Gen. Math. Notes, Vol. 2, No. 1, January 2011, pp. 209-220
ISSN 2219-7184; Copyright © ICSRS Publication, 2011
www.i-csrs.org
Available free online at http://www.geman.in

A Study of First-order Disturbance of the Second Order Approximation of a Vectorial Function with Application in Engineering other and Sciences

Cristina Tănăsescu and Amelia Bucur

Lucian Blaga University of Sibiu, Faculty of Economics Sciences,
Str.Calea Dumbrăvii, No.17, 550324, Sibiu, Romania.
E-mail: cristina.tanasescu@ulbsibiu.ro
Lucian Blaga University of Sibiu, Faculty of Science, Department of Mathematics,
Str. I.Ratiu,No.5-7, 550012, Sibiu, Romania.
E-mail: amelia.bucur@ulbsibiu.ro

(Received 23.11.2010, Accepted 21.12.2010)

Abstract

The subject of this article is to present a study of first-order disturbance of second-order approximation of a vectorial function in the presence of measurement and numerical computation errors, with applications in engineering and other sciences. The study can be used in the approximations of functions of one or more variables from any other domain of activity.

Keywords: *measurement errors, errors in numerical computation, errors of propagation.*

1 Introduction

Any experimental measurement is affected by errors. After the cause that produces them, they can be divided into three categories: systematic errors, random errors and trivial errors.

1. **Systematic errors** have three possible sources: *observer errors, instrument errors and errors of method* [2]. Whatever are the causes of systematic errors, they have a common characteristic: it is assumed that the value of an individual measurement is the same whenever we repeat the measurement, and hence the error is the same.

The absolute error δ_x of a measured size x represents the module of the maximum difference possible between the measured value and the true one, and the relative error ε_x is the ratio between the absolute error and the module of the true value, being given by the ratio between the absolute error and the module of the measured value (obviously, with the condition that the denominator to be nonzero).

Then, if a measure indirect determined is given by:

$$z = x \pm y \quad , \tag{1}$$

its absolute error is

$$\delta_z = \delta_x + \delta_y \quad , \tag{2}$$

and if the measure is given by:

$$z = xy^{\pm 1} \quad , \tag{3}$$

its relative error is

$$\varepsilon_z = \varepsilon_x + \varepsilon_y \quad . \tag{4}$$

2. **Random errors** are determined by statistical considerations. Experience shows that the directly measured quantities are of two possible types: discreet (eg number of impulses recorded by a detector) and continous.

The theoretical analysis of statistics of the discrete measures proves that their values are distributed according Poisson probability distribution

[1]. According this, the probability to obtain a number n of impulses at a measurement is

$$p(n) = e^{-a} \frac{a^n}{n!} \quad ,$$

(5)

where

$$a = \sum_{n=0}^{\infty} np(n)$$

(6)

is the "true" value of the number of impulses (and, in general, it is a real number), and the error with which was determined the number a (the standard error or the root mean square deviation) is

$$\sigma_a = \sqrt{\sum_{n=0}^{\infty} (n-a)^2 \, p(n)} = \sqrt{a} \quad .$$

(7)

If we make a number N of measurements in identical conditions, obtaining the values $n(1)$, $n(2)$, ... , $n(N)$, then the estimated true value is given by the mean value:

$$\tilde{n} = \frac{1}{N} \sum_{i=1}^{N} n(i).$$

(8)

The error that affects an individual measurement $n(i)$ would be then

$$\sigma_{n(i)} = \sqrt{n(i)}$$

(9)

and that of the mean value would be

$$\sigma_{\tilde{n}} = \sqrt{\frac{\tilde{n}}{N}} \quad .$$

(10)

Regarding to the case of continuous measures, statistics show that the values of these measures are distributed according to normal distribution

(Gauss)[1]. Let's consider first the case of a single measure x. Its probability density will then

$$\mathrm{P}(x) \equiv \frac{dp(x, x+dx)}{dx} = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left[-\frac{(x-a_x)^2}{2\sigma_x^2} \right], \qquad (11)$$

where

$$a_x = \int_{-\infty}^{\infty} x \mathrm{P}(x) dx$$

(12)

is its "true" value, and

$$\sigma_x = \sqrt{\int_{-\infty}^{\infty} (x-a_x)^2 \, \mathrm{P}(x) dx}$$

(13)

is its standard error. In the case in which we make a number N of measurements in identical conditions, obtaining the values $x(1)$, $x(2)$, ... , $x(N)$, then the estimated true value is given by the mean value:

$$\tilde{x} = \frac{1}{N} \sum_{i=1}^{N} x(i) \quad ,$$

(14)

the error which affect an individual measurement $x(i)$ will be

$$\sigma_{x(i)} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x(i) - \tilde{x})^2} \quad ,$$

(15)

and that of the mean value will be

$$\sigma_{\tilde{x}} = \frac{\sigma_{x(i)}}{\sqrt{N}} \quad .$$

(16)

3. **Trivial errors** are caused by careless or accidental damage and should be removed from the calculations. In general, this is easily done, because these values differ from other very much.

Also, influence on the interpretation of some results have **the numerical computation errors**.

It is known that a numerical computation formula is usually applied repeatedly. Consequently, it is not only important the error introduced in one stage, but its tendency to increase, or conversely, to mitigate errors previously introduced, namely the stability of the numerical method used.

Numerical error sources are:

1. inherent errors – errors related to the approximate knowledge of some values derived from measurements or by the fact that we are dealing with irrational numbers (algebraic or transcendental). Obviously, the results of any calculation depends also on the accuracy of the initial input data. As inherent errors can also be regarded the converting errors made in the transition to the second base of some numbers that are placed in the memory of current digital computers. For example, number 0.1 represented by a finite number of digits in the 10th base, becomes a decimal periodic ratio in the second base (0.110 = 0.0(0011)2).

2. method errors or **truncation errors** that come from the approximations made in the deduction of the calculation formulas. Examples: the rest RN(x) in the polynomial interpolation, distance from root, from the iterative methods of calculation, the error introduced by the formula of integration of the trapezes on an interval equal to h step, the error introduced by the truncation of the series at a certain rank, etc. Unlike the inherent errors, in principle, method errors can be reduced as much as possible.

3. rounding errors are related to the limited possibilities of representation of numbers in numerical computers. In general, any computer can represent the numbers using a reduced number of significant digits, depending on word length (number of bits) used to store a number. Currently it is working with an equivalent of about 7 significant digits in single precision and about 15 significant digits in double precision. As is known, the internal memory of current computers use floating point representation, in normalized form. Thus, any real number x is written:

$$x = f \cdot b^n, \qquad |f| < 1 \tag{17}$$

where f is a real number named *mantissa*, b > 0 (b ≠1) is the *base* of the numeration system used, and n (integer) is the *exponent*. In normalized form, the mantissa is in the range $[b^{-1}, 1)$

$$b^{-1} \leq |f| < 1 . \quad (18)$$

The only exception to this rule is the number zero representation. Consequently, a real number with more significant digits is "rounded" to the maximum number of digits. This is done by rounding the mantissa. Other roundings are performed during operations. Let t be the number of significant digits. For convenience, let's suppose that we are working under the tenth base (b = 10). Then, a number *x* whose initial value is supposed to be accurately known, will be written:

$$x = f \cdot 10^n + g \cdot 10^{n-t}, \quad |f| , |g| \in [0.1, 1), \quad (19)$$

where *g* contains the digits that cannot be included in the mantissa f. Rounding is usually made symmetrical, that is replaced

$$|g| = 1 \text{ dacă } |g| \geq 0.5, \quad |g| = 0 \text{ if } |g| < 0.5 . \quad (20)$$

In this way, the bound of the relative error is

$$|e_r| = |g| \cdot 10^{n-t} / |f| \cdot 10^n \leq 5 \cdot 10^{-t} . \quad (21)$$

The errors with the edge given by (21) are made in the introduction of real numbers in the numerical computer memory. They affect the results depending on the operations to which are undergone the input values.

2 Main Results

2.1. First-order perturbation of the second order approximation of a function

Both measurement errors and the numerical computation are propagated in calculations, leading to disruption of the values obtained. In the followings we will calculate the first order perturbation of second order approximation of a function of several variables. Also we will show that the method is applicable for functions involved in shaping technical problems, issues in engineering or functions that occur in other areas.

Let the function be $f: R^n \to R$, of class C^3 şi $x^0 = (x_1^0, \dots, x_n^0) \in R^n$ fixed. By applying the Taylor formula with the rest in the Lagrange form in the neighborhood of the calculation point x^0 results that there exists $\theta \in (0,1)$ so that noting with $\bar{x} = (1-\theta)x^0 + \theta x$ we have:

$$f(x) = f(x^0) + \frac{1}{1!}\, df(x^0)(x - x^0) + \frac{1}{2!} d^2 f(x^0)(x - x^0, x - x^0) + \frac{1}{3!} d^3 f(x^0)(\bar{x}, \bar{x}) =$$

=

$$f(x^0) + \left(\frac{\partial f(x^0)}{\partial x_1}, \dots, \frac{\partial f(x^0)}{\partial x_n}\right)(x - x_0) + \frac{1}{2!}(x - x^0)^T Hf(x^0)(x - x^0) + R_n$$

(22)

We note the sum of the first three terms on the right of the previous formula with $\bar{f}(x)$.

Note that function $\bar{f}(x)$ is hipersquared, having degree less or equal to 2. Then, if the degree of f is at his turn less or equal to 2, the following equality will take place $f = \bar{f}$.

Let's consider:

$$\delta_x = (\delta_{x_1}, \dots, \delta_{x_n})$$

(23)

a disruption of $\bar{f}$. arguments.

Then:

$$\bar{f}(x + \delta_x) = f(x^0) + \left(\frac{\partial f(x^0)}{\partial x_1}, \dots, \frac{\partial f(x^0)}{\partial x_n}\right)(x + \delta_x - x_0) + \frac{1}{2!}(x + \delta_x - x^0)^T Hf(x^0)(x + \delta_x - x^0) \sim \bar{f}(x) + \left(\frac{\partial f(x^0)}{\partial x_1}, \dots, \frac{\partial f(x^0)}{\partial x_n}\right)(\delta_x) + (x - x^0)^T Hf(x^0)\delta_x$$

(24)

if you neglect the term $\frac{1}{2}(\delta_x)^T Hf(x_0)\delta_x$.

Definition. *We call the first order perturbation second order approximation of the function f the expression*

$$E(x, \delta_x) = \left(\frac{\partial f(x^0)}{\partial x_1}, \dots, \frac{\partial f(x^0)}{\partial x_n}\right)(\delta_x) + (x - x^0)^T Hf(x^0)\delta_x$$

(25)

Observation. If the degree of f is less or equal to 2, then $f = \bar{f}$ and the point x^0 is not involved in the calculations. It can be chosen equal to 0.

As examples of functions of several variables on which to apply first-order disturbance model presented above we can choose:

Example 1. $f(x_1, x_2) = x_1^5 x_2 + 5x_1 x_2^3$ (for which it is created the first program in section II.2 of this article).

Example 2. $f(x_1, x_2) = 583{,}2 + 264{,}75\, x_1 - 195{,}25 x_2$ (function used in the calculation of the theoretical decomposition time, see [3])

Example
3. $f(x_1,x_2) = 55{,}73 + 15{,}793x_1 + 8{,}3122\,x_2 - 2{,}1629x_1^2 - 3{,}1744\,x_2^2$
(function used in the study dependence of the breaking strength of steel in relation to its chemical composition , see [3])
Example
4. $f(x_1,\dots,x_4) = 1{,}0431 - 0{,}2316x_1 + 0{,}1097x_2 - 0{,}0837x_3 + 0{,}2323x_4$
(function used in the study of laser engraving stainless steel, see [3])
Example
5. $f(x_1,\dots,x_4) = 11{,}866 + 0{,}832x_1 + 1{,}829x_2 - 2{,}218x_3 + 1{,}353x_1x_2 - 1{,}508x_1x_3 - 2{,}0722x_2x_3 - 2{,}728x_1^2 - 1{,}717x_2^2 - 1{,}619x_3^2 - 1{,}407x_4^2$ (function used in the study electrical erosion, see [3])
(for which it is created the second program in section II.2 of this article).
For applying the first order disturbance model to the functions chosen in these examples and
making programs by this model, we observe that all are of the form

$$f(x_1,\dots.,x_n) = c_{00} + 2c_{01}x_1 + \cdots + 2c_{0n}x_n + c_{11}x_1^2 + \cdots + c_{nn}x_n^2 + 2c_{12}x_1x_2 + \cdots + 2c_{n-1}c_nx_{n-1}x_n$$

(26)
hence,

$$E(x,\delta_x) = 2\sum_{i=1}^{n} c_{0i}\delta_{x_i} + 2\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_i\delta_{x_j}$$

(27)

2.2. Programs in C + +
We present below a C + + program that I created using the method of section II of this article, for the previously proposed function as example 1:

```
# include<iostream.h>
# include <conio.h>
float f(float x,float y)
{return x*x*x*x*x*y+5*x*y*y*y*y;}
float dfdx(float x,float y)
{return 5*x*x*x*x*x*y+5*y*y*y;}
float dfdy(float x,float y)
{return x*x*x*x*x+15*x*y*y;}
float d2fdx2(float x,float y)
{return 20*x*x*x*y;}
float d2fdxdy(float x,float y)
```

```
{return 5*x*x*x*x+15*y*y;}
float d2fdydx(float x,float y)
{return 5*x*x*x*x+15*y*y;}
float d2fdy2(float x,float y)
{return 30*x*y;}
float e,x,y,x0,y0,deltax,deltay,Deltax,Deltay;
void main()
{clrscr();
cout<<"x0=";cin>>x0;cout<<"y0=";cin>>y0;
cout<<"Delta x=";cin>>Deltax;cout<<"Delta y=";cin>>Deltay;
cout<<"delta x=";cin>>deltax;cout<<"delta y=";cin>>deltay;
x=x0+Deltax;y=y0+Deltay;
e=dfdx(x0,y0)*deltax+dfdy(x0,y0)*deltay;
e=e+d2fdx2(x0,y0)*Deltax*deltax;
e=e+d2fdxdy(x0,y0)*Deltax*deltay;
e=e+d2fdydx(x0,y0)*Deltay*deltax;
e=e+d2fdy2(x0,y0)*Deltay*deltay;
cout<<"e="<<e<<endl;
getch();
}
```

In the followings it is presented a C + + program that I created using the method of section II of this article, for the previously proposed functions as examples 2,3,4,5:

```
# include<iostream.h>
# include <conio.h>
float c[10][10],x[10],x0[1],Deltax[10],deltax[10],e;
int self,n,i,j,k;
char dorescfunctie,dorescx0,dorescx;
void main()
{clrscr();
 do
 {cout<<endl<<"The functions aret:"<<endl;
  cout<<"1. f(x1,x2)=583.2+264.75*x1-195.25*x2"<<endl;
```

```
 cout<<"2.  f(x1,x2)=55.73+15.793*x1+8.3122*x2-2.1629*x1^2-
3.1744*x2^2"<<endl;
 cout<<"3.  f(x1,x2,x3,x4)=1.0431-0.2316*x1+0.1097*x2-
0.0837*x3+0.2323*x4"
  <<endl;
 cout<<"4.  f(x1,x2,x3,x4)=11.866+0.832*x1+1.829*x2-
2.218*x3+1.353*x1x2-"<<endl;
 cout<<"-1.508*x1*x3-2.0722*x2*x3-2.728*x1^2-1.717*x2^2-
1.619*x3^2-1.407*x4^2"
  <<endl;
 cout<<endl<<"select the function (1,2,3,4,0):"; cin>>self;
 if((self<1)||(self>4))return;
 if(self==1)
 {n=2;for(i=0;i<=n;i++)for(j=0;j<=n;j++)c[i][j]=0;
  c[0][0]=583.2;c[0][1]=264.75/2;c[0][2]=-195.25/2;
  c[1][0]=264.75/2;c[2][1]=-195.25/2;
 }
 if(self==2)
 {n=2;for(i=0;i<=n;i++)for(j=0;j<=n;j++)c[i][j]=0;
  c[0][0]=55.73;c[0][1]=15.793/2;c[0][2]=8.3122/2;
  c[1][1]=-2.1629;c[2][2]=-3.1744;
 }
 if(self==3)
 {n=4;for(i=0;i<=n;i++)for(j=0;j<=n;j++)c[i][j]=0;
  c[0][0]=1.0431;c[0][1]=-0.2316/2;c[0][2]=0.1097/2;
  c[0][3]=-0.0837/2;c[0][4]=0.2323/2;c[1][0]=-0.2316/2;
  c[2][0]=0.1097/2;c[3][0]=-0.0837/2;c[4][0]=0.2323/2;
 }
 if(self==4)
 {n=4;for(i=0;i<=n;i++)for(j=0;j<=n;j++)c[i][j]=0;
  c[0][0]=11.866;c[0][1]=0.832/2;c[0][2]=1.829/2;c[0][3]=-2.218/2;
  c[1][0]=0.832/2;c[1][1]=-2.728;c[1][2]=1.353/2;c[1][3]=-1.508/2;
  c[2][0]=1.829/2;c[2][1]=1.353/2;c[2][2]=-1.717;c[2][3]=-2.0722/2;
  c[3][0]=-2.218/2;c[3][1]=-1.508/2;c[3][2]=-2.0722/2;c[3][3]=-1.619;
  c[4][4]=-1.407;
```

```
        }
        do
        {cout<<endl<<"enter the base point"<<endl;
         for(k=1;k<=n;k++)
         {cout<<"x0["<<k<<"]=";cin>>x0[k];
         }
         do
         {cout<<endl<<"enter the calculation point"<<endl;
          for(k=1;k<=n;k++)
          {cout<<"x["<<k<<"]=";cin>>x0[k];Deltax[k]=x[k]-x0[k];
          }
cout<<"enter the disturbances from the point of calculation "<<endl;
          for(k=1;k<=n;k++)
          {cout<<"deltax["<<k<<"]=";cin>>deltax[k];
          }
          e=0;
          for(k=1;k<=n;k++) e=e+(c[0][k]+c[k][0])*deltax[k];
          for(i=1;i<=n;i++) for(j=1;j<=n;j++)
          {e=e+c[i][j]*x[i]*deltax[j];
           if(i==j) e=e+c[i][i]*x[i]*deltax[i];
          }
          cout<<"e="<<e<<endl<<endl;
          cout<<"Do you want another calculation point?(y/n)?:";cin>>dorescx;
         }
         while((dorescx=='y')||(dorescx=='Y'));
         cout<<" Do you want another current point?(y/n):";cin>>dorescx0;
        }
        while((dorescx0=='y')||(dorescx0=='Y'));
        cout<<" Do you want another function?(y/n):";cin>>dorescfunctie;
       }
       while ((dorescfunctie=='y')||(dorescfunctie=='Y'));
      }
```

References

[1] C. Berbente, S. Mitran and S. Zancu, Numerical Methods, *Technical Publishing House*, Bucharest, (1997).

[2] M. Țîțu and C. Oprean, Experimental research and data processing.Part I, *Lucian Blaga University of Sibiu Publishing House*, (2006).

[3] M. Țîțu, C. Oprean and I. Tomuță, Experimental research and data processing.Part III, *Lucian Blaga University of Sibiu Publishing House*, (2007).