J. Ind. Geophys. Union (January 2007)
Vol.11, No.1, pp.59-72

A user friendly environment for the Spectral Analysis of Geomagnetic Data

Y.N.T.Seshagiri Rao
National Geophysical Research Institute, Hyderabad - 500 007

ABSTRACT

This paper describes a user-friendly Graphical User Interface also called Windows (You click and get something) environment or platform and computational paradigms developed using Java programming language for the spectral analysis of Geomagnetic Data. For the spectral analysis Digital Signal Processing methods such as Fast Fourier Transform (FFT), Cosine and Sine Transforms are used. The FFT transforms a function from the time domain to the frequency domain and is an extremely important and widely used method of extracting useful information from sampled signals. In the design of the system, the methodology used is Unified Modeling Language (UML), an advanced flowchart that uses Use Case Diagrams (UCD). The background and motivation is the automation of the entire process of obtaining spectra analysis.

INTRODUCTION

The system developed aims at solving geomagnetic data spectrum analysis problems in the windows environment. The observatory records the time variations of the earth's magnetic field (Jacobs 1987 and Parkinson 1983), which give us valuable information on the electrical and magnetic state of the upper atmosphere, crust and deep interior of earth. The users require frequency values of the magnetic signal for their work. By using FFT the users get the values of the frequency domain about the availability of the given signal they needed and the details of the wave whether it is Quiet Day, Storm Day, and Solar Flare. Using these frequency values, they calculate the impedance values that are useful in solid earth research. In this system, we use Fast Fourier Transform to convert time domain data into frequency domain under Java Platform in order to increase accuracy, efficiency and to lessen human errors as well as the time of signal processing.

It is interesting and important to develop an automatic system for doing the spectral analysis. Studies of long periodicities in the geomagnetic field can be approached from either a time or frequency domain point of view. This paper presents a power spectrum analysis of data. The advantage of this type of analysis is that the frequency components of the entire spectrum can be investigated. This data is transformed into frequency domain using the FFT program. The naïve approach for using FFT is hard and time consuming. But using the developed system, you can get things by pressing a button.

In previous approaches FFT was written in FORTRAN and FORTRAN was not good enough to develop the GUI system. The Java is a general purpose and universal programming language that came into existence lately. It is used for developing both GUI and computations.

RELATED WORK

After thorough search on the Internet and literature, could not find an integrated Java environment for the spectral analysis of geomagnetic data or any other Geophysical data. One can find many FFT algorithms coded in FORTRAN and C.

SYSTEM DEVELOPMENT

The system mainly consists of two modules: The GUI and computational paradigms. The steps involved in the computational paradigm are the development of algorithm for FFT, Sine and Cosine transforms and coding the algorithms into java language.

FFT ALGORITHM

The Fast Fourier Transform (Cooley & Tukey 1965, Brigham 1988, and Oppenheim & Schafer 1989) is an optimized computational algorithm to implement the Discreet Fourier Transform to an array of 2 ^ N samples. It allows determining the frequency of a discreet signal, representing the signal in the frequency domain.

The FFT is a mathematical procedure, which can be thought of as transforming a function from time domain to the frequency domain. In digital signal processing, the spectrum of a signal refers to the way energy in the signal is distributed over its various frequency components.

DFT enables to analyze, manipulate and synthesis signals in ways not possible with continuous (analog) Signal Processing. A Discrete signal sequence is a set of values obtained by periodic sampling of a continuous signal in the time domain. In the field of continuous signal processing,

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-j2\Pi ft} dt$$

x (t)= some continuous time domain signal is used to transform an expression of continuous time domain function x(t) into a continuous frequency domain function X(f). The evaluation of X (f) expression helps us to determine the frequency content of any practical signal of interest.

Discrete (DFT) is given by,

$$X(m) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\Pi nm/N} dt$$

X(n) is a discrete sequence of time domain sampled values of continuous variable.

X (m)= the m^{th} DFT output component i.e., X(0)……..X(N-1).

m= Index of DFT output in frequency domain, m=0,…….. N-1.

N= No. of samples of input sequence and the no. of frequency points in DFT output.

The value N is a important parameter because it determines how many input samples are needed, the resolution of frequency domain results and the amount of processing time necessary to calculate the N- point DFT.

The exact frequencies of the different sinusoid depends on both the sampling rate (fs) at which the original signal was sampled and the number of samples (N).

When the DFT is applied to a discrete signal, the result is a set of sine and cosine coefficients. When sine and cosine waves of appropriate frequencies are multiplied by these coefficients and then added together, the original signal waveform is exactly reconstructed. The sine and cosine waves are the frequency components of the original signal, in the sense that the signal can be built up from these components. The coefficients determined by the DFT represent the amplitudes of each of these components. The procedure by which the sine and cosine coefficients are calculated is straightforward in principle, although in practice it requires a great deal of computation. To determine each individual coefficient, every one of the sampled values of the signal must be multiplied by the corresponding sampled value of a sine or cosine wave of the appropriate frequency. These products must then be added together, and the result then divided by the number of samples involved giving the value of the coefficient.

If the signal consists of a number of samples *N*, the DFT requires the calculation of *N* sine and *N* cosine coefficients. For each coefficient to be determined, *N* products of samples of the signal and the appropriate sine or cosine wave must be evaluated and summed. The total number of steps in the computation of the DFT is thus N^2, each step requiring the evaluation of a sine or cosine function together with a multiplication (and this does not include the calculation of the *N* products in order to find each coefficient).

Fast Sine and Cosine Transforms

The Fourier transforms of functions can be used to solve differential equations. The most common boundary conditions for the solutions are:

1. They have the values zero at the boundaries, or
2. Their derivatives are zero at the boundaries.

In these instances two more transforms arise naturally, the Sine and Cosine transforms given by,

$$F_k = \sum_{j=1}^{N-1} f_j \sin(\Pi jk/N) \qquad \text{Sine transform}$$

$$F_k = \sum_{j=0}^{N-1} f_j \cos(\Pi jk/N) \qquad \text{Cosine transform}$$

where f_j j=0….N-1 is the data array.

At first blush these appear to be simple the imaginary and real parts respectively of the discrete Fourier Transform. The argument of the sin and cosine differ by a factor of two from the value that would make that so. The sin transform uses sines only as a complete set of functions in the interval from 0 to 2∏, and the cosine transform uses cosines only.

In the case of sine transform we extend the data to twice their lengths in such away as to make them

an odd function about $j=N$, with $f_n = 0$,

$F_{2N-j} = -f_j \quad j=0,\ldots\ldots\ldots N-1$

$$F_k = \sum_{j=0}^{2N-1} f_j \, e^{2\Pi ijk/(N/2)}$$

the half of this sum from $j=N$ to $j=2N-1$ can be rewritten with the substitution $j=2N-j$.

$$\sum_{j=N}^{2N-1} f_j \, e^{2\Pi ijk/2N} = \sum_{j'=1}^{N} f_{2N} - j' e^{2\Pi i(2N-j')k/2N}$$

$$= -\sum_{j'=0}^{N-1} f_{j'} \, e^{-2ij'k/(2N)}$$

This array is of the same dimension as the original. Notice that the first term is symmetric about $j=N/2$ and the second is anti-symmetric. Consequently, when real FFT is applied to y_j, the result has real parts R_k and imaginary parts I_k given by,

$$R_k = \sum_{j=0}^{N-1} y_j \cos(2"jk/N)$$

$$= \sum_{j=1}^{N-1} (f_j + f_{N-j}) \sin(j\Pi/N) \cos(2\Pi jk/N)$$

$$= \sum_{j=0}^{N-1} 2f_j \sin(j\Pi/N) \cos(2\Pi jk/N)$$

$$= \sum_{j=1}^{N-1} f_j \left\{ \sin \frac{(2k+1j\Pi/)}{N} - \text{sing} \frac{(2k-1j\Pi)}{N} \right\}$$

$$= F_{2k+1} - F_{2k-1}$$

$$I_k = \sum_{j=0}^{N-1} y_j \sin(2\Pi jk/N)$$

$$= \sum_{j=0}^{N-1} (f_j - f_{N-j}) \frac{1}{2} \sin(2\Pi jk/N)$$

$$= \sum_{j=0}^{N-1} f_j \sin(2\Pi jk/N)$$

$$= F_{2k}$$

Therefore F_k can be determined as follows:

$F_{2k} = I_k \quad F_{2k+1} = F_{2k-1} + R_k \qquad k=0,\ldots\ldots\ldots, (N/2-1)$

The even terms of F_k are thus determined very directly. The odd terms require a recursion, the starting point of which is,

$$F_1 = \sum_{j=0}^{N-1} f_j \sin(j\Pi/N)$$

The Sine transform, curiously, is its own inverse. If you apply it twice, you get the original data, but multiplied by a factor of N/2.

The Cosine transform is slightly more difficult, but the idea is the same. Now, the auxiliary function is,

$$y_0 = f_0, \qquad y_j = \frac{1}{2}(f_j + f_{N-j}) - \sin(j\Pi/N)(f_j - f_{N-j})$$

and the same analysis leads to

$F_{2k} = R_k \quad F_{2k+1} = F_{2k-1} + I_k \quad k=0,\ldots\ldots,(N/2-1)$

The starting value for the recursion in this case is

$$F_l = \sum_{j=0}^{N-1} f_j \cos(j\Pi/N)$$

This sum does not appear naturally among the R_k and I_k, and so we accumulate it during the generation of the array y_i.

An additional complication is that the cosine transform is not its own inverse. To derive the inverse of an array F_k, we first compute an array f_l as if the transform were its own inverse,

$$\tilde{f}_l = \sum_{j=0}^{N-1} F_k \cos(\Pi kl/N)$$

One easily verifies the relations between f_l and the desired inverse f_l,

$$\tilde{f}_0 = Nf_0 + \sum_{j \text{ odd}} f_j$$

$$\tilde{f}_0 = \frac{N}{2} f_l + \sum_{j \text{ even}} f_j \qquad \text{for l odd}$$

$$\tilde{f}_0 = \frac{N}{2} f_l + \sum_{j \text{ odd}} f_j \qquad \text{for l even , } 1 \neq 0$$

The unknown sums on the right of these equations are determined as follows:

$$C_1 = \sum_{j \text{ even}} \tilde{f}_j = \frac{N}{2} \left[f_0 + \sum_{j=0}^{N-1} f_j \right]$$

$$C_2 = \sum_{l \text{ odd}} \tilde{f}_j = \frac{N}{2} \sum_{j=0}^{N-1} f_j$$

follows that

$$\frac{N}{2} f_0 = C_1 - C_2$$

so,

$$\sum_{j\ odd} f_j = \tilde{f}_0 - 2(C_1 - C_2) \qquad \sum_{j\ even} f_j = \frac{2}{N} C_2 - \sum_{j\ odd} f_j$$

Knowing these sums, the desired f_j are now easily recovered from equation.

The FFT is calculated in two parts (Press et al., 1992). The first one transforms the original data array into a bit-reverse order array by applying the bit-reversal method. The second part processes the FFT in N*log2 (N) operations (application of the Danielson-Lanzcos algorithm). The algorithm starts with an array of complex data. This could be an array of floats in witch the data [even index] is the real part and the data [odd index] is the complex part. This algorithm has a complexity of O (N*log2 (N)). Actually, the complexity of the algorithm is a little higher because the data needs to be prepared by bit-reversal. The original array must be transformed in order to perform the Danielson-Lanzcos algorithm. For example, the complex [*index*] will swap places with the complex [*bit-reverse-order-index*]. If the *index* (in binary) is 0b00001, the *bit-reverse-order-index* will be 0b10000

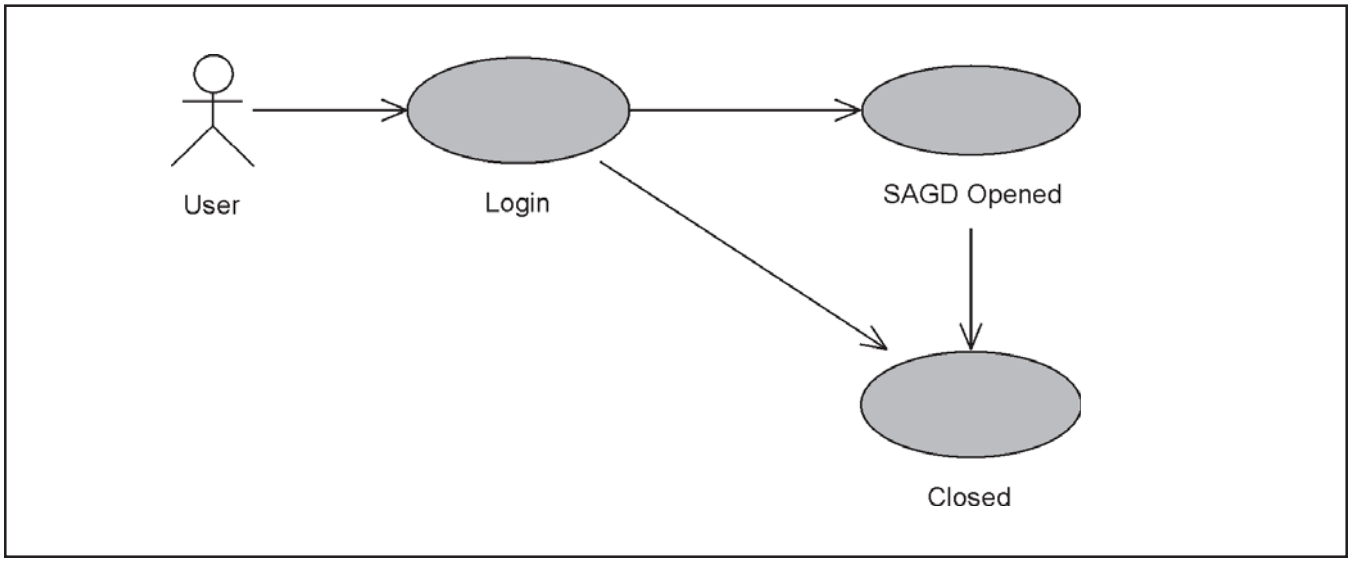
System Design

Computer programming is easier when you use a flowchart. A flowchart is a step-by-step procedure to solve a problem. To write programs in Java, Unified Modeling Language (UML) that uses Use Case Diagrams (UCD) is an efficient way of representing the flow of programming and is an advanced flow chart.

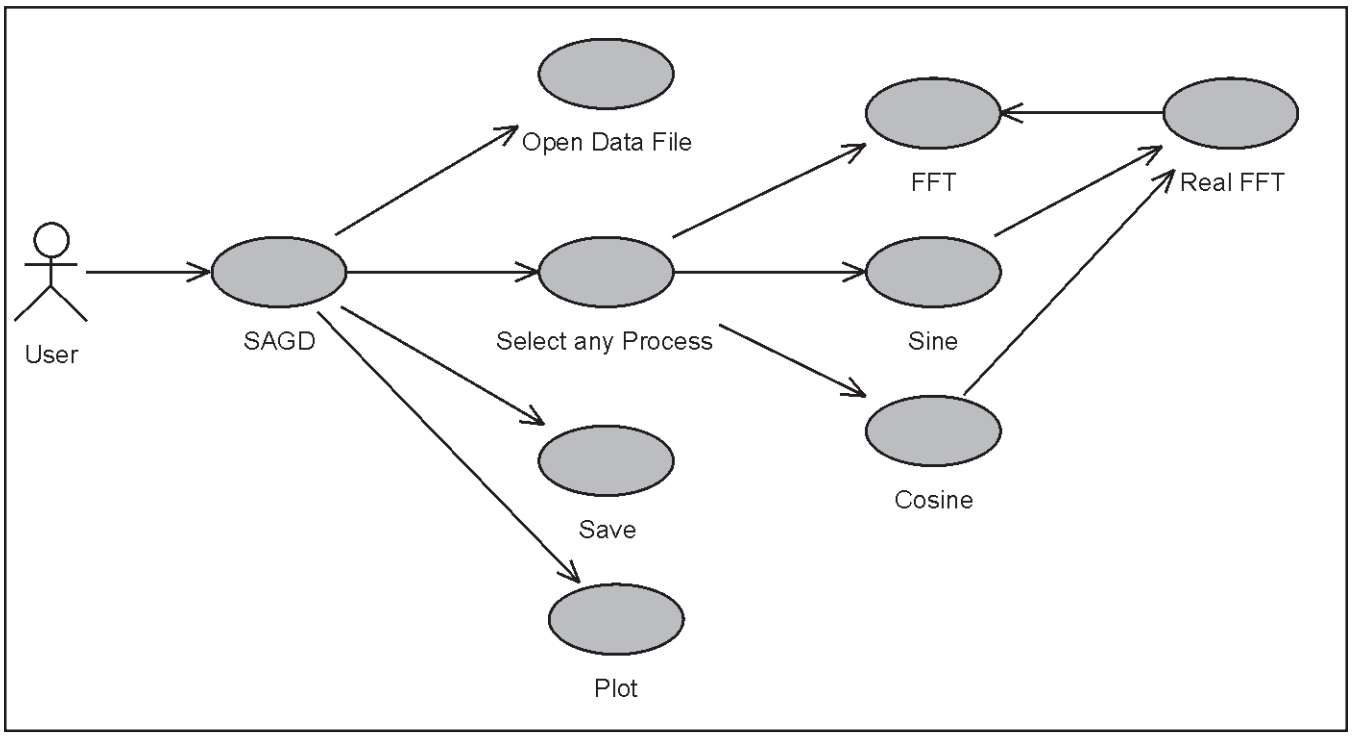
This section presents the design approach for the environment. Unified Modeling Language (UML) (Booch, Rumbaug & Jacobson 2005) is used in the design of the system. UML is a standard language for specifying, visualizing, constructing and documenting the artifacts of software systems and modeling of large and complex systems. The UML uses mostly graphical notations to express the design of software projects. UML is best preferred for Object oriented programming (OOP) and the system would be implemented in Java, an Object Oriented Programming language.

UML uses Use Case Diagrams (UCD). Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the analysis phase of software development to articulate the high-level requirements of the system.

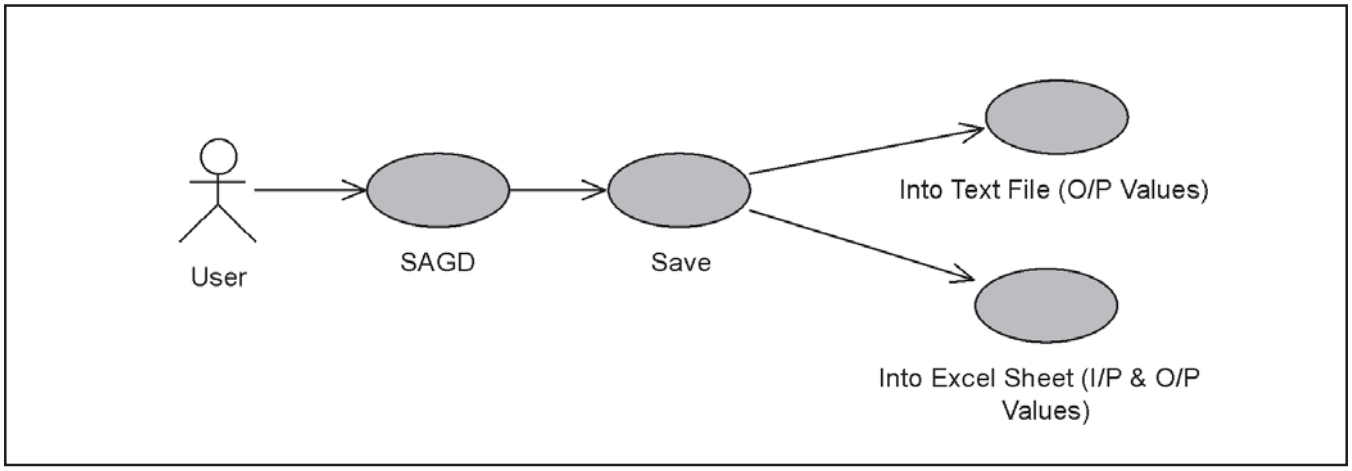
Use Case Diagram for Login



Use Case Diagram for Process

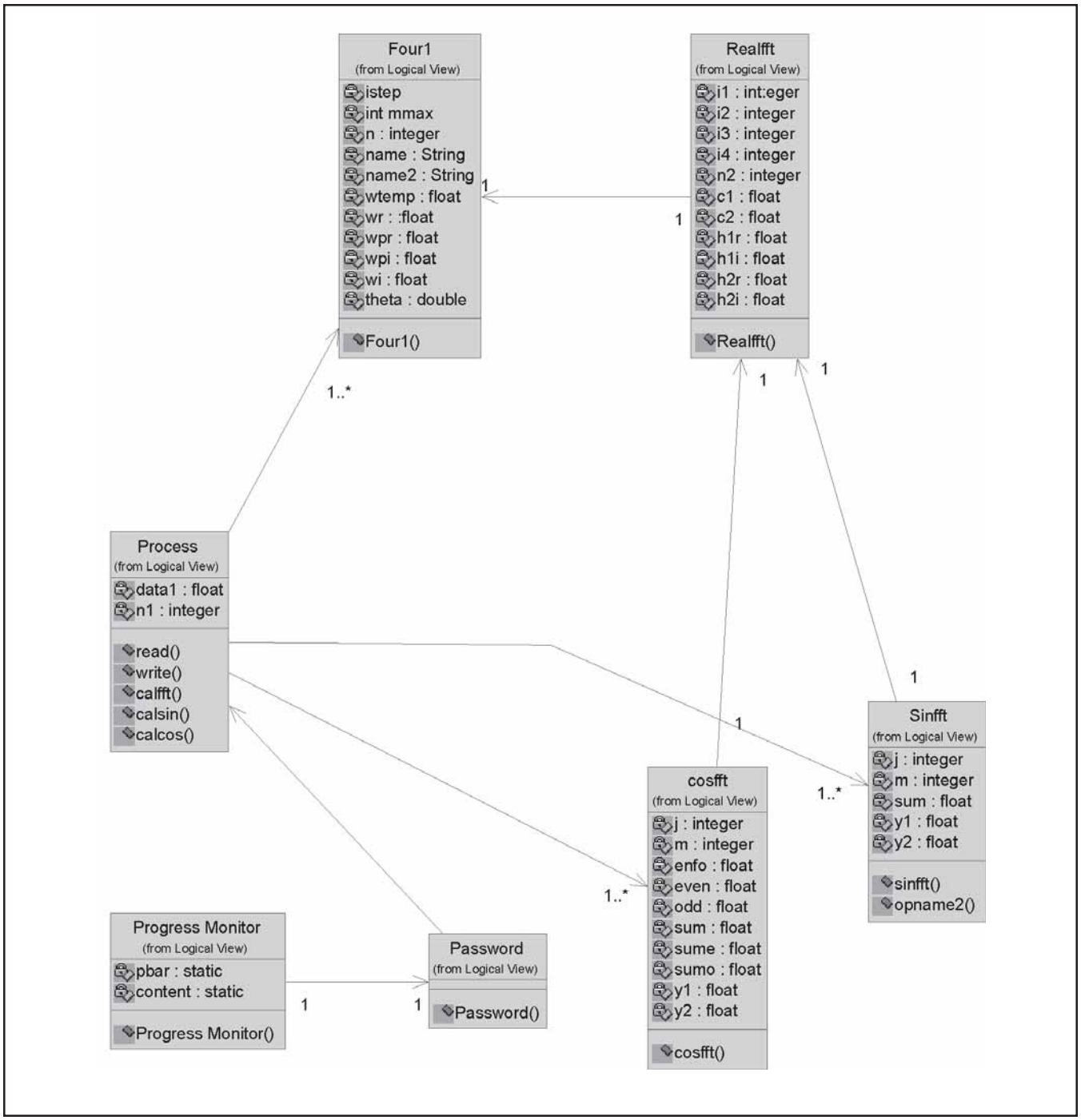


Use Case Diagram for Save



CLASS DIAGRAM:

The Class Diagram is used to refine the Use Case Diagram and define a detailed design of the system. The Class Diagram classifies the actors defined in the Use Case Diagram into a set of interrelated classes. Each Class in the Class Diagram may be capable of providing certain functionalities. These functionalities provided by the Class are termed “Methods” of the Class. Apart from this, each Class may have certain “Attributes” that uniquely identify the Class. A class defines the abstract characteristics of a thing, including the thing’s characteristics (its attributes or properties) and the things it can do (its behaviors or methods) or features).



IMPLEMENTATION AND PERFORMANCE

In the implementation stage of the project Java programming (Roberts, Heller & Ernest 2001 and Deitel & Deitel 2005) is used. Java a universal programming language and has an extensive library of routines. It also supports network computing. Java is intended for writing programs that must be readable in a variety ways. Java puts a lot of emphasis on early checking for possible problems, later dynamic checking, and eliminating situations that are error prone.

The Java compiler detects many problems that in other languages would only show up at runtime. The Java interpreters can execute Java byte codes directly on any machine to which the interpreter has been ported. Java contains classes core to the language contributing to the ease of programming.

With Java you can build Graphical User Interface (windows) for user-friendly operations and computational paradigms. The following sample data is used to test the system for obtaining the spectrum and for testing the data.

Sample Data:

The data collected is the H component from the "LA-COUR VARIOMETER " generated magnetic signals (D, H, Z). The analog charts are also shown in the figures.

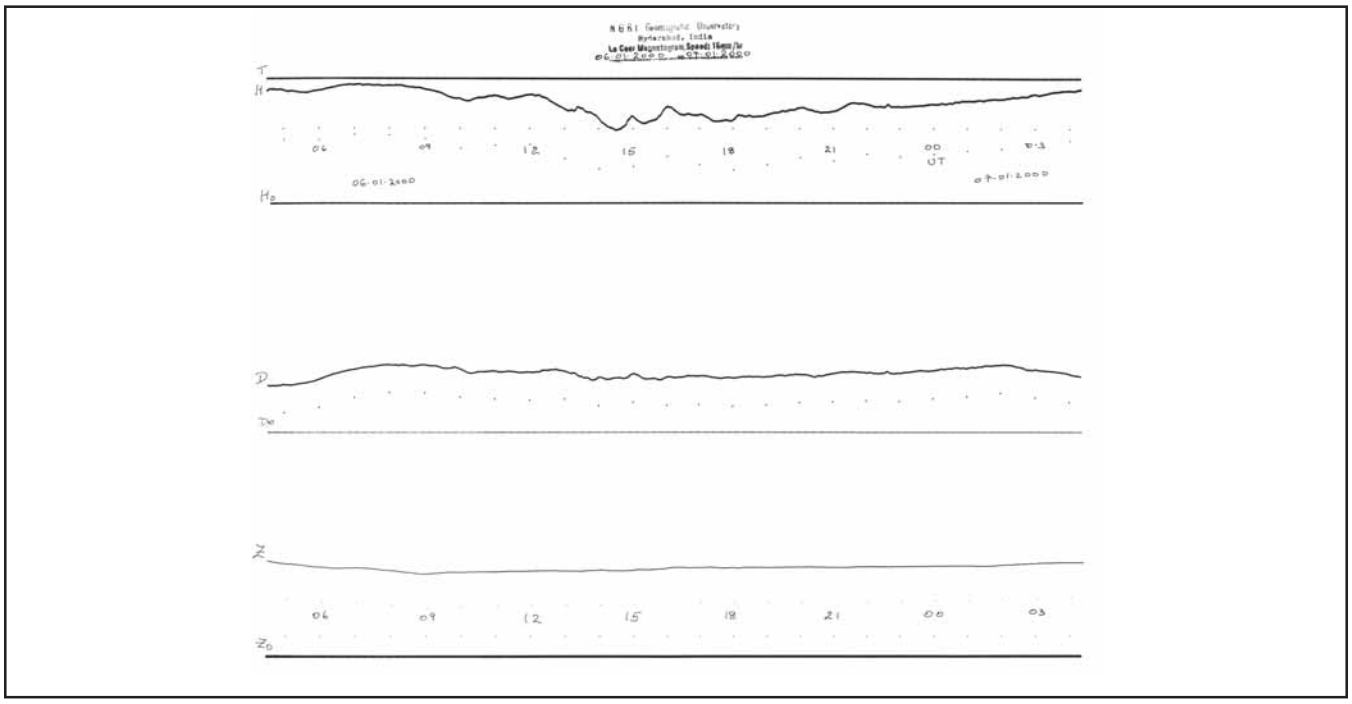
Input Data

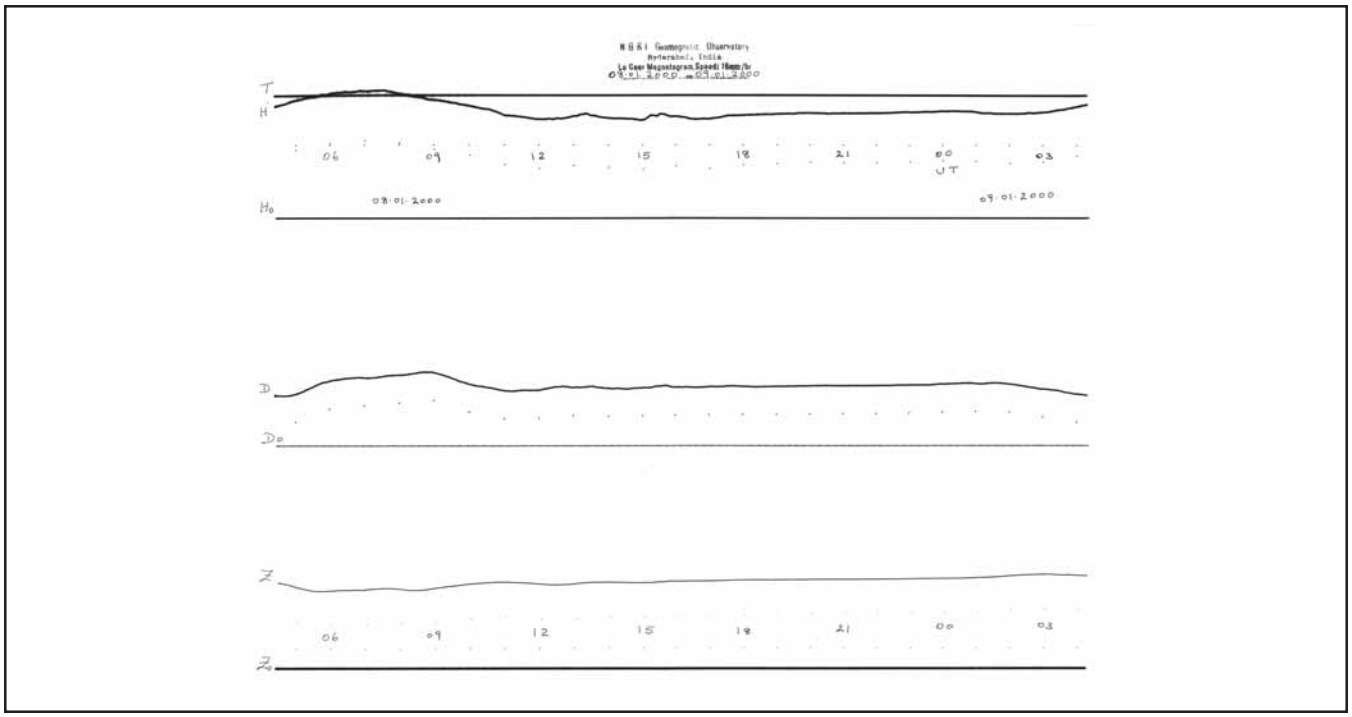
48.0	47.8	47.6	47.5	47.6	47.5	47.4	47.4	47.3	47.9	48.3	48.5
48.3	48.0	47.8	47.6	47.6	47.5	47.4	47.2	47.1	47.2	47.4	47.1
47.1	47.1	47.3	47.3	47.4	47.4						

Output Data

Input	FFT output	Sin Transform	Cosine Transform
1.35568E-19	1.64951E-33	1.64951E-33	1.65E-33
2.16927E-33	0.001730573	0	-0.0201
4.13902E-11	0.001858399	5.20663E-12	0
1.706E-07	-0.001724749	-0.0019235	-1.3E-05
5.11125E-07	0.001515063	9.84258E-05	4.88E-07
1.03833E-05	-0.000608912	0.000269517	0.000197
2.16927E-33	-0.001424011	0.00028478	7.04E-06
4.07081E-11	0.001128334	0.00017689	0.000707
1.71531E-07	0.000301156	0.000506755	2.51E-05
4.99145E-10	-0.000887552	0.004431823	-0.00038
1.706E-07	0.000731163	0.000354223	-2.3E-05
2.0445E-06	-0.000485802	0.001864041	-0.00012
1.03833E-05	-0.00128558	0.000192704	-3.1E-05
1.16216E-19	0.001402252	0.001773727	0.000425
1.03829E-05	-0.000769803	6.76446E-05	-4.6E-05
2.16927E-33	0.001280762	0.000544768	0.000163
4.02533E-11	0.00100687	0.000212471	-1.7E-05
1.706E-07	-0.000433692	0.000543091	-0.00018
3.19453E-08	0.001305164	0.000284058	-6E-05
1.03833E-05	-0.000776239	0.001141789	0.000505
4.99145E-10	-0.00031906	0.000154395	-3.6E-05
2.59601E-06	0.000616375	0.001580485	-8.1E-05
2.16922E-33	-0.000280704	-6.2894E-05	-7.8E-05
4.04807E-11	5.0564E-05	0.000632272	-0.00018
1.69669E-07	0.000677925	-9.6299E-05	-0.00011
3.2712E-05	-0.000335178	0.000547115	0.000254
6.48972E-07	-0.000218606	-8.4492E-05	-0.00016
2.16912E-33	0.000541899	0.000975354	0.000303
4.16176E-11	-8.07978E-05	-0.00014241	-5.9E-05
4.25874E-31	0.000134924	0.000297113	1.75E-05

Analog Data

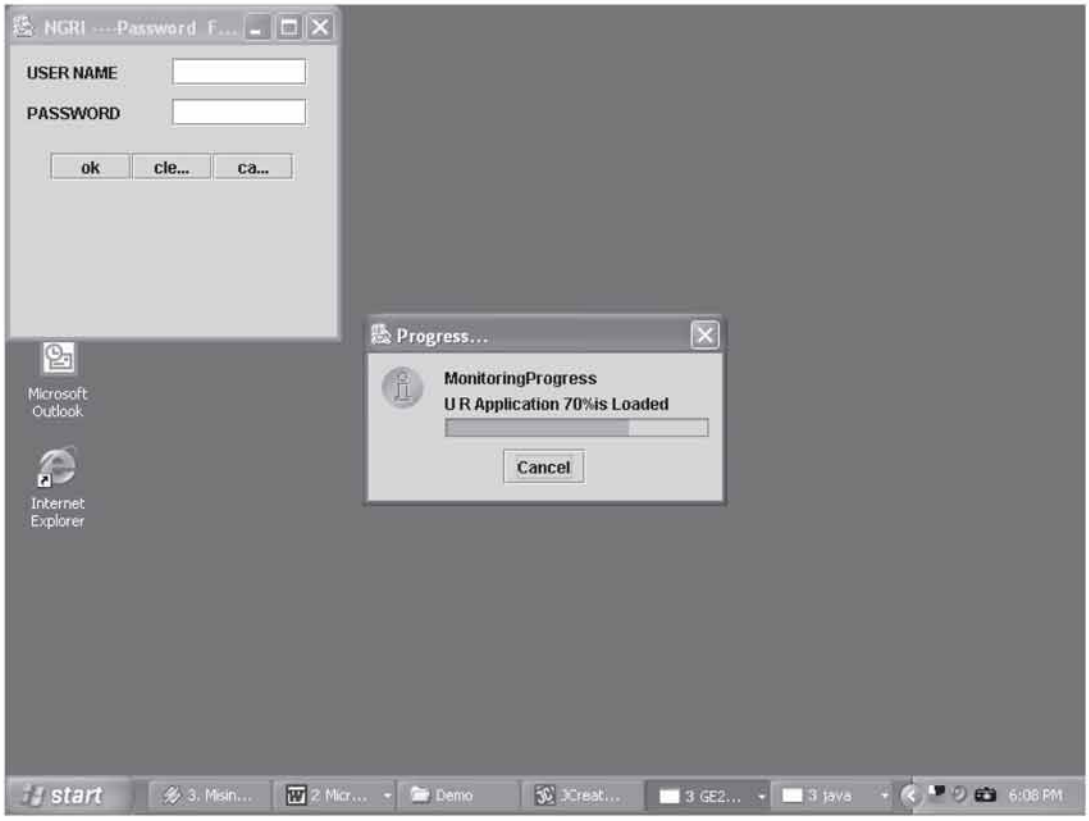


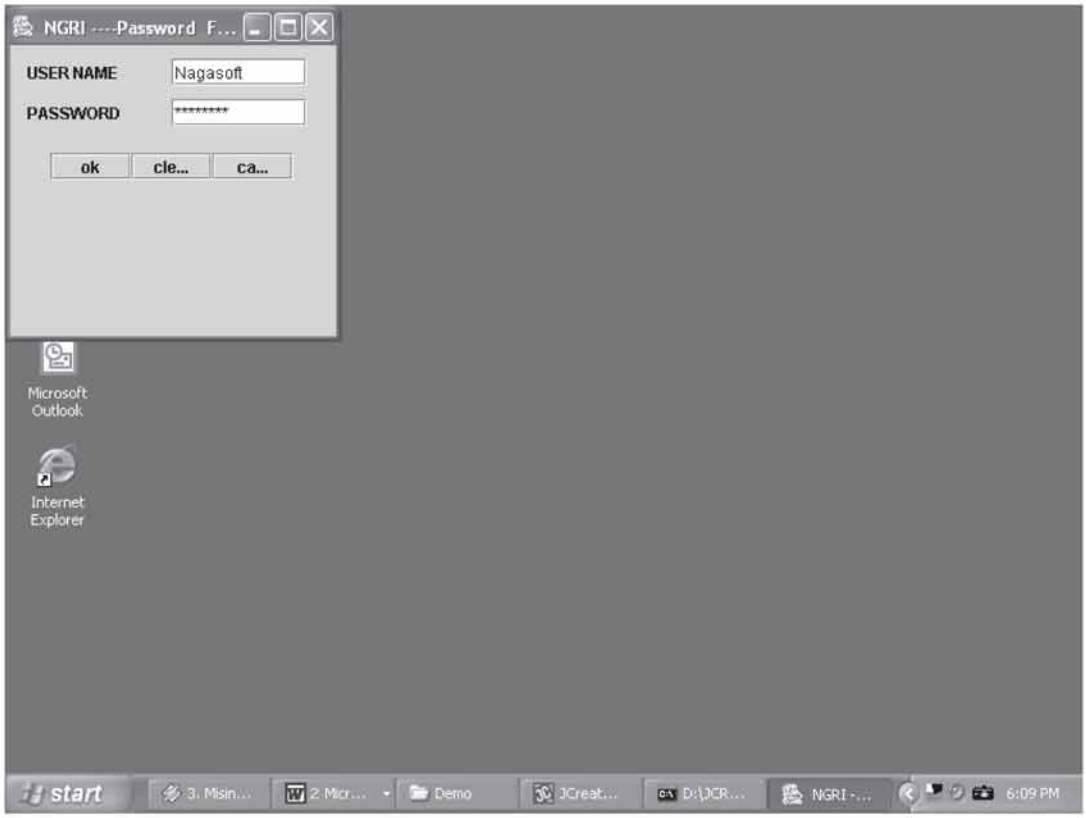


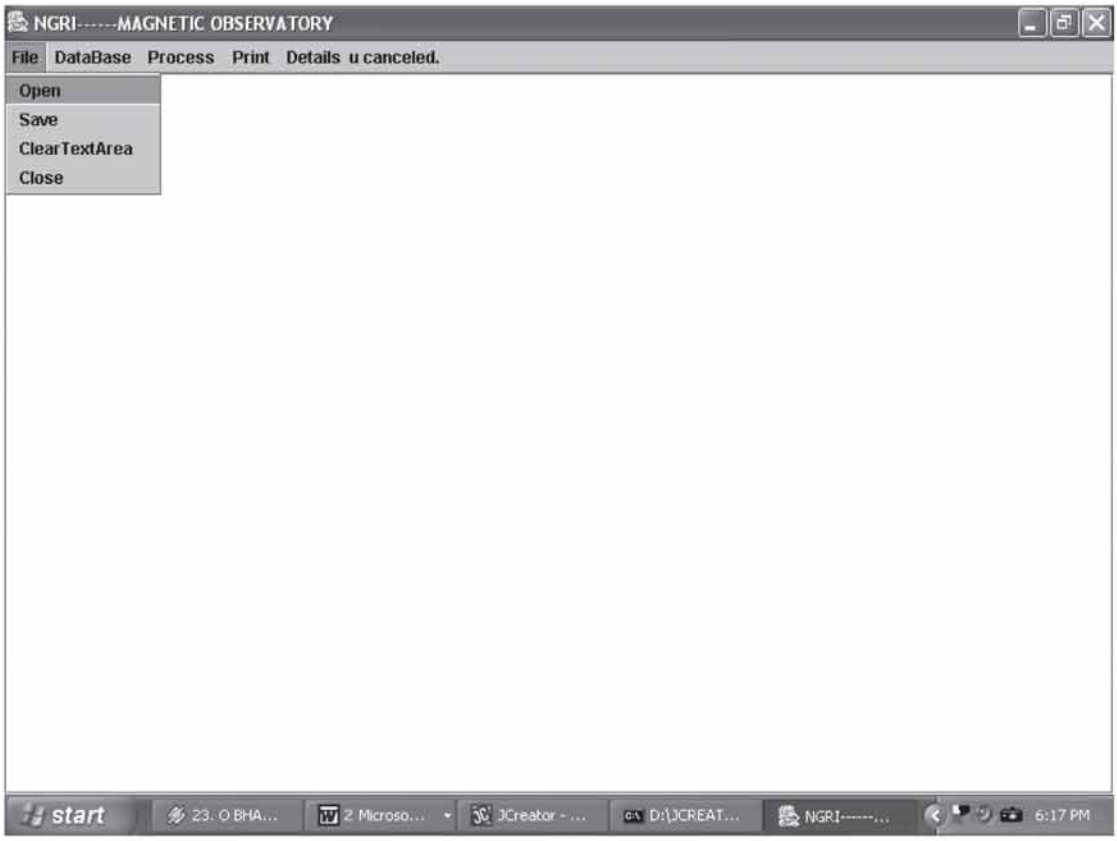
FUTURE WORK

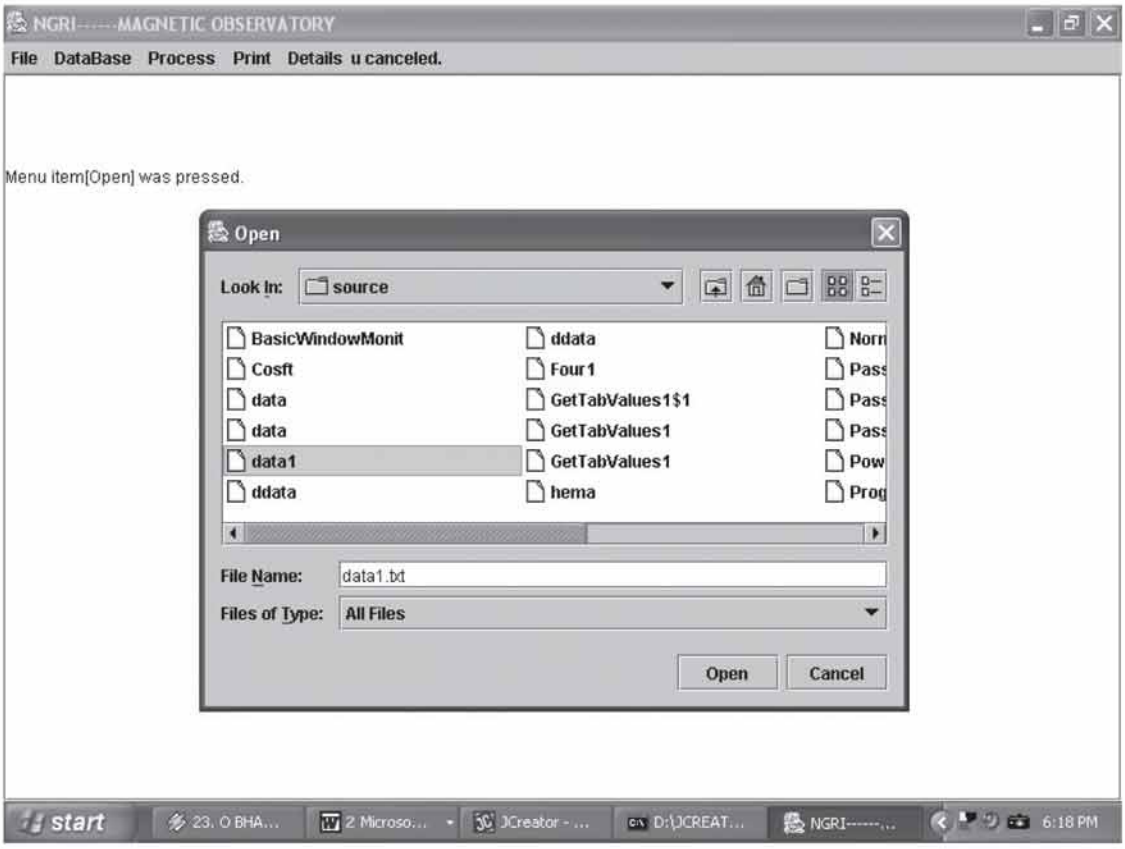
With slight modifications, the integrated environment can be extended to other geophysical data as seismic data, MT data, gravity data and so on depending on the nature of data. A comprehensive geophysical data analysis system can be aimed at to include and switch over from one kind of data to another.

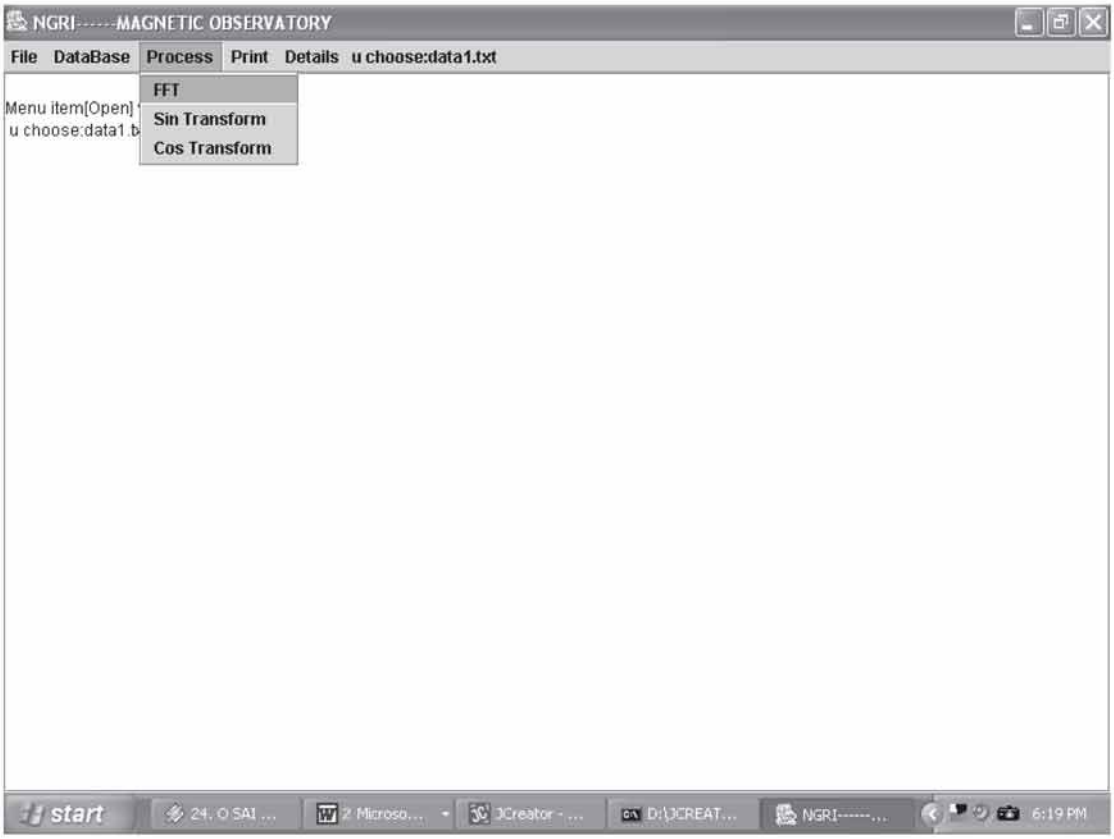
SCREENS for interfacing, inputting the data and obtaining the results.

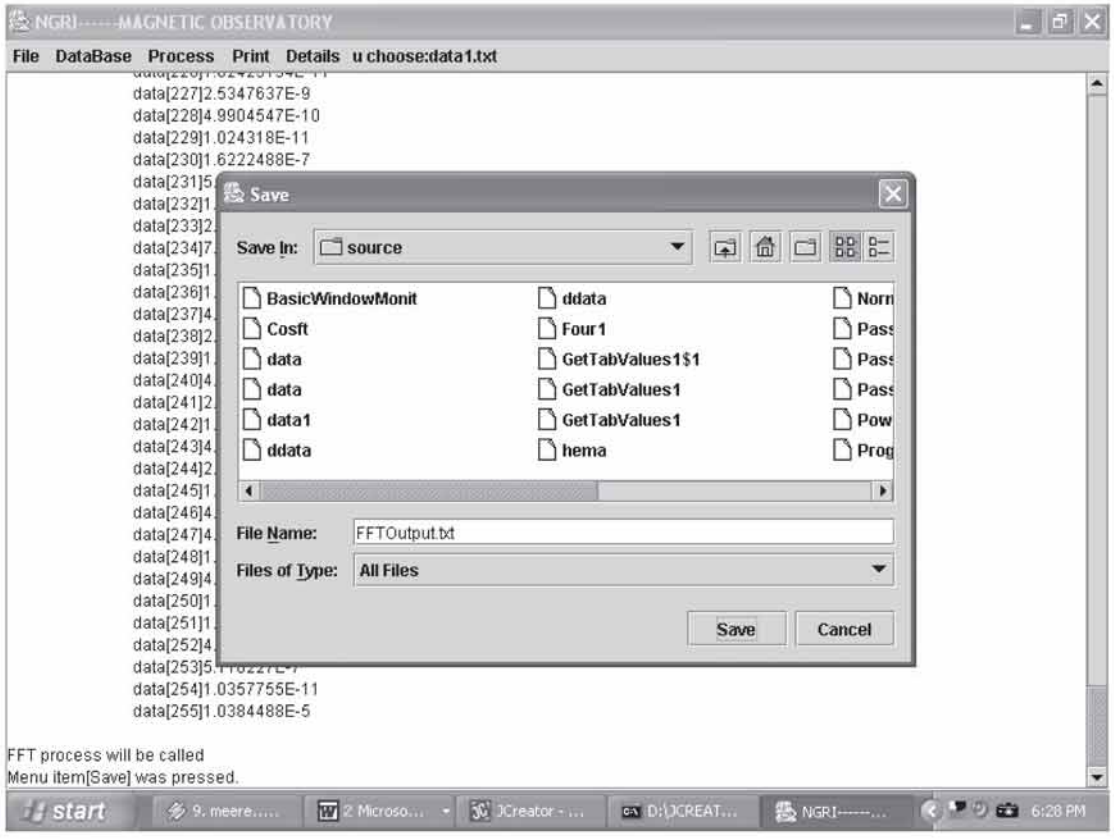


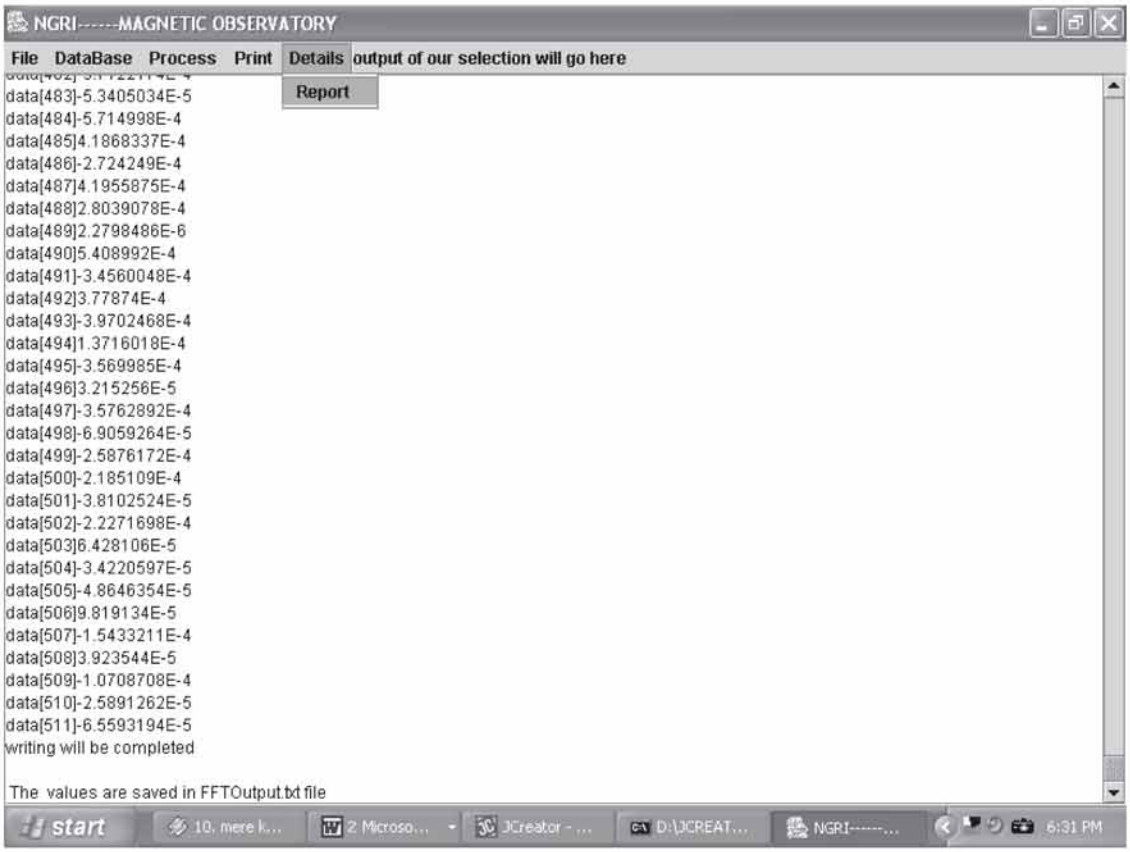


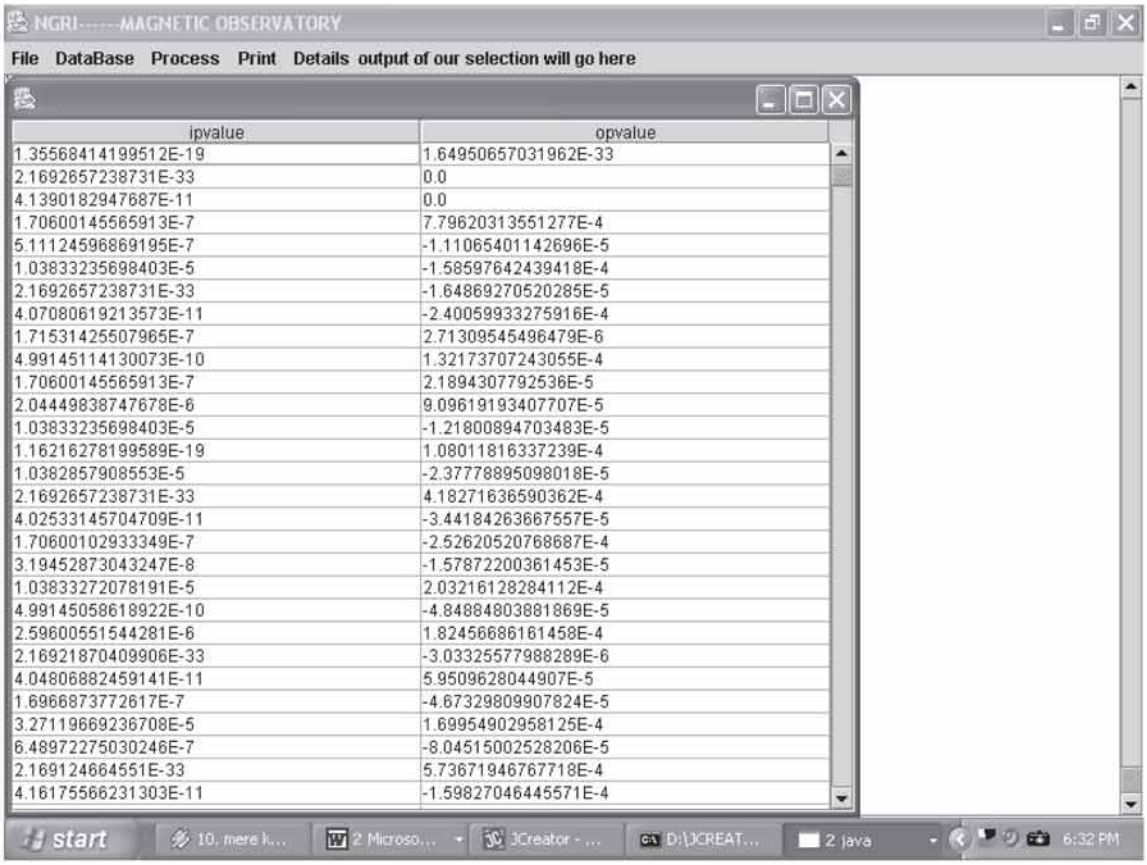

NGRI------MAGNETIC OBSERVATORY
File DataBase Process Print Details u canceled.
Open
Save
ClearTextArea
Close
start
23. O BHA...
2 Microso...
JCreator - ...
D:\JCREAT...
NGRI------...
6:17 PM

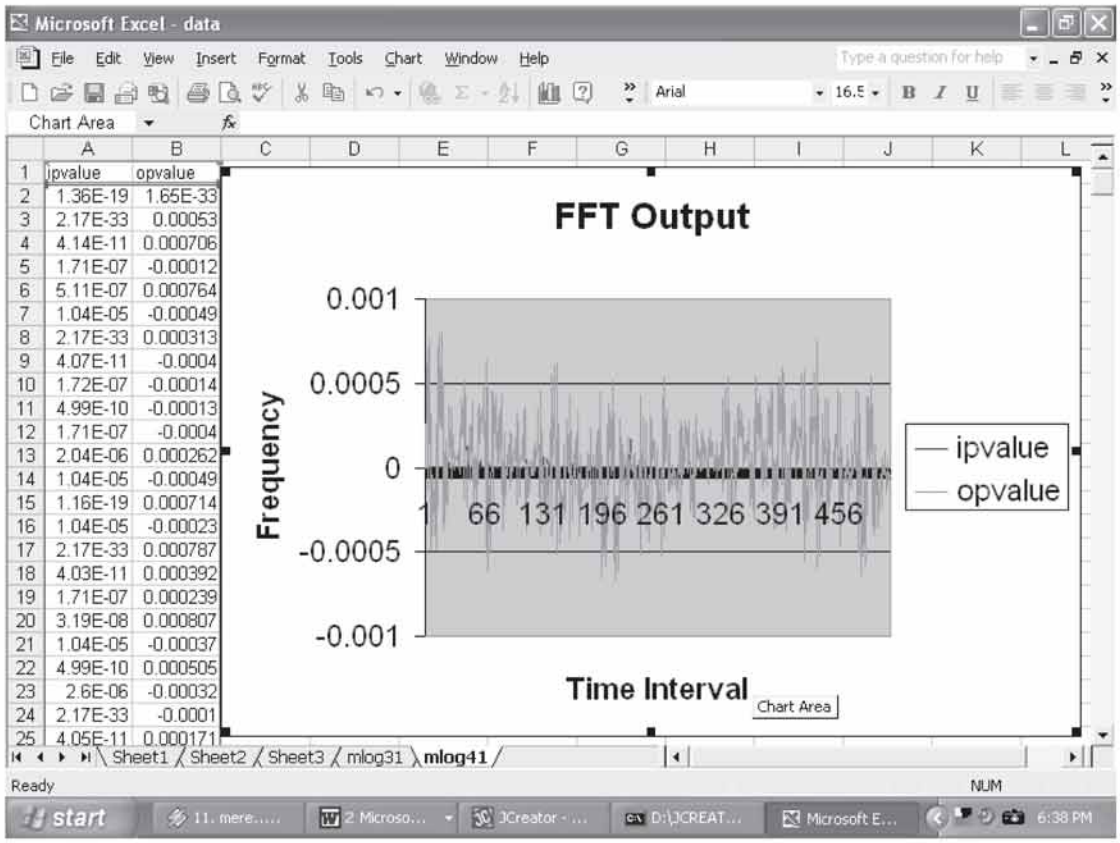

NGRI------MAGNETIC OBSERVATORY
File DataBase Process Print Details u canceled.
Menu item[Open] was pressed.
Open
Look In: source
BasicWindowMonit
Cosft
data
data
data1
ddata
ddata
Four1
GetTabValues1$1
GetTabValues1
GetTabValues1
hema
File Name: data1.txt
Files of Type: All Files
Open
Cancel
start
23. O BHA...
2 Microso...
JCreator - ...
D:\JCREAT...
NGRI------...
6:18 PM


NGRI------MAGNETIC OBSERVATORY
File DataBase Process Print Details u choose:data1.txt
FFT
Sin Transform
Cos Transform
Menu item[Open]
u choose:data1.b
start
6:19 PM

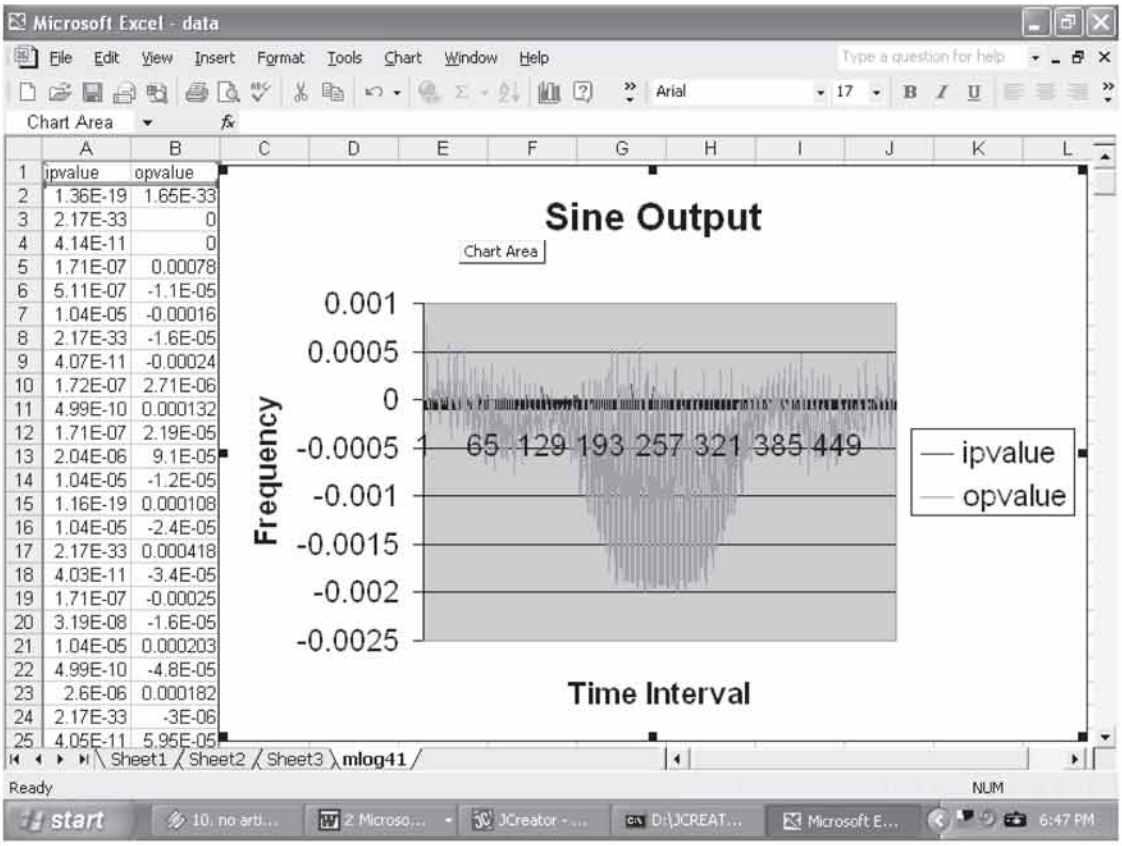

NGRI------MAGNETIC OBSERVATORY
File DataBase Process Print Details u choose:data1.txt
data[227]2.5347637E-9
data[228]4.9904547E-10
data[229]1.024318E-11
data[230]1.6222488E-7
data[254]1.0357755E-11
data[255]1.0384488E-5
FFT process will be called
Menu item[Save] was pressed.
Save
Save In: source
BasicWindowMonit
Cosft
data
data
data1
ddata
ddata
Four1
GetTabValues1$1
GetTabValues1
GetTabValues1
hema
File Name: FFTOutput.txt
Files of Type: All Files
Save
Cancel
start
6:28 PM


NGRI------MAGNETIC OBSERVATORY
File DataBase Process Print Details output of our selection will go here
Report
data[483]-5.3405034E-5
data[484]-5.714998E-4
data[485]4.1868337E-4
data[486]-2.724249E-4
data[487]4.1955875E-4
data[488]2.8039078E-4
data[489]2.2798486E-6
data[490]5.408992E-4
data[491]-3.4560048E-4
data[492]3.77874E-4
data[493]-3.9702468E-4
data[494]1.3716018E-4
data[495]-3.569985E-4
data[496]3.215256E-5
data[497]-3.5762892E-4
data[498]-6.9059264E-5
data[499]-2.5876172E-4
data[500]-2.185109E-4
data[501]-3.8102524E-5
data[502]-2.2271698E-4
data[503]6.428106E-5
data[504]-3.4220597E-5
data[505]-4.8646354E-5
data[506]9.819134E-5
data[507]-1.5433211E-4
data[508]3.923544E-5
data[509]-1.0708708E-4
data[510]-2.5891262E-5
data[511]-6.5593194E-5
writing will be completed
The values are saved in FFTOutput.txt file
start
10. mere k...
2 Microso...
JCreator - ...
D:\JCREAT...
NGRI------...
6:31 PM

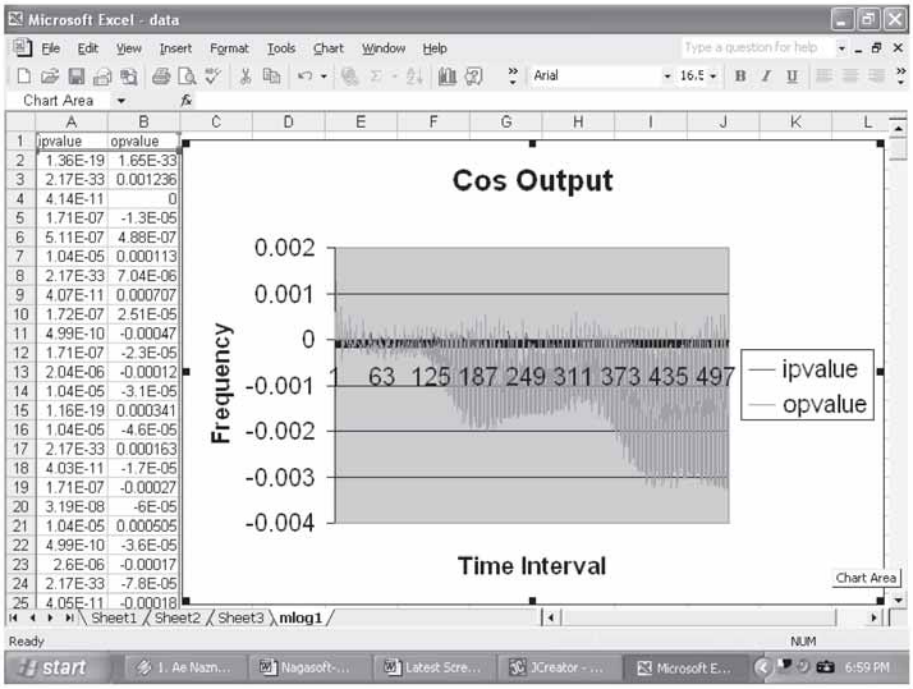

NGRI------MAGNETIC OBSERVATORY
File DataBase Process Print Details output of our selection will go here
ipvalue opvalue
1.35568414199512E-19 1.64950657031962E-33
2.1692657238731E-33 0.0
4.1390182947687E-11 0.0
1.70600145565913E-7 7.79620313551277E-4
5.111245968691 95E-7 -1.110654011426 96E-5
1.03833235698403E-5 -1.58597642439418E-4
2.1692657238731E-33 -1.64869270520285E-5
4.07080619213573E-11 -2.40059933275916E-4
1.71531425507965E-7 2.71309545496479E-6
4.99145114130073E-10 1.32173707243055E-4
1.70600145565913E-7 2.1894307792536E-5
2.04449838747678E-6 9.09619193407707E-5
1.03833235698403E-5 -1.21800894703483E-5
1.16216278199589E-19 1.08011816337239E-4
1.0382857908553E-5 -2.37778895098018E-5
2.1692657238731E-33 4.18271636590362E-4
4.02533145704709E-11 -3.44184263667557E-5
1.70600102933349E-7 -2.526205207686 87E-4
3.19452873043247E-8 -1.57872200361453E-5
1.03833272078191E-5 2.03216128284112E-4
4.99145058618922E-10 -4.84884803881869E-5
2.59600551544281E-6 1.82456686161458E-4
2.16921870409906E-33 -3.03325577988289E-6
4.04806882459141E-11 5.9509628044907E-5
1.6966873772617E-7 -4.67329809907824E-5
3.27119669236708E-5 1.69954902958125E-4
6.48972275030246E-7 -8.04515002528206E-5
2.169124664551E-33 5.73671946767718E-4
4.16175566231303E-11 -1.59827046445571E-4
start
10. mere k...
2 Microso...
JCreator - ...
D:\JCREAT...
2 java
6:32 PM



	A	B
1	ipvalue	opvalue
2	1.36E-19	1.65E-33
3	2.17E-33	0.00053
4	4.14E-11	0.000706
5	1.71E-07	-0.00012
6	5.11E-07	0.000764
7	1.04E-05	-0.00049
8	2.17E-33	0.000313
9	4.07E-11	-0.0004
10	1.72E-07	-0.00014
11	4.99E-10	-0.00013
12	1.71E-07	-0.0004
13	2.04E-06	0.000262
14	1.04E-05	-0.00049
15	1.16E-19	0.000714
16	1.04E-05	-0.00023
17	2.17E-33	0.000787
18	4.03E-11	0.000392
19	1.71E-07	0.000239
20	3.19E-08	0.000807
21	1.04E-05	-0.00037
22	4.99E-10	0.000505
23	2.6E-06	-0.00032
24	2.17E-33	-0.0001
25	4.05E-11	0.000171



	A	B
1	ipvalue	opvalue
2	1.36E-19	1.65E-33
3	2.17E-33	0
4	4.14E-11	0
5	1.71E-07	0.00078
6	5.11E-07	-1.1E-05
7	1.04E-05	-0.00016
8	2.17E-33	-1.6E-05
9	4.07E-11	-0.00024
10	1.72E-07	2.71E-06
11	4.99E-10	0.000132
12	1.71E-07	2.19E-05
13	2.04E-06	9.1E-05
14	1.04E-05	-1.2E-05
15	1.16E-19	0.000108
16	1.04E-05	-2.4E-05
17	2.17E-33	0.000418
18	4.03E-11	-3.4E-05
19	1.71E-07	-0.00025
20	3.19E-08	-1.6E-05
21	1.04E-05	0.000203
22	4.99E-10	-4.8E-05
23	2.6E-06	0.000182
24	2.17E-33	-3E-06
25	4.05E-11	5.95E-05



CONCLUSIONS

From the environment created the geomagnetic data signal can be converted in time domain to frequency domain using the FFT, Sine, Cosine, and the power spectrum functions. By conversion of the original signal into frequency values the impedance values can be computed. From those values one can determine the ground characteristics and whether the data is quiet data or disturbed data. One can also find the presence of magnetic storms, their amplitude and phases (Start, Middle, End) etc.

Java is used because of its capability for the development of user-friendly envirinment such as Graphical User Interface and computations.

FUTURE WORK

This environment can be adaptable with slight modifications to other geophysical data as seismic data, MT data, gravity data and so on.

REFERENCES

Booch, G., Rumbaug, J. & Jacobson, I., 2005. Unified Modeling Language User Guide, The (2nd Edition), Addison-Wesley Object Technology Series.

Brigham, E. O., 1988. The Fast Fourier Transform and Applications, Englewood Cliffs, NJ: Prentice Hall.

Cooley, J. W. & Tukey, J. W. "An algorithm for the machine computation of the complex Fourier series," *Mathematics of Computation*, 19, 297–301, Apr. 1965

Deitel, D. H. & Deitel, P. J., 2005, Java How to Program, 5th Edition, Pearson Education.

Jacobs, J. A., (Ed.) Geomagnetism, Vol.1, 1987, Academic Press.

Oppenheim, A. V. & Schafer, R. W., 1989. Discrete-Time Signal Processing, Prentice Hall, Engle Wood Cliffs, New Jersey 07632.

Parkinson W. D., Introduction to Geomagnetism, 1983, Scottish Academic Press.

Press, W. H., Vellorıng, W. T., Teukolsky, S. A. & Flannery, B.P., 1992. Numerical Recipes in C: The Art of Scientific Computing, Second Edition, Cambridge University Press.

Roberts, S., Heller, P. & Ernest, M., 2001. Complete Java2 Certification Guide, Second Edition, B P B Publications.

(Accepted 1st September 2006. Received 16th August 2006; in original form 21st July 2006)

Y.N.T. Seshagiri Rao did B.E. (Electronics & Communications) from Govt. (JNTU) College of Engineering, Kakinada, A.P. during 1964-69 and was a National merit scholarship holder. He has been with the National Geophysical Research Institute, Hyderabad since 1973. He worked in the areas of Geophysical Instrumentation, Artificial Neural Networks. Scientific Data Warehousing and wrote software in Java Language and delivered lectures.