A Simplified Float Filter

Historical Remarks

I gave a talk about the exact computation paradigm in Dagstuhl at a meeting on exact methods in numerical analysis.

Arnold Neumaier (Uni Wien) suggested a simpler way to derive our floating point filter.

He first argued that our filter is an application of interval arithmetic. Intervals may be represented either

- by their endpoints (endpoints representation) or
- by their midpoint and their radius (midpoint representation)

The endpoints representation usually gives sharper bounds. The midpoint representation induces less computational overhead. We use the midpoint representation in our floating point filter. Here is an example to show the difference. Consider $(1 \pm 0.1)^2 = 1 \pm 0.2 + 0.01 = 1.01 \pm 0.2$. In the endpoint representation we get an interval of length 0.2. In the center point representation we take 1.0 as the center point and get an interval of length 0.21.

We derive a bound of the form

$$|E - \tilde{E}| \leq B$$

where

$$B = \epsilon \cdot ind_E \cdot mes_E,$$

i.e., we use $\tilde{E}$ as the center point of our interval and we use B as the radius of our interval. In our main theorem (validity of the filter) we prove the equation above and (as an auxiliary claim) $|\tilde{E}| \leq mes_E$.

The trick (which he admitted to be new) is that our expression for the radius consists of two parts: the part ind_E which is easily precomputed and a part mes_E which is easily computed on-line.

He suggested to define ind_E as $((1+\epsilon)^{d_E} - 1)$ and to use the simple recursive definition given in Table 1. We write δ instead of $1 + \epsilon$.

The New Formulation

We derive a bound of the form

$$|E - \tilde{E}| \leq B$$

where

$$B = ((1+\epsilon)^{d_E} - 1) \cdot mes_E,$$

and d_E and mes_E are defined in Table 1. The intuitive interpretation is as follows: mes_E is an upper bound for the E and $\tilde{E}$ and d_E is a measure for the levels of rounding in E.

E	$\tilde{E}$	mes_E	d_E
a, integer	$fl(a)$	$\lvert fl(a)\rvert$	1
a, float integer	$fl(a)$	$\lvert fl(a)\rvert$	0
$A+B$	$\tilde{A} \oplus \tilde{B}$	$mes_A \oplus mes_B$	$1+\max(d_A, d_B)$
$A-B$	$\tilde{A} \ominus \tilde{B}$	$mes_A \oplus mes_B$	$1+\max(d_A, d_B)$
$A \cdot B$	$\tilde{A} \odot \tilde{B}$	$mes_A \odot mes_B$	$1+d_A+d_B$

Table 1: The recursive definition of mes_E and ind_E. The first column contains the case distinction according to the syntactic structure of E, the second column contains the rule for computing $\tilde{E}$ and the third and fourth columns contain the rules for computing mes_E and ind_E; $\oplus$ and $\odot$ denote the floating point implementations of addition and multiplication. We use the abbreviation $fl(a) = a.to\_double(\,)$.
In my original write-up, there was an additional rule dealing with square roots. The line was stated without proof. When I prepared for the lecture, I was unable to fill in the proof.

Each operator introduces an additional level. In an addition, the arguments contribute the maximum of their levels, and in a multiplication, the arguments contribute their sum. Each level of rounding increases the range of uncertainty by a multiplicative factor of $1+\epsilon$. Subtraction of -1 reflects the fact that we are interested in the error.

Theorem 1 *If mes_E and d_E are computed according to Table 1 then $|\tilde{E}| \le mes_E$ and $|\tilde{E} - E| \le ((1+\epsilon)^{d_E} - 1) \cdot mes_E$*

Proof: We use induction on the structure of the expression E. The claim $|\tilde{E}| \le mes_E$ follows immediately from the monotonicity of floating point arithmetic. For the second claim we have to work slightly harder.

Assume first that E is an integer a. Then

$$|a - fl(a)| \le \epsilon \cdot |fl(a)|$$

by item (a) and the claim is certainly true. If a is a floating point integer then $fl(a) = a$ and hence the degree can be set to zero for floating point integers.

For the induction step we make a case distinction according to the operation combining A and B.

Assume $E = A + B$. Then

$$|\tilde{E} - E| = |\tilde{A} \oplus \tilde{B} - (A+B)| \le |\tilde{A} \oplus \tilde{B} - (\tilde{A} + \tilde{B})| + |\tilde{A} - A| + |\tilde{B} - B|.$$

Item (d) with $f_1 = \tilde{A}$ and $f_2 = \tilde{B}$ implies that the first term is bounded by $\epsilon|\tilde{A} \oplus \tilde{B}|$ and monotonicity of floating point arithmetic implies that

$$|\tilde{A} \oplus \tilde{B}| \le mes_A \oplus mes_B = mes_E.$$

For the other two terms we use the induction hypothesis to conclude

$$\begin{aligned}
|\tilde{A} - A| + |\tilde{B} - B| &\leq ((1+\epsilon)^{d_A} - 1) \cdot mes_A + ((1+\epsilon)^{d_B} - 1) \cdot mes_B \\
&\leq ((1+\epsilon)^{\max(d_A,d_B)} - 1) \cdot (mes_A + mes_B) \\
&\leq ((1+\epsilon)^{\max(d_A,d_B)} - 1) \cdot (1+\epsilon) \cdot mes_E.
\end{aligned}$$

Putting the two bounds together we obtain:

$$\begin{aligned}
|\tilde{E} - E| &\leq [\epsilon + ((1+\epsilon)^{\max(d_A,d_B)} - 1) \cdot (1+\epsilon)] \cdot mes_E \\
&= [(1+\epsilon)^{1+\max(d_A,d_B)} - 1] \cdot mes_E.
\end{aligned}$$

We turn to multiplications, $E = A \cdot B$. We have

$$|\tilde{E} - E| = |\tilde{A} \odot \tilde{B} - A \cdot B| \leq |\tilde{A} \odot \tilde{B} - \tilde{A} \cdot \tilde{B}| + |\tilde{A} \cdot \tilde{B} - A \cdot \tilde{B}| + |A \cdot \tilde{B} - A \cdot B|.$$

Item (d) with $f_1 = \tilde{A}$ and $f_2 = \tilde{B}$ implies that the first term is bounded by $\epsilon|\tilde{A} \odot \tilde{B}|$ and monotonicity of floating point arithmetic implies that

$$|\tilde{A} \odot \tilde{B}| \leq mes_A \odot mes_B = mes_E.$$

For the second term we use the induction hypothesis to conclude

$$\begin{aligned}
|\tilde{A} \cdot \tilde{B} - A \cdot \tilde{B}| &= |\tilde{A} - A| \cdot |\tilde{B}| \\
&\leq ((1+\epsilon)^{d_A} - 1) \cdot mes_A \cdot mes_B \\
&\leq ((1+\epsilon)^{d_A} - 1) \cdot (1+\epsilon) \cdot (mes_A \odot mes_B) \\
&= ((1+\epsilon)^{d_A} - 1) \cdot (1+\epsilon) \cdot mes_E,
\end{aligned}$$

and for the third term we conclude similarly

$$\begin{aligned}
|A \cdot \tilde{B} - A \cdot B| &= |A| \cdot |\tilde{B} - B| \\
&\leq (|\tilde{A}| + |\tilde{A} - A|) \cdot |\tilde{B} - B| \\
&\leq (1+\epsilon)^{d_A} \cdot mes_A \cdot ((1+\epsilon)^{d_B} - 1) \cdot mes_B \\
&\leq (1+\epsilon)^{1+d_A} \cdot ((1+\epsilon)^{d_B} - 1) \cdot mes_A \odot mes_B \\
&= (1+\epsilon)^{1+d_A} \cdot ((1+\epsilon)^{d_B} - 1) \cdot mes_E
\end{aligned}$$

Putting the three bounds together we obtain

$$\begin{aligned}
((1+\epsilon)^{d_E} - 1) &\leq \epsilon + (1+\epsilon) \cdot ((1+\epsilon)^{d_A} - 1) + (1+\epsilon)^{1+d_A} \cdot ((1+\epsilon)^{d_B} - 1) \\
&= \epsilon + \delta^{d_A+1} - 1 - \epsilon + \delta^{1+d_A+d_B} - \delta^{1+d_A} \\
&= \delta^{1+d_A+d_B} - 1
\end{aligned}$$

and the induction step is completed.

∎

The bound in our new floating point filter involves $((1+\epsilon)^{d_E} - 1)$. This number is somewhat hard to compute. We therefore upper bound it using the fact that $(1+\epsilon)^{d_E} \approx 1 + d_E\epsilon$ provided d_E is not too large.

Lemma 1 *If* $d \leq \sqrt{1/\epsilon} - 1$ *then* $((1+\epsilon)^d - 1) \leq (d+1)\epsilon$

Proof: We have

$$(1+\epsilon)^d - 1 = \sum_{1 \leq i \leq d} \binom{d}{i} \epsilon^i \leq \sum_{i \geq 1} (d \cdot \epsilon)^i = d\epsilon/(1-d\epsilon).$$

Next observe that $d\epsilon/(1-d\epsilon) \leq (1+d)\epsilon$ iff $d/(1-d\epsilon) \leq (1+d)$ iff $d \leq d+1-d^2\epsilon - d\epsilon$ iff $d(d+1) \leq 1/\epsilon$. This is certainly the case when $(d+1)^2 \leq \epsilon$ or $d \leq \sqrt{1/\epsilon} - 1$. ∎

Theorem 2 *If* $d_E \leq \sqrt{1/\epsilon} - 1$ *then* $|E - \tilde{E}| \leq (d_E + 1) \cdot \epsilon \cdot mes_E$.

Proof: Follows immediately from Theorem 1 and the preceding Lemma. ∎

Observe that the condition $d_E \leq \sqrt{1/\epsilon} - 1$ is hardly constraining. For $\epsilon = 2^{-53}$ it amounts to $d_E < 2^{26.5}$.

Let us apply our filter to the orientation predicate for points in homogeneous coordinates.

```
(axd*bwd - bxd*awd) * (ayd*cwd - cyd*awd) -
                  (ayd*bwd - byd*awd) * (axd*cwd - cxd*awd);
```

Let us compute d_E. We have:
The degree of an integer a is $s_1 = 1$;
The degree of an expression of the form $a \cdot a$ is $s_2 = 1 + 1 + 1 = 3$.
The degree of an expression of the form $a \cdot a + a \cdot a$ is $s_3 = 1 + s_2 = 4$.
The degree of an expression of the form $(a \cdot a + a \cdot a) \cdot (a \cdot a + a \cdot a)$ is $s_4 = 1 + s_3 + s_3 = 9$.
The degree of the orientation predicate is $s_5 = 1 + s_4 = 10$.
We can therefore take 11 in our bound by Theorem 2.

In his master's thesis Stefan Funke discussed floating point filters for a larger variety of operations, in particular, divisions and roots.

Static Filters: In the semi-dynamic filter, ind_E is precomputed and mes_E is computed on-line. Can we also precompute mes_E?

On page 620 of the LEDAbook, we argue that one can upper bound mes_E in certain situations. If an a-priori bound on the size of all inputs is known, say $mes_a \leq 2^L$ for all inputs a to a computation, then the maximal possible value of mes_E can be precomputed. For example, $mes_E = 2^{4L+3}$ for the orientation predicate. We concluded that static filters cannot be used for on-line algorithms. Also it is a nuisance to precompute L at the beginning of an algorithm. It requires to iterate over all inputs and hence a change of the algorithm.

For this reason, Stefan and I decided against static filters and implemented semi-dynamic filters.

Stefan Näher, Mathias Bäsken, and Oliver Zlotowski (personal communication, November 10, 2001) observed that our arguments against static filters are not convincing. They suggest to do the following. We keep a global variable L. Whenever, a new input arrives, we update L. If the value of L changes, we recompute mes_E and B for all geometric predicates used by the algorithm (or for all geometric predicates in the LEDA kernel). The hope is that L and hence B change only rarely. In this way, the filter becomes very fast. We simply compute $\tilde{E}$ and compare it against B. This has hardly any overhead, since B will be in cache (as it is accessed very frequently). Mathias and Oliver have implemented the new scheme. It will become part of the next release of LEDA. There will be two filter stages in the next release.

1. stage 1: static filter as described in the preceding paragraph
2. stage 2: old semi-dynamic filter
3. stage 3: exact integer computation