

NR - ISSN 0001-3099



abacus

THE JOURNAL OF THE MATHEMATICAL
ASSOCIATION OF NIGERIA

VOLUME 35, NUMBER 2
MATHEMATICS SERIES
AUGUST 2008

DR (Mrs.) OLADELE
ABACUS

**THE JOURNAL OF
MATHEMATICAL ASSOCIATION
OF NIGERIA (M.A.N)**

NR-ISSN-3099

**EDITOR-IN-CHIEF
PROFESSOR J. S. SADIKU**
DEPARTMENT OF MATHEMATICS, FACULTY OF SCIENCE,
UNIVERSITY OF ILORIN, ILORIN, NIGERIA.

© THE MATHEMATICAL ASSOCIATION OF NIGERIA

THE MATHEMATICAL ASSOCIATION OF NIGERIA (M.A.N)

COUNCIL MEMBERS 2007/2008

Alh. A. O. Okuboyejo	-	National President
Prof. I. A. Ochepe	-	National Vice President
Dr. S. A. Okunuga	-	National Secretary
Prof. J. S. Sadiku	-	Editor-in-Chief, ABACUS (The Journal of Mathematical Association of Nigeria)
Dr.(Mrs.) M. F. Salman	-	National Financial Secretary
Mrs. P. N. Ezenwelu	-	National Treasurer
Mr. Christopher Yelposu	-	National Assistant Secretary
Mal. Mohammed Tukur	-	National Publicity Secretary
Mr. Ishaya Amutu	-	National Business Manager
Dr. (Mrs.) A. A. Eniayeju	-	Immediate Past National President and Ex-officio Member.
Mal. S. M. Sirajo	-	Immediate Past National Secretary and Ex-officio Member
Dr. M. O. Ibrahim	-	Ex-officio Member
Mr. E. S. Ebisine	-	Ex-officio Member

CHAIRPERSONS OF ALL STATE BRANCHES

ABACUS:

The Journal of the Mathematical Association of Nigeria, Volume 35, Number 2, August 2008.

Abacus, the Journal of the Mathematical Association of Nigeria, publishes very carefully selected papers covering all areas of Mathematics, Mathematics Education, Statistics Computer Science and Engineering.

The Journal publishes five types of contributions namely: survey papers, contributed papers, technical notes, technical comments and book reviews, the latter strictly by invitation of the editors through the Editor-in-Chief of the Journal.

Two issues (Mathematics series and Mathematics Education series) of the Journal are published in August or September of every year. Author(s) is/are to pay ₦1,200.00 for processing a paper. Three copies of the manuscript should be forwarded for review.

The papers should be typeset using 7" by 9" dimension, margin set at: left 1", right 0.5, top 0.5 and bottom 0.5". Send three copies of your article to the Editor-in-Chief:

EDITOR-IN-CHIEF

PROFESSOR J. S. SADIKU

DEPARTMENT OF MATHEMATICS, FACULTY OF SCIENCE,
UNIVERSITY OF ILORIN, ILORIN, NIGERIA.

The reviewers' comments are then sent to the authors(s). The revised article should be forwarded to the Editor-in-Chief in a CD or flash drive with two hard copies included. Microsoft word is preferred. The use of Microsoft Drawing for equation and grouping of equations is not accepted. Any other software used in typing your article aside from Microsoft word should be accompanied with the installation CD so as to make editing easy.

EDITORIAL BOARD

PROFESSOR J. S. SADIKU

(Editor-in-Chief)

Department of Mathematics, Faculty of Science,
University of Ilorin, Ilorin, Nigeria.

Professor P. Onumanyi
National Mathematical Centre,
Abuja, Nigeria.

Professor S. O. Iyehen
National Mathematical Centre,
Abuja, Nigeria.

Professor J. Amazigo
Department of Mathematics
University of Nigeria, Nsukka
Nigeria.

Professor M. S. Audu,
Department of Mathematics
University of Jos, Jos, Nigeria.

Professor S. O. Ale
National Mathematical Centre,
Abuja, Nigeria.
Nigeria

Professor S. A. Ilori
Department of Mathematics
University of Ibadan, Ibadan,

Professor R. O. Ayeni
Department of Pure & Applied
Mathematics, Ladoke Akintola
University of Technology, Ogbomoso,
Nigeria

Professor P. N. Lassa
Faculty of Education
University of Jos, Jos, Nigeria

Professor M. O. Fajemidagba
Department of Science Education
University of Ilorin, Ilorin, Nigeria.

Professor K. V. Aderogba
Department of Mathematics
University of Lagos, Lagos,
Nigeria.

ASSOCIATE EDITORS

Professor L. O. Adetula
National Mathematical Centre, Abuja
Nigeria.

Dr. I. U. Jahun
Federal College of Education
Bichi, Kano, Nigeria

Professor E. T. Jolayemi
Department of Statistics
University of Ilorin, Ilorin, Nigeria

Professor E. A. Akinrelere
Department of Mathematics
Obafemi Awolowo University
Ile-Ife, Nigeria.

Professor Herbert Wills
Department of Mathematics
Education, Florida State University,
Tallahassee, Florida, U.S.A.

Dr. A. A. Tijjani
Department of Mathematics
Ahmadu Bello University, Zaria,
Nigeria.

Dr. M.O. Oyesanya
Department of Mathematics
University of Nigeria, Nsukka,
Nigeria

Professor J. A. Adepoju
Department of Mathematics
University of Lagos, Lagos,
Nigeria.

Professor O.S. Adegboye
Department of Statistics
University of Ilorin, Ilorin, Nigeria

Professor M. A. Ibiejugba
Department of Mathematics
University of Ilorin, Ilorin, Nigeria

Professor G. M. Gladwell
University of Waterloo, Waterloo,
Canada

Professor A. U. Afuwape
Department of Mathematics
Obafemi Awolowo University
Ile-Ife, Nigeria.

Professor G. O. S. Ekhaguere
Department of Mathematics
University of Ibadan, Ibadan, Nigeria

TABLE OF CONTENTS

	Page
1. Reuben, U. and Iyai, D. - Asymptotic Null Controllability of Linear Neutral Volterra Integrodifferential Systems With Distributed Delays In Control	1
2. Okunuga, S. A. and Sofoluwe, A. B. - A Fifth Order Padé Integration Formula for Stiff Problems	6
3. Ette, A.M. - Perturbation Scheme on the Stability of Cylindrical Shells Stressed by External Pressure	14
4. Ifiok, A. E. - An Alternative Method for Calculating the Distance between Two Points on the Surface of the Earth.	27
5. Odule, T. J. - Understanding and Managing Computer Viruses	38
6. Oluyo, T.O and Ayeni, R.O - Mathematical Analysis of the Global Dynamics for Infectious Diseases	47
7. Ajala, O.A and Ayeni, R.O - A Note on the Existence and Uniqueness of a Bounded MHD Boundary Layer Flow Over a Moving Vertical Cylinder	50
8. Oladele, T. O., Bamigbola, O. M. and Gbadeyan, J. A. - Concept of Strings and Trees in Bioinformatics.	56
9. Amponsah, S. K., Darkwah, K. F., Oduro, F. T. and Ofori-Kusi, D. - Fibonacci Search Method On a TI-89 Graphing Calculator	68
10. Oladele, T. O. - A Survey of Programming Languages Relevant to Bioinformatics	85
11. Gbadeyan, J. A. Sadiku, J. S. and Babatunde, A. O. - Analysis of Pascal, Basic and C Programming Languages	92
12. Lasode, O. A. - Heatline Visualization of Unsteady Natural Convection in Square Enclosures with Top Conducting Wall	96
13. Ajibade, A. O., Yakubu, M. I. and Alkali, A. - Reversibility on the Set of Singular Matrices	133
14. Makanjuola, S.O. - A Class of Type w - Semigroup	140
15. Oyesanya, M. O. - Some Remarks on the Stability of the Imperfect Structure	143

16. Ngiangia, A., Ogbondah, C., Womuru, E. N., Wonu, N. & Ejekudo, N.
- A Study on the Effect of Transverse Magnetic Field on the Trajectory
of a Free Jet 154
17. Oyelami, B. O., Ale, S. O. & Sesay, M. S. - On Existence and Stability with
Respect to Invariant Sets for Impulsive Differential Equations with Variable
Times. 160
18. Kaurangini, M.L. & Jha, B. K. - Combined Effects of Free and Forced
Convection Flow in a Vertical Annulus Filled with Porous Material Having
Variable Porosity 179
19. Nwachukwu, G. C. & Otunta, F. O. - An Extension of Variable Order
One-Step Methods for Initial Value Problems. 193-199

Sufficient conditions for the asymptotic null controllability of linear neutral Volterra integrodifferential systems with distributed delays in the control variables are developed. The results are obtained by using the nondegeneracy of the controllability matrix. An example is provided to illustrate the obtained result.

1. INTRODUCTION

Neutral functional differential equations are characterized by a delay in the derivative. Equations of this type have applications in the study of electrical networks containing lossless transmission lines. It is well known that the mixed initial-boundary hyperbolic partial differential equation which arises in the study of lossless transmission lines can be replaced by an associated neutral functional differential equation [2]. Volterra integrodifferential equations occur often in applied mathematics [4]. For instance, in [8] a simplified model for compartmental systems with pipes is represented by nonlinear neutral Volterra integrodifferential equation.

The problem of controllability and null controllability of linear neutral systems has been studied by several authors [3,6,10,14]. Chukwu [11] and Angeli [7] discussed the functional controllability of nonlinear neutral systems and Underwood and Chukwu [9] studied the null controllability for such systems. Onwunatu [5] discussed the problem of null controllability for nonlinear systems of neutral functional differential equations with limited controls. The purpose of this paper is to establish a set of sufficient conditions for the asymptotic null controllability of linear neutral Volterra integrodifferential systems with distributed delays in control. This is motivated by the fact that models for systems with delay in the control occur in population studies and in some complex economic systems. Specifically, models for systems with distributed delays in the control occur in the study of agricultural economics and population dynamics [1, 12].

2. PRELIMINARIES

Consider the linear neutral Volterra integrodifferential systems with distributed delays of the form

$$\frac{d}{dt} \left[x(t) - \int_0^t C(t,s)x(s)ds - g(t) \right] = A(t)x(t) + \int_0^t G(t,s)x(s)ds + \int_0^t d_\theta B(t,\theta)u(t+\theta)$$

A SURVEY OF PROGRAMMING LANGUAGES RELEVANT TO BIOINFORMATICS

OLADELE T. O.

Department of Computer Science, University of Ilorin, Ilorin, Nigeria.

ABSTRACT

Currently, the most popular use of computer programming languages in the biological sciences is in the field of bioinformatics. A frequent task needed in bioinformatics is to find certain patterns in data files. These data files often contain information in various formats. A solution is to use regular expressions to specify these patterns so that the information can be mined from the data files. Scripting languages are often used in bioinformatics due to their ability to work with regular expressions.

In this paper, a general overview of the programming languages used for scientific research and discovery in the biological sciences is presented.

1. INTRODUCTION

Several common bioinformatics tools are available to the bioinformatician. A great deal of tasks can be accomplished solely with them. There are times when one need to address questions for which an application does not exist. This is not a rare occurrence since original research projects often require original analysis.

More importantly, many situations in the modern biology lab require manipulation of very large datasets in order to process data for use in core applications and to help organize and interpret the results emanating from them. In these circumstances, it is useful for a biologist to have at least some basic programming skills at their disposal. Apart from the lab specific tasks, programming can also open up to broader applications for use by the general biological community.

This paper presents a general survey of the programming languages used for scientific research and discovery in bioinformatics.

2. PROGRAMMING LANGUAGES

Computer languages have undergone dramatic evolution since the first electronic computers were built to assist in telemetry calculations during world war II. Earlier on, programmers worked with the most primitive computer instructions: machine language. These instructions were represented by long strings of ones and zeros. Soon, assemblers were invented to map machine instructions to human-readable and manageable mnemonics such as add and mov.

The purpose of a programming language is to instruct the computer on how to solve problems or to perform tasks. Programming languages differ from other languages in the level of specificity required. There are thousands of programming languages, with new ones being introduced each year, but only a few of them have gained widespread usage. The first widely used language was FORTRAN, which was good at handling numeric data, but COBOL was subsequently developed for business applications. The Pascal, C, C++, and Java programming languages have been widely implemented since that time. These programming languages can take extensive amount of time to learn or master. Furthermore, building a "full-featured" application can take years, an investment that the typical researcher will not be interested in making. What is frequently needed in bioinformatics is not a full featured elaborate program with a stylistic

graphical user interface (GUI), but a simple, efficient, and accurate method to manipulate data for use in other applications. Fortunately, a programming language (Perl) exists that is particularly well suited for these tasks.

Scripting languages such as Perl, Python, PHP and Ruby gives an environment that is easier to use. Just a few lines of scripting could replace many lines of Fortran and the interactivity means that researchers could work with the data more easily and flexibly. In addition, these programs came with source code so that new features could be added when the need arises.

PROCEDURAL, STRUCTURED AND OBJECT-ORIENTED PROGRAMMING LANGUAGES

Until recently, programs were taught of as a series of procedures that acted upon data. A procedure, or function, is a set of specific instructions executed one after the other. The data was quite separate from the procedures, and the trick in programming was to keep track of which functions called which other functions, and what data was changed. To make sense of this potentially confusing situation, structured programming was created.

The principle idea behind structured programming is as simple as the idea of divide and conquer. A computer program can be thought of as consisting of a set of tasks. Any task that is too complex to be described simply would be broken down into a set of smaller component tasks until the tasks were sufficiently small to be executed and self-contained enough that they were easily understood. As an example, computing the average salary of every employee of a company is a rather complex task. It can however, be broken down into the following subtasks:

- Find out what each person earns
- Count how many people you have
- Total all the salaries
- Divide the total by the number of people you have.

Totaling the salaries can be broken down into:

- Get each employee's record
- Access the salary
- Add the salary to the running total
- Get the next employee's record.

Structured programming remains an enormously successful approach for dealing with complex problems. By the late 1980s, however, some of the deficiencies of structured programming had become all too clear.

First, it is natural to think of your data (employee records, for example) and what you do with your data (sort, edit, and so on) as related ideas.

The way we are now using computers – with menus and buttons and windows give a more interactive, event-driven approach to computer programming. Event-driven means that when an event happens the user presses a button or chooses from a menu and the program responds. Programs are becoming increasingly interactive, and it has become important to design for this kind of functionality.

Object-oriented programming attempts to respond to these needs, providing techniques for managing enormous complexity, achieving reusability of software components, and coupling them with the tasks that manipulate that data.

The essence of object-oriented programming is to treat data and the procedures that act on the data as a single "object", a self-contained entity with an identity and certain characteristics of its own. Some Object-oriented programming languages used in developing applications in bioinformatics are given below.

C++

C++ fully supports object-oriented programming, including the four pillars of object-development: encapsulation, data hiding, inheritance and polymorphism. For instance, when an engineer needs to add a resistor to the device he is creating, he doesn't typically build a new one from scratch. He walks over to a bin of resistors, examines the coloured bands that indicate the properties, and picks the one he needs. The resistor is a "black box" as far as the engineer is concerned. He doesn't care how it does its work as long as it conforms to his specifications.

The property of being a self-contained unit is called encapsulation. With encapsulation we can accomplish data hiding.

Similarly, when the engineer uses the resistor, he doesn't have to know anything about the internal state of the resistor. All the properties of the resistor are encapsulated in the resistor object; they are not spread out through the circuitry. C++ supports the properties of encapsulation and data hiding through the creation of user-defined types, called classes. C++ supports the idea of inheritance. A new type, which is an extension of the existing type, can be declared. This new subclass is said to be derived from the existing type and is sometimes called a derived type.

C++ is a superset of C. Its implementation exist for nearly modern platforms including both Unix and non-unix operating systems. You can learn C++ without having to learn C.

Programming with C++

Your compiler may have its own built-in text editor, or you may be using a commercial editor or word processor that can produce text files. The important thing is that whatever you use to write your program in, it must save simple, plain-text files, with no word processing commands embedded in the text. Examples of safe editors include windows Notepad, the Dos Edit command, Epsilon, EMACS or vi.

The files you create with your editor are called source files, and for C++ they typically are with the extension .cpp.

JAVA

Java is considered as a powerful tool for Internet development projects. Java is a simplified version of C++.

Java is "simple, object-oriented, statically typed, compiled, architecture neutral, multithreaded, garbage collected, robust, secure, and extensible". The following list of Java attributes however should give your more insight.

Simple - Java's developers deliberately left out many of the unnecessary features of other high-level programming languages. Java does not support pointers, implicit type casting, structures or unions, operator overloading, templates, header files, or multiple inheritance.

- * Objected-oriented - Just like C++, Java uses classes to organize code into logical modules.
- * Statically typed - All objects used in a program must be declared before they are used. This enables the Java Compiler to locate the report type conflicts.
- * Compiled - Before you can run a program written in Java language, the program must be compiled by the Java compiler.
- * Multi-threaded - Java programs can contain multiple threads of execution, which enables programs to handle several tasks concurrently. For example, a multi-threaded program can render an image on the screen in one thread while continuing to accept keyboard input from the user in the main thread.
- * Garbage collected - Java programs do their own garbage collection, which means that programs are not required to delete objects that they allocate to memory. This relieves programmers of virtually all memory management problems.
- * Robust - Because the Java interpreter checks all system access performed within a program, Java programs cannot crash the system.
- * Secure - The Java system does not only verifies all memory access but also ensures that no viruses are hitching a ride with a running applet. Because pointers are not supported by the Java language, programs cannot gain access to areas of the system for which they have no authorization.
- * Extensible - Java programs support native methods, which are functions written in another language, usually C++.
- * Well understood - Java can be quickly and easily understood by anyone with experience with modern programming languages such as C++.

Java Programs

Java can be used to create two types of programs: applets and stand-alone applications. An applet is simply a part of a web page, just as an image or a line of text can be.

The Java Developer's Kit

Java is actually more than a computer language; it is also a programming environment that includes a complete set of programming tools. Those tools include a compiler, an interpreter, a debugger, a disassembler, a profiler, and more.

To create a Java program, you first use a text editor to create the source code file. You write the source code, in the Java language. After completing the source code, which is always saved with a Java file extension, you compile the program into its byte-code format, the file for which has the .class extension. It is the .class file that the interpreter loads and executes.

3. PROGRAMMING LANGUAGES RELEVANT TO BIOINFORMATICS

Common languages relevant to Bioinformatics are C, C++, C#, JAVA, PERL, and PYTHON. These languages have been used generally in developing applications in bioinformatics.

Scripting languages are found to be more efficient in text or string processing and data manipulation in bioinformatics. Scripting languages are described below:

(a) PERL (Practical Extraction and Report Language)

Perl is a scripting language with strong support for text processing and directory file manipulation. PERL is a special-purpose language designed for text processing applications, especially those that require text search, extraction and text-base reporting. Its syntax is similar to that of C. Perl is widely used because of its strong support for string, string editing and regular expression. Perl has become a very popular bioinformatics programming language because of its

A SURVEY OF PROGRAMMING LANGUAGES RELEVANT TO BIOINFORMATICS

suitability for rapid prototyping – the ability to quickly write a working program. It has support for objects. Data structures like list and associative arrays (Perl hashes) are fairly easy to use, but have idiosyncratic syntax and semantic. Perl modules are extremely easy to install, and Perl modules for a wide variety of applications are readily on the CPAN archive.

Also, Perl is excellent for string manipulation, and bioinformatics deals mainly with large strings of genetic sequences and base pairs. Since, Perl has a highly developed capacity to detect patterns in data; it has become one of the most popular languages for biological data analysis.

Perl is often a first language for scientists, engineers, and system administrator whose primary training is not in software engineering or computer science. It is often used for small software tools or prototypes rather for larger, organized development projects.

Perl is mainly used with Unix/Linux based operating systems; however, you can install both for every windows based operating systems.

BioPerl

The BioPerl project is a coordinated effort to collect computational methods routinely used in bioinformatics into a set of standard CPAN-style, well documented, and freely available Perl modules.

(b) PYTHON

Python is a scripting Language which has become a popular language choice for researchers in the field of bioinformatics. Python has the support of advanced numerical capabilities through the Scientific Tools of Python (SciPy) project.

BioPython

BioPython project was created in 1999 and modeled after the BioPerl Project. Most of the work of the BioPython project was focused on creating parsers for biological data and designing a useful interface to represent sequences. One of the unique features of BioPython project is the use of a standard event-oriented parse design.

(c) PHP

PHP is a scripting language created by the BioPHP project.

BioPHP

The BioPHP project (formerly known as GenePHP) seeks to extend PHP language so that it can be used to develop bioinformatics applications. The main purpose of the BioPHP project is to encourage the use of PHP as a “glue” language to bind web-based bioinformatics applications and databases.

(d) Ruby

Ruby is a scripting language which has a native support for object-oriented programming with a simple but powerful syntax.

BioRuby

In addition to the BioRuby project, several educators have advocated the desirability of introducing undergraduate students to Ruby programming language.

4. ESSENCE OF PROGRAMMING IN BIOINFORMATICS

The majority of biological researchers are not programmers.

Biologists often feel that basic computing skills are all that they need to fulfill their everyday research tasks.

In many cases it was observed that "you can accomplish quite a bit using existing tools" (Tisdall, 2001).

"What happens when you want to do something a pre-existing tool doesn't do? What happens when you can't find a tool to accomplish a particular task, and you can't find someone to write it for you?" (Tisdall, 2001).

4.1 BENEFITS OF PROGRAMMING

Programming skills come to play when pre-existing tools cannot accomplish a task.

Programming skill is in strong demand.

Makes you more marketable as a biologist.

Saves time.

It is an easy skill to pick up.

5. PROGRAMMING LANGUAGE OF THE FUTURE

The programming language of the future in bioinformatics must have the capability of using fewer codes to get results efficiently as indicated by the scripting languages. It must be easy to learn, understand and implement for the desired application. Its library facilities must be free and open to all for research purposes.

The only chance biologists have of keeping up with the job of analyzing data is by developing libraries of reusable software tools.

6. CONCLUSION

This paper has presented a general overview of the various programming languages used in the biological sciences. This can help to understand the decisions involved in choosing a programming language for a specific application.

REFERENCES

1. Allen, B.T, JR. (1997), The Computer Science and Engineering Handbook CRC Press Florida.
2. Andrew, C. R. Perl Programming. Developing Key Tools for Bioinformatics.
3. Baas, B., (2002), Ruby in the cs Curriculum, *Journal of Computing Sciences in Colleges* 17, (5), 95-103.
4. Chapman, B., Chang, J., (2000), BioPython tools for computational biology, *ACM SIGBIO Newsletter*, 20, (2), 15-19.
5. Dalke A., (2004), Scientific Software LLC.
6. De Hoon, M. J. L., Chapman, B., Friedberg, I., (2003), Bioinformatics and Computational biology with biopython. *Genome Informatics*. 14, 298-299.
7. DeLano P. (2000), Reusable Objects for Bioinformatics. Biochem 218 Final Project - Fall 2000.

REVIEW OF PROGRAMMING LANGUAGES RELEVANT TO BIOINFORMATICS

Goto, N., Nakao, M. C., Kawashima, S., Katayama, T., Kanehisa, M., (2003), BioRuby: Open-source bioinformatics library, *Genome informatics*, 14, 629-630.

Gregorio, S. (2003), The BioPHP Project (formerly GenePHP), <genephp.sourceforge.net>

McGuffee, J. M., Programming Languages and the Biological Sciences, St. Edward's University. Austin, TX 78704.

Medinets D., Perl 5 by Example.

Oliphant, T., (2006), Scientific tool for python. www.scipy.org

Richard, F.G. and Behrouz, A.F. (1998), Data Structures. A Pseudocode Approach with C. PWS Publishing Company. Boston. MA

Timothy Schmidt, M. S., (2006), Programming for Bioinformatics. OK INBRE Bioinformatics Bulletin. University of Oklahoma Health Sciences Center.

Tisdall, J. D., (2001), Beginning Perl for Bioinformatics, O'Reilly & Associates. Biocomputingassociates.com/resume.html

Walnut C., Java by Example.