PhD Startup Seminar

HIPERFIT

<oleks@oleks.info>

May 24, 2016

Problem Statement (1/2)

Can we guarantee that our methods are
not only *fast* and (mathematically) *correct*,
but also (numerically) *accurate*?

If not,

- When are they inaccurate?
- Can we guard, or warn our users?
- How can we do better?

Problem Statement (2/2)

Approximating real arithmetic (universally) is hard.

Trade-offs between,

- Performance
- Accuracy
- Range
- Ease of use
- Memory use
- Power use

Among others…

Floating-Point

A floating-point value is a triplet (s, f, e), representing the mathematical value[1]

$$(-1)^s \cdot f \cdot \beta^e.$$

Fitting this triplet onto the conventional digital quanta of 8, 16, 32, 64, etc. bits presents a number of challenges:

- Should we prioritze f (significand) or e (exponent)?
- Can we have an arithmetic closed under basic operations?
- What about exceptional behaviour (e.g., divide by 0)?

[1] Typically, $\beta = 2$.

IEEE 754 Floating-Point

For the functional programmer:

```
data FP
  = Normal Sign Significand Exponent
  | Infinity Sign
  | NaN Payload -- Payload carries the reason for the NaN.
  | Zero Sign
  | Subnormal Sign Significand -- Allows gradual underflow.
```

- Designed for portability (but doesn't always succeed).
- Closure is achieved through rounding (next slide).

IEEE 754 Floating-Point — Rounding

Perform every operation as if with infinite precision, rounded to fit the desired resulting precision.

- Round towards positive infinity.
- Round towards negative infinity.
- Round towards zero.
- Round to nearest, ties to even (most common, default).
- Round to nearest, ties away from zero (optional for base 2).
- Custom rounding modes are allowed…

Rounding?

There is no best way to round.

It is the accumulation of error that causes trouble.

Accuracy is a Measure of Error

Let $\hat{x}$ be an approximation of x.

An approximation has an error.

Absolute Error:
$|x - \hat{x}|$

Relative Error:
$|(x - \hat{x})/x|$

Accuracy:
The absolute or relative error of $\hat{x}$.

Precision

The accuracy with which individual operations are performed.

- Overall accuracy may change with each operation.
- Precision remains the same throughout a computation.
  - Although it may be increased or decreased *explicitly*.

Half, Single, Double, Quadruple Precision! (1/3)

	Total	Exponent	Significand
Half	16	5	10
Single	32	8	23
Double	64	11	52
Quadruple	128	15	112

You've probably heard of single and double precision.

What is the difference in terms of the trade-offs above?
Higher means better, right? Right?

Half, Single, Double, Quadruple Precision! (2/3)

- Performance
- Accuracy
- Range
- Ease of use
- Memory use
- Power use

Half, Single, Double, Quadruple Precision! (2/3)

- Performance
- Accuracy
- Range
- Ease of use
- Memory use
- Power use

- Single and double are typically supported hardware.
  - Performance is good.
  - Higher precision requires more resources.

Half, Single, Double, Quadruple Precision! (2/3)

- Performance
- Accuracy
- Range
- Ease of use
- Memory use
- Power use

- Single and double are typically supported hardware.
  - Performance is good.
  - Higher precision requires more resources.
- Programmer can typically choose the precision.
  - Interface not always clear-cut, but usually doable.

Half, Single, Double, Quadruple Precision! (2/3)

- Performance
- Accuracy
- Range
- Ease of use
- Memory use
- Power use

- Single and double are typically supported hardware.
  - Performance is good.
  - Higher precision requires more resources.
- Programmer can typically choose the precision.
  - Interface not always clear-cut, but usually doable.
- Higher precision means higher range of exponent.

Half, Single, Double, Quadruple Precision! (3/3)

- Higher precision does not guarantee better accuracy[1].
- Identical output in higher precision is unreliable[1].
- Instability can happen without cancellation[2].

[1] A. Cuyt, B. Verdonk, S. Becuwe, and P. Kuterna. *A Remarkable Example of Catastrophic Cancellation Unraveled*. Computing 66(3): 309–320, 2001.

[2] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second Edition. SIAM, 2002.

Example: The Chaotic Bank Society[1]

$$
\begin{aligned}
a_0 &= e - 1 \\
a_1 &= 1 \cdot a_0 - 1 \\
a_2 &= 2 \cdot a_1 - 1 \\
a_3 &= 3 \cdot a_2 - 1 \\
&\dots \\
a_n &= n \cdot a_{n-1} - 1
\end{aligned}
$$

$$
\lim_{n \to \infty} a_n = \begin{cases} -\infty & \text{if} \quad a_0 < e - 1 \\ 0 & \text{if} \quad a_0 = e - 1 \\ +\infty & \text{if} \quad a_0 > e - 1 \end{cases}
$$

[1] J.M. Muller, N. Brisebarre, F. de Dinechin, C.P. Jeannerod, V. Lefévre, G. Melquiond, N. Revol, D. Stehle, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser, 2009.

Example: Creeping Crud [Gustafson (2015)]

```
volatile REAL sum = START;
for (i = 0; i < N; ++i) {
  sum += ADDEND;
}
```

- A generalization of the example in [Gustafson (2015)].

Truth about Sum

- There is more than one way.
- Magnitude matters.
- Sign (order) matters.

Perform sum as if with infinite precision,
rounded to fit the desired resulting precision.

[1] Nicholas J. Higham. *The Accuracy of Floating Point Summation*. SIAM J. Sci. Comput., 14(4): 783–799, 1993.

[2] E. Kadric, P. Gurniak and A. Dehon. *Accurate Parallel Floating-Point Accumulation*. Computer Arithmetic (ARITH), 2013, 21st IEEE Symposium on, pp. 153–162.

Numbers: Some (Universal) Approaches

- Finite-Precision Arithmetic.
  - Fixed-point
  - Floating-point
  - Rational arithmetic
- Interval Arithmetic.
  - Avoid rounding, use an interval.
  - Interval boxing.
- "Dependent Arithmetic".
  - Model the dependency of variables on one another.
  - Affine Arithmetic: linear dependencies.
  - Taylor Series Methods: polynomial dependencies.

[1] Nedialko S. Nedialkov, Vladik Kreinovich, and Scott A. Starks. *Interval Arithmetic, Affine Arithmetic, Taylor Series Methods: Why, What Next?* Numerical Algorithms, 37(1): 325–336, 2004.

Domain-Specific Number Formats

- FPGAs
- Reconfigurable Computing
- ASICs

[1] M. Courbariaux, Y. Bengio, and J.P. David. *Training deep neural networks with low precision multiplications*. 3rd International Conference on Learning Representations (ICLR 2015). arXiv:1412.7024 [cs.LG].

[2] F. Fang, T. Chen and R. A. Rutenbar. *Floating-point bit-width optimization for low-power signal processing applications*. Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE Int. Conf., pp. III-3208–III-3211.

[3] A. A. Gaffar, O. Mencer and W. Luk. *Unifying bit-width optimisation for fixed-point and floating-point designs*. Field-Programmable Custom Computing Machines (FCCM), 2004, 12th Ann. IEEE Symp., pp. 79–88.

- Use automatic differentiation to find necessary and sufficient bit-widths.

Unums — Universal Numbers

The Good:

- Eliminate subnormal numbers.
- Don't round — there's no good way to do it anyway.
- NaNs have no payload but are signalling/non-signalling.

The Bad:

- No formal specification.
- Not even a standard.

The Ugly?

[Gustafson (2015)]
John L. Gustafson. *The End of Error: Unum Computing*. CRC Press, 2015.

PhD Startup Hypotheses

- Function composition can be accuracy-preserving.
- There exist better alternatives to IEEE-754.
- Lazy arithmetic is worthwhile.
- Practical error-approximation is feasible.

More Information

https://github.com/oleks/phd/tree/master/exp/