point operations III

- After the problem is solved, want to know how many are zeros?
- Should you use

```
for (i=0; i < 10; i++)
     if (x[i] == 0) count++ ;
```

- People said: don't do floating-point comparisons

```
epsilon = 1.0e-12 ;
for (i=0; i < 10; i++)
     if (x[i] <= epsilon) count++ ;
```

Why learn more about floating-point operations IV

How do you choose ϵ?

- Is this true?

Floating-point Formats I

- We know float (single): 4 bytes, double: 8 bytes
  Why?
- A floating-point system
  base β, precision p, significand (mantissa) $d.d\ldots d$
- Example

$$
\begin{aligned}
0.1 &= 1.00 \times 10^{-1} \qquad (\beta = 10, p = 3) \\
&\approx 1.1001 \times 2^{-4} \qquad (\beta = 2, p = 5)
\end{aligned}
$$

  exponent: -1 and -4
- Largest exponent $e_{\max}$, smallest $e_{\min}$

Floating-point Formats II

- β^p possible significands, $e_{\text{max}} - e_{\text{min}} + 1$ possible exponents

  $$\lceil \log_2(e_{\text{max}} - e_{\text{min}} + 1) \rceil + \lceil \log_2(\beta^p) \rceil + 1$$

  bits for storing a number

  1 bit for $\pm$
- But the practical setting is more complicated

  See the discussion of IEEE standard later
- Normalized: 1.00×10^{-1} (yes), 0.01×10^1 (no)
- Now most used normalized representation

Floating-point Formats III

but an issue is we cannot represent zero

- A natural way for 0: $1.0 \times \beta^{e_{\min}-1}$

  This preserves the ordering
- Will use $p = 3, \beta = 10$ for most later explanation

Relative Errors and Ulps I

- When $\beta = 10, p = 3$, 3.14159 represented as 3.14×10^0

  $\Rightarrow$ error $= 0.00159 = 0.159 \times 10^{-2}$, i.e. 0.159 units in the last place

  10^{-2}: unit of the last place
- `ulps`: unit in the last place
- relative error $0.00159/3.14159 \approx 0.0005$
- For a number $d.d \ldots d \times \beta^e$, the largest error is

$$0.\underbrace{0\ldots0}_{p-1}\beta' \times \beta^e, \beta' = \beta/2$$

Relative Errors and Ulps II

- Error $= \frac{\beta}{2} \times \beta^{-p} \times \beta^{e}$

$$1 \times \beta^{e} \leq \text{ original value } < \beta \times \beta^{e}$$

relative error between

$$\frac{\frac{\beta}{2} \times \beta^{-p} \times \beta^{e}}{\beta^{e}} \quad \text{and} \quad \frac{\frac{\beta}{2} \times \beta^{-p} \times \beta^{e}}{\beta^{e+1}}$$

so

$$\text{relative error } \leq \frac{\beta}{2}\beta^{-p} \tag{1}$$

- $\frac{\beta}{2}\beta^{-p} = \beta^{-p+1}/2$ is called machine epsilon

Relative Errors and Ulps III

That is, the bound in (1)

- When a number is rounded to the closest, relative error bounded by ϵ

ulps and ϵ I

- $p = 3$, $\beta = 10$
- Example: $x = 12.35 \Rightarrow \tilde{x} = 1.24 \times 10^1$
  error $= 0.05 = 0.005 \times 10^1$
- ulps $= 0.01 \times 10^1$, $\epsilon = \frac{1}{2}10^{-2} = 0.005$
- error 0.5 ulps
  relative error $0.05/12.35 \approx 0.004 = 0.8\epsilon$
- $8x = 98.8, 8\tilde{x} = 9.92 \times 10^1$
  error $= 4.0$ ulps
  relative error $= 0.4/98.8 = 0.8\epsilon.$
- ulps and ϵ may be used interchangeably