INPE-5420-PRE/1762

SEARCH FOR PERIODICITIES IN GEOPHYSICAL TIME SERIES BY ITERATIVE REGRESSION ANALYSIS IN C

DANIEL J. ROGER NORDEMANN

ACEITO PARA APRESENTAÇÃO NO 13º CONGRESSO NACIONAL
DE MATEMÁTICA APLICADA E COMPUTACIONAL

INPE
São José dos Campos
1992

SECRETARIA DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-5420-PRE/1762

SEARCH FOR PERIODICITIES IN GEOPHYSICAL TIME SERIES BY ITERATIVE REGRESSION ANALYSIS IN C

DANIEL J. ROGER NORDEMANN

ACEITO PARA APRESENTAÇÃO NO 13º CONGRESSO NACIONAL
DE MATEMÁTICA APLICADA E COMPUTACIONAL

INPE
São José dos Campos
1992

CDU: 550.3
PALAVRAS-CHAVE: PERIODICITIES; GEOPHYSICS; ITERATIVE REGRESSION; C LANGUAGE

SEARCH FOR PERIODICITIES IN GEOPHYSICAL TIME SERIES BY ITERATIVE REGRESSION ANALYSIS IN C.

DANIEL JEAN ROGER NORDEMANN

Instituto Nacional de Pesquisas Espaciais - INPE
CP 515, 12201 São José dos Campos, Brazil

ABSTRACT: In order to determine periodicities in time series from geophysical phenomena, an iterative regression analysis is performed in compiled C language. This program uses a least square regression described by Wolberg [1]; It provides the amplitude, period and phase of individual sine components with their uncertainties. Weighting input data according to their uncertainties is also incorporated.

KEY WORDS: Time series, periodicities, iterative regression analysis, C language, geophysics.

RESUMO: Para determinar as periodicidades em séries temporais oriundas de fenômenos geofísicos, um método iterativo de regressão foi desenvolvido em linguagem C. O tratamento usado é baseado num método de regressão por mínimos quadrados descrito por Wolberg [1] e aplicado à determinação da amplitude, pulsação e fase dos componentes senoidais individuais e de seus respectivos desvios padrão. A vantagem principal do método de Wolberg é sua capacidade de levar em conta as incertezas sobre os pontos experimentais através da ponderação dos dados.

PALAVRAS-CHAVE: Series temporais, periodicidades, regressão iterativa, linguagem C, geofísica.

1. INTRODUCTION

An iterative regression analysis was developed in compiled C language as a method for searching periodicities in geophysical time series. The process used is based on a regressive least square analysis described in Wolberg's book "Prediction Analysis" [1] and applied to the determination of the parameters of a sine functions, one at a time, within time series [2]. The main advantages of Wolberg's method is the imbedded capability to take into account the uncertainties on the experimental points through weighting of data and to give the standard deviations on the parameters of the searched function which best fits with the experimental data. The necessary condition of derivability on regard to the parameters and to the variable(s) is obviously satisfied when dealing with sine function. The other condition to realize is that the experimental points must lead to convergence during the search of the unknown parameters, which was achieved with most of data sets tested, when searching for the parameters of a sine function, if significant, within a time series.

2. ITERATIVE REGRESSION ANALYSIS FOR SINE FUNCTION IN C LANGUAGE.

The summary of the principle of the method used is given in Appendix A. The iterative regression analysis used has been already described in detail by Wolberg [1]. It has been applied to searching periodicities in time series, through a program in C language to be run in microcomputers and which is described in this article. It has also been used to prepare a Lotus 1-2-3* spreadsheet dedicated to the same problem [3].

The list of the program written in C, including comments is given in Appendix B. The presence of many lines of comments turns it quite self-explained for readers familiar with C language, even for beginners. Most of variables used are declared as global. As usual in C programming, the program is structured and divided in several functions, facilitating the

understanding of its articulation and also turning easier the substitution by the user of some functions with equivalent ones written with different code, such as for resolution of the linear system or inversion of matrix. The program has been written, compiled and linked within the Turbo C*** environment.

The compiled and linked (.EXE) ready to run version may be executed in any IBM-PC** compatible machine, without need of loading the Turbo C*** software. The execution time depends, of course, directly on the microcomputer (XT, 286 AT or 386 AT), on its clock frequency and on the presence of an arithmetic coprocessor. Some examples of running times are given ahead for comparison.

At the beginning, the user is asked through short character strings the names of the data file (drive, path, name and extension if any). Data need to be written in an ASCII file of 4 numerical columns: time, uncertainties on time, values which are function of time and their uncertainties. The data file is read just after entering its path and name, and input data are shown on the screen. The number of sample points is determined automatically from the number of data lines read.

The user is then asked the path and name of the result file. After this, the calculation begin. A descriptive header is shown and shortly after, the first line of running iteration is shown. The results appear on the monitor screen as lines containing the period, amplitude, pulsation, phase values, each one followed by its uncertainty, the sum of the square of the residues, as an indicator of convergence and the number of iterations already performed for the considered interval of frequency. The last line shown on the bottom of the screen describes the running iteration and is updated at each iteration. When a convergence is achieved, the running line is completed by an asterisk as an indicator of achieved convergence, and promoted one line higher, by vertical scrolling of the screen. This provides permanently the informations

relative to the last convergences achieved. The lines corresponding to achieved convergence are also saved, one by one in append mode, within the ASCII file dedicated to the results obtained.

Just after the interrogation on the path and name of input data, the user is asked if weighting is desired. This option is offered to make use of the possibility of the program to search for periodicities taking into account the uncertainties on data as entered in columns 2 and 4 of the input file. The "no weighting" option is offered to perform calculations with the following conditions: the weighting coefficients are substituted by a constant value 1, which is equivalent to even weighting or weighting with equal absolute uncertainty on all Y values and no uncertainty on time values.

The option to perform the search for periodicities with and without weighting on the data, has been included because it has been observed that the results obtained with the same set of data, with no weighting or with weighting with different uncertainties, may lead to different results, possibly including loss of convergence for some periodicities. This has to be carefully investigated, from the point of view of artificial time series and geophysical applications. From this remark, it seems highly advisable to perform, in most cases, the search for periodicities with and without weighting. This possibility is one of the most important advantages of this iterative regression analysis applied to time series.

3. TESTS AND COMPARISON OF PROCESSING TIME.

The search for periodicities was performed by two different microcomputers, on a 100-sample artificial time series treated by the program in C language described in this paper and also within the Lotus 1-2-3* spreadsheet already described [3]. The machines used were IBM-PC** compatible microcomputers: an XT machine running at 12.5 MHz, without hard disk and an AT 386 running at 25 MHz, with arithmetic coprocessor and hard disk.

For all the calculations performed, the same conditions were used: Number of data samples: 100; Pulsation sweep: from 3.15 to 0 ; Pulsation sweep interval: 0.025 ; Maximum number of iterations per interval: 30 ; Convergence if sum of squared corrections lower than 10^{-6}. The results on the processing time necessary to perform the search for periodicities, as described, is given in TABLE I.

TABLE I

Processing times.

		8088 XT 12 MHz no coprocessor		80386 AT 25 MHz with 80387	
	Size (KB)	No weight. (min)	With weight. (min)	No weight. (min)	With weight. (min)
TSIRA.C	6.3				
TSIRA.OBJ	11.0				
TSIRA.EXE	44.7	148	175	5	5

As it may be seen from Table I, there is roughly a factor of 30 between the processing times for the execution of the program with the machines tested. Weighting or not the data does not change very much the processing time. Of course any other programming style may give different results for these tests. A similar calculation was formerly developed to be run within a commercial spreadsheet. As expected, the program in C presents a much shorter processing time, approximately half time, in relation to the calculations performed in spreadsheet environment. The commercial spreadsheet solution has shown several advantages [3], but besides the shorter processing time, the program in C presents the following advantages:

- Input of data much easier in C, by reading an ASCII external file.
- Less memory occupation, and the same program may be easily used for calculations with or without weighting.
- Program easier to understand, to explain, to maintain or to modify.
- Other advantages of C language, including easy portability, for instance to more powerful machine like workstations.

4. CONCLUSIONS.

In this work, a review of the conditions of application of the iterative regression method described by Wolberg [1] to problems of regression and prediction analysis has been given. The main advantages of the method are its application to any derivable function, its capability of weighting the input data to take into account their reliability, the determination of the uncertainties associated to the calculated parameters and its easy programming in various languages in any micro-, mini- or main-frame computer.

Through the example of artificial series, it has been observed that besides the values of the parameters calculated, the analysis of their uncertainties represents a powerful tool to measure the reliability of the results, which has the highest importance in the case of prediction analysis. The implementation in a C language algorithm is, as it was shown in this work, very easy, and, in our opinion, this process is satisfactorily time efficient, which was not the case for the implementation in commercial spreadsheet.

ACKNOWLEDGMENT

The author wishes to thank Nalin Babulal Trivedi and Severino Luiz Guimarães Dutra for their participation in fruitful discussions and revision of the manuscript.

REFERENCES

[1] J. R. WOLBERG, Prediction Analysis, Van Nostrand, Princeton, NJ,1967.

[2] D. J. R. NORDEMANN, Pesquisa de periodicidades em séries temporais geofísicas por regressão iterativa em C, XIII Congresso Nacional de Matemática Aplicada e Computacional, Brazil, 1990.

[3] D. J. R. NORDEMANN, Pesquisa de periodicidades em séries temporais geofísicas por regressão iterativa em planilha, XIII Congresso Nacional de Matemática Aplicada e Computacional, Brazil, 1990.

* Lotus 1-2-3 is a trademark of Lotus Development Corporation, Cambridge, MA.

**IBM-PC is a trademark of International Business Machines Corporation.

*** Turbo C is a trademark of Borland International, Scotts Valley, CA.

APPENDIX A

The general principle of the iterative regression method is given in detail in Wolberg's book [1], solely dedicated to it. Here, we give only a simple presentation of the method, applied to the search of periodicities in time series; for this reason, we use the example of the fitting data to a three-parameter sine function in the form of $Y=a_0*\sin(a_1*t+a_2)$. To help follow the demonstrations given in Wolberg's book, the same symbols were used in most formulas; Underwritten indexes were used in this text whereas conventional square brackets were used for array indexes in the C code (Appendix B).

At the beginning, initial values are given to the parameters to allow the first iteration to be performed. Any initial value may be indicated, but practically there is a need to indicate initial values not too far from the unknown final values. This is needed generally to save time but also to help the algorithm converge to final values. The convergence may be easy, from any initial value, but in the case of fitting several sine function to the same set of data, which is the general case, one can figure that in the 4-dimension space (a "vertical" axis for the function F, as defined ahead, and three "horizontal" axis for the three parameters a_0, a_1 and a_2), the experimental data can generate a surface with several "valleys" and "cups", the bottom of which are convergence points for a set of parameters. A set of initial values outside a given cup may not lead to the bottom of the cup.

The starting point of the principle of the method is the error function which is calculated for every t through the following formula:

$$F = Y-a_0*\sin(a_1*t+a_2) \qquad (1)$$

where Y is the signal measured; t is the time and a_0, a_1 and a_2 are the three unknown parameters for one sine function. The demonstration of the process starts with the derivation in relation to the parameters of the sum of the squares of the function F, conveniently multiplied by individual weighting factors. This leads to a linear system, the coefficients and second members of which are sums, for all the experimental points, of products including these derivatives. The corrective terms A_0, A_1 and A_2 to apply to the parameters a_0, a_1 and a_2 respectively, are solution of this linear system.

In that way, an iteration gives as a result a set of corrective terms to apply, by subtraction, to the initial values used for this iteration, in order to obtain the new values of the parameters to begin with the next iteration. This process is replicated up to the moment when the corrective terms are smaller than a given value compatible with the precision of the results.

The values of A_0, A_1 and A_2 are solutions of the system:

$$[C]*[A]=[V] \tag{2}$$

with $$C_{ij}=(dF/da_i)*(dF/da_j)/L \tag{3}$$

$$V_i = (dF/da_i)*F/L \tag{4}$$

$$S = F*F/L \tag{5}$$

Formula (3), (4) and (5) represent the sum over all the experimental points, although indexes and sum symbols relative to the experimental data samples are omitted, for reason of clarity.

$$L = (dF/dY)^2 D_Y^2 + (dF/dt)^2 D_t^2 = D_Y^2 + (dF/dt)^2 D_t^2 \tag{6}$$

L is the weighting coefficient corresponding to an experimental point, D_Y and D_t representing the standard deviation or error associated to Y and t respectively.

The values of L (Eq. 6) for every sample measured and the sums described in equations (3), (4) and (5) are calculated and transferred into matrixes [C] and [V]. Matrix [C] is inverted and matrix [A] is obtained by:

$$[A] = [C]^{-1} * [V] \qquad (7)$$

If the quadratic sum of the A_i contained in [A] is greater than a given value, say for instance 0.000001, the iterations must proceed and the values of A_i are subtracted from the values of a_i , to give the new initial values and begin a new iteration.

When the criterion of convergence is achieved, the values a_i of the parameters are considered as being the best estimates, and the standard deviation D_{ai} for these values is given by:

$$D_{ai} = (C_{kk}^{-1})^{1/2} S^{1/2} (n-p)^{-1/2}$$

C_{kk} being the diagonal term of matrix $[C]^{-1}$ corresponding to parameter a_k ; S, the sum already defined; n, the number of experimental points and p, the number of unknown parameters. (n-p) represents the number of degrees of freedom.

As stated before, it is not possible to determine all the parameters for several sine functions at the same time, because of the practical impossibility to converge to the desired values. For this reason, and because of the orthogonality of sine functions of different periods, this method was developed in order to look for the three parameters relative to a unique sine function and to repeat adequately the process for the other unknown sine functions, through a sweep process beginning by line while(...) of function main() as shown in the list presented in Appendix B.

APPENDIX B

The following listing presents the program in C as it was used for the tests; A few lines of comments were added in order to explain how the program was built and works.

```
                        /*   TSIRA.C            */

#include <stdio.h>
#include <math.h>
#include <io.h>
#include <stdlib.h>
     int k,kmax,iter,itermax;
     double ts[4][500],a[3],A[3],sd[3],c[3][3],e[3][3],v[3],tmax;
     double F0,Fa0,Fa1,Fa2,L,arg,omega,domega,omegamax,delta,d,S,sdT;
     char dataname[40],outputname[40],s[20],u[20],ds[20],du[20],w[4];
     char number[5];
     const double f2pi=6.283185308;

main ()
{

                        /* Data input          */
     readfile();
                        /* Pulsation value sweep */
     omega=0;
     domega=f2pi/tmax;
     omegamax=(double)domega*kmax/2;
                        /* Screen & output file header     */
     writehead();
```

```
    while((omega+=domega)<omegamax)
    {
                        /* Initial guess for parameters    */
    a[0]=10.0;
    a[1]=omega;
    a[2]=0.5*f2pi;
                        /* Iterations              */
    iter=1;
    do    { recalc();
         solve();
         newvalue(); }
    while (iter++<itermax && delta>1e-6 && delta<1e9);
                        /* Control iteration number,
                           convergence and divergence */

                        /* Convergence achieved: write
                           one line in result file    */
    if (delta<1e-6) writefile();
    }
}                                       /* End on allowed pulsation
                           value exhaust          */

recalc()
                        /* Error function         */
              /* F =    Y     - a[0]*sin(a[1]*t       +a[2])    */
              /* F = ts[1][k] - a[0]*sin(a[1]*ts[0][k]+a[2])    */
    {
                        /* Initializing                */
    c[0][0]=c[1][1]=c[2][2]=c[0][1]=c[1][2]=c[0][2]=0.0;
    v[0]=v[1]=v[2]=S=0.0; k=-1;
                        /* Sums on all samples         */
    while (++k<=kmax)
    {
```

```
                    /* Derivatives / parameters   */
Fa0=-sin(arg=a[1]*ts[0][k]+a[2]);
Fa2=-a[0]*cos(arg);
Fa1=Fa2*ts[0][k];
F0=ts[1][k]+a[0]*Fa0;           /* Difference function      */

                    /* Without or with weighting  */
if(strncmp(w,"y",1)) L=1.0;
else L=ts[3][k]*ts[3][k]+a[1]*a[1]*Fa2*Fa2*ts[2][k]*ts[2][k];
/*   L= sigmaY * sigmaY + (sigmat*dF/dt) * (sigmat*dF/dt)   */

                    /* Filling matrix c[3][3]     */
c[0][0]+=Fa0*Fa0/L;
c[1][1]+=Fa1*Fa1/L;
c[2][2]+=Fa2*Fa2/L;
c[0][1]+=Fa0*Fa1/L;
c[1][2]+=Fa1*Fa2/L;
c[0][2]+=Fa0*Fa2/L;
                    /* Filling matrix v[3]        */
v[0]+=Fa0*F0/L;
v[1]+=Fa1*F0/L;
v[2]+=Fa2*F0/L;
S+=F0*F0/L; }
}
```

```
solve()
                        /* Any resolution may be used      */
     {
     d =c[0][0]*(c[1][1]*c[2][2]-c[1][2]*c[1][2]);
     d+=c[0][1]*(c[0][2]*c[1][2]-c[0][1]*c[2][2]);
     d+=c[0][2]*(c[0][1]*c[1][2]-c[0][2]*c[1][1]);

     e[0][0]=(c[1][1]*c[2][2]-c[1][2]*c[1][2])/d;
     e[1][1]=(c[0][0]*c[2][2]-c[0][2]*c[0][2])/d;
     e[2][2]=(c[0][0]*c[1][1]-c[0][1]*c[0][1])/d;
     e[0][1]=(c[0][2]*c[1][2]-c[0][1]*c[2][2])/d;
     e[0][2]=(c[0][1]*c[1][2]-c[0][2]*c[1][1])/d;
     e[1][2]=(c[0][1]*c[0][2]-c[0][0]*c[1][2])/d;

     A[0]=e[0][0]*v[0]+e[0][1]*v[1]+e[0][2]*v[2];
     A[1]=e[0][1]*v[0]+e[1][1]*v[1]+e[1][2]*v[2];
     A[2]=e[0][2]*v[0]+e[1][2]*v[1]+e[2][2]*v[2];
     }

newvalue()
     {                  /* Sum of squared corrections      */
     delta=A[0]*A[0]+A[1]*A[1]+A[2]*A[2];
     S=sqrt(S/(kmax-2));
                        /* Degree of freedom = kmax+1-3
                           = sample nr - parameter nr */

                        /* New value;standard deviation    */
     a[0]-=A[0]; sd[0]=S*sqrt(e[0][0]);
     a[1]-=A[1]; sd[1]=S*sqrt(e[1][1]);
     a[2]-=A[2]; sd[2]=S*sqrt(e[2][2]);
```

```
                        /* Only valid or allowed values */
    a[0]=fabs(a[0]);
    a[1]= fabs(a[1]-omega)<domega/2 ? a[1] : omega;
    a[2]-=f2pi*floor(a[2]/f2pi);

    printa(); printf("\r");
    }

readfile()
    {
       FILE *fp;

    title();
                        /* Data file name or default  */
    printf("\nEnter drive:filename.ext for data input: ");
    gets(dataname);
    if(!strcmp(dataname,""))
    {strcpy(dataname,"DATA.PRN");
    printf("\t\t\t\t\t%s\n",dataname);}
                        /* Test if reading file OK    */
    if((fp=fopen(dataname,"r"))==NULL)
    {printf("\ncannot open file\n"); exit(1);}

                        /* Ask if weighting desired   */
    weight();
                        /* Ask number max of iterations */
    iternr();
                        /* Read and show on screen    */
    k=0;
    while(fscanf(fp,"%s%s%s%s\n",s,ds,u,du)!=EOF)
    {printf("%4d  %8.3f %8.3G\t",k,ts[0][k]=atof(s),ts[2][k]=atof(ds));
    printf("%8.3f  %8.3G\n",ts[1][k]=atof(u),ts[3][k++]=atof(du));
    }
```

```
    printf("\nnumber of samples = %d\n\n",k);
    kmax=k-1;

    tmax=ts[0][kmax]-ts[0][0];

    fclose(fp);
    }

weight()
    {
    puts("\nWeighting data? (y/n) (default = y): "); gets(w);

                        /* [ENTER], y, Y  give  "y"   */
    if(!strcmp(w,"")||!strncmp(w,"Y",1)) strcpy(w,"y");
                        /* For no weighting: n or N   */
    if(strncmp(w,"y",1)&&strncmp(w,"N",1)&&strncmp(w,"n",1))
    weight();           /* if no 1st char OK, ask again    */
    else printf("\t\t\t\t\t%s\n\n",w);
    }

iternr()

    {
    printf("\nMaximum number of iteration per run? ");
    gets(number);
    if(!strcmp(number,"")) strcpy(number,"30");
    itermax=atoi(number);

    if(itermax<5 || itermax>999) itermax=30;
    }
```

```
writehead()
                        /* Screen & output file header     */
    {
       FILE *fp;
                        /* Result file name or default     */
    printf("\nEnter drive:filename.ext for output: ");
    gets(outputname);
    if(!strcmp(outputname,""))
    {strcpy(outputname,"FILE.ASC");
    printf("\t\t\t\t\t%s\n",outputname);}
                        /* Test if write file OK */
    if((fp=fopen(outputname,"a"))==NULL) printf("cannot open file\n");

    fprintf(fp,"\nData  : %-23s Results: %-23s\n",dataname,outputname);
    fprintf(fp,"Sample: %-23d omegamax= %-8.3f\n",kmax+1,omegamax);
    if(!strcmp(w,"y")) fprintf(fp,"Weight: yes");
    else fprintf(fp,"Weight: no ");
    fprintf(fp,"\t\t\tdomega = %-8.3f\n",domega);

    fprintf(fp,"Iterat: %-23d\n",itermax);

    fprintf(fp,"\n  PERIOD      ");
    fprintf(fp,"    AMPLITUDE      PULSATION       PHASE      ");
    fprintf(fp,"  residue    Nr of");

    fprintf(fp,"\n  T    +-sd(T)");
    fprintf(fp,"   a[0] +-sd[0]   a[1] +-sd[1]    a[2] +-sd[2]");
    fprintf(fp,"   delta     iter\n");
    fclose(fp);
```

```
    title();
    printf("Data  : %-23s Results: %-23s\n",dataname,outputname);
    printf("Sample: %-23d omegamax= %-8.3f\n",kmax+1,omegamax);
    if(!strcmp(w,"y")) printf("Weight: yes");
    else printf("Weight: no ");
    printf("\t\t\tdomega = %-8.3f\n",domega);

    printf("Iterat: %-23d\n",itermax);

    printf("\n  PERIOD      ");
    printf("    AMPLITUDE      PULSATION       PHASE      ");
    printf("  residue    Nr of");
    printf("\n  T    +-sd(T)");
    printf("   a[0] +-sd[0]   a[1] +-sd[1]    a[2] +-sd[2]");
    printf("   delta     iter\n");
    }

writefile()
                        /* File and screen:
                           Write one line with results     */
    {
    FILE *fp;
    if((fp=fopen(outputname,"a"))==NULL)
    printf("cannot open file\n");

    fprintf(fp,"%6.3f %6.3f   ",f2pi/a[1],f2pi*sd[1]/(a[1]*a[1]));
    fprintf(fp,"%6.2f %6.2f   ",a[0],sd[0]);
    fprintf(fp,"%6.3f %6.3f  ",a[1],sd[1]);
    fprintf(fp,"%6.2f %6.2f   ",a[2],sd[2]);
    fprintf(fp,"%6.2E %4d\n",delta,iter);
    fclose(fp);
    printa(); printf("\n");
    }
```

```
printa()
                        /* Show on screen during iteration
                           and after convergence */
     {
     sdT=f2pi*sd[1]/(a[1]*a[1]);

     printf("%6.3f %6.3f   ",f2pi/a[1],sdT<1E+3? sdT : 999.);
     printf("%6.2f %6.2f   ",a[0],sd[0]<1E3  ? sd[0] : 999.);
     printf("%6.3f %6.3f  ",a[1],sd[1]<1E4  ? sd[1] : 999.);
     printf("%6.2f %6.2f   ",a[2],sd[2]<f2pi ? sd[2] : f2pi);
     printf("%6.2E %4d  ",delta,iter);
     }

title()
     {
     printf("\n\n\n\tTIME SERIES ITERATIVE REGRESSION ANALYSIS");
     printf("\n\n\t\tDN SOFTWARE\n\n");
     }
```

PRESIDÊNCIA DA REPÚBLICA
SECRETARIA DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

AUTORIZAÇÃO PARA PUBLICAÇÃO

TÍTULO

Search for Periodicities in Geophysical Time Series by Iterative Regression Analysis in C

AUTOR

Daniel Jean Roger Nordemann

TRADUTOR

EDITOR

ORIGEM	PROJETO	SÉRIE	Nº DE PÁGINAS	Nº DE FOTOS	Nº DE MAPAS
CEA/DGE	QUIATM		21		

TIPO

☐ RPQ ☒ PRE ☐ NTC ☐ PRP ☐ MAN ☐ PUD ☐ TAE ☐ ___

DIVULGAÇÃO

☒ EXTERNA ☐ INTERNA ☐ RESERVADA ☐ LISTA DE DISTRIBUIÇÃO ANEXA

PERIÓDICO/EVENTO

XIIICNMAC - Congresso Nacional de Matemática Aplicada e Computacional

CONVÊNIO

AUTORIZAÇÃO PRELIMINAR

___/___/___ _____ ASSINATURA

REVISÃO TÉCNICA

☐ SOLICITADA ☐ DISPENSADA _____ ASSINATURA

RECEBIDA ___/___/___ DEVOLVIDA ___/___/___ _____ ASSINATURA DO REVISOR

REVISÃO DE LINGUAGEM

☐ SOLICITADA ☐ DISPENSADA _____ ASSINATURA

Nº ___

RECEBIDA ___/___/___ DEVOLVIDA ___/___/___ _____ ASSINATURA DO REVISOR

PROCESSAMENTO/DATILOGRAFIA

RECEBIDA 01/10/90 DEVOLVIDA 20/10/90 Maria de Lourdes Tavares Lemos ASSINATURA

REVISÃO TIPOGRÁFICA

RECEBIDA ___/___/___ DEVOLVIDA ___/___/___ _____ ASSINATURA

AUTORIZAÇÃO FINAL

11/08/92 José Humberto Andrade Sobral ASSINATURA

PALAVRAS-CHAVE

Periodicities - Geophysics - Iterative Regression, C Language

550.3

INPE - 106.1