

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2002

An on-line acid-base titration applet in the generic tutorial system for the sciences project

Thomas Lee Gummo

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Computer Engineering Commons](#), and the [Inorganic Chemistry Commons](#)

Recommended Citation

Gummo, Thomas Lee, "An on-line acid-base titration applet in the generic tutorial system for the sciences project" (2002). *Theses Digitization Project*. 2045.

<https://scholarworks.lib.csusb.edu/etd-project/2045>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

AN ON-LINE ACID-BASE TITRATION APPLLET IN THE
GENERIC TUTORIAL SYSTEM FOR THE SCIENCES PROJECT

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Thomas Lee Gummo

March 2002



AN ON-LINE ACID-BASE TITRATION APPLET IN THE
GENERIC TUTORIAL SYSTEM FOR THE SCIENCES PROJECT

A Project
Presented to the
Faculty of
California State University,
San Bernardino

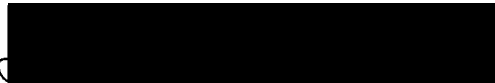
by
Thomas Lee Gummo

March 2002

Approved by:



Arturo Concepcion, Co-Chair,
Department of Computer Science

15 Mar 2002
Date


Kimberley Cousins, Co-Chair,
Department of Chemistry


George Georgiou, Computer Science


Kerstin Voigt, Computer Science

ABSTRACT

The purpose of this Master's Project was to develop an Acid-Base Titration Simulator. It was also to be part of the California State University - San Bernardino's GTSS, Generic Tutorial System for the Sciences, project.

The titration applet developed is interactive: as a button is depressed, an amount of titrant is dispensed, and a plot of the generated pH curve is displayed. The user will be able to compare their results with the computer-generated curves. The algorithm uses non-standard pH equations to accurately simulate these curves for 16 different acid-base systems, including several biochemistry applications. The first and second derivatives of the ideal pH curves can be displayed.

The main benefit is that students will be able to conduct titration experiments over the Internet without being in the laboratory and without costly equipment or dangerous chemicals. Instructors at the high school and college level can demonstrate the key chemical principles of titration. Learning will be enhanced, as the simulation can be repeated multiple times and much more rapidly than the laboratory experiment.

ACKNOWLEDGMENTS

The Office of Graduate Studies provided a grant for this project. The Grant was used to purchase *The JBuilder3.0* software package by Borland. This application builder tool was a great help in developing the software. The grant also enabled the purchase of more RAM memory for the computer used during development. *The JBuilder3.0* program required more memory than the computer possessed at the start of the project. The remaining grant money was used to produce the required printed copies of this project.

Thanks need to be given to the professors, who helped with this project. Dr. Arturo Concepcion, whose GTSS project was the genesis of this project. Dr. Kimberley Cousins' help with the chemical theories was invaluable. Dr. George Georgiou and Dr. Kerstin Voigt were two more instructors, who acted as advisors and helped a lot during the whole masters process.

Of course, without the help and support of my family, I would have never completed this project. To my wife, Pat, and my two daughters, Tina and Carin, I love you.

Remember 9-11-2001.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF TABLES	vii
LIST OF FIGURES	ix
CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION	
1.1 Introduction	1
1.2 Overall Description	8
1.3 Specific Requirements	18
CHAPTER TWO: NUMERICAL ANALYSIS OF TITRATION EQUATIONS	
2.1 Titration Equations	35
2.2 Graph Scales	45
CHAPTER THREE: DESIGN	
3.1 Architecture (Class Diagram)	48
3.2 Detailed Design (Pseudo-Code)	60
CHAPTER FOUR: OPERATING INSTRUCTIONS	
4.1 Operating Instructions	69
4.2 Hints for the Instructor	72
4.3 Hints for the Student	74
4.4 Testing	76
4.5 How to Input Your Own Acid System	76

CHAPTER FIVE: MAINTENANCE

5.1 Files	82
5.2 Directories	84
5.3 How to Compile	85

CHAPTER SIX: FUTURE DEVELOPMENTS AND CONCLUSIONS

6.1 Ideas for Future Developments	86
6.2 Derivative Engine	88
6.3 Conclusions	90

APPENDIX A: QUESTIONNAIRE	92
-------------------------------------	----

APPENDIX B: TABLES OF DISSOCIATION CONSTANTS	95
--	----

APPENDIX C: DERIVATIVEENGINE JAVA CODE	98
--	----

APPENDIX D: SAMPLE OF AN EXCEL SPREADSHEET	104
--	-----

BIBLIOGRAPHY	111
------------------------	-----

LIST OF TABLES

Table 1.	Minimum System Requirements	9
Table 2.	Error in Hydronium Ion Concentration	37
Table 3.	Error in pH to Two Decimal Places	38
Table 4.	GTSSChemApplet Class Diagram	49
Table 5.	SGFrame Class Diagram	50
Table 6.	pHMeterPanel Class Diagram	51
Table 7.	UserSupplied Class Diagram	52
Table 8.	Derivative Class Diagram	53
Table 9.	StudentMultiGraph Class Diagram	53
Table 10.	PlotMultiGraph Class Diagram	54
Table 11.	FirstMultiGraph Class Diagram	54
Table 12.	SecondMultiGraph Class Diagram	54
Table 13.	GraphPanel Class Diagram	55
Table 14.	GraphPanel Class Diagram - Continued	56
Table 15.	TitrationData Class Diagram	58
Table 16.	TitrationEngine Class Diagram	59
Table 17.	GTSSChemApplet Pseudo-Code	60
Table 18.	SGFrame Pseudo-Code	61
Table 19.	TitrationEngine Pseudo-Code	62
Table 20.	TitrationData Pseudo-Code	63
Table 21.	GraphPanel Pseudo-Code	63

Table 22.	pHMeterPanel Pseudo-Code	64
Table 23.	StudentMultiGraph Pseudo-Code	65
Table 24.	PlotpHMultiGraph Pseudo-Code	65
Table 25.	FirstMultiGraph Pseudo-Code	66
Table 26.	SecondMultiGraph Pseudo-Code	67
Table 27.	Derivative Pseudo-Code	68
Table 28.	Java Source Files	83
Table 29.	HTML Files	83
Table 30.	Compiler Created Class Files	84

LIST OF FIGURES

Figure 1.	Generic Tutorial System for the Science's Structure with Chemistry Objects Added	3
Figure 2.	Deployment Diagram	12
Figure 3.	Use Case Diagram	15
Figure 4.	Generic Tutorial System for the Science's Home Page	19
Figure 5.	Applet Selection Page	20
Figure 6.	The Applet Frame	21
Figure 7.	The Select Menu	22
Figure 8.	Monoprotic Acid with Strong Base Menu	22
Figure 9.	Diprotic Acid with Strong Base Menu	23
Figure 10.	Triprotic Acid with Strong Base Menu	24
Figure 11.	Strong Base with Strong Acid Menu	24
Figure 12.	User Supplied Menu	25
Figure 13.	The Dispense Buttons	25
Figure 14.	Graph Menu	27
Figure 15.	User and Computer pH Curves	28
Figure 16.	User and Computer First Derivative Curves	29
Figure 17.	User and Computer Second Derivative Curves	30
Figure 18.	The Repeat Titration Menu	31
Figure 19.	Class Diagram Overview	48

CHAPTER ONE
SOFTWARE REQUIREMENTS
SPECIFICATION

1.1 Introduction

1.1.1 Purpose

The goal of this Master's Project is to promote and facilitate the use of Java-based technologies in the development of teaching materials for chemistry. This project built on the GTSS (Generic Tutorial System for the Sciences) Project and promotes active learning by providing educators with the tools to develop their own interactive teaching materials. The objective was to develop a chemical application, which is interactive, easily distributed, flexible, intuitive and easy to use, and easy for instructors to implement.

Initially, college and high school level instructors will be able to use this application to demonstrate chemical principles to their students. Instructors with a basic working knowledge of Java will be able to create their own applications by building on and using of engines and applets written for this project.

1.1.2 Scope

The GTSS project originally focused on developing Java materials for computer science, mathematics, and physics, the area of expertise of Dr. Arturo Concepcion, Professor of Computer Science, Dr. Javier Torner, Professor of Physics, and Dr. Charles Stanton, Professor of Mathematics (9). They received a grant to develop GTSS products for these three discipline areas. The materials developed are designed to be usable by other science instructors in order to build demonstrations and tutorials. The object-oriented approach was used to build the three-layered structure of GTSS: Core Objects, Subject Engine Objects, and Application Objects. The Internet address or URL for the current version of the GTSS Project is:

<http://gtss.ais.csusb.edu/GTSSProject/index1.html>

The GTSS structure is shown in Figure 1. The core objects are in the center and are used by all disciplines. The next layer has the engines written for specific disciplines but may be used by the others. For example, a statistics engine written for mathematics might be used by

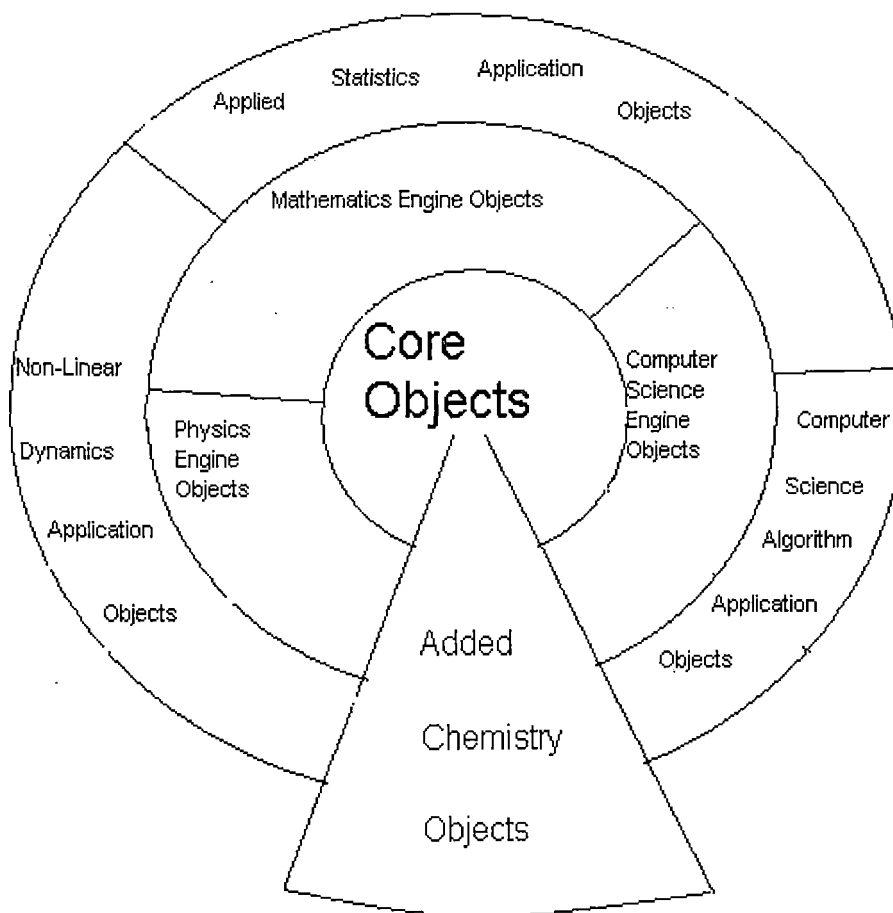


Figure 1. Generic Tutorial System for the Science's Structure with Chemistry Objects Added

any discipline using any of the objects, engines, or applications written for GTSS. In fact, it is hoped there will be crossover within the disciplines to eliminate duplication of effort.

This project, An On-Line Acid-Base Titration Applet in the GTSS Project, includes a chemical applet for GTSS. This is an interactive tutorial and demonstration system

that enhances the students learning of several chemical principles involving acid-base titrations. This chemical applet provides instructors of chemistry with tools by which they can create, edit, or modify tutorials to suit their instructional needs. With the help of Dr. Kimberley Cousins, an Associate Professor of Chemistry at CSUSB, this Master's Project added a chemical application to the GTSS system.

Java based programs can provide animated demonstrations and allow students to interact with the material. Compared to other programming languages, Java and the Internet is clearly the medium of choice for easy distribution. Programs written on one campus can be put on a server and be available for immediate use at other campuses. Since Java is object-oriented, programs will be very modular and readily adapted to new uses. As a teaching tool, a well-written application will allow the instructor to focus attention on the material while minimizing the need to know the Java computer language. While this application will contain animated and graphic objects, it will be impossible to produce movie like special effects or photographic quality images. The

objective was to produce an application, which clearly and accurately demonstrate the principles of the chemical processes involved. One chemical application and the needed support programs have been developed for this project. This chemical applet demonstrates acid-base titrations by allowing the students to practice with sixteen different acids and bases.

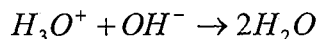
The main benefit is that the students are able to conduct experiments without being in the laboratory and without costly equipment and chemicals. In a real laboratory setting, it is likely the number of trials would be limited. Learning will be enhanced as the simulation can be repeated multiple times and much more rapidly than the experiment itself. Also, a major advantage is safety, that is, the student is not handling any dangerous chemicals, such as acids and bases. Students are able to work on their own without the supervision required in the laboratory setting. Finally, the student will be able to compare their curves to the actual curves computed by the applet or obtained from experiments to ensure they understand the equations involved.

1.1.3 Definitions, Acronyms, And Abbreviations

JAVA APPLETS. These are programs written in Java that are downloaded from the World Wide Web by a Web browser and run inside an HTML Web page. A Java-enabled browser such as Netscape Navigator or Microsoft's Internet Explorer is required to run these applets.

JDK. Java Developer's Kit is a Java development environment. There are several available at this time and more are being developed: Java Workshop, Semantec's Visual Café for Java, and Microsoft's Visual J++. Sun supplies a JDK with Java 1.3 and includes tools for compiling and testing Java applets and applications.

TITRATION. Neutralization curves are the key to this chemical demonstration. Neutralization is a process in which hydronium ions unite with hydroxyl ions to form water:



There are several cases which will be investigated: a strong acid with a strong base, a strong base with a strong acid, a weak acid with a strong base, and the biochemical problem of amino acid end point equivalents (an area that is under serviced by tutorial software).

A typical wet chemical titration involves adding in a small amounts the titrant to the solution being tested, the analyte, while recording the pH changes until the endpoint is passed.

CORE OBJECTS. This is a set of primitive reusable objects that are used to build graphics user interface (GUI), such as frames, test canvasses, graphics canvasses, buttons, menus, etc. These are the basic building blocks of all tutorial systems. Although there are already objects defined and implemented in Java, they still need to be customized and refined for use in displaying specific windows and graphics in GTSS.

SUBJECT ENGINE OBJECTS. The engine is built on top of the core objects and through inheritance and polymorphism. These engines will support the generation of windows and graphics for the specified subject.

APPLICATION OBJECTS. These objects are implemented on top of their corresponding subject engine objects and through inheritance and polymorphism. They will support the generation of windows and graphics for the particular topic in the subject.

UTILITY ENGINES OBJECTS. These object are used to aid in data entry, keyboard functions, VCR controls, and include several engines whose functions don't align with the other groups of objects.

VCR CONTROLS. This panel includes several buttons that are used to control the functions of the applet. This panel is called "VCR CONTROL" because the buttons have the look and function of the buttons found on most VCR machines.

1.2 Overall Description

1.2.1 Project Perspective

1.2.1.1 System Interfaces. GTSS is installed on a server computer in the Department of Computer Science at CSUSB. Users are able to access the Web pages and Titration Applet with Netscape Navigator, Internet Explorer, or any other Java compatible Web browser.

1.2.1.2 User Interfaces. The applets require a display area in which to operate. The top-level window is called a frame. Inside the frame are panels in which to display graphics and the graphical user interfaces used by the applet. The frames have the look and feel of any of MicroSoft's windows; so will be easily used by most

students and teachers. The panels are used to display the graphical demonstrations, plot the data, control the functions and rate of the experiments, and input and export data. The User Interface of this program consists of one window in which the program displays one frame. The execution of the program is completely mouse or pointing device based. All selections from the menus and activation of the buttons are accomplished with the mouse left button.

1.2.1.3 Hardware Interfaces. This project was written and tested on an IBM Aptiva Computer with a Pentium II processor. This program may run on slower machines but no testing was conducted. The system specifications are:

Table 1. Minimum System Requirements

Processor Type	Intel MMX, Pentium II
Processor Speed	166 MHz
System memory	32 MB
Video Memory	2 MB
External Cache Memory	256 KB
Sound Card	Crystal PnP Audio System CODEC
Modem	LT Win Modem
BIOS Version	BVAUS4E
BIOS Date	02/20/97

1.2.1.4 Software Interface. This project was designed to run over the Internet. By writing it in Java, the code will run on any computer independent of type and operating

system. As long as the user's computer has an Internet connection, appropriate browser, and correct JAVA plug-in, there will be no other limits on the users system.

Java is one of the newest programming languages (4). It has been developed by Sun MicroSystem Inc. Sun's object was to provide a common system development kit (SKD) that would run across many different platforms. The Internet is growing geometrically and unfortunately has become one of the major battlefields in the computer world. Over the Internet, there are various systems providing services. Each platform has its own system development environment. Even in the Unix environment, different vendors have their specially designed Unix systems. They are not compatible with each other. A common language was not a major issue in the past because the operating environment was not complex. However, with the growth of the Internet, a common SDK becomes more important. For this reason, Java was developed.

The developers of Internet browsers seem to be lagging behind the developers of Java. Therefore, a problem exists. If you use the latest version of Java, the current Internet browsers are unable to run the Java applets. If

you use the version of Java that the browsers can handle, you are forced to use obsolete and out of date versions of Java. The decision was made to use the JDK 1.3 version of the Java language. It is compatible with the current versions of the popular browsers.

1.2.1.5 Communications Interfaces. Figure 2, the Deployment Diagram, shows the relationship between the GTSS server, the network user and the Internet user (2). To access this applet, the user needs an Internet connection to the GTSS server here at CSUSB. The web address is:

<http://gtss.ais.csusb.edu/GTSSProject/index1.html>.

The user will also need a JAVA enabled browser, such as, Netscape or Internet Explorer versions 5.0 or higher.

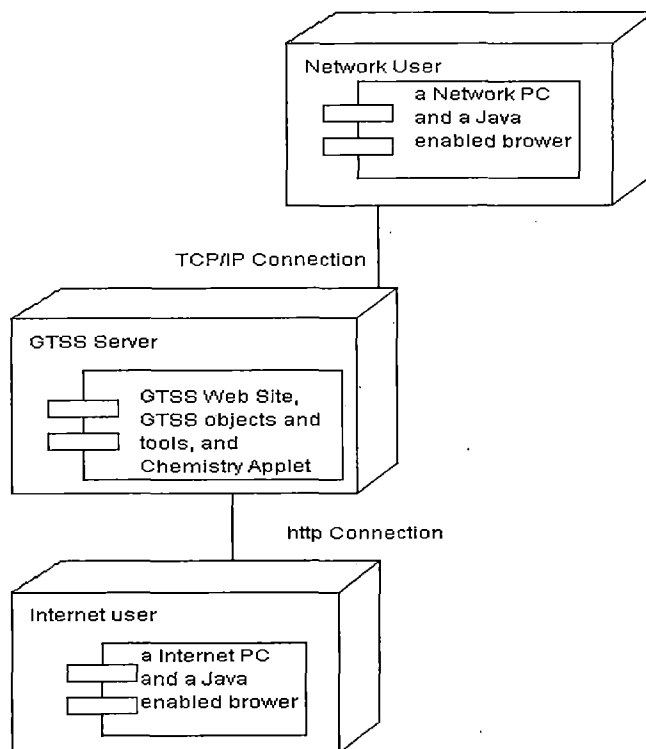


Figure 2. Deployment Diagram

1.2.1.6 Memory Constraints. The minimum RAM, Random Access Memory, tested, for this program, was 64 Megabytes. However, any computer with enough memory to effectively use the Internet should be able to use this program.

1.2.1.7 Operations. The applet will be accessed on a server on the World Wide Web. It will remain active as long as the browser containing the applet is running. The program is interactive and needs the user to interface with it. Nothing happens unless the user activates a function. Currently, the program does not use a database. Therefore,

there are no backup or recovery operations required. Once running, no one has been able to crash the applet. However, if a problem should occur, simply using the browser refresh button will reset the applet.

1.2.1.8 Site Adaptation Requirements. This project is being written and tested with Microsoft Windows 98 and Internet Explorer 5.0. Earlier Versions of Windows and Internet Explorer will not be tested.

However, the browser needs to have the Java 1.3 plug-in installed. By downloading and installing this plug-in, the required version of Java Virtual Machine is added to the browser. A user without this plug-in, who attempts to uses this applet, will be advised of this requirement and given the links to download it.

1.2.2 Project Functions

This applet has one main function. It is used to demonstrate the chemical principles of acid-base titrations. This concept can be expanded for students and instructors. This applet allows a student to have open-ended study sessions, that is, no time limits. Students are allowed to proceed at their own pace. Unlike in the laboratory, there are unlimited repetitions of the

titrations available and where the student can choose the same or random endpoints. They can repeat using the same endpoint until they are happy with the results or understand the concept. They can practice titrations with random endpoints. The applet is being conducted in a non-laboratory environment, which is safer, more cost effective, and faster. Figure 3, Use Case Diagram, shows the relationship of the student/user and instructor to the applet.

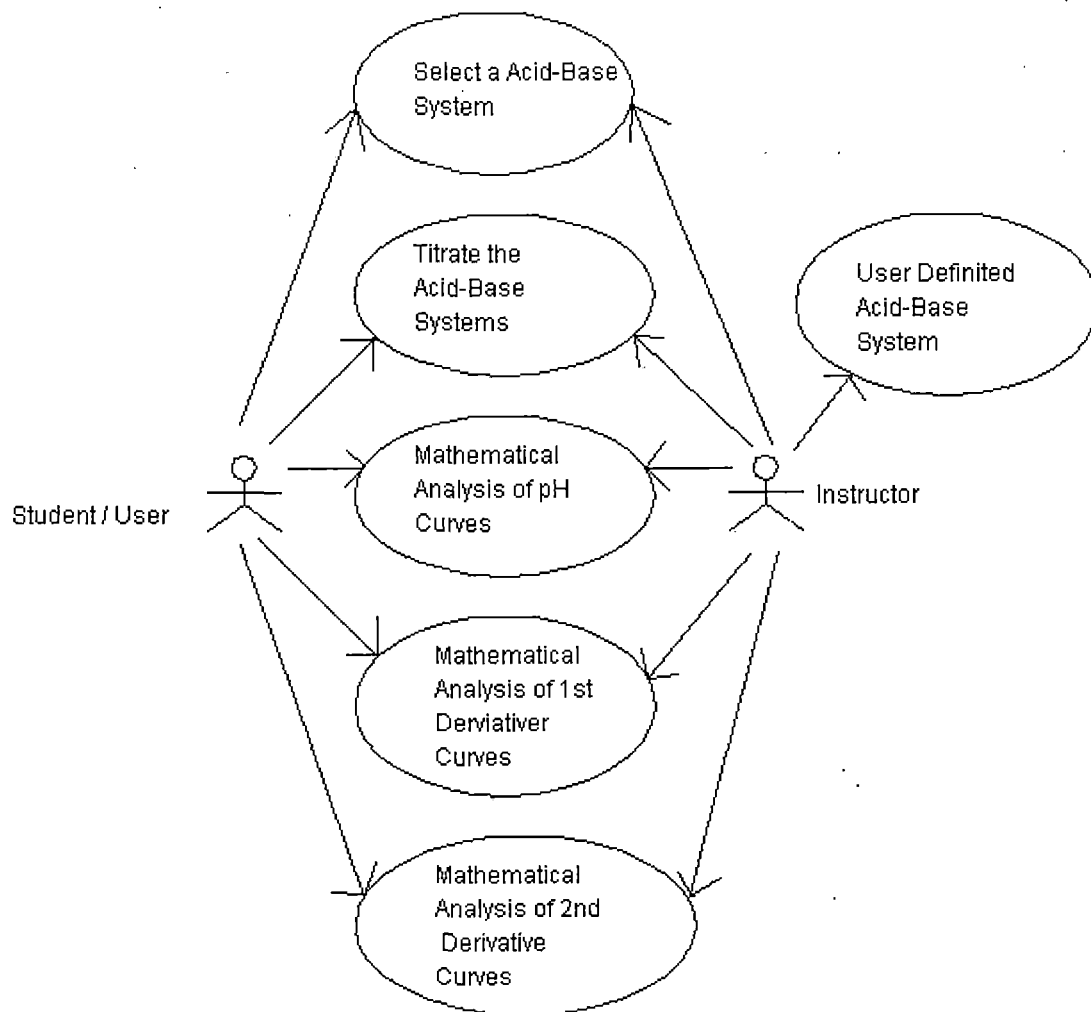


Figure 3. Use Case Diagram

It can be assigned as a pre-laboratory exercise or used as guided tutorials by instructors. For example, it can be used as a classroom exercise to calculate acid concentrations or solution pHs. Another classroom exercise could be to "discover" data irregularities, such as, non-resolution of multiple endpoints.

Students will be able to practice the mathematic methods, including the first and second derivatives of the pH curves, used to solve acid-base problems. Instructors can demonstrate titration methods, numerical analysis on data, characteristics of the pH curve, and the uses of the first and second derivative curves. The applet illustrates acid-base titration chemistry to include strong acids versus strong bases, weak acids versus strong bases, monoprotic systems, and polyprotic systems.

1.2.3 User Characteristics

As stated before, the audience for this project is college chemistry professors, high school chemistry instructors, and their students. The typical high school student, who takes chemistry, is college bound and also takes biology and physics. They will have taken algebra, geometry, and possibly trigonometry. The college level chemistry students can be broken down into two groups: non-science majors and science and engineering majors. The grade level of the material in the project is geared for the higher levels of high school students and the level of non-science students in college. However, this is not to say the science majors will not be able to benefit too.

While, the chemical concepts being considered may be too simple for the science and engineering students, it will still be a nice review of titration principles.

The instructors who will use this project will fall into two camps: Java literate and those without Java skills. Even without Java skills, instructors will be able to use the applet as written to supplement their instruction. The applet is written in such a manner that no knowledge of Java is required. If the instructor and student can open and run basic Internet browsers, they will be able to run the applets. The Java literate instructors will be able to modify applets to meet their own needs and build on them to write their own applets.

1.2.4 Constraints

Currently, there are no hardware or software constraints. Anybody, who would like to modify or create new GTSS applications, will need to know how to program in Java.

1.2.5 Assumptions and Dependencies

None.

1.2.6 Apportioning of Requirements

In the context of this project, there are no elements that are being delayed until future versions are developed. The program code has been finalized at this time. However, there are several items that could be improved by another computer science or chemistry student in the future.

1.3 Specific Requirements

1.3.1 External Interfaces

The applet can be found by going to the GTSS server and loading GTSS home page. There, the user will find a link to the applets available for physics, math, computer science and chemistry.

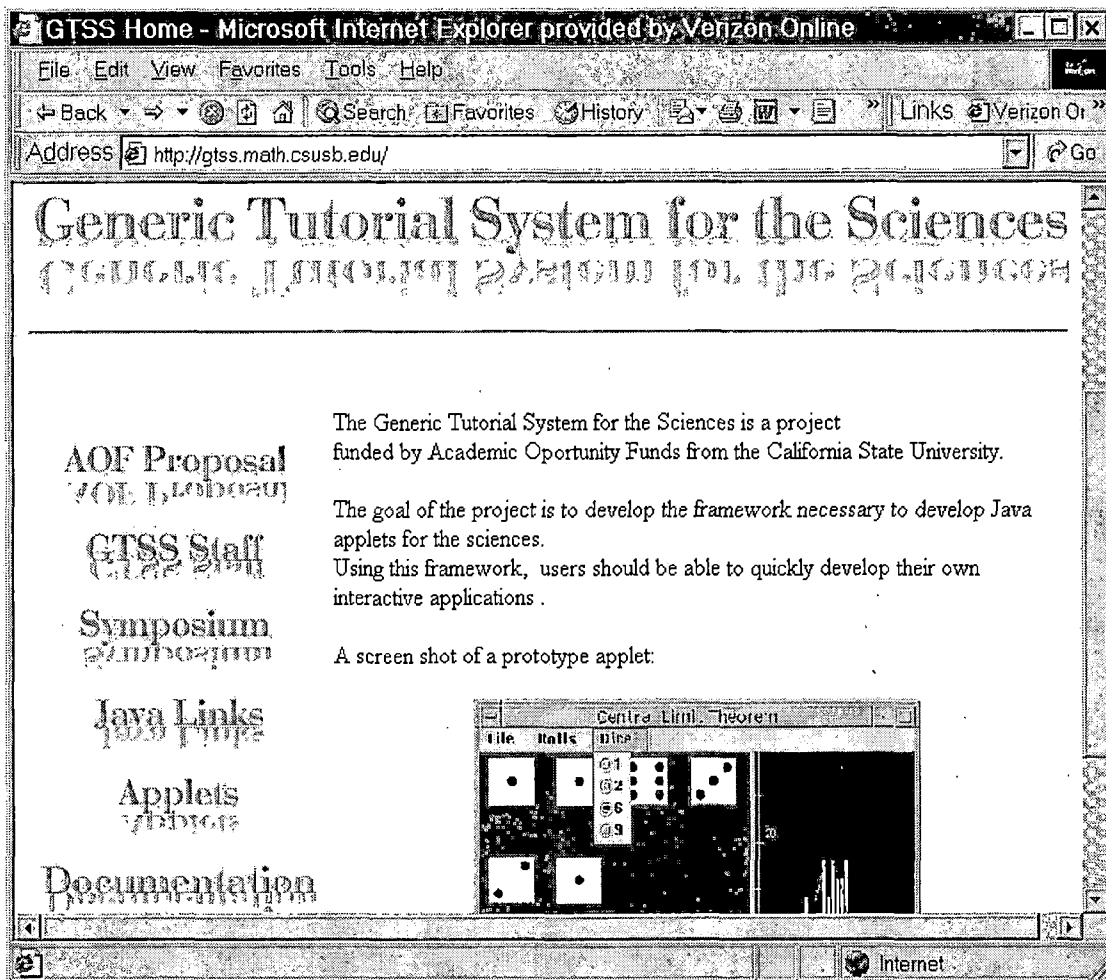


Figure 4. Generic Tutorial System for the Science's Home Page

On the applet page, the user selects the chemistry link where the acid-base titration demonstration applet and the accompanying html pages will be found.

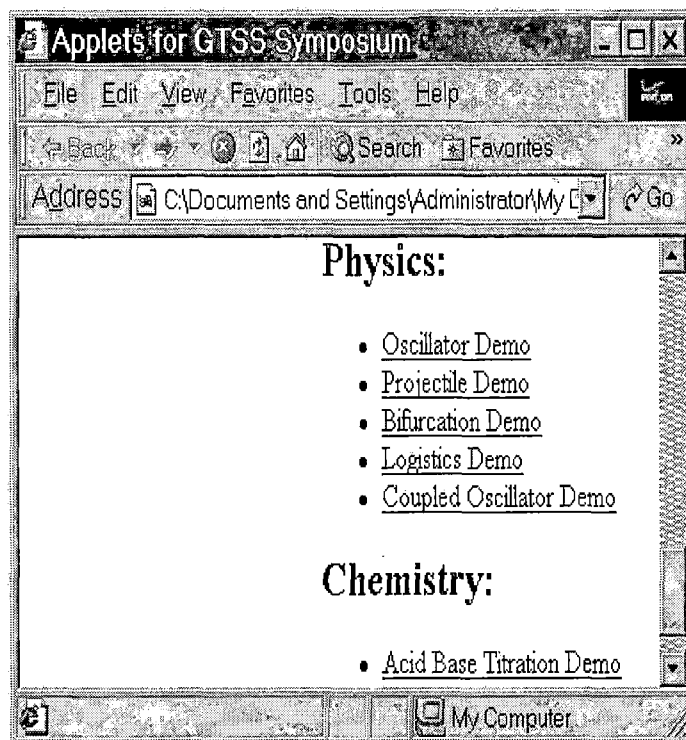


Figure 5. Applet Selection Page

The Web pages will provide instruction on how to use the applet for both the student and instructor. The text contents of the web pages are included in the appendix.

This application will use a frame, Figure 6, and is divided into two panels and a menu bar. The main part of the frame is divided into two unequal panels, split vertically. The left panel is the "Gummo Cousins Titration Machine" and includes a set of buttons, which regulate the titration plus a set of windows to monitor the progress. It also displays the current pH and the "volume of titrant"

added during the titration. The right panel displays the generated pH curves, the first derivative curves, or the second derivative curves.

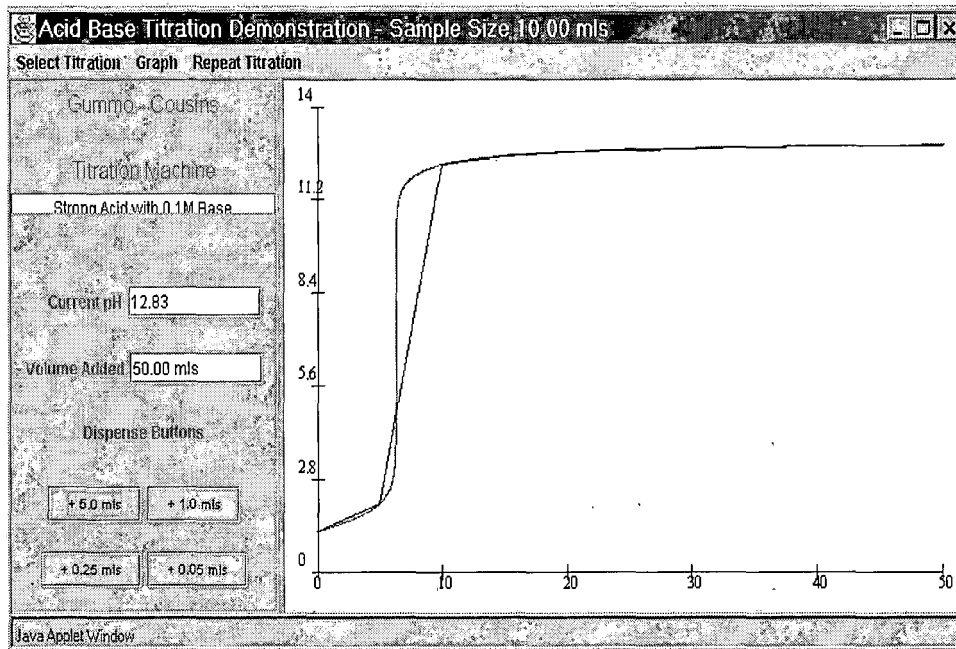


Figure 6. The Applet Frame

The first step is for the user to select an acid-base system. Clicking the mouse button on the "Select Titration" menu will cause a drop down menu to appear.

Select Titration	Graph	Repeat Titration
Mono Protic Acid with Strong Base	▶	
DiProtic Acid with Strong Base	▶	
TriProtic Acid with Strong Base	▶	
Strong Base with Strong Acid	▶	
User Supplied	▶	

Figure 7. The Select Menu

This menu allows the user to select from five more menus:

- 1) Monoprotic Acid with Strong Base,
- 2) Diprotic Acid with Strong Base,
- 3) Triprotic Acid with Strong Base,
- 4) Strong Base with Strong Acid,
- and 5) User Supplied.

Select Titration	Graph	Repeat Titration
Mono Protic Acid with Strong Base	▶	1. Demo
DiProtic Acid with Strong Base	▶	2. HCl
TriProtic Acid with Strong Base	▶	3. Nitrous
Strong Base with Strong Acid	▶	4. Hypochlorous
User Supplied	▶	5. Acetic
		6. Ethylamine

Figure 8. Monoprotic Acid with Strong Base Menu

The first menu is the "Monoprotic Acid with Strong Base." Moving the mouse cursor onto this menu drops down another menu with the six selections available. The first is the Demo mode. The endpoint is fixed and can't be changed. This is true about all the Demos in the other

menus. This allows the instructor to demonstrate titration techniques, while knowing the characteristics of the pH curve that will be produced. Selection of any of the other systems will create a system with a randomly generated endpoint. This selection can be made by either clicking the mouse button or by using the enter key. The other five selections can be seen in figure 8.

Select Titration	Graph	Repeat Titration
Mono Protic Acid with Strong Base ▶		14
DiProtic Acid with Strong Base ▶		1. Demo
TriProtic Acid with Strong Base ▶		2. Tartaric
Strong Base with Strong Acid ▶		3. Carbonic
User Supplied ▶		4. Alanine
		5. Adenine(+1)
		6. L-Leucine

Figure 9. Diprotic Acid with Strong Base Menu

Moving the mouse cursor farther down will highlight the "Diprotic Acid with Strong Base" menu and cause it's drop down menu to appear. Figure 9 displays the six possible selections.

Select Titration	Graph	Repeat Titration
Mono Protic Acid with Strong Base ▶		14
DiProtic Acid with Strong Base ▶		
TriProtic Acid with Strong Base ▶		1. Demo
Strong Base with Strong Acid ▶		2. Glutamic
User Supplied ▶		3. Phosphoric
		4. Arginine
		5. Tetracycline(+1)

Figure 10. Triprotic Acid with Strong Base Menu

Moving the mouse cursor down to the next option will highlight the "TriProtic Acid with Strong Base" menu. It has five selections to choose from.

Select Titration	Graph	Repeat Titration
Mono Protic Acid with Strong Base ▶		14
DiProtic Acid with Strong Base ▶		
TriProtic Acid with Strong Base ▶		
Strong Base with Strong Acid ▶		1. Demo
User Supplied ▶		2. NaOH

Figure 11. Strong Base with Strong Acid Menu

Figure 11 highlights the strong base with strong acid selection and its two possible systems. This time a base, NaOH, will be titrated with an acid, HCl.

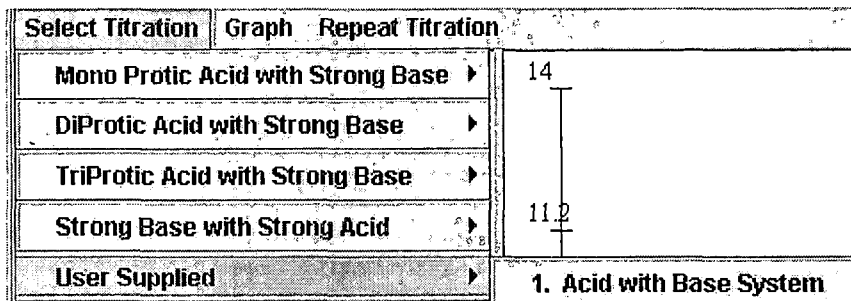


Figure 12. User Supplied Menu

Once the selection of the acid-base system has been accomplished, the menus will disappear and the applet enters the titration phase. The titrant is added to the starting sample solution of 10.0 mLs by clicking on one of the "Dispense Buttons."

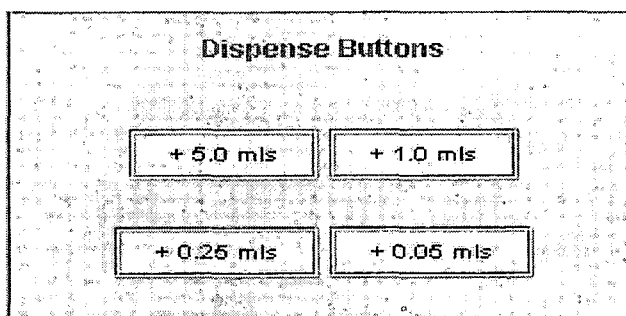


Figure 13. The Dispense Buttons

The amount of titrant that can be added each time a button is clicked with the mouse button is limited to the quantities listed on the buttons: 5.0 mLs, 1.0 mLs, 0.25 mLs, and 0.05 mLs (the size of one drop from the buret or

approximately the size of the smallest amount a student would be able to dispense manually from a buret). Each time a dispense button is clicked; the amount in the volume added window is updated; the current pH window is updated; and the graph of the pH curve has a new point displayed. The titration phase continues until a maximum of 50.0 mls has been added. At that point, clicking on any of dispense buttons will have no effect.

The titration phase was designed to prevent the student or user from "looking ahead" and seeing the answer. Therefore, once the titration phase is ended by any means, it can't be restarted. For example, the user can't select the display of the computer generated pH curve and then go back to continue the titration. The titration phase is ended by any of the following actions: reaching 50.0 mLs of titrant added, selecting from the "Graph" Menu any of the computer generated graphs, or selecting a new endpoint from the "Repeat Titration" menu.

Once the titration phase is complete, the user may want to check on the quality of their titration. Selecting one of the items from the "Graph" menu starts the analysis phase: pH Plot, first Derivative, or second Derivative.

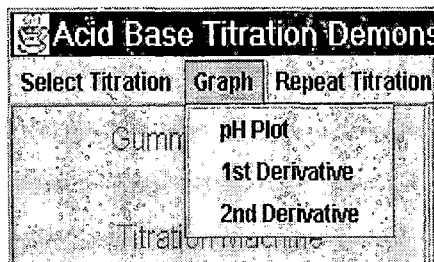


Figure 14. Graph Menu

Selecting "pH Plot" will display the student's curve and the curve produced by the computer. The student's curve is plotted in black and the computer's curve is in red. If any of the points of the computer's curve are located at the same coordinates of the student's, only the black line will be seen (the student's line is drawn last). Figure 15 shows a typical student titration along with the computer-generated pH curve.

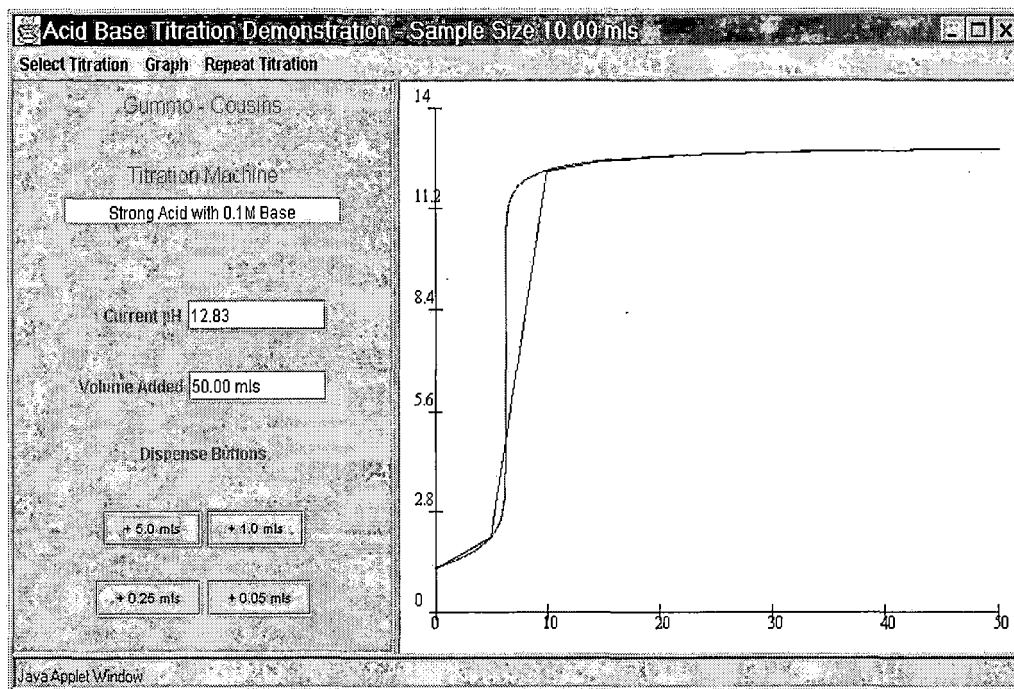


Figure 15. User and Computer pH Curves

When the "First Derivative" is selected, the computer mathematically computes the first derivative curves from the data stored in the pH curves. The scale on the vertical axis is changed to 0 to 3 units. The dimensions of the units are not important and will be covered more in Chapter Two. The maximum was fixed in a manner to show the user's results best. In fact, the peak of the computer's curve was found to exceed 1,000,000 units at times. If the vertical scale were adjusted to show the top of the computer's curve, the user's curve would appear as a straight line at the bottom of the graph. The location of

the peak on the volume axis is the endpoint. (Note: if any of the pH curves fits on the graph, they will be displayed too.)

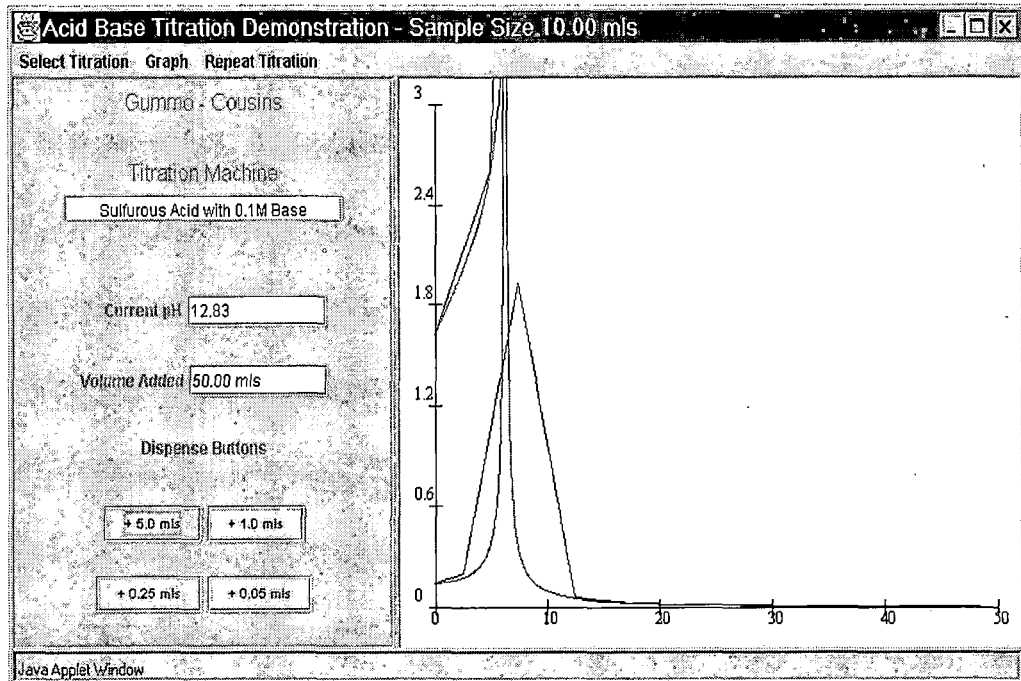


Figure 16. User and Computer First Derivative Curves

The last option on the "Graph" menu is "2nd Derivative." Again, the computer takes the data from the pH curves and mathematically calculates the derivative twice. The results are then displayed. The vertical scale is changed again to maximize the results. This time it is where the curve crosses the volume axis from the positive side that shows the endpoint.

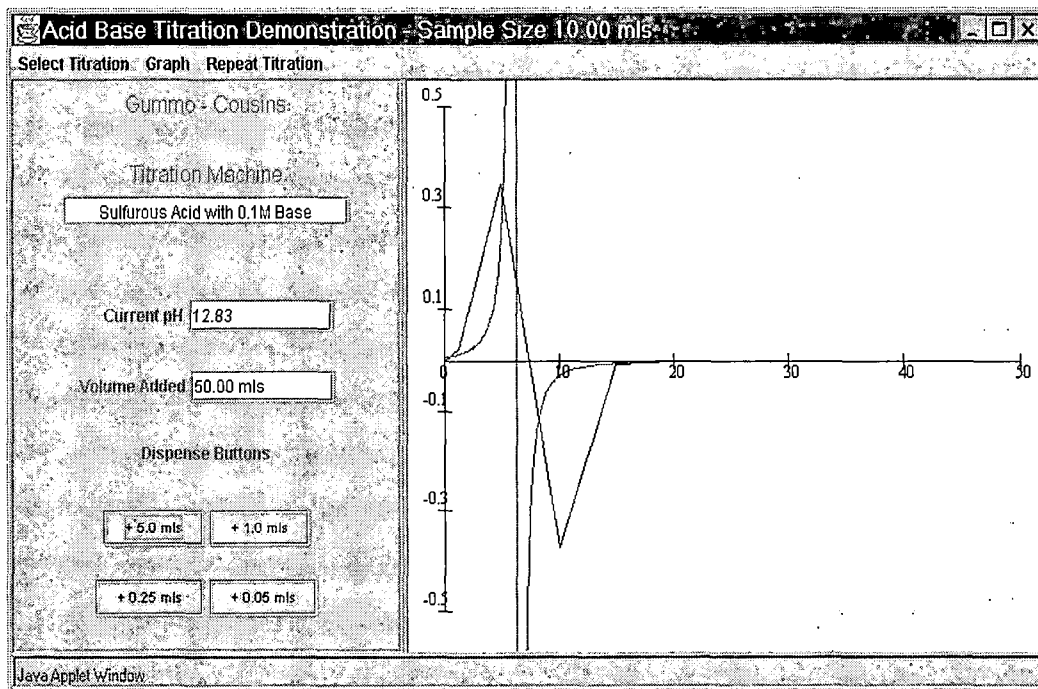


Figure 17. User and Computer Second Derivative Curves

During this analysis phase, the user can go back and forth between any of the "Graph" menu's options. However, as stated before, it is impossible to continue a titration at this point that was not finished.

Once the analysis phase is complete, the user has several options. The first is to use the same acid-base system again. The "Repeat Titration" menu, as shown in Figure 18, is used to select the "Same Equivalence Point" or "New Equivalence Point." The "Same" menu starts the

titration phase with the same endpoint as before. The "New" menu creates a random endpoint for the titration phase.

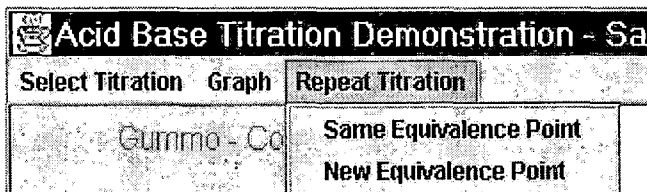


Figure 18. The Repeat Titration Menu

1.3.2 Functions

One of the functions that needed to be defined is how the program accepts and processes its inputs and outputs. The only input, once the applet is downloaded and running on the user's browser, is the mouse and the left mouse button. Therefore, there is no requirement to check the validity of the inputs. Abnormal situations like data overflows, communication failures, error handling, and recovery are not encountered. Unlike some programs, the exact sequence of operations is not important. There are no outputs generated by this program.

1.3.3 Performance Requirements

This applet is designed to run on one computer at a time. Once downloaded to the user's computer, it is a

stand-alone program. The number of simultaneous users supported is limited by the bandwidth of the GTSS server and is beyond the scope of this project.

1.3.4 Logical Database Requirements

This applet has no database requirement.

1.3.5 Design Constraints

In the design phase, two standards were used. The first is the Unified Modeling Language (UML). This modeling language is the graphical notation used to express designs. It was used to develop the Deployment Diagram, the Use Case Diagram, and the Class Diagrams.

The second is Object-Oriented software engineering methods. It uses five main methods: establish core requirements, develop a model of behavior, create the architecture, evolve the implementation, and maintenance. It also has four micro processes: identify the classes and objects needed, identify the semantics of these classes, identify the relationships among these classes, and specify the interfaces required.

1.3.6 Software System Attributes

1.3.6.1 Reliability. The reliability of this applet was verified through extensive testing of all features. The applet performed consistently throughout the testing phase.

In Phase one, the results from the equations used to calculate pH for the different acid-base systems were compared to known solutions in chemistry textbooks (1, 3, 8). The conclusion was the accuracy was greater than 0.01 units of pH. This is well within the accuracy required.

In Phase two, all the menus and buttons, the applet's Graphic User Interface (GUI), were tested. Each selection from the menus performed its' function as designed. There was no way found to crash or lock up this applet.

In the final phase of testing, the calculation of pH was added to the GUI to complete the applet. Again, all functions and calculations performed without any failures.

1.3.6.2 Availability. This applet is available anytime the GTSS server is running. Presently, the server runs 24 hours a day and seven days a week.

1.3.6.3 Security. There is no security, such as passwords, required by this applet. The idea is for open

access to the titration applet. It is the server's responsibility to provide the security needed by the GTSS programs. The applet does not have access to the files on the server other than the ones needed for the application to run.

1.3.6.4 Maintainability. The applet runs in the Java 1.3 environment. If the Java Runtime Environment used by the GTSS server is updated, then the Java code for this project will need to be recompiled and reinstalled. There are no other maintenance requirements for this applet.

1.3.6.5 Portability. Java as a programming language was designed to run on most platform types. For the Internet user, the applet will port to any computer with the proper Java enabled browser. For the user, who downloads the code, they will need a copy of the Java Developer's Kit (JDK).

CHAPTER TWO
NUMERICAL ANALYSIS OF
TITRATION EQUATIONS

2.1 Titration Equations

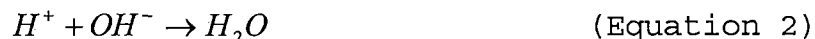
Several interesting problems developed during this project. First, it was assumed the acid-base pH equations commonly found in chemistry texts would work with no problem. However, this was not the case. A brief discussion of some chemical principles is required to be able to illustrate what had to be overcome. The main goal of this project was to develop a teaching tool, which deals with acid-base titrations. One of the sub-goals was to make it as accurate as possible with no artificial limitations imposed. Therefore, the simulator and the data produced needed to provide several things. First, it must react in a manner like the real world problem, such as, strong base versus weak acid, or polyprotic systems. Second, the algorithm needed to provide smooth, continuous, and chemically accurate curves which would allow the mathematical analysis to work properly.

A monoprotic strong acid being titrated with a monoprotic strong base was the first case to be investigated. The generic equation is:



HA is the acid; BOH is the base; and BA is a salt.

The equation can be reduced to just the ions which react with each other. The two normal forms are:



H⁺ is a Hydrogen ion; H₃O⁺ is the Hydronium ion; OH⁻ is the Hydroxide ion; and H₂O is water.

The two equations above are the same from a chemical point of view. H⁺ and H₃O⁺ designate the same acidic unit and will be used interchangeably throughout this project. In the strong acid-strong base system, there are no buffer systems or dissociation factors to deal with and pH is easily determined from the concentration of the H₃O⁺ ion, or [H₃O⁺]. The pH range in a water solution is limited to 0.00 to 14.00, with the neutral point of 7.00. At the start of a typical titration of this type, the acid concentration determines the pH. At the endpoint or neutralization point, the pH of water is the overriding factor. Finally,

after the endpoint, the concentration of the basic solution determines the pH. Chemistry texts (1, 3, 8) give the following equations:

$$\text{Before endpoint, pH} = -\log[H_3O^+] \quad (\text{Equation 4})$$

$$\text{At endpoint, pH} = 7.00$$

$$\text{After endpoint, pH} = 14 + \log[OH^-] \quad (\text{Equation 5})$$

However, buried in the fine print is the statement that the H_3O^+ ions provided by water can be ignored. In fact, this is not true, as Table 1 and Table 2 demonstrate. As can be clearly seen, the error in calculated pH increases as you near the endpoint of 7.00. For pHs between 6.00 and 7.00, Equation 1 produces erroneous results. The problem mirrors itself on the other side of the endpoint as well between pHs of 7.00 to 8.00.

Table 2. Error in Hydronium Ion Concentration

Calculated PH	$[H_3O^+]$ from Acid	$[H_3O^+]$ from Water	$[H_3O^+]$ total	Percent error in $[H_3O^+]$
1.00	0.1000000	0.0000001	0.1000001	0.0001
2.00	0.0100000	0.0000001	0.0100001	0.001
3.00	0.0010000	0.0000001	0.0010001	0.01
4.00	0.0001000	0.0000001	0.0001001	0.10
5.00	0.0000100	0.0000001	0.0000101	1.00
6.00	0.0000010	0.0000001	0.0000011	10.00
7.00	0.0000001	0.0000001	0.0000002	100.00

Table 3. Error in pH to Two Decimal Places

Calculated pH	Actual pH	Percent pH error
1.00	1.00	0.00
2.00	2.00	0.00
3.00	3.00	0.00
4.00	4.00	0.00
5.00	5.00	0.00
6.00	5.96	0.67
7.00	6.70	4.29

While the errors in pH look acceptable, clearly the errors in concentration are large enough to cause problems. An examination of several texts showed that they carefully selected the conditions of their examples so the error was minimized. In fact, it is possible to limit the concentrations and minimum drop size so that this problem can be avoided but this defeats our goal of having "no limitations." In addition, the errors cause the graphing program to produce very unsatisfactory results. The limitations of these equations caused some artificial factors to be introduced, which were unwanted but necessary. By careful selection of starting concentrations and limiting drop size, the areas of pH = 6.00 to pH = 6.99 and pH = 7.01 to pH = 8.00 could be avoided. Once the

limitations were implemented, the model appeared to work correctly and produced satisfactory results.

The next case examined was a weak monoprotic acid with a strong monoprotic base; the generic equation is:



K_a is the dissociation constant.

Unlike the strong acid, the weak acid only partially dissociates in the water solution. The dissociation constant, K_a , is determined by the equation:

$$K_a = \frac{[\text{H}^+][\text{A}^-]}{[\text{HA}]} \quad (\text{Equation 7})$$

$[\text{H}^+]$ is the Hydrogen ion concentration.

$[\text{A}^-]$ is the anion concentration.

$[\text{HA}]$ is the concentration of the un-dissociated acid.

Now the pH curve has to be broken down into four regions: 1) Before Base is Added, 2) Before the Equivalence Point, 3) At the Equivalence Point, and 4) After the Equivalence Point. Again, the equations were readily available in the chemistry texts examined (1, 3, 8). Before the starting of the titration (the first region),

the equation to determine pH is determined by the initial concentration of the acid, [HA], and it's dissociation constant, K_a :

$$\frac{[H^+][A^-]}{([HA]-[H^+])} = K_a \Rightarrow [H^+] \Rightarrow pH \quad (\text{Equation 8})$$

The left side of the Equation 8 allows the calculation of the $[H^+]$ and then the pH. Once the titration has started (the second region), a buffer solution is formed and the following equation is used:

$$pH = pK_a + \log \frac{[A^-]}{[HA]} \quad (\text{Equation 9})$$

The endpoint or the third region is no longer at a pH of 7.00 but can be calculated with the following equations:

$$\frac{[A^-]^2}{([HA]-[A^-])} = K_b = \frac{K_w}{K_a} \Rightarrow [OH^-] \quad (\text{Equation 10})$$

$$pH = -\log \frac{K_w}{[OH^-]} \quad (\text{Equation 11})$$

Once the endpoint is passed or in the last region, the concentration of basic solution was again calculated with Equation 5. The same problem reappeared. Four different and independent equations would not generate points that constitute a smooth curve. For each weak acid-base system,

a different "fudge" factor was required to make the equations work.

The Internet was checked to see if someone else had solved this problem. Several pH titration programs were found but conditions were very limited, such as, pK_a was limited to a range of 4.0 to 6.0. Again, the goal is to have a simulator, which would be able to perform the same range of experiments the normal student would find in their classroom setting. The normal equations were just not working.

A search in the chemistry publications was started to try to find the limits of the available equations and for possible new equations. An article was found in which Dr. Robert de Levie talked about a "different" way to tackle this problem (5). A search was conducted for earlier Dr. Robert de Levie articles. In one of his earlier articles, a new set of equations was found (6). His approach was not to try to find the pH as the titration is being performed, but to use the pH and find the conditions that would have to be present to produce it. Specifically using the starting conditions and the pH to determine how much titrant had been added. The best part is that one set of

equations are used to produce the points for the pH curve or as Dr. Robert de Levie calls it a progress curve, and it doesn't matter whether it is strong, weak, monoprotic or polyprotic:

$$V_t = -V_s \frac{(\sum F_s C_s + [H^+] - [OH^-])}{(\sum F_t C_t + [H^+] - [OH^-])} \quad (\text{Equation 12})$$

$$F = \frac{(\sum_{j=0}^p (p-q-j)H^{p-j}) \prod_{i=0}^j K_i}{\sum_{j=0}^p H^{p-j} \prod_{i=0}^j K_i} \quad (\text{Equation 13})$$

V_t is the titrant volume added

V_s is the sample volume

C_t is the titrant concentration

C_s is the initial sample concentration

H is used as shorthand for $[H^+]$

$K_0=1$

"p" denotes the maximum number of dissociable protons an acid or base can accommodate

"q" defines the actual number of protons associated with a particular species

For monoprotic acids, the F factor is:

$$F = \frac{-K_1}{(H + K_1)} \quad (\text{Equation 14})$$

For diprotic acids, the F factor is:

$$F = \frac{(-HK_1 - 2K_1K_2)}{(H^2 + HK_1 + K_1K_2)} \quad (\text{Equation 15})$$

For triprotic acids, the F factor is:

$$F = \frac{(-H^2K_1 + 2HK_1K_2 - 3K_1K_2K_3)}{(H^3 + H^2K_1 + HK_1K_2 + K_1K_2K_3)} \quad (\text{Equation 16})$$

The above equations, derived from Equation 13, are just shown for illustration and will not be explained. How they were developed and how they work are discussed in Dr. Robert de Levie's two referenced articles (5, 6). It is understood that these equations form the basis for the program.

The next step is to determine the methodology required to use these equations to simulate the function of a pH meter. The accuracy needed by the pH meter in this simulator was determined to be able to show 0.01 pH units. This is the accuracy of the pH meter that most students will find in the laboratory.

During the simulation, the user dispenses a volume of titrant each time a button is selected. This changes the overall volume of the solution and causes a virtual reaction between the acid and base ions. Both of these affect the resulting pH. However, Equation 12 is used to calculate the volume of titrant from the pH and initial concentration of reactants, not the other way around.

In order to use Equation 12 to simulate a titration, the approach is to calculate all the available points on the curve once the system has been selected but before the first volume of titrant is added. The program calculates the required titrant volume added for all the possible displayed pH values and stores them in an array. The during the titration, the program searches the array for the closest calculated volume added to the actual volume added by the simulation user and displays that pH.

Equation 12 has a unique feature. For non-valid pH values, the equation produces negative value results. The algorithm in the simulator starts in the middle of the pH ranges, 7.00, and works out in both directions until a non-valid result is found to determine valid values for storage. The accuracy of the simulator's pH meter is 0.01 pH units. The TitrationEngine class calculates values for every 0.001 units of pH. The results of this algorithm are stored into two objects. One is a Java Vector object of Java Point2dDouble objects, which the GraphPanel class is able to display as the computer-generated pH curve. The other is a two dimensional array of Java Double objects of all the possible pH and volume added values. As the size

of a drop of titrant is fixed, it is possible and very likely that an exact match of volume added and pH is impossible. The algorithm searches the array for the closest match and returns that value to be displayed in the simulator's pH meter. The look of the curves generated shows the success of this algorithm.

2.2 Graph Scales

Once the equations were working properly and without limitations, the scale for the various plots had to be determined. The size of the pH plots is very straightforward. The pH scale goes from 0.00 to 14.00 and the buret chosen is 50.00 mLs in capacity. However, the size of the graphs when displaying the first and second derivatives was not clear-cut. It turns out that the height of these curves can be very large. Using the Demo cases for each, the height of the Y-axis for the first derivatives was determined to be about $3000 \Delta(\text{pH})/\Delta(\text{mLs})$ units for the monoprotic acids, $18 \Delta(\text{pH})/\Delta(\text{mLs})$ units for the diprotic acids, and $0.7 \Delta(\text{pH})/\Delta(\text{mLs})$ units for the triprotic acids. Of course, if the limit was set at $3000 \Delta(\text{pH})/\Delta(\text{mLs})$ units, all the other lines disappeared at the bottom of the graph. Even when the limit was set at 18

$\Delta(\text{pH})/\Delta(\text{mLs})$ units, the smaller peaks were very hard to see. Therefore, it was determined to set values which would cut off the top of the larger peaks but would allow the smaller peaks to be displayed. The Y-axis scale was set to three $\Delta(\text{pH})/\Delta(\text{mLs})$ units for the monoprotic systems, two $\Delta(\text{pH})/\Delta(\text{mLs})$ units for the diprotic, and one $\Delta(\text{pH})/\Delta(\text{mLs})$ unit for the triprotic. Even with the top of the larger peaks cut off, it is clear where the peaks would be.

As for the second derivative peaks, the monoprotic system was above 8,000,000 $\Delta(\Delta(\text{pH})/\Delta(\text{mLs}))/\Delta(\text{mLs})$ units, the diprotic was over 300 $\Delta(\Delta(\text{pH})/\Delta(\text{mLs}))/\Delta(\text{mLs})$ units, and the triprotic was around 0.5 $\Delta(\Delta(\text{pH})/\Delta(\text{mLs}))/\Delta(\text{mLs})$ units. The endpoint is now found when the curve crosses the X-axis. So it was determined in order to have the best view of these crossings to limit the Y-axis scale to plus and minus 0.5 $\Delta(\Delta(\text{pH})/\Delta(\text{mLs}))/\Delta(\text{mLs})$ units. The X-axis scale remained from 0.0 to 50.0 mLs for all graphs.

One last comment on the size or readability of the graphs, it is not possible to read the endpoints of the titration with any real accuracy from the graphs. The graphs are just to show the trends and give the user an

idea of what their results should look like. If the user wants accuracy, they will need to record the pH and volume during titration, just as if they were in the laboratory. The program does no recording of data but displays only the current conditions. See Appendix D for an example of a method to store the data in an Excel spreadsheet. It also shows how to use Excel's chart function to find the endpoints.

CHAPTER THREE

DESIGN

3.1 Architecture (Class Diagram)

In Figure 19, the overview of the class structure can

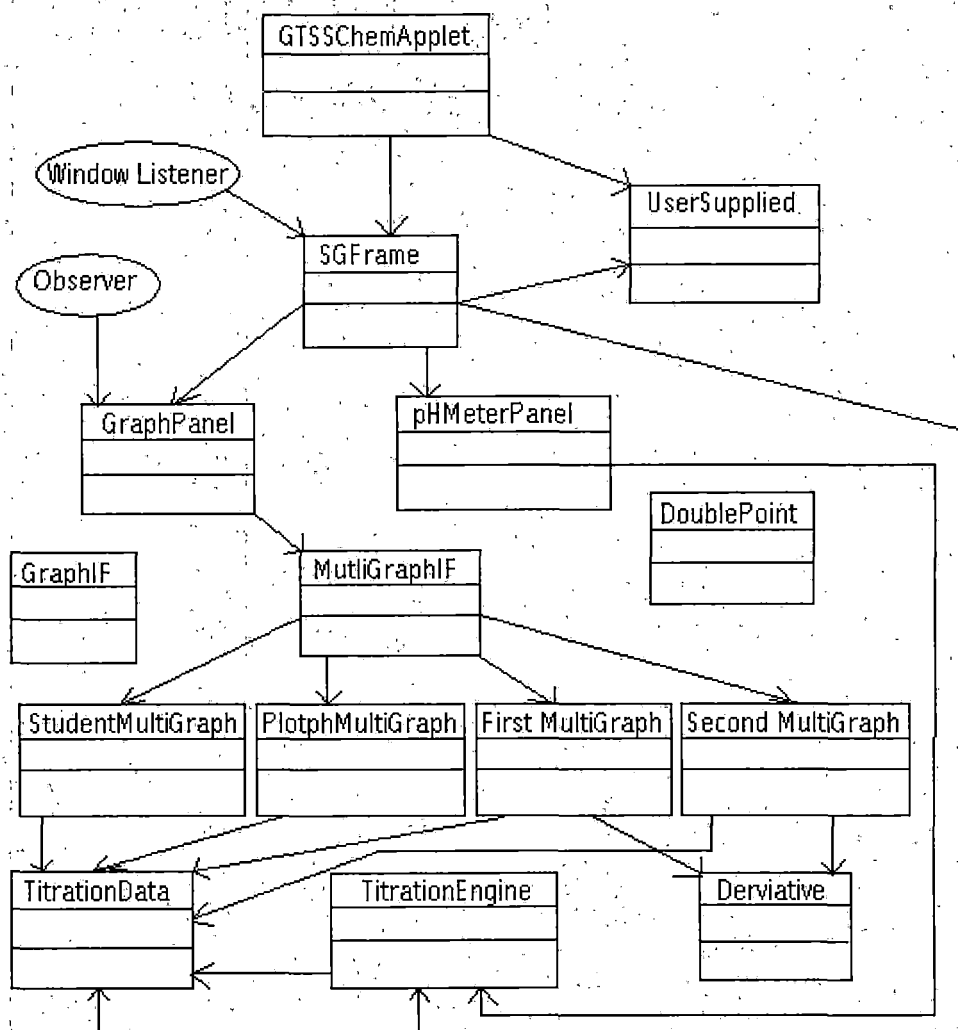


Figure 19. Class Diagram Overview

be seen. Note that the details of each class were not included, as this would make the figure too large. Therefore, each class will be listed separately. In Table 4, GTSSChemApplet's class diagram is shown.

Table 4. GTSSChemApplet Class Diagram

GTSSChemApplet
GtssChemFrame : SGFrame
StoredData : TitrationData
GraphPanel : GraphPanel
PHMeter : pHMeterPanel
Titration : TitrationEngine
UserValues : UserSupplied
Init() : void
Validate() : void
GtssChemFrame.pack() : void
GtssChemFrame.show() : void
paint() : void
destroy() : void
getSGFrame() : SGFrame
setSGFrame() : void

Once the user selects the link to the applet, the GTSSChemApplet class is started. It has the Init() function, which takes the place of the main() function and is executed first. This class has associations with the major classes of this project: SGFrame (Table 5), and UserSupplied.

Table 5. SGFrame Class Diagram

SGFrame
<pre> split : JSplitPane component1 : JComponent component2 : JComponent menuBar : JMenuBar frame : JFrame contentPane : Container </pre>
<pre> SGFrame(String title, JComponent aComponent1, Jpanel aComponent2, TitrationEngine _titration, TitrationData _storedData) : SetLeftComponent(JComponent component) : void SetRightComponent(JComponent component) : void WindowClosing(WindowEvent e) : void WindowOpened(WindowEvent e) : void WindowClosed(WindowEvent e) : void WindowIconified(WindowEvent e) : void WindowDeIconified(WindowEvent e) : void WindowActivated(WindowEvent e) : void WindowDeactivated(WindowEvent e) : void AddMenu() : void </pre>

This class creates the frame in which the applet runs. It sets up the frame with two panels. It associates with TitrationData, TitrationEngine, GraphPanel, pHMeterPanel (Table 6), and UserSupplied (Table 7).

Table 6. pHMeterPanel Class Diagram

<pre> PHMeterPanel Nf : NumberFormat = Number Format.getNumber() GridLayout1 : GridLayout = new GridLayout(8, 0) MachineTitleOnePanel : JPanel = new JPanel MachineTitleOneLabel : JLabel = new JLabel MachineTitleTwoPanel : JPanel = new JPanel MachineTitleTwoLabel : JLabel = new JLabel TitrationTitlePanel : JPanel = new JPanel() TitrationTitleTextField : JTextField = new JTextField(20) PHPanel : JPanel = new JPanel() PHLabel : JLabel = new JLabel() PHTextField : JTextField = new JTextField() AddedVolumePanel : JPanel = new JPanel() AddedVolumeLabel : JLabel = new JLabel() AddedVolumeTextField : JTextField = new JTextField() AddLabelPanel : JPanel = new JPanel() AddLabel : JLabel = new JLabel() AddButtonOnePanel : JPanel = new JPanel() add5mlsButton : JButton = new JButton("+ 5.00 mls") add1mlsButton : JButton = new JButton("+ 1.00 mls") addButtonTwoPanel : JPanel = new JPanel() add_25mlsButton : JButton = new JButton("+ 0.25 mls") add_05mlsButton : JButton = new JButton("+ 0.05 mls") PHMeterPanel (TitrationEngine _titration, TitrationData _storedData) : SetNewPanel(SGFrame gtssChemFrame, GraphPanel graphPanel) : void </pre>

The pHMeterPanel is the class with the controls and displays of the pH Meter and is displayed in the left half of the frame. It associates with TitrationData and TitrationEngine.

Table 7. UserSupplied Class Diagram

<pre> UserSupplied title : String demo : boolean numProtic : int Ka1 : double Ka2 : double Ka3 : double EndPoint : double UserSupplied() : GetTitle() : String SetTitle(String NewTitle) : void GetDemo() : boolean SetDemo(boolean new Demo) : void GetNumProtic() : int SetNumProtic(int newNumProtic) : void getKa1() : double setKa1(double newKa1) : void getKa2() : double setKa2(double newKa2) : void getKa3() : double setKa3(double newKa3) : void getEndPoint() : double setEndPoint(double newEndPoint) : void </pre>

The UserSupplied class is used to set the default acid-base system used in the applet. This class has also been designed to be used by users to input acid-base systems not included in this applet.

The Derivative class, Table 8, is used to mathematically take the derivative of the pH curves. It has been added as an engine to the GTSS system.

Table 8. Derivative Class Diagram

Derivative
lowY : double = 0 lowX : double = 0 highY : doouble = 0 highX : double = 0 first : boolean = true
Derivative() : GetDerivative(Vector testVector) : Vector GetLowY() : double GetLowX() : double GetHighY() : double GetHighX() : double

StudentMultiGraph, PlotMultiGraph, SecondMultiGraph, and FirstMultiGraph are all classes, which are used to pass the vector with the plot points to GraphPanel. They all have a constructor and a function, which returns the Vector. StudentMultiGraph, Table 9, gets the stored data and returns a vector which is used to plot the users pH curve.

Table 9. StudentMultiGraph Class Diagram

StudentMultiGraph
StudentMultiGraph(TitrationData _storedData) : GetPlotPoints() : Vector

PlotMultigraph, Table 10, is used to display both the user and the computer's pH curves.

Table 10. PlotMultiGraph Class Diagram

PlotMultiGraph
PlotMultiGraph(TitrationData _storedData) :
GetPlotPoints() : Vector

FirstMultiGraph, Table 11, is used to display the first derivatives of the pH curves.

Table 11. FirstMultiGraph Class Diagram

FirstMultiGraph
FirstMultiGraph(TitrationData _storedData) :
GetPlotPoints() : Vector

SecondMultiGraph, Table 12, is used to display the second derivatives of the pH curves.

Table 12. SecondMultiGraph Class Diagram

SecondMultiGraph
SecondMultiGraph(TitrationData _storedData) :
GetPlotPoints() : Vector

The GraphPanel, Table 13, class is used to display all the data produced by this applet in the right side of the

Table 13. GraphPanel Class Diagram

GraphPanel
PlotPoint[^] : Vector XRangeLow : double = 0.0 XRangeHigh : double = 50.0 YRangeLow : double = 0.0 YRangeHigh : double = 14.0 mutli : boolean = false graphColor[^] : Color = {Color.red, Color.black, ... scaleColor : Color = Color.blue fontColor : Color = Color.black panelWidth : int panelHeight : int leftOffset : int = 28 rightOffset : int = 18 topOffset : int = 18 bottomOffset : int = 28 tickDivisions : int = 5 graphWidth : int graphHeight : int drawAxes : boolean = true drawAtEdges : boolean = false minDataScaled : double maxDataScaled : double minScaleValue : double maxScaleValue : double nf : NumberFormat = NumberFormat.getNumber() xMaxFractionDigits : int = 2 yMaxFractionDigits : int = 2 f : Font = new Font("Serif", ... log10 : double = Math.log(10.0)

Table 14. GraphPanel Class Diagram - Continued

```

GraphPane(MultiGraphIF multiGraphIF):
GraphPanel(GraphIF graphIF) :
SetMaxFractionDigits(double max, double min) : int
paint (Graphics g) : void
scaleX(double x) : int
scaleY(double y) : int
paintHorizontalScale(Graphic g, double hScalePosition)
: void
paintVerticalScale(Graphic g, double vScalePosition) :
void
setViewport(double _xRangeLow, double _xRangeHigh,
double _yRangeLow, double _yRangeHigh) : void
setGraphColor(Color color) : void
setGraphColors (Color color[]) : void
setScaleColor(Color color) : void
setFontColor(Color color) : void
setTickDivisions(int value) : void
setDrawAxes(boolean value) : void
isDrawAxes() : boolean
setDrawAtEdges(boolean value) : void
isDrawAtEdges() : void
setMulti() : boolean
setXRangeLow(double _xRangeLow) : void
setXRangeHigh(double _xRangeHigh) : void
setYRangeLow(double _yRangeLow) : void
setYRangeHigh(double _yRangeHigh) : void
update(Observable o, Object x) : void
AttachObservable(TitrationData storedData) : void

```

frame. It has two constructors. GraphPanel(GraphIF graphIF) is used when there is only one line to display (not used by this applet). GraphPanel(MultiGraphIF multiGraphIF) is used when multiple lines are displayed. The function setMaxFractionDigits(double max, double min) is used to set the number of digits in the scales. The

scales for the axes are set with `scaleX(double x)` and `scaleY(double y)`. `Paint(Graphics g)` function calls the Java super class `Paint()` and draws the graphs. The functions

```
    paintHorizontalScale(Graphic g, double hScalePosition)
```

and

```
    paintVerticalScale(Graphic g, double vScalePosition)
```

are used if the vertical and horizontal scales are to be included. The function `setViewport(double _xRangeLow, double _xRangeHigh, double _yRangeLow, double _yRangeHigh)` is used to set the size of the area to be displayed.

`TitrationData`, Table 15, is the class, which stores the ideal vector and adds the user's inputs to the student vector.

Table 15. TitrationData Class Diagram

TitrationData
<p>pH : double volumeAdded : double = 0.0 maxVolume : double = 50.0 idealVector : Vector lowY : double = 0 lowX : double = 0 highY : double = 0 highX : double = 0</p>
<p>TitrationData(TitrationEngine _titration) : AddToStudentpHPlotVector(Point2D.Double newData) : void Changed() : void GetLowY() : double GetLowX() : double GetHighY() : double GetHighX() : double SetLowY(double _lowY) : void SetLowX(double _lowX) : void SetHighY(double _highY) : void SetHighX(double _highX) : void</p>

TitrationEngine, Table 16, is the class, which does the work for the applet. It takes the acid-base parameters and calculates the ideal vector. It also takes the volume added and calculates the pH of the solution.

Table 16. TitrationEngine Class Diagram

<pre> TitrationEngine H3Omolar : double OHMolar : double AcidMolar : double InitialAcidVolume : double = 10.0 BaseMolar : double = 0.10 EndPoint : double = -1 Kw : double = Math.pow(10, -14) Ka1 : double = 1 Ka2 : double = 0 Ka3 : double = 0 demo : boolean = false numProtic : int = 1 maxVolume : double = 50.0 minpH : int = -1 maxpH : int = -1 volumeTitratedAdded[^] : double = new double [1400] title : String TitrationEngine() : TitrationEngine(String title, boolean demo, int numProtic, double Ka1, double Ka2, double Ka3, double endPoint) : pHCal (double volumeAdded) : double initialpH_idealVector() : Vector </pre>

GraphIF and MultiGraphIF are the interfaces used by GraphPanel to get the vectors with the plot points. They only have one function, which returns the vectors to be plotted. Their class diagrams are not listed as they are GTSS engines and are used un-modified by this applet.

3.2 Detailed Design (Pseudo-Code)

This section deals with the pseudo-code of this applet. The tables are divided into two parts. The first part shows the overall scope of the class: class name, where used, purpose and note. The second part shows the pseudo-code.

For this project to run as an applet over the Internet, the browser firsts loads a HTML file from the GTSS web site. The file, GTSSChemApplet.html, contains the link to GTSSChemApplet.java (Table 17).

Table 17. GTSSChemApplet Pseudo-Code

Class Name	GTSSChemApplet
Where Used	GTSSChemApplet.html
Purpose	Starts the Applet
Note	GTSS Web Site, Chemistry Page
Begin	
Declare and Create Global Classes	
TitrationEngine	
TitrationData	
pHMeterPanel	
GraphPanel	
UserSupplied	
Setup Display	
Call SGFrame	
End	

This declares and creates the classes used by the applet as well as the Window frame in which the program runs on the user's computer.

The frame is created by SGFrame (Table 18). It uses two sides and a menu bar. The left panel is filled with the buttons, labels, and current values used by the pH meter. The right panel displays the results of the graphing functions of the program: pH titration, first derivative, and second derivative curves. The menu bar is created and all the options of the program are displayed.

Table 18. SGFrame Pseudo-Code

Class Name	SGFrame
Where Used	GTSSChemApplet
Purpose	Sets up the Display area
Note	Main frame of the program
<pre> Begin Create left panel from pHMeterPanel Create right panel from GraphPanel Create Menu Bar Select Titration System Select Graph to be displayed Repeat Titration Once selected create new left and right panels and update display Display frame End </pre>	

Once a titration system has been selected, the TitrationEngine class, Table 19, calculates the "ideal" or computer generated pH curve and stores the results into a vector to be used by GraphPanel. The results are also stored in an Array, which is used during the user's titration to determine current pH.

Table 19. TitrationEngine Pseudo-Code

Class Name	TitrationEngine
Where Used	SGFrame
Purpose	Calculates the pH curve
Note	Calculates pH during titration as well
Begin	<pre> Get the required parameters Calculate Volume Titrant Required Array Calculate the "ideal" pH curve vector Using the Array, find pH which corresponds titrant volume when passed by pHMeterPanel </pre>
End	

The TitrationData class, Table 20, is used to store the results of the user's titration. After each addition of titrate, the current pH is obtained from the TitrationEngine and is added to the user's vector. GraphPanel is then updated to display current vectors.

Table 20. TitrationData Pseudo-Code

Class Name	TitrationData
Where Used	SGFrame
Purpose	Stores all the data during a titration
Note	Notifies GraphPanel when to update display
Begin	<pre> Initialize the parameters Store values during titration Create vector of student points during titration Notify GraphPanel when a new point has been created </pre>
End	

The GraphPanel class, Table 21, is used to graph the data created by this program. It set ups the colors for the different curves, the size of the pH and Volume Added axis, and displays the current values.

Table 21. GraphPanel Pseudo-code

Class Name	GraphPanel
Where Used	SGFrame
Purpose	Converts vector data into graphs
Note	Notified by TitrationData of new points
Begin	<pre> Initialize the right panel Setup the colors Size of the X axis and Y axis Setup labels Get vectors and create graphs Update display </pre>
End	

The pHMeterPanel class, Table 22, allows the user to manage the titration and display the results. It displays the current titration system, volume added, and pH value.

Table 22. pHMeterPanel Pseudo-Code

Class Name	pHMeterPanel
Where Used	SGFrame
Purpose	Dispensing the titrant during the titration
Note	Displays the data during the titration
<pre> Begin Initialize the left panel Setup the colors Create the display windows Create the dispense buttons Label the panel If button is depressed Pass volume of titrant selected to TitrationData Display returned values End </pre>	

The StudentMultiGraph class, Table 23, uses the GTSS interface, MultiPlotIF, to pass two vectors to GraphPanel. The first is a dummy vector and is hidden under the axis lines. It is needed as MultiPlotIF requires more than one vector. The second is the user's pH vector.

Table 23. StudentMultiGraph Pseudo-Code

Class Name	StudentMultiGraph
Where Used	GraphPanel
Purpose	Passes vectors to GraphPanel for display
Note	Doesn't display the "ideal" pH curve
Begin	<pre> Get student vector from TitrationData Create a dummy vector (hide under axis lines) Pass vectors to GraphPanel </pre>
End	

The PlotpHMultiGraph class, Table 24, is used to pass two vectors to GraphPanel to be displayed. The first is the user pH curve and the second is the "ideal" curve, which is computer generated.

Table 24. PlotpHMultiGraph Pseudo-Code

Class Name	PlotpHMultiGraph
Where Used	GraphPanel
Purpose	Passes vectors to GraphPanel for display
Note	Both Student and "ideal"
Begin	<pre> Get student and "ideal" vectors from TitrationData Pass vectors to GraphPanel </pre>
End	

The FirstMultiGraph class, Table 25, is use to display the mathematical analysis of the pH curves. It gets the two vectors like before but now it sends them to the

derivative class to have the first derivative of each curve calculated.

Table 25. FirstMultiGraph Pseudo-Code

Class Name	FirstMultiGraph
Where Used	GraphPanel
Purpose	Passes vectors to GraphPanel for display
Note	First derivative of pH curves
Begin	<pre> Get student and "ideal" vectors from TitrationData Create Derivative class Create derivative vectors from student and "ideal" Pass vectors to GraphPanel Pass the scale of the graphs to GraphPanel. </pre>
End	

The SecondMultiGaph class, Table 26, is used to display the second derivative of the curves. It gets the two vectors and passes them to the derivative class. The resulting vectors are then passed to the derivative class again to produce the second derivative.

Table 26. SecondMultiGraph Pseudo-Code

Class Name	SecondMultiGraph
Where Used	GraphPanel
Purpose	Passes vectors to GraphPanel for display
Note	Second derivative of pH curves
<pre> Begin Get student and "ideal" vectors from TitrationData Create Derivative class Create derivative vectors from student and "ideal" Repeat to obtain second derivatives of the vectors Pass vectors to GraphPanel Pass the scale of the graphs to GraphPanel End </pre>	

The last class is the Derivative class, Table 27. It accepts the input vectors and returns the calculated derivative vector.

Table 27. Derivative Pseudo-Code

Class Name	Derivative
Where Used	FirstMultiGraph & SecondMultiGraph
Purpose	Take the derivative of the curve
Note	
<pre> Begin Get passed vector If Vector is too small Return a vector with one point (0,0) Else Loop through vector Get Objects stored at i, i+1 Convert Objects into X, Y Calculate derivative Create a new Object Store value and location in Object Store Object in Returned Vector End Loop End If Return the Created Vector End </pre>	

CHAPTER FOUR

OPERATING INSTRUCTIONS

4.1 Operating Instructions

The titration applet has two primary functions.

First, it is an acid-base titration demonstrator. The user can titrate monoprotic, diprotic, and triprotic acids and one base, NaOH. Using the selection menu, the user selects the type of acid-base system to be demonstrated.

There are two type of systems, one with fixed endpoints and one where the endpoints are unknown. In the Demo mode, the endpoints are fixed: 20.83 mLs for the monoprotic acids and the base; 15.83 and 31.66 mLs for the diprotic acids; and 10.83, 21.66, and 32.49 mLs for the triprotic acids. Note that the smallest unit, which can be dispensed is 0.05 mLs. 0.05 mLs was picked as the smallest volume as it was felt that was the minimum size drop that an average student could deliver with a buret. The size of the drop the other buttons deliver was selected to allow the ability to rapidly perform a titration while, at the same time, have fine enough control to accurately duplicate the conditions found in the laboratory. It is impossible to stop exactly on these endpoints. It is felt this

simulates the experience the student would find in the laboratory setting.

For non-demo acid-base systems, an unknown endpoint is randomly generated. This simulates the type of problem the user would face in the laboratory.

Once a titration is complete, the type of graph is selected from the "Graph" menu, which is displayed on the right side of the window. Selection of any of the items from this menu ends the titration, that is, no more titrant can be added. During the titration, the default screen is the typical pH versus volume added graph created by the user. The options are: pH Plot, 1st Derivative, and 2nd Derivative. The "pH Plot" shows both the student titration as well as the pH curve generated by the program. It allows the comparison of the student's effort against the "ideal." The second option shows the first derivative of both these pH curves. The endpoints are mathematically shown as the peaks of these new curves. The third option displays the second derivative of the pH curves. The endpoints are now when the curves cross the X-axis. The student curve is always printed last so the computer's red lines are covered by the student's black line.

The Repeat Titration Menu allows the same type of titration to be repeated. If the same endpoint is selected, the program is just reset and the titration is started again. The idea is to allow the student to correct any mistakes made during earlier titrations. In addition, it allows the technique of quickly performing a titration to coarsely find the endpoint and then refining the endpoint on succeeding titrations. If the new endpoint option is selected, a new endpoint is randomly generated by the program: monoprotic from 20 to 45 mLs; diprotic from 15 to 24 and 30 to 48 mLs; and triprotic from 10 to 15, 20 to 30, and 30 to 45 mLs. (This option is over-ridden if the Demo mode is selected.)

The other main function is to demonstrate the principles of pH and the properties of weak acids. The users can by-pass the titration phase. After selecting the acid-base system, the user can select from the Graph Menu and display just the computer-generated curves. The shape of these curves clearly demonstrates many of these pH principles.

4.2 Hints for the Instructor

This Acid-Base Titration Simulator has been designed to allow the demonstration of several chemical principles. Ethylamine(+1), one of the monoprotic acids, has a K_a such that no endpoint can be seen during the titration. Several of the diprotic and triprotic acids have endpoints, which don't resolve. For example, D-Tartaric acid's K_a constants are too close together to titrate properly. Phosphoric and L-(+)-Arginine(+2) acids have their K_{a3} values so small that the third endpoint doesn't resolve properly.

The main goal was to allow you, the instructor, to demonstrate the proper method to titrate acid-base solutions without being in the laboratory setting. For example, if the entire titration is performed with the "+5.00 mLs" button, the endpoint can't be accurately determined. However, it is one way to show how easy it is to overshoot the endpoint.

The selection of the Demo option fixes the endpoint. This way, the instructor can rapidly titrate to near the known endpoint with the "+5.00 mLs" and "+1.00 mLs" buttons and then slowly approach the endpoint with the "+0.25 mLs" and "+0.05 mLs" buttons. For example, the monoprotic acid

endpoint of 20.83 mLs can be reached by four pushes of the "+5.00 mLs" button (20.00 mLs), three pushes of the "+0.25 mLs" button (20.75 mLs), and one push of the "+0.05 mLs" button (20.80 mLs). Then one more push of the "+0.05 mLs" button will take you just past the endpoint to 20.85 mLs. Of course, there are many other ways to reach this endpoint.

One of the limitations of this program is the inability to show exactly the magnitude of the student error. There are times when looking at the graphs displayed, the student error doesn't look that large. As the largest amount that can be added is 5.00 mLs, the largest error is 2.50 mLs. This distance on the displayed graph is quite small. Of course, this visual problem increases with the use of the 1.00 mLs, the 0.25 mLs, and the 0.05 mLs buttons. The mathematical analysis of these errors is left to the instructor.

In addition, the height of the peaks for the first derivative curves can be so large as to make the graph unusable. The tops of the larger peaks are cut off but there is no doubt, where the peak would be. The same is true for the second derivative curves but now the endpoint

is where the curve crosses the X-axis. The scale of the Y-axis was picked to allow the best viewing of these crossings.

4.3 Hints for the Student

This program was designed for you. It will allow you to titrate several different acid-base systems as many times as you would like. The first hint deals with the "Graph menu." The purpose of this menu is to show the results of your titration efforts. Therefore, **IT STOPS THE TITRATION.** There is no value in "looking ahead" as you can't do that in the real world. So, complete any titration before selecting a different graph to be displayed. If you want to run the exact situation again, select the "Repeat Titration" menu and click on "Same Equivalence Point." The titration will be set back up exactly as before.

Secondly, the graphs are not accurate enough for you really see the value of taking the first and second derivatives of the pH curves. The program also **DOES NOT RECORD** the "pH" versus "volume added" values but only displays the current values. Therefore, you should make a

table of pH versus volume added so you can do your own graphs of the curves. Appendix E has an example of a titration and the mathematical analysis required. This example uses an Excel spreadsheet and the charts it is able to create.

In the real world, the endpoint will be completely unknown. Therefore, it is hard to determine how much titrant to add. One way is to always add the smallest amount possible but this takes a long time. Another method is to run a titration using just the 5.00 mLs button. While this won't find the endpoint accurately, it will get you in the proper range. Run the titration again using the 5.00 mLs button but stop before reaching the endpoint point area. Now use the 1.00 mLs button until past the endpoint. Now you know the endpoint to plus or minus 1.00 mLs. Rerun the same titration using the 5.00 and 1.00 buttons to quickly arrive at a point just short of the endpoint and now the 0.25 and 0.05 mLs buttons can be used to fine-tune the endpoint.

4.4 Testing

The majority of the testing of this applet was performed at Victor Valley (VVCC) and Barstow (BCC) Community Colleges. Chemistry students at VVCC and computer science students at BCC tested the prototype of this program. There were no cases of a user being able to crash or lock-up the applet. The only input from the user is the mouse. This makes it almost impossible for the user to do anything that will crash the applet.

This project was also presented to the Computational Chemistry Council during one of its' conferences at California State University-San Bernardino from 9-10 July 2001. During the poster presentation session, several of the professors used the program. It received favorable comments from all of them.

A copy of the questionnaire used is in Appendix A. The questionnaire provided no trends or any statistically valid data.

4.5 How to Input Your Own Acid System

For the users with some computer or Java experience, it is possible to add your own acid to the applet.

However, this will require all the files to be downloaded to the user's computer and ran there.

Step 1. From the GTSS web site, download the files: j2sdk-1_3_0_02-win.exe and ChemApplet.zip to the desktop or other location on the user's computer.

Step 2. The first file is self-extracting and will set up Java 1.3 on the user's computer (This procedure has not been tested with other versions of Java). Double click on the j2sdk-1_3_0_02-win.exe file and follow the instructions to install Java 1.3.

Step 3. Create a new folder on the C drive called GTSSChemJava and move or copy the ChemApplet.zip file into it. Double click on it and extract all the files into the folder.

Step 4. Using a text editor open the GTSSChemApplet.html file. Using the search function, locate the three places where the word GTSSChemJava is. At the first location, insure that the line segment reads:

```
CODEBASE VALUE = "C:\GTSSChemJava\"
```

At the other two locations, insure that the line reads:

```
CODEBASE = "C:\GTSSChemJava\"
```

Of course if you put the files in another location, it will be your responsibility to change the three CODEBASE values to the proper path so the applet can find the required files. Once the proper paths have been changed, save the file.

Step 5. Load the UserSupplied.java file into the text editor. There are several pieces of information required: the title of the acid/base system, the demo mode, the number of Hydrogen ions dissociated, the appropriate dissociation constants, and the endpoint value. As you read the instructions inside the file, you will see the default values and examples of how to replace them with the new values.

Step 5a. On the line of code:

```
title = "Sulfurous Acid with 0.1M NaOH";
```

delete Sulfurous and replace with the name of the new acid (At this point, the only base is 0.1M NaOH).

Step 5b. The demo mode is used to fix the endpoint, so that, it can not be changed by the program or the user.

```
demo = true; is used to fix the endpoint.
```

or

demo = false; is used to have random endpoints.

(Do not use quotes but only the words true or false with all lower case letters.) If demo is set to true, a nonzero or nonnegative value has be assigned to the endpoint value in Step 5e.

Step 5c. Next, set the number of protons available (ONLY VALUES of 1, 2, or 3 ARE VAILD).

numProtic = 1; is for monoprotic acids.

numProtic = 2; is for diprotic acids.

or

numProtic = 3; is for triprotic acids.

Step 5d. Now, the dissociated constants, the "Ka"s, are entered. Use 1.0 for Ka1 for strong acids. Use zeros for "Ka"s not used. The following is an example for a monoprotic acid with a $K_a = 7.1 \times 10^{-4}$:

```
Ka1 = 7.1 * Math.pow(10, -4);
```

```
Ka2 = 0.0;
```

```
Ka3 = 0.0;
```

(Java uses a function to calculate the value of 10^{-4} . The function is Math.pow(a,b). Where "a" will always be 10 and "b" the exponent.) Is an example of a diprotic acid:

```
Ka1 = 4.70 * Math.pow(10, -3);
```

```
Ka2 = 1.80 * Math.pow(10, -10);
```

```
Ka3 = 0.0;
```

The last is an example of a triprotic acid:

```
Ka1 = 1.0 * Math.pow(10, -4);
```

```
Ka2 = 1.0 * Math.pow(10, -8);
```

```
Ka3 = 1.0 * Math.pow(10, -12);
```

Step 5e. Lastly, the value of the endpoint is set.

Set endpoint to -1, if you want the computer to compute a random endpoint or set endpoint to a value you want (ONLY USE VALUES FROM 5.00 to 40.00 mL for monoprotic acids, 5.00 to 20.00 mL for diprotic acids, and 5.00 to 15.00 mL for triprotic acids). For example, the following endpoint, 6.30 mL, allows the use of each of the 5.00, 1.00, 0.25, and 0.05 mL buttons to find the endpoint:

```
endPoint = 6.30;
```

(Double check to see if the endpoint is set to -1 that the demo mode is set to false.) REMEMBER, there is no error checking preformed. It is up to the user to ensure the values entered are correct. DO NOT CHANGE THE SPELLING OR THE CASE (LOWER OR UPPER) OF ANY OF THE VARIABLES. Java thinks endpoint and endPoint are two different things. Save the changes and close the text editor.

Step 6. Once the changes have been made, the file needs to be re-compiled. Click on the "Start" button at the bottom of the Window screen. Select "Programs" and then "MS-DOS Prompt." You should see "C:/WINDOWS>" which means you are in the Windows directory. Change the directory to GTSSChemJava by the following commands.

Type:

```
cd ..
```

Type:

```
cd GTSSChemJava
```

Now, compile the new UserSupplied.java.

Type:

```
c:\jdk1.3.0_02\bin\javac UserSupplied.java
```

Test the results with appletviewer.

Type:

```
c:\jdk1.3.0_02\bin\appletviewer GTSSChemApplet.html
```

Once back in the Windows mode, the applet will also run by double clicking on the GTSSChemApplet.html file and then run in your browser. Have fun!

CHAPTER FIVE

MAINTENANCE

5.1 Files

The files for this applet are stored in the GTSS Server Computer in Computer Science department. In addition, all the files are copied on a CD ROM disk. A copy of this disk is stored in the back of the hardbound copy of this project stored in the Computer Science Office.

There are four main types of files used by this applet: Java source code (*.java), compiled Java Unicode (*.class), zip compressed files (*.zip), and HTML web pages (*.html). Inside the HTML files, several graphic styles are used, for example, jpeg, gif, and bmp.

The files found in Table 28 are the source files created for this project. They are stored in ASCII format and can be viewed and modified by a text editor type program.

Table 28. Java Source Files

Derivative	DoublePoint	FirstMultiGraph
GraphIF	GraphPanel	GTSSChemApplet
MultiGraphIF	PHMeterPanel	PlotMultiGraph
SGFrame	TitrationData	SecondMuiltGraph
UserSupplied	TitrationEngine	StudentMultiGraph

Table 29, shows the two HTML files. These are the files used by the browser and are stored in ASCII format. They can be viewed and modified by a text editor.

Table 29. HTML Files

GTSSChemApplet	GTSSTitration
----------------	---------------

There is one zip file: GTSSChem.zip. This file was created by the deployment wizard of the *JBuilder3.0* program. It contains all the files needed to run the applet once downloaded to the users computer.

When the source code is compiled with `javac.exe`, forty three files are created. These files shown in Table 5 are the ones that the Java Virtual Machine executes to run the applet.

Table 30. Compiler Created Class Files

SGFrame	SGFrame\$11	SGFrame\$22	FirstMultiGraph
SGFrame\$1	SGFrame\$12	SGFrame\$23	GTSSChemApplet
SGFrame\$2	SGFrame\$13	SGFrame\$24	TitrationData
SGFrame\$3	SGFrame\$14	DoublePoint	pHMeterPanel\$1
SGFrame\$4	SGFrame\$15	MultiGraphIF	pHMeterPanel\$2
SGFrame\$5	SGFrame\$16	GraphPanel	pHMeterPanel\$3
SGFrame\$6	SGFrame\$17	Derivative	pHMeterPanel\$4
SGFrame\$7	SGFrame\$18	GraphIF	PlotMultiGraph
SGFrame\$8	SGFrame\$19	UserSupplied	SecondMulitGraph
SGFrame\$9	SGFrame\$20	PHMeterPanel	TitrationEngine
SGFrame\$10	SGFrame\$21		

5.2 Directories

In the GTSS Server Computer, there are several directories or folders used by the GTSS project. The path to find the main GTSS directory is `\gtss\edu\csusb\gtss`. Upon opening the `gtss` directory, several folders are displayed: `csci` for Computer Science, `math` for Mathematics, `phys` for Physics, `util` for GTSS engines, `stat` for Statistics and `chem` for Chemistry. Opening the `chem` folder will display a `titration` folder in which all the applet files are stored. Therefore, the total path to this applet files is `\gtss\edu\csusb\gtss\chem\titration`. As Java skilled users can download and modify this applet, it was felt that all files would need to be in one location. This way a user would not have to download the entire GTSS system to be able to use the titration demo at home.

5.3 How to Compile

The administrator of the GTSS Server Computer mapped the Java compiler, javac.exe, so that it will run from any directory on the computer. If there is a need to re-compile the applet, the first step is to delete all the *.class files. Second is to run the Java compiler, javac.exe, on the source code. Change the directory to \gtss\edu\csusb\gtss\chem\titration\

Then, at the Linux prompt, \gtss\...\chem\titration\%,

```
type rm *.class
```

then

```
type javac GTSSChemApplet.java
```

Compiling the GTSSChemApplet.java file will cause the compiler to compile the all source *.java files and generate the *.class files for each Java source file.

If the files are moved to another directory or folder, there are three places in the GTSSChemApplet.html file where the "BASECODE = \GTSS\edu\csusb\gtss\chem\titration\" variable needs to be changed. The BASECODE need to reflect the path to the directory where the files were moved.

CHAPTER SIX
FUTURE DEVELOPMENTS AND
CONCLUSIONS

6.1 Ideas for Future Developments

There are several limitations to the program as written. The first is there is no way to insert the data required to titrate an acid-base system not already in the "Select Titration" menu. The UserSupplied class is a partial fix for this problem. If a Java skilled user downloads the files, they would be able to change the variables to that of their acid-base system and re-compile. Then run the applet on their computer.

This could be fixed by adding another menu, which would allow the frame to display a form or window in which to fill in all the required information. Once filled in and checked for errors, it would pass the parameters to various classes just like the "Select Titration" menu does.

A better solution would be to design the applet to access a spreadsheet or database to get the variables needed to define an acid-base system. The user would be able to download just the spreadsheet or database template

to their computer and add the new data. That way, the user would not have to re-compile the java source code.

There are several limitations on the ability to add just any acid-base system. The equations used in this applet are based on a couple of factors: p and q . The equation was written for systems where the p factor equals the q factor. If you want to add an acid-base system where the p factor equals three and the q factor equals two, for example, then the equations in the titration engine would have to be modified. In addition, the present program stops at triprotic acids. Therefore, for acid-base systems where the p factors are larger than three, the equations would have to be changed.

The X-axis scale of the graph is fixed to 0.00 to 50.00 mLs, which displays the different curves. Another possible enhancement is to be able to select different areas of the graph and expand them. This would more clearly show the magnitude of the student errors and determine the endpoints more accurately.

As there is no recording of results during the titration, the student cannot go back and reconstruct the data. Another enhancement would have the "pH" and "volume

added" values be displayed when the user places the cursor on the displayed curve and the mouse button clicked. The closest student data point values would be displayed. That is, a student can reconstruct the data from the plots after the titration is completed.

6.2 Derivative Engine

In the process writing the code for this applet, it was determined that there was a need to perform a mathematical analysis on a graphical curve, the pH curve. The normal analysis includes the first and second derivatives. At first, the Derivative Class was developed to take a vector of DoublePoint Class objects and return another vector with DoublePoint objects, which represented the derivative curve. The DoublePoint Class was used by the GraphPanel Class and was written for the GTSS project. However, later versions of Java included a point object of the type double, Point2dDouble.

Therefore, the development of the Derivative Class switched to use the updated Java objects. The method to find $\Delta y/\Delta x$, the derivative, of a curve involved taking each pair of adjacent points (i, i+1) and performing some simple mathematics. First, the value of $\Delta y/\Delta x$ or Y value

was found by the equation: $(Y_{i+1} - Y_i)/(X_{i+1} - X_i)$. The X value is found by the equation: $(X_{i+1} + X_i)/2$. This new point with its' X and Y values is added to a new vector. Once all the pairs had been analyzed, the vector with the derivative curve is returned to the GraphPanel class.

Once the class was working properly, it was felt this Derivative Class would work as a GTSS engine and could be used by other GTSS programs. Some of the early GTSS programs used the DoublePoint object and others used the newer Java Point2dDouble objects. There is also another method to store the X and Y values for a curve and that is to use a two-dimensional array. To make this class as flexible as possible, the Derivative Class was rewritten to use all the objects listed above and was renamed DerivativeEngine Class. Presently, the GTSS project doesn't use JavaBeans but the style used in the development of this engine is that of a JavaBean (10). This is for future development.

There are three interfaces needed to use this new class. There is one for each type of object. The job of these interfaces is to allow the DerivativeEngine Class to use different types of objects. The first is ArrayDoubleIF and it uses the two-dimensional array of double type

objects. The second is VectorDoublePointIF and it uses a vector of the DoublePoint type objects. The last is VectorPoint2dDoubleIF and it uses a vector of Java Point2dDouble type objects.

Copies of files are included in the GTSS utility directory. The path is /gtss/edu/csusb/gtss/util. The Java code for these files is in Appendix C.

6.3 Conclusions

All the design features proposed for this project have been implemented in accordance with the Software Engineering Standards Committee of the IEEE Computer Society's *IEEE Recommended Practice for Software Requirements Specifications* (7). The applet runs on the GTSS Web site; it accurately demonstrates several acid-base principles; and it allows users to practice titrations outside the laboratory. The program does all that was hoped for in the initial design phase.

The benefits of this applet are easily demonstrated. For instructors, it has the ability to quickly and graphically show the methods for acid-base titrations. The pH curves generated clearly demonstrate several chemical principles. When demonstrating techniques using laboratory

equipment, it is hard to have the entire class be able to observe properly. In most modern classrooms, the applet can be shown on the classroom TV or monitor, which allows the whole class to watch at the same time. For students, it allows them to practice acid-base titrations at their computer instead of in the laboratory with dangerous and poisonous chemicals. They can progress at their own pace. In the laboratory, the student is limited to the number of acid-base titrations permitted. There are unlimited titrations available while using the applet.

APPENDIX A:
QUESTIONNAIRE

QUESTIONNAIRE

This questionnaire is for Thomas Gummo's Masters Project,
acid-base titration demonstration applet.

Your help is requested in evaluating this program.

Check _____ student or _____ instructor.

Fill in educational institution _____

Were you able to crash the program? If so, HOW?

Did all the menus and buttons work properly?

How would you change the menus or buttons?

Evaluate the setup of the menus and buttons?

Evaluate the choices for titration?

What additional features would you like?

Were you able to run the program on your system?

What computer system did you use?

What browser? Version?

Type of connection? Speed?

Was the speed adequate?

Evaluate the appearance of the program?

How would you change the appearance of the program?

Was the Web Page helpful?

How would you change the Web Page?

APPENDIX B:
TABLES OF DISSOCIATION
CONTANTS

TABLES OF DISSOCIATION CONSTANTS

The data for the following tables was obtained from the following references.

Reference 1: "Quantitative Chemical Analysis," 4th ed. Daniel C. Harris, W.H. Freeman: New York, 1995; Appendix G.

Reference 2: "Lange's Handbook of Chemistry", 14th ed. John A. Dean, Ed. McGraw Hill: New York, 1992; 8.19-8.71.

Table1. Ka Factors for Monoprotic Acids

Monoprotic Acid Name	Reference	Ka
Demo	N/A	1.00
HCl	N/A	1.00
Sulfurous	1	1.4×10^{-2}
Nitrous	1	7.1×10^{-4}
Acetic	1	1.75×10^{-5}
Hypochlorous	1	3.0×10^{-8}
Ethylamine (+1)	1	2.31×10^{-11}

Table 2. Ka Factors for Diprotic Acids

Diprotic Acid Name	Reference	Ka1	Ka2
Demo	N/A	1.00×10^{-4}	1.00×10^{-8}
D-Tartaric	2	9.20×10^{-4}	4.30×10^{-5}
Carbonic	1	4.45×10^{-7}	4.69×10^{-11}
Alanine (+1)	1	4.49×10^{-3}	1.36×10^{-10}
Adenine (+1)	2	6.76×10^{-5}	1.78×10^{-10}
L-Leucine (+1)	1	4.69×10^{-3}	1.79×10^{-10}

Table 3. Ka Factors for Triprotic Acids.

Triprotic Acid Name	Ref	Ka1	Ka2	Ka3
Demo	N/A	1.00×10^{-3}	1.00×10^{-6}	1.00×10^{-9}
L-Glutamic(+1)	1	5.9×10^{-3}	3.8×10^{-5}	1.12×10^{-10}
Phosphoric	1	7.11×10^{-3}	6.34×10^{-8}	7.1×10^{-13}
L-(+)-Arginine(+2)	1	1.50×10^{-2}	1.02×10^{-9}	3.3×10^{-13}
Tetracycline(+1)	2	5.01×10^{-4}	2.09×10^{-8}	2.04×10^{-10}

Note 1: The Ka for a strong acid or base is 1.00 and is common knowledge among chemistry students and instructors.

Note 2: The only base used by the program is NaOH and is a strong base.

Note 3: N/A, not applicable, is used as the reference for strong acids and the demo acids. Demo acid-base systems had their Ka factors made up.

APPENDIX C:
DERIVATIVEENGINE JAVA CODE

DERIVATIVEENGINE SOURCE CODE

DerivativeEngine.java

```
/*    File DerivativeEngine.java */

import java.util.Vector;
import java.awt.geom.*;

public class DerivativeEngine {

    // values to store the largest and smallest X and Y
    // used by GTSS GraphPanel
    private double lowY = 0;
    private double lowX = 0;
    private double highY = 0;
    private double highX = 0;

    // used to select case
    private int typecase = 0;
    // used to set up lows and highs
    private boolean first = true;

    // a Vector to store derivative plot points
    public Vector returnedVector;
    // a Vector to store testing plot points
    private Vector testVector;
    // an Array to store derivative plot points
    public double [][] returnedArray;
    // an Array to store testing
    private double [][] testArray;

    // constructor for a Vector of DoublePoints
    public Derivative(VectorDoublePointIF inputVector) {
        testVector = inputVector.getVector();
        typecase = 0;
    }

    // constructor for a Vector of Point2D.Doubles
    public Derivative(VectorPoint2dDoubleIF inputVector) {
        testVector = inputVector.getVector();
        typecase = 1;
    }

    // constructor for an Array of Doubles
    public Derivative(ArrayDoubleIF inputArray) {
        int len;
        len = inputArray.getArray().length;
        testArray = new double[len][2];
        testArray = inputArray.getArray();
        typecase = 2;
    }

    public void getDerivative () {
        // each point has an X and Y
        double x1, x2, y1, y2;
        // temp values
        double temp1, temp2;
    }
}
```

```

if (typecase == 0){
    // Vector has objects of type DoublePoint
    DoublePoint p1, p2;
    if (testVector.size() < 2) {
        // vector too small - return vector with one point of 0,0
        returnedVector.addElement((new DoublePoint (0.0, 0.0)));
    }
    else
    {
        // loop throught vector
        for (int i = 1; i < testVector.size(); i++){
            // get the Point2D.Doubles from the vector
            p1 = (DoublePoint)testVector.elementAt(i-1);
            p2 = (DoublePoint)testVector.elementAt(i);
            // get the X's and Y's from the DoublePoint
            x1 = p1.getX();
            y1 = p1.getY();
            x2 = p2.getX();
            y2 = p2.getY();
            // delta Y divided by delta X
            temp1 = (y2 - y1) / (x2 - x1);
            // located at the average of the two X values
            temp2 = (x2 + x1) / 2;
            // add to returned vector
            returnedVector.addElement(new DoublePoint(temp2, temp1));
            // find the high and low values of new vector
            // useful if vector is to be plotted by GraphPanel
            if (first == true){
                highY = temp1;
                lowY = temp1;
                highX = temp2;
                lowX = temp2;
                first = false;
            }
            if (temp1 > highY){
                highY = temp1;
            }
            if (temp1 < lowY){
                lowY = temp1;
            }
            if (temp2 > highX){
                highX = temp2;
            }
            if (temp2 < lowX){
                lowX = temp2;
            }
        }
    }
}
else if (typecase == 1){
    // Vector has objects of type Point2D.Double
    Point2D.Double p1, p2;
    if (testVector.size() < 2) {
        // vector too small - return vector with one point of 0,0
        returnedVector.addElement((new Point2D.Double (0.0, 0.0)));
    }
    else
    {
        // loop throught vector

```

```

for (int i = 1; i < testVector.size(); i++){
    // get the Point2D.Doubles from the vector
    p1 = (Point2D.Double)testVector.elementAt(i-1);
    p2 = (Point2D.Double)testVector.elementAt(i);
    // get the X's and Y's from the Point2D.Doubles
    x1 = p1.getX();
    y1 = p1.getY();
    x2 = p2.getX();
    y2 = p2.getY();
    // delta Y divided by delta X
    temp1 = (y2 - y1) / (x2 - x1);
    // located at the average of the two X values
    temp2 = (x2 + x1) / 2;
    // add to returned vector
    returnedVector.addElement(new Point2D.Double(temp2, temp1));
    // find the high and low values of new vector
    // useful if vector is to be plotted by GraphPanel
    if (first == true){
        highY = temp1;
        lowY = temp1;
        highX = temp2;
        lowX = temp2;
        first = false;
    }
    if (temp1 > highY){
        highY = temp1;
    }
    if (temp1 < lowY){
        lowY = temp1;
    }
    if (temp2 > highX){
        highX = temp2;
    }
    if (temp2 < lowX){
        lowX = temp2;
    }
}
}
}
else if (typecase == 2){
    if (testArray.length < 2) {
        // array too small - return array with one point of 0,0
        returnedArray[0][0] = 0;
        returnedArray[0][1] = 0;
    }
    else
    {
        // loop through array
        for (int i = 1; i < testArray.length; i++){
            // get the X's and Y's
            x1 = testArray[i-1][0];
            y1 = testArray[i-1][1];
            x2 = testArray[i][0];
            y2 = testArray[i][1];
            // delta Y divided by delta X
            temp1 = (y2 - y1) / (x2 - x1);
            // located at the average of the two X values
            temp2 = (x2 + x1) / 2;
            // add to returned vector

```

```

        returnedArray[i-1][0] = temp2;
        returnedArray[i-1][1] = temp1;
        // find the high and low values of new vector
        // useful if vector is to be plotted by GraphPanel
        if (first == true){
            highY = temp1;
            lowY = temp1;
            highX = temp2;
            lowX = temp2;
            first = false;
        }
        if (temp1 > highY){
            highY = temp1;
        }
        if (temp1 < lowY){
            lowY = temp1;
        }
        if (temp2 > highX){
            highX = temp2;
        }
        if (temp2 < lowX){
            lowX = temp2;
        }
    }
}
else
{
    // error
}
}

public double getLowY(){
    return lowY;
}

public double getLowX(){
    return lowX;
}
public double getHighY(){
    return highY;
}

public double getHighX(){
    return highX;
}

public Vector getVector(){
    getDerivative();
    return returnedVector;
}

public double[][] getArray(){
    getDerivative();
    return returnedArray;
}
}

```


ArrayDoubleIF.java

```
/* File ArrayDoubleIF.java */

/**
 * interface ArrayDoubleIF defines the method necessary for
 * DerivativeEngine
 * @version 1.0      TLG
 */

public interface ArrayDoubleIF {

    public double[][] getArray();

}
```

VectorDoublePointIF.java

```
/* File VectorDoublePointIF.java */

/**
 * interface VectorDoublePointIF defines the method necessary for
 * DerivativeEngine
 * @version 1.0      TLG
 */

import java.util.Vector;

public interface VectorDoublePointIF {

    public Vector getVector();

}
```

VectorPoint2dDoubleIF.java

```
/* File VectorPoint2dDouble.java */

/**
 * interface VectorPoint2dDoubleIF defines the method necessary for
 * DerivativeEngine
 * @version 1.0      TLG
 */

import java.util.Vector;
import java.awt.geom.*;

public interface VectorPoint2dDoubleIF {

    public Vector getVector();

}
```

APPENDIX D:
SAMPLE OF AN EXCEL SPREADSHEET

SAMPLE OF AN EXCEL SPREADSHEET

A quick titration of the default acid-base system, 10.00 mLs solution with an unknown molarity of Sulfurous Acid with 0.10M NaOH, was performed and the data stored in an Excel spreadsheet.

pH	Vol	dph/dvol	ave vol	ddph/dvol	ave vol
1.63	0.00	0.140	0.500	0.020	1.000
1.77	1.00	0.160	1.500	0.010	2.000
1.93	2.00	0.170	2.500	0.040	3.000
2.10	3.00	0.210	3.500	0.090	4.000
2.31	4.00	0.300	4.500	0.390	5.000
2.61	5.00	0.690	5.500	3.952	5.813
3.30	6.00	3.160	6.125	404.267	6.200
4.09	6.25	63.800	6.275	0.000	6.300
7.28	6.30	63.800	6.325	-404.800	6.400
10.47	6.35	3.080	6.475	-6.708	6.638
11.24	6.60	0.900	6.800	-0.771	7.150
11.60	7.00	0.360	7.500	-0.190	8.000
11.96	8.00	0.170	8.500	-0.050	9.000
12.13	9.00	0.120	9.500	-0.015	12.250
12.25	10.00	0.038	15.000	-0.003	20.000
12.63	20.00	0.011	25.000	0.000	30.000
12.74	30.00	0.006	35.000	0.000	40.000
12.80	40.00	0.003	45.000		
12.83	50.00				

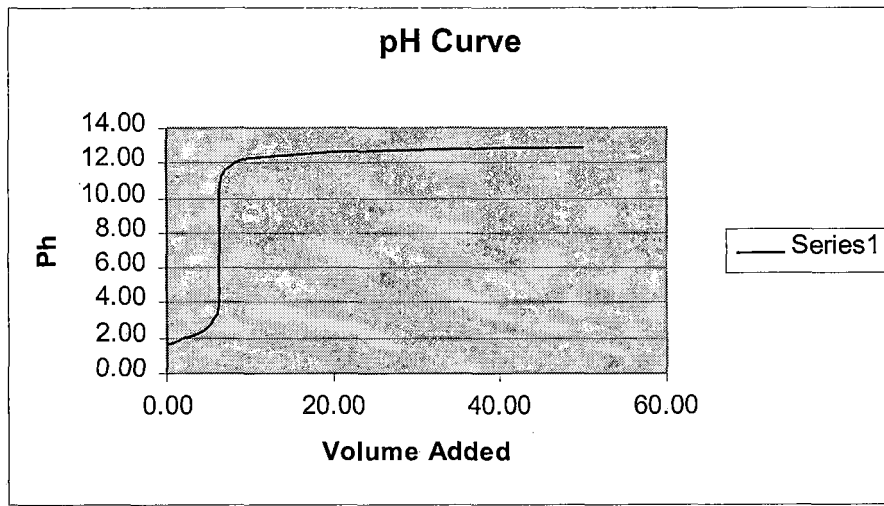
Sample Excel Spreadsheet.

The values recorded during the titration are in the first two columns of the above Excel spreadsheet (pH from the pH meter window and total volume from the volume added window). The next two columns are the first derivative

values and the average of the volume added values. The last two columns are the second derivative values.

PH	vol	dpH/dvol	ave vol	ddpH/dvol	ave vol
A2	B2	C2	D2	E2	F2
A3	B3	C3	D3	E3	F3

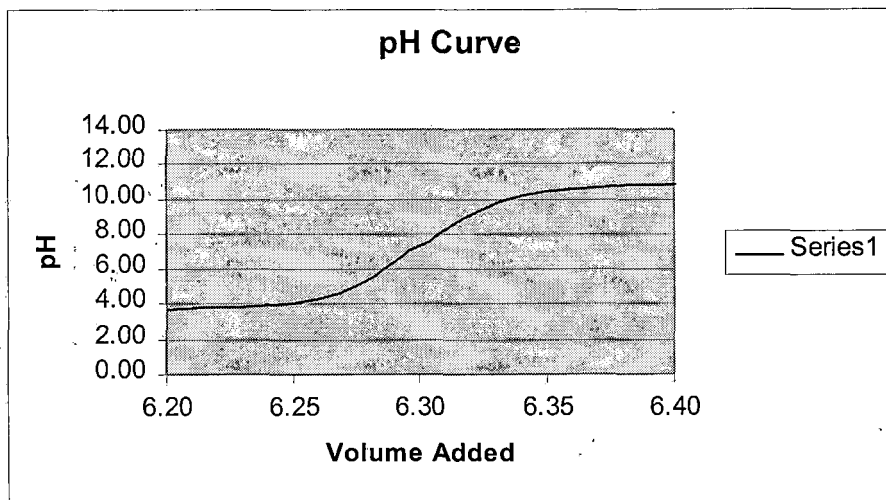
After recording the experimental values in the spreadsheet's A and B columns, dpH/dvol, C2, is set equal to $(A3-A2)/(B3-B2)$ and D2 is set equal to $(B3+B2)/2$. The columns are then filled in using the "Fill Down" command. E2 is set equal to $(C3-C2)/(D3-D2)$ and F2 is set equal to $(D3+D2)/2$. Again, the columns are filled in using the "Fill Down" command. A complete set of instructions on how to make spreadsheets and their charts will not be included in this project. This example is just to show that it is possible.



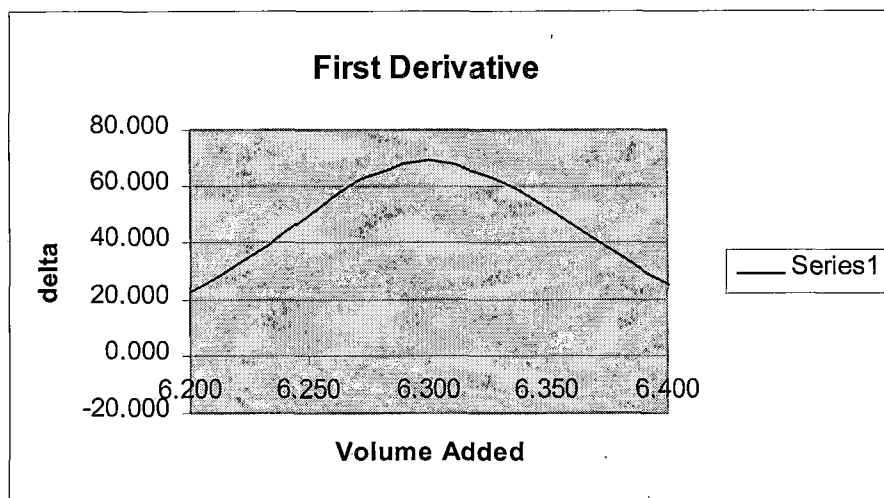
Excel Chart One.

Excel Chart One is an Excel chart of the pH curve created from the first two columns of the spreadsheet. Note that it is impossible to read the endpoint accurately. The area around the endpoint needs to be expanded.

Even if the normal pH curve is expanded, Excel Chart Two, it is hard to see exactly where the endpoint is located on the curve. A better method is to plot the first derivative of the pH curve.

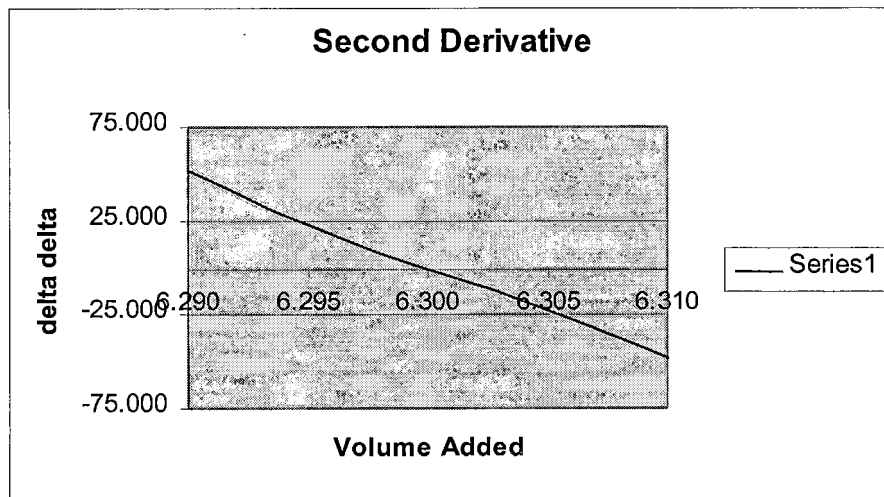


Excel Chart Two.



Excel Chart Three.

Excel Chart Three plots the first derivative of the pH curve, labeled delta, and expands the scale around the endpoint. The endpoint is much easier to locate and measure accurately. The endpoint is the peak of the curve. The peak is between 6.29 to 6.31 mLs but it is still hard to determine exactly. However, the second derivative curve is a better graph and allows a more expanded view.



Excel Chart Four.

Excel Chart Four plots the second derivative of the pH curve, labeled delta delta. When plotted against the volume, the chart shows the endpoint where the line crosses the X-axis. The volume added scale was once again expanded

to show the just the endpoint plus or minus 0.01 mLs. It clearly shows that the endpoint is 6.300 mLs. Remember, that there are only three significant figures in the volume added numbers so the final answer is 6.30 mLs.

Now, with a 10.00 mLs sample and 6.30 mLs of 0.10M titrant, the concentration of the sample can be calculated.

$$\frac{6.30\text{mLs} * 0.10\text{M}}{10.00\text{mLs}} = 0.063\text{M}$$

BIBLIOGRAPHY

1. John A. Dean, *Lange's Handbook of Chemistry* 14th ed., McGraw Hill, New York, NY, 1992; 8.19-8.71.
2. Martin Fowler, *UML Distilled*, Addison-Wesley, Reading, MA, 1997.
3. Daniel C. Harris, *Quantitative Chemical Analysis* 4th ed., W.H. Freeman and Company, New York, NY, 1995; Appendix G.
4. Steven Holzner, *Java 1.2*, SYBEX Inc., Alameda, CA, 1998.
5. Robert de Levie, *A general Simulator for Acid-Base Titrations*, *Journal of Chemical Education* Vol. 76, No.7, July 1999; pp 987-991.
6. Robert de Levie, *Explicit Expressions of the General Form of the Titration Curve in Term of Concentration*, *Journal of Chemical Education*, Vol. 70, No. 3, March 1993; pp 209-217.
7. Software Engineering Standards Committee of the IEEE Computer Society, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE, 1994.
8. Gershon J. Shugar and Jack T. Ballinger, *Chemical Technicians' Ready Reference Handbook Third Edition*, McGraw-Hill, Inc., 1990.
9. Charles Stanton, Javier Torner, and Arturo Concepcion, *GTSS: Generic Tutorial System for the Sciences*, Symposium at California State University, San Bernardino, 23 October 1998.
10. Emily Vander Veer, *JavaBeans for Dummies*, IDG Books Worldwide, Inc., 1997.