日本応用数理学会論文誌
Vol. 7, No. 1, 1997, pp. 79～89

IEEE754規格を利用した丸め誤差の測定法について

幸谷智紀* 永坂秀子**
*石川職業能力開発短期大学校 **日本大学理工学部

On the Round-off Error Estimation Using IEEE754 Floating-Point Arithmetic

Tomonori Kouya* Hideko Nagasaka**
*Ishikawa Polytechnic College
**College of Science and Technology, Nihon University

Abstract. We propose the round-off error estimation using IEEE754 floating-point arithmetic which is widely used on various workstations and personal computers. Interval Analysis is one of the standard methods in order to estimatite round-off errors in numerical processes, but it is not more convenient for a lot of users. In this paper, we explain the proposed estimation and show the source programs which can run on SparcStaion, PC-9801 and IBM PC/AT compatible. Finally, numerical experiments demonstrate the effectiveness of the proposed estimation.

1 はじめに

数値計算の誤差解析には一般に理論誤差(打ち切り誤差・離散化誤差)と丸め誤差の双方を含んだものである。本論文ではこのうち丸め誤差をどのように測定するかということついて考える。

例えば，有限回の代数的操作で連立一次方程式の真の解が得られるGaussの消去法の場合，解に含まれる誤差は，係数行列，定数項に含まれていた初期誤差が無ければ、全て演算中に発生した丸め誤差が原因である。丸め誤差は演算回数と演算精度に依存する。その証拠に，演算精度を増やせばそれに伴って誤差も縮小する。従って，丸め誤差の測定ができれば，数値解の誤差の量も判明する。消去法に限らず，丸め誤差を測定することは誤差解析，および数値解の精度保証を行う上でも重要な仕事の一つである。

現在，精度保証を機械的に行なうにはInterval Analysisを利用したパッケージが使われることが多い。信頼性が高い上，高速微分を利用するなど様々な改良が施されており，将来に渡って有力な解析手段である。しかし，パッケージを利用するためには，精度を測定したいプログラムの演算部分を全て書き換える必要がある。C++言語など，演算子のオーバーロードができる言語であればその必要はないが，数値計算の全てのプログラムがそのような言語で書かれているわけではない。既存のプログラムに適用するには手間がかかることは否めない。

そこで，有効桁の末尾bitを変化させて，それによる計算値の相違を比較して丸め誤差をはかる方法が山下眞一郎[7]により考案されている。これならば次のようなメリットがある。

- プログラム中のアルゴリズムの主要な部分を変える必要がない。
- 有効桁数を増やす必要がない。

実用上はこれでかなり良好な測定が可能になるが，ユーザがプログラムでこれを実行するには浮動小数点数のフォーマットを熟知した上，仮数部のビット操作を行なうルーチンを作らねばならず，かなり煩雑で

ある。一部のコンパイラのオプションとしてこの機能が使えるものもあるが，プログラム全体の計算が変化してしまう上，特定のコンパイラがないと使用できないことになる。

我々が提案する丸め誤差測定法は，山下の方法の利点を生かしつつ，丸め方法を変化させるだけで，数値計算における丸め誤差を測定することが出来る。現在主要なCPU(+FPU)で広範に使用されているIEEE754規格の浮動小数点演算の丸めモード設定の機能を使用すれば，本論文の手法は手軽に利用できる。本論文では具体例として連立一次方程式を取り上げ，実際に丸め誤差を測定する。IEEE754規格の浮動小数点数を採用しているSparcStationとPC9801, IBM PC/AT互換機(DOS/V機)で実行可能な，現在使用されている主要なコンパイラで実行可能なプログラム例も併せて提示する。

2 IEEE754規格の丸めモードと誤差の測定法

IEEE754規格では次の４つの丸めモード(RNモード, RPモード, RMモード, RZモード)が採用された[2]。

RNモード：最も近い値に丸める(Round to Nearest)

RPモード：$+\infty$に向かって丸める(Round toward Plus infinity)

RMモード：$-\infty$に向かって丸める(Round toward Minus infinity)

RZモード：0に向かって丸める(Round toward Zero)

RNモードはデフォルトの丸めモードであり，2進数のIEEE754規格では，0捨1入になる。プラス方向へのバイアス誤差をなくすため，仮数部を2進整数と見たときにそれが最も近い偶数になるように設定されている。

RPモード, RMモードはいわゆる「切り上げ」にあたるモードである。$+\infty$と$-\infty$という２方向への丸めが用意されているため，Interval Analysisなどに応用することができる。

RZモードは「切り捨て」に当たる。このRZモードはRPモード, RMモードとは異なり，浮動小数点数の符号に依存せずに丸め方向が0に向いている。

以上の丸めモードは現在使用されている多数のワークステーション，パソコンに搭載されているCPU内の浮動小数点演算ユニット，あるいはCPUとは別のプロセッサとして搭載し浮動小数点演算を行なうFPU内のコントロールレジスタ内に保持されている。特にユーザーが指定しない限りデフォルトは丸め誤差が小さく抑えられ，しかも蓄積されにくいRNモードになっている。このコントロールレジスタの値を変更するにはプロセッサが用意している変更用の命令を実行するだけでよい。

丸め誤差の測定は次のようにして行なう。以下，$R(x)$は$x \in \mathrm{R}^n$に含まれる丸め誤差の絶対値を表わすものとする。

1. デフォルトのRNモードで計算を実行する。ここで得られた計算値をx^{RN}と書くことにする。
2. 同じ計算をRZ, RP, RMモードでそれぞれ実行する。同様に計算値をx^{RZ}, x^{RP}, x^{RM}とする。
3. $E(x^{RN})$を次のように決める。

$$E(x^{RN}) = \max\{\|x^{RN} - x^{RZ}\|_\infty, \|x^{RN} - x^{RP}\|_\infty, \|x^{RN} - x^{RM}\|_\infty\}$$

4. $E(x^{RN})$のorder $\approx \|R(x^{RN})\|_\infty$のorderとして丸め誤差を推定する。

この測定法には2つの利点がある。一つは手順が簡単で，計算の主要部分のサブルーチンの変更が必要ないことである。プログラム例は次の節で示す。

もう一つの利点は，丸め誤差の伝播の追跡が可能になることである。実際の計算では，丸め誤差の蓄積よりも桁落ちによって丸め誤差が拡大されることによる影響が大きいことが知られている。最も影響が大きい丸め誤差そのものを丸めモード変更により変化させることで，誤差伝播の影響をスピーディーに，しかもライブラリルーチンの変更を一切行なわずに調べることができる。但し，最も影響が大きいRNモードで発生する丸め誤差がRZモードでも変化しなければ，誤差伝播の影響をとらえることは難しくなる。そこで，測定の正確さを期すためにRP, RMモードでも計算することが望ましい。さすれば，RN以外の全てのモードで発生する丸め誤差が変化しない可能性は皆無になる。RNモードでの計算値と他の3つのモードでの計算値の差異が最も大きくなるものを採用することで誤差伝播の影響を見逃すことはなくなる。

以上が測定法の概要，および利点である。これを10進演算を使用した簡単な数値例で示すことにする。

Example 1 (丸めモード変更による丸め誤差測定法)

次のような3項から成る和を，10進数の仮数部5桁の浮動小数点数で計算する問題を考える。

$$S_3 = \sum_{i=1}^{3} x_i \tag{2.1}$$

まず，$x_1 = 1.2345 \times 10^1$, $x_2 = 9.8765 \times 10^2$, $x_3 = 3.1415 \times 10^3$の場合を考える。このとき，RNモードで計算した値S_3^{RN}は

$$S_3^{RN} = 4.1415 \times 10^3$$

である。RZモードで計算した値S_3^{RZ}は

$$S_3^{RZ} = 4.1414 \times 10^3$$

であるから，この差の絶対値を評価値$E(S_3^{RN})$とすれば

$$E(S_3^{RN}) = 1.0000 \times 10^{-1}$$

となる。$S_3 = 4.14149 \times 10^3$が真値であるから，実際のRNモードでの丸め誤差の絶対値$R(S_3^{RN})$は

$$R(S_3^{RN}) = 1.0000 \times 10^{-1}$$

となり，この測定法による評価値$E(S_3^{RN})$と全く一致する。

次に，桁落ちが発生し，RNモードでの計算値とRZモードの計算値が全く一致する場合を考える。$x_1 = 1.2344 \times 10^1$, $x_2 = 9.8765 \times 10^2$, $x_3 = -9.9999 \times 10^2$とする。

RNモードおよびRZモードでの計算値は

$$\begin{aligned} S_3^{RN} &= 0.0000 \\ S_3^{RZ} &= 0.0000 \end{aligned}$$

となり，実際の丸め誤差$R(S_3^{RN}) = 4.0000 \times 10^{-3}$の評価をすることができない。ここで，RPモードで同じ計算を行なうと

$$S_3^{RP} = 1.0000 \times 10^{-2}$$

となり，この値とS_3^{RN}との差の絶対値を評価値$E(S_3^{RN})$とすれば，

$$E(S_3^{RN}) = 1.0000 \times 10^{-2}$$

となり，適切な丸め誤差の推定値を与えることができる。

3 プログラム例

この丸めモードはIEEE754規格の中に規定されているため，現在使用されている殆ど全てのワークステーション，パソコンのCPU(FPU)とコンパイラであれば問題なく使用できるものである。但し，OS及びコンパイラの種別によって利用の仕方を変える必要が出てくる。この節ではSparcStationとPC9801, IBM PC/AT互換機(DOS/V機)上のコンパイラでの丸めモード設定のための関数を提示する。

まず，SUN SparcStation(使用OS:Solaris 2.3(SUN OS 5.3 + OpenWindows 2.3))上でANSI準拠のCコンパイラを使用したプログラム例を示す。

丸めモード変更も含めた，IEEE754規格の浮動小数点数の設定・関数・変数・定義を納めたヘッダは“ieeefp.h”である。これをincludeすると現在の丸めモード設定を得る関数“fpgetround(void)”と丸めモードを設定する関数“fpsetround(fp_rnd)”が使用できる。この２つの関数と，丸めモードの定義“FP_RN”(RN), “FP_RP”(RP), “FP_RM”(RM), “FP_RZ”(RZ)を使って，

void view(void) ：現在の丸めモード設定を表示する関数

void set(fp_rnd) ：丸めモードを設定する関数

という２つの関数を定義する(Fig.1)。なお，この関数はgcc 2.7.0でコンパイル及び実行可能であることを確認済みである。

同様の関数をPC-9801上で動作するように変更したのが，Fig.2である。使用コンパイラはBorland C++ Ver.3.00である。DOS/V機でも同じコンパイラであれば全く同じプログラムを使うことができる。

SparcSationと異なり，浮動小数点演算ユニットのコントロールレジスタに丸めをセットする関数も，現在の設定を取り出すのも同じ関数“_control87”を使用している。現在の設定はコントロールレジスタ(16bit)をそのままunsigned intとして返すため，丸めモードの部分のみ取り出すために，ビット演算を使用してる。また，定義や変数型は全てSparcStationのそれ(Fig.1)と統一してある。

以上で取り上げた関数は全てC言語を使用して書いてあるが，現在ではMixed Language環境が整っており，必要に応じてFortranから呼び出すことも簡単にできる[1]。次にその例を示す。

Fig.1で定義した関数を使い，SparcStation上でFortranでも使用可能にしたプログラムがFig.3である。このCプログラムをコンパイルし，Fortranプログラムとリンクすることで，C言語と同様，次の２つのサブルーチンをCALL文で呼び出し，使用することが可能になる。

VIEW ：現在の丸めモードを表示する。引数なし。

SET(MODE) ：丸めモードをセットする。引数(MODE)は整数で設定する。MODE = 1...RNモード，MODE = 2...RPモード，MODE = 3...RMモード，MODE = 4...RZモードとする。

これらのプログラムを利用して，C言語でもFortranでも丸めモードの変更が自由にできるようになる。その使用例がFig.4である。左のC言語のプログラムは，

SparcStationの場合は，Fig.1を

PC-9801, DOS/V機の場合は，Fig.2を

それぞれリンクさせて実行する。左のFortranプログラムは，Fig.1とFig.3をそれぞれリンクし，SparcStation上で実行する。

```
#include <ieeefp.h>
/*************************/
/* View current round mode */
/*************************/
void view(void)
{
    fp_rnd currentmode;

    currentmode = fpgetround();
    printf("---- Round Mode ---- ");
    switch(currentmode)
    {
        case FP_RN: /* round to nearest number */
            printf("NEAREST");
            break;
        case FP_RP: /* round toward plus infinity */
            printf("PLUS INFINITY");
            break;
        case FP_RM: /* round toward minus infinity */
            printf("MINUS INFINITY");
            break;
        case FP_RZ: /* round toward zero */
            printf("ZERO");
            break;
    }
    printf(" ----\n");
}

/*****************/
/* Set Round Mode */
/*****************/
void set(fp_rnd mode)
{
    fpsetround(mode);
}
```

Fig. 1. The C program to change and to view the current round mode.(ANSI C, SparcStation + Solaris 2.3)

```
#include <float.h>

#define FP_RZ    RC_CHOP
#define FP_RP    RC_UP
#define FP_RM    RC_DOWN
#define FP_RN    RC_NEAR
typedef unsigned int fp_rnd;

/*************************/
/* View current round mode */
/*************************/
void view(void)
{
    fp_rnd currentmode;

    currentmode = _control87(CW_DEFAULT, 0);
    currentmode |= (0xffff ^ MCW_RC);
    currentmode -= (0xffff ^ MCW_RC);
    printf("---- Round Mode ---- ");
    switch(currentmode)
    {
        case FP_RN: /* round to nearest number */
            printf("NEAREST");
            break;
        case FP_RP: /* round toward plus infinity */
            printf("PLUS INFINITY");
            break;
        case FP_RM: /* round toward minus infinity */
            printf("MINUS INFINITY");
            break;
        case FP_RZ: /* round toward zero */
            printf("ZERO");
            break;
    }
    printf(" ----\n");
}

/*****************/
/* Set Round Mode */
/*****************/
void set(fp_rnd mode)
{
    _control87(mode, MCW_RC);
}
```

Fig. 2. The C program to change and to view the current round mode.(Borland C++ Ver.3.00, PC-9801 and IBM PC/AT Compatible)

```
/* for Fortran */
void view_(void)
{
    view();
}

/* for Fortran */
void set_(int *mode)
{
    fp_rnd fp_mode;

    fp_mode = fpgetround(); /* Default */

    /* 1: Nearest, 2: Plus Infinity, 3: Minus Infinity, 4: Zero */
    switch(*mode)
    {
        case 1:
            fp_mode = FP_RN;
            break;
        case 2:
            fp_mode = FP_RP;
            break;
        case 3:
            fp_mode = FP_RM;
            break;
        case 4:
            fp_mode = FP_RZ;
            break;
        default: /* No Change the Current Round Mode */
            break;
    }

    set(fp_mode);
}
```

Fig. 3. The Fortran program to change and to view the current round mode.(SparcStation + Solaris 2.3)

C

```
...
main()
{
/* View the Current Round Mode */
    view();
/* Change the Current Round
              Mode to RZ Mode*/
    set(FP_RZ);
      ...
    (Main Processes)
       ...
}
```

Fortran

```
      REAL*8 X, Y
C
C  View the Current Round Mode
C
      CALL VIEW
C
C Change the Current Round
C             Mode to RZ Mode
C
      CALL SET(4)
        ...
      (Main Processes)
        ...
      STOP
      END
```

Fig. 4. The C and Fortran programs using the functions to change and to view the current round mode.

4 数値実験

前節までに，SparcStation, PC-9801, DOS/V機で使用できる，現在の丸めモードを表示する関数“view(void)”と丸めモード設定関数“set(fp_rnd)”を定義した。この節ではこの2つの関数を使用し連立一次方程式

$$Ax = b \tag{4.1}$$

をGaussの消去法で解いた際の丸め誤差を測定する。この節の計算は全てSparcStation LX + gcc 2.7.0の倍精度計算を使用して行なった。

Example 2 (Hilbert行列)
係数行列Aを5次のHilbert行列

$$A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix}$$

とし，3種類の真の解x_1, x_2, x_3を用意する。

$$\begin{aligned} x_1^T &= [1\ 1\ 1\ 1\ 1] \\ x_2^T &= [1\ 2\ 3\ 4\ 5] \\ x_3^T &= [-1\ 1\ -1\ 1\ -1] \end{aligned}$$

これに伴って，3種類の定数項b_1, b_2, b_3ができる。

$$\begin{aligned} b_1^T &= \left[\frac{137}{60}\ \frac{29}{20}\ \frac{153}{140}\ \frac{743}{840}\ \frac{1879}{2520}\right] \\ b_2^T &= \left[5\ \frac{71}{20}\ \frac{197}{70}\ \frac{657}{280}\ \frac{1271}{630}\right] \\ b_3^T &= \left[-\frac{47}{60}\ -\frac{23}{50}\ -\frac{109}{420}\ -\frac{167}{840}\ -\frac{409}{2520}\right] \end{aligned}$$

この3つの問題を解き，x^{RN}とx^{RZ}, x^{RP}, x^{RM}との差の絶対値の最大値$\|x^{RN} - x^{RZ}\|_\infty$, $\|x^{RN} - x^{RP}\|_\infty$, $\|x^{RN} - x^{RM}\|_\infty$，及び$x^{RN}$の真の丸め誤差の絶対値の最大値$\|R(x^{RN})\|_\infty = \|x^{RN} - x\|_\infty$を比較する(Table 1)。1, 2番目の問題では，全ての測定値が真の誤差の上界になっている。3番目の問題では，真の解が± 1と符号が互いに反転しているため，丸めモードの違いが出にくいため，RMモードとの差ではかなり真の誤差に近い測定値を示している。

Table 1. Hilbert Matrix.(SparcStation + gcc 2.7.0)

Number	$\|R(x^{RN})\|_\infty$	$\|x^{RN} - x^{RZ}\|_\infty$	$\|x^{RN} - x^{RP}\|_\infty$	$\|x^{RN} - x^{RM}\|_\infty$
1	3.336E − 12	1.037E − 11	1.875e − 11	1.037e − 11
2	1.043E − 11	2.439e − 11	2.552E − 11	2.439E − 11
3	1.162E − 11	7.644E − 13	1.140E − 11	8.202E − 12

Example 3 (三重対角行列)
係数行列Aを三重対角行列

$$A=\begin{bmatrix} 2 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & 1 & 2 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & 2 & 1 \\ & & & & 1 & 2 \end{bmatrix}$$

とし，前の例題と同様，真の解xを

$$x^T=[1\ 1\ \ldots\ 1]$$

とする。さすればbは

$$b^T=[3\ 4\ \ldots\ 4\ 3]$$

となる。

この三重対角行列は次元数が高くなるに従って条件数が大きくなることが知られている。そこで，次元数が10, 100, 1000の場合を実際に解き，x^{RN}とx^{RZ}, x^{RP}, x^{RM}との差の絶対値の最大値$\|x^{RN}-x^{RZ}\|_\infty$, $\|x^{RN}-x^{RP}\|_\infty$, $\|x^{RN}-x^{RM}\|_\infty$，及びx^{RN}の真の誤差の絶対値の最大値$\|R(x^{RN})\|_\infty$を比較する(Table 2)。いずれの丸めモードにおいても全て丸め誤差の良好な推定値に収まっており，誤差限界を与えている

Table 2. Tridiagonal Matrix.(SparcStation + gcc 2.7.0)

Dimension	$\|R(x^{RN})\|_\infty$	$\|x^{RN}-x^{RZ}\|_\infty$	$\|x^{RN}-x^{RP}\|_\infty$	$\|x^{RN}-x^{RM}\|_\infty$
10	1.554E − 15	2.220E − 15	2.775E − 15	2.220E − 15
100	4.352E − 14	6.261E − 14	6.339E − 14	6.261E − 14
1000	1.045E − 12	1.783E − 12	1.822E − 12	1.783E − 12

ことがわかる。

Example 4
統計的な傾向を調べるために，50次元の係数行列A，真の解xを整数の一様乱数でそれぞれ10000個の例題として作り，それを前の例題と同様に消去法で解いた。A, xを与える関数をFig.5に示す。一様乱数を与える関数randはSolaris 2.4の標準ライブラリのものを使用した。乱数のseedは問題番号(1〜10000)を与え，問題毎に異なる乱数列を発生させるようにした。

その結果をFig. 6に示す。縦軸が$\log_{10}$(真の相対誤差の絶対値/評価値)を，横軸で示した各例題(1〜10000)が何桁の過小・過大評価となっているかを示したものである。グラフから明らかなように，1桁以上の過小評価となった事例は一つも発見できなかった。これを表にするとTable 3のようになる。実際の相対誤差より1桁程度，過大評価になっている例題が最も多いことが分かる。

```
void get_data(int seed)
{
    int i, j;

    srand((unsigned int)seed);

    for(i = 0; i < DIM; i++)
        tx[i] = rand();
    for(i = 0; i < DIM; i++)
    {
        b[i] = 0.0;
        for(j = 0; j < DIM; j++)
        {
            a[i][j] = (double)rand();
            b[i] += a[i][j] * tx[j];
        }
    }
}
```

Fig. 5. Generating random matrices.

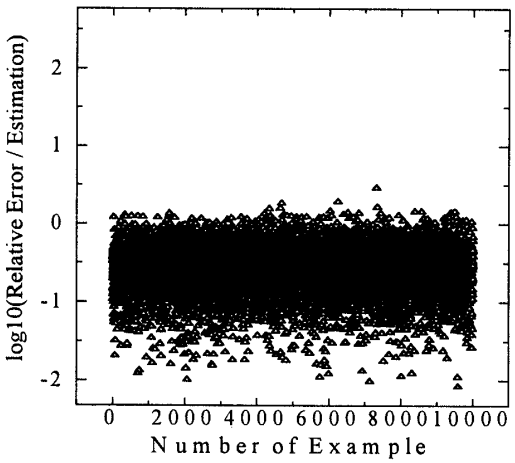

Fig. 6. Random Matrices (Dimension: 50).

Table 3. Random Matrices (Dimension: 50).

$\log_{10}\frac{\text{Relative Error}}{\text{Estimation}}$	Number of Examples
$-2.5\sim-2.0$	5
$-2.0\sim-1.5$	84
$-1.5\sim-1.0$	807
$-1.0\sim-0.5$	4876
$-0.5\sim0.0$	4119
$0.0\sim0.5$	109

5 結論

以上より次のことが判明した。

1. 本論文で提案した丸め誤差評価法は簡単なプログラムによって，特別なツールやライブラリを使用することなく実行できる。
2. 精密な丸め誤差の測定を行なうには，RNモード以外の３つ全てのモードで計算を行ない，その差が最大になるところ見つける必要がある。主たる計算部分には全く変更が必要ないため，最大３倍の計算量で済む。
3. 数値実験により，この評価法は真の丸め誤差に近い評価値を与えることが出来ることが示された。

今後はこの判定法を利用して，広く様々な数値計算に適用し，丸め誤差の蓄積と伝播による影響を調べて行きたい。

謝辞 本稿に対して有益な助言をいただき，議論に付き合ってくださった山下眞一郎氏，戸川隼人教授，査読者に感謝致します。

参考文献

[1] K. Dowd, High Performance Computing, インターナショナル・トムソン・パブリッシング・ジャパン (1994).

[2] インターフェース編集部 編, 数値演算プロセッサ, CQ出版社(1987).

[3] 川上一郎, 数値計算の基礎, 日本大学大学院テキスト(1993).

[4] 永坂秀子, 計算機と数値解析, 朝倉書店(1980).

[5] J.H.Wilkinson, Algebraic Eigenvalue Problem, Clarendon Press(1967).

[6] 山下眞一郎, On the Error Estimation in Floating-point Arithmetic, 日本大学博士論文, 1974.

[7] 山下眞一郎, 浮動小数点演算の誤差, bit別冊 数値計算における誤差, pp.14-26, 共立出版(1976).

幸谷智紀 (正会員) 〒927 石川県鳳至郡穴水町由比ヶ丘イ-45

1993年 日本大学大学院理工学研究科修士課程修了，同年 石川職業能力開発短期大学校情報処理科講師，現在に至る。数値解析，誤差解析の研究に従事。情報処理学会会員。

永坂秀子 (正会員) 〒410 東京都品川区荏原4-4-12 グリーンプラザ武蔵小山505

1949年 都立女子専門学校数学科卒業，同年 東京都立大学助手，1959年 日本大学助手。助教授，教授を経て，1992年 定年退職，同年 日本大学理工学部非常勤講師，現在に至る。理学博士。数値計算の誤差解析の研究に従事。情報処理学会，日本数学会各会員。

(1996年2月7日受付)