Sindh Univ. Res. Jour. (Sci.Ser.), Vol: 34 (2) 29 -36 (2002)

DESIGN, DEVELOPMENT AND IMPLEMENTATION OF JAVA MATHEMATICAL KIT

Nasrullah Memon, Mushtaq Ahmed Korai,* Kamal Mustafa Shaikh,**
Jamal Mustafa Shaikh***

Department of Computer Systems and Software Engineering, Mehran University of Engg., and Tech. Jamshoro

Abstract

An approach for design, development and implementation of Java Mathematical Kit (JMK) is considered. It aims to establish a framework to support solution of various mathematical problems by using click and done methodology. The JVM is a mathematical software which has some unique features (Step Evaluator, etc.), those are not available in other mathematical software.

Key Words: Development and Implementation of Java Mathematical Kit (JMK).

1. Introduction

During the initial development of computers in the 1940's and early 1950's the chain of intermediate technologists between the computer designer/builder and the computer user was very short. In fact it was not uncommon for one individual to be actively involved in the design, construction, and application of a computer. By 1970 there was a significant amount of infrastructure between the computing machine and the end user - for example, from low-level to high-level one could cite operating systems, compilers, general-purpose software, and applications-specific software. Each level of this infrastructure absorbed the creativity and energies of its own community of specialists trying to improve its link in the chain. This frequently took the form of trying to meet the perceived needs of the higher levels in the infrastructure while coping with the limitations and peculiarities of the resources provided by the lower levels.

The Java mathematical Kit is a contribution to the general-purpose mathematical software stratum of the computing infrastructure.

2. Background and Motivation for the Java Mathematical Kit

Through the late 1960's and up to about 1971 a number of collections or libraries of mathematical software had been assembled. Typically each of these libraries was developed for, and useable on, just one brand of computer system. Examples would be a library collected and/or developed by a computer vendor for use with their computer, or a library collected and/or developed by a large organization's computing center for use within that organization. For instance, Michael Powell recalls developing a math library at the Harwell Laboratory in the early 1960's.

Even the two organizations, NAG [1] and IMSL [2], that later became the world's main suppliers of math libraries across diverse machine types, each started in 1970 by targeting a single machine type. NAG started as an informal cooperative effort by six institutions in

* Department of BS&RS, Quid-e-Awam University of Engg., Sciences and Technology, Nawabshsh
** Institute of Information Technology, MUET, Jamshoro
*** Institute of Information Technology, University of Sindh Jamshoro

England to develop a math library for their newly acquired ICL 1906A systems. Their first library, in both Algol and Fortran versions, was available to the participating institutions in October 1971, and they immediately started attacking the problem of developing the library for other computer types.

IMSL was founded in 1970 as a company to develop and market mathematical and statistical libraries. A major motivation for the founding of IMSL was a perception of customer dissatisfaction with the math libraries provided by IBM for use on their systems. IMSL's first product, available in 1971, was a Fortran library for the IBM/370-360.

Designing math library software to be portable across diverse systems was particularly difficult in the 70's because there were significant differences between the arithmetic characteristics of computers from different manufacturers, the state of standardization of programming languages was in its infancy, and operating systems (encompassing file naming and file storage) were quite different on different computer brands. Fortran was the only language widely supported in the U.S., and the only language for which there was an ANSI standard (1966). However, the standardized language had major shortcomings so programmers tended to use vendor-specific extensions and that worked against portability. Algol and Fortran were both used in Europe.

A project called NATS (National Activity for Testing Software or NSF, Argonne, Texas, Stanford) [3] undertook during 1971-1972, to investigate the problems of achieving Fortran portability for mathematical software by testing, and as necessary modifying, a Fortran version of a set of dense matrix eigenvalue/eigenvector algorithms. These 30 algorithms were originally developed in Algol and published in a series of papers in the 1960's by J. H. Wilkinson and others, and subsequently published together as the *Handbook* ... [4]. The collection was converted to Fortran by Virginia Klema and others at Argonne National Laboratory and then processed for portability by NATS. NATS announced the availability of EISPACK [2, 6, 7] in 1972, the package having been successfully tested on IBM 360-370, CDC 6000-7000, Univac 1108, Honeywell 635, and PDP-10 computers. The same EISPACK code ran on all these systems, with the exception of one variable that needed to be set to indicate the precision of the arithmetic on the host system. EISPACK was widely requested and distributed. This project demonstrated the great utility of carefully tested portable mathematical software based on state of the art algorithmic research. But did (Fortran) portability mean sacrificing efficiency? Library developers began giving more attention to this issue, and new hardware designs presented new challenges and opportunities.

Nowadays many mathematical software are available in market, for example, MathCad, Mathematica, Matlab, TK Solver, Adams, etc. We developed our software (Java Mathematical Kit) using Java Technology as our tool of development.

3. In What sense it is different from other softwares .

This software holds some extra ordinary features like,

- Graphical User Interface (Click and done).
- Step Evaluator.
- Matrixulator.
- Dynamic Graphs (GML).
- Dynamic Error Indicator.

3.1 Graphical User Interface:

This Software has User Friendly and Self Explanatory Graphical Interface (as shown in figure 1), means user doesn't have to bother about Syntaxes of Commands .He/she has to just enter the function with proper conventions and then select the required command, to cope with problem. For Invalid inputs, the software generates dynamic error with exact location, indicating where the user has made the error. Thus by knowing what and where error has occurred, user has a facility to correct it without much botheration.

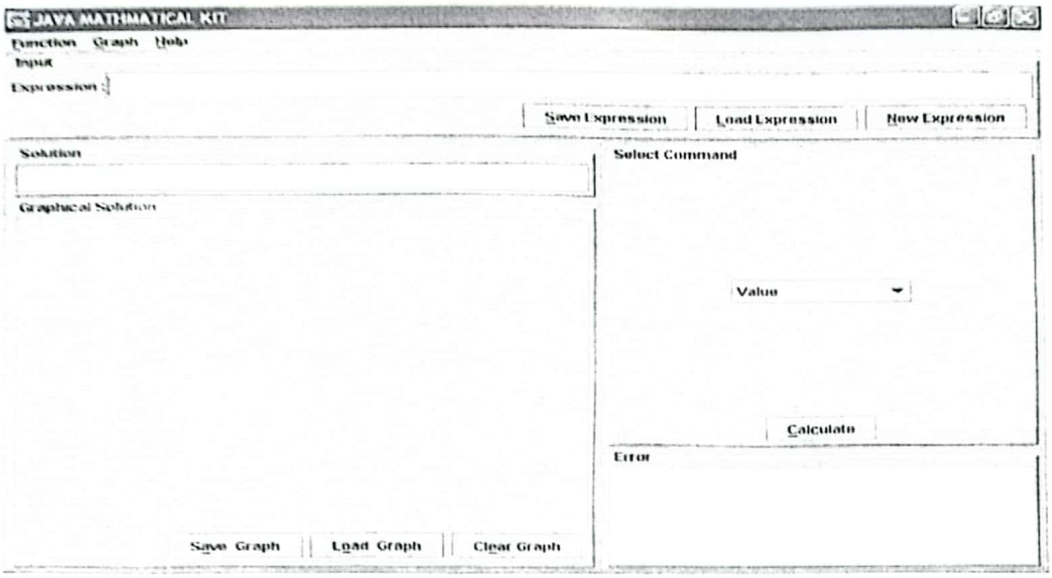


Figure 1: Graphical User Interface

3.2 Step Evaluator

This Software has a very unique quality of solving an expression or function Step by Step. With Step Evaluator, user views the way of solving an expression, which he/she does manually, step by step. Therefore user may solve an expression with Step Evaluator, save it and print it. This feature is not present in any Software, as shown in figure 2.

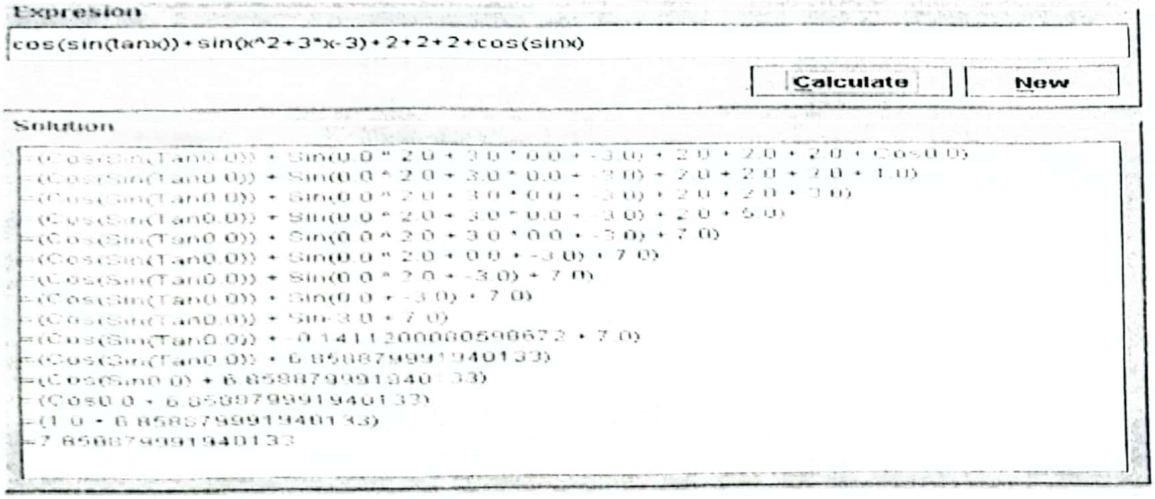


Figure 2: Preview of Step-Evaluator

3.3 Matrixulator:

This software has a special feature named as "Matrixulator" for matrix manipulation. By this user may add, subtract, and multiply any two suitable matrices. The user may find determinant of a matrix. Moreover, all these operations can be performed step by step. This is also a unique property of the software.

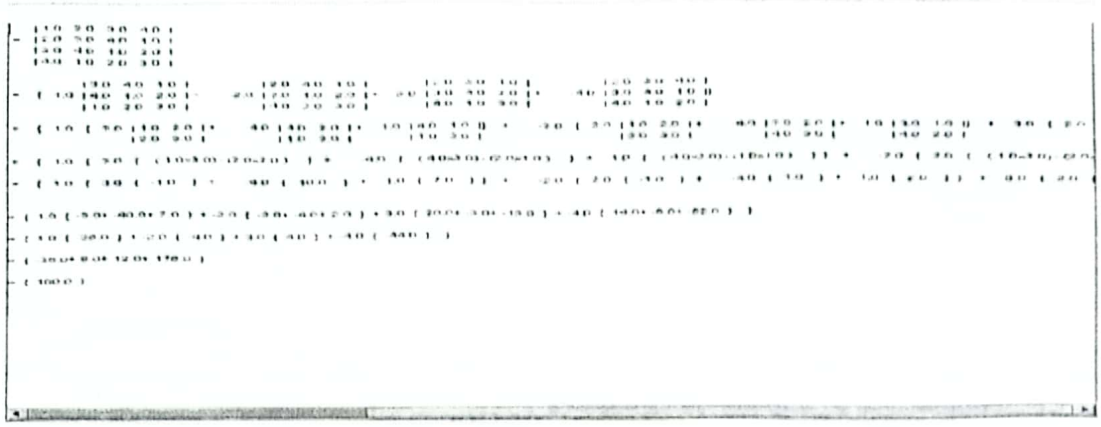


Figure 3 : Preview of Matrixulator

3.4 Dynamic Graphs:

This Software has efficacies to draw graph of any function in one variable as shown in figure 4. As the name suggests user may change the scale and view of the graph dynamically, this means at runtime, after drawing the graph. The user may save the current graph or load previously saved graphs. This is done by GML (stands for Graphic Markup Language), a special format in which this software saves and retrieves the graph. Again you won't find these facilities common.

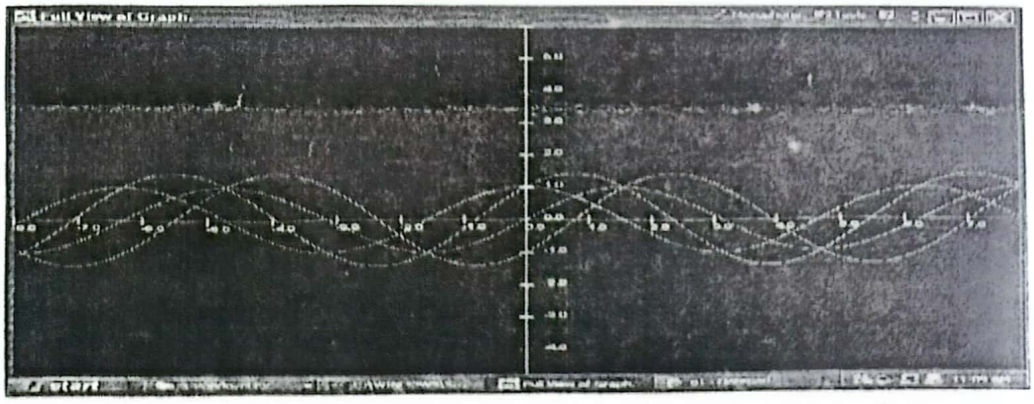


Figure 4: Dynamic Graphs

3.5 Dynamic Error Handling:

This Software generates different type of errors, with the location numbers to help the user to correct it. This property is shown in Figure 5 and discussed in detail in the next section.

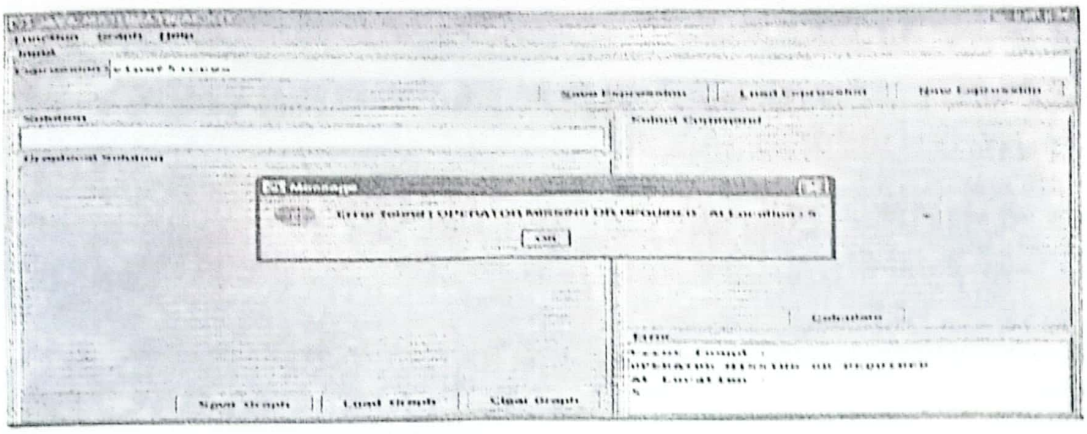


Figure 5: Dynamic Error Handling

4. Dynamic Error Detecting Facilities:

This Software has some special Error Trapping Routines, which are designed to locate all possible types of dynamic errors which a user may commit, during input session.

Self Detecting Skills of this software not only locate the error, but also let the user know what type of error he/she has made and where he has made (exact position in input).

The Table 1 shows errors that appear when user enters invalid inputs.

5.0 Working of Java Mathematical Kit

The user inputs a string which is send to the Expression class, which in turn converts it to postfix and then evaluates it for the instant to find the value of function at that instant. This value is used by class Command and its subclasses to implement their algorithms. Then the answer is converted back to string. This String is calculated Output. This whole process is shown in Figure 6.

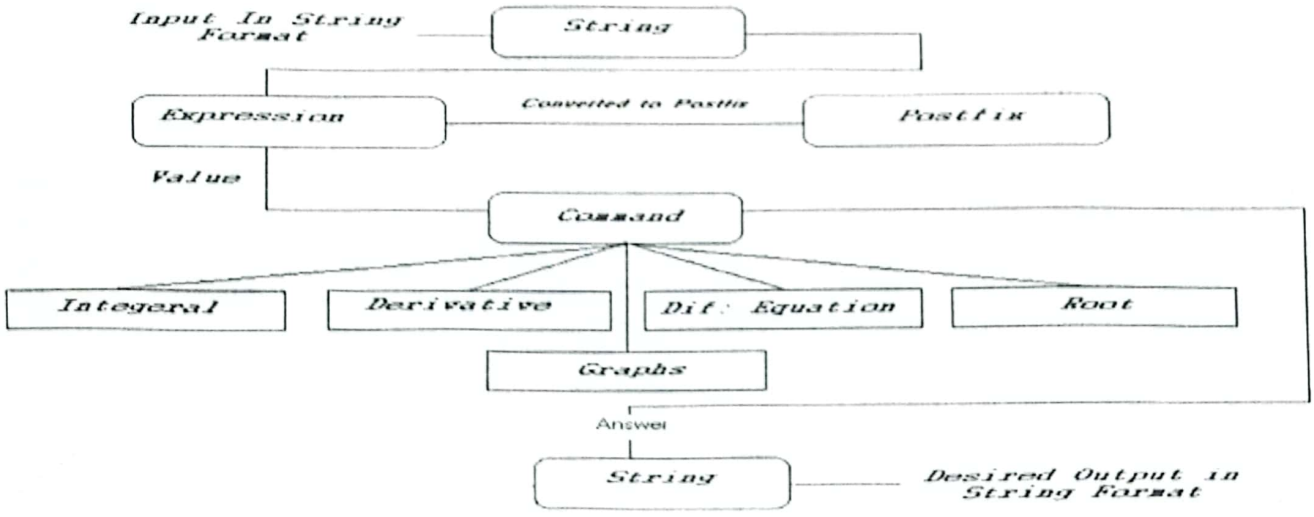


Figure 6: Working of Java Mathematical Kit

6. Function writing conventions:

The following rules may be applied in writing functions.

- No White space in Expression.
- Function must be in one variable, that is, "x".
- The Fundamental Binary Operators which a user can use in Function are

1. " + " for Addition
2. " - " for Subtraction
3. " * " for Multiplication
4. " / " for Division
5. " ^ " for Power

- The Unary Operators which a user can use in Function are

1. Sinx for Sine
2. Cosx for Cosine
3. Tanx for Tangent
4. Sin~x for arcSine
5. Cos~x for arcCosine
6. Tan~x for arcTangent
7. Log for Logarithm with base 10
8. Ln for Natural Logarithm
9. Exp for e

- Parenthesis may be used to increase or decrease the precedence of any operation.
- Function is not Case Sensitive.

5. Future Extensions

The authors are still working for the extension of the software and hope that the following extensions be implemented in very short span of time.

1. Simultaneous Linear Equation Solver
2. Non Linear Equation Solver
3. Step by Step Derivative Calculator
4. Discrete Mathematician
5. 3D graphs (graphing the functions involving two variables)

6. Conclusions

One of the motivations for developing the Java Mathematical Kit is to provide features, which are not available in any other mathematical software and we hope that this software will help both students and teachers.

In this paper, we presented the design, development, and implementation of mathematical software named as Java Mathematical Kit. With this software we can:

- Integrate a function and find its definite value,
- Differentiate a function at a point,
- Find Roots of an equation and solve simultaneous equations,
- Find Value of function at different instants,
- Finding the roots of equations graphically,
- Drawing, Saving and Loading graphs of Function.

This Software is Robust. It possesses Self-detecting ability for any type of dynamic and input errors. It cannot only detect what type of error is made but also hits where the error has occurred (if the error is in input by user).

This Software not only finds the values of functions but can also show all the steps required for performing the task. This functionality is an important feature of this software in the shape of " Step - Evaluator "(a special part of this software).

Moreover, this software is also capable of performing matrix arithmetic, means it can add, multiply, and subtract two suitable matrices. Besides, it can also find determinants and inverse of the matrix. If the user wants it can also show the steps which one will perform manually.

This Software is dynamic, and can save or load your work and graphs if you wish. This is done with the help of GML (Graphics Markup Language), a special format, which saves and loads the graph.

7.0 Acknowledgements

The authors are very thankful to the Chairman Department of Computer Systems and Software Engineering, Mehran University of Engineering and Technology, Jamshoro, for providing all facilities at Advanced Software Engineering Lab of the department for the development of the software.

References

[1] B. FORD, *The Nottingham Algorithms Group (NAG) Project*, ACM SIGNUM Newsletter, 8, 2, April 1973, pp. 16-21.

[2] O. G. JOHNSON, *IMSL's ideas on subroutine library problems*, ACM SIGNUM Newsletter, 6, 3, Nov 1971, pp. 10-12.

[3] ANNOUNCEMENT, *NATS Project - Collaborative Research toward the Development of a Certified Sub-routine Library*, ACM SIGNUM Newsletter, 6, 3, Nov 1971, p. 5.

[4] J. H. WILKINSON AND C. REINSCH, eds., *Handbook for Automatic Computation, Vol. 2, Linear Algebra*, Springer-Verlag, New York, 1971.

[5] ANNOUNCEMENT, *The Certified Eigensystem Package*, EISPACK, ACM SIGNUM Newsletter, 7, 2, July 1972, pp. 4-5.

[6] B. S. GARBOW, J. M. BOYLE, J. J. DONGARRA, AND C. B. MOLER, *Matrix Eigensystem Routines - EISPACK Guide Extension*, Vol. 51 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1977.

[7] B. T. SMITH, J. M. BOYLE, B. S. GARBOW, Y. IKEBE, V. C. KLEMA, AND C. B. MOLER, *Matrix Eigensystem Routines - EISPACK Guide*, Vol. 6 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1974 (2nd ed., 1976, with additional author, J. J. DONGARRA).

[8] MathML International Conference 2000, `www.mathmlconference.org`, UIUC Illinois USA, Oct. 20-21, 2000.

[9] Workshop on *The Future of Mathematical Communication* http://www.msri.org/activities/events/9900/fmc99/, Dec. 1999.

[10] ActiveMath, http://www.mathweb.org/activemath

[11] Calc101, http://www.calc101.com/

[12] Ezmath, http://www.w3.org/People/Raggett/EzMath/

[14] Institute for Computational Mathematics, demos of mathematical computation http://icm.mcs.kent.edu/research/demo.html

S.No.	Type	Input	Error Type displayed	Error location displayed	Example
1	Ambiguous Operators	Empty parenthesis like "()"	Yes	Yes	5+()+4
2	Extra "(" in Expression	Extra parenthesis like "("	Yes	No	5+(()+4
3	Extra ")" in Expression	Extra Parenthesis like ")"	Yes	No	5+())+4
4	Unexpected Decimal Point	An extra Decimal point	Yes	Yes	2.3.4
5	Unnecessary Decimal Point	Continuous Decimal points	Yes	Yes	2..3
6	Unidentified Letter or Symbol	Letter which is not used in any symbol	Yes	Yes	Sin χr+ +cosx
7	Operator Missing or Required	Two operands without operator	Yes	Yes	2x
8	Operand Missing or Required	Two operators without operand	Yes	Yes	2+*x
9	Undefined Symbol	A symbol comprising of letters which are used in other Symbols	Yes	Yes	cinx
10	Incomplete Equation	An unusual end of expression is made	Yes	No	5+4+
11	Nan	Type of Error comes when the result of required operation comes as a non real value	Yes	No	Root of x^2+2
12	Infinity	Type of Error comes when the result of expression is infinity	Yes	No	8/0

Table 1: Dynamic Error Detecting Facilities