WEB-BASED RE-ENGINEERING OF DYNAMIC LAKE WATER QUALITY MODELLING SOFTWARE, MINLAKE

An Abstract of a Thesis

Presented to

the Faculty of Civil and Environmental Engineering Department

University of Houston

In partial fulfillment

of the Requirements for the Degree

Master of Science

in Environmental Engineering

by

Pranam Joshi

August 2002

ABSTRACT

This research demonstrates that complex ecological modeling can be performed using a hybrid concept for computational resources delivery and object-oriented programming paradigm. Traditionally, simulation applications were distributed to a client and the client was responsible for the applications. Infrequently, server-side processing was also used. This research focuses on developing a hybrid simulation application that is run on the client side but maintained on the distribution server. A user may run the application in the client browser. FORTRAN is one of the most used programming tools used in Engineering. Though a trend is observed with an increase in usage of other tools like C and C++, the basic programming paradigm (Procedure-Oriented Programming) still remains unchanged. With advances in computer science and technology, new programming tools and a new programming paradigm have also emerged. This research demonstrates usage of this new paradigm to develop a complex ecological model.

The ecological model used to achieve the research goals is called MINLAKE. MINLAKE is a one-dimensional vertical dynamic lake water quality model developed by St. Anthony Falls Hydraulic Laboratory, University of Minnesota (1987). However this application has several limitations: it is a stand-alone application, it requires independent graphical tools to interpret the data, it has no built-in help provided, and it must be run on a client's machine. In this research, MINLAKE was re-engineered as a web-based application with its own graphical display tool. It has a built-in help menu and can run in the client's browser implementing hybrid computation resource delivery mechanism.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Lake water quality and related environmental models are widely used for estimating the condition of lakes, the effects of lake treatment techniques, the effects of discharges on the lake life, etc. Various assumptions and techniques used to develop lake models are discussed in Section 2.2 and a representative list of such models is presented in section 2.3. Most lake models, even though they use generic assumptions, are tailored to a specific problem. To address a new problem, modifications are required, even to a general-purpose model code. It is unlikely that this approach to modeling will change in the near future, however delivery of computational resources to modelers and researchers can be accomplished in ways that did not exist a decade ago. Additionally, the new programming techniques and languages provide a model developer with several powerful tools to develop and modify a model.

The delivery of computational resources can be accomplished in one of the following ways:

1. Timeshare/server-side processing
2. Workstation/client-side processing
3. Hybrid Server-Client processing

Server-side processing is a delivery mechanism where the analyst prepares data for modeling, sends data to a service center for processing and reviews the results that are generated at the service center. The advantage of this type of processing is that someone else (other than the analyst) is responsible for the maintenance and operation of the model. The analyst is not responsible to invest and purchase expensive machines to maintain and run the code. The biggest disadvantage of this type of system is the high bandwidth requirement for transferring data. The other disadvantages include graphics rendering dependence on server capabilities and data storage requirement on the server. Sometimes this type of communication may take a long time and is likely to include an additional service cost for failed model runs and runs with incorrect data.

Client-side processing is a mechanism where the analyst prepares and runs the data on their local machine. Some of the advantages to client-side processing are the following; 1) communication to remote systems is negligible, 2) graphics rendering is the analyst's choice, 3) data are stored locally, and 4) the failed model runs and incorrect data runs are billed as analyst's time rather than a direct cost. The disadvantage is that the analyst is now responsible for model code maintenance and functioning. Therefore, the client has to invest in computational resources to perform the modeling.

Hybrid processing is a mechanism where some processing is done on the client machine and some on the server. This mechanism is extremely useful for advanced computational analysis. It can be used to separate a model code that requires extensive computational resources and a model code that can be run locally. Global circulation modeling and advanced fluid dynamic computations are examples where numerical computations are conducted remotely and the rendering of graphics is done locally. In fact, hybrid processing is quite routine where specific machine capabilities are used for parts of the computation (vector and parallel processors for numerically intensive work). The hybrid approach can eliminate most of the disadvantages of other schemes except for the communications requirement.

For client-side processing in most modeling applications, the program must be loaded and compiled for a particular processor and operating system. Sometimes, even an executable code is provided for the users to run on their machines. The distribution is generally handled over the Internet. This distribution system is considered to be the current state-of-practice for this research. The distribution or implementation of machine specific code is one of the limitations observed in this and all other traditional approaches. For client-side processing, the distribution of code is machine/operating-system specific and similarly for server-side processing, the source code is machine/operating-system specific for the server. This limitation is also observed in the hybrid approach.

The limitation of hybrid approach can be attributed to the programming tools, rather than delivery systems. Several programming tools like FORTRAN, C, C++, Visual FORTRAN, JAVA, Visual Basic, etc. are available. With the exception of JAVA, all the other programming applications generate a machine/operating system-specific executable file. JAVA, on the other hand, is an interpreted language and so the code written using JAVA is write-once-use-all code. It is neither machine-specific nor operating system-specific. Any machine or operating system loaded with a JAVA interpreter (JAVA virtual machine) can run the code.

The sever-side approach can be implemented using several tools like HTML combined with Common Gateway Interface (CGI), Active Server Pages, Java Server Pages, JAVA SERVLETS, etc. Although this approach is preferred only for specific types of applications (e.g. applications requiring extensive computational resource which all the users cannot afford to purchase or maintain), it has its own advantages. The analysis would always use the current version of the modeling system, assuming that the system is only available on single server. Issues related to secure connections and storage space for analyst's data are significant, and if the user base of the system were large, this approach would become overwhelming.

The approach adopted in this thesis is a hybrid approach coupled with device independent programming technique (JAVA). The modeling tool is still maintained on the server-side, but every time a user wants to run the application, the source code is transferred over the server to the user's browser and executed on the local machine. The model code is device independent and functions similarly with different machines and operating systems. Because of several security or technological limitations, the database is still maintained on the server. JAVA's applet technology does not easily support writing data on the user side. In order to write the data on the client's machine, "signed applets" have to be generated. The signed-applets are applets having digital signatures of the programmer who has constructed the application. The responsibility falls with the server maintenance group and the software developer to secure the code. However, a malicious programmer can use this technology to hack into the user's machine. With such security concerns and limitations in the technology, the user database is maintained on the server-side. Hence, the data resides on the server-side and code is executed on the client-side. The principle advancement in this model is the uniqueness of the code. There is only one copy in existence at any instant in time. The responsibility for the code maintenance rests with the server administrator; however, the data input and processing actually occur at the client's machine. This hybrid approach is not new, but to the author's knowledge, it has never been tried for a complex ecological or engineering modeling system. Using the device independent approach in the hybrid model allows the client to use any machine that has a JAVA enabled browser to perform the modeling. Additionally, JAVA allows development of an object-oriented application.

The purpose of this research is to demonstrate that complex ecological or engineering calculations can be conducted using the hybrid approach and to document the problems involved in such an implementation. This research also explores the development of a complex ecological or engineering application with an object-oriented programming paradigm. Object-oriented programming paradigm is explained in section 2.4.

The ecological model to be re-engineered is MINLAKE, a dynamic lake water quality model. It is a one-dimensional lake water quality model developed for the state of Minnesota. St. Anthony Falls Laboratory, University of Minnesota developed this model in August 1987, to understand the short-term dynamics of the lakes in Minnesota. The lack of understanding of short-term dynamics of the lake was considered responsible for the failure of several lake treatment methods. MINLAKE facilitated understanding short-term hydrodynamic and bio-chemical simulation for lake water quality. In the present version of MINLAKE, short-term dynamics (i.e. vertical hydrodynamics only) are modeled and provisions are made to incorporate bio-chemical simulation components and lake treatment components. MINLAKE has evolved as MINLAKE95, MINLAKE96 in the corresponding years 1995 and 1996. The present version of MINLAKE in FORTRAN programming language is called LAKE36 and in JAVA programming language it is called JAMINLAKE. LAKE36 and JAMINLAKE were developed as a part of this research.

Among all the state variables that define the state of a lake, temperature is the most important one. The modeling of all other variables is dependent on the temperature variable and a minimum feedback effect of other variables is observed on temperature itself. In the present version of JAMINLAKE, only the temperature modeling is performed. This temperature modeling includes heat flux across the water body, density stratification, natural convection and effects of wind mixing. The effect of inflow and outflow is not considered on the thermal stratification. However, a provision is made to incorporate this in later revisions of the model. MINLAKE is representative of a fairly complex ecological model incorporating numerical methods like finite difference and Gaussian Elimination Method. Thus, it is a good challenge to demonstrate the hybrid-computing concept.

The goals of the overall project were achieved by dividing the project into four distinct tasks.

1. Development of Internet-Shared Computing Environment:

   This is similar to server-side computing. This task was deemed necessary to develop the skills to handle the data communication for subsequent tasks. Here, the simulation is written in the FORTRAN programming language. A wrapper application is developed to share the simulation component over the Internet without downloading it on the user's machine.

2. Development of simulation components for device-independent coding:

   This task involves comprehensive analysis of lake-water quality simulation models to identify each distinct physical and biochemical model component that can be treated as a separate object in the device independent framework. This analysis ensures that there are minimum shared resources between different objects. The end product is developed using JAVA because it allows writing machine-independent code and developing an object-oriented application.

3. Development of proper communication among model components:

   This task involves forming a scheme for simulation and defining interactions between different objects involved into lake water quality simulation.

4. Testing the system:

   This task verifies that the new application can produce identical results as the original (reference) application.

2. LITERATURE REVIEW

Lake treatment techniques are typically selected based on various assumptions about lake dynamics, treatment goals, time span, etc. Several lake models have been generated to analyze effects of various lake treatment techniques. Lake models can be as simple as a single completely mixed reservoir model and can be as complex as a 3-D lake treated using computational fluid dynamics (CFD). As a rule of thumb, a simpler model reduces the chance of misinterpreting results and reduces the data and computational requirements [Anderson and Woessner, 1992]. Thus, one should select the simplest model that addresses all the lake modeling objectives. Section 2.1. describes some limnological terms associated with most lake models. Section 2.2. is a review of commonly applied lake modeling concepts. Section 2.3. is a partial list of actual models built using these concepts. Section 2.4. examines the fundamental concepts of Object Oriented Programming (OOP). This examination is crucial in order to understand the philosophy and reasoning behind JAMINLAKE and the entire hybrid-computing concept.

LAKE36/JAMINLAKE reports simulated Temperature and Dissolved Oxygen (DO) in an ASCII text format. A graphical representation greatly facilitates analysis of these results. ML2DPLOT1, a device independent web served graphics package, was also created in this research. Section 2.5. explains the different contouring algorithms, comparison between algorithms and problems associated with grid generation. The selected approach for ML2DPLOT1 application is also explained.

2.1 LIMNOLOGY

Limnology is defined as a study of lakes, ponds, rivers, streams, swamps, and reservoirs that make up inland water systems [Parker, 1977]. The density-temperature relation of water makes it an extremely unusual substance for a compound of its molecular weight. Water has a maximum density at a temperature of 4°C or 39°F. During winter, under ice cover conditions, water temperature increases with depth from O°C at the surface of the lake to 4°C at bottom. The maximum temperature of 4°C, occurs usually at a depth of 3-5 meters for most inland lakes and remains constant below this depth. In the spring, surface water warms until it reaches a maximum density at 4°C and the entire lake has constant temperature (4°C) and density. With this uniform temperature profile there is no density induced resistance to mixing. Wind shear can mix the water of the lake. This mixing allows the nutrients from the bottom of the lake to rise up with the water from the bottom of the lake. It also allows the dissolved oxygen from the surface of the lake to move down to the bottom of the lake with the surface water. This process is called "turn over" of the lake. No mixing of a lake is called amixis, complete mixing of a lake is called holomixis and partial mixing of a lake is called meromixis. Tropical lakes having poor mixing are called oligomictic lakes. Lakes having many mixing periods, even to the extent that they are mixed continuously throughout the year, are called poylmictic lakes. Lakes having one regular mixing period during the year are called monomictic lakes. Lakes having two mixing periods, one in spring and the other in fall, are called dimictic lakes [Parker, 1977].

The thermal profile of the lake controls the ease of mixing and the temperature of different layers in the lake influences modeling of other state variables. Thus, modeling of temperature and heat transfer processes is fundamental in understanding the lake behavior. Within the vertical profile of a lake there are several distinct temperature and mixing regimes. When the temperature increases after winter ice cover, the ice cover of the lake melts and the lake starts getting warmer. As the temperature of upper layer increases, its density decreases and hence it starts forming an upper layer of lighter and warmer water. This upper layer insulates lower layers from heat. This upper warmer layer is called epilimnion. The layer immediately beneath the epilimnion is called the hypolimnion. It is typified by nearly uniform temperature distribution [Scavia and Robertson, 1979].

When there is a distinct temperature gradient between the well-mixed warm upper layer and the uniform lower layer, and the variation is over some depth, the middle layer is called the metalimnion. In the absence of a significant metalimnion, the interface between the two layers is called thermocline. Another operational definition of thermocline is the depth where the temperature decreases to 4°C. The hypolimnion is located below the thermocline.

Light is an important metabolic source of energy in the ecosystem. Light intensity is maximal at the surface, and declines exponentially as the depth increases. Because the amount of photosynthesis accomplished is related to the intensity of light, there is a depth at which the production of organic matter by photosynthesis is equal to its utilization by respiration. This depth is called compensation level [Parker, 1977]. The euphotic zone is the upper, illuminated zone of aquatic ecosystems. It is above the compensation level and therefore the zone of effective photosynthesis.

2.2. LAKE WATER QUALITY MODELS

A lake can be characterized by various attributes often called "state variables". Some of the state variables are temperature, dissolved oxygen, suspended solids, biomass as chlorophyll-a, available phosphorus, detritus as BOD, nitrate-nitrite, ammonia, and internal cellular nutrient levels. The objective of a typical lake water quality simulation model is to compute the value of some or all of these state variables. The complexity of model increases with an increase in the variables and the number of spatial dimensions considered.

The simplest models are *temperature only simulation models* [Riley and Stefan, 1987]. Though temperature is required to simulate other state variables, other state variables have a minimal impact on temperature. Thus a "temperature only simulation" can produce useful results for a relatively low computational cost. As the number of variables increase, complexity increases drastically and so does the computational cost. Additional complexity is added by considering treatment of algal growth and chlorophyll concentration. The more detailed and costly models are only justified when adequate in-lake data are available for the model calibration.

In MINLAKE, phosphorous and light are used as the variables that affect algal growth. A higher level of complexity can be introduced by including nitrogen. The nitrogen cycle is simulated with algae utilizing nitrate-nitrite and ammonium-nitrogen for growth. Thus, the nitrogen cycle adds two state variables to the model. However, many lakes are strongly phosphorous-limited and nitrogen modeling is optional. Though LAKE36 (a derivative of MINLAKE) takes user input for many state variables (e.g. suspended solids concentration, dissolved solids concentration), it does not model them all. However, it simulates temperature and dissolved oxygen (DO). The additional data are collected in anticipation of future model development.

The mathematical models for lake and reservoirs can be broadly classified into two categories:

- Stochastic Models – The models for which state variables and boundary conditions are treated as random variables from known or unknown probabilistic distributions.
- Deterministic Models – These models are designed to produce an explicit statement of the state or condition of the system at particular times and locations when operated with specific input data and under specified external constraints, called boundary conditions. All the models discussed under this section are deterministic models. Deterministic models can be steady state models or time variant models. These models can be zero, one, two or three dimensional depending on the degree of simplification used in representing the prototype [Taub, 1984].

2.2.1. Lakes as completely mixed Systems:

Geometric complexity also plays an important role in modeling. A geometrically simple model is a completely mixed lake model. Such a condition would exist if the state variables have the same value throughout the lake. This model is often classified as an input-output model or a box model. It can be justified for modeling conservative substances like chloride that do not undergo any biological or chemical reaction in the lake [Scavia and Robertson, 1979; Chapra and Reckhow, (b), 1983]. However, this assumption is a gross approximation to most actual lakes because many of these lakes exhibit stratification and the modeled substance is usually non-conservative [Chapra and Reckhow, (a), 1983; Thomann and Mueller, 1987; Scavia and Robertson, 1979]. A different scheme like finite difference should be adopted for such cases.

The completely mixed model is a useful check against more complex models because the total mass balances computed by complete mixing should be close to the more complex methods. Biffi (1963) was amongst the first to apply this approach to chemical substances in the lake.

2.2.2. Complex lake interaction models (Series, Parallel or Series and Parallel):

This model can be considered as a special form of a box model. A lake modeled as reservoirs in series, reservoirs in parallel, or reservoirs in series and parallel, represent an added geometrical complexity. In reservoirs as series model, output from one reservoir is used as input to the succeeding reservoir. In reservoirs in parallel model, input is distributed over all the reservoirs and output from all the reservoirs is cumulated to obtain the total output. Series and parallel model is a complex system of a few reservoirs in series and a few in parallel [Scavia and Robertson, 1979]. A mass balance can be carried out around each reservoir. Using the conservation of mass principles, an equation for concentration into each lake can be obtained [Thomann and Mueller, 1987].

O'Connor and Mueller (1970) were among the first investigators to calibrate and verify successfully the box model approach for predictive water quality in lakes. Figure 1 displays the system that was adopted for modeling. The system consists of two lakes (Superior and Michigan) in parallel configuration. Three lakes (Huron, Erie and Ontario) are in series with this configuration. Each lake is treated as one box, within which the rate of accumulation of mass was determined by the rates of mass inflow and mass outflow.

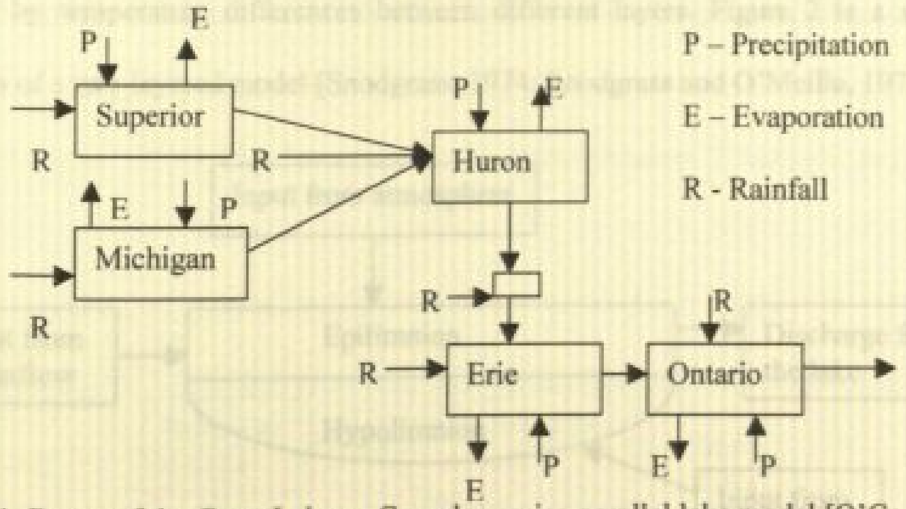


Figure 1. Boxes of the Great Lakes – Complex series-parallel lake model [O'Connor and Mueller, 1970]

2.2.3. Deterministic Finite Segment Models:

Vertical behavior of lakes is of particular importance, especially during periods of stratification. Surface and bottom waters exhibit quite different water quality. Therefore, estimation of vertical gradients is important for a number of water quality problems. In addition, horizontal gradients may occur that are of particular significance such as the "trapping" of waste discharges in a near-shore zone or in the interaction between a cove or embayment of the lake and the lake proper. There are various deterministic finite segment models. Broadly they can be divided into one-dimensional, two-dimensional and three-dimensional models. Some of the specific cases are discussed below.

2.2.3.1. Two-Layer, Stratified Lake Model

Additional limnological complexity is added to the geometrical complexity of this one-dimensional lake model by adding limnological compartments. These compartments are formed by temperature differences between different layers. Figure 2 is a schematic diagram of a two-layered model [Snodgrass, 1974; Snodgrass and O'Meilla, 1975].

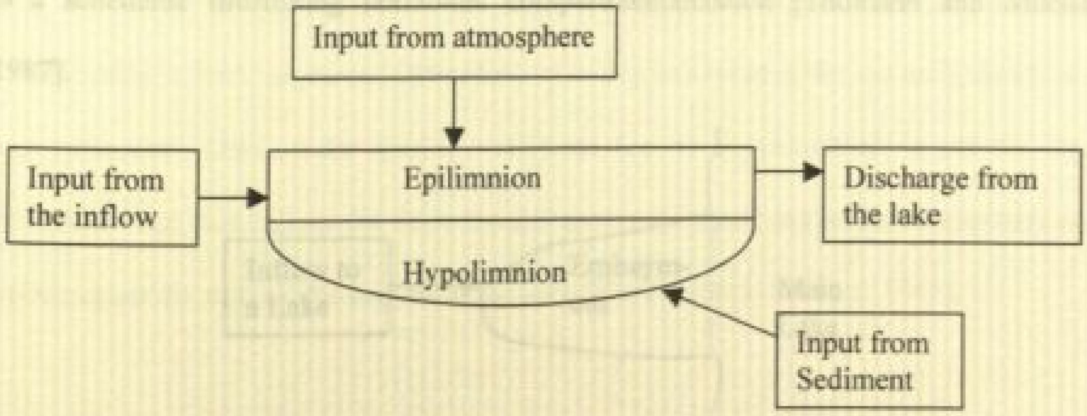


Figure 2. Schematic diagram of two-layer stratified lake.

Though there are normally two distinct layers, during winter such lakes are isothermal and completely mixed vertically and horizontally. Thomann and Mueller, 1987 (204pp.) used such a two-layer model to approximate the temperature profile in Lake Ontario.

2.2.3.2. Multi-Layer, Stratified Lake Model

MINLAKE, and hence Lake36/JAMINLAKE, is based on this model and is explained in the subsequent sections. In multi-layer, stratified lake model, the vertical variation is modeled using finite difference methods. The lake is conceptualized as a one-dimensional continuum of volume elements, each of uniform thickness and bounded by horizontal planes defined by the limits of the impoundment. In general, the model receives flow or allows withdrawal at any level, resulting in vertical transport of mass between volume elements [Taub, 1984].

2.2.3.3. Embayment of Lake Model

It represents the conditions in which a lake receives a discharge and exchanges with the main lake. This condition can be modeled using the finite difference techniques. Figure 3 is a schematic illustrating horizontal compartmentalization [Thomann and Mueller, 1987].

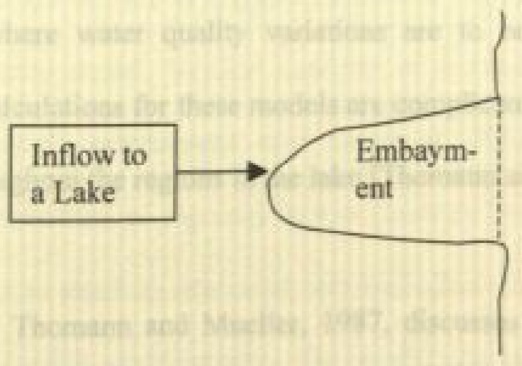


Figure 3. Schematic diagram for embayment of a lake.

A widely known and discussed example of this configuration is Saginaw Bay. It is in the southwest quadrant of Lake Huron. Several studies have been carried out including chloride loading to the lake [Richardson, 1976; Canale and Squire, 1976].

2.2.3.4. 2-Dimensional Lakes Model

The two-dimensional lake system is conceptualized, according to either of the two popular methods, finite difference (FDM) or finite element (FEM), treating it as an assemblage of volume elements. The geometric arrangement differs between the two techniques. FDM requires an orthogonal network of uniform spacing while the FEM affords the flexibility of using elements of variable size and geometry. In the FDM, properties of the flow, such as velocity components and water surface elevations are identified with the centroid of the element, while in the FEM they are identified with the "nodes" situated around the periphery of the element [Taub, 1984].

2.2.3.5. 3-Dimensional Lake Model

The FDM and FEM methods are readily adapted to 3-Dimensional geometries. Additionally, a finite segment (difference) approach can be used for the analysis of lakes and reservoirs where water quality variations are to be calculated in all the three directions. The calculations for these models are complicated and include computation of net transport throughout the regions in the lake [Thomann and Mueller, 1987].

The reference by Thomann and Mueller, 1987, discusses the Multi-Dimensional Lake model in detail. Several lakes including Lake Ontario are briefly discussed with an emphasis to this model.

2.3. LIST OF LAKE MODELING SOFTWARE

The modeling principles discussed in the earlier section are encoded in many software applications for client-side processing. These applications are developed and compiled on the server side and distributed to the client via HTTP protocols or CD media distribution. A representative list is present below.

- The U.S. Army Corps Engineers has developed several lake, river and estuaries water quality modeling applications. These applications are distributed electronically via:

  http://www.wes.army.mil/el/elmodels/index.html#wqmodels

  These are client side applications. The analyst has to download the executable files to appropriate hardware to operate the models.

  - **CE-QUAL-R1**

    CE-QUAL-R1 is a one-dimensional (vertical), stratified reservoir water quality model. Vertical transport of thermal energy and materials occurs through entrainment and turbulent diffusion. The model simulates interactions of physical factors (such as flow and temperature), chemical factors (such as nutrients), and biological assemblages in both aerobic and anaerobic environments. The model can perform stochastic simulations using Monte Carlo methods. Probabilistic estimates of key output variables are provided using statistical data describing biological and chemical effects.

- **CE-QUAL-W2**

  CE-QUAL-W2 is a two-dimensional (longitudinal & vertical), hydrodynamic and water quality model. It is best suited for relatively long and narrow water bodies exhibiting longitudinal and vertical water quality gradients because the model assumes lateral homogeneity. The model has been applied to rivers, lakes, reservoirs, and estuaries. This application can model hydrodynamics, water quality and ice cover of a water body. It can also perform long-term simulations using larger time steps and thus, lower computational cost.

- **TWQM**

  This program computes a steady state, longitudinal distribution of water quality downstream of a reservoir

- **BATHTUB**

  The U.S. Army Corps of Engineers developed a reservoir eutrophication model called **BATHTUB**. Steady state water and nutrient balance calculations are performed in a spatially segmented hydraulic network. These types of calculations account for advective and diffusive transport and nutrient sedimentation.

- **FLUX**

  This program estimates tributary mass discharges (loadings) from sample concentration data and continuous flow records

- **PROFILES**

This application analyzes in-lake water quality data. It includes several response variable calculations for eutrophication. **FLUX**, **BATHTUB** and **PROFILES** are all interrelated programs.

- **CE-QUAL-ICM**

  In **CE-QUAL-ICM**, ICM stands for "integrated compartment model". This method is analogous to the finite volume numerical method. The model computes constituent concentrations resulting from transport and transformations in well-mixed cells that can be arranged in arbitrary one-, two-, or three-dimensional configurations. Thus, the model employs an unstructured grid system. The model computes and reports concentrations, mass transport, kinetics transformations, and mass balances. **CE-QUAL-ICM** is coded in ANSI Standard FORTRAN 77.

- The Wisconsin Lake Modeling Suite (**WiLMS3.3.8**) is a Lake Water Quality Model developed by Department of Natural Resources, Wisconsin. The model can be downloaded from:

  http://www.dnr.state.wi.us/org/water/fhp/lakes/laketool.htm

  WiLMS 3.3.8. is a lake water quality-planning tool. The model uses an annual time step and predicts spring overturn (SPO), growing season mean (GSM) and annual average (ANN) total phosphorus concentration in lakes.

- Open source distribution of software is a new concept similar to freeware distribution application. Unlike freeware, this concept ensures that distribution of subsequent products of an open source code is also an open source product. An example of such a type of system is:

http://iee.umces.edu/AV/Simmod.html

The website contains several different models including statistical models and lake models. But the usage of this system is still related to the client side simulation and the user is forced to download a model.

- US EPA has developed a lake water quality model to carry out tasks including eutrophication analysis. This model is called as **AQUATOX**. This model can be downloaded from:

  http://www.epa.gov/waterscience/models/aquatox/download.html

  AQUATOX is a PC-based ecosystem model that simulates the transfer of biomass and chemicals from one compartment of the ecosystem to another. It does this by simultaneously computing important chemical and biological processes over time. **AQUATOX** can predict not only the fate of chemicals in aquatic ecosystems, but also their direct and indirect effects on the resident organisms. Therefore it has the potential to help establish the cause and effect relationships between chemical water qualities, the physical environment, and aquatic life. **AQUATOX** can simulate the behavior of numerous inter-related components including multiple algal species, submerged aquatic vegetation, benthic invertebrates, zooplankton, fish, etc. It can represent a variety of aquatic ecosystems like vertically stratified lakes, reservoirs ponds, rivers, streams, etc. It can simulate the fate and effects of multiple environmental stressors like nutrients, organic toxicants, etc.

- Water Quality Analysis Simulation Program (**WASP**)

  This model can be downloaded from:

  http://www.scisoftware.com/products/wasp_overview/wasp_overview.html

  Water Quality Analysis Simulation Program (**WASP**), is a US EPA generalized modeling framework that simulates contaminant fate in surface waters. Based on the flexible compartment modeling approach, **WASP** can be applied in one, two, or three dimensions. **WASP5**, a later version **WASP**, is designed to permit easy substitution of user-written routines into the program structure. Biochemical oxygen demand, dissolved oxygen dynamics, nutrients, bacterial contamination and toxic chemical movement have been studied using this application. This model is also a client side model for predicting water quality. User has to download this application in order to use it.

- Cornell Mixing Zone Expert System (**CORMIX**)

  This software can be downloaded from:

  http://www.epa.gov/waterscience/models/cormix.html

  CORMIX is a water quality modeling and decision support system designed for assessment of environmental impact on mixing zones resulting from wastewater discharge from point sources. The system emphasizes the role of boundary interactions to predict plume geometry and dilution in relation to regulatory mixing zone requirements.

CORMIX contains three major subsystems. The first subsystem, **CORMIX1**, is used to predict and analyze environmental impacts of submerged single port discharges to lakes, rivers, and estuaries. The second subsystem, **CORMIX2**, can be used to predict plume characteristics of submerged multi-port discharges. The third subsystem, **CORMIX3**, is used to analyze positively and neutrally buoyant surface discharges to lakes, rivers, and estuaries with a high degree of accuracy.

- **DYRESM**: Dynamic Reservoir Simulation Model

  It is a computer model used to predict the stratification in lakes and reservoirs over seasonal and decadal time scales, making it an important tool for scientists, engineers and managers. It is a one-dimensional model based on an assumption that the variations in the vertical dimension play a more important role than variations in the horizontal direction. So, the lake is represented as a series of horizontal layers that are dynamic in time. It includes heat fluxes similar to MINLAKE. However, it also includes mass fluxes due to rainfall and evaporation. The model can be downloaded from the following web site:
  http://www.cwr.uwa.edu.au/services
  DYRESM was first developed as a research tool in the late 1970s and had several revisions till date. The latest model has been released in December 2000.

As this brief list illustrates, the current state-of-practice is for client-side models. All the programs in this list are certainly adaptable in the hands of a skilled programmer. Like MINLAKE, these programs represent complex engineering or ecological modeling tools.

2.4. CONCEPTS OF OBJECT ORIENTED PROGRAMMING:

Object-oriented programming is often referred to as a new programming paradigm. In the engineering world much of the programming applications were generated using the FORTRAN language. As computer science technology advanced, a gradual transition is observed to a more advanced language like C and C++. However, the basic programming paradigm still remains the same. This paradigm is often times referred to as POP (Procedure-Oriented Programming) or Imperative-programming paradigm. FORTRAN, C, Pascal and other such languages support this paradigm. Prolog supports the logic-programming paradigm. FP or ML languages support the functional programming paradigm [Hailpern, 1986; Budd, 1991]. There are several different ways to visualize this Object Oriented Programming paradigm. It can be viewed as a discreet event driven simulation model. This means for every real object a simulation object is created, inter-relations between objects are defined, and the objects are set in motion. Another way to view object-oriented programming paradigm is to visualize objects as a close resemblance to the actual natural objects considered in the conceptual model.

When computing was in its infancy, a single individual wrote programs usually in assembly language. The tasks were very simple as compared to most computing tasks being performed today. Because regular tasks started becoming more complex, a requirement was felt for higher-level languages. Several languages like FORTRAN, COBOL and ALGOL were developed to provide some features like automatic management of local variables, and implicit matching of arguments to parameters. As the problems started increasing in complexity and size, it started becoming more difficult for a single person to finish the project. Thus, a team of programmers working together to undertake major programming efforts became a commonplace. However, this did not solve the problem. A task that took a programmer 2 months to finish the project was not possible to complete in 1 month with 2 programmers. The reason for this non-linear behavior was complexity, in particular, the interconnections between the software components were complicated and large amounts of information had to be communicated among various members of the programming team [Budd, 1991]. This interconnectedness of one portion of code with the other portions causes large programs to break when even the simplest changes are made. Object oriented programming was developed to address this issue.

Object oriented programming is not simply a few new features added to programming language. It is a new way of thinking about the process of decomposing problems and developing new solutions. There are several features introduced in this paradigm that allows simplification of complex tasks and a better method to share information (or handle communication) between various programmers.

Methods long used in computer science for managing complexity:

- Organization/Behavior Modeling

  This method involves differentiating experience into particular objects and their attributes, differentiating whole objects and their parts, and formation of different classes of objects. In object oriented paradigm this is related to forming the objects and sometimes it is called as object decomposition.

- Abstraction
  - Procedural

    In principle it is similar to functions and subroutines in traditional programming languages. We can consider any operation as a single entity if it achieves the desired results inspite of a need for several lower level operations [Anton, 1994; Budd, 1991].
  - Data

    It includes defining a data-type in terms of the operations that apply to objects, with the constraint that the values of such objects can be modified and observed only by the use of the operations [Anton, 1994].
- Encapsulation

  It is the process of hiding all of the details of an implementation that do not contribute to its essential characteristics. The interface is created between objects in such a way as to reveal as little as possible about its inner workings. An object encapsulates data, and procedures that operate on the data into a single module.

There are various languages that give us tools to construct object-oriented application. C++, Eiffel, Small-talk and JAVA are examples of programming languages that supports object-oriented software development. Though languages like Visual Basic support development of objects, they are not object-oriented languages. They are object-based languages i.e. they do not contain all the features of object-oriented programming paradigm. The essential features for object-oriented application are [Budd, 1991;Anton, 1994;David, et. al., 1992]:

1. The language should support development of classes/objects.
2. A mode of communication between different objects (messaging system) should be established and defined.
3. It should support polymorphism. This means that the same method name can be used in more than one object.
4. It should support inheritance. A subclass should be able to ***inherit*** all the attributes and methods of the parent class. This means that an object **Palm Tree** should be able to inherit all the attributes and methods of a **Tree.**
5. It should support dynamic binding. There are two ways of passing a message. In static binding, a message is always bound to a specific method code based on the static declaration of the type of the receiver variable. In dynamic binding, the runtime system discovers the type of the receiver based on its value at the time the message is sent.

Though C++ gives user a tool to develop an object-oriented application, it depends on the user to create either an object-oriented application or procedure-oriented application. In an object-oriented application the idea is to make objects that depict the universe for the system in consideration. Hence, a better understanding of the system results in better reusable objects. There are several diagrammatic ways for representation of such a system [Anton, 1994].

1. Class diagram/template: This describes functionality of the classes and their relations to the other classes.
2. Object diagram/template – dynamics: This characterizes the dynamic behavior of (a collection of) objects. Hence it describes inter-relationship of objects rather than classes.
3. Module diagram/template: This describes relationship between modules.
4. Process diagram/template – physical design: describes the physical architecture of the distributed systems.
5. State transition diagram – describes behavioral characteristics of the object during a state transition.
6. Timing diagram - describes constraints on the order in which methods may be invoked.

In later chapters, class diagrams for ML2DPLOT1 and JAMINLAKE are shown. Because, methods and functions are rather extensive, they are described separately from the diagram.

2.5 CONTOURING DATA

MINLAKE is a FORTRAN based stand-alone application. The output of this application is written to an ASCII text file. In order to meaningfully understand the results, some graphical display application has to be used. In most cases, the data have to be arranged and the graphical display application plots this rearranged data. MINLAKE does not have any built-in graphical display facility; so, it cannot be used in isolation as a tool for modeling lake water quality. Several different software packages are available for contouring data. However, there is an extremely limited number (if any) of software package for online contour display.

Some of the published and more traditional contouring programs and subroutines are [Watson, 1992]:

- Antoy (1983) FORTRAN program to contour transverse data
- Barrodale and others (1983) FORTRAN subroutine for spectrum analysis
- Baumann (1978) FORTRAN program using inverse distance weighting
- Bourke (1987) BASIC subroutine to contour gridded data
- Braile (1978) FORTRAN subroutine for inverse distance weighting
- Bregoli (1982) BASIC subroutine for line printer plotting
- Davis and David (1980) FORTRAN program for bicubic splines
- Devereux (1985) FORTRAN program using inverse distance weighting
- Dierckx (1980) FORTRAN subroutines for cubic splines

- Dimitriadis, Tselentis, and Thanassoulas (1987) BASIC subroutines for Fourier analysis
- Esler and Preston (1967) FORTRAN program for power spectrum
- Eyton (1984) FORTRAN subroutines for raster contouring
- Inoue (1986) FORTRAN subroutines for cubic splines
- Holroyd and Bhattacharya (1970) FORTRAN bicubic splines
- James (1966) FORTRAN program for Fourier analysis of scattered data
- Kane and others (1982) FORTRAN program for Fourier analysis of scattered data
- Kane and others (1982) FORTRAN program for inverse distance weighting
- Liszka (1984) FORTRAN subroutine for Taylor interpolation
- Mason (1984) BASIC subroutines for splines
- Oldknow (1987) BASIC subroutines for Beizier splines
- Rogers and Adams (1976) BASIC subroutines for surfaces
- Sampson and Davis (1967) FORTRAN response surface
- Swain (1976) FORTRAN program for minimum curvature
- Tartar, Freeman, and Hopkins (1986) FORTRAN subroutines for Fourier analysis
- Watson (1982) FORTRAN program for linear interpolation
- Watson (1983) BASIC program for linear interpolation in stereo
- Watson (1986) FORTRAN subroutine for triangular prism volumes
- Yarnal (1984) FORTRAN programs for gridded data
- Yates (1987) FORTRAN subroutines for linear interpolation
- Yfantis and Borgman (1981) FORTRAN subroutines for Fast Fourier Transform
- Yeo (1984) FORTRAN program for linear interpolation

Several recently available contouring programs are:

- Generalized digital contouring program by Open Channel Foundations. This software can be purchased online at the following web-link.

  http://www.openchannelfoundation.org/projects/Generalized_Digital_Contouring/

- Surfer by Golden Software Inc. More information can be obtained about Surfer as well as Golden Software Inc. on their web site given below.

  http://www.goldensoftware.com/

- A list of contouring applications available for purchase with a free trial on the following web link.

  http://software.geocomm.com/contouring/

- A company called Petrosys develops software related to petroleum industry. One of its products deals with plotting contours. The web link to the product is:

  http://www.petrosys.com.au/product/gridcontour/main.htm

- A contouring algorithm developed in several different languages including JAVA, FORTRAN, BASIC, etc. is available on the following website.

  http://astronomy.swin.edu.au/~pbourke/projection/conrec/

  However, this application is only for contouring and does not grid data.

Though there are several free contouring packages available, a few (if any) software packages are available online. The recent contouring packages discussed above have a web link and can be downloaded from their websites. However, in order to run them, the user must install or at least download them. These free contouring packages are not consistent with the hybrid concept. JAVA allows developing software (applets) that run in the client browser and gets destroyed when the browser is closed. In order to display contours and rationalize the output of LAKE36/JAMINLAKE, ML2DPlot1 contouring package was developed using JAVA's applet technology. Details of the package are explained in section 3.5.1.

Gridding is an essential step of contouring. The next section provides information on various gridding techniques. The Inverse distance technique is used in ML2DPLOT1 and this technique is discussed in detail under Section 2.5.1.

2.5.1. GRIDDING TECHNIQUES

A grid is defined as a regular network or mesh, having a configuration of nodes that have constant spacing in each direction. Gridding is defined as interpolating a raw data set at the nodes of a grid [Watson, 1992]. In our case, the locations of the nodes represent a particular depth (y value) on a particular day (x value). The value of the node represents Temperature/ Dissolved Oxygen.

The first step in grid development is to develop a gird framework (identifying location of grid nodes). This procedure includes identifying the range of data and marking a series of equally spaced horizontal and vertical lines; identifying grid limits (minimum and maximum X and Y coordinates) and the grid spacing (grid steps). The grid spacing can be different in X and Y directions. Figure 4 below is a schematic diagram of a grid. The x interval is Δx and the y interval is Δy. Such a grid forms only the location (x and y values) of nodes. There is no z value associated with each node.

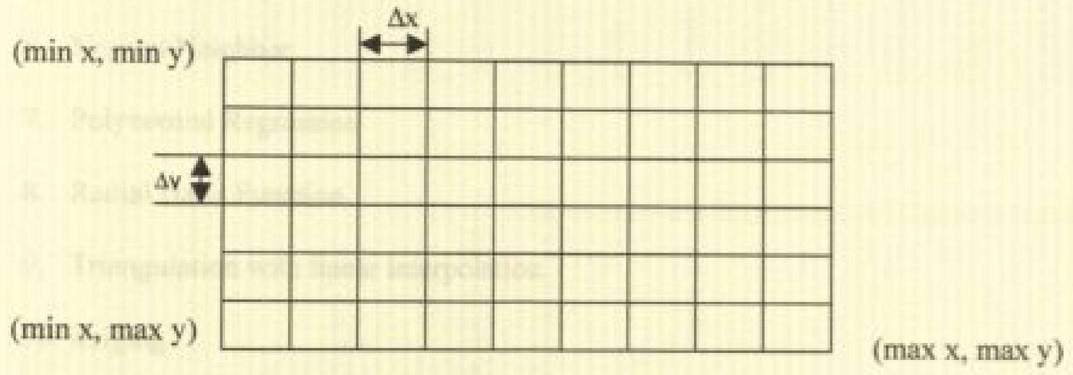


Fig 4. Schematic grid diagram

The next step is to define the Temperature/Dissolved Oxygen value for each point of intersection (node) of the grid. There are various types of computational methods for forming a grid.

a. <u>Independent node calculations:</u>

Independent node calculations include calculation at each grid node. The original dataset is used to calculate all the grid nodes. The calculation at one node is independent of the value at the adjacent node. At each intersection, data to be used are selected. A calculation for the node value is made with the help of these data. There are various techniques for making these calculations. A few are listed below:

1. Linear Interpolation

2. Inverse Distance
3. Minimum Curvature
4. Modified Shepherd's method
5. Natural Neighbor
6. Nearest Neighbor
7. Polynomial Regression
8. Radial Basis Function
9. Triangulation with linear interpolation
10. Kriging

Linear interpolation and inverse distance are examples of weighted averages of the selected data values. Kriging is an entirely different form of gridding. This method accounts for variability in the data being mapped using an empirical semi-variogram. A semi-variogram is a statistical calculation that correlates data variation to distance; it quantifies the expectation that nearby data points should be more similar than distant points.

Table 1 is a list of comparisons between weighted average methods and krigging for the grid formation. A list of advantages and disadvantages for both gridding techniques is discussed.

Table 1. Comparison of weighted average methods and krigging for grid formation [Jones, et. al. , 1942].

Weighted Average Method (Inverse Distance)		Krigging	
ADVANTAGES	DISADVANTAGES	ADVANTAGES	DISADVANTAGES
▪ Speed of computation ▪ Simplicity of programming	▪ Even the simple layouts may not be mapped correctly. For e.g., if several co-planar points are gridded, the map in that region may not show a plane. ▪ The result cannot exceed the highest value and cannot be lower than the lowest value in the dataset. ▪ Inaccurate usage of this method can result into wrong mapping of data.	▪ It takes into account spatial relationships between data points and between data points and the grid node. ▪ Value at each grid intersection has minimum error variance. ▪ Grid of error variances & maps with regions of high-low confidence can be generated	▪ It is extremely time consuming and lengthy process. It takes into account, the whole data set. ▪ A major assumption is that the surface or the variable is stationary without any broad trends. ▪ Surface that dips uniformly over the entire map area could give poor results. ▪ Lack of ability to project values ▪ Can be unacceptable visually because of highly variable contours. ▪ Contour may not honor data values in some cases.

b. Node calculations Spreading from Centers

A second approach to gridding uses previously calculated node values to calculate new values. In this method, calculations spread outward from each data point, filling in nodal values as multiple passes are made over the grid. Values are based on other grid nodes, so they are not independent.

The program first calculates values for the four grid intersections surrounding each data point. This calculation is performed using the previously described procedures of selecting nearby data points and calculating values at the intersections. Surface fits are preferred to averages in order to retain trends or gradients.

After the nodes around all data points are calculated, the data points are removed from further consideration. The program then makes a series of passes over the grid. At each pass it calculates the values for any grid nodes that have not yet been assigned a value and that are adjacent to an assigned node. In other words, each iteration enlarges the calculated region around the original well location. This procedure is similar to above except that node values are considered instead of well values.

This procedure works well with surfaces that are not extremely complex, and generally honors data with little difficulty. It also has the advantage of allowing trends to be projected, although caution should be used because extrapolations can become extreme more rapidly than with independent calculations.

2.5.2. COMMON PROBLEMS ASSOCIATED WITH INVERSE DISTANCE METHOD

General formula used for inverse distance is [Wingle, 1992],

$$g_i = \frac{\sum_{i=1}^{n}\left(\frac{x_i}{d_i^p}\right)}{\sum_{i=1}^{n}\left(\frac{1}{d_i^p}\right)}, \qquad (1)$$

where

g_i is the estimated value at the grid location,

d_i is the distance between the grid location and the sample data,

p is the power to which the distance is raised.

However, if p =1 then it becomes a case of simple linear interpolation.

There are several considerations however before this method can be used. Some of them are listed below with a possible solution:

1. Location of grid point between two contour lines and close to one of the lines:

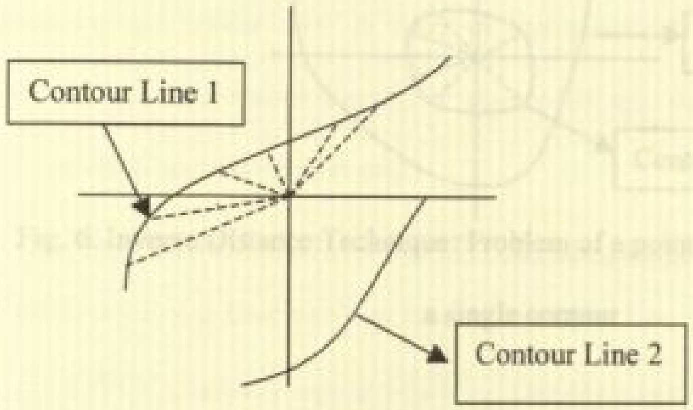


Figure 5. Inverse distance Technique: Problem of location of a grid point too close to a single contour line

Consider a case in which only the nearest 6 points are being considered for inverse distance interpolation. If all the nearest points lie on only one contour line then the grid takes the value of that contour line instead of a value between contour line 1 and contour line 2. Figure 5 is an illustration of such a case.

In order to remove this limitation, the gridding method should incorporate a quadrant search technique. Thus, instead of searching for nearest *n* points, the algorithm should search for nearest *n/4* points in each quadrant. The quadrant search technique ensures that all the contours surrounding a node, contribute in determining the value of the node.

2. A point completely surrounded by one contour

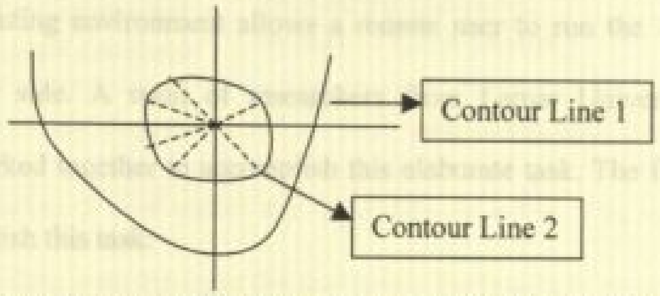


Fig. 6. Inverse Distance Technique: Problem of a point completely surrounded by a single contour

Another problem that typically occurs is at the pits and peaks as shown in the figure above. If one contour line is surrounding a node then it becomes difficult to extrapolate, and the node ends up taking the value of the nearest surrounding contour line. The quadrant search technique can solve this problem only if the surrounding contour does not have a huge number of points in each quadrant.

3. DEVELOPMENT OF INTERNET-SHARED COMPUTING ENVIRONMENT

The Internet shared computing environment allows a remote user to run the LAKE36 application on the server side. A team of researchers from Lamar University and University of Houston worked together to accomplish this elaborate task. The following steps were used to accomplish this task:

1. Modify MINLAKE to access input data by reading ASCII text files

   MINLAKE and later versions have a command line interface. Such applications use STDIN (standard input), STDOUT (standard output) and STDERR (standard error) streams for input/output operations. When such applications are invoked over the web, the associated standard streams are not readily available. Some programming tools, like JAVA, have developed methods to overcome this limitation. However, accessing these standard streams via the Internet is not a desired solution because the server side operations are extensive. A different approach must be adopted. For this research, the input and output operations were carried out using ASCII text files. The derivative of MINLAKE incorporating this operational change is called LAKE36. LAKE36 also contains recent research advances in lake modeling.

   Figure 7 is a schematic diagram of the original MINLAKE application and Figure 8 is the LAKE36 approach that allows a remote user to operate the program. The horizontal lines represent the lines of data communication. As shown in Figure 7, MINLAKE uses command line interface for data communications.

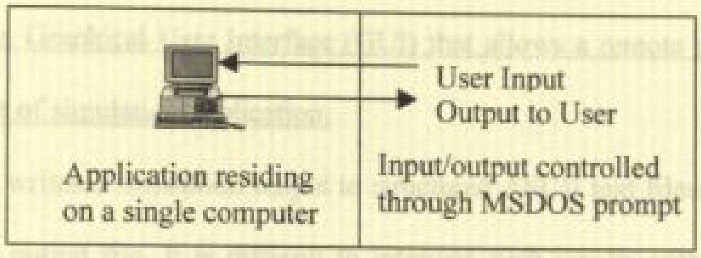

User Input
Output to User
Application residing on a single computer
Input/output controlled through MSDOS prompt

Figure. 7. Schematic diagram for MINLAKE input/output operations.

In LAKE36 application, a user writes the input data to a file. The simulation application uses the data from the input file and writes the results to an output file. Figure 8 is a schematic diagram to illustrate this concept.

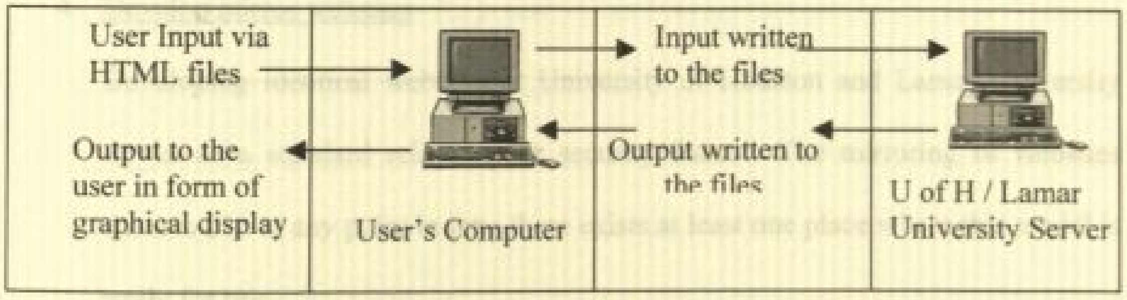

User Input via HTML files
Output to the user in form of graphical display
User's Computer
Input written to the files
Output written to the files
U of H / Lamar University Server

Figure. 8. Schematic diagram for Lake36 input/output operations.

2. Develop a web based wrapper to run LAKE36 over the Internet

This task allows a remote user to connect either to the Lamar University server (http://lakefish.lamar.edu) or the University of Houston server (http://cleveland1.cive.uh.edu/lakeindex.html) and run the simulation application. User input via HTML files (Appendix D) requires setting up a web server (IIS 4.0), developing HTML files that allow a user to provide input data, and maintaining a user database. This MS Access based user database preserves the user's earlier simulation runs and can be accessed using JAVA's JDBC tool.

3. Develop a Graphical User Interface (GUI) that allows a remote user to interpret the output of simulation application.

   LAKE36 writes a simulation output to structured ASCII text files. Appendix C is a sample output file. It is difficult to interpret such results without a graphical utility. Graphical display tool (ML2DPLOT1) was developed using JAVA's applet technology. It displays Temperature/DO (Dissolved Oxygen) profiles, Temperature/DO histories, and Temperature/DO contours.

4. Develop mirror websites

   Developing identical websites at University of Houston and Lamar University serves as a standard reliability or security feature. The mirroring of websites ensures that at any point in time there exists at least one place where this model is ready for use.

The task of developing Internet shared computing environment was divided among several researchers. However, mirroring the websites required active involvement of the University of Houston and Lamar University in all aspects of the research. All the tasks not identified as a University of Houston tasks were carried out entirely by Lamar University. The following is a list of tasks performed at the University of Houston.

1. Conceptualizing possible solutions (JAVA, CGI-scripting, etc.) to allow LAKE36 access on the World Wide Web
2. Developing a mirror website at University of Houston
3. Searching a technique to run FORTRAN application over the Internet

4. Developing input forms for user input
5. Developing ML2DPLOT1 application

3.1 <u>Conceptualizing possible solutions to allow LAKE36 access on the World Wide Web:</u>

There are several different solutions possible for this task. JAVA SERVLETS were used in this research. A few solutions are discussed below:

1. JAVA SERVLETS

   SERVLETS are designed for portability and device independent coding. They are supported on all platforms that support JAVA, and work well with all the major web servers. They are developed and endorsed by JAVA. Thus, this solution provides a total uniformity and consistency in the project. It also provides compatibility with ML2DPLOT1. JAVA allows customized security settings and thus, provides higher security. Sun Microsystems provides a set of classes or objects that give basic SERVLET support. Using SERVLETS simplifies development and deployment of applications like LAKE36. They provide a closer interaction with the server and use multi-threading to manage resources optimally. Thus, this approach was chosen to harness the power of JAVA and its SERVLET technology.

2. COMMON GATEWAY INTERFACE (CGI):

   CGI was one of the first practical techniques for creating dynamic contents and providing access to a single user application via Internet. Using this technique, a web server can pass certain requests to the executable program residing on the server. The output of this program is then sent to the client as an HTML page. CGI scripts can be written either in C, Perl, Visual Basic, etc. When a server receives a request for a CGI program, it creates a new process and then passes necessary information to the process using environment variables and STDIN. Such a request requires time and significant server resources [Jason and Crawford, 1998]. Thus, CGI limits the number of requests a server can handle concurrently.

3. FastCGI

   A company named Open Market developed an alternative to "standard CGI" called FastCGI. FastCGI works in a manner similar to CGI, but it creates only a single persistent process that is reused for each identical request. This feature eliminates a need to create a new process for each request and improves the performance of CGI. However, a few limitations still exist. In the event of concurrent requests, FastCGI must create a new process for each request. Thus, even FastCGI needs a pool of processes to handle concurrent requests. Each concurrent process incurs the overhead of a system call. FastCGI does nothing to help a program to interact more closely with the server.

4. mod_perl

   This option can be used on Apache Web Server to improve CGI performance. It embeds a copy of the Perl interpreter into the Apache *httpd* executable, providing complete access to Perl functionality within Apache. As a result, the CGI scripts are precompiled and executed without forking a new process and thus running more quickly and efficiently. For this research, Internet Information Server 4.0 and JavaWebServer 2.0 are used but not Apache web server. Therefore, mod_perl is not a feasible approach.

5. Server Extension APIs

   Several companies have created proprietary server extension APIs for their web server. Netscape provides an internal API called NSAPI and Microsoft provides ISAPI. Using these APIs, server extensions can be written that enhance or change the base functionality of a server and allow it to handle tasks that were once relegated to external CGI programs. Though this solution is extremely fast, it is not perfect. CGI applications provide significant security and reliability hazards, and are also difficult to develop and maintain. They are specific for each server and operating system. Thus, they do not produce a device independent code.

6. Active Server Pages

   Microsoft has developed a technique for generating dynamic web content called Active Server Pages. An HTML page on the web server can contain fragments of embedded code using ASP. This code is read and executed by the web server and then the HTML page is sent to the client. ASP is optimized to generate small portions of dynamic content. This technique is good for small server side applications. Larger more computationally intensive server-side applications severely limit the number of concurrent users. This research relies extensively on server side processing and so this technique cannot provide an adequate solution.

7. Server-side JavaScript

   Netscape has developed a technique for the server-side scripting called server-side JavaScript (SSJS). Like ASP, SSJS allows small fragments of code to be embedded in HTML pages to generate dynamic web content. However, it suffers the same limitations as ASP.

3.2 Developing mirror website at University of Houston:

The following is a list of tasks to develop a mirror website at University of Houston.

1. Install IIS 4.0 (Internet Information Server 4.0)
2. Install and configure JAVA-WEBSERVER 2.0
3. Copy all the HTML, JAVA and supporting files (e.g. picture files) developed/maintained at Lamar University
4. Establish MS Access Database, add it as an ODBC data-source and provide SERVLETS with an access to this database
5. Change reference in HTML files to University of Houston server instead of Lamar University server
6. Alter programming techniques of various SERVLETS in consideration of the security settings at University of Houston. This step is crucial because the security and computer settings at both the universities are different.

Initially the simulation application was developed using IIS 4.0 and JSDK2.1 (JAVA SERVLET DEVELOPMENT KIT). JSDK2.1 has capabilities of starting the web-service. However, it is only for development use. JSDK2.1 stops the web-service when the user logs off from his or her Windows NT account. JAVA WEBSERVER (JWS) was installed to provide complete web service (SERVLET support). JWS can also support HTML files. However, because IIS 4.0 was already installed and working, JAVA WEBSERVER was used only for SERVLET support.

Figure 9 is a schematic diagram of the working of a SERVLET and its role in the web-based application (detailed explanation of Figure 8). The solid lines represent data communication among several components of the simulation application. It shows a complete data cycle starting from user input via HTML pages to the graphical display in the ML2DPLOT1 applet.

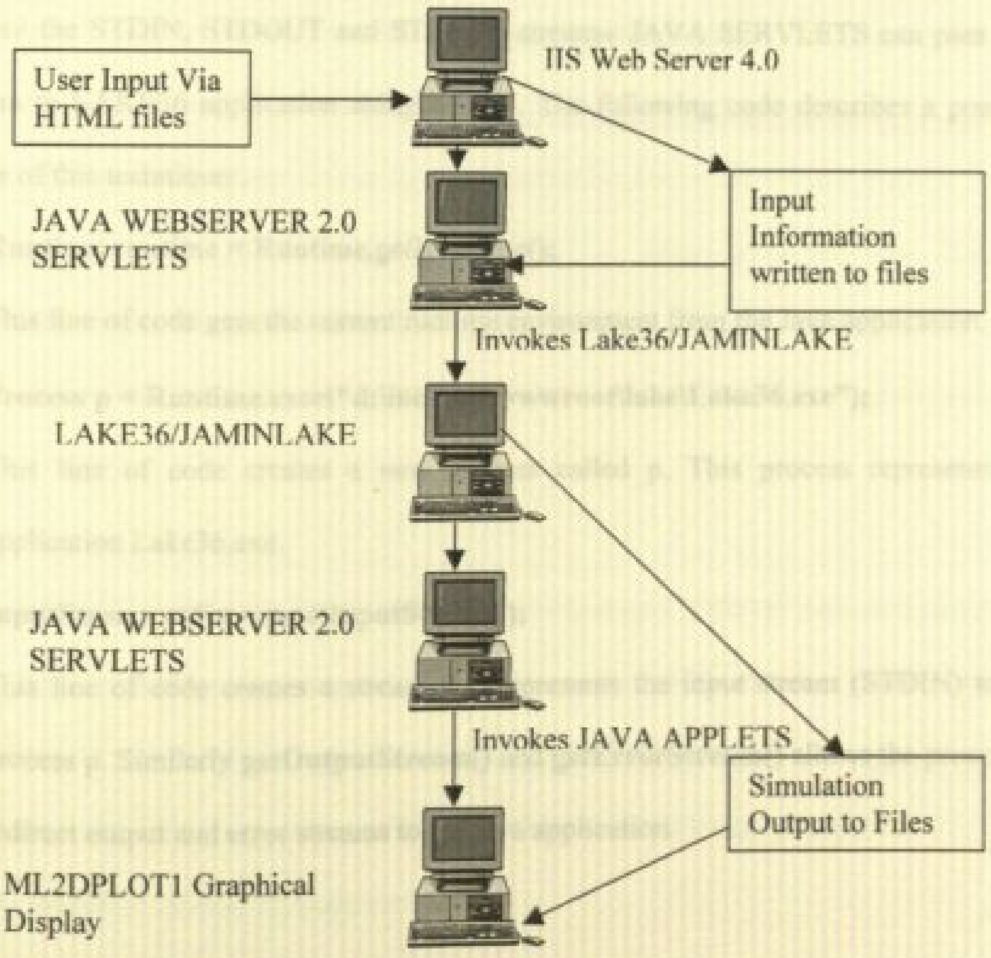

User Input Via HTML files
IIS Web Server 4.0
Input Information written to files
JAVA WEBSERVER 2.0 SERVLETS
Invokes Lake36/JAMINLAKE
LAKE36/JAMINLAKE
JAVA WEBSERVER 2.0 SERVLETS
Invokes JAVA APPLETS
Simulation Output to Files
ML2DPLOT1 Graphical Display

Figure. 9. Schematic diagram for the working of SERVLET and its interactions with LAKE36/JAMINLAKE and ML2DPLOT1

3.3 <u>Searching a technique to run FORTRAN application over the Internet:</u>

The input data are collected through HTML forms and passed to the JAVA SERVLETS. JAVA SERVLETS invoke LAKE36 application and pass the input parameters to

LAKE36. Every programming language provides some manner to handle external executable applications. JAVA provides a **Runtime** class. Every Java application has a single instance of **Runtime** class that allows the application to interact with the operating system. Runtime has a capability to invoke any executable application. An instance of a class called **Process** represents this executable application. **Process** class has an ability to control the STDIN, STDOUT and STDERR streams. JAVA SERVLETS can pass data stream to LAKE36 application using STDIN. The following code describes a possible usage of this technique:

a. **Runtime runtime = Runtime.getRuntime();**

   This line of code gets the current runtime environment from the Java application.

b. **Process p = Runtime.exec("d:\inetpub\wwwroot\lake\Lake36.exe");**

   This line of code creates a new process called p. This process represents the application Lake36.exe.

c. **InputStream stdin = p.getInputStream();**

   This line of code creates a stream that represents the input stream (STDIN) to the process p. Similarly **getOutputStream()** and **getErrorStream()** allows the process to redirect output and error streams to the java application.

3.4 *Developing Input Forms for the User Input:*

HTML pages were developed to provide a GUI for user input. University of Houston was responsible for identifying all the required input data and creating dynamic HTML pages to handle different input conditions. These input forms create input.ini input file.

Appendix B is a sample input file. Lamar University modified these HTML pages to make them user friendly and integrate them into the simulation application.

3.5 Developing graphical output for simulation results:

Graphical display consists of two HTML pages. The first page gives the users a choice to select the display type (Figure 10) and the second page is a graphing applet (Figure 11) for result output. The first page is called MinlakeSimulation SERVLET and the second page is called ML2DPLOT1 applet. ML2DPLOT1 applet plots the data from the structured output file generated by LAKE36 application. There is a separate output file for each year of simulation. The files for temperature and dissolved oxygen simulation are also different.

MinlakeSimulation SERVLET provides a user with several different choices based on the simulation mode (**Temperature only** or **Temperature and dissolved** oxygen). However, if nothing is selected then a standard help page is displayed. For a **temperature only simulation** the following options will be available on the first page (Figure 10).

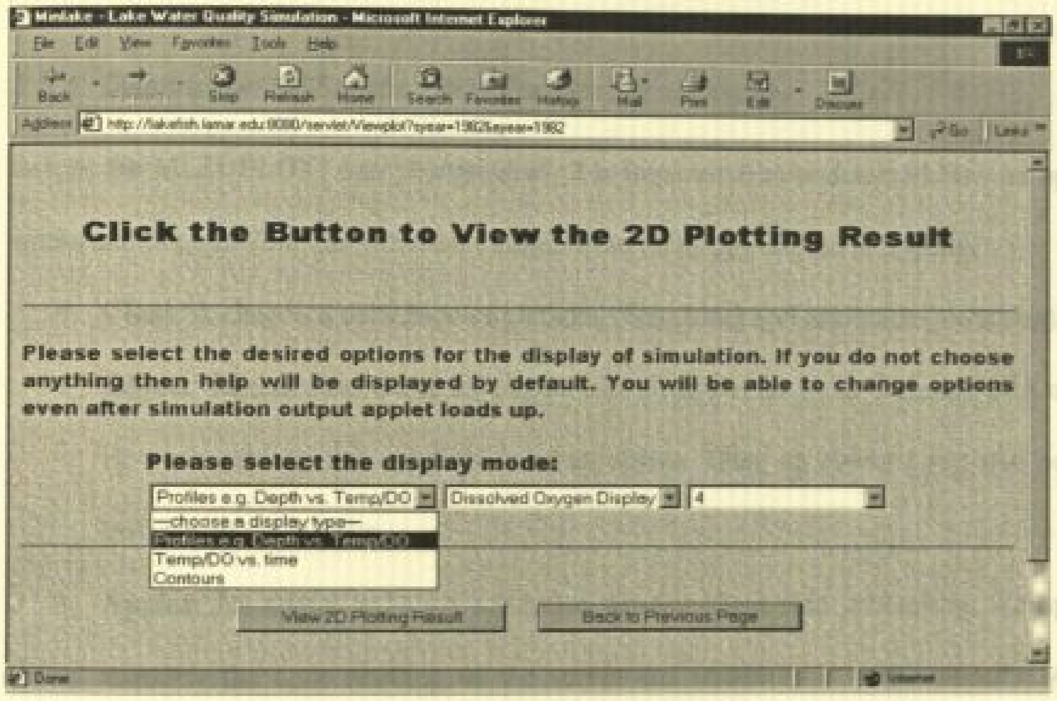
1. Profile or History or Contours
2. Temperature only display.
3. If profile is selected, then choice of day (in Julian days) is available. If history is selected, then choice of a depth is available and if contour is selected, then no further choice is available. These dynamic changes in the web page are carried out using JavaScript.

However, if the user has selected **Temperature and Dissolved Oxygen** simulation, then he/she has the following choices.

1. Profile or History or Contours
2. Temperature display or Dissolved Oxygen display.
3. If "profile" is selected then choice of day (in Julian days) is available. If history is selected, then choice of a depth is available and if contour is selected then no further choice is available. These dynamic changes in the web page are carried out using JavaScript.

An HTML page with the contouring applet is displayed after the user submits the choices. This applet is either displayed with the choices made by the user or a standard help page is displayed.

Figure 10 is a screen capture of MinlakeSimulation SERVLET output. It serves as a gateway to the ML2DPLOT1 application. The user is allowed to choose a display type (contours, profiles, history), a variable (temperature, DO), and a value (day for history and depth for profiles). Figure 11 is a screen capture of ML2DPLOT1 application. It is a display of **temperature only** simulation result.


Minlake - Lake Water Quality Simulation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address
Go Links
Done
Internet

http://lakefish.lamar.edu:8080/servlet/Viewplot?syear=1982&eyear=1982

Click the Button to View the 2D Plotting Result

Please select the desired options for the display of simulation. If you do not choose anything then help will be displayed by default. You will be able to change options even after simulation output applet loads up.

Please select the display mode:

Profiles e.g. Depth vs. Temp/DO | Dissolved Oxygen Display | 4

—choose a display type—
Profiles e.g. Depth vs. Temp/DO
Temp/DO vs. time
Contours

View 2D Plotting Result | Back to Previous Page

Figure. 10. Screen capture of MinlakeSimulation SERVLET

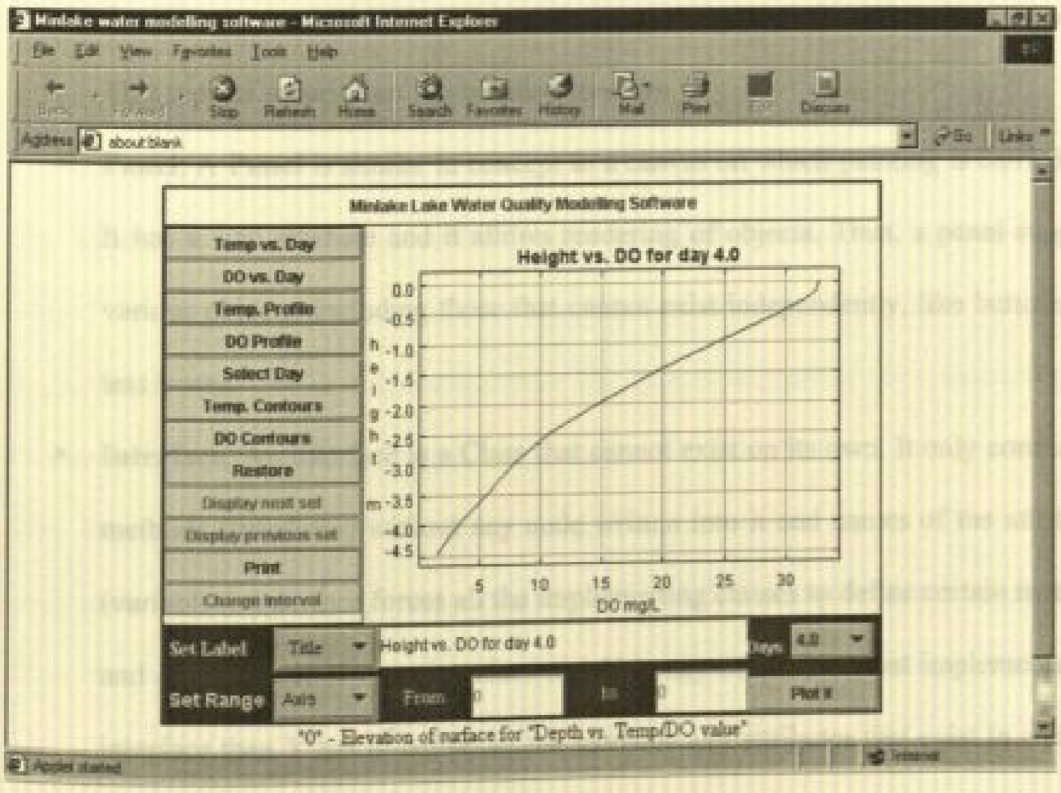

Minlake water modelling software - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address about:blank
Go Links
Minlake Lake Water Quality Modelling Software
Temp vs. Day
DO vs. Day
Temp. Profile
DO Profile
Select Day
Temp. Contours
DO Contours
Restore
Display next set
Display previous set
Print
Change interval
Height vs. DO for day 4.0
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-4.5
height m
5
10
15
20
25
30
DO mg/L
Set Label
Title
Height vs. DO for day 4.0
Days
4.0
Set Range
Axis
From
0
to
0
Plot it
"0" - Elevation of surface for "Depth vs. Temp/DO value"
Applet started
Internet

Figure. 11. Graphical display of ML2DPLOT1. Temperature only simulation outputs

3.5.1. Description Of Source Code For Graphical Display

Before the ML2DPLOT1 can be explained, it is imperative to understand how an applet works and become familiar with some common terminology related to applets.

- **Class**: A class is a collection of methods (functions) and attributes (variables).
- **Object**: In this discussion, an object will always refer to an instance of a class. However, this is not a definition of an object. Thus, an object Lake will be the instance of class Lake.
- **Applet**: An applet is JAVA's solution for the client side processing. An applet transfers the code residing on the server and transmits it to the client's JAVA enabled browser (Internet explorer/ NetScape). The applet executes the code on the client's machine complying with the security setting of the client browser. This type of security setting is called sandbox.
- **Panel**: A **Panel** is similar in concept to a canvas on which painting is carried out. It has a size attribute and it allows rendering of objects. Thus, a panel supports various objects, including those that cannot exist independently, like buttons and text boxes.
- **Interface**: An interface is a Class that cannot exist on its own. It only consists of methods (functions) without any code written into it and names of the attributes (variables). Interface forces all the implementing classes to define certain methods and attributes. If X is an interface and Y is any other class that implements this interface then Y has to define all the methods and attributes that exist in class X. An example of the interface used in the creation of ML2DPLOT1 is DataReader. This interface forces the implementing class to define methods to read the data

from a database. Data are read from ASCII text files in ML2DPLOT1 application. A class called TDOFile is responsible for this job. ML2DPLOT1 object has a reference to TDOFile class and recognizes it as an instance of DataReader. However, if the data have to be read from MSAccess Database, then a new class has to be constructed. ML2DPlot1 should have a reference to that class so that it can be recognized as a DataReader.

For procedure oriented programming a flow chart is an adequate representation of the mode of operation. However, it is well known in the object-oriented literature that flow charts are inadequate for object-oriented application description. A diagrammatic representation is provided using class diagrams. Figure 12 shows a class diagram for ML2DPLOT1 application. Though class diagrams are not useful in analyzing the logical steps in a simulation, they are extremely useful in identifying class interactions.

The lighter shaded classes are interfaces and dark shaded classes are real classes. The dashed lines represent an interface class relationship. This means that the class involved in this relationship implements the interface. The solid double arrow lines connecting two classes indicate some kind of interaction between the two classes or interfaces. A dot-dash line represents parent child relationship. The parent class (where the dashed line does not have a beginning arrow) passes all its attributes and methods to the child class. Thus, the child class can always replace a parent class, however the reverse is not true.

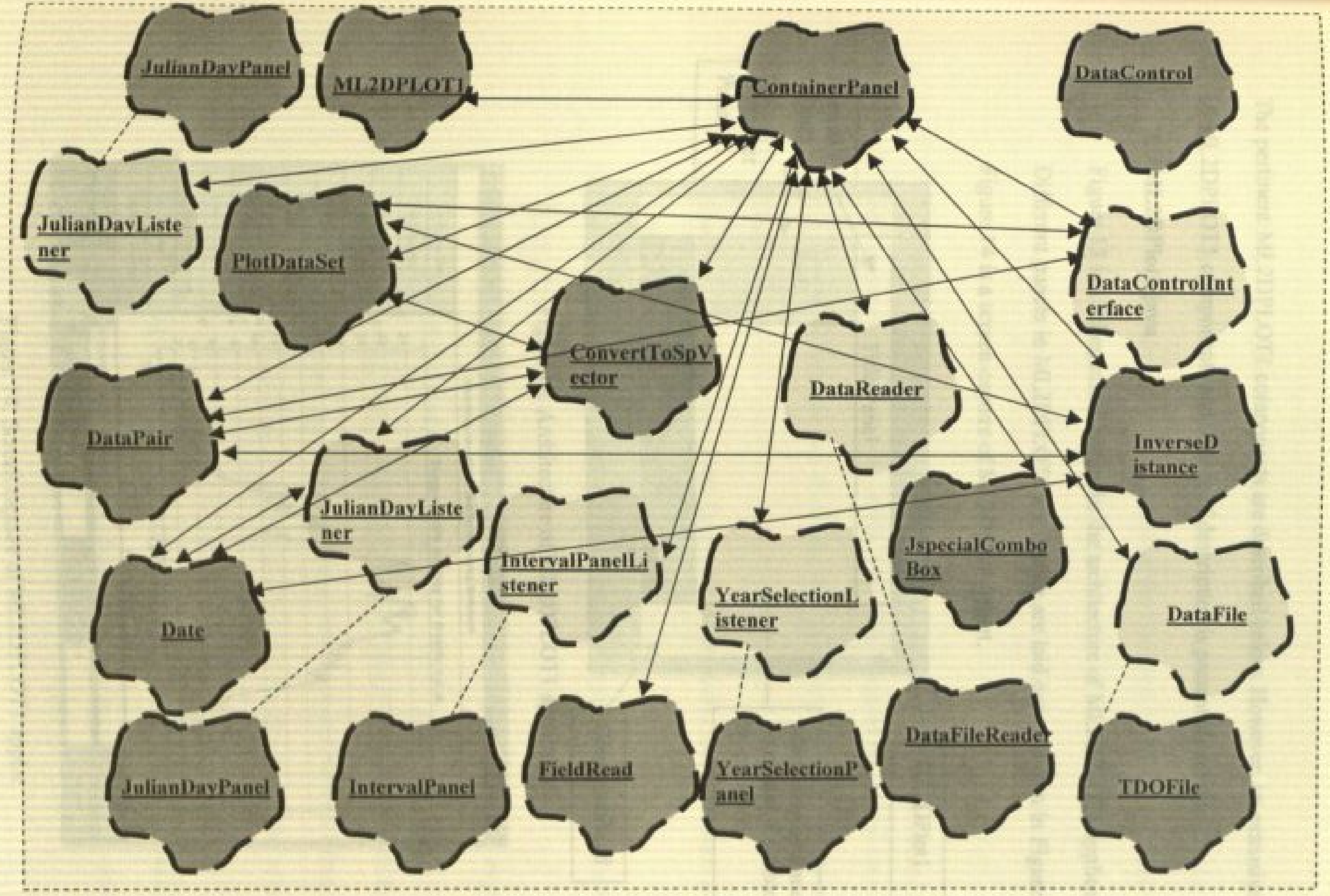


Figure 12. Class diagram for ML2DPLOT1 application. Green classes are actual classes while sky blue objects are interfaces.

The pertinent ML2DPLOT1 components are described below. However, an extensive list of ML2DPLOT1 components and their brief description is presented in Table 2.

1. **ML2DPlot1.java**

   Figure 13 is a schematic diagram for the architecture of ML2DPLOT1 application Different panels in ML2DPLOT1 application are laid out as shown in Figure 13. Figure 14 is a screen capture of ML2DPlot1 applet.

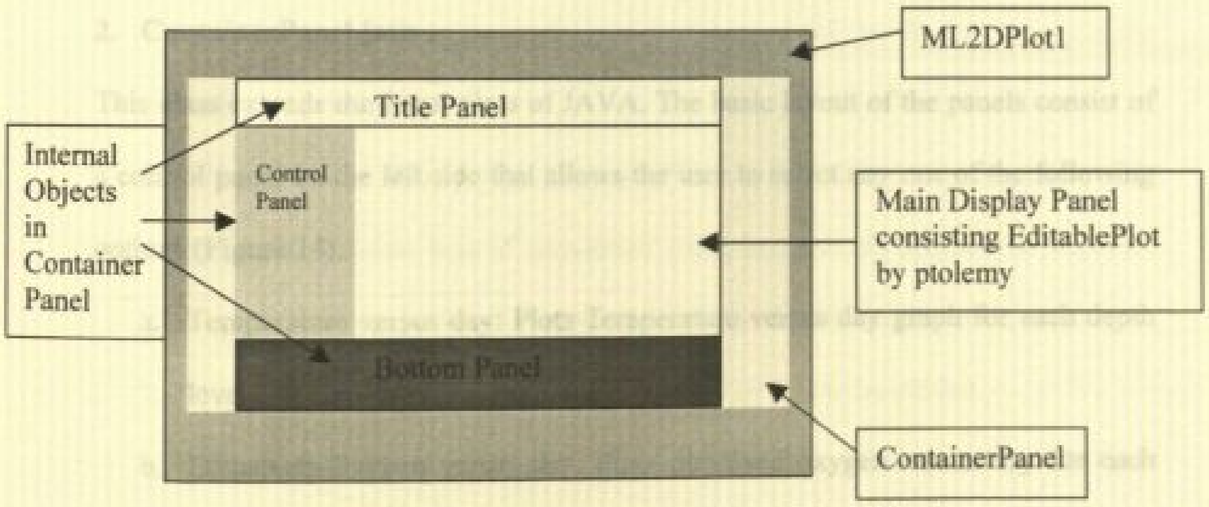

ML2DPlot1
Title Panel
Internal Objects in Container Panel
Control Panel
Main Display Panel consisting EditablePlot by ptolemy
Bottom Panel
ContainerPanel

Figure 13. Architecture of ML2DPLOT1 application

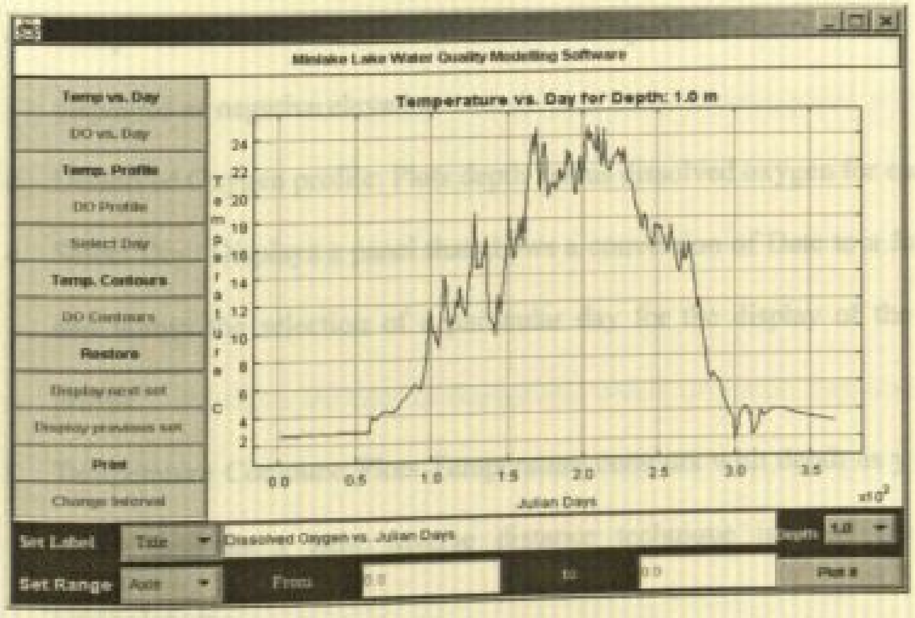


Figure 14. Screen capture of ML2DPLOT1 applet

ML2DPlot1 is a child class of **JApplet**. It means that it inherits all the attributes and methods of the **JApplet** class. This class initiates the applet, creates the **ContainerPanel** and gathers the names of the files that the **ContainerPanel** has to use in order to display results. The names of the files are passed to ML2DPlot1.java through HTML attribute parameter values (PARAM values).

2. ContainerPanel.java

This class extends the Panel class of JAVA. The basic layout of the panels consist of a control panel on the left side that allows the user to select any one of the following options (Figure 14):

a. Temperature versus day: Plots Temperature versus day graph for each depth level

b. Dissolved Oxygen versus day: Plots dissolved oxygen versus day for each depth level

c. Temperature Profile: Plots depth versus Temperature for each day; depth is displayed as negative elevation

d. Dissolved Oxygen profile: Plots depth versus dissolved oxygen for each day

e. Select Day: Displays a panel that shows a conversion of Date to a Julian Day and allows the selection of a particular day for the display of the selected profile.

f. Temperature Contours: Plots Temperature contours with depth as y axis and Julian days as x axis; inverse distance technique is used for contour interpolation.

g. Dissolved Oxygen Contours: Plots dissolved oxygen contours with depth as y-axis and year of the month as x-axis.

h. Restore: The graphing package allows zoom in and zoom out facility by using mouse and selecting regions; restore button restores everything to its initial state.

i. Print: Allows the graphical display to be sent to printer. It seeks user permission to allow the applet and send a print (of the display panel) to the selected printer.

j. Display Next: Displays the next set of data to be plotted. The display is split between different sets of simulation years because the application becomes extremely slow and unreliable for the display of all the simulation years.

k. Display Previous: Displays the previous set of data to be plotted.

l. Change Interval: Displays the dialog box to change contour settings. It allows the user to define contour intervals, remove intervals, add intervals or even change the grid resolution of the contour

The other part of the basic layout is the bottom panel. The bottom panel provides the following features:

a. Set label: It allows the user to make x-axis label or y-axis label or title ready for editing.

b. Set axis: It allows the user to change the x-axis or y-axis. For example, the title of x-axis is altered by selecting x-axis in this drop-down combo box, typing in the new title value in the "display change box" and pressing the return key.

c. Display change box: It is the central box in the bottom panel that allows the user to change title, x-axis label, or y-axis label by typing the new value and pressing the return key.

d. From and to (range): It allows the user to change the range of graphical display. User gives the new range and presses the "Plot It" button to change the range of graphical display.

e. Depth/Days: This is a JSpecialComboBox that allows users to change depth for the "Temp/DO vs day" and days for the "Temp/DO vs depth".

f. Plot It: Allows the users to change the display after changing the settings.

Yet another part of the basic layout is the center panel that holds the graphical display panel. This graphical display panel is developed by University of California under the name of "Ptolemy". Ptolemy accepts a stream of data and plots it. Some minor modifications have been done to the original Ptolemy application. Legends display was modified. Legends were displayed on the right side in Ptolemy. In ML2DPLOT1 legends are displayed on the graph and on each line with the line's own color. One of the major bugs found in Ptolemy was a mistake in naming the vector holding the dataset. Because of this bug, the application threw null pointer exception when more than one line was drawn on the panel. The bug was removed by referencing the correct name of the vector.

The disclaimer and the permission for using this component are located on the following URL:

http://ptolemy.eecs.berkeley.edu/java/ptplot3.1/ptolemy/plot/doc/index.htm

Contouring is carried out using inverse distance method and using a power of 5. However the user is allowed to change the grid resolution. Theoretically any resolution is acceptable. However, beyond a resolution of 200 x 200, the time required for computation far exceeds the increase in accuracy of the contour plots.

3. **IntervalPanel.java**

This class implements the **IntvervalPanelListener** interface. When the **IntervalPanel** is invoked, it displays a panel and lays out the components (Figure 15). The list of numbers on left hand side in the Figure 15 shows the levels for which contours are plotted. The radio buttons with labels ***Interval Value: 1*** and ***Interval Value: 2*** allows changing the contour interval to either 1 or 2. A user can also customize the contour interval by typing the desired interval in the ***Custom Interval*** text box and pressing the return key. The ***Grid Resolution*** button allows a user to change the grid resolution and ***No. of Points*** button allows the user to change the number of points used to estimate grid in the inverse distance method. The default number of points is 20. The ***Add, Remove*** buttons can be used to add or remove specific interval value. The ***Remove All*** button allows a user to remove all the listed contour levels. The **Restore** button allows a user to restore the original contour levels. However, until ***OK*** button is pressed no effect takes place on the actual contour layout. To discard any changes to the contour intervals, the ***Close*** button can be used. The label below the button assembly on the right displays the current resolution and the number of points used to estimate grid.

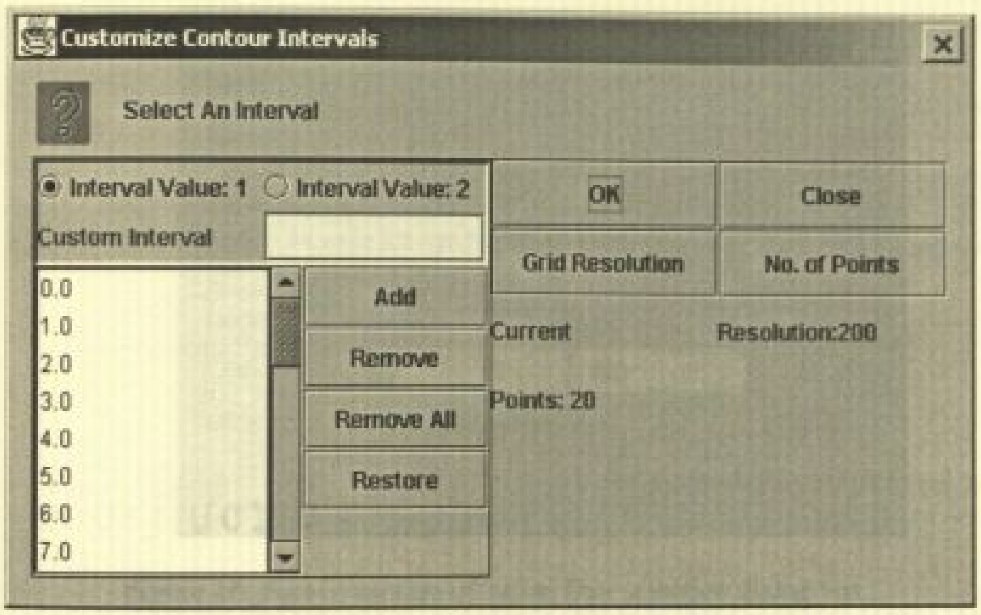
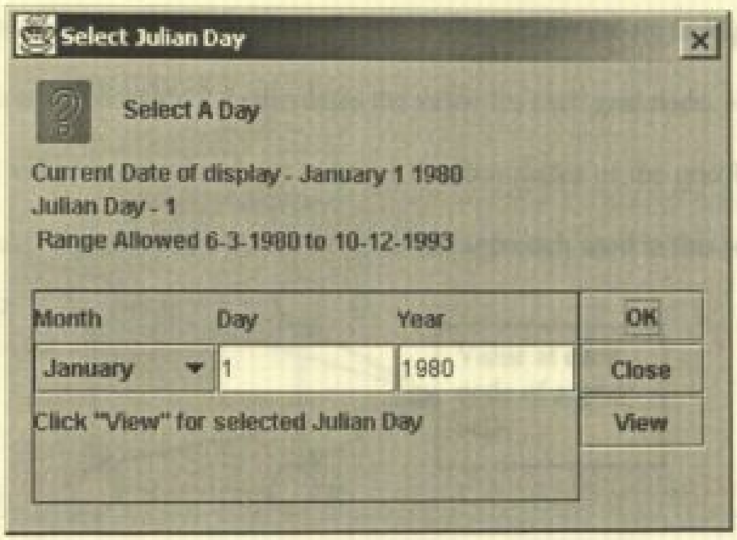


Figure 15. Screen capture of contour Interval selection dialog box

As a user makes a choice, the contour level vector is altered and the panel is destroyed. A new contour plot is generated using this altered contour level vector. If the user decides to close the panel, all the changes in the contour level vector are discarded and the display remains unaltered.

4. JulianDayPanel.java

This class implements **JulianDayListener**. Figure 16 is a captured image of an instance of this class. This class allows a user to choose a Julian Day by defining a month, day and year by Christian calendar. It allows the user to view the present Julian day, to verify the Julian days before selection, and to select a new day for display using ***Month*** combo box, ***Day and Year*** text box. The ***View*** button computes and displays the corresponding Julian day for the date selected. The ***OK*** button changes the display to the selected date.



Select Julian Day

Select A Day

Current Date of display - January 1 1980
Julian Day - 1
Range Allowed 6-3-1980 to 10-12-1993

Month	Day	Year
January	1	1980

Click "View" for selected Julian Day

Figure 16. Screen capture of Julian Day selection dialog box

5. InverseDistance.java

This class is the kernel of contouring calculations. It forms the grid from a raw data set and calculates the value of its nodes. The calculations are based on the contouring technique described in the independent node calculations for Inverse distance algorithm (Section 2.5). The common problems associated with inverse distance technique are eliminated by using quadrant search technique. This class defines two important methods to form contours.

The method computeGrid(Vector xValues, Vector yValues, Vector zValues, int gridResolution, int nearestPoints) allows formation of grid and calculation of its node values. The xValues, yValues and zValues are the raw dataset vectors corresponding to x values, y values and z values of each data point. The value of gridResolution is the number of equal sized intervals for an axis. It is the same for the x and y-axis. The value of nearestPoints defines the number of points influencing the value of each grid node. As shown in Figure 17, the gridding technique under Section 2.5 calculates the value for each grid node. However, in ML2DPLOT1 the value is calculated not only for the four nodes of the grid but also for the center of the grid. Figure 18 is a sketch of modified approach used in this research.

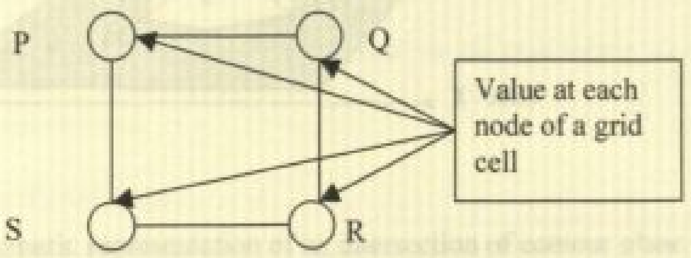


Figure 17. Schematic representation of a single grid cell having contour values for the four nodes

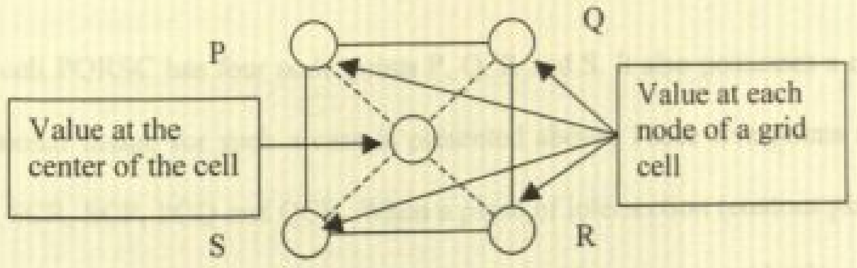

Q
P
Value at each node of a grid cell
Value at the center of the cell
S
R

Figure 18. Schematic representation of a single grid cell having contour values for the four nodes and the center

The method getPlot(Vector _grid, Vector contInterval, int resolution) allows the calculation of plot points. The technique used to draw a contour is to draw a line through every two adjacent data points. This ensures that only the points lying on the continuing contour are connected. Figure 19 is a sketch of 5 points PQRSC (same as in figure 18) on a 3-dimensional axis and a contour plane is forced to intersect this structure.

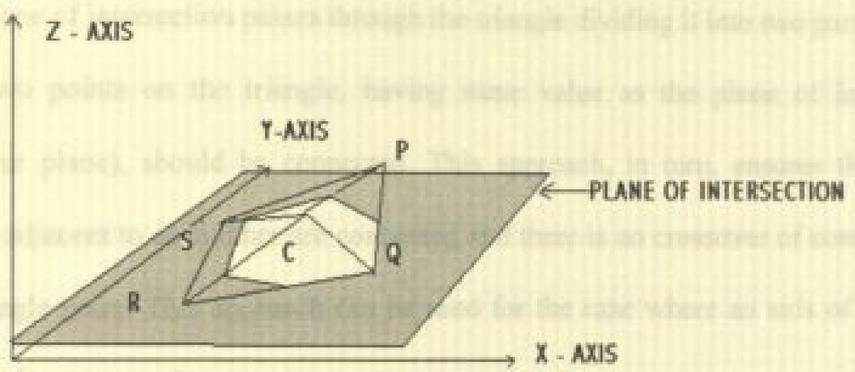


Figure 19. Schematic representation of an intersection of contour plane with a grid cell PQRSC; the shaded portion represents the part below the plane of intersection and the white portion represents the part above it.

The grid cell PQRSC has four node points P, Q, R and S. It also possesses a center C. A 3-Dimensional image for such a case is presented above. These five points make up 4 triangles, SCR, SCP, PCQ and QCR. When a plane of intersection (contour plane) passes through this grid cell one of the following can occur with respect to each triangle:

a. The plane of intersection passes above the triangle.

   As this contour value is not found in the triangle under consideration no line has to be plotted.

b. The plane of intersection passes below the triangle.

   As this contour value is not found in the triangle under consideration no line is plotted.

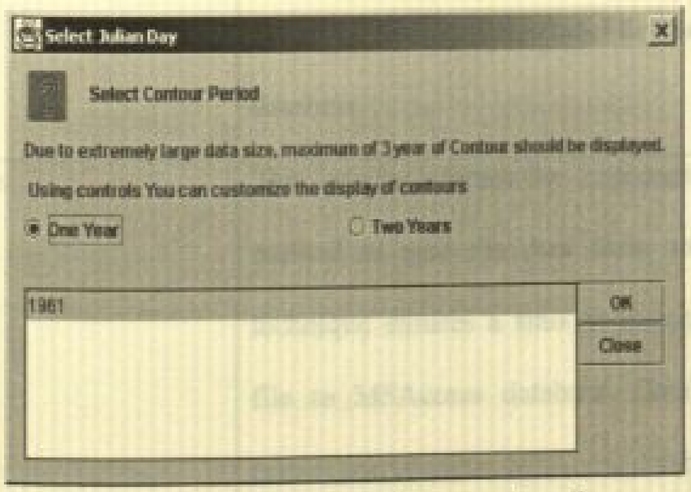
c. The plane of intersection passes through all the three vertices of the triangle.

   This condition defines that the entire triangle lies on the contour plane. This condition also dictates that no line should be plotted.

d. The plane of intersection passes through the triangle dividing it into two parts. The two points on the triangle, having same value as the plane of intersection (contour plane), should be connected. This approach, in turn, ensures that all the points adjacent to each other are connected and there is no crossover of contours over any single point. This approach can be used for the case where an axis of a triangle lies on the contour plane.

6. YearSelectionPanel.java

This class implements **YearSelectionListener**. Figure 20 is a captured image of an instance of this class. This class allows the user to choose a simulated year. When **OK** button is pressed and the panel is closed, the ML2DPLOT1 application will display the contour for the selected year.



Due to extremely large data size, maximum of 3 year of Contour should be displayed.
Using controls You can customize the display of contours
[✓ One Year] Two Years
1961

Figure 20. Screen capture of Year selection panel to allow a user to choose a simulation year for display.

Table 2 lists and describes the remaining components of ML2DPLOT1 application. JAVADOCS and the source code give more information about each of these classes.

Table 2. A list of ML2DPLOT1 components (Components not described above)

ML2DPLOT1 CLASSES	DESCRIPTION
DataReader	This interface forces the implementing classes to open a connection to a database object. The implementing class should contain information on the type of database and the connection.
DataFileReader	It implements **DataReader** interface and defines a method to open a connection to the database. The **DataReader** should refer to the correct **DataFileReader** depending on the database.
DataFile	This interface forces the implementing classes to define a method to read the data from an open connection. This technique ensures a smooth transition from an ASCII text file to MSAccess database. DataReader passes an open connection of either MSAcess database or an ASCII text file to a **DataFile**. An appropriate class implementing **DataFile** reads and stores the data as a **PlotDataSet** object.
TDOFile	It implements **DataFile** interface and is responsible for reading data from an ASCII text file. It returns the data to **ML2DPlot1** (calling object) as a **PlotDataSet** object.

DataControlInterface	This interface forces the implementing classes to transform the data in the desired format, before supplying it to the container panel for drawing.
DataControl	It implements **DataControlInterface** and interpolates data from random depth values (often non-integer values obtained from simulation) to discreet integer depth values.
JSpecialComboBox	It is a special type of drop-down combo box. It stores a list of depth and julian day values. This technique ensures that any change in the selected value of depth or julian day can be easily communicated to the display panel.
PlotDataSet	It is a vector of points stored as a **DataPair** object. It records; 1) min and max values of x, y and z point coordinates, and 2) titles for chart, x-axis, y-axis and z-axis.
DataPair	It has three coordinates corresponding to x, y and z value. It defines several methods to set and access these values.
Date	It corresponds to a particular simulation day and defines methods to calculate julian day from a date and vice versa.
ConvertToSpVector	The data format required by Ptolemy is different from **PlotDataSet** (default format for data storage in ML2DPLOT1). This class ensures the conversion of data format from **PlotDataSet** to Ptolemy requirements. It also facilitates easy search of coordinates for inverse distance technique by setting up data points in form of a hash table.

DataProcessorInterface	This interface forces the implementing class to reconstruct the data for a graphical display. This interface is called just before plotting the data and ensures the correct display.
ReverseAxis	It implements **DataProcessorInterface** and reverses the depth axis. The surface of the lake has a depth of zero and it increases downwards. However, in Ptolemy x and y axis intersection always lies at the origin. And so, depth values cannot increase vertically down. So elevation is used with a minus sign, instead of depth. **ReverseAxis** uses a minus sign to convert depths to elevations.
FieldRead	This class is responsible for collecting the field data and providing them to the display panel.
IntervalPanelListener	This interface forces the implementing class to define a method to change contour level upon a user request.
JulianDayListener	This interface forces the implementing class to define a method to change the present julian day upon a user request.
Grid	This class represents the virtual grid used to contour the data set. Each grid node has x, y and z value. Each grid cell has a center of the grid with x, y and z value.
YearSelectionListener	This interface forces the implementing class to define a method to navigate between different years of simulation.

4. DEVELOPMENT OF SIMULATION COMPONENTS FOR DEVICE INDEPENDENT CODING AND DEVELOPMENT OF COMMUNICATION AMONG THEM

As this task was extremely large, it was divided into two distinct sub-tasks. The first part was to translate from a FORTRAN application to a JAVA application and the second part was to develop an object–oriented application for LAKE36. Lamar University was responsible for first task while the University of Houston was responsible for the second.

1. Develop a direct translation from FORTRAN to JAVA.

   This intermediate application does not have any technological advantage over the LAKE36 application, with the exception that the application was created in a modern language.

2. Develop an object-oriented application for LAKE36.

   Converting a standard Procedure Oriented Programming application (POP) into an Object Oriented Programming (OOP) requires a meticulous understanding of the application and its components. This depth of understanding in turn is an excellent cross check of the original model. Usually a result aberrant from the field data warrants a further check into the model. However, most programmers tend to accept an application if the result closely fits the field data. The exercise of redeveloping LAKE36 as an object-oriented application helped in identifying some errors in LAKE36 itself. Using JAVA to develop this object-oriented technique allowed device independent coding. JAVA is an interpreted language and does not produce executable files. Instead, compilation in JAVA produces class files and these class files are interpreted on individual machines. The interpreter differs from operating system to operating system but the class files remain the same. This technique ensures portability between different machines and operating systems.

4.1. STRUCTURE OF JAMINLAKE

To understand the structure of JAMINLAKE, it is important to understand the modeling principles of LAKE36. JAMINLAKE deals with **temperature only** modeling. The following algorithm provides an overview of computational scheme for establishing a year round temperature simulation of a lake. This entire temperature simulation scheme is recorded with **Minlake** class and is presented in the explanation below. Steps 1 through 5 are used to establish initial conditions and so they are not used for daily simulations. Steps 6 onwards are the steps of daily simulation procedure.

1. <u>Gather input data for lake water quality simulation</u>

   A sample input data set is provided in Appendix B. **DataInputOutput** interface is responsible for this task. **FileOperation** is the class that implements this interface and performs the task.

2. <u>Setup initial lake conditions e.g. number, depth, area and volume of layers</u>

   Lake is responsible to carry out this task. Further details on this task are presented in the explanation of class **Lake.**

3. <u>Setup initial lake temperature profile</u>

   Lake is responsible to set up the initial temperature profile in the lake and thus the temperature of each layer. It is a user supplied temperature profile.

4. <u>Set up initial sediment temperature profile</u>

   Sediment is responsible to establish the initial temperature profile in the sediment. Further details on calculating initial sediment temperature profile are presented in Appendix E6.

5. Calculate mixed layer depth

   Lake is responsible for identifying the mixed layer and recording the location of mixed layer. Further details are presented in explanation of class **Lake.**

6. Determine snow and ice thickness for each day

   Snow and **Ice** are responsible to determine their thickness at the start of each simulation-day. Further details are presented in Appendix E7 and E8.

7. Estimate daily chlorophyll concentration (for each layer)

   ChlorophyllEstimator is responsible for this task. **Estimator** implements the **ChlorophyllEstimator** interface and performs this duty. Further details are presented in explanation of class **Estimator**.

8. Calculate heat flux in each layer and subsequent temperature profile in the lake

   The first layer gets the heat input from atmosphere and so the **Coefficients** class carries out this simulation. The **Lake** is responsible for the heat transmission among different layers in the lake. Further details on temperature simulation scheme are presented in Figure 24 and Appendix E.

9. Correct the temperature profile by calculating the effect of natural and forced convective mixing

   Lake and **Wind** are responsible for this task. Lake is responsible for the natural convective mixing in the lake. However, **Wind** is responsible for the wind mixing induced in the lake. Further details on temperature simulation scheme are presented in Figure 24 and Appendix E.

4.1.1. CLASS DIAGRAM OF JAMINLAKE

As explained in section 3.5.1., class diagrams are adequate representation of object-oriented application. The lighter shaded classes are interfaces and dark shaded classes are real classes. The dashed lines with double arrows represent a class implementing an interface. The solid lines connecting two classes indicate some kind of interaction between the two classes or interfaces. A dot-dash line represents parent child class relationship. The parent class (where the dashed line does not have a beginning arrow) passes all its attributes and methods to the child class. Thus, the child class can always replace a parent class. However, the reverse is not true.

In order to use JAMINLAKE application, some peripheral applications had to be developed. The peripherals include applications; 1) to read data from the files, 2) to use mathematical calculations, 3) to write output of the simulation result, and 4) to compare the results of JAMINLAKE simulation with LAKE36 results. Figure 21 depicts interaction among various packages. Figure 22 is a class diagram depicting interaction among javamodel package of JAMINLAKE application. Javamodel is the most important package in the entire simulation. Only this package is explained in detail.

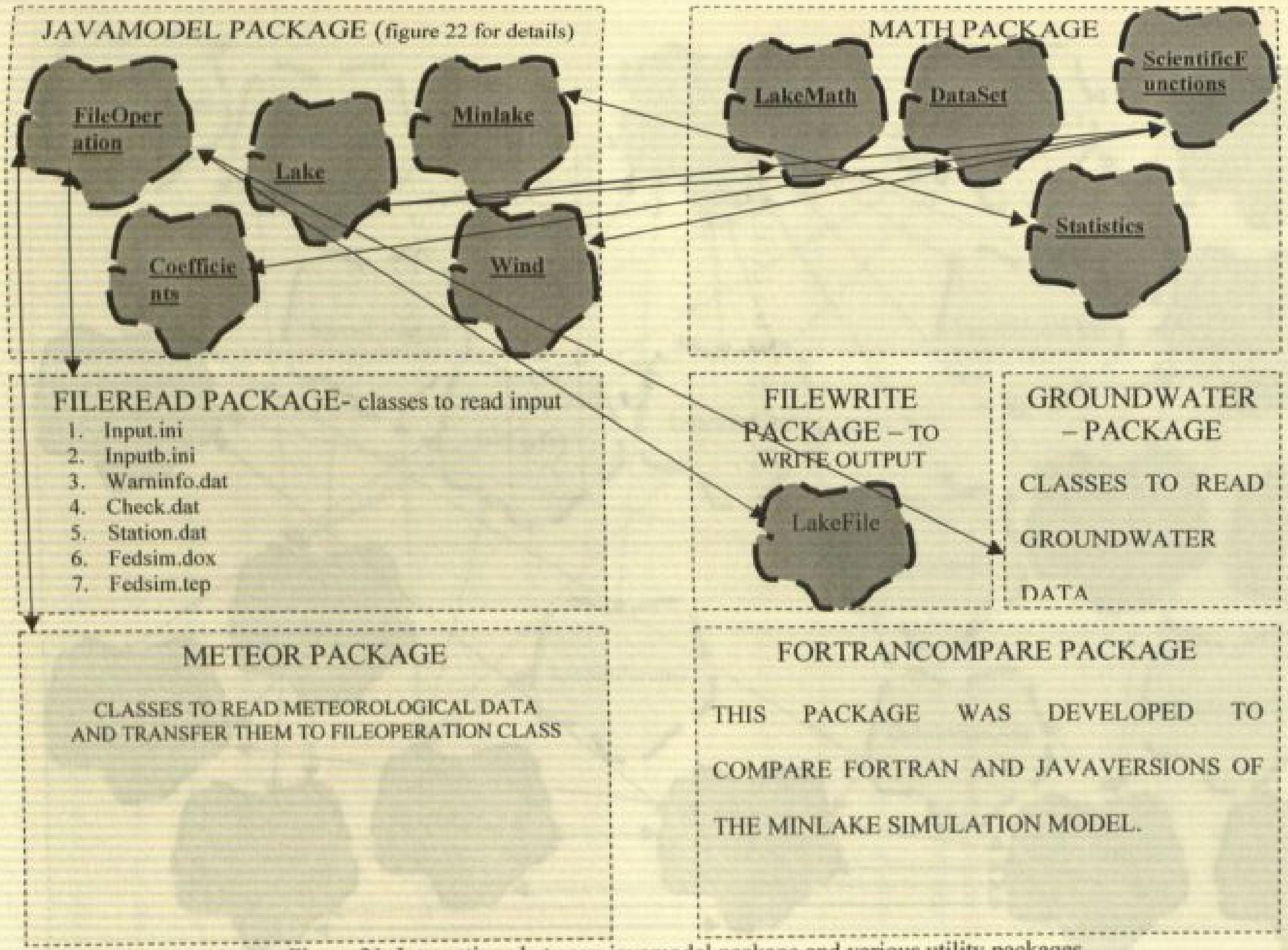

JAVAMODEL PACKAGE (figure 22 for details)
FileOperation
Lake
Minlake
Coefficients
Wind
MATH PACKAGE
LakeMath
DataSet
ScientificFunctions
Statistics
FILEREAD PACKAGE- classes to read input
1. Input.ini
2. Inputb.ini
3. Warninfo.dat
4. Check.dat
5. Station.dat
6. Fedsim.dox
7. Fedsim.tep
FILEWRITE PACKAGE – TO WRITE OUTPUT
LakeFile
GROUNDWATER – PACKAGE
CLASSES TO READ GROUNDWATER DATA
METEOR PACKAGE
CLASSES TO READ METEOROLOGICAL DATA AND TRANSFER THEM TO FILEOPERATION CLASS
FORTRANCOMPARE PACKAGE
THIS PACKAGE WAS DEVELOPED TO COMPARE FORTRAN AND JAVAVERSIONS OF THE MINLAKE SIMULATION MODEL.

Figure 21. Interactions between javamodel package and various utility packages

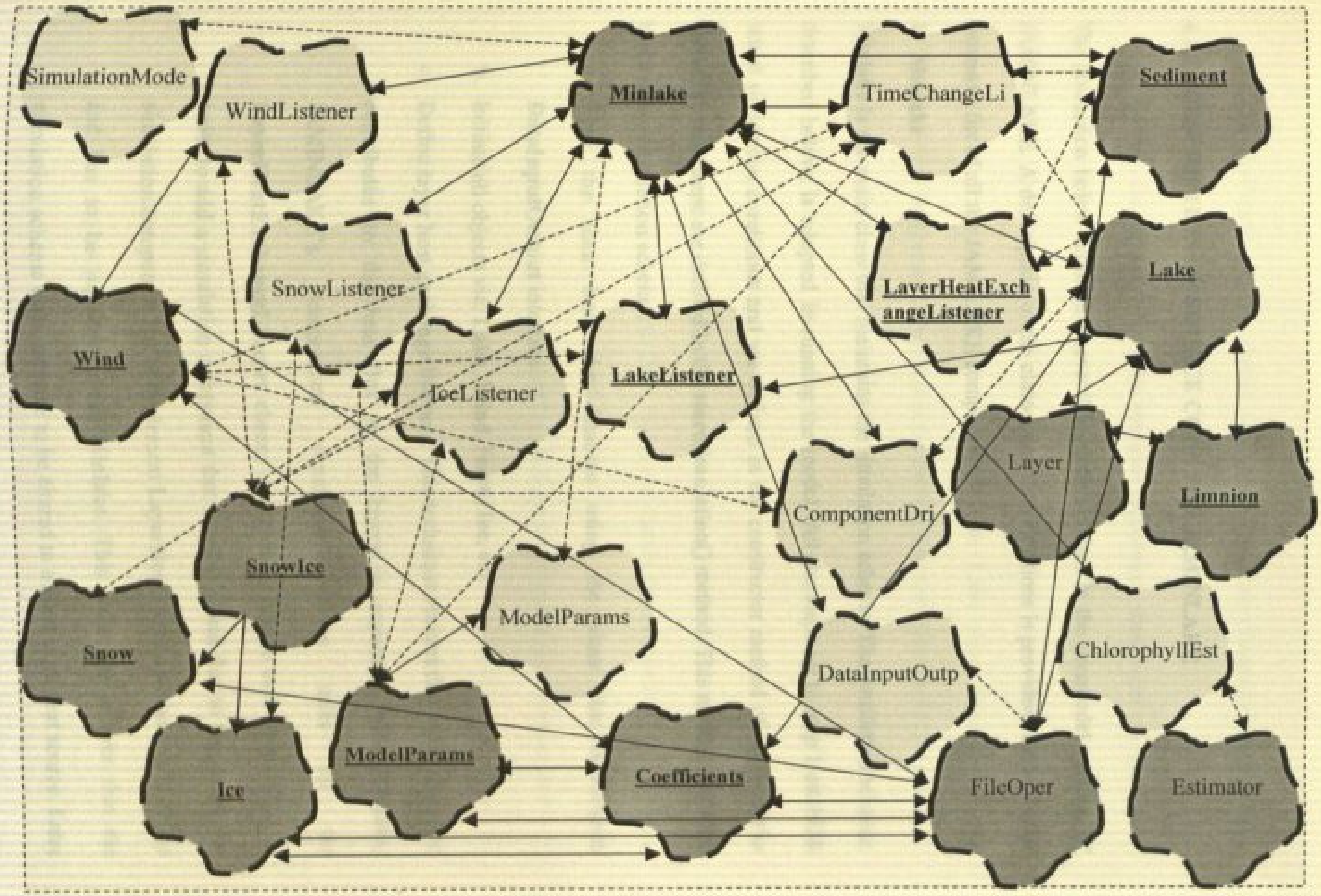


Figure 22. Class diagram for javaversion.javamodel package.

4.1.2. DESCRIPTION OF SOURCE CODE FOR JAMINLAKE

The discussion below highlights several important classes in the javamodel package of JAMINLAKE. A description of the classes and their functions is provided. Table 3 is an extensive list of all the JAMILAKE components.

1. Minlake

Minlake is the main class that maintains the simulation scheme. The simulation scheme, described below, is adopted in running the model. All the steps up to step 1 establish initial simulation conditions and are defined in the constructor method of the **Minlake** class. The next steps are defined in the **startSimulation**() method. This method holds the temperature simulation scheme.

a. Read the data from the database using the class that implements **DataInputOutput** interface.

b. Initiate all objects viz. **Lake**, **Wind**, **Snow**, **Ice**, etc.

c. Define layer heat exchange listeners. This technique ensures that every class that contributes to the heat flux in the layers is registered with **Minlake**. JAMINLAKE's temperature simulation scheme calls all the **LayerHeatExchangeListener** classes to contribute heat to each layer. Thus, in order to add a simulation component that supplies a heat flux to the layers, the simulation component has to implement **LayerHeatExchangeListener** interface and has to be registered with **Minlake**. This technique ensures that the simulation scheme does not have to be altered to add an extra heat source. **Lake** and **Sediment** are the classes that implement **LayerHeatExchangeListener** interface in JAMINLAKE.

d. Set up the layers. This task includes calculating dimensions of the layers, checking for the feasibility of the dimensions and splitting the layers if required. Then, lake dimensions are recalculated.

e. Set the initial temperature profile in the lake. **Lake** object handles this task. ***setInitialTemperatureProfile()*** is the method that is responsible for the task.

f. Set the initial sediment temperature profile, using the scheme described under the class **Sediment**. The method **setInitialTemperatureProfile**() defined in the class **Sediment** is responsible for this job.

g. Register the classes that implement **TimeChangeListener** interface. This technique ensures that all these registered classes are informed of any change in time. Thus, all these classes can keep track of the present date of simulation. The classes that implement this interface in the present model of simulation are:

- **Lake**
- **Wind**
- **Coefficient**
- **Snow**
- **Ice**
- **LakeFileWriter**
- **ModelParams**

h. Facilitate registering **LakeListeners** with the Lake object. The **LakeListeners** are:

- **Wind**

- **Snow**
- **Ice**

i. Facilitate registering **WindListeners** with the **Wind** object. The **WindListeners** are:
   - **Snow**
   - **Ice**

j. Facilitate registering **SnowListeners** with the **Snow** object. The **SnowListeners** are:
   - **Ice**
   - **ModelParams**

k. Facilitate registering **IceListeners** with the **Ice** object. The **IceListeners** are:
   - **Snow**
   - **ModelParams**

l. All the classes implementing **ComponentDriver** interface are requested to pass a reference to all their listeners. **Lake, Wind, Snow** and **Ice** implement **ComponentDriver** interface and pass their own reference to their listeners (e.g. **LakeListener, WindListener, SnowListener, IceListener**).

m. Starts day-to-day simulation. For every simulation day, simulate method of classes implementing **WaterQualitySimulator** interface is invoked. The class that implements this interface in JAMINLAKE is **Minlake**. This scheme is extremely important because it allows various water quality simulation modules to be added. For example, only two changes have to be made if Dissolved Oxygen Simulation has to be carried out. Firstly, a class implementing **WaterQualitySimulator** interface has to be constructed or altered to carry out

Dissolved Oxygen simulation. Secondly, this class has to be registered with **Minlake** as a **WaterQualitySimulator**. Figure 23 describes interaction of various objects in **Minlake** class. Figure 24 describes temperature simulation scheme defined in **simulate()** method of **Minlake** class.

OBJECT INTERACTION DIAGRAM OF JAMINLAKE

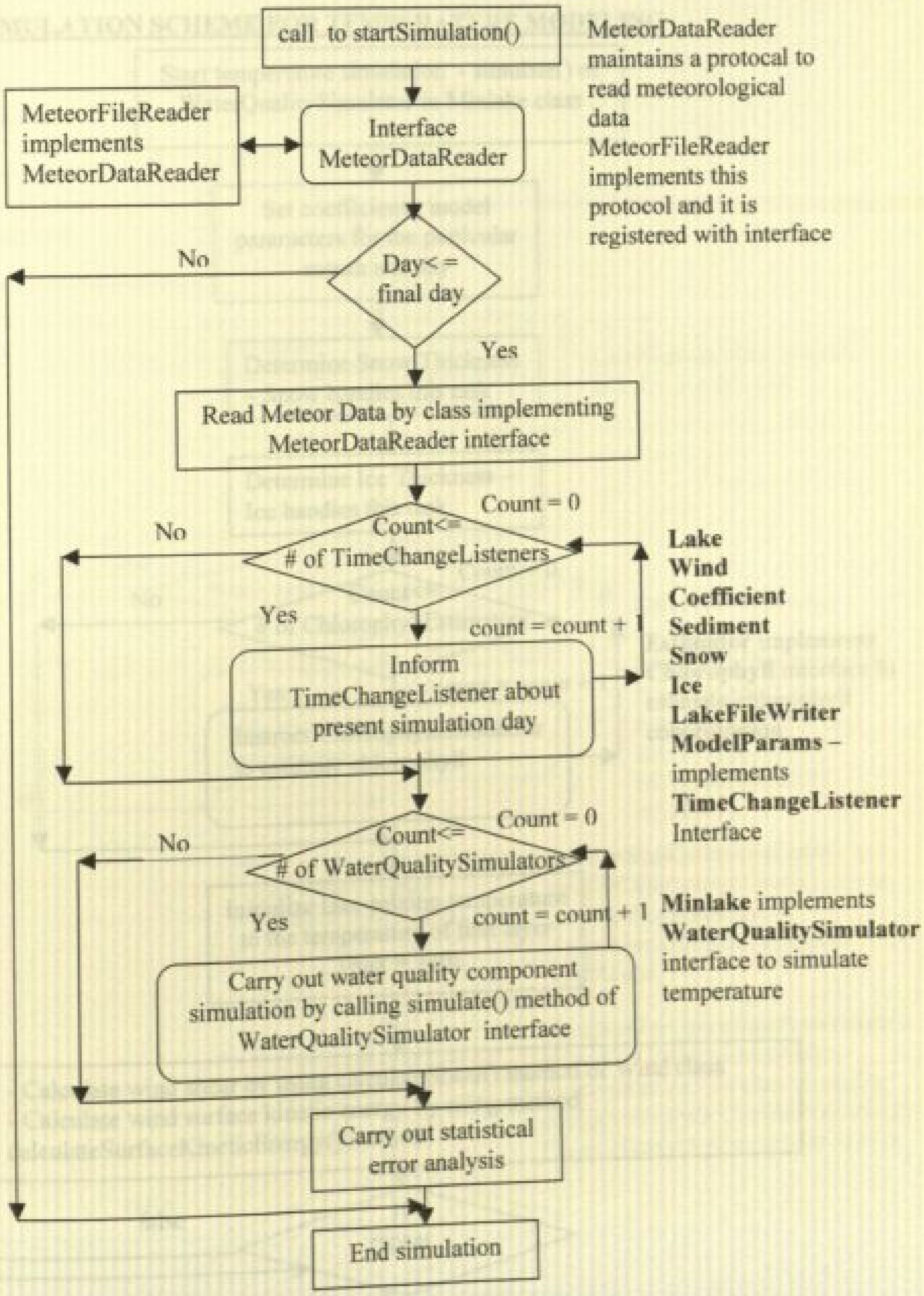
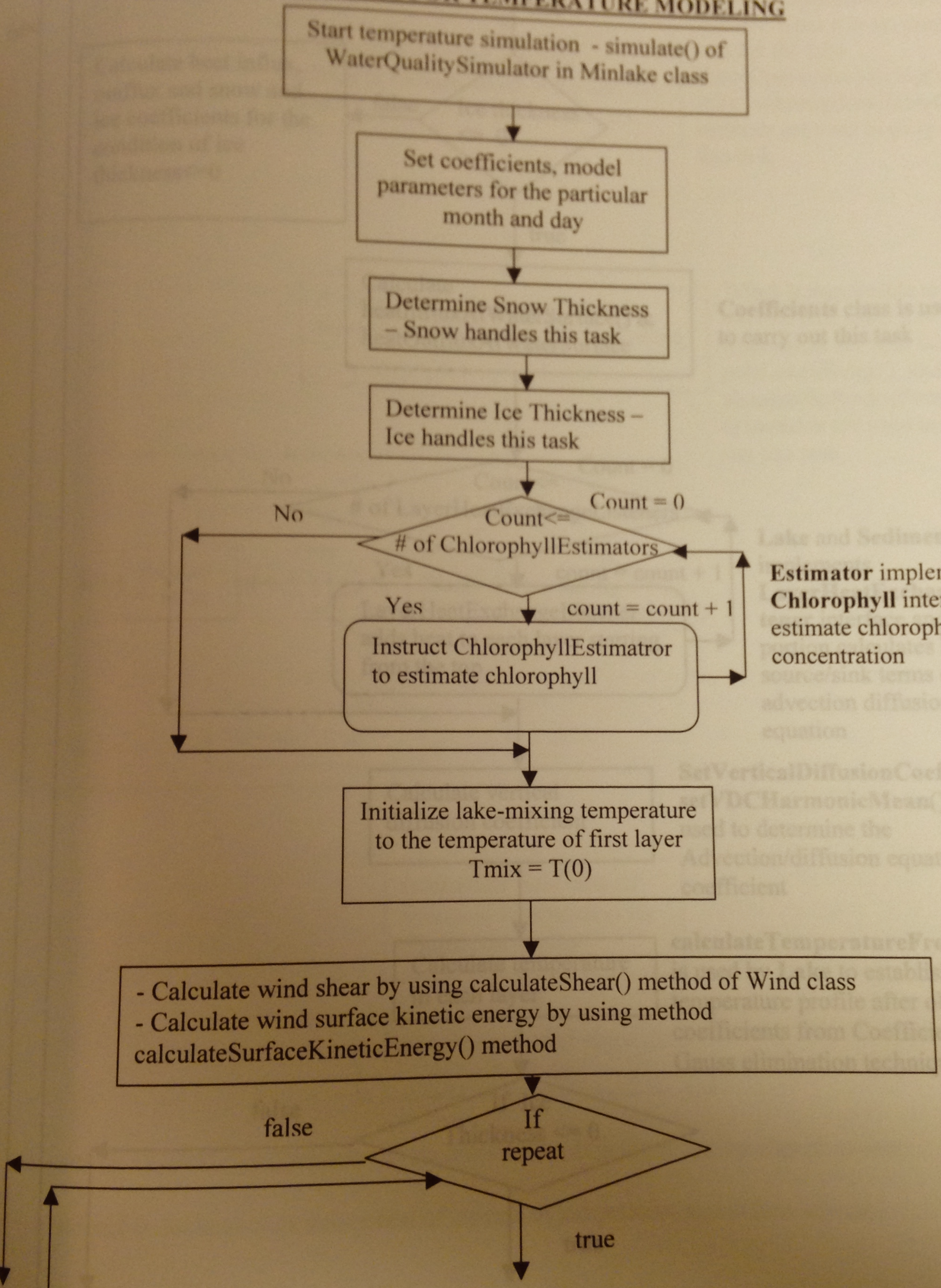
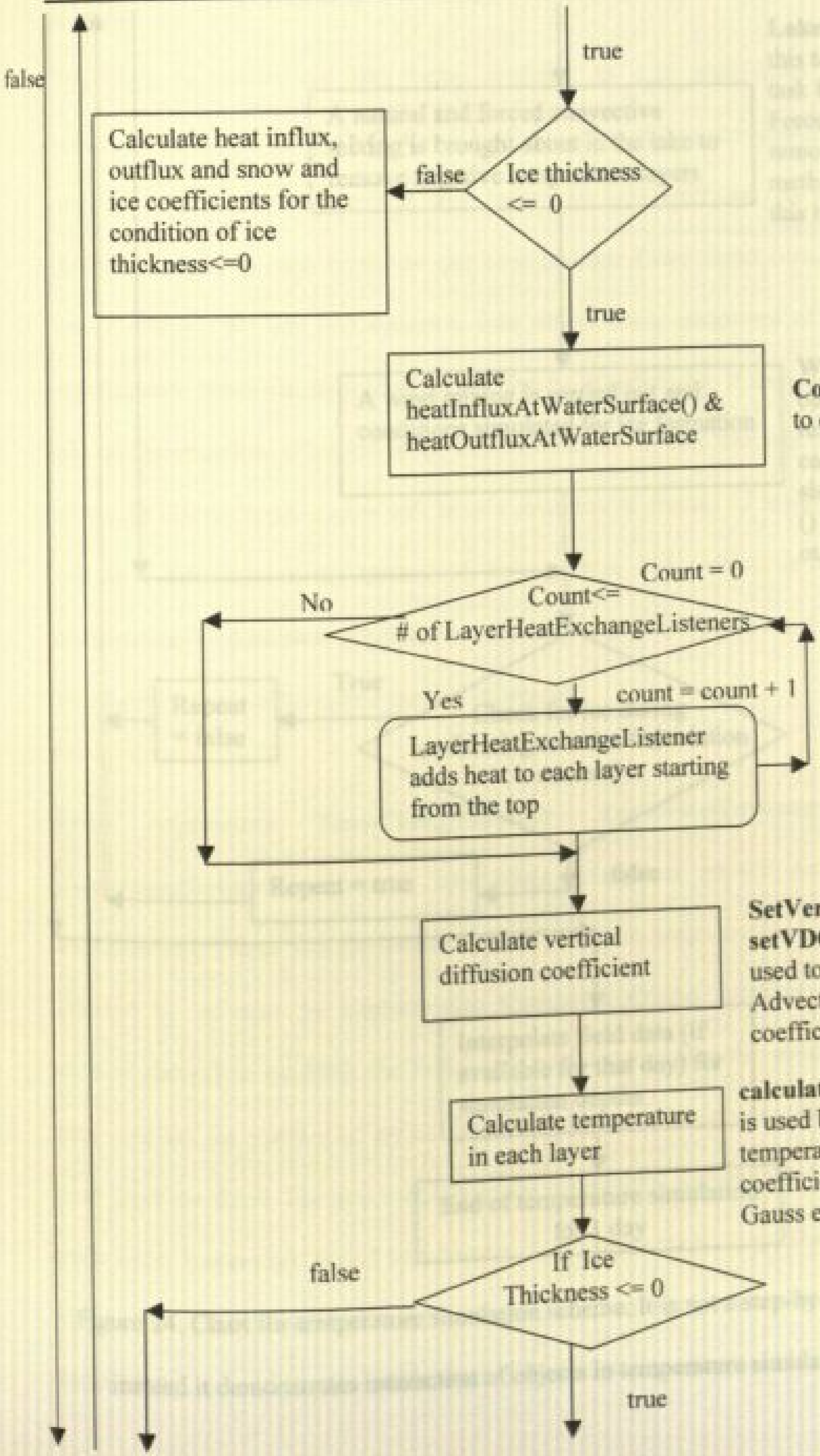

call to startSimulation()
MeteorDataReader maintains a protocal to read meteorological data MeteorFileReader implements this protocol and it is registered with interface
MeteorFileReader implements MeteorDataReader
Interface MeteorDataReader
No
Day<= final day
Yes
Read Meteor Data by class implementing MeteorDataReader interface
Count = 0
No
Count<= # of TimeChangeListeners
Yes
count = count + 1
Inform TimeChangeListener about present simulation day
Lake
Wind
Coefficient
Sediment
Snow
Ice
LakeFileWriter
ModelParams – implements TimeChangeListener Interface
Count = 0
No
Count<= # of WaterQualitySimulators
Yes
count = count + 1
Minlake implements WaterQualitySimulator interface to simulate temperature
Carry out water quality component simulation by calling simulate() method of WaterQualitySimulator interface
Carry out statistical error analysis
End simulation

Figure 23. Object Interaction Diagram displaying interaction scheme defined in Minlake

SIMULATION SCHEME FOR TEMPERATURE MODELING


Start temperature simulation - simulate() of WaterQualitySimulator in Minlake class
Set coefficients, model parameters for the particular month and day
Determine Snow Thickness – Snow handles this task
Determine Ice Thickness – Ice handles this task
No
Count = 0
Count<= # of ChlorophyllEstimators
Yes
count = count + 1
Instruct ChlorophyllEstimatror to estimate chlorophyll
Estimator implements Chlorophyll interface to estimate chlorophyll concentration
Initialize lake-mixing temperature to the temperature of first layer Tmix = T(0)
- Calculate wind shear by using calculateShear() method of Wind class
- Calculate wind surface kinetic energy by using method calculateSurfaceKineticEnergy() method
false
If repeat
true

SIMULATION SCHEME FOR TEMPERATURE MODELING (continued page 2/3)


true
false
Calculate heat influx, outflux and snow and ice coefficients for the condition of ice thickness<=0
false
Ice thickness <= 0
true
Calculate heatInfluxAtWaterSurface() & heatOutfluxAtWaterSurface
Count = 0
No
Count<= # of LayerHeatExchangeListeners
Yes
count = count + 1
LayerHeatExchangeListener adds heat to each layer starting from the top
Calculate vertical diffusion coefficient
Calculate temperature in each layer
false
If Ice Thickness <= 0
true

Coefficients class is used to carry out this task

Lake and **Sediment** implements **LayerHeatExchangeListener** interface and this portion calculates heat source/sink terms of advection diffusion equation

SetVerticalDiffusionCoeff() and **setVDCHarmonicMean**() can be used to determine the Advection/diffusion equation coefficient

calculateTemperatureFromFlux() is used by **Lake** to establish a temperature profile after obtaining coefficients from Coefficients class – Gauss elimination technique is used

SIMULATION SCHEME FOR TEMPERATURE MODELING (continued page 3/3)

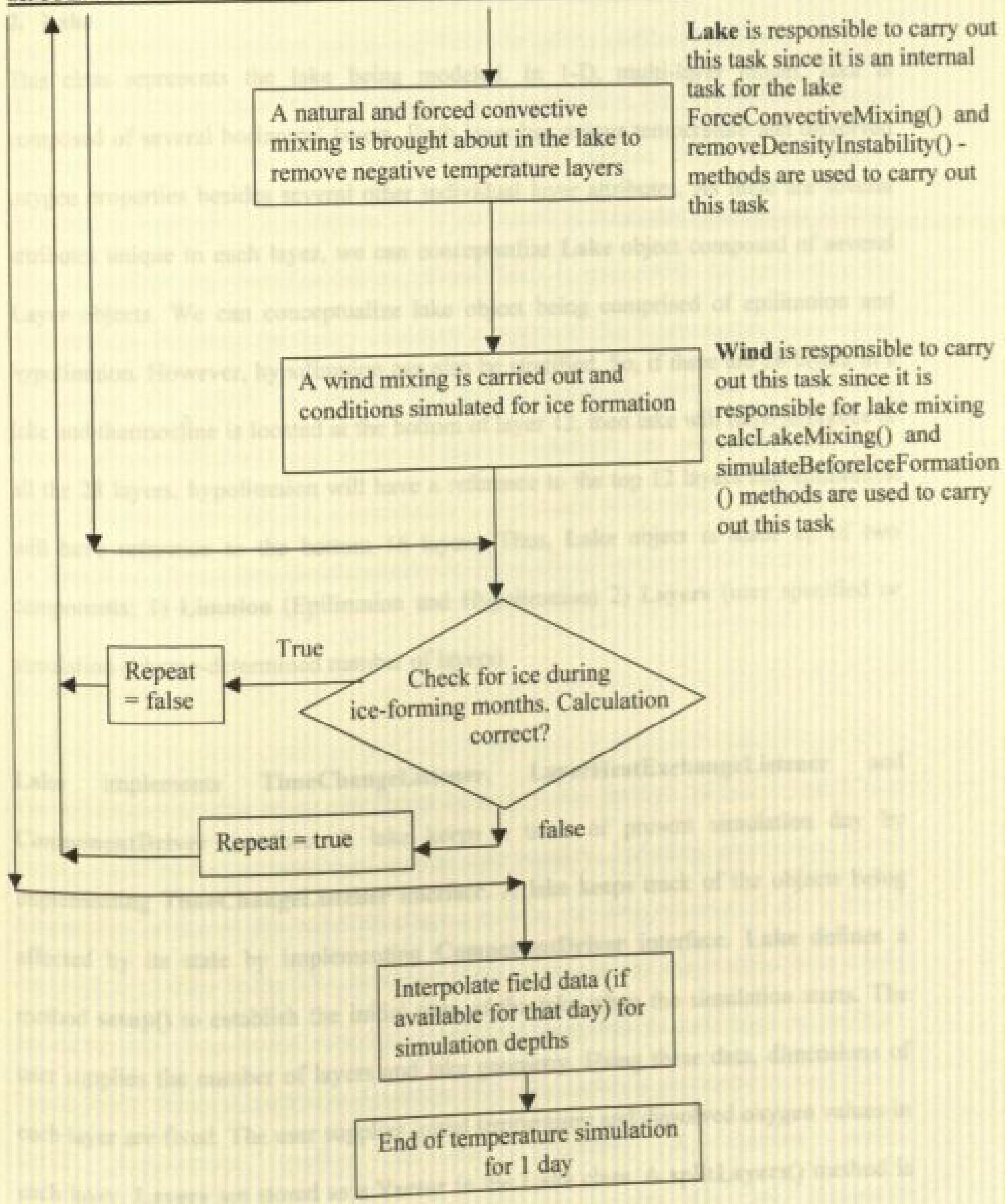

A natural and forced convective mixing is brought about in the lake to remove negative temperature layers
Lake is responsible to carry out this task since it is an internal task for the lake ForceConvectiveMixing() and removeDensityInstability() - methods are used to carry out this task
A wind mixing is carried out and conditions simulated for ice formation
Wind is responsible to carry out this task since it is responsible for lake mixing calcLakeMixing() and simulateBeforeIceFormation() methods are used to carry out this task
Check for ice during ice-forming months. Calculation correct?
True
Repeat = false
false
Repeat = true
Interpolate field data (if available for that day) for simulation depths
End of temperature simulation for 1 day

Figure 24. Chart for temperature simulation scheme. It is not a step-by-step procedure; instead it demonstrates interaction of objects in temperature simulation scheme.

2. Lake

This class represents the lake being modeled. In 1-D, multi-layer model, lake is composed of several horizontal layers. Each layer has unique temperature and dissolved oxygen properties besides several other individual layer attributes. As there are several attributes unique to each layer, we can conceptualize **Lake** object composed of several **Layer** objects. We can conceptualize lake object being comprised of epilimnion and hypolimnion. However, hypolimnion can also be stratified. So, if there are 28 layers in a lake and thermocline is located at the bottom of layer 12, then lake will have reference to all the 28 layers, hypolimnion will have a reference to the top 12 layers and epilimnion will have reference to the bottom 16 layers. Thus, **Lake** object is made up of two components; 1) **Limnion** (Epilimnion and Hypolimnion) 2) **Layers** (user specified or simulation-scheme-determined number of layers)

Lake implements **TimeChangeListener, LayerHeatExchangeListener** and **ComponentDriver** interface. A lake keeps a track of present simulation day by implementing **TimeChangeListener** interface. A lake keeps track of the objects being affected by its state by implementing **ComponentDriver** interface. **Lake** defines a method **setup()** to establish the initial state of the lake when the simulation starts. The user supplies the number of layers and lake geometry. Using these data, dimensions of each layer are fixed. The user supplies initial temperature and dissolved oxygen values in each layer. **Layers** are stored as a **Vector** in the **Lake** class. A **splitLayers**() method is invoked to establish a check on the thickness of layers. If the thickness of any layer exceeds the maximum permissible thickness specified by the user, then the layer is split into two equal parts with half the thickness of the original layer. **Lake** implements **LayerHeatExchangeListener** and so it is forced to define **addHeatToLayers**() method. Based on the theory described in Appendix E2, a calculation for heat exchange across each layer is carried out using **addHeatToLayers**() method. Based on the theory described in Appendix E3, a calculation is performed to determine temperature in each layer. A method **calculateTemperatureFromFlux**() is responsible for performing the computations. Based on the theory described in Appendix E4, a calculation is performed to estimate the effects of natural and forced convective mixing. The methods **forceConvectiveMixing**() and **removeDensityInstability**() are responsible for these tasks. Based on Appendix E5, a calculation is performed to simulate conditions for the ice formation. The method **simulateBeforeIceFormation**() is responsible for this task.

3. Sediment

The sediment is another major contributor to the heat flux through the lake layers. It also plays a major role in simulation of several state variables including temperature. Initial temperature profile in the sediment plays a role in the determination of the eventual temperature profiles in the lake. However, it does not have any major effect over a long period of time. A sediment temperature profile corresponds to each layer of the lake. Based on the theory described in Appendix E6, a calculation is performed to establish an initial temperature profile in the sediment. A method called **setInitialTemperatureProfile**() is responsible to perform this task. **Sediment** implements **TimeChangeListener** interface and keeps a track of present simulation day.

Sediment implements **LayerHeatExchangeListener** interface and so it has to define a method called **addHeatToLayers**(). This method supplies heat to each layer. Corresponding to the difference between the **previous day**'s temperature and **present day**'s temperature, heat flux is added to the corresponding lake layer for that day. Assumptions used for this technique are; 1) there is no horizontal heat flux between two different sets of sediment layers, 2) the bottom of the sediments is a no flux boundary. These assumptions ensure that any change in temperature profile of each sediment layers corresponds to the heat exchange with the adjacent water body.

4. SnowIce

This class is the parent class of **Snow** and **Ice**. Though snow and ice are different, they have many similarities. They have some common attributes and methods. Thus, **SnowIce** class serves as a container for all these similar attributes and methods. **SnowIce** contains attributes like thickness, conductivity, absorption coefficient, reflection coefficient, and attenuation coefficient. **SnowIce** implements **TimeChangeListener**, **LakeListener, WindListener** and **ComponentDriver** interface. Wind forces a convective mixing in the lake and at the same time creates a shear on ice and snow. Thus, stable ice or snow formation can occur only at low wind speeds. To simulate these conditions ice and snow has to obtain wind information. Implementing **WindListener** interface ensures that **Snow** and **Ice** always has reference to the present state of **Wind**. **Snow** and **Ice** affects the simulation by altering the temperature profile. So different objects are affected by the state of snow and/or ice. Thus, implementing **ComponentDriver** interface keeps track of of all objects being affected by the state of **Snow** or **Ice** (Listeners). Thus, at any point in time the listeners always have a reference to the present state of the **Snow/Ice**. It also keeps track of the present simulation day by implementing **TimeChangeListener** interface.

5. Snow

This class represents the snow that forms over the surface of the lake. It serves as a placeholder for attributes and methods that the snow possesses. It is a derivative or a child class of **SnowIce** and so it possesses all the attributes and methods of **SnowIce.** It differs from **SnowIce** by possessing **Snow** specific attributes and methods. **Snow** object implements **IceListener** interface. This implementation allows snow to have a reference to the **Ice** condition at any point in time. **SnowIce** has an abstract method. This ensures that the child classes have to define this method. This method, **determineThickness()** defined in the class **Snow** determines snow thickness. Appendix E7 describes the theory underlying computation of snow formation at the start of each simulation day.

6. Ice

This class represents ice that forms over the surface of the lake. It serves as a placeholder for attributes and methods of ice. It is a derivative or a child class of **SnowIce**. Thus, it possesses all the attributes and methods of **SnowIce** and differs from it by possessing **Ice** specific attributes and methods. Similar to Snow, Ice also has to define abstract method of **SnowIce**. Appendix E8 describes the theory underlying the computation of ice thickness at the start of each simulation day. The method **determineThickness()** is responsible for this task.

7. Estimator

This class implements **ChlorophyllEstimator** interface. The **ChlorophyllEstimator** interface forces **Estimator** to define a method for obtaining the chlorophyll data for each day of simulation.

In this simulation model, chlorophyll-a is not simulated. Instead, the analyst has to supply information regarding the chlorophyll-a concentration. Mean chlorophyll-a concentrations are specified as model input, dependent upon the trophic status. Trophic status can be determined based on the Secchi depth scale. In this model, Secchi depths of 1.2, 2.5 and 4.5 corresponds to Eutrophic, Mesotrophic and Oligotrophic lakes. For a regional lake analysis, mean chlorophyll-a concentrations for eutrophic, mesotrophic and oligotrophic lakes that correspond to these secchi depths are 15,6 and 2 µg/l.

The analyst can provide either yearly-mean chlorophyll-a data or field data for cholorphyll-a concentration. For the latter case, the mean chlorophyll-a concentration is calculated from the field data by taking their arithmetic mean value. A distinct hypolimnion and epilimnion mean concentrations can be obtained for the simulation runs with the field data.

A relationship of % difference from mean chlorophyll-a concentration to the month of year is well established. Chlorophyll concentration is obtained for each simulation day by the following formula,

Chla = {1 + (Ratio/100)} * Mean Annual Chlorophyll Concentration.

This relationship is available in file "chla.pat".

8. Wind

This class represents wind blowing over the surface of the lake. Wind significantly alters the lake water temperature profile and thus, the sediment temperature profile. Wind also causes a shear in ice and snow. Stable ice formation cannot take place with high wind blowing over it. **Wind** possesses various attributes like wind coefficient, air temperature, surface kinetic energy, etc. This class implements **TimeChangeListener, LakeListener** and **ComponentDriver** interface. **TimeChangeListener** interface allows **Wind** to keep track of date and time. **LakeListener** interface allows **Wind** to obtain a reference to the present state of the **Lake**. **ComponentDriver** interface allows **Wind** to keep a track of the objects that are influenced by wind. Appendix E9 describes effect of wind mixing on the lake. The method **calcLakeMixing**() is responsible for this task.

Table 3 lists the remaining components of javamodel package of JAMINLAKE application. It also gives a brief description of each class. Further details on each of these classes and classes in the other packages of JAMINLAKE are presented in the JAVADOCS and the application source code.

Table 3. A list of components of javamodel package in JAMINLAKE (Components not described above)

NAME OF THE CLASSES	DESCRIPTION
DataInputOutput	This interface forces the implementing classes to define a method to access the input data and to write the output data of the simulation application. This interface ensures easy transition of JAMINLAKE from an ASCII text files (for data storage) to any other type of database.
FileOperation	It implements **DataInputOutput** interface and defines a method to read data from ASCII text files.
ComponentDrivers	This interface forces the implementing class to define a method to communicate a reference of their state to all their listeners.
LakeListener	This interface forces the implementing class to register itself with **Lake** and obtain a reference to the current state of the **Lake**. This interface is used to obtain information about **Lake**, based on simulation scheme.

TimeChangeListener	This interface forces an implementing class to define a method to receive information on the change of simulation date. The **TimeChangeListeners** are informed about the simulation date at the start of each simulation day.
LayerHeatExchangeListener	This interface ensures that the implementing classes define a method to supply heat to the layers and thus, make a provision for additional heat sources. Depending on the simulation scheme, **Minlake** object calls all the classes implementing **LayerHeat ExchangeListener** to supply heat to each layer.
Limnion	Each lake has several properties common to the epilimnion and hypolimnion compartments. Thus, each lake is formed of two limnions (epilimnion and hypolimnion).
Layer	In a stratified lake, state variables have a constant value in a horizontal layer. Thus, several layers constitute a lake. The users provide number of layers and application performs a counter check to ensure that thickness of each layer is below the maximum.
SnowListener	This interface forces the implementing class to register itself with **Snow** and obtain a reference to the current state of **Snow**.

IceListener	This interface forces the implementing class to register itself with the **Ice** and obtain current reference to its state.
ChlorophyllEstimator	This interface forces the implementing class to define a method to obtain daily chlorophyll data. **Estimator** class implements this interface in JAMINLAKE.
Coefficients	This class holds the value of various coefficients used in JAMINLAKE. Based on the theory described under Appendix E1, heat input and output at the surface of the lake is calculated. The methods **heatInfluxAtTheWater Surface()** and **heatOutfluxAtTheWaterSurface()** are responsible for this task. Based on the theory described under Appendix E10, vertical diffusion coefficients are calculated. A method **setVerticalDiffusionCoeff()** is responsible for this task. Diffusion coefficients are calculated for each layer and they are estimated at the interface by taking a harmonic mean.
ModelParams	This class holds the value of the fixed model parameters. It stores information like the simulation mode; 1) temperature only simulation, 2) temperature and dissolved oxygen simulation.
WindListeners	This interface forces the implementing class to register

	itself with the **Wind** and obtain a reference to the current state of the **Wind**.
Date	It corresponds to a particular date and defines methods to calculate julian day from a date and vice versa.
ModelParamsListener	This interface forces the implementing class to register itself with **ModelParams** and obtain a reference to its current state.

5. TESTING THE SYSTEM

FORTAN, a mathematically intensive programming language, is not a hard coded language. A hard coded language ensures that common programming errors are always trapped. There are several errors that FORTRAN will not detect during the compilation. A common programming error that is difficult to diagnose is the array indexing issue. An attempt to index an array value outside the array range, returns a "0" value. However, this error is usually a result of incorrect programming. JAVA on the other hand, does not allow accessing array elements outside its range. Thus, a redevelopment illustrates any such attempt of accessing the array outside its range. Some other errors were also detected in LAKE36. It is necessary to emphasize that without an attempt to redevelop this model, these errors would have remained undetected. JAVA is an extremely hard coded language that does not allow these kinds of errors to propagate. The errors are listed below.

1. Precision error:

   Typically, FORTAN assumes a 4-byte precision for a real variable. This means that for a normal 32-bit machine, the precision is only up-to 4 decimal places. The rest of the digits are garbage and are picked up at random. These incorrect digits can lead to several problems. One of them is discussed below.

   For calculating forced mixing due to wind shear, the kinetic energy supplied by the wind has to be lesser than the total potential energy to lift the mass of the water from the mixed layer depth to the surface. Potential energy calculations require mass of the water between the top of the surface and the mixed layer has to be calculated. This in turn requires multiplication of density with volume of water. Volume of a typical lake can be in the range of 10^3 to 10^7 m^3. For calculating mass, volume is multiplied with density. However, if the precision of density is real 4, then the mass computed might be incorrect. Thus, at least a precision of real 8 is required. This precision ensures that mass is always computed correctly. An example is presented to illustrate the importance of precision in mass balance calculations.

This example is an actual result of one of the simulation runs and is documented in Table 4. The first column of Table 4 labels the data presented for each type of simulation. The second column is a simulation run of JAMINLAKE (JAVA model) & third column is a simulation run of LAKE36. Potential energy is calculated using equation 2 and values of volume, depth, thickness, and density of a layer. Kinetic energy is calculated during simulation and its value is presented below. The comparison is carried out for each layer (assuming that it is a mixed layer). If the kinetic energy supplied by the wind to the surface of the lake is less than the potential energy of the assumed mixed layer (to raise the assumed mixed layer to the surface of the lake) than that layer is indeed a mixed layer. In this example only the density of the mixed layer is different between the JAVA MODEL and the FORTRAN MODEL. The difference lies only beyond the third decimal point. Inspite of such a trivial difference in density precision, the interpretation is significantly altered.

The finite difference formula for calculating potential energy from the mixed layer depth is given below.

Potential Energy=9.81*TV(I)*(Z(I)+DZ(I)/2.0–DCM)*(DENH-DENL) , (2)

where

TV(I) – is the volume of the water above the ith layer,

Z(I), DZ(I) – thickness and depth of ith layer,

DENH, DENL–density of water above & below mixed layer,

DCM – length up to center of the mass of a layer from bottom.

Table 4. An example to illustrate importance of precision in mass balance calculations

	JAVA MODEL	FORTRAN MODEL
Volume above the layer -TV(I)	2182784.37500000000000000	2182784.37500000000000000
Depth of the layer - Z(I)	5.14000000000000000	5.14000000000000000
Thickness of the layer - DZ(I)	0.59000000000000000	0.58999980000000000
DCM	2.28252251721892000	2.28252251721892000
Density above mixed layer - DENH	999.99685925063800000	999.99690000000000000
Density of layer below mixed - DENL	999.98941465879900000	999.9894
Total Potential Energy	**502542.42234417800000000**	**506282.69883887200000000**
Associate kinetic energy	**502650.00000000000000000**	**502650.00000000000000000**
	Kinetic Energy>Potential Energy	Potential Energy>Kinetic Energy
Inference	**This layer is not mixed layer**	**This layer is mixed layer**

Thus, precision of the density calculation can lead to a change in identifying the mixed layer. The volume above the mixed layer is uniformly mixed and so the temperature above it is a volumetric average of the entire water above the mixed layer. If the mixed layer changes, the temperature above the mixed layer also changes altering the temperature profile in the lake.

Mixed mode processing is another source of precision error. By default, FORTRAN takes a literal numeric variable as a single precision variable unless a programmer explicitly specifies otherwise. In a mixed mode operation the result is always single precision number, unless a particular compiler is instructed to force all real variables to double precision. LAKE36 had to be changed to make this correction.

Surface elevation (ST) is calculated using maximum lake depth (ZMAX) and the datum for the bottom of the lake (DBL) and Equation 3. Maximum lake depth and datum are user provided data.

$$ST = ZMAX + DBL \tag{3}$$

Maximum lake depth is provided as a user input. However, it is recalculated from datum and surface elevation of the lake.

$$ZMAX = ST - DBL \tag{4}$$

This step is unnecessary and it should not alter results. However, it does alter results slightly because of the precision error. The error is not significant in terms of standard deviation with the field data, but it should still be corrected.

2. Array indexing issue:

   Sometimes, a code can try to access the value outside the range of an array. Most programmers have encountered such a condition. FORTRAN behavior under such conditions depends on the compiler. FORTRAN 90 should not allow outside range indexing. ANSI F77 does trap the error but allows the execution to continue. JAVA is designed to throw an exception (terminating the program and listing the problem) for such a situation.

   A specific example of such an error is presented below. Sediment temperature profile exists for each lake layer. If a user wants to use 10 sediment layers then for each lake layer 10 sediment layers exist. The sediment temperature profile is used to determine the heat flux to the lake. The difference between present day's sediment temperature profile and yesterday's sediment temperature profile corresponds to the heat flux transferred to the corresponding lake layer. For each simulation day, yesterday's sediment temperature profile is recorded in the variable TSOL. It is a double subscript array, one subscript corresponds to the lake layer and the other corresponds to the sediment layer. The present day's sediment temperature is recorded under variable TSN. It is similar in dimension to TSOL. The following line of code calls the subroutine Bottom in order to calculate the heat flux to the layer:

   ```
   CALL BOTTOM (TSO(21,K),TSN(21,K),T2MBOT,JDY)
   ```

The objective of the above subroutine is to establish a difference between the sediment temperature of **previous day**'s temperature profile and **present day**'s sediment temperature profile corresponding to each layer. However, the index 21 is actually a reference to the sediment corresponding to the next lake layer. For example, if the code is used for first lake layer then the sediment temperature corresponding to the second layer supplies the heat flux to it. As this calculation is incorrect, the above code has to be modified to reflect the lake layer in consideration. The following code corrects the problem:

```
CALL BOTTOM (TSO(1,K),TSN(1,K),T2MBOT,JDY)
```

3. Simulation time error:

   LAKE36 carries out the simulation from the starting simulation day to the end of the month of the final simulation day. However, because the user specifies the specific simulation dates for start and end, the program should be corrected to reflect that. A simple modification inserted in the original program, solves this problem. The following lines of code, inserted at the start of the loop for each day calculation, solves the problem:

   ```
   IF (MYEAR.EQ.FYEAR.AND.MONTH.EQ.FMON) THEN
          KDAYS = FDAY
   END IF
   ```

4. Logic issue

   Originally LAKE36 carried out computation in the following sequence. First lake profile was established. Corresponding to each lake layer, sediment layers were constructed. Thus, sediment temperature profile formed a double subscript array TSOL [lake layers][sediment layers].

After establishing sediment temperature profile, a thickness check for the lake layers was carried out. If the thickness was greater than maximum permissible thickness, then the layer was split into two identical layers and thus, increased the total number of layers in the lake. However, the original application did not increase the corresponding number of sediment layers.

Thus, when a value of the array element corresponding to the lake-sediment interface (TSOL [maximum old lake layers + 1][sediment layers]) was requested, LAKE36 returned a zero value. The returned value was incorrect. The correct return value was the temperature corresponding to the bottom of the lake layer.

This mistake was corrected by performing layer thickness check before setting up the initial sediment temperature profile. This modification was simple using JAVA. However, it was complicated using FORTAN.

5. Statistics package issue

There was a minor error in the way statistical calculations were carried out in LAKE36. LAKE36 assumed that the maximum field depth supplied by the user was the maximum depth of the lake. The statistical calculations produced incorrect results because of this assumption. In LAKE36, the statistical calculations always performed interpolations of field depths. The application was modified and logic for extrapolating under extreme conditions was incorporated.

6. Test condition issue:

   A test condition was introduced in LAKE36 for models that do not run for the complete year. On march 1st, every year, the lake temperature profile was reset to 4°C. However, for year round simulation models this reset should not be done. Because of a mistake in programming, the simulation conditions were reset every first day of March. To eliminate this occurrence, the portion of the code corresponding to the reset conditions was deleted.

6. RESULTS AND DISCUSSION

Several simulation runs were carried out using JAMINLAKE & LAKE36. LAKE36 was significantly altered during the testing phase. Several simulation runs were carried out for each significant correction. The important corrections in LAKE36 were the array indexing and the logic correction. The comparisons were performed to determine the correction that had the greatest effect on the results.

Two distinct types of simulation results were used for comparison; 1) Simulation results with the field data, and 2) Simulation results without the field data. This differentiation was implemented to illustrate the identical simulation results for JAMINLAKE and LAKE36 even under different simulation conditions. Results of JAMINLAKE were compared with the results of several testing stages of LAKE36. A comparison of JAMINLAKE with LAKE36 (after corrections) was included to verify correctness of the re-engineered application JAMINLAKE. A comparison of JAMINLAKE with LAKE36 (before corrections) was included to show a difference in results of JAMINLAKE simulation with the original LAKE36 (with errors). A comparison of JAMINLAKE with LAKE36 (before logic correction) was included to estimate the effect of logic issue on the simulation results. Similarly, a comparison of JAMINLAKE with LAKE36 (before array indexing issue correction) was included to estimate the effect of array indexing issue on the simulation results. For the simulation results containing the field data, a comparison of JAMINLAKE with field data, LAKE36 (after corrections) with field data, and LAKE36 (before corrections) with field data was included to show the effect of testing/corrections on the simulation results.

In this section, comparison of five data sets is presented. Each data set consists of lake geometry, simulation data (user input data) and result comparison. The user input data is important in identifying the lake being simulated and conditions for simulation. The actual input files are very large, the data are repetitive and the file structure is complicated. Thus, only a sample input file is provided in the Appendix B instead of providing an input data file for each data set. The field data, initial lake data and some other parameters are required to run the simulation application. However, they do not constitute important input information. Such information is omitted in the sample input file. The first data set is for the Cedar Lake. Each HTML page used for simulation of data set 1 is displayed in Appendix D. This is an example of typical simulation procedure.

The comparison of results is based on assessment of seven parameters:

1. Mean Error: Mean error is simply a difference of observed value (value in either Lake36 or modified Lake36) and predicted value (JAMINLAKE calculations). It sums all the differences and divides it by the number of data used for comparison.
2. Mean Absolute Error: Mean Absolute error computation is similar to Mean Error computation. However, it uses absolute error value instead of normal error value.
3. Root Mean Square Error (RMS): The RMS error computes sum of squared errors, takes the square root of the result and divides it by the number of data.

4. Maximum Absolute Error: Maximum absolute error calculates the maximum absolute error between observed and predicted values.
5. Slope: A least square analysis is carried out and the slope of the regression line is calculated.
6. Regression coefficient: A least square analysis is carried out and regression coefficient is calculated.
7. Standard Deviation: Standard deviation around the mean is calculated.

In this simulation model, a user specifies the number of layers in the lake. However, if the thickness the layer turns out greater than the critical thickness then the layer is split into two equal parts. This action increases the total number of layers used for simulation. The original number of layers provided by the user is presented in the input data. The number of layers actually used in simulation is gathered from the output data file. This number is presented at the end of input data for each comparison data set. If the actual number of layers used in the model is greater than the user supplied number of layers, then the split subroutine is invoked for that simulation. And so the logic error should exist.

6.1. DataSet 1:

6.1.1. Area.sdf

Each line has area, depth and volume relation. This data set contains field data. This is an actual lake.

m	m^2	m^3
0.0	0.0	0.0
0.1	259000.0	12950.0
0.9	1331500.0	649150.0
1.7	2404000.0	2143350.0
3.2	2842000.0	6077850.0
4.7	3280000.0	1.066935E7

6.1.2. Input Data (inputNames.ini)

Simulation for lake : Cedar, for user : fangxu@hal.lamar.edu

Meteorological data file name MNMINN80

Simulation Start Date	4 16 1980	
Simulation End Date	10 12 1993	
Number of Layers	16.0	
Maximum Lake Depth	4.7	m
Initial Lake Stage (w.r.t datum)	251.0	m
μ_W (Light Attenuation Coefficient for water)	0.44	1/m
μ_{CH} (Light Attenuation Coefficient for chlorophyll)	20.0	1/m
Summer Shelter Coefficient	0.7	

Fall Shelter Coefficient		0.7
Sheltering Coefficient		1.0
Maximum Snow/Ice ratio		1.0
Thermal Diffusivity		0.035 m^2/day
Heat Capacity	550.0	kcal/m^3/°C
Compaction Factor		0.4
Ice conductivity	2.6	c.w/ °C^{-1}/m^{-1}
Snow conductivity	0.27	c.w /°C^{-1}/m^{-1}
Turbulent Heat Coefficient	11.35	kcal/m /s/°C
Critical Depth	0.1	m
ICE:		
Absorption coefficient		0.17
Reflection Coefficient		0.55
Attenuation coefficient		1.6
SNOW:		
Absorption coefficient		0.34
Reflection Coefficient		0.8
Attenuation coefficient		40.0
Intial ice thickness	0.0	m
Intial snow thickness	0.0	m

Simulation mode : temperature and DO

Number of layers actually used for simulation is 16. This is same as the user specified, hence the layers were not split.

6.1.3. Result Comparison:

Table 5 lists the results for data set 1. The first column is the name of the parameter used to carry out the comparison. The second, third, fourth and fifth column are comparisons of JAMINLAKE and LAKE36 at different stages of correction. The sixth, seventh and eighth column are comparisons of JAMINLAKE, LAKE36 (after all corrections) and LAKE36 (before all correction) simulation results with the field data.

Table 5. Result comparisons for data set 1.

Fitness Value	Jaminlake & Lake36 after corrections	Jaminlake & Lake36 before all corrections	Jaminlake an& Lake36 before logic correction	Jaminlake & Lake 36 before array issue	Jaminlake & Field Data	Lake36 & Field Data	Lake 36 before corrections and field data
Mean Error	2.506E-5	0.00187	2.506E-5	-0.0061	-	-	-
Mean Abs Error	2.506E-5	0.01596	2.506E-5	0.01213	-	-	-
Root Mean Error	1.659E-4	0.09902	1.659E-4	0.06520	-	-	-
Max. Absolute Error	0.001	8.21399	0.001	7.99999	5.96661	-5.97	-5.87
Slope	1.00000	1.0024	1.00000	0.99764	0.88894	0.89	0.9
Regression coefficient	0.99999	0.99694	0.99999	0.99772	0.89475	0.89	0.9
Standard deviation	6.674E-4	0.49604	6.674E-4	0.42780	1.70844	1.71	1.68

6.2. Data Set 2:

6.2.1. Area.sdf

This data set contains field data, however, this is not an actual lake and nor is the field data real. Hence, only a comparison of JAMINLAKE with Lake36 and other versions is presented.

m	m^2	m^3
0.0	0.0	0.0
0.5	97337.0	24334.25
1.0	204956.0	99907.5
1.5	339057.0	235910.75
2.0	506157.0	447214.25
2.5	714377.0	752347.75
3.0	973834.0	1174400.5
3.5	1297138.0	1742143.5
4.0	1700000.0	2491428.0

6.2.2. Input data (inputNames.ini)

Simulation for lake: huang20, for user: fangxu@hal.lamar.edu

Meteorological data file name MNDULU86

Simulation Start Date 4 16 1986

Simulation End Date 10 16 1992

Number of Layers 28

Maximum Lake Depth 14.600000381469727 m

Initial Lake Stage (w.r.t datum) 637.0 m

μ_W (Light Attenuation Coefficient for water)	0.44	1/m
μ_{CH} (Light Attenuation Coefficient for chlorophyll)	20.0	1/m
Summer Shelter Coefficient	0.01	
Fall Shelter Coefficient	0.01	
Sheltering Coefficient	1.0	
Maximum Snow/Ice ratio	1.0	
Thermal Diffusivity	0.035	m^2/day
Heat Capacity	550.0	kcal/m^3/c
Compaction Factor	0.3	
Ice conductivity	2.6	c.w/ $°C^{-1}/m^{-1}$
Snow conductivity	0.27	c.w/ $°C^{-1}/m^{-1}$
Turbulent Heat Coefficient	11.35	kcal/m /s/°C
Critical Depth	0.1	m
ICE:		
Absorption coefficient	0.17	
Reflection Coefficient	0.55	
Attenuation coefficient	0.16	
SNOW:		
Absorption coefficient	0.34	
Reflection Coefficient	0.8	
Attenuation coefficient	40.0	
Intial ice thickness	0.0	m
Intial snow thickness	0.0	m

Initial lake conditions and field data are not provided here.

Simulation mode: Temperature and DO

Number of layers actually used for simulation is 28. This is same as the user specified, hence the layers were not split.

6.2.3. Result comparison:

Table 6 lists the results for data set 2. The first column is the name of the parameter used to carry out the comparison. The second, third, fourth and fifth column are comparisons of Jaminlake and LAKE36 at different stages of correction.

Table 6. Result comparisons for data set 2.

Fitness Values	Jaminlake & Lake36 after corrections	Jaminlake & Lake36 before all corrections	Jaminlake & Lake36 before logic correction	Jaminlake & Lake36 before array issue correction
Mean Error	5.638E-5	-0.044	5.817E-5	-0.0260
Mean Abs Error	5.638E-5	0.0631	5.817E-5	0.0379
Root Mean Error	2.4652E-4	0.248	2.502E-4	0.1593
Max. Absolute Error	0.001	8.907	0.006	6.853
Slope	1.00000	1.00833	0.99999	1.01041
Regression coefficient	0.99999	0.97276	0.99999	0.98999
Standard deviation	6.660E-4	0.9489	7.565E-4	0.57496

6.3. Dataset 3:

6.3.1. Area.sdf

Each line has area, depth and volume relation. This data set contains field data. This is an actual lake.

m	m^2	m^3
0.0	0.0	0.0
2.6	23000.0	29900.0
5.6	121000.0	245900.0
8.6	463000.0	1121900.0
11.6	649000.0	2789900.0
13.6	786000.0	4224900.0
14.6	921000.0	5078400.0

6.3.2. Input data (inputNames.ini)

Simulation for lake : Fish, for user : fangxu@hal.lamar.edu

Meteorological data file name MNMINN80

Simulation Start Date	1 1 1980	
Simulation End Date	9 24 1992	
Number of Layers	29.0	
Maximum Lake Depth	14.6	m
Initial Lake Stage (w.r.t datum)	200.0	m
μ_W (Light Attenuation Coefficient for water)	1.2	1/m
μ_{CH} (Light Attenuation Coefficient for chlorophyll)	20.0	1/m

Summer Shelter Coefficient		0.2
Fall Shelter Coefficient		0.2
Sheltering Coefficient		1.0
Maximum Snow/Ice ratio		1.0
Thermal Diffusivity		0.035 m^2/day
Heat Capacity	550.0	kcal/m^3/°C
Compaction Factor		0.3
Ice conductivity	2.6	c.w/ °C^{-1}/m^{-1}
Snow conductivity	0.27	c.w/ °C^{-1}/m^{-1}
Turbulent Heat Coefficient	11.35	kcal/m/s/°C
Critical Depth	0.1	m

ICE:

Absorption coefficient	0.17
Reflection Coefficient	0.55
Attenuation coefficient	1.6

SNOW:

Absorption coefficient	0.34
Reflection Coefficient	0.8
Attenuation coefficient	40.0

Intial ice thickness	0.0	m

Intial snow thickness 0.0 m

Simulation mode: Temperature and DO

Number of layers actually used for simulation is 29. This is same as the user specified, hence the layers were not split.

6.3.3. Result comparison:

Table 7 lists the results for data set 3. The first column is the name of the parameter used to carry out the comparison. The second, third, fourth and fifth column are comparisons of Jaminlake and LAKE36 at different stages of correction. The sixth, seventh and eighth column are comparisons of JAMINLAKE, LAKE36 (after all corrections) and LAKE36 (before all correction) simulation results with the field data.

Table 7. Result comparisons for data set 3.

Fitness Values	Jaminlake & Lake36 after corrections	Jaminlake & Lake36 before all corrections	Jaminlake an& Lake36 before logic correction	Jaminlake & Lake 36 before array issue	Jaminlake & Field Data	Lake36 & Field Data	Lake 36 before corrections and field data
Mean Err.	2.6078E-5	0.00481	-1.4453E-5	-0.01232	-	-	-
Mean Abs Error	2.6078E-5	0.02848	-1.6229E-5	0.02082	-	-	-
Root Mean Error	1.6565E-4	0.16761	0.00163	0.11948	-	-	-
Max. Abs. Error	0.001	9.079	8.46699	9.134	10.1519	10.15	10.01
Slope	0.99995	1.00670	0.99938	1.00147	0.97258	0.97	0.95
Regression coefficient	0.99999	0.98188	0.99868	0.98809	0.73223	0.73	0.72
Standard deviation	6.690E-4	0.86794	0.23377	0.70371	3.06510	3.07	3.13

6.4. Dataset 4:

6.4.1. Area.sdf

Each line has area, depth and volume relation. This data set contains field data. This is an actual lake.

m	m^2	m^3
0.0	0.0	0.0
0.6	4618.06	1385.418
1.6	8393.25	7891.073
2.6	11596.98	17886.188
3.6	14330.47	30849.913
4.6	17691.85	46861.073
5.6	20353.43	65883.713
6.6	23989.82	88055.338
7.6	27629.57	113865.033
8.6	33295.91	144327.773
9.6	39933.84	180942.648
10.6	47188.81	224503.973
11.6	53005.31	274601.033
12.6	58557.78	330382.578
13.6	62487.64	390905.288
14.6	66156.58	455227.398

6.4.2. Input data (inputNames.ini)

Simulation for lake : wu28, for user : fangxu@hal.lamar.edu

Meteorological data file name MNDULU86		
Simulation Start Date	1 1 1986	
Simulation End Date	10 16 1992	
Number of Layers	28.0	
Maximum Lake Depth	14.6	m
Initial Lake Stage (w.r.t datum)	637.0	m
μ_W (Light Attenuation Coefficient for water)	0.44	1/m
μ_{CH} (Light Attenuation Coefficient for chlorophyll)	20.0	1/m
Summer Shelter Coefficient	0.01	
Fall Shelter Coefficient	0.01	
Sheltering Coefficient	1.0	
Maximum Snow/Ice ratio	1.0	
Thermal Diffusivity	0.035	m2/day
Heat Capacity	550.0	kcal/m^3/°C
Compaction Factor	0.3	
Ice conductivity	2.6	c.w/ °C^{-1}/m^{-1}
Snow conductivity	0.27	c.w/ °C^{-1}/m^{-1}
Turbulent Heat Coefficient	11.35	kcal/m/s/°C
Critical Depth	0.1	m
ICE:		
Absorption coefficient	0.17	
Reflection Coefficient	0.55	
Attenuation coefficient	1.6	

SNOW:

Absorption coefficient	0.34	
Reflection Coefficient	0.8	
Attenuation coefficient	40.0	
Intial ice thickness	0.0	m
Intial snow thickness	0.0	m

Simulation mode: Temperature and DO

Number of layers actually used for simulation is 29. This is not the same as the user specified and hence the layers were split in this case.

6.4.3. Result comparison:

Table 8 lists the results for data set 4. The first column is the name of the parameter used to carry out the comparison. The second, third, fourth and fifth column are comparisons of Jaminlake and LAKE36 at different stages of correction. The sixth, seventh and eighth column are comparisons of JAMINLAKE, LAKE36 (after all corrections) and LAKE36 (before all correction) simulation results with the field data.

Table 8. Result comparisons for data set 4.

Fitness values	Jaminlake & Lake36 after corrections	Jaminlake & Lake36 before all corrections	Jaminlake an& Lake36 before logic correction	Jaminlake & Lake 36 before array issue	Jaminlake & Field Data	Lake36 & Field Data	Lake 36 before corrections and field data
Mean Err	5.4748E-5	-0.00979	-0.00469	- 0.01322	-	-	-
Mean Abs. Error	5.4748E-5	0.03477	0.00471	0.01984	-	-	-
Root Mean Error	2.4214E-4	0.14974	0.02586	0.09044	-	-	-
Max Abs. Error	0.001	12.732	4.64400	5.07	9.4625	9.46	8.09
Slope	0.99999	0.99578	0.99567	1.00849	0.97021	0.97	0.98
Regression coefficient	0.99999	0.97707	0.99668	0.99496	0.88921	0.89	0.94
Standard deviation	6.724E-4	0.86125	0.32763	0.40361	1.74384	1.74	1.30

6.5. Dataset 5:

6.5.1. Area.sdf

Each line has area, depth and volume relation. This data set contains field data. This is not an actual lake and so no comparison with the field data is provided.

m	m^2	m^3
0.0	0.0	0.0
0.5	97337.0	24334.25
1.0	204956.0	99907.5
1.5	339057.0	235910.75
2.0	506157.0	447214.25
2.5	714377.0	752347.75
3.0	973834.0	1174400.5
3.5	1297138.0	1742143.5
4.0	1700000.0	2491428.0

6.5.2. Input data (inputNames.ini)

Simulation for lake : lake05, for user : fangxu@hal.lamar.edu

Meteorological data file name MNDULU61

Simulation Start Date	1 1 1961	
Simulation End Date	12 31 1978	
Number of Layers	28.0	
Maximum Lake Depth	4.0	m
Initial Lake Stage (w.r.t datum)	204.0	m
μ_w (Light Attenuation Coefficient for water)	0.62	1/m

μ_{CH} (Light Attenuation Coefficient for chlorophyll)	20.0	1/m
Summer Shelter Coefficient	0.5	
Fall Shelter Coefficient	0.5	
Sheltering Coefficient	1.0	
Maximum Snow/Ice ratio	1.0	
Thermal Diffusivity	0.035	m^2/day
Heat Capacity	550.0	kcal/m^3/°C
Compaction Factor	0.3	
Ice conductivity	2.6	c.w/ $^{\circ}C^{-1}/m^{-1}$
Snow conductivity	0.27	c.w/ $^{\circ}C^{-1}/m^{-1}$
Turbulent Heat Coefficient	11.35	kcal/m/s/°C
Critical Depth	0.1	m
ICE:		
Absorption coefficient	0.17	
Reflection Coefficient	0.55	
Attenuation coefficient	1.6	
SNOW:		
Absorption coefficient	0.34	
Reflection Coefficient	0.8	
Attenuation coefficient	40.0	
Intial ice thickness	0.0	m
Intial snow thickness	0.0	m

Simulation mode: Temperature and DO

This lake does not have a field data. It is a simulation model of a lake without any field data.

Number of layers actually used for simulation is 28. This is same as the user specified, hence the layers were not split.

6.5.3. Result comparison:

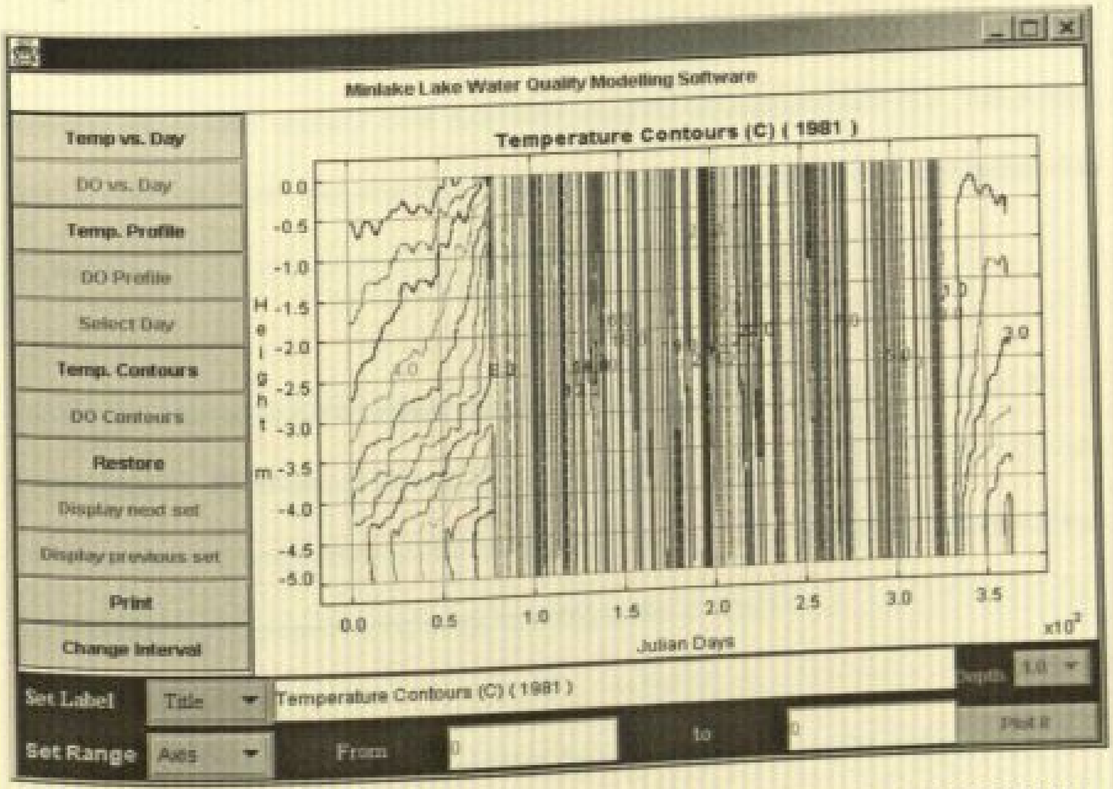
Table 9 lists the results for data set 5. The first column is the name of the parameter used to carry out the comparison. The second, third, fourth and fifth column are comparisons of Jaminlake and LAKE36 at different stages of correction. The sixth, seventh and eighth column are comparisons of JAMINLAKE, LAKE36 (after all corrections) and LAKE36 (before all correction) simulation results with the field data.

Table 9. Result comparisons for data set 5.

Fitness Values	JAMINLAKE & Lake36 after corrections	JAMINLAKE & Lake36 before corrections	JAMINLAKE & Lake36 without the logic issue correction	JAMINLAKE & Lake36 without the array index issue correction
Mean Error	2.642E-5	-0.00569	2.642E-5	-0.00984
Mean Abs Error	2.642E-5	0.02849	2.642E-5	0.02833
Root Mean Error	1.652E-4	0.16634	1.652E-4	0.16875
Max. Absolute Error	0.001	29.053	0.001	12.718
Slope	1.00000	1.00044	1.00000	0.99593
Regression coefficient	0.99999	0.98581	0.99999	0.98924
Standard deviation	6.6642E-4	0.88606	6.6642E-4	0.77138

6.6. COMPARISON OF CONTOUR OUTPUT DISPLAY OF ML2DPLOT1 AND SURFER:

Figure 25 is a contour display for data set 1 (Cedar Lake) year 1981. The file MNMINN81.tep is plotted using ML2DPLOT1 application. Figure 26 is an output of the same file rearranged and renamed as MNMINN81.sur. LakeFileWriter can also generate an output in XYZ format. The files created with XYZ format have an extension of "sur". Figure 26 is a screen capture of Surfer™ output. The two figures presented below show a visual comparison between the contour display by ML2DPLOT1 and Surfer™.



Set Label: Title — Temperature Contours (C) (1981)
Depth: 1.0
Set Range: Axis — From 0 to 0

Figure 25. Display of MNMINN81.tep (temperature output file) using ML2DPLOT1; First dataset (Cedar Lake) for year 1981.

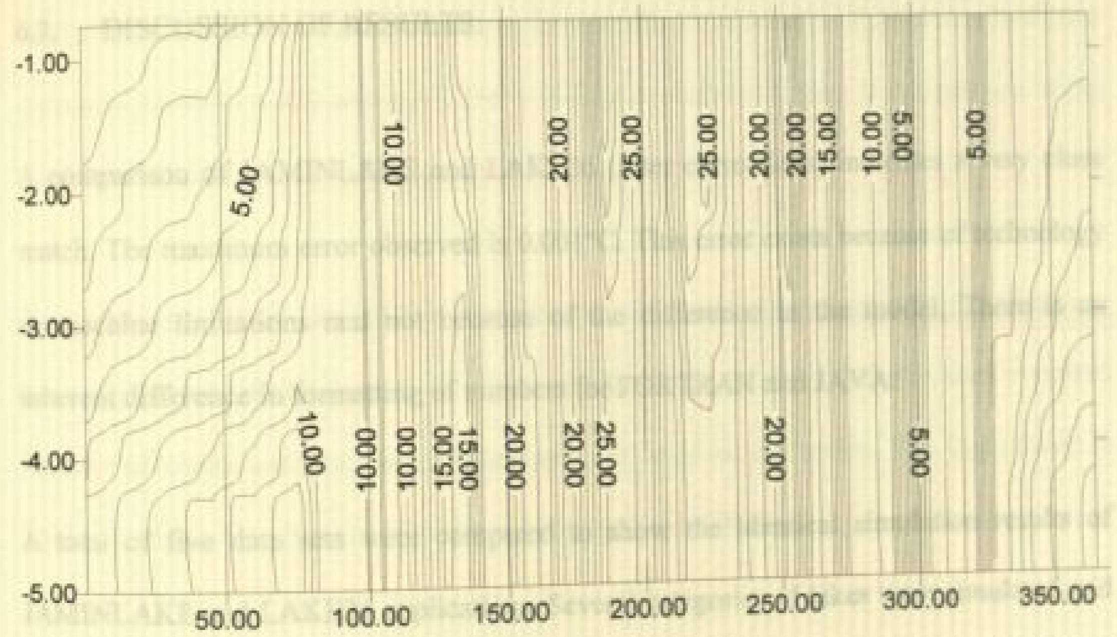


Figure 26. Display of MNMINN81.tep (temperature output file) using Surfer™; First dataset (Cedar Lake) for year 1981.

6.7. DISCUSSION OF RESULTS:

A comparison of JAMINLAKE and LAKE36 (after corrections) indicates a very close match. The maximum error observed is 0.001°C. This error exists because of technology or machine limitations and not because of the difference in the model. There is an inherent difference in formatting of numbers for FORTRAN and JAVA.

A total of five data sets were compared to show the identical simulation results of JAMINLAKE and LAKE36 applications. Several categories of lakes were simulated and the results were compared. The categories included; 1) shallow lake, 2) 2 deep lake, and 3) lake without filed data. Two data sets of shallow lakes, two data sets of deep lakes and one data set of lake without field data were used for simulation. However, the second data set was not a real lake and thus, comparison with the field data was not included in the results.

For all the data sets, a very close match of JAMINLAKE and LAKE36 (after all the corrections) is observed. However, a discrepancy is observed for JAMINLAKE and LAKE36 (before corrections). The comparison shows that the biggest effect for this variation can be attributed to the array indexing issue.

A comparison using field data shows that for most cases the results are identical. This similarity exists because of the absence of field data at the largest deviation of JAMINLAKE simulation results from the LAKE36 (before corrections) simulation results. If the field data were available at the specific location of maximum absolute error (between JAMINLAKE and LAKE36 – before corrections), then a comparison using field data would show a significant error.

The error related to the logic issue is observed in dataset 4. As expected, for all other data sets very minor (if any) error is observed. Summary of data includes user-supplied number of layers and number of layers actually used in the model. The logic issue is observed only when these two numbers (of layers) are different.

A "Compare Files" utility tool was developed in JAVA and it compares any two ASCII/EBCDIC files. It reads both the files word-by-word and tries converting it to a number. Then the corresponding numbers are compared. Any error exceeding 0.001, between the corresponding numbers in the two files, is reported. Using this tool, the output files were compared and all the errors were found smaller than 0.001.

A visual comparison of ML2DPLOT1 and SURFER™ suggests that the trend developed in ML2DPLOT1 is accurate. However, both contours are not identical because of the difference in the plotting technique. ML2DPLOT1 has some disadvantages compared to SURFER™. SURFER™ provides several gridding techniques. It is faster than ML2DPLOT1. It also produces smoother contours by implementing smoothing algorithms. On the other hand, ML2DPLOT1 possesses several advantages over SURFER™. ML2DPLOT1 is tailored to a specific application. It is accessible on the

Internet. With very few modifications, this application could also be used for producing contours from any user-supplied data values.

A single overturn in the lake during the initial months of the year is visible in the figures 25 and 26. This phenomenon is described under Section 2.1. The lake gets almost completely mixed and hence the lake modeled is holomictic and monomictic.

7. CONCLUSION AND RECOMMENDATIONS

The following goals have been accomplished in this research:

1. LAKE36 is available on the Internet.
2. MINLAKE is re-engineered as JAMINLAKE using JAVA as an object oriented programming application.
3. The proof of principle for the hybrid concept used for complex engineering or ecological modeling is demonstrated in this research.

JAMINLAKE is an object-oriented application that is built on the hybrid concept. It can be systematically modified to incorporate additional heat supplying objects, simulation variables, simulation techniques, lake models, statistical calculations and lake treatment components. It can be modified to run entirely in the client's browser. JAVA is an interpreted language and so this modification could render it slower. This may become less significant over time as the bandwidth and speed of computational resources increases.

Minor errors in programming are often overlooked, especially if the field data closely fits the simulation results. Sometimes such errors could produce better result. So a thorough check of source code is very essential. Redeveloping LAKE36 as JAMINLAKE not only helped update the technology but also served as an excellent bug check for LAKE36.

JAVA is an excellent tool for creating reusable objects. However, it is inherently slow for repeated creation of instances of an object. For example, if a huge data set of points has to be maintained then an Array of x, y and z values can be accessed a lot faster than JAVA objects like Vectors that hold each point. However, JAVA objects provide a lot of other facilities like synchronization, dynamic size and reusability.

FORTRAN is not a hard coded language and it allows execution of programs containing certain types of errors. Thus, it could provide incorrect simulation results. The analyst has to be extremely careful in developing application using FORTRAN.

Future efforts could alter the model to include all the state variables of a lake and establish the lake water quality. This model can also be used to incorporate lake water treatment techniques and estimate the effects of treatment techniques on water quality of a lake. This model can also be used for fish habitat projection.

The present commercial simulation and modeling applications are client-side applications. The hybrid concept and online simulation applications can facilitate a more useful tool to run the simulation applications. Instead of limiting a user to purchase expensive simulation applications, the companies can charge them per usage over the Internet. This simulation application is the first step towards achieving this goal.

REFERENCES:

Adams, W.P., "Snow and ice on lake", In: D.M. Gary and D.H. Male (Editors), Handbook of snow, Pergamon Press, Oxford, Ch. 10, 1981.

Anderson, M.P., Woessner, W.W., "Applied Groundwater Modeling – Simulation of flow and advective transport", Academic Press, 1992, 28-96

Anton, E, "Principles of Object-Oriented software developments", Addison-Wesley Publishing Co., 1994, 3-31

Antoy, S., "Contour plotting for function specified at nodal points of a mesh based on a set of irregular profiles", Computer and Geosciences, v. 9, no. 2, 1983, 235-244

Ashton, G.D., "River and lake ice engineering", Water resources publications, Littleton, Co.

Ashton, G.D., "Freshwater ice growth, motion and decay", In dynamics of Snow and Ice Masses by S.C.Colbeck (editor), Academic Press, Inc., New York, 1980.

Baker, D.G., Ruschy, D.L., "The recent warming in eastern Minnesota shown by ground temperatures", Geophys. Res. Lett., 1993, 20(5), 371-374.

Bannister, T.T., "Production equations in terms of chlorophyll concentration, quantum yield, and upper limit to production", Limnology and Oceanography, Vol. 19, number 1, 1974a

Barrodale, I., Delves, L.M., Erickson, R.E., and Zala, C.A., "Computational experience with Marple's algorithm for autoregressive spectrum analysis", Geophysics, v. 48, no. 9, 1983, 1274-1286

Baumann, P.R., "Iso: a FORTRAN IV program for generating isopleth maps on small computers", Computers and Geosciences, v. 4, 1978, 23-32

Biffi, F., 1963, "Determining the Time Factor as a Characteristic Trait in the Self Purifying Power of Lago d'orta in Relation to a Continual Pollution," Ist. Ven Sci. Lettl. Arti. 121:131-136

Bourke, P.D., "A contouring subroutine", Byte, v. 12, 1987, 143-150

Braile, L.W., "Comparison of four random to grid methods", Computers and Geosciences, v. 4, 1978, 341-349

Bregoli, L.J., "A BASIC plotting subroutine sophisticated plotting with your MX-80", Byte, v. 7, no. 3, 1982, 142-156

Budd, T., "An Introduction to Object-Oriented Programming", 1991, 2-3

Campbell, B., Darnell, R., "Dynamic HTML in a week", Sams Publishing, First Edition, 1997

Canale, R.P., Squire, J., "A model for total phosphorous in Saginaw Bay", J. Great Lakes Res., 1976, 2(2): 364-373

Chapra, S.C., Reckhow, K.H., (a) "Engineering Approached for Lake Management", Volume 1: Data Analysis and Empirical Modeling, Boston, MA, 1983, 205-215.

Chapra, S.C., Reckhow, K.H., (b) "Engineering Approached for Lake Management", Volume 2: Mechanistic Modeling, Butterworth, Boston, MA, 1983, 492pp.

Dake, J.M.K., Harleman, D.R.F., "Thermal stratification in lakes: Analytical and laboratory studies", Water Resources Research, Vol. 5, number 2, 1969, pp: 404-495

David, W.E., Barry, D.K., Scott, N.W., "Object-oriented systems analysis- A model-driven approach", Yourdon Press Computing Series, 1992

Davis, M.W.D., David, M., "Generating bicubic spline coefficients on a large regular grid", Computers and Geosciences, v. 6, 1980, 1479-1503

Devereux, B.J., "The construction of digital terrain model on small computers", Computers and Geosciences, v.11, no. 10, 1963, 620-622

Dhamotharan, S., "A mathematical model for temperature and turbidity stratification dynamics in shallow reservoirs", Ph. D Thesis, University of Minnesota, 319, pp 1979

Diercks, P., "An algorithm for cubic spline fitting with convexity constraints", Computing, v. 24, 1980, 349-371

Dimitriadis, K., Tselentis, G.A., and Thanassoulas, K., "A BASIC program for 2-D spectral analysis of gravity data and source depth estimation", Computers and Geosciences, v. 13, no. 5, 1987, 549-560

Ellis, C.R., Stefan, H.G., "Density current intrusions and vertical eddy diffusivity in an ice-covered lake", Limnology and Oceanography, 1996

Ellis, C.R., Stefan, H.G. and Gu., R., "Water temperature dynamics and heat transfer beneath the ice cover of a lake", Limnology and Oceanography, 36 (2), 1991, 324-335

Esler, J.E., Preston, F.W., "FORTRAN IV program for the GE625 to compute the power spectrum of geological surfaces", Kansas Geol. Survey Computer Contrib., v. 16, 1967, 1-10

Eyton, J.R., "Raster contouring", Geo-Processing, v.2, 1984, 221-242

Fang, X., Stefan, H.G., "Temperature variability in lake sediments", Water resources research, Vol. 34, number 4, April 1998, 717-729

Fang, X., Ellis, C.R., Stefan, H.G., "Simulation and observation of ice formation (freeze over) in a lake", Cold regions science and technology, 24, 1996, 129-145

Fang, X., Stefan, H.G., "Long-term lake water temperature and ice-cover simulation measurements", Cold regions science and technology, 24, (1996) 289-304.

Fang, X., Stefan, H.G., "Temperature and DO simulations for a lake with ice-cover", St. Anthony Falls hydraulic laboratory, Project report 356, University of Minnesota, December 94

Fang, X., Stefan, H.G., "Validation of Water Quality Model MINLAKE96 with New Winter Data", Project report 390, USEPA, Washington, D.C., June 96

Ford, D.E., and Stefan, H.G., "Thermal predictions using integral energy model", Journal of Hydraulic division, ASCE, 106(1), 1980, 39-55

Gu, R., Stefan, H.G., "Year round temperature simulation of cold climatic lakes", Cold regions science and technology, 18(2), 1990, 147-160.

Harleman, D.R.F., "Hydrothermal modeling of reservoirs in cold regions; status and research needs, In: Proc. Cold Reg. Hydrol. Symp, AWRA, 1986, pp 39-49.

Hailper, B., "Multi-paradigm Languages and Environments", IEEE Software, 3(1), 1986, 6-9.

Hobbs, A., "Database programming with JDBC in 21 days", Sams Publishing, First Edition, 1997

Holroyed, M.T., and Bhattacharyya, B.K., "Automatic contouring geophysical data using bicubic spline interpolation", Geol. Survey of Canada, Paper 70-55, 1970, 40

Hondzo, M., Stefan, H.G., "Riverbed heat conduction prediction", Water resources research, 30(5), 1994, 1503-1513

Horstmann, C.S., Cornell, G., "Core JAVA –Fundamentals", Vol. 1, Sun Microsystems Press, A Prentice Hall Title, 1999

Horstmann, C.S., Cornell, G., "Core JAVA –Fundamentals", Vol. 2, Sun Microsystems Press, A Prentice Hall Title, 1999

Idso, S.B., Jackson, R.D., "Thermal radiation from the atmosphere", Journal of geophysics research, Vol. 73, number 23, October 20, 1969, 5397-5403

Inoue, H., "A least-squares smooth fitting for irregularly spaced data: finite element approach using cubic B-spline basis", Geophysics, v. 51, no. 11, 1986, 2051-2066

James, W.R., "FORTRAN IV program using double Fourier series for surface fitting of irregularly spaced data", Kansa Geol. Survey Computer Contrib., v. 5, 1966, 1-18

Jason, H., Crawford, W., "JAVA Servlet Programming", O'Reilly, 1st Edition, 1998

Jones, T.A., Hamilton, D.E., Johnson, C.R., "Contouring geologic surfaces with computer", 1942

Kane, V.E., Bogovich, C.L., Butz, T.R., Myers, D.E., "Interpolation of regional geochemistry using optimal interpolation parameters", Computers and Geosciences, v. 8, no. 2, 1982, 117-135

Light, P., "Analysis of high rates of snow-melting, Reports and papers, Snow-survey conference, Transactions, American Geophysical Union", 1941

Lisizka, T., "An interpolation method for an irregular net of nodes", Intern. Jour. Numerical Methods in Engineering, v. 20, 1984, 1599-1612

Mason, J.C., "BASIC matrix methods", Butterworths, London, 1984, 160

Moncur, M., "JavaScript 1.3 in 24 Hours", Sams Publishing, First Edition, 1999

O'Connor, D.J., Mueller, J.A., "Water Quality Model of Chlorides in the Great Lakes", J. Sanit. Eng. Div., Proc. Am. Soc. Civ. Eng., 1970, 96(SA4): 955-975

Oldknow, A., "Drawing curves, surfaces and solids: some recent applications of mathematics from computer graphics", Bull. Inst. Math. Applic, v. 23, 1987, 293-300.

Parker, S.P., "Environmental Science", McGraw-Hill Book Company, 1980, 424-428

Pivovarov,A.A., "Thermal conditions in freezing lakes and rivers", Wiley, New York, New York, 1972

Richardson, W.L., "An evaluation of the transport characteristics of the Saginaw Bay using a mathematical model of chloride", in Modeling Biochemical Processes in Aquatic Ecosystems, R.P. Canale, Ed. (Ann Arbor, MI: Ann Arbor Science Publishers, Inc., 1976), pp. 113-140

Riley, M.J., Stefan, H.G., "Dynamic Lake water Quality simulation model 'MINLAKE' ", St. Anthony Falls Hydraulic Laboratory, Project report 263, University of Minnesota, August 1987, 1-2

Rogers, D.F., and Adams, J.A., "Mathematical elements for computer graphics", McGraw-Hill, New York, 1976, 239

Sampson, R.J., Davis, J.C., "Three dimensional response surface program FORTRAN II for the IBM 1620 computer", Kansas Geol. Survey Computer Contrib., v. 10, 1967, 1-7

Scavia, D., Robertson, A., "Perspectives on lake ecosystem modeling", Ann Arbor Science Publishers Inc, 1979, 171-192

Snodgrass, W.J., "A predictive phosphorous model for Lakes-Development and testing", Ph.D Dissertation, University of North Carolina at Chapel Hill, 1974.

Snodgrass, W.J., O'Melia, C.R., "Predictive models for phosphorous in lakes", Environ. Sci. Technology, 1975, 9:937-944

Swain, C.J., "A FORTRAN IV program for interpolation of irregularly spaced data using the difference equations of minimum curvature", Computers and Geosciences, v.1, 1976, 231-240

Tartra, M.E., Freeman, W., Hopkin, A., "A FORTRAN implementation of univariate Fourier series density estimation", Commun. Stat.-Simulat., v. 15, 1986, 71-72

Taub, F.B., "Lakes and Reservoirs – Ecosystems of the world 23", Elsevier, 1984

Thomann, R.V., Mueller, J.A., "Steady state modeling of Toxic Chemicals – Theory and Application to PCB's in the Great Lakes and Saginaw Bay, in Physical behavior of PCBs in the Great Lakes", D. Mackay, S. Paterson, S.J. Eisenreich and M.S. Simmons (Eds.), Ann Arbor Science Pub., MI, 1983, pp. 283 - 309

Thomann, R.V., Mueller, J.A., "Principles of surface water quality modeling and control", 1987, 385-386,173-218

Todd, D.K., Groundwater Hydrology, John Wiley, New York, 1980.

Watson, D.F., "ACORD – Automatic contouring of raw data", Computer and Geosciences, v. 8, no. 1, 1982, 33-36

Watson, D.F., "Two images for three dimensions", Practical Computing, August, 1983, 104-107

Watson, D.F., "Contouring - A guide to the analysis and display of spatial data", Pergamon Press, 1992, 195-196

Watson, D.F., "Natural neighbor sorting", The Australian Computer Jour., v. 17, no. 4, 1985, 189-193

Wingle, W.C., "Examining common problems associated with various contouring methods, particularly inverse distance method, using shaded relief surfaces", Geotech 1992 conference proceedings, Lakewood, Colorado, September 1992

Yarnal, B., "A procedure for the classification of synoptic weather maps from gridded atmospheric pressure surface map", Computers and Geosciences, v. 10, no. 4, 1984, 397-410

Yates, S.R., "Contour: a FORTRAN algorithm for two dimensional high-quality contouring", Computers and Geosciences, v. 13, no. 1, 1987, 61-76

Yfantis, E.A., Borgman, L.E., "Fast Fourier transforms 2-3-5", Computers and Geosciences, v. 7, 1981, 99-108

Yeo, M.F., "An interactive contour plotting program", Engineering Computations, v. 1, 1984, 273-279

APPENDIX A – Partial list of Lake treatment methods

[Riley and Stefan, August 1987]

INFLOW CONTROL METHODS

- Bio-filtration and uptake of nutrients
  - Detention ponds utilizing macrophytes to remove nutrients
  - Wetland filtration of nutrients
  - Seepage Trenches
- Chemical Treatments
  - Aluminum oxide columns
  - Aluminum sludge
  - Ferric aluminum blocks
  - Ferric iron treatment plant
- Sediment Removal
  - Sediment retention basins
  - Perimeter road sediment traps
- Storm-water
  - Aluminum sludge treatment
  - Detention and storage
  - Sewage overflow elimination
  - Underground filtration
  - Wetland filtration
- Wastewater
  - Central sewerage
  - Evapo-transpiration bed
  - Grey water management
  - Mound systems
  - Sand filter
  - Septic tank systems
  - Tertiary treatment
  - Wastewater modification

WATERSHED CONTROL METHODS

- Agriculture
  - Animal waste contamination
  - Animal waste management
  - Waste storage pond
  - Waste treatment lagoons
  - Filter strips
- Urban
  - Product and use modifications to reduce nutrients
  - Street Sweeping
  - Use of porous concrete
- Lake Watershed
  - Diversion
  - Greenbelts
  - Perimeter road sediment traps
  - Regrade shore banks
  - Shore erosion protection stabilization
- Regulatory control
  - Boating regulations
  - Conservation easements
  - Control of regulation removal
  - Erosion control ordinance
  - Improve enforcement of existing regulation
  - Population density control
  - Sanitary codes
  - Shoreland protection ordinance
  - Subdivision regulations
  - Wetland conservancy zoning
  - Zoning controls

INLAKE CONTROL METHODS

Lake Deepening

- Sediment Removal
  - Dredging
  - Excavation
- In-situ consolidation - physical
  - Aeration
  - Nitrogen addition
  - Hydrogen peroxide addition
  - Ozone addition
  - Proprietary organism addition

Sediment Nutrient Inactivation

- Chemical
  - Hypolimnetic aeration
  - Hypolimnetic oxygenation
  - Aluminum injection into sediment
  - Aluminum floc layering
  - Fly ash layering
  - Water treatment plant sludge
- Physical
  - Plastic, vinyl and synthetic liners
  - Clay-Montmorillite layering
  - Sand layering

Water column nutrient inactivation

- Aluminum treatment
- Potassium aluminum sulfate

Macrophyte

- Plant harvesting
- Macrophyte herbicides

o Ferric chloride o Zirconium o Lanthanum <u>Water column mixing</u> o Air o Water jet <u>Biological Process treatment</u> o Copper sulfate o Cuprose o Cutrine <u>Nutrient outflow acceleration</u> o Selective discharge o Flushing/dilution o Hypolimnetic withdrawal	o Bottom barriers o Shades o Dyes o Fish predation o Shellfish o Insects o Disease organisms o Competitive plants o Hydraulic dredging o Diver dredging o Manual harvesting o Use of rotovator o Shallow water tillage o Drawdown o Fish o Pesticides

APPENDIX B – Sample input file

Input file used for first data set:

```
Simulation for lake : huang20, for user : fangxu@hal.lamar.edu

MNDULU86

32 1

4 16 1986 10 16 1992

28 14.600000381469727 637.0

0.44 20.0

0.01 0.01

1.0 1.0

0.035 550.0

0.3 2.6 0.27 11.35 0.1

0.17 0.55 1.6 0.34 0.8 40.0

0.0 0.0

0.025 0.050 0.100 0.200 0.600 1.000 1.590 2.180 2.770 3.360

3.950 4.550 5.140 5.730 6.320 6.910 7.500 8.090 8.680 9.270

9.860 10.450 11.050 11.640 12.230 12.820 13.410 14.000

4.00 4.00 4.00 4.00 4.00 4.00 4.00 4.00 4.00 4.00

4.00 4.00 4.00 4.00 4.00 4.00 4.00 4.00 4.00 4.00

4.00 4.00 4.00 4.00 4.00 4.00 4.00 4.00

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

```
0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020

0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020

0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020 0.0020

1

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5

0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5

0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5

10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

0.1 0.4 0.1 9.6

1.0 -2.0

0 0 0

5 0 106

1 1

5

4 3

  1986  2

  428 513

  1987  1

  113

  1988  1

  213
```

```
        0.0012 5.5E-4 0.0012 0.0012

        0.0012 0.00168 0.0012 0.0012

3 2   4 28 1986

5.0

1  6

0.0 1.0 2.0

12.0 11.9 10.0

9.8 10.2 10.9

3 2   5 13 1986

5.0

1  6

0.0 1.0 2.0

12.0 11.9 10.0

9.8 10.2 10.9

3 2   1 13 1987

5.0

1  6

0.0 1.0 3.0

0.8 2.0 3.3

14.0 14.6 11.2

3 2   2 13 1988

5.0

1  6

0.0 1.0 3.0

0.8 2.0 3.3

14.0 14.6 11.2
```

APPENDIX C – Sample output file

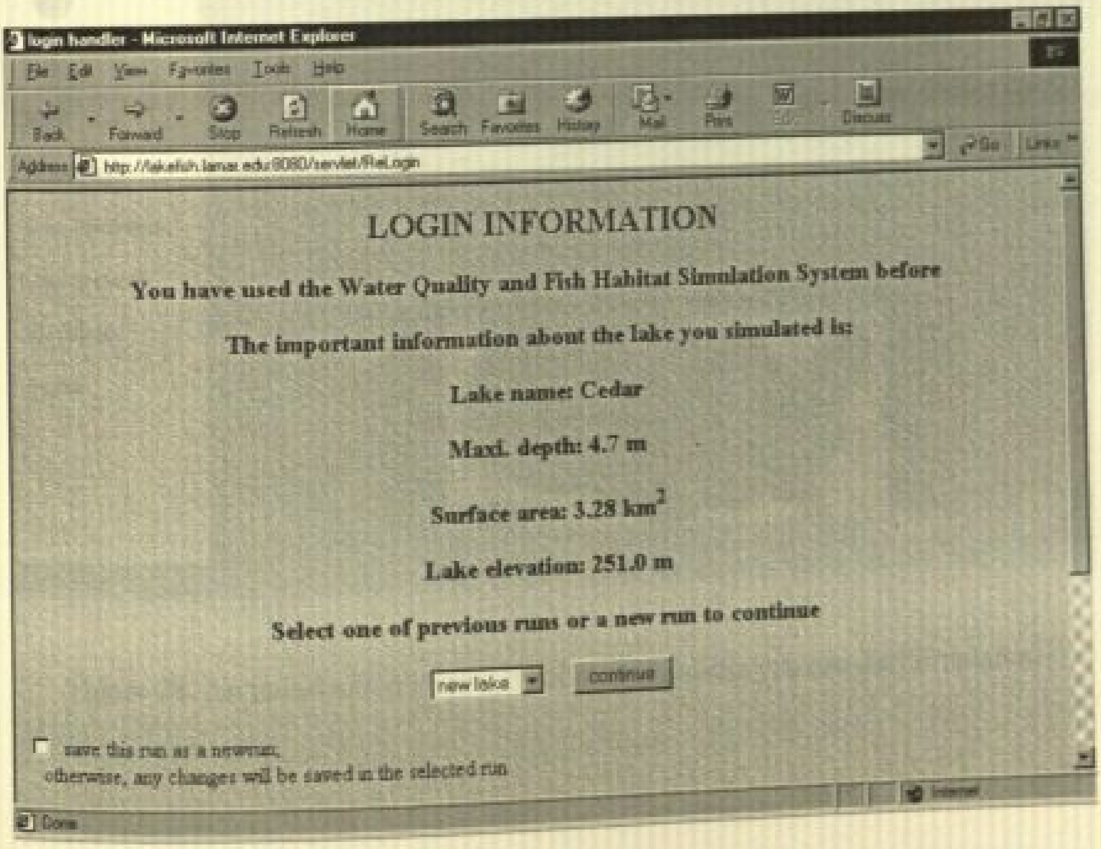
Output data set for the first month of the first simulation year of the first data set:

The first line gives month of the year, start day of the month, number of days in the month, year of simulation and number of layers. The first set of data in the first 3 lines (28 elements equal to the number of layers) is the value of depth of each layer. The next sets are temperature at each day starting form the first day and with an interval of 1 day.

```
4 16 15 1986 28
 .019  .056  .113  .275  .600  1.047  1.590  2.180  2.770  3.360
3.953  4.548  5.140  5.730  6.320  6.910  7.500  8.090  8.680  9.270
9.860 10.453 11.047 11.640 12.230 12.820 13.410 14.153
5.145  5.145  5.145  5.145  5.145  5.145  5.145  5.145  5.145  4.604
4.499  4.411  4.345  4.296  4.255  4.230  4.213  4.213  4.213  4.213
4.213  4.213  4.213  4.213  4.213  4.213  4.213  4.213
5.111  5.111  5.111  5.111  5.111  5.111  5.111  5.111  5.111  5.111
5.111  5.111  5.111  4.549  4.479  4.433  4.408  4.408  4.408  4.408
4.408  4.408  4.408  4.408  4.408  4.408  4.408  4.408
5.429  5.429  5.429  5.429  5.429  5.429  5.429  5.429  5.429  5.429
5.429  5.429  5.429  4.876  4.714  4.630  4.590  4.590  4.590  4.590
4.590  4.590  4.590  4.590  4.590  4.590  4.590  4.590
6.093  6.093  6.093  6.093  6.093  6.093  6.093  6.093  6.093  6.093
6.093  6.093  6.093  5.186  4.964  4.844  4.777  4.771  4.771  4.771
4.771  4.771  4.771  4.771  4.771  4.771  4.771  4.771
5.820  5.820  5.820  5.820  5.820  5.820  5.820  5.820  5.820  5.820
5.820  5.820  5.820  5.537  5.210  5.056  4.964  4.946  4.946  4.946
4.946  4.946  4.946  4.946  4.946  4.946  4.946  4.946
6.078  6.078  6.078  6.078  6.078  6.078  6.078  6.078  6.078  6.078
6.078  6.078  6.078  5.809  5.508  5.319  5.196  5.140  5.137  5.137
5.137  5.137  5.137  5.137  5.137  5.137  5.137  5.137
6.931  6.931  6.931  6.931  6.931  6.931  6.931  6.931  6.931  6.667
6.556  6.450  6.332  6.079  5.781  5.571  5.425  5.342  5.324  5.324
5.324  5.324  5.324  5.324  5.324  5.324  5.324  5.324
9.293  9.293  9.293  9.293  9.293  9.293  9.293  7.763  7.530  7.214
6.983  6.792  6.594  6.319  6.018  5.795  5.635  5.535  5.502  5.502
5.502  5.502  5.502  5.502  5.502  5.502  5.502  5.502
9.087  9.087  9.087  9.087  9.087  9.087  9.087  8.367  7.857  7.501
7.224  6.990  6.756  6.476  6.185  5.960  5.799  5.695  5.662  5.662
5.662  5.662  5.662  5.662  5.662  5.662  5.662  5.662
7.256  7.256  7.256  7.256  7.256  7.256  7.256  7.256  7.256  7.256
7.256  7.256  7.256  7.256  6.363  6.134  5.969  5.861  5.821  5.821
5.821  5.821  5.821  5.821  5.821  5.821  5.821  5.821
8.499  8.499  8.499  8.499  8.499  8.499  8.052  7.880  7.756  7.655
7.582  7.514  7.438  7.295  6.679  6.335  6.152  6.036  5.982  5.982
5.982  5.982  5.982  5.982  5.982  5.982  5.982  5.982
8.495  8.495  8.495  8.495  8.495  8.495  8.495  8.258  8.072  7.920
7.803  7.695  7.578  7.367  6.860  6.505  6.306  6.187  6.131  6.131
6.131  6.131  6.131  6.131  6.131  6.131  6.131  6.131
```

```
8.263  8.263  8.263  8.263  8.263  8.263  8.263  8.263  8.263  8.263
8.028  7.881  7.721  7.469  7.023  6.671  6.459  6.336  6.277  6.277
6.277  6.277  6.277  6.277  6.277  6.277  6.277  6.277
11.123 11.123 11.123 11.123 11.123  9.935  9.470  9.163  8.953  8.766
8.495  8.238  7.999  7.687  7.251  6.890  6.655  6.519  6.444  6.436
6.436  6.436  6.436  6.436  6.436  6.436  6.436  6.436
9.401  9.401  9.401  9.401  9.401  9.401  9.401  9.401  9.401  9.026
8.729  8.436  8.156  7.820  7.402  7.046  6.805  6.663  6.584  6.578
6.578  6.578  6.578  6.578  6.578  6.578  6.578  6.578
```

APPENDIX D – SCREEN CAPTURES & STEP-BY-STEP SIMULATION OF DATA SET 1 – CEDAR LAKE


login handler - Microsoft Internet Explorer

Address: http://lakefish.lamar.edu:8080/servlet/FirLogin

LOGIN INFORMATION

You have used the Water Quality and Fish Habitat Simulation System before

The important information about the lake you simulated is:

Lake name: Cedar

Maxi. depth: 4.7 m

Surface area: 3.28 km^2

Lake elevation: 251.0 m

Select one of previous runs or a new run to continue

new lake | continue

☐ save this run as a newrun, otherwise, any changes will be saved in the selected run

Figure 27. Simulation of a new lake – first screen

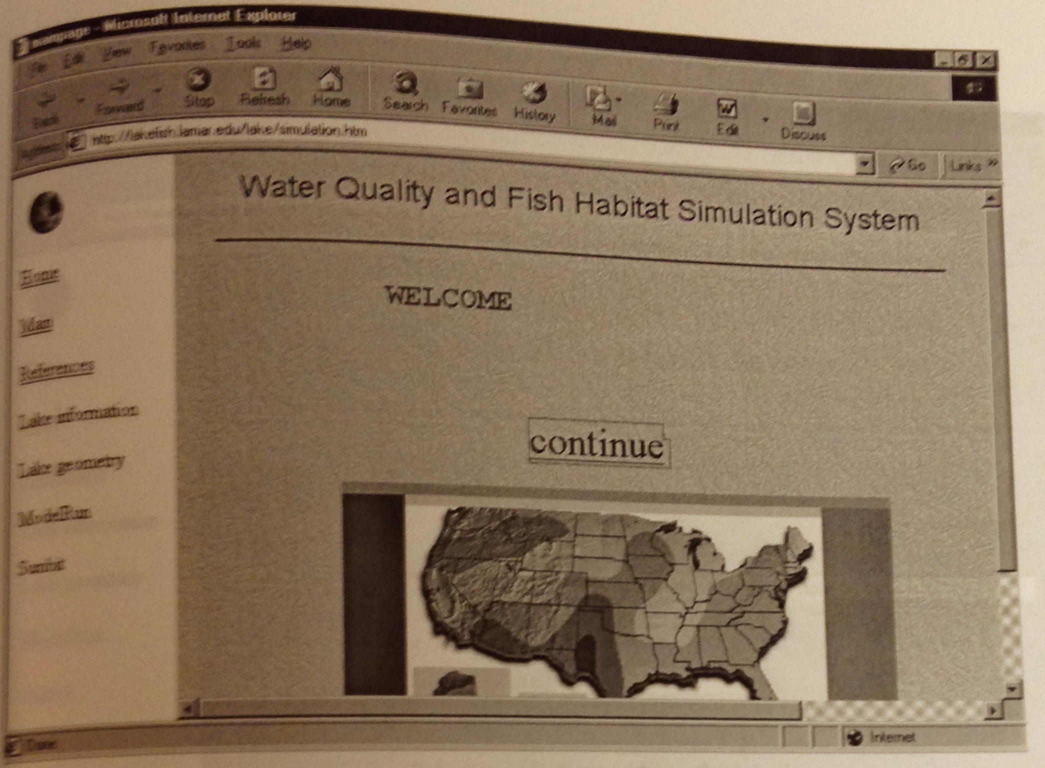
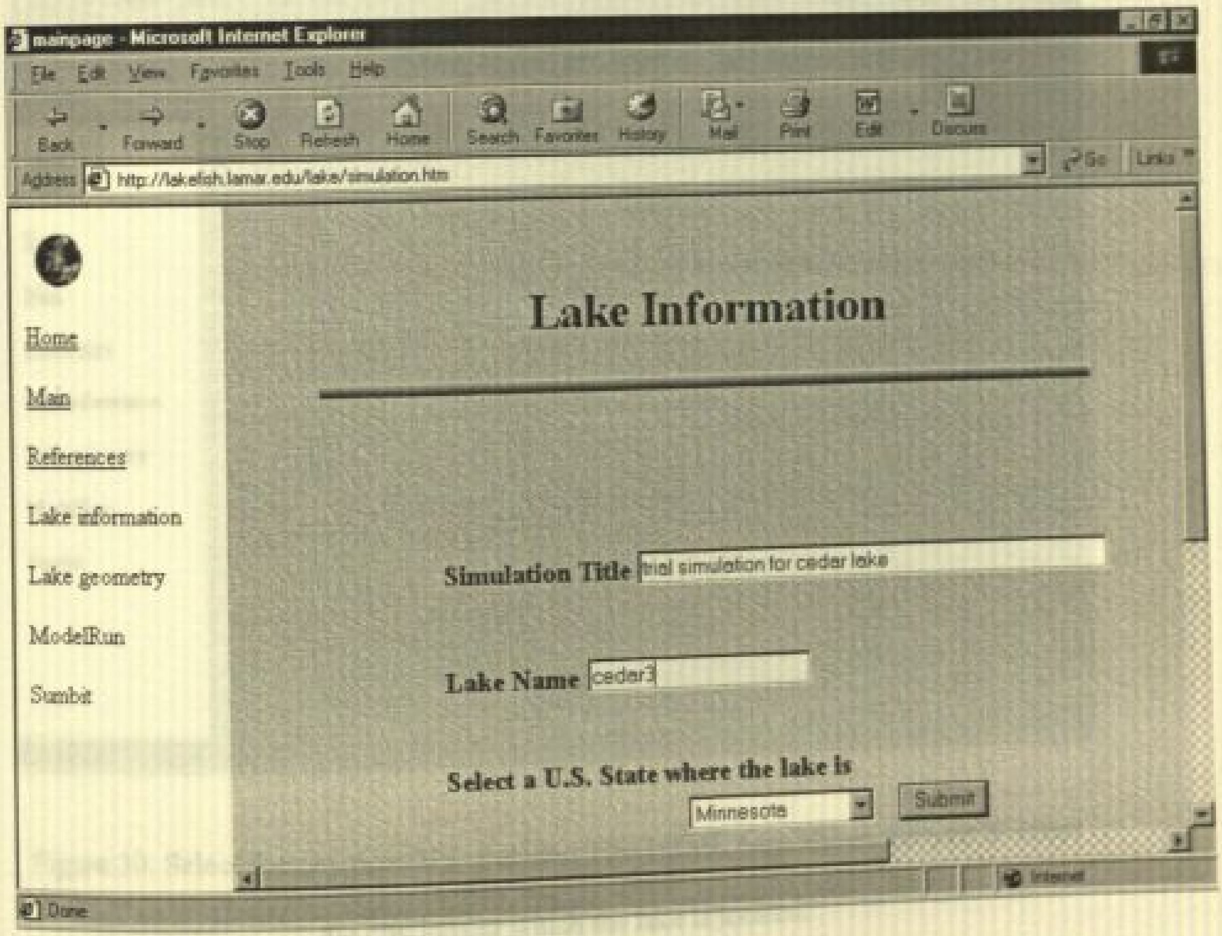

mainpage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu/lake/simulation.htm
Go Links
Home
Main
References
Lake information
Lake geometry
ModelRun
Sunfish
Water Quality and Fish Habitat Simulation System
WELCOME
continue
Done
Internet

Figure 28. An introduction screen. By clicking continue you can start simulation.


mainpage - Microsoft Internet Explorer

Address: http://lakefish.lamar.edu/lake/simulation.htm

File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Go Links
Home
Main
References
Lake information
Lake geometry
ModelRun
Sumbit
Submit
Done
Internet

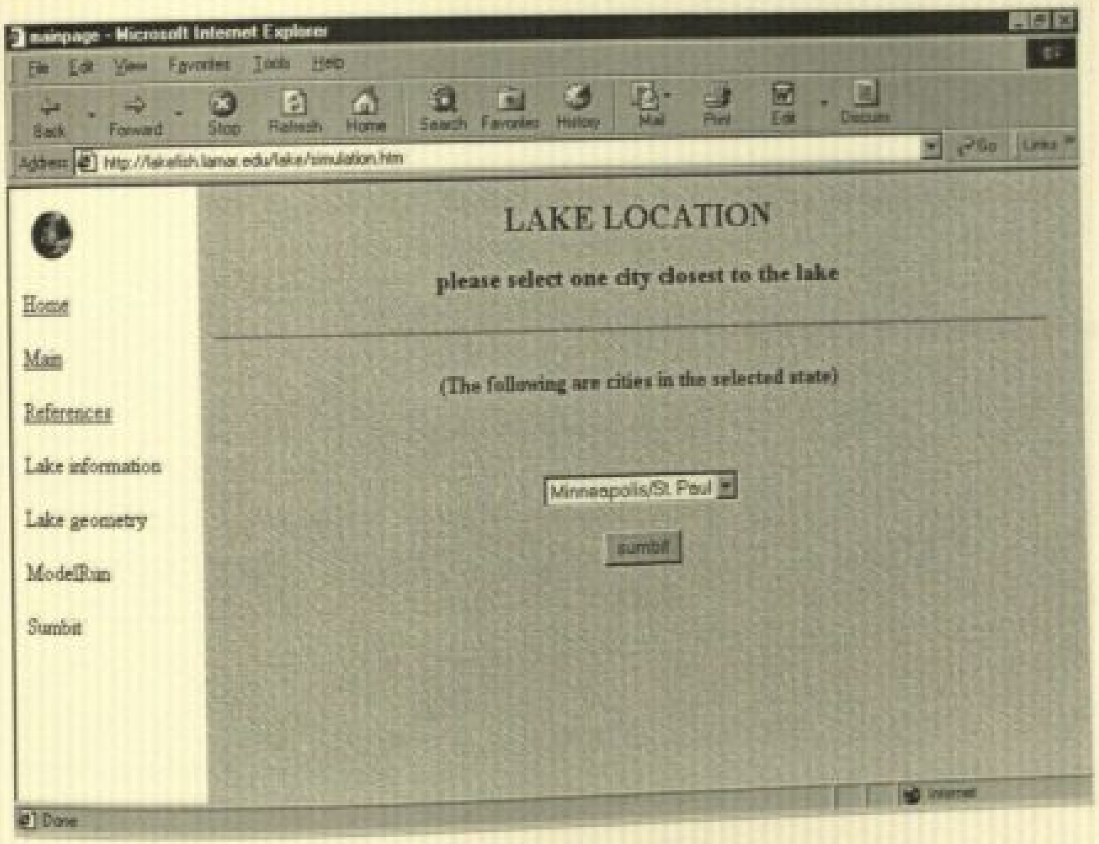
Lake Information

Simulation Title: trial simulation for cedar lake

Lake Name: cedar3

Select a U.S. State where the lake is: Minnesota

Figure 29. Basic Lake Information Page; This HTML page allows the user to define the title of simulation, name of the lake and the state in which this lake is located

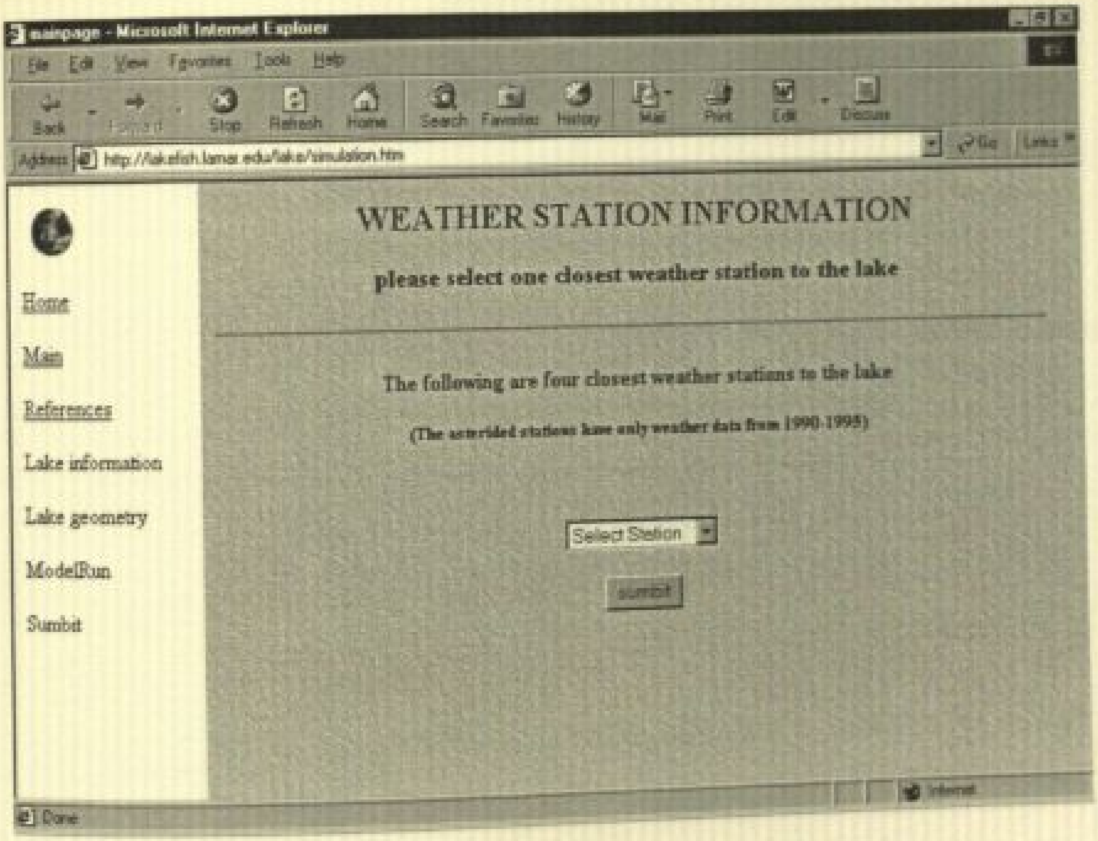

mainpage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu/lake/simulation.htm
Go Links
Home
Main
References
Lake information
Lake geometry
ModelRun
Sumbit
Minneapolis/St. Paul
sumbit
Done
Internet

LAKE LOCATION

please select one city closest to the lake

(The following are cities in the selected state)

Figure 30. Select the city nearest to the lake; This HTML page allows the user to select the city in/near which the lake is located.


mainpage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu/lake/simulation.htm
Go Links
Home
Main
References
Lake information
Lake geometry
ModelRun
Sumbit
Select Station
sumbit
Done
Internet

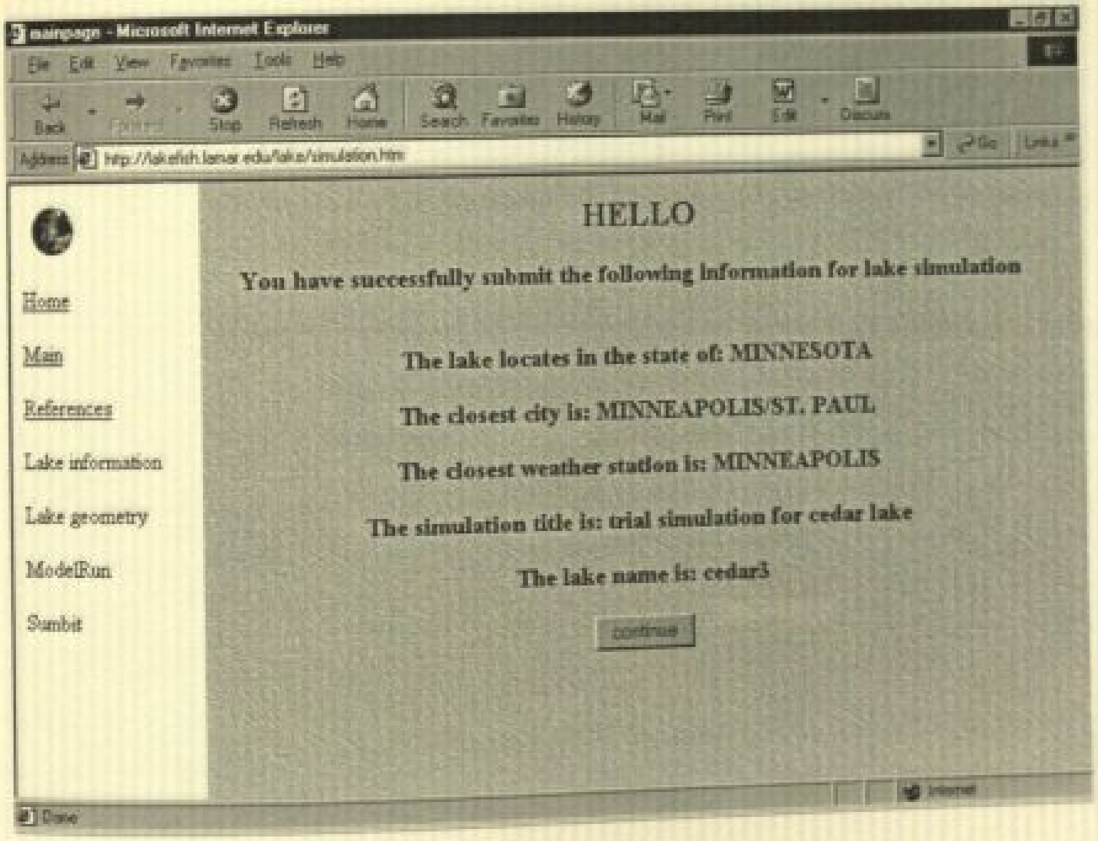
WEATHER STATION INFORMATION

please select one closest weather station to the lake

The following are four closest weather stations to the lake

(The asterided stations have only weather data from 1990-1995)

Figure 31. Select the weather station nearest to the lake; This figure allows the user to select the near weather station from which meteorological data can be obtained.


mainpage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu/lake/simulation.htm
Go Links
Home
Main
References
Lake information
Lake geometry
ModelRun
Sumbit
continue
Done
Internet

HELLO

You have successfully submit the following information for lake simulation

The lake locates in the state of: MINNESOTA

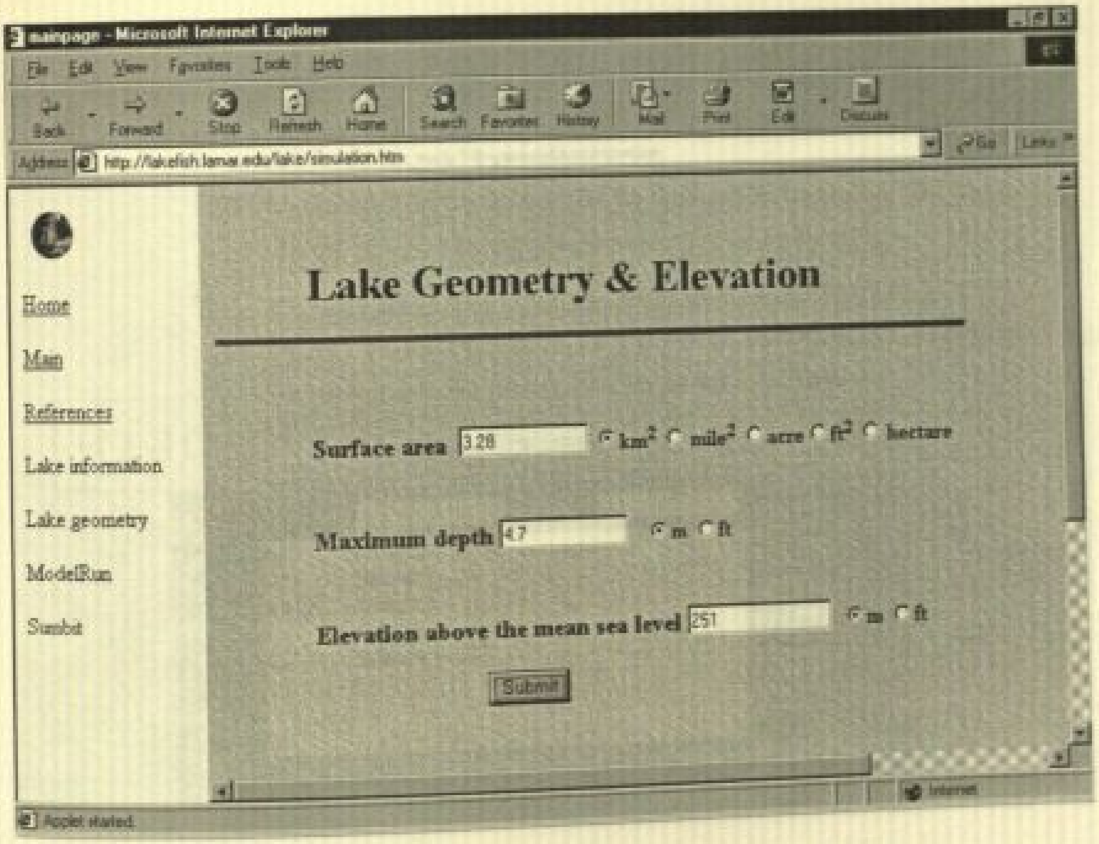
The closest city is: MINNEAPOLIS/ST. PAUL

The closest weather station is: MINNEAPOLIS

The simulation title is: trial simulation for cedar lake

The lake name is: cedar3

Figure 32. Chosen weather station display; This HTML page simply displays the simulation parameters chosen.


mainpage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu/lake/simulation.htm
Home
Main
References
Lake information
Lake geometry
ModelRun
Sumbit
Submit
Applet started
Internet

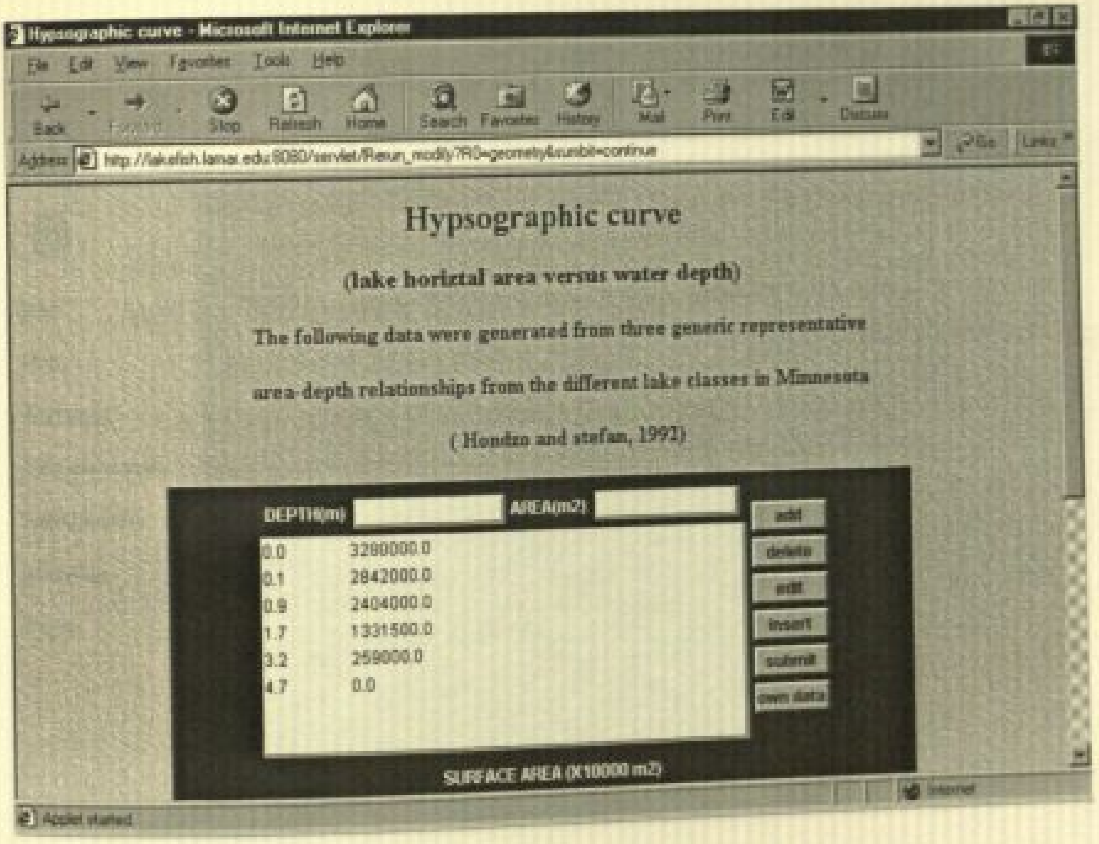
Lake Geometry & Elevation

Surface area 3.28 [✓ km^2] mile^2 acre ft^2 hectare

Maximum depth 4.7 [✓ m] ft

Elevation above the mean sea level 251 [✓ m] ft

Figure 33. Lake Geometry HTML Page; This HTML page allows the user to provide the surface area of the lake, maximum depth of the lake and the elevation of the lake above the mean sea level.


Hypsographic curve - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu:8080/servlet/Return_modify?R0=geometry&sunbit=continue
Go Links
add
delete
edit
insert
submit
own data
Internet
Applet started

Hypsographic curve

(lake horiztal area versus water depth)

The following data were generated from three generic representative

area-depth relationships from the different lake classes in Minnesota

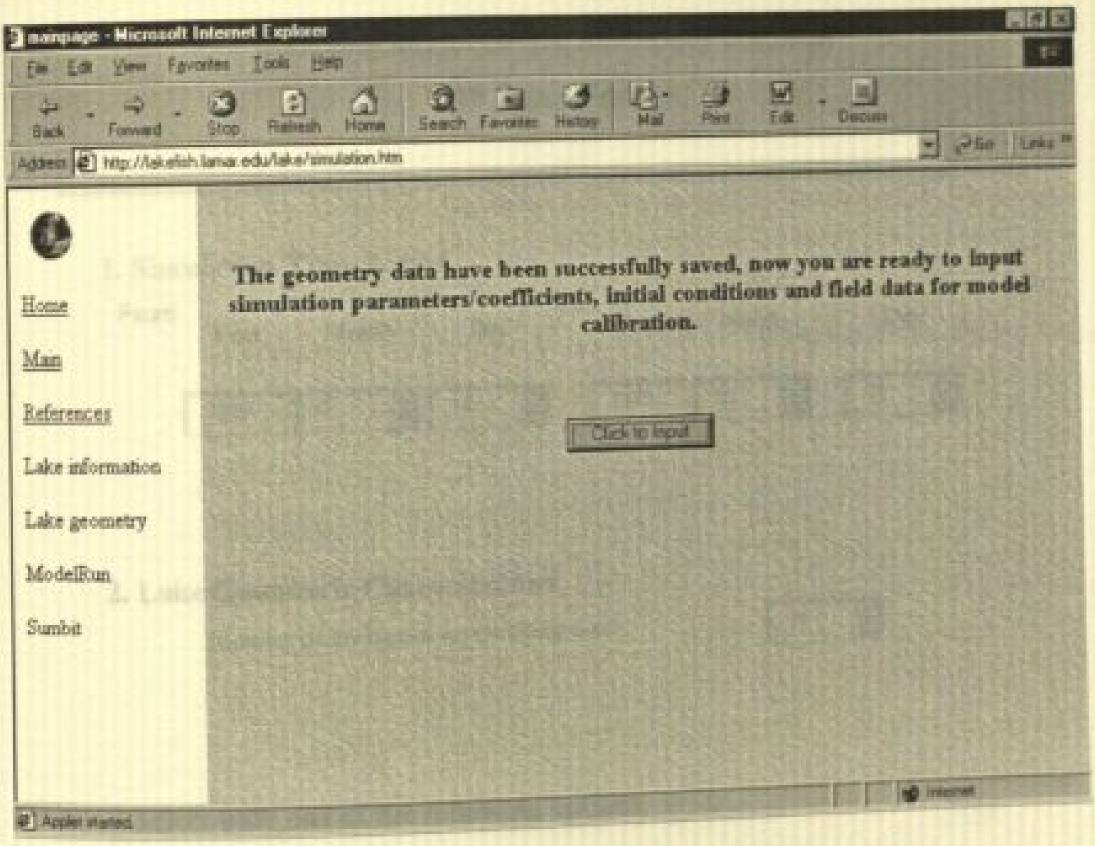
(Hondzo and stefan, 1992)

DEPTH(m) AREA(m2)

```
0.0     3280000.0
0.1     2842000.0
0.9     2404000.0
1.7     1331500.0
3.2     259000.0
4.7     0.0
```

SURFACE AREA (X10000 m2)

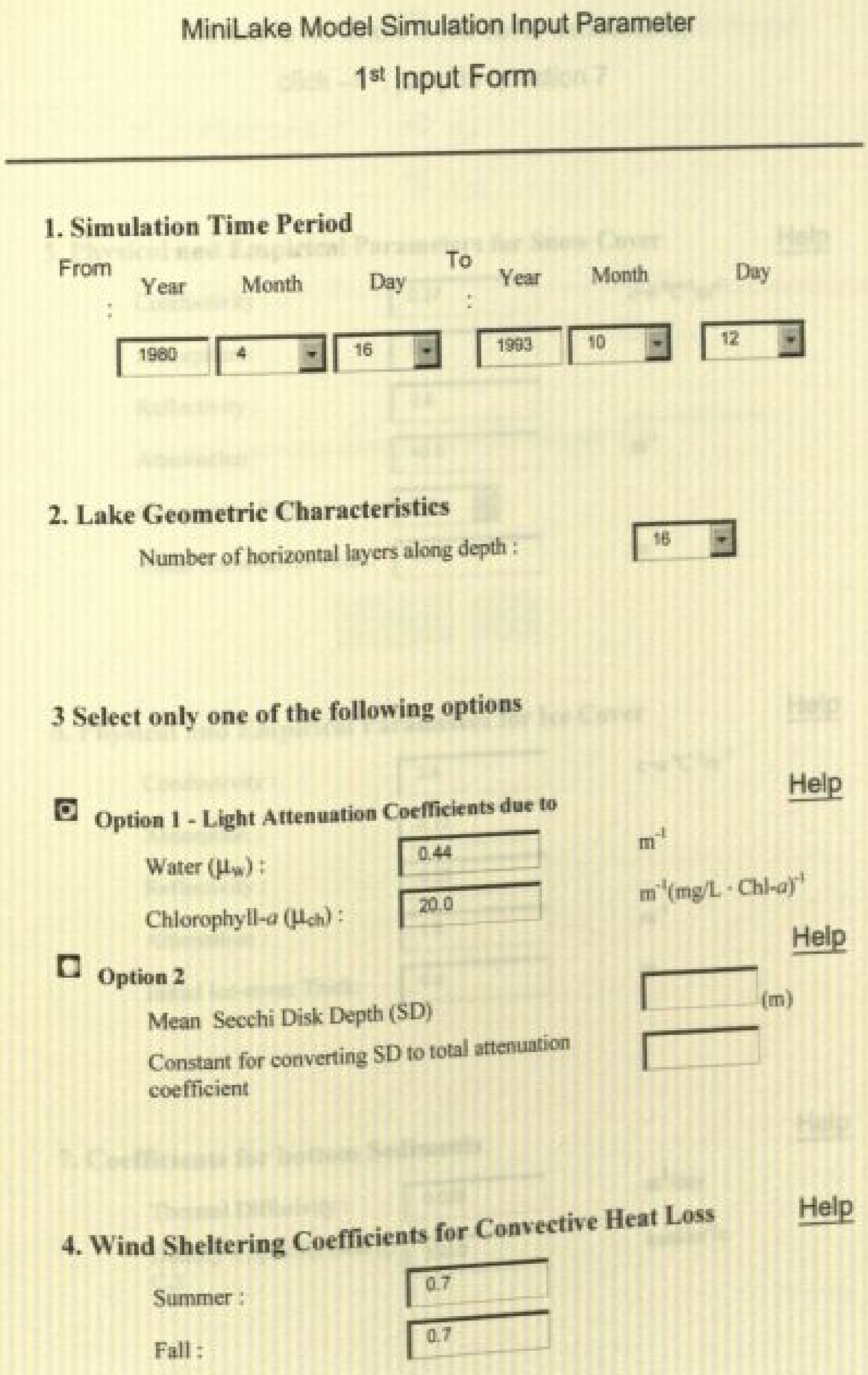
Figure 34. Lake Geometry Applet Page; Based on the data supplied by the user in the earlier email page, a default lake geometry is provided. The user can either alter this geometry or create a new one altogether.


mainpage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu/lake/simulation.htm
Home
Main
References
Lake information
Lake geometry
ModelRun
Sumbit
Click to Input
Applet started
Internet

The geometry data have been successfully saved, now you are ready to input simulation parameters/coefficients, initial conditions and field data for model calibration.

Figure 35. Lake Geometry Confirmation Page; This HTML page is generated following a creating of area.sdf file in the user directory. This page simply gives information that lake geometry has been recorded.

The following pages are screen captured HTML pages of input data forms.



MiniLake Model Simulation Input Parameter

1st Input Form

1. Simulation Time Period

	Year	Month	Day		Year	Month	Day
From :	1980	4	16	To :	1993	10	12

2. Lake Geometric Characteristics

Number of horizontal layers along depth : 16

3 Select only one of the following options

[✓] **Option 1 - Light Attenuation Coefficients due to** — Help

Water (μ_w) :	0.44	m^{-1}
Chlorophyll-*a* (μ_{ch}) :	20.0	$m^{-1}(mg/L \cdot Chl\text{-}a)^{-1}$

☐ **Option 2** — Help

Mean Secchi Disk Depth (SD)		(m)
Constant for converting SD to total attenuation coefficient		

4. Wind Sheltering Coefficients for Convective Heat Loss — Help

Summer :	0.7
Fall :	0.7

If there is no Snow & Ice cover simulations,

click --> , jump to question 7

5. Physical and Empirical Parameters for Snow Cover Help

Conductivity :	0.27	$c\cdot w\,^{\circ}C^{-1}m^{-1}$
Absorption :	0.34	
Reflectivity :	0.8	
Attenuation :	40.0	m^{-1}
Compaction Factor :	0.4	
Initial Snow-cover Thick:	0.0	m

6. Physical and Empirical Parameters for Ice Cover Help

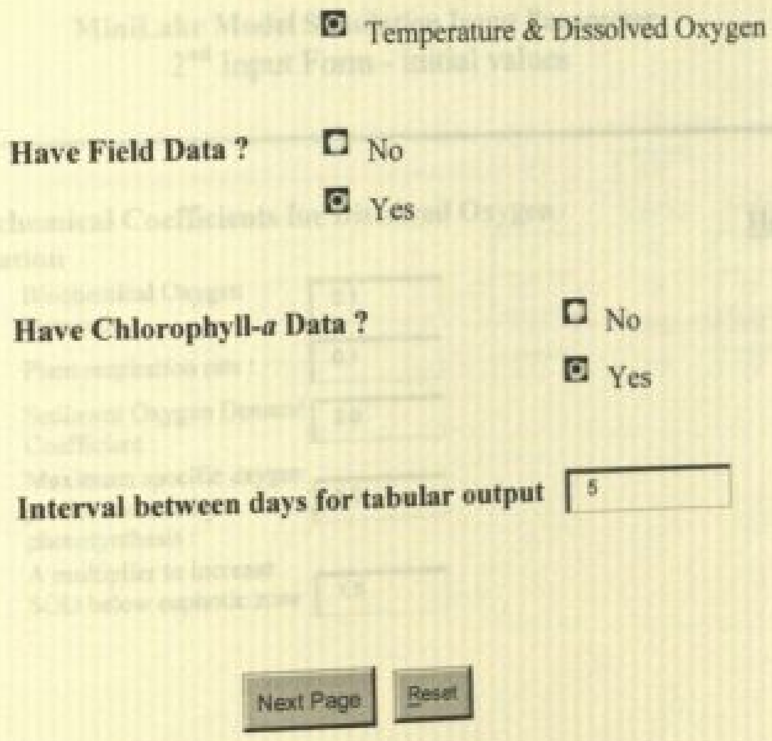
Conductivity :	2.6	$c\cdot w\,^{\circ}C^{-1}m^{-1}$
Absorption :	0.17	
Reflectivity :	0.55	
Attenuation :	1.6	m^{-1}
Initial Ice-cover Thick:	0.0	m

7. Coefficients for bottom Sediments Help

Thermal Diffusivity :	0.035	m^2/day
Density x Specific Heat (ρ C_p)	550.0	kcal/m^3/c

8. Control Parameters for Simulation

Simulate ? ☐ Temperature Only



[✓ Temperature & Dissolved Oxygen]

Have Field Data ? No [✓ Yes]

Have Chlorophyll-*a* Data ? No [✓ Yes]

Interval between days for tabular output 5

MiniLake Model Simulation Input Parameter
2nd Input Form - initial values

---

9. Biochemical Coefficients for Dissolved Oxygen Simulation **Help**

Biochemical Oxygen Demand :	0.1
Plant respiration rate :	0.1
Sediment Oxygen Demand Coefficient :	2.0
Maximum specific oxygen production rate by photosynthesis :	9.0
A multiplier to increase SOD below euphotic zone :	1.5

10. Initial Depth *(*required)* [✓ m] feet

Set default depthes here -->

Layer	Value	Layer	Value	Layer	Value	Layer	Value
1	0.02	2	0.04	3	0.1	4	0.15
5	0.2	6	0.25	7	0.45	8	0.5
9	1.0	10	1.5	11	2.0	12	2.5
13	3.0	14	3.5	15	4.0	16	4.4

11. Initial Temperature in [✓ °C] °F

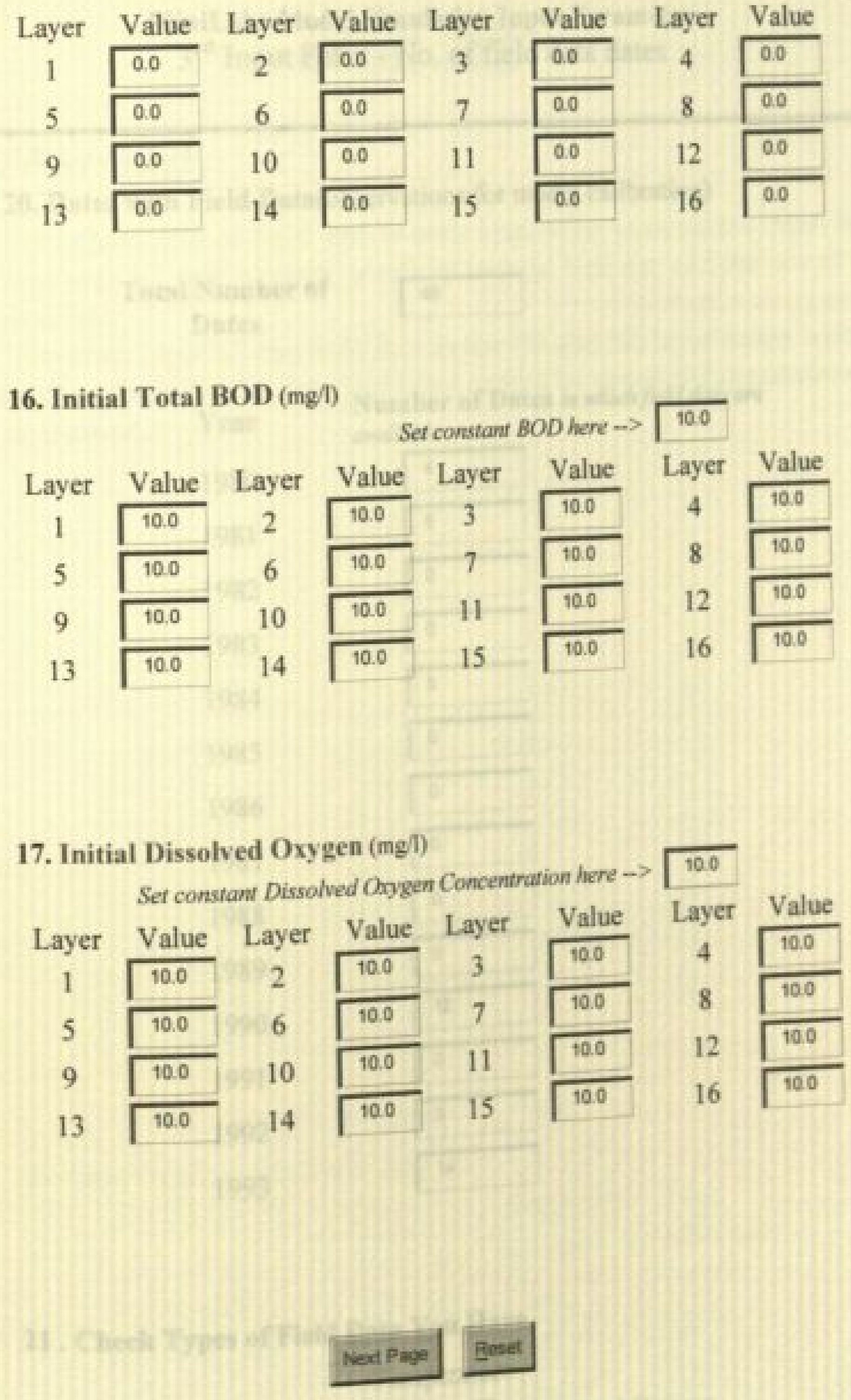
Set constant temperatures here --> 4.0

Layer	Value	Layer	Value	Layer	Value	Layer	Value
1	4.0	2	4.0	3	4.0	4	4.0
5	4.0	6	4.0	7	4.0	8	4.0
9	4.0	10	4.0	11	4.0	12	4.0
13	4.0	14	4.0	15	4.0	16	4.0

12. Initial Suspended Solids Concentration (mg/l)

Set constant Suspended Solids Concentrations here --> 0.0

Layer	Value	Layer	Value	Layer	Value	Layer	Value
1	0.0	2	0.0	3	0.0	4	0.0
5	0.0	6	0.0	7	0.0	8	0.0
9	0.0	10	0.0	11	0.0	12	0.0
13	0.0	14	0.0	15	0.0	16	0.0



Layer	Value	Layer	Value	Layer	Value	Layer	Value
1	0.0	2	0.0	3	0.0	4	0.0
5	0.0	6	0.0	7	0.0	8	0.0
9	0.0	10	0.0	11	0.0	12	0.0
13	0.0	14	0.0	15	0.0	16	0.0

16. Initial Total BOD (mg/l)

Set constant BOD here --> 10.0

Layer	Value	Layer	Value	Layer	Value	Layer	Value
1	10.0	2	10.0	3	10.0	4	10.0
5	10.0	6	10.0	7	10.0	8	10.0
9	10.0	10	10.0	11	10.0	12	10.0
13	10.0	14	10.0	15	10.0	16	10.0

17. Initial Dissolved Oxygen (mg/l)

Set constant Dissolved Oxygen Concentration here --> 10.0

Layer	Value	Layer	Value	Layer	Value	Layer	Value
1	10.0	2	10.0	3	10.0	4	10.0
5	10.0	6	10.0	7	10.0	8	10.0
9	10.0	10	10.0	11	10.0	12	10.0
13	10.0	14	10.0	15	10.0	16	10.0

MiniLake Model Simulation Input Parameter

3rd Input Form - No. of field data dates

---

20. Dates with Field Data(observations for model calibration)

Total Number of Dates 40

Year	**Number of Dates** *in which field data are available*
1980	4
1981	5
1982	0
1983	0
1984	5
1985	0
1986	0
1987	0
1988	0
1989	0
1990	12
1991	0
1992	0
1993	14

21 . Check Types of Field Data You Have

☐ Temperature

☐ Dissolved Oxygen

Note/Warning:

1. If you have lots of field data in each simulation year, please input field data year by year or even month by month. You can add the rest of field data later.
2. You must input all the field data in the 4th and 5th input forms within 8 hours, otherwise the data you input cannot be saved and the session will be expired.

MiniLake Model Simulation Input Parameter

4th Input Form -Field data infomation in each date

22 Field Data information in 1980

Month		Day	No. of Data	Chlorophyll-*a* (mg/l)	
				Epilimnion	Hypolimnion
6	/	3	4	0.0073	0.0073
7	/	7	4	0.66	0.66
8	/	6	4	0.22	0.22
9	/	5	4	0.175	0.175

23 Field Data information in 1981

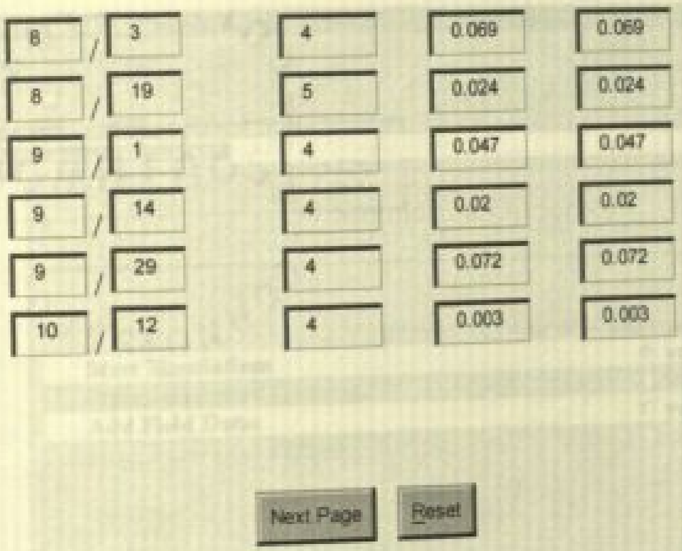
Month		Day	No. of Data	Chlorophyll-*a* (mg/l)	
				Epilimnion	Hypolimnion
5	/	7	4	0.075	0.075
6	/	5	4	0.0045	0.0045
7	/	7	4	0.034	0.034
8	/	3	4	0.035	0.035
9	/	2	4	0.054	0.054

24 Field Data information in 1984

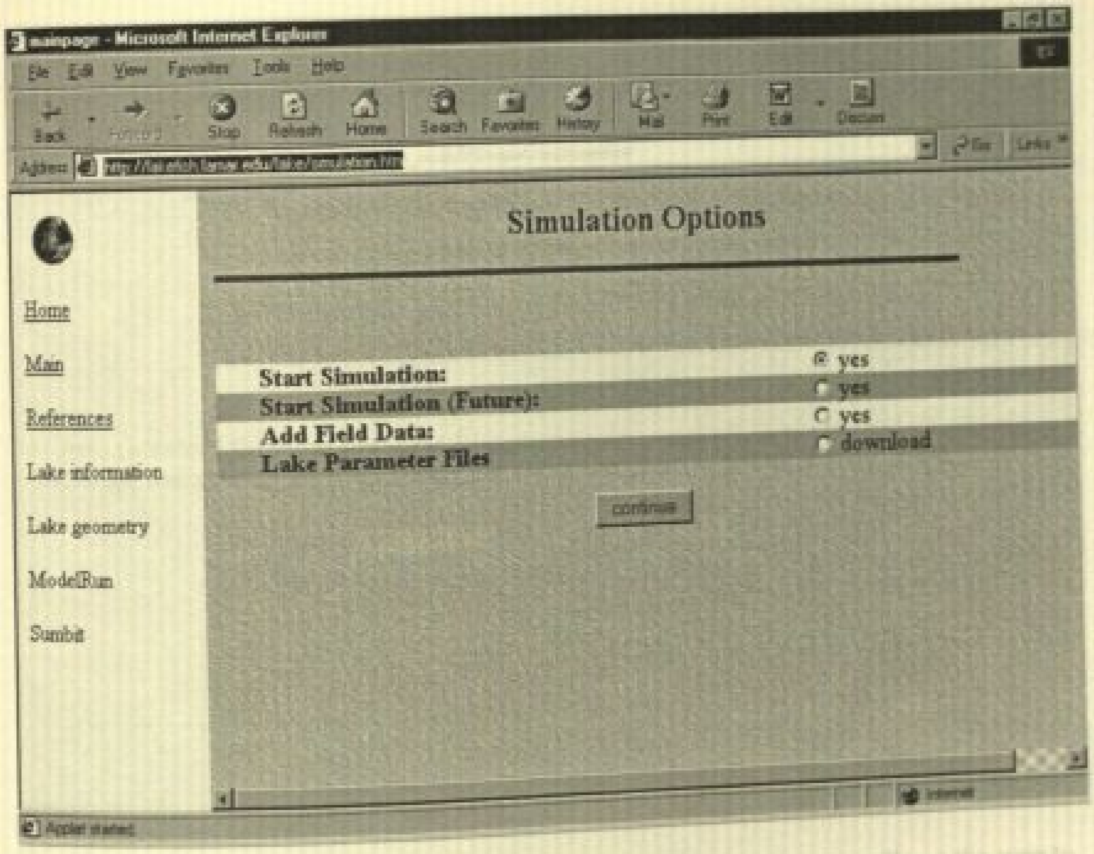
Month		Day	No. of Data	Chlorophyll-*a* (mg/l)	
				Epilimnion	Hypolimnion
5	/	21	4	0.0062	0.0062
6	/	22	4	0.09	0.09
7	/	26	4	0.099	0.099
8	/	27	4	0.13	0.13
9	/	24	4	0.039	0.039

25 Field Data information in 1990

			Chlorophyll-*a* (mg/l)	
Month	Day	No. of Data	Epilimnion	Hypolimnion
4	16	3	0.023	0.023
5	1	4	0.064	0.064
5	15	4	0.03	0.03
5	19	4	0.011	0.011
6	13	4	0.009	0.009
6	26	4	0.021	0.021
7	10	4	0.098	0.098
7	24	4	0.079	0.079
8	7	4	0.142	0.142
9	5	4	0101	0.101
9	19	4	0.194	0.194
10	2	4	0.082	0.082



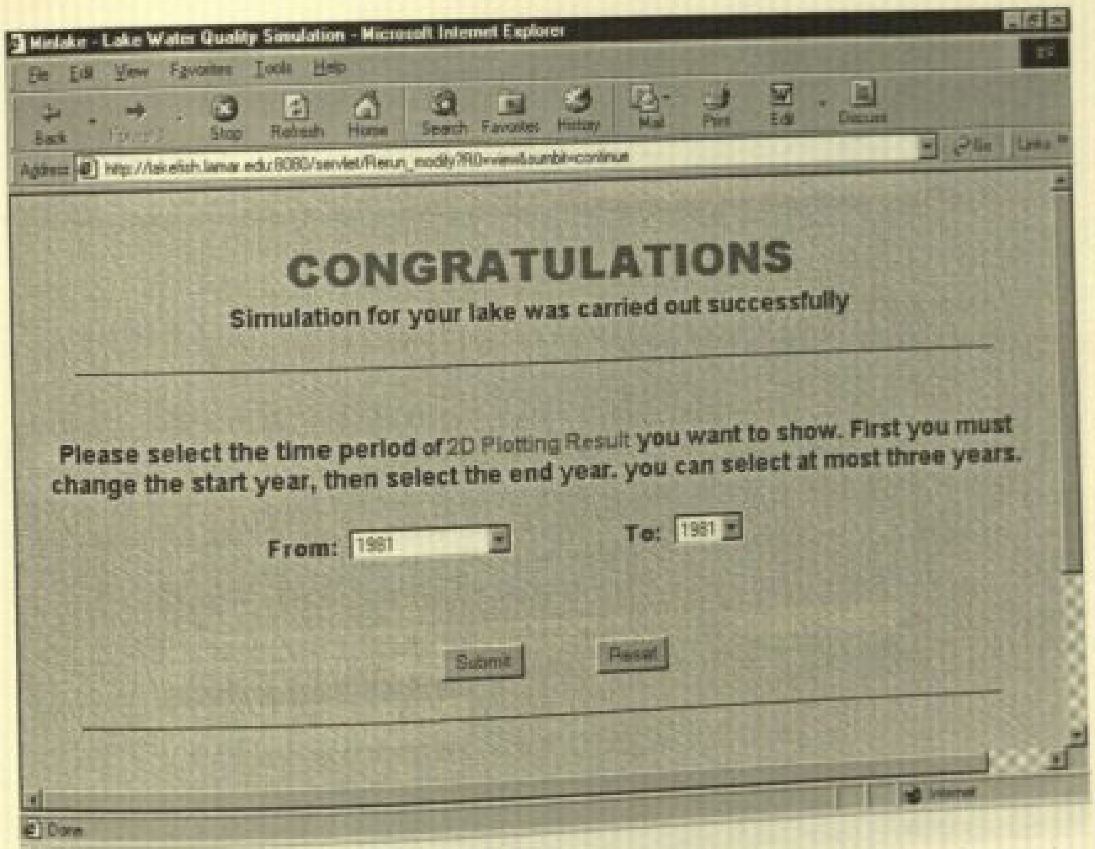
8 / 3		4	0.069	0.069
8 / 19		5	0.024	0.024
9 / 1		4	0.047	0.047
9 / 14		4	0.02	0.02
9 / 29		4	0.072	0.072
10 / 12		4	0.003	0.003


mainpage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address
Go Links
Home
Main
References
Lake information
Lake geometry
ModelRun
Sumbit
continue
Applet started
Internet

Simulation Options

Start Simulation:	[✓ yes]
Start Simulation (Future):	yes
Add Field Data:	yes
Lake Parameter Files	download

Figure 36. Simulation run options page; This HTML page is displayed once the user has supplied all the input data. It gives an option of running a simulation model or adding field data or changing lake parameter files.


Minlake - Lake Water Quality Simulation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu:8080/servlet/Rerun_modify?RD=view&sumbit=continue
Go Links
Submit
Reset
Done
Internet

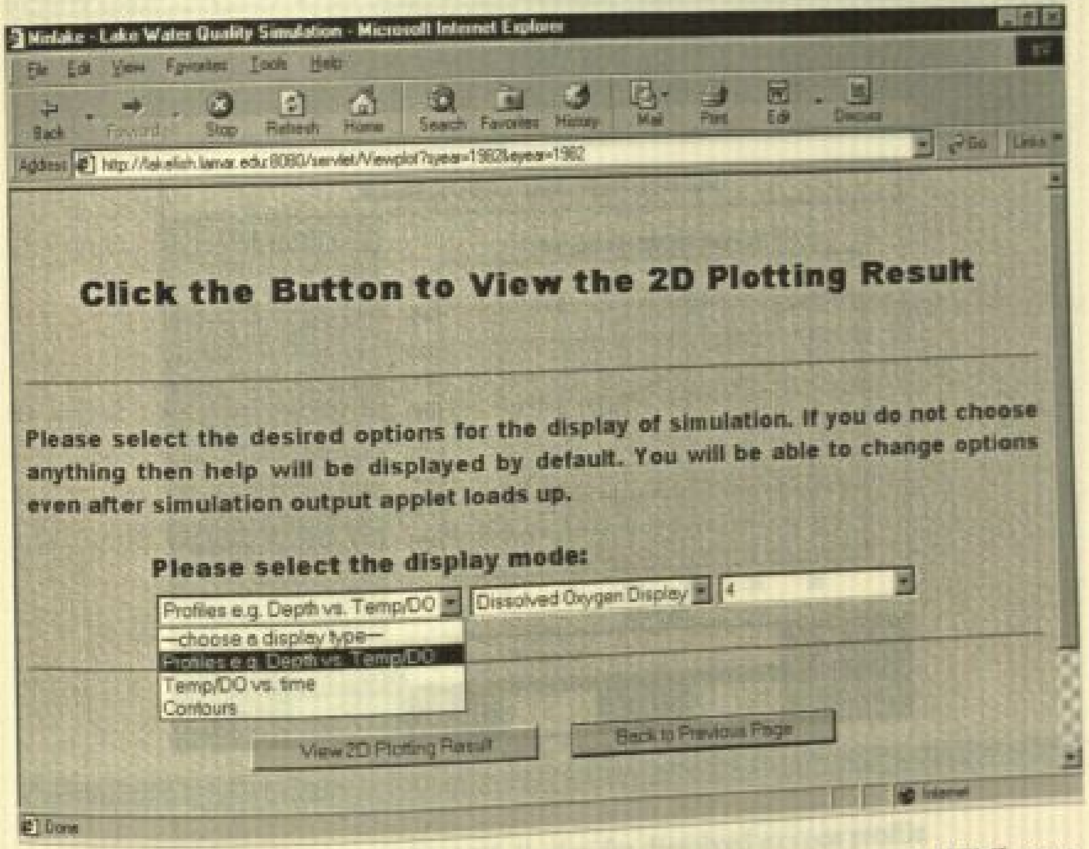
CONGRATULATIONS

Simulation for your lake was carried out successfully

Please select the time period of 2D Plotting Result you want to show. First you must change the start year, then select the end year. you can select at most three years.

From: 1981 To: 1981

Figure 37. Simulation display year selection page; This HTML page allows the user to select the year(s) for which a graphical display is required.


Minlake - Lake Water Quality Simulation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://lakefish.lamar.edu:8080/servlet/Viewplot?syear=1982&eyear=1982
Go Links

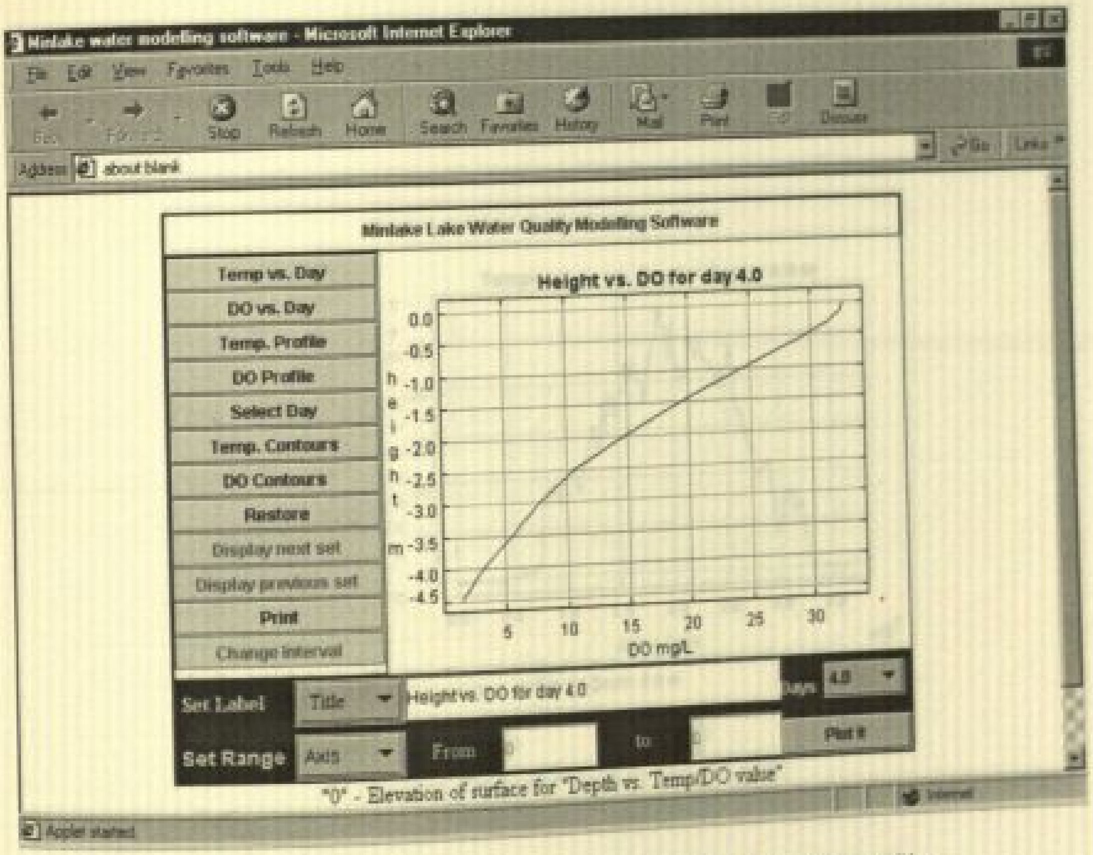
Click the Button to View the 2D Plotting Result

Please select the desired options for the display of simulation. If you do not choose anything then help will be displayed by default. You will be able to change options even after simulation output applet loads up.

Please select the display mode:

Profiles e.g. Depth vs. Temp/DO
Dissolved Oxygen Display
4
--choose a display type--
Profiles e.g. Depth vs. Temp/DO
Temp/DO vs. time
Contours
View 2D Plotting Result
Back to Previous Page
Done
Internet

Figure 38. Minlake Simulation Servlet Page;This HTML page is a dynamic HTML page generated by MinlakeSimulation servlet. And it is discussed in detail under chapter 3.



Minlake water modelling software - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address about:blank
Go Links
Minlake Lake Water Quality Modelling Software
Temp vs. Day
DO vs. Day
Temp. Profile
DO Profile
Select Day
Temp. Contours
DO Contours
Restore
Display next set
Display previous set
Print
Change interval
Height vs. DO for day 4.0
height m
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-4.5
5
10
15
20
25
30
DO mg/L
Set Label
Title
Height vs. DO for day 4.0
Days
4.0
Plot it
Set Range
Axis
From
to
"0" - Elevation of surface for "Depth vs. Temp/DO value"
Applet started
Internet

Figure 39. ML2DPLOT1 display for dissolved oxygen profile

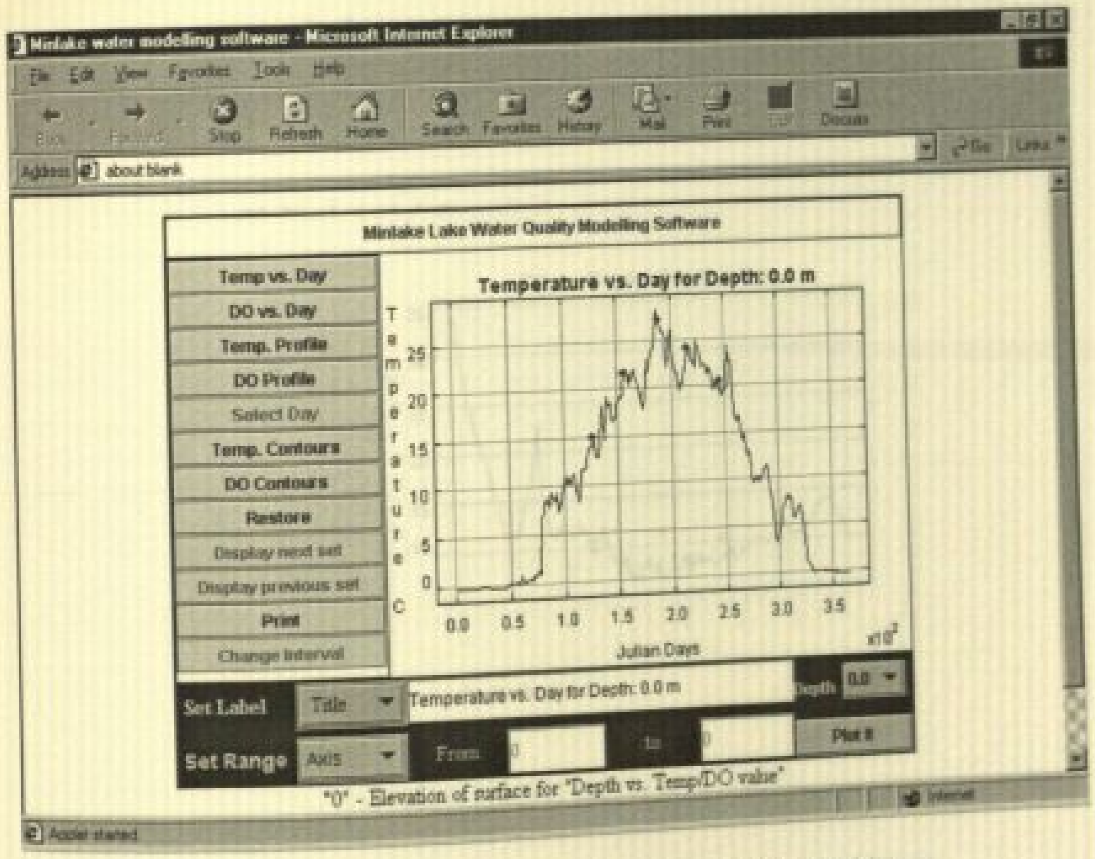

Minlake water modelling software - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Discuss
Address about:blank
Minlake Lake Water Quality Modelling Software
Temp vs. Day
DO vs. Day
Temp. Profile
DO Profile
Select Day
Temp. Contours
DO Contours
Restore
Display next set
Display previous set
Print
Change Interval
Temperature vs. Day for Depth: 0.0 m
Temperature C
Julian Days
x10²
Set Label
Title
Temperature vs. Day for Depth: 0.0 m
Depth
0.0
Set Range
Axis
From
to
Plot it
"0" - Elevation of surface for "Depth vs. Temp/DO value"
Applet started
Internet

Figure 40. ML2DPLOT1 display for temperature history

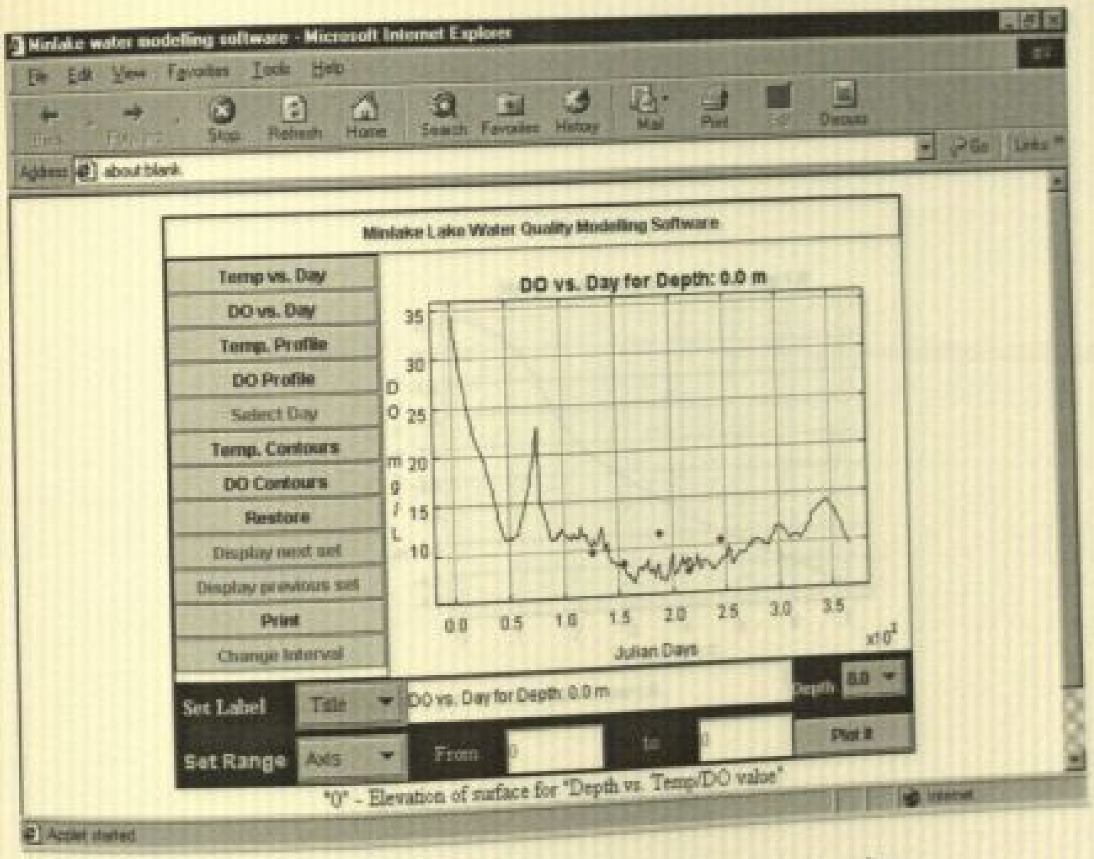
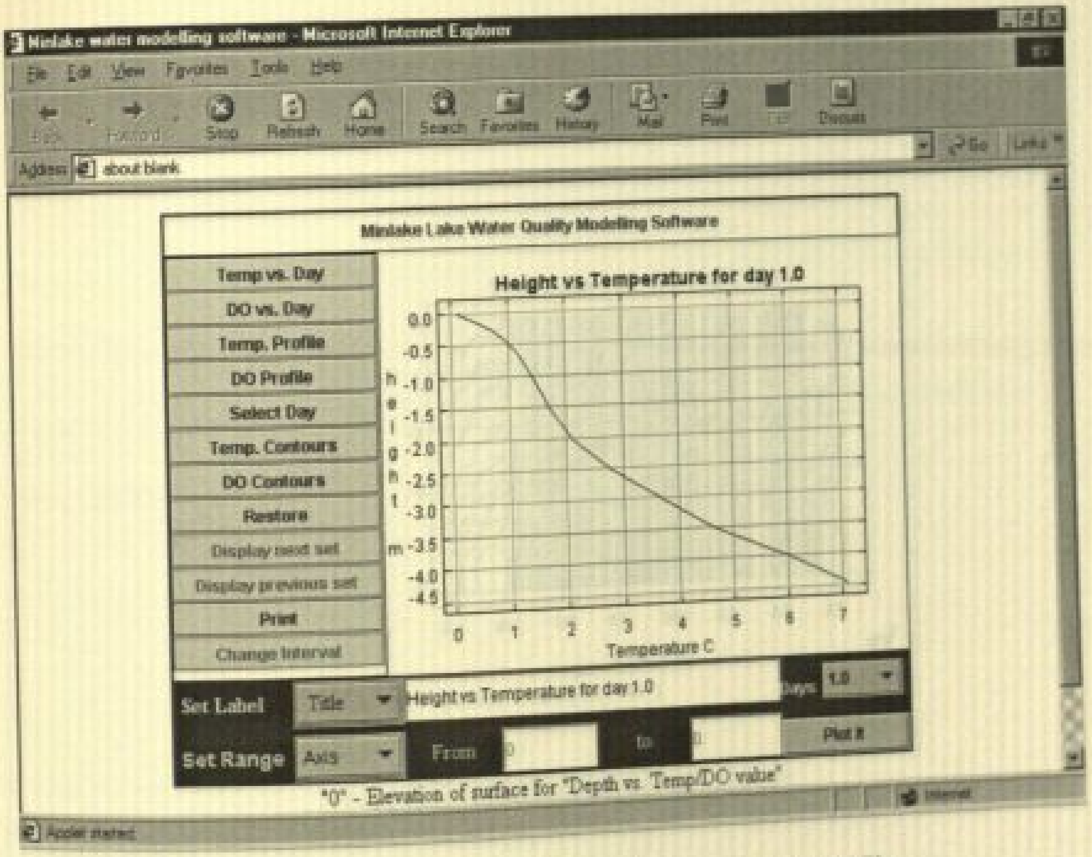

Minlake water modelling software - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Discuss
Address about:blank
Go Links
Minlake Lake Water Quality Modelling Software
Temp vs. Day
DO vs. Day
Temp. Profile
DO Profile
Select Day
Temp. Contours
DO Contours
Restore
Display next set
Display previous set
Print
Change Interval
DO vs. Day for Depth: 0.0 m
DO mg/L
35
30
25
20
15
10
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
x10^2
Julian Days
Set Label
Title
DO vs. Day for Depth: 0.0 m
Depth
0.0
Set Range
Axis
From
to
Plot it
"0" - Elevation of surface for "Depth vs. Temp/DO value"
Applet started
Internet

Figure 41. ML2DPLOT1 display for dissolved oxygen history

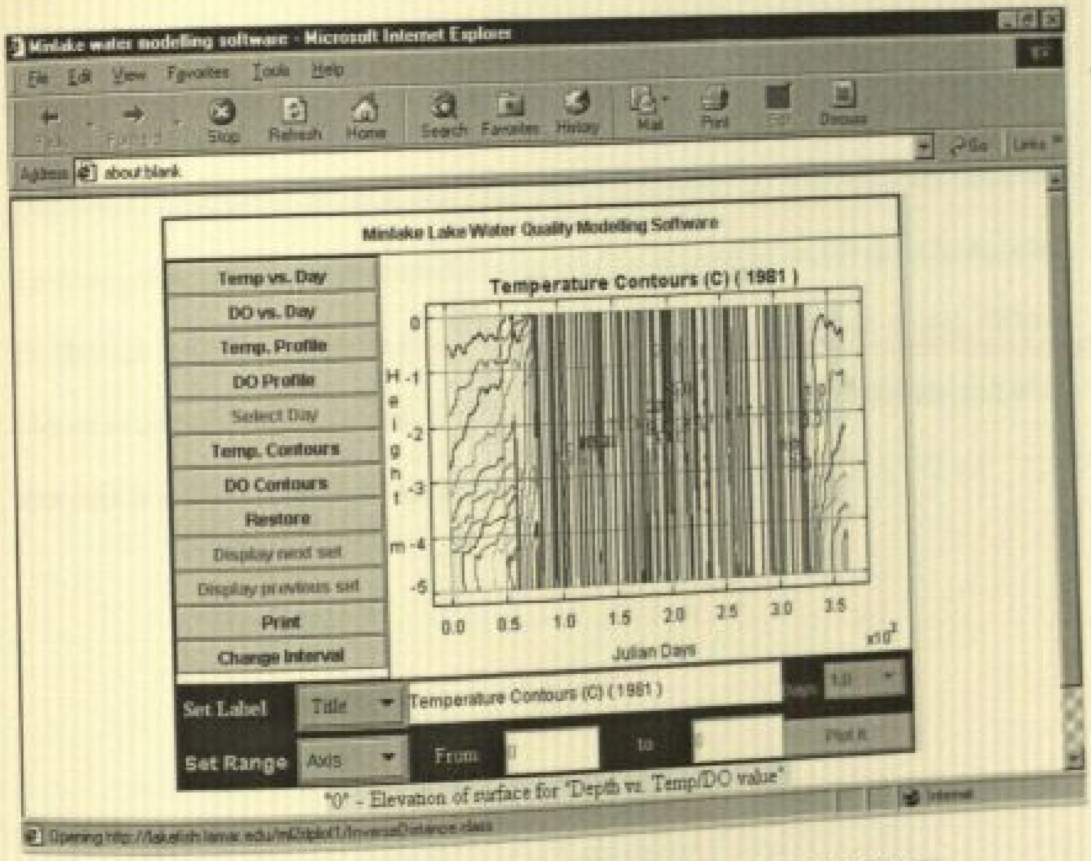


Set Label: Title — Height vs Temperature for day 1.0 — Days: 1.0

Set Range: Axis — From 0 to 0

"0" - Elevation of surface for "Depth vs. Temp/DO value"

Figure 42. ML2DPLOT1 display for temperature profile

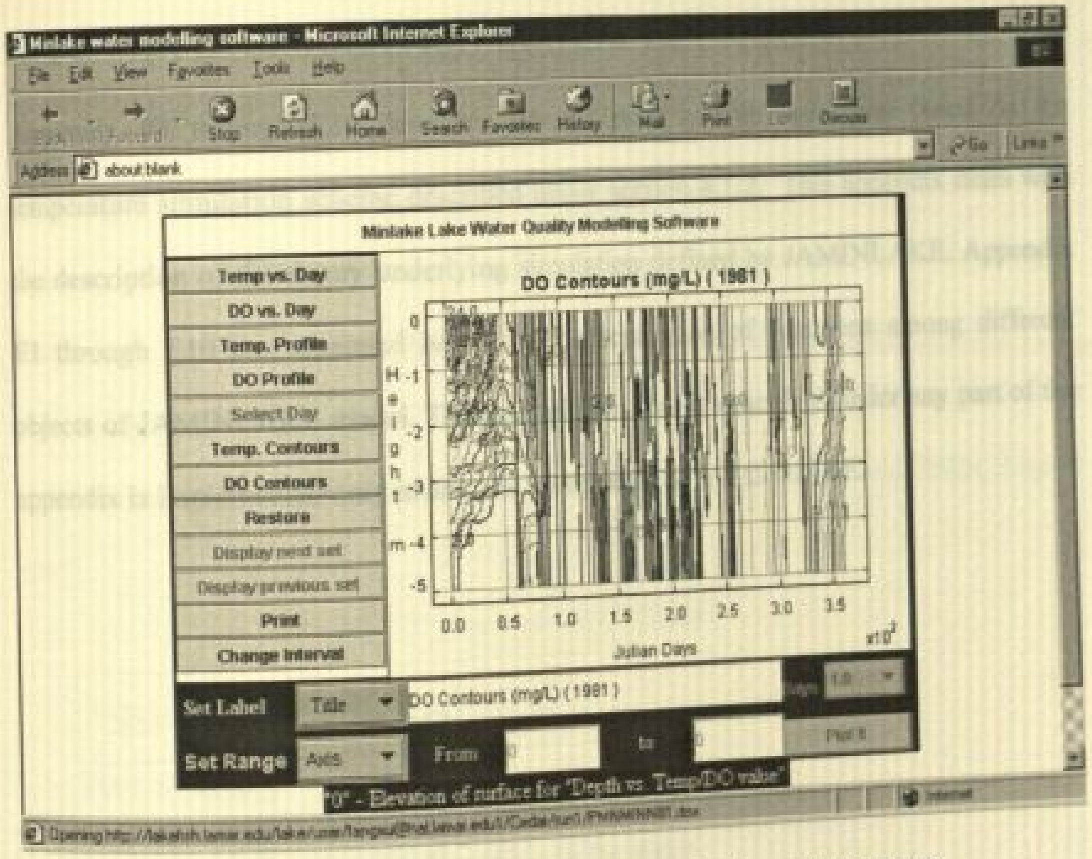

Minlake water modelling software - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Stop Refresh Home Search Favorites History Mail Print Discuss
Address about:blank
Minlake Lake Water Quality Modelling Software
Temp vs. Day
DO vs. Day
Temp. Profile
DO Profile
Select Day
Temp. Contours
DO Contours
Restore
Display next set
Display previous set
Print
Change Interval
Temperature Contours (C) (1981)
Height m
Julian Days
x10^2
Set Label
Title
Set Range
Axis
From
to
Plot it
Internet

Temperature Contours (C) (1981)

"0" - Elevation of surface for "Depth vs. Temp/DO value"

Opening http://lakefish.lamar.edu/ml2dplot1/InverseDistance.class

Figure 43. ML2DPLOT1 display for temperature contours



DO Contours (mg/L) (1981)

"0" - Elevation of surface for "Depth vs. Temp/DO value"

Opening http://lakeshh.lamar.edu/lake/.../Cedar/run1/PMINNN81.dox

Figure 44. ML2DPLOT1 display for dissolved oxygen contours

APPENDIX E –SIMULATION CONCEPTS

Appendix E1 through E10 describes several parts of the simulation based on the temperature simulation scheme described under section 4.1.2. This appendix deals with the description of the theory underlying simulation defined by JAMINLAKE. Appendix El through E10 are separated based on the separation of functions among different objects of JAMINLAKE model. Thus, simulation theory described under any part of the appendix is implemented only in any one component of JAMINLAKE.

APPENDIX E1- HEAT FLUX AT THE SURFACE OF A LAKE

Incoming heat from solar and long-wave radiation and the outflow of heat through convection, evaporation and back radiation plays a major role in establishing a temperature profile along the depth of the lake. The net increase in heat results in an increase in water temperature. The heat balance equation is given by the following equation [Ford and Stefan, 1980; Thomann and Mueller, 1987; Riley and Stefan, August 1987].

$$\Delta H = H_s + H_a - H_c - H_e - H_b , \tag{5}$$

where

ΔH = Net Heat Flux to the water,

H_s = Net Solar radiation,

H_a = Long wave radiation from atmosphere,

H_c = Convective Heat Flux,

H_e = Evaporative Heat Flux,

H_b = Back Radiation.

If an ice cover exists there is no heat flux from/to the water. Hence the only heat flux that affects in case of an ice-snow cover is the flux from sediment.

Figure 45 shows a schematic diagram for the heat exchange between different layers.

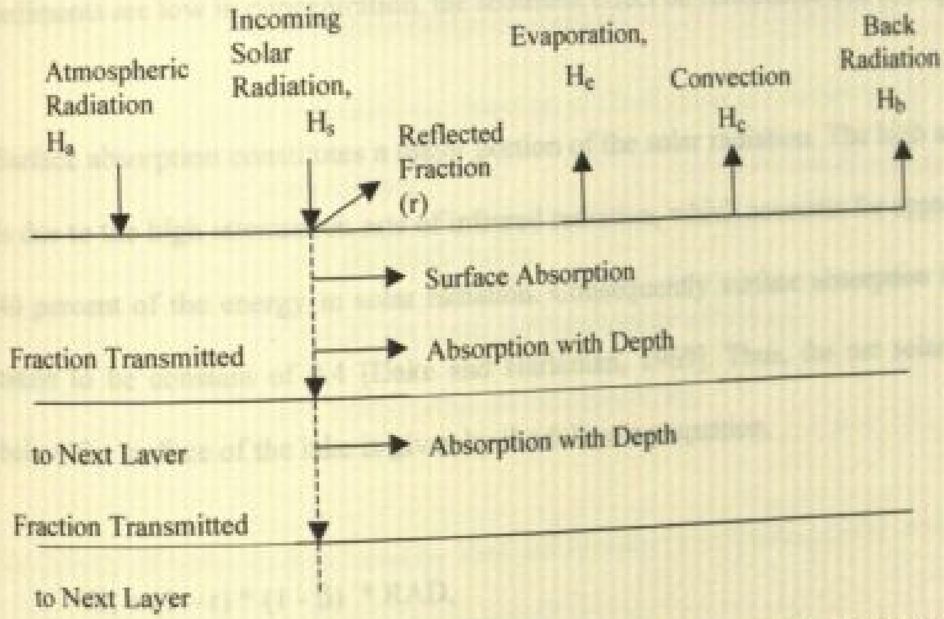


Figure 45. Schematic diagram of heat transfer (flux) between the atmosphere and the lake.

SOLAR RADIATION:

Based on the amount of radiation and concentration of suspended solids, a part of the solar radiation is reflected back and the rest is transmitted to the water. The reflected portion is given by the following equation [Riley and Stefan, August 1987].

$$r = 0.087 - 6.76 \times 10^{-5} * RAD + 0.11\{1 - e^{(-0.01*SS)}\}, \tag{6}$$

where

r = reflected fraction,

RAD = total daily solar radiation (cal/cm^2-day),

SS = suspended sediment concentration in first layer.

This equation was developed for highly turbid southern lakes, thus if the suspended sediments are low in concentration, the sediment effect on reflectance can be neglected.

Surface absorption constitutes a major portion of the solar radiation. The high absorption is due to the high attenuation rate of infrared radiation, which accounts for approximately 40 percent of the energy in solar radiation. Consequently surface absorption fraction is taken to be constant of 0.4 [Dake and Harleman, 1969]. Thus, the net solar radiation below the surface of the lake is given by the following equation.

$$H_s = (1 - r) * (1 - \beta) * RAD, \tag{7}$$

where

H_s = net solar radiation (kcal/m^2-d),

r = reflected fraction,

β = surface absorption fraction = 0.4,

RAD = incoming solar radiation (kcal/m^2-d).

<u>LONGWAVE RADIATION</u>:

Long-wave radiation is energy emitted by all objects. The heat budget must account for both atmospheric long wave radiation received by the lake and long wave radiation emitted by the lake. It is given by [Riley and Stefan, August 1987] the following equation.

$$H_A = \sigma \epsilon T^4, \tag{8}$$

where

H_A = Atmospheric long-wave radiation (kcal/m^2-d),

σ = Stefan-Boltzman constant = 1.71* 10^{-6} kcal/m^2-K^4,

ε = Emissivity, function of cloud cover for emissivity of

atmosphere,

= 1 for black boxes,

= 0.97 for water (approximately constant),

T = Temperature of the object (°K).

Emissivity of water as developed by Idso and Jackson [1969] with a cloud cover correction is given by the following equation.

$$\varepsilon a = \{1\text{-}0.261\exp(-7.7710^{-4}T_a^{\,2})\}(1+K^*C_c^{\,2}), \qquad (9)$$

where

Ta = atmospheric radiation (°C),

C_c = cloud cover ratio,

K = cloud height coefficient.

The coefficient K varies between 0.04 and 0.25. A TVA [1968] study recommended using a value of 0.17 and that is being used in this model.

<u>EVAPORATIVE HEAT TRANSFER:</u>

Evaporative heat loss is due to the change in state of water from liquid to vapor. The evaporative heat loss can be calculated directly if the mass of water evaporated is known.

$$H_e = k * E, \qquad (10)$$

where

H_e = Evaporative heat loss (kcal/m^2-d),

k = latent heat of vaporization = 5900 kcal/cm-m^2-day.

However in most cases, it is extremely difficult to know the mass of water evaporated. In such cases a standard empirical formula for evaporative heat loss can be used:

$$H_e = 20.63\left(2.7\times\Delta T_{vi}^{\frac{1}{3}} + 63.9\times U_a\right)\left(e_s - e_a\right). \qquad (11)$$

where

e_s = saturated vapor pressure at the water surface,

$$e_s = 6.1078*\exp[17.27*Ts/(Tsk-35.86)]. \qquad (12)$$

e_a = vapor pressure of the air,

U_a = the average daily wind speed in m/s,

ΔT_{vi} = Difference between the virtual temperature at the water surface and in the ambient air (°C).

$$e_a = 6.035*10(7.45T_d / (235+T_d)), \qquad (13)$$

where

T_d = Dew point temperature (°C),

CONVECTIVE HEAT TRANSFER:

This heat transfer occurs due to the difference in temperature between the water and air. Convective heat loss is given by the following equation,

$$H_c = 20.63 \times \left(\frac{0.61}{1000} \times P_a\right)\left(2.7 \times \Delta T_{vi}^{\frac{1}{3}} + 63.9 \times U_a\right)(T_s - T_a), \tag{14}$$

where

Ts = water surface temperature (°C),

Ta = air temperature (°C),

ΔT_{vi} = Difference between the virtual temperature at the water surface and in the ambient air (°C),

U_a = Wind speed in m/sec,

H_c = convective heat flux (kcal/m^2-d).

BACK RADIATION

Back radiation is calculated with the same formula as for Equation 8. However, the emissivity is fixed at 0.97 and the temperature is the surface water temperature. Thus, the following equation calculates back radiation.

$$H_{BR} = 0.97 \times \sigma \times T_s^4, \tag{15}$$

where all the constants are as explained for Equation 8

APPENDIX E2 - HEAT FLUX FROM ONE LAYER TO THE NEXT

There are two distinct lake heat sources/sinks used in this model: 1) atmosphere 2) sediments. Several heat fluxes are associated with atmosphere. It includes long wave radiation, short wave radiation, evaporation losses, back radiation and convective losses. However, these fluxes are only associated with the lake surface. The rest of the layers transmit a part of this flux. So the **Lake** is responsible for distribution of these heat source sink term between its layers. The following discussion clarifies the method that a lake employs in order to transmit heat flux among its layers.

The scheme for heat flux transmission to the lower layers from the surface is presented below.

There are several heat sources and sinks at the surface of the lake. These sources/sinks are calculated in the class **Coefficients**. If the net solar radiation reaching the first layer is H_{sn} then the following equation describes the net absorption in the first layer [Ford and Stefan, 1980; Thomann and Mueller, 1987; Riley and Stefan, August 1987].

$$Q_1 = (1-r)(1-\beta)*H_{sn}*[A_1-A_2e^{(-k*z1)}] + \beta H_{sn}A_1, \tag{16}$$

*Subscript 1 refers to Layer 1 and subscript 2 refers to Layer 2

where

r = Reflected fraction,

β = Surface absorption fraction = 0.4,

A_1 = Surface are of layer 1,

A_2 = Surface area of layer 2,

k = Extinction coefficients.

The heat flux is transmitted from the top layer to the layer below it and so on. The general formula for heat flux transmission is given below.

$$Hs_{(i)} = Hs_{(i-1)} * e^{-kz}, \qquad (17)$$

where

$Hs_{(i)}$ = Heat flux at the bottom of the layer (kcal/m^2-day),

$Hs_{(i-1)}$ = Heat flux at the top of the layer (kcal/m^2-day),

k = extinction coefficient (m^{-1}),

z = thickness of the layer (m).

The following equation calculates the solar radiation reaching the top of the second layer.

$$Hs_{(2)} = (1-r)(1-\beta)*H_{sn}*e^{(-k*z1)}, \qquad (18)$$

The following equation calculates the absorbed solar radiation in the subsequent layer

$$Q_{(i)} = Hs_{(i)}[A_{(i)}-A_{(i+1)}*e^{(-k*z(i))}], \qquad (19)$$

where the subscript i refers to the ith layer and i+1 refers to the (i+1)st layer.

The extinction coefficient is unique to each layer and is defined by a linear function of three terms; the extinction coefficient of water, extinction coefficient of the suspended sediment concentration in the layer, and the extinction coefficient of the chlorophyll-a concentration in the layer. The total extinction coefficient is given by the following equation [Riley and Stefan, 1987].

$$k = kw + 0.043*SS + k_2*Chla, \tag{20}$$

where

k = Total attenuation coefficient (m^{-1}),

k_w = Extinction coefficient of water (m^{-1}). It is not the extinction coefficient of pure water, but includes the combined effect of all dissolved substances especially color caused by dissolved organics,

k_2 = Extinction coefficient due to chlorophyll $\frac{1}{m - mg_{CHLA}}$, can range from 0.1 to 0.25. However from Bannister (1974a) this value is taken as 0.016,

SS = suspended solids concentration $\frac{mg}{l}$,

Chla = chlorophyll-a concentration $\frac{mg}{l}$.

APPENDIX E3 - CALCULATION OF TEMPERATURE PROFILE FROM THE HEAT FLUX DISTRIBUTIONS:

A method called calculateTemperatureFromFlux() is used to carry out this task. A set of simultaneous equations is constructed based on finite difference form of advection diffusion equation applied to the temperature state variable. This equation is formed for each layer. The Coefficients class sets the coefficients of this equation. The finite difference form of the equation is presented below.

$$-\frac{\Delta t}{A_i \Delta z_i}\left(\frac{A_{i-\frac{1}{2}} K_{i-\frac{1}{2}}}{\frac{1}{2}\left[\Delta z_i + \Delta z_{i-1}\right]}\right) C_{i-1,j+1} + \left\{1 + \frac{\Delta t}{A_i \Delta z_i}\left(\frac{A_{i-\frac{1}{2}} K_{i-\frac{1}{2}}}{\frac{1}{2}\left[\Delta z_i + \Delta z_{i-1}\right]} + \frac{A_{i+\frac{1}{2}} K_{i+\frac{1}{2}}}{\frac{1}{2}\left[\Delta z_i + \Delta z_{i+1}\right]}\right) \pm S_1 \Delta t\right\} C_{i,j+1} \tag{21}$$

$$-\frac{\Delta t}{A_i \Delta z_i}\left(\frac{A_{i+\frac{1}{2}} K_{i+\frac{1}{2}}}{\frac{1}{2}\left[\Delta z_i + \Delta z_{i+1}\right]}\right) C_{i+1,j+1} = C_{i,j} \mp S_2 \Delta t$$

where

A - stands for the cross sectional area,

z – stands for depth of the layer,

Δz – stands for thickness of the layer,

C – state variable in consideration (for our case temperature),

K – vertical diffusion coefficient between the two layers inconsideration.

Subscript i – denotes the ith layer. Properties between the two layers are defined by a fractional ith value. For example, a value of i equal to 1 + ½ means that the property at the interface of 1st and 2nd layer. Subscript j – denotes the index for the time. S_1, S_2 are the heat source/sink terms.

A tri-diagonal matrix is set up using these simultaneous equations and a Guass Elimination technique is used to solve these simultaneous equations.

APPENDIX E4- EFFECTS OF CONVECTIVE MIXING ON TEMPERATURE PROFILE

A method called forceConvectiveMixing() is responsible for this task. A solution of advection diffusion equation constructs a temperature profile in the lake. However, if any layer has a negative temperature then the profile has to be corrected. This is done by equating temperature of all the layers having negative temperatures to zero. The difference between the negative temperature and a zero corresponds to the heat flux removed from this water body up to the mixed layer. These are simply volume average calculations.

Natural and forced convective mixing causes a change in temperature profile. It also affects ice formation and thus net solar absorption. The following steps can simulate these effects [Fang and Stefan, 1996]:

a. First determine the number of negative layers due to surface cooling. Let the number of layers having negative temperature be NP. Layer NP is at a depth where water temperature changes from negative to positive.
b. Set all negative temperatures to zero and determine the change in heat content Q_{NEG}, associated with this water temperature adjustment.

$$Q_{NEG} = \sum_{i=1}^{NP} \rho c_p V_{(i)} \left[0 - T_{(i)} \right], \tag{22}$$

where

ρ is the density of the water,

c_p is the specific heat of water,

$V_{(i)}$ is the volume of the ith layer,

$T_{(i)}$ is the temperature of the ith layer.

This procedure is performed for each layer. If the density of a layer is higher than the density of the layer below it, then the state variables of these layers are volumetrically averaged.

APPENDIX E5- SURFACE COOLING OF THE LAKE:

Based on the boundary condition of 0°C at the surface, a calculation is carried out for the ice formation conditions over the surface of the lake in **simulateBeforeIceFormation()** method. Based on the natural convective flux (Q_{NEG}) calculated with the **forceConvectiveMixing()** method, thickness of ice is calculated. This is followed by a recalculation of mixed layer based on forced mixing calculations similar to calculations described in **calcLakeMixing()** method in class Wind.

The heat content change Q_{NEG} is restored to the mixed layer (with volume V_M) to get a new mixed layer temperature T_N.

$$T_N = T_M - \left[\frac{Q_{NEG}}{\rho\, c_p V_M}\right], \tag{23}$$

where

ρ is the density of water,

c_p is the specific heat of water,

V_M is the volume of the mixed layer,

Q_{NEG} is the heat content change,

T_N – New Temperature of the mixed layer,

T_M – Old Temperature of the mixed layer.

Re-compute the mixed layer depth and temperature using T_N instead of T_M because of the change of the density difference across the interface. The final mixed layer depth and temperature are Z_{MF} and T_{MF}. Typically, $z(NP) \leq z_{MF} \leq z_M$. T_{MF} can be between zero and T_N or less than zero.

This natural convective mixing and forced convective mixing also affects the thickness of ice. There are three cases where ice formation is indicated:

a. The mixed layer temperature T_M, after wind mixing is still zero. The wind is too weak to mix the water from the surface to the mixed layer depth. In this case the surface layer temperature will be less than calculated and the surface water will be super cooled indicating ice formation. The thickness of ice can be determined from the following equation.

$$H_{ICE} = \frac{Q_{NEG}}{[\rho A_s \lambda]}, \tag{24}$$

where

ρ is the density of water,

A_s is the surface area of the lake,

H_{ICE} is the thickness of ice,

λ is the latent heat of water (335 kJ/kg),

Q_{NEG} is the heat content change.

b. If the adjusted mixed layer temperature is below 0°C. The ice thickness is determined by the equation given below.

$$H_{ICE} = \frac{\rho c_p V_M [0 - T_N]}{[\rho A_s \lambda]}, \tag{25}$$

where

ρ is the density of water,

A_s is the surface area of the lake,

H_{ICE} is the thickness of ice,

λ is the latent heat of water (335 kJ/kg),

Q_{NEG} is the heat content change,

c_p is the specific heat of water,

V_M is the volume of the mixed layer.

The final adjusted mixed layer temperature T_{MF} is less than zero. Ice thickness can be determined by the above two equations using T_{MF} instead of T_N.

c. The final adjusted mixed layer temperature T_{MF} is less than zero. Ice thickness can be determined by the above two equations using T_{MF} instead of T_N.

APPENDIX E6 - INITIAL SEDIMENT TEMPERATURE PROFILE

The initial Sediment temperature profile is influenced by the lake water temperature and ground water temperature. Because lake water temperature varies with depth, sediment below each lake layer exhibits a different temperature profile. In order to incorporate this behavior, a temperature profile for sediment is associated with each lake layer. Thus, the lake sediment interface temperature for each depth is different. However the temperature at 10m below the lake sediment is found constant [Baker and Ruschy, 1993; Fang and Stefan, 1998].

It has been shown that ground water temperatures can be effectively predicted using yearly mean air temperature [Todd, 1980; Fang and Stefan, 1998]. An extensive database for Meteorological Data has been created in this project. A unique location is obtained by identifying state and station location closest to the lake being modeled. Using this mean air temperature and mean January air temperature can be obtained.

If the January mean air temperature is less than -3°C, then ground water temperature is 3°C higher than the yearly mean air temperature. If this condition is not satisfied then the groundwater temperature is 2°C higher than the yearly mean air temperature.

A clear relationship has been established between lake geometry ratio ($As^{0.25}/Hmax$), groundwater temperature and sediment temperature at 10m below lake bottom.

If the lake geometry ratio is less than 4, it refers to a stratified dimictic lake. Sediment temperature 10m below the lake bottom is calculated for this condition by the following equation.

$$T_{S10(H_{MAX})} = \frac{T_G - 2}{2} + \frac{T_G + 6}{3}\log\left(\frac{A_S^{0.25}}{H_{max}}\right), \qquad (26)$$

where

T_G refers to the ground water temperature.

If the lake geometry ratio is greater than 5 then the sediment temperature below 10m is predicted equal to the groundwater temperature.

Sediment temperature at the lake sediment interface is equal to the lake water temperature. This condition is in line with continuity at the interface. Hence, sediment temperature at the lake sediment interface is different for each different layer of the lake.

For stratified lakes (lakes having lake geometry ratio <4) the following method can be employed for calculating sediment temperature profiles:

A direct relationship of a new depth scale Z_{norm} and a normalized sediment temperature T_{norm} has been established earlier [Fang and Stefan, 1998].

where,

$$z_{norm} = \frac{Z}{[Z_S^{0.35} * (A_S H_{MAX})^{0.125}]} \text{ and}$$

$$T_{norm}^{\cdot} = \frac{[T_{S10}(0) - T_{S10}(z)]}{T_{S10}(0) - T_{S10}(H_{MAX})}$$

Using the values of this relationship normalized sediment temperature is obtained based on the "new depth scale" value set for the specific condition. Using this equation, the boundary layer condition for other sediment layers corresponding to interim lake depths, is established.

These boundary conditions along with an exponential temperature growth, establishes a temperature profile in the sediment at different lake depths.

APPENDIX E7 – DETERMINING SNOW THICKNESS

Snow thickness is determined by snow accumulation, followed by compaction and melting of snow by surface heat input (convection, rainfall, solar radiation) and melting within the snow layer due to internal absorption of short wave radiation [Fang and Stefan, December 1994].

The snow cover forming on the ice surface of a lake is likely to be thinner than direct accumulations of a snowfall; therefore a snow compacting factor (0.2 to 0.4) is used before adding snowfall to the existing snow depth [Adams, 1982; Gu and Stefan, 1990]. The change in snow thickness for 1 day is given by the following mass balance equation.

$$\frac{dz_{sw}}{dt} = C_{sm} P_{sm} - z_s - z_c - z_e - z_r, \tag{27}$$

where

z_{sw} = Snow thickness m,

P_{sw} = Snow fall from given weather date (m/day),

z_s = Snow depth reduction rate due to melting as a result of solar radiation (m/day),

z_c = Snow depth reduction rate by convective heat transfer(m/day),

z_e = Snow depth reduction rate by evaporation (m/day),

z_r = Snow depth reduction rate by rainfall (m/day).

- Determining snow melt on the surface due to solar radiation [Gu and Stefan, 1990]:

$$z_s = \frac{H_{sw}}{\rho_{sw} \lambda_{sw}}, \tag{28}$$

$$H_{sw} = H_s \{\beta_{sw}(1-\alpha_{sw}) + (1-\alpha_{sw})(1-\beta_{sw})[1-\exp(-\mu_{sw} z_{sw})]\}, \tag{29}$$

where

H_{sw} = solar radiation flux (kcal/m^2/day),

H_s = net solar radiation on the snow surface,

β_{sw}=surface absorption coefficient for snow,

α_{sw}=surface reflectivity for snow,

μ_{sw}=attenuation coefficient in the snow,

ρ_{sw} = density of snow,

λ_{sw}= Latent heat of snow.

- Snow depth reduction rate due to convective heat transfer [Light, 1941]:

$$z_c = 0.000376 * 10^{-0.0000156z}{}_w U_a T_a, \qquad \text{for } T_a > 0°C \tag{30}$$

where

U_a = wind speed (m/sec),

T_a = Air temperature (°F),

z_w = weather station-elevation (ft),

z_c = Snow depth reduction rate (m/day) due to convective heat transport rate.

- Snow depth reduction rate due to evaporation (latent heat transport):

$$Ze = 0.00118U_a(e_s-e_a), \qquad \text{for } T_s > 0°C \tag{31}$$

where

U_a = wind speed (m/sec),

e_s =vapor pressure above snow surface (mbars),

e_a = vapor pressure of the air (mbars).

- Snow melting due to rainfall:

Equation for this condition is developed from an energy balance, i.e. heat energy released by rain is equal to the heat energy absorbed by the snow.

$$z_{ir} = \frac{P_r \rho_w c_p T_a}{\lambda_{sw} \rho_{sw}}, \qquad \text{for } T_a > 0°C \qquad (32)$$

where

z_r = snow melting (m/day) due to rainfall,

P_r =Precipitation (m/day),

ρ_w=water density (kg/m^3),

c_p =specific heat of water (kcal/kg°C),

T_a = Air temperature °F,

λ_{sw}=Latent heat of snow(kcal/kg),

ρ_{sw}=Density of snow (kg/m^3).

There is a possibility of snow cover only under the following two circumstances; 1) If either there is a snowfall on the present simulation day or 2) a snow cover exists from the previous day. The present day's atmospheric condition can melt the snow and snow thickness can go to zero. Hence, for either of the two earlier stated cases, thickness check has to be performed.

Total snowmelt is calculated by a sum of all individual snowmelt. Thickness of snow is the difference between the thickness of the snow cover and the snowmelt of the present simulation day.

APPENDIX E8 – ICE FORMATION ON THE SURFACE OF THE LAKE

Ice formation for each simulation day is governed by heat budget and wind shear [Fang, Stefan, December 1994].

HEAT BUDGET FOR ICE FORMATION

Prediction of ice thickness is essential to the winter temperature modeling. Ice usually grows up at the ice-water interface, but ice can decay at the snow-ice interface, ice-water interface, and within the ice layer. The change in the ice thickness over one day is given by the following mass balance equation.

$$\frac{dz_i}{dt} = z_{ic} - z_{is} - z_{ir}, \tag{33}$$

where

z_{ic} = ice growth/decay rate (m/day) due to conduction/convection,

z_{is} = ice growth/decay rate (m/day) due to solar radiation,

z_{ir} = ice growth/decay rate (m/day) due to precipitation.

The following is a mass balance equation of the ice thickness due to conduction and convection [Ashton, 1986; Gu, Stefan, 1990].

$$z_{ic} = \frac{1}{\rho_i \lambda_i}\left[\frac{T_{iw} - T_a}{\frac{z_i}{k_i} + \frac{z_{sw}}{k_{sw}} + \frac{1}{h_{sa}}} - k_w\left(\frac{dT}{dz}\right)_{z=0}\right], \quad (34)$$

where

ρ_i = ice density (kg/m^3),

λ_i = latent heat of fusion of ice (kcal/kg),

T_{iw} = Temperature at ice-water interface,

k_i = Thermal conductivity of ice,

k_{sw} = Thermal conductivity of snow,

k_w = Thermal conductivity of water,

h_{sa} = bulk heat transfer coefficient at the snow/air interface

(kcal/day-°C-m^2).

$$h_{sa} = C_w U_a \text{ [Fertuck, et al., 1971]}, \quad (35)$$

where

C_w = empirical constant with a value ranging from 0.29 to 0.37,

U_a = wind speed in mph,

h_{sa} = bulk heat transfer coefficient at snow/air interface with a unit

of BT/hr-ft^2-°C.

In the Equation 34, thickness increases (ice growth) if air temperature T_a is less than zero; otherwise the ice thickness decreases (ice decay). Accurate prediction of ice thickness requires accurate estimation of thermal conduction coefficient of water k_w, water temperature gradient at the ice-water interface and C_w for the bulk heat transfer coefficient. Prediction of ice-thickness is sensitive to k_w and temperature gradients at the ice-water interface.

Internal melting in the ice layer results from internal absorption of solar radiation. This internal melting can be expressed by the following formula.

$$z_{ia} = \frac{H_{ia}}{\rho_i \lambda_i}, \tag{36}$$

where

z_{ia} = ice decay rate due to internal melting from internal absorption of solar radiation,

H_{ia} = Absorbed short-wave radiation.

$$H_{ia} = H_{sw}\,\{\beta_i + (1-\beta_i)(1-\alpha_i)[1-\exp(\mu_i z_i)]\}, \tag{37}$$

where

H_{sw} = solar radiation available at the top surface of the ice,

β_i = Surface absorption coefficient for ice,

α_i = Surface reflectivity for ice,

μ_i = Attenuation coefficient,

z_i = ice thickness.

Ice decay due to rainfall is calculated by performing energy balance. This condition exists when snow depth is zero.

$$z_{ir} = \frac{P_r \rho_w c_p T_a}{\lambda_i \rho_i}, \qquad \text{for } T_a > 0°C \qquad (38)$$

where

z_r = snow melting (m/day) due to rainfall,

P_r =Precipitation (m/day),

ρ_w=water density (kg/m^3),

c_p =specific heat of water (kcal/kg°C),

T_a = Air temperature °F,

λ_i=Latent heat of snow(kcal/kg),

ρ_i=Density of snow (kg/m^3).

ICE COVER DUE TO SNOWMELT:

Snowmelt induced by internal solar radiation absorption, convective heat transfer, evaporation and rainfall, can result in ice growth from the ice surface up when air temperature falls below freezing in the days following the snowmelt. This effect of ice refreezing of the snowmelt accumulating on the ice occurs frequently during late winter when air temperature falls below freezing. However, if the air temperature is continuously above freezing then the snowmelt will be lost by evaporation or runoff.

This situation can be modeled by the principle of conservation of mass. It is possible to transfer the depth of the melted snow to a thickness of ice growth based on densities of ice (typically 920 kg/m^3) and compacted snow (typically 200-500 kg/m^3).

ICE FORMATION DUE TO WATER COMING THROUGH CRACKS IN THE ICE COVER:

Another situation in which ice forms is due to the water coming through cracks in the ice cover. The free-water-level is approximately one tenth of the ice thickness below the top of the ice sheet. When the snow accumulates on the ice, lake water may be pushed through the cracks in the ice cover onto the ice surface to the form wetted snow. The critical snow depth determines if the free-water-level is above the top of the ice sheet. It is determined by the following equation.

$$\delta_s = \left\{ \frac{(\rho_w - \rho_i)\,\delta_i}{\rho_s} \right\}, \tag{39}$$

where

δ - Thickness of ice or snow,

ρ- Density of ice or snow or overflow layer,

subscript i, w, s, of – corresponds to ice, water, snow and overflow layer correspondingly.

An equilibrium equation for weight of the dry snow, wetted snow layer, and ice and the buoyancy force, gives us a formula for calculating the distance of the free-water-level above the top of the ice sheet. The following equation calculates free-water-level above the top of the ice sheet.

$$\delta_{of} = \left[\delta_s \rho_s + \delta_{cr}(\rho_{cr} - \rho_s) + \delta_i(\rho_i - \rho_w)\right] / (\rho_w + \rho_s - \rho_{of}), \tag{40}$$

where

δ_{of} - distance of the free-water-level above the top of the ice sheet,

δ_{cr} – distance of the capillary rise above the free water level,

Subscript cr represents the capiliary rise above the free-water-level.

Hence the total thickness of snow is the simulated snow thickness minus the distance of the free-water-level above the top of the ice sheet. If this level is more than the snow thickness then the snow thickness and ice thickness is zero.

The ice thickness is equal to the simulated ice thickness plus the distance of the free-water-level above the top of the ice sheet.

EFFECT OF WIND SHEAR ON ICE FORMATION:

Wind shear tears down the ice. Hence a stable ice formation requires a absence of wind or a wind with low velocity. Recent study by Ashton (1980) and Harleman (1986) shows that the approximate threshold conditions for the formation of an intact ice-cover are:

a. An average daily air temperature below -5°C.
b. A daily average wind speed less than 5 m/s
c. Volume average water temperature (T_{mean}) less than 2°C

Gu and Stefan (1990) found that appropriate conditions to determine the freeze over date for a specific lake called Calhoun, Minnesota, were:

a. An average daily air temperature below -2°C.

b. A daily average wind speed less than 5 m/s

c. Volume average water temperature (T_{mean}) less than 2.65°C

Some field observations in previous studies on ice formation suggest that the formation of ice covers on lakes generally start in the calm bays and along the shores, where a thin surface film of ice develops. For small lakes and ponds the ice cover typically forms some time after the water has cooled to 4°C, with the actual occurrence generally associated with a calm, clear, cold night. Wind may subsequently break up this ice cover, and the water will cool further until the winds are calm again. Hence the initial permanent ice cover and water temperature beneath it depend on the sequence of meteorological events during the cooling period. In some cases the water may cool to 0°C while in others the water may be nearer to 4°C.

After temperature falls below 4°C, ice can form temporarily several times, before a permanent ice cover can be established. The formation of a permanent ice cover depends on:

a. Level and duration of freezing temperatures at the surface

b. Size and depth of the lake

c. Wind mixing

Prior to ice formation, lake goes through 3 stages of cooling from summer to the time of lake freeze up:

a. Water cools to a more or less uniform 4°C from water surface to the lake bottom

b. Surface water cools from 4°C to 0°C as an inverse temperature stratification forms below the lake surface

c. Latent heat is removed and ice formation occurs

We can handle first stage of cooling by traditional open-water lake temperature stratification models employing a diffusive heat transport equation and convective mixing (natural as well as forced).

Latent heat removal can be treated as an instantaneous process in a daily time step model. However, it is extremely difficult to model the second stage (inverse temperature stratification).

ICE THICKNESS CALCULATION IN JAMINLAKE/LAKE36:

Ice is either formed during a simulation day or it exists from the previous day. Using the heat budget, snowmelt and rise of water in the cracks of the ice cover; an ice thickness is determined for the start of a simulation day using the method **determineThickness**().

Using the effect of wind shear and daily meteorological data, final ice thickness at the end of the day is calculated based on the natural convective flux calculated in the **forceConvectiveMixing**() method of the class **Lake.**

APPENDIX E9 – WIND EFFECTS ON THE LAKE MIXING

The effect of heat budget is to change the total potential energy of the thermal stratification by increasing or decreasing the internal energy of the lake. Wind inherently possesses kinetic energy due to its velocity. A part of this energy is transferred to the lake and is used for mixing.

The major source of turbulent kinetic energy is assumed to be the wind shear stress. The turbulent kinetic energy available for possible entrainment at the interface was estimated by the following equation.

$$\text{TKE (Turbulent Kinetic Energy)} = \int_{A_s} C W_* \tau dA , \qquad (41)$$

where

A = surface area,

W_* = shear velocity in water,

τ = shear stress at the air-water interface,

C = empirical constant.

In MINLAKE C, W_*,τ are considered constant over the surface area and the surface kinetic energy is calculated by $C W_* \tau A_s$. This kinetic energy acts to entrain the layer immediately below the mixed layer. Entrainment results in work. If the work required to lift the mass from its position at the mixed layer to the surface (the total potential energy) is less than the available kinetic energy then mixing occurs.

The work required to lift mass, $\Delta\rho\Delta V$, from its position at the interface to the center of mass of mixed layer is given by the following equation.

$$W_L = \Delta\rho\Delta V g(D-Z_g), \tag{42}$$

where

$\Delta\rho$ = Density difference between the mixed layer and underlying layer,

ΔV = Incremental Volume,

g = Acceleration due to gravity,

D = depth of the mixed layer,

Z_g = Depth of the center of gravity of mixed layer.

Usage of stability criterion is another approach that can be used to find out the entrainment of layers due to wind mixing. Interaction between the mixed layer and the layer below it can be estimated by Equations 41 and 42 or by the stability criterion.

The stability criterion is given by the following equation.

$$\sigma = \frac{\int_{ts} W_* \tau dA\Delta t}{\Delta\rho\Delta V g(D - Z_g)}, \tag{43}$$

where

σ = Stability criterion.

The other variables in Equation 43 have the same meaning as earlier variables. Coefficient C in the Equation 41 was removed from the surface integral and combined in to the stability criterion σ.

This criterion compares the kinetic influx at the time increment Δt, to the potential energy of the mixed layer relative to the layer below it. If this ratio is larger than critical value σ_{cr}, then entrainment occurs resulting in an increase in the volume of the mixed layer. The process is repeated until the ratio falls below the critical value. At this point kinetic energy is dissipated into internal energy.

APPENDIX E10- CALCULATION OF VERTICAL THERMAL DIFFUSION COEFFICIENT

Vertical thermal diffusion coefficient is calculated for each layer based on the temperature profile set earlier. $K_z(z,t)$ depends on the depth (z) and time (t) in the following manner [Fang, 1996;Thomann and Mueller, 1987].

$$K_z = 8.98\times10^{-4}(N^2)^{-0.43}, \tag{44}$$

N = Brunt Vasala Frequency

$$N^2 = \frac{g}{\rho}\frac{d\rho}{dz}, \tag{45}$$

where

g = acceleration due to gravity m/sec^2,

ρ = density of water in kg/m^3.

Vertical thermal diffusion coefficient near the lake bottom is defined by equation:

$$K_z = 100\frac{|H_{SED}|}{\rho c_p C_{w1}}(z_s + C_{w2}), \tag{46}$$

where

H_{SED} = heat flux from the sediment to the water (kcal-m^{-2}-day^{-1}),

z_s = distance from sediment-water interface (positive upward), (m),

C_{w1} = 0.623 °C,

$$C_{w2} = 0.65 \text{ m.}$$

The parameters C_{w1} and C_{w2} are determined by regression.

In this model K_z is determined at each depth and time-step by Equation 44 downward and from the ice –water interface and by Equation 46 upward from the lake bottom. N is determined from the already known water temperature profile calculated in the previous time-step or by an iteration process. The maximum turbulent diffusion coefficient K_{zmax} is used as an upper bound.