浮動小数点演算における桁落ち追跡のアルゴリズム*

7L－1

鈴木 弘　(都立航空高専 電子工学科)
大岩 元　(慶應大学 環境情報学部)

1　はじめに

浮動小数点演算には、丸め、情報落ち、桁落ちなどの誤差を含み、従って、数値計算の解析結果にも誤差を含むことになる。場合によっては、演算を繰り返すうちに誤差が蓄積し、実際とは全く違った結果が出ていることもある。

このように浮動小数点演算には必ず誤差が含まれるが、一般には、どれだけ誤差が含まれているかはユーザーは知ることはできない。そこで誤差そのものではないが、誤差が生じるもっとも大きな要因である桁落ちに着目し、計算過程で桁落ちを追跡し、最終的に桁落ちの影響を受けていない桁数をユーザーに知らせるシステムを作成した。そのための桁落ち追跡アルゴリズムについて述べる。

具体的には、プログラミング言語C++でクラスとオペレータオーバロードの機能を使い、実数(浮動小数点)変数から指数部を取り出し、演算の前後での指数部の変化を見ることにより桁落ちを判断するもので、これを演算が行われている間、常に監視し続ける。

2　桁落ち追跡システム

桁落ちを追跡することによって得られる有効桁を以下のように定義する。

「仮数部を表すビット列において、演算による桁落ちの影響を受けていない部分のビット長を有効桁数とする。」

浮動小数点演算の誤差には、丸め、情報落ち、桁落ちがあるが、今回は桁落ちのみを考慮した。すなわち、有効桁数が、そのまま計算精度を正確に示すわけではない。

以前、我々はC++を使い浮動小数点演算をソフト的にシミュレートし、演算過程における桁落ちを演算と同時にカウントし、有効桁数を常に監視するシステムを作成した。[1][2] そのシステムは、桁落ち追跡の他にも、区間演算が行なえる、仮数部長は任意に変更できより正確な値が計算できる、という特徴を持っていたが、浮動小数点演算をソフト的にシミュレートするために、実行速度が非常に遅かった。

今回作成したシステムは、桁落ちによる有効桁数の監視だけを行うが、演算部分はソフト的にシミュレートするのではなく、ハードウェアを使用するため、実行速度は格段に速くなった。まだ1桁の差はあるが、実用にそれほど問題は無いだろう。

表 1:　実行速度比較 (10,000,000 回の演算)
SPARCstation2 [単位：秒]

	普通の double	以前のソフトウェアシミュレーション	今回の方法
加算	3	36,224	124
減算	3	44,612	123
乗算	6	19,730	57
除算	14	64,710	65
加減乗除	22	123,770	350
比率	1	5,626	16

*The Algorithm for following up cancellation of significant digits , Hiroshi Suzuki (Tokyo Metopolitan College of Aeronautical Engineering), Hajime Ohiwa(Keio University)

3　桁落ち追跡のアルゴリズム

倍精度浮動小数点数は、符号1bit, 指数部11bit, 仮数部52bitの8バイトからなる。また、プログラミング言語C, C++などでは、変数の格納アドレスを使うことができ、それによって計算機のメモリ内部から変数の指数部のみを取り出すことができる。

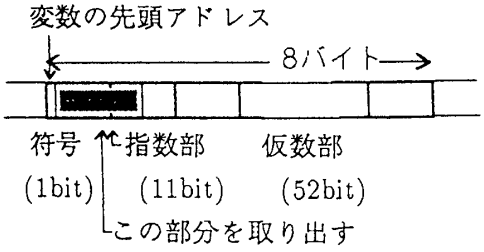

図1 メモリ内部の浮動小数点表現

演算の結果変数が桁落ちした場合は、仮数部は正規化され、結果的には指数部が減少するわけである。このことから、逆に演算の前後での指数部の変化を見れば、何桁桁落ちしたかがわかる。

C++のオペレータオーバーロードを使えば、例えば加算の演算子 + に対して、加算以外のことも定義できる。すなわち、加算と同時に演算前後の指数部を比較する機能を持たせれば、桁落ちを常に監視できるわけである。今回はさらに、桁落ちの影響を受けていない有効桁数も同時に計算している。

さて、有効桁数の計算方法であるが、加減算の場合は、$z = x - y$とすると、x, yの大きさによって2つのパターンに分かれる。1つめは、指数部の大きさが同じ場合であり、この場合はxとyの有効桁数の小さいほうから桁落ちの桁数を引くことによって演算後の有効桁数を得る。2つめは、指数部の大きさが異なる場合であり、この場合は、桁合わせが行われるので、指数部の小さいほうの有効桁数に桁合わせが行われる桁数を加えた上で、有効桁数の小さいほうから桁落ちの桁数(ただし、指数部の大きいほうと比較する)を引く。

乗除算の場合は、桁落ちは起こらないので、2数の有効桁の少ないほうを新しい有効桁とする。

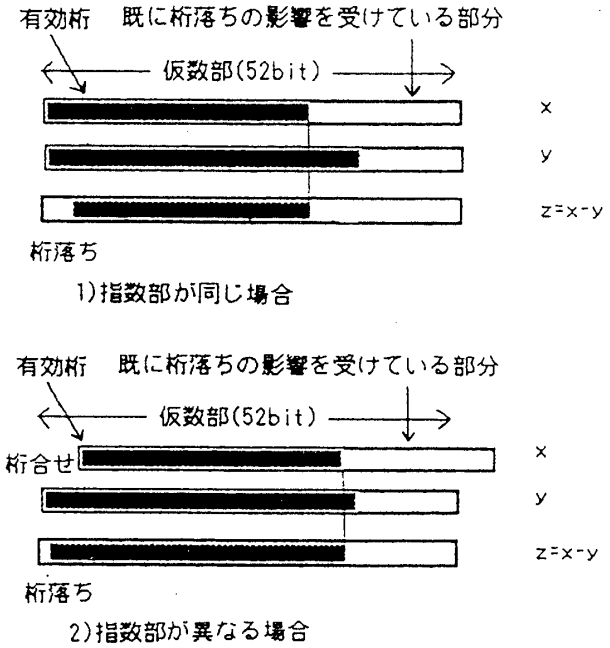

図2 減算のパターンと有効桁数計算

4　まとめ

実行速度が前回よりも格段に早くなり、実用に差し支えない速度と言える。

C++が動作すれば、他機種への移植も容易である。

桁落ちのみを考慮した有効桁数は、実際の精度よりも小さく見積ってしまう場合もあれば、丸め、情報落ちを考慮していないために、実際の精度よりも大きく見積ってしまう場合もあり、実際の計算精度とは多少違う。しかし、桁落ちがあることを利用者に知らせ、注意を促すことは出来る。実行速度を多少犠牲にしても計算結果に信頼性を持たせる価値はあるのではないだろうか。

[参考文献]

[1] 鈴木、大岩：桁落ちを考慮した浮動小数点演算の有効桁、情報処理学会 第19回数値解析シンポジウム 98-101,1990

[2] 鈴木、大岩：桁落ちを追跡する浮動小数点演算、情報処理学会第42回全国大会講演論文集 1-57, 1991