1981 Winter Simulation Conference Proceedings
T.I. Ören, C.M. Delfosse, C.M. Shub (Eds.)

AN ON-LINE SIMULATOR AND DATABASE SYSTEM FOR MANAGEMENT OF A COMMERCIAL FISH FARM

A. Neil Arnason, Carl J. Schwarz and David H. Scuse
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, R3T 2N2
Canada

ABSTRACT

We have developed an on-line system to aid in the management of enclosed, multi-tank fish farms with water filtration, heating and recirculation, where fish are grown from fingerling through to market size (about 250 gr.). The manager has a high degree of control over tank loadings and fish rearing conditions, but he must maintain high production rates within biological constraints on water quality and fish density and within system constraints imposed by the physical plant. A multi-level interactive simulator permits exploration of management strategies, with nested levels of constraints. A database is used for storage of both real and simulated data on management interventions and their consequences on fish growth and water quality. Interaction between the simulators and the database aids model calibration and validation and permits storage of promising management strategies.

1. INTRODUCTION

Since April, 1980, we have been developing a system of programs and a database designed to aid in the management of commercial fish farms. The work has been carried out in conjunction with the Aquaculture project of the Freshwater Institute, Fisheries and Oceans Canada. This research section of a Canadian Federal Government ministry operates the Rockwood Experimental Hatchery near Winnipeg, Manitoba. One of the purposes of the hatchery is to demonstrate the commercial feasibility of enclosed, environment-controlled fish rearing facilities for the production of market size (200-250 gr.) salmonid fish reared from 5-10 gr. fingerlings.

Such facilities, of which Rockwood is a small-scale prototype, have several tanks, each of a given size and shape, each holding a given volume of water in which the fish are reared. The water temperature and quality are maintained by the water flow system. Water from tanks is partly discharged, but the majority passes to a filter (usually one to each tank) where pollutants (fish metabolites and particulate matter such as feces and excess food) are removed by mechanical filtration and bacterial action. This action also removes oxygen from the water, beyond what has already been removed by the fish in the tank. The filtered water is then recirculated back into the tank after mixing it with heated make-up water. The make-up water, which replaces any discharge, is heated by mixing unheated ground water with water heated by low-grade (solar or waste) heat. Large holding tanks of heated and unheated water are maintained, and this can be fed in any combination to provide make-up water of different temperatures to individual tanks. The combined make-up and recirculated water is fed back into the rearing tanks through venturies and spray heads which partly or completely re-oxygenate the water (depending on flow rates, fish densities, etc.). Typically the turnover rate in a tank (i.e. the time to input one tank volume of water) is around a half hour but there may be considerable variation in this variable among hatcheries or hatchery configurations. The size of the filter, in combination with the turnover (flow) rate may place limits on the loadings of fish and food that can be placed in the tank. In any case, filters gradually decline in their efficiency of pollutant removal and must periodically (every few weeks) be flushed in a process called backwashing. While this is being done, the fish in the tank are maintained on straight flow-through of make-up water with no recirculation. The process requires 15-30 minutes per tank.

Many variations on this physical system are possible: variations in numbers and volumes of tanks, in filter sizes, in the degree of temporal and

by-tank control of flow, temperature, filtration, etc., in the amount of water and heat available. Our management programs and database are designed to incorporate information about this physical configuration and to adapt their functions to existing or hypothetical configurations, different from, but of the same general design as that at Rockwood.

The salient feature of the facilities that we are considering is that the manager has a high degree of control over the rearing conditions. Rearing conditions are those variables which have a direct effect on the growth and death rate of fish. They include water temperature, feeding variables (food type, feeding rate and method), and water quality variables (mainly oxygen level, toxic metabolite levels such as N-NH3, N-NO2 and pH as it affects equilibria). Growth rates also differ among species and among genetic strains of the same species. At Rockwood, most of the rearing experiments have been done using various strains of rainbow trout (Salmo gairdneri), but work is also beginning on arctic charr (Salvelinus alpinus). Many of the system variables have no appreciable effect on growth or death except at extremes of high fish density or low water quality but these limits, which we call biological constraints, depend on fish size, growth rate, and temperature (among other things). Biological constraints can be considered, for the moment, to be the maximum density and minimum water quality conditions that are tolerable by the fish. We shall need to define this term more precisely when we introduce the various levels of growth models.

In an actual hatchery, the manager has varying degrees of direct or indirect control over the environmental variables which affect rearing conditions and determine if biological constraints are exceeded. Changes to such variables, over which the manager can exercise direct and independent control are called interventions. The easiest changes to make involve tank loadings of fish and the feeding variables. Each tank can be stocked with a given number of fish, of a given weight or distribution of weights. A feeding regimen for each tank is carried out: a certain amount of a specified food is distributed to each tank in a set way daily. Amounts are specified as a certain feeding percent, typically aimed at or just under maximum ration which is the (size and temperature dependent) amount just needed to achieve maximum growth rate. We will also refer to a parameter called feeding level which is the actual feeding percent as a fraction of this maximal ration. Feeding level thus varies from starvation (feeding level = 0) to maximum ration (feeding level = 1). Tanks are censused or sampled on a regular basis (usually weekly) to determine their current total weight and mean weight per fish so that the feeding regimen can be updated. As the fish grow, tank loadings (kg./tank) generally increase and so the tank loadings must be re-distributed. Some or all of the fish in a given tank may be moved randomly (i.e. not size selectively) or by grading (usually larger fish are moved or "promoted") to an empty tank or to a tank with other fish already in it. Three types of movements are distinguished: an ADD of fish from outside the farm into a tank, a MOVE of fish from one tank to another, and a REMOVE of fish from a tank for sale or discard. All such movements can potentially be done at any given time, from or to any given tank, on a numbers or percentage by weightclass basis. In practice, movements are usually kept as simple as possible, since the consequences of all but the most regular interventions are difficult to foresee and because there is a need to keep fish sizes within tanks fairly uniform. A typical strategy is to empty out tanks with (mostly) market size fish every 6 to 8 weeks and to use the spare tanks to split the fish in the remaining most heavily loaded tanks. If any tanks remain, a new batch of fingerlings is started.

The water variables are also subject to interventions by altering the percentage and temperature of the make-up water, possibly on a tank-by-tank basis. Flow rates are usually fixed by pumping rates, but may be variable in some systems. Changes to any of these will produce new equilibrium levels for the water quality variables which must be kept within the biological constraints for the current tank loadings. The interventions must also be within the range of system constraints that are imposed by the availability of heated and unheated water in the holding tanks, maximum pumping rates, filter capacities etc.

Two points should be evident from this brief outline of hatchery management practices. The first is that the system generates a large amount of data that is important both for day-to-day operations and for planning purposes. The manager must keep track of tank censuses and the interventions carried out. Measurements of water temperature and quality (pH, oxygen concentration, ammonia levels) are made regularly, possibly in any or all of the tank-, the recirculation- and the make-up-water. Data is also collected on the various flow rates and on the volume and temperature of water in the holding tanks as well as ambient air temperature. These data permit monitoring for deterioration of environmental conditions, checking how close the system was run to the biological and system constraints, resetting of feeding regimens, and historical reporting of the consequences of interventions on growth, deaths and environmental conditions. In a production commercial facility, a considerable amount of data on costs and benefits will also be required. The second point is that, even with ready access to this information, management is a complex and difficult task. While the manager has a high degree of control, he is also subject to complex and inter-dependent constraints. Moreover, he must balance the interventions open to him now against the possible consequences of each at some point down the road, and these consequences can range from the merely suboptimal through to the irreversibly disastrous. He must make a trade-off between production and risk: at lower loadings and slower growth rates, the fish are less stressed, there is less need to worry about exceeding biological constraints, more leeway for error and more time to correct mistakes. At high tank densities and growth rates, he is subject to higher risk, yet it is precisely at such levels that the facility should be run if the higher capital and operating costs (relative to simpler open-pond systems) are to be compensated for.

Despite the higher fixed costs and greater demands for skilled management, there are considerable advantages, over open-pond systems, to controlled environment systems. They provide managers with more opportunities to optimise operations and to maintain production at high, year-round yields, with minimal and foreseeable risk. Moreover, their efficiency is high because up to 90% of the water can be recycled and because heating can be provided by solar heat or waste heat from generating plants, refineries and compressor stations. There is an increasing interest in Canada in using such low-grade but non-transportable heat sources to produce high value, easily marketed food resources.

We have developed a database to meet the data storage and reporting needs of management and a set of simulator programs that permit the manager to simulate the growth and water quality conditions that would result from arbitrary interventions, starting from specified initial conditions, within a specified physical configuration. In Sections 3 and 4 of this paper, we will describe the simulator capabilities and the database capabilities as independent entities. In fact, however, their individual usefulness is greatly enhanced by integrating these two capabilities to support one another. The reasons and means for doing so are the subject of Section 5. As a preliminary to these discussions, the next section describes the software developments implemented to support the on-line interactive simulation and database query systems.

2. INTERACTIVE SOFTWARE SUPPORT PROGRAMS

Most of the programs used by hatchery management (the user) are on-line interactive programs where the user is either prompted for specific inputs (of names, numeric values, responses such as yes/no, or lists of names or values) or he is issued a command prompt. At this point the user can enter a command directing the system to carry out an operation from the repertoire of specific actions that are appropriate at that point in the simulation or database query system. Each command type begins with a specific keyword (such as RUN, LIST, MOVE, etc.) and is followed by a definite syntactic structure for that command. The syntax is designed to be simple and natural and to make a reasonable English sentence.

The parser interface for handling this dialogue is sufficiently general and self-contained that it can be used with almost no modifications in any similar system where a powerful, user-oriented command language must be supported. The parser interface contains features that make both the development of an interactive system by application programmers and the use of the system by inexperienced users quite simple.

The parser interface was patterned on the parser used in the SIMSCRIPT system (Kiviat et al. 1975). The parser accepts both iterative and recursive specifications of grammars so that recursive languages are supported but iterative languages are not complicated unnecessarily with recursive definitions. Also, the parser interface handles error recovery in ways that are not possible in batch systems such as SIMSCRIPT.

To use the parser interface, the administrator of the parser, in consultation with the users, first defines the language to be used in the application system. The administrator then creates the formal BNF (Backus-Normal Form) grammar for the language, defining basic keywords and their operands as primitives and grouping primitives together to form the commands of the language. Instructions for defining a command grammar are given by Karasick (1981) in terms that can be understood by programmers with no experience of parsers. A notation in the BNF specification permits choices among alternatives at any point in the command and/or repetition factors for parsing lists or repeating phrases within commands. Repetition factors also control whether primitives are optional or compulsory in the command. (The administrator normally includes optional descriptive phrases in the language that act as documentation for the inexperienced user but which may be omitted by the more sophisticated user.)

After the administrator has defined the grammar for the language, the grammar is read and analyzed by a batch program, the parser generator. This program ensures that the language is valid; if no errors are detected by the parser generator, a set of tables that define the language is created.

The processing of user commands is performed by the scanner and parser routines; these routines are included with the other routines in the application system. The scanner prompts the user for each command line and then reads the command line and breaks the line up into the component tokens. Using the table produced by the parser generator, the scanner determines the type of each token (keyword, operand, etc.). Each token and its type are then passed to the parser routine which builds a parse tree consisting of the operands entered by the user. The parser front-end then passes the parse tree to an application program, referred to as an action routine, which performs the processing for that command. There is one action routine for each command in the language; however, since much of the processing for each command is common to other commands, each action routine normally consists of CALL statements to subordinate routines which perform the actual processing. If the language is non-recursive, register variables can be specified in the grammar as being associated with particular keywords or operands in the parse tree. This facility allows a "window" directly into portions of the parse tree so that the programmer knows exactly where user-entered operands are located without having to perform a search of the parse tree. If a register variable is associated with an operand that can be repeated, the register variable points to a list of pointers into the parse tree.

Since both the scanner and the parser are driven by the tables produced by the parser generator, almost no modifications are required in order to use the scanner and parser routines in a different application. The scanner should never need to be modified, and the only change to the parser that is required is the addition of a front-end that contains a CALL statement to invoke the associated action routine for each command in the language.

In order to make it as easy as possible for the user to enter a command correctly, the parser generator automatically abbreviates keywords as much as possible. (For example, "TANKTEMPERATURE" was abbreviated to "TANKT".) However, the administrator may over-ride this feature for a keyword and specify explicitly the number of characters that must be entered before a keyword can be recognized. This feature ensures that certain major commands (for example, the END command that terminates a session) are not executed as the result of a typing error.

As a part of the grammar, the administrator may also define aliases for keywords and frequently used operands. These aliases are automatically abbreviated by the parser generator. (For example, "FTYPE" was defined as an alias for "FOODTYPE"; "FTYPE" was then automatically abbreviated to "FT" by the parser generator.) As part of its processing of a grammar, the parser generator examines the abbreviations and aliases to ensure that there are no ambiguities. If an ambiguity is found, the parser generator issues a warning message and indicates how the ambiguity will be resolved. The administrator may either accept the parser generator's resolution of the ambiguity or make the appropriate change to the grammar to remove the ambiguity. At the end of processing, the parser generator prints a list of the abbreviations and aliases that have been defined for each keyword and operand.

The scanner also supports a text-substitution feature (similar to SIMSCRIPT's "DEFINE TO MEAN") and sophisticated command retrieval and editing features that allow the user to reduce the typing of long inputs. The scanner performs all of the manipulations of abbreviations, aliases, and substitutions in order to keep the parser independent of the grammar. As a result, abbreviations and aliases can be changed by regenerating the parser tables without having to modify the action routines.

Because of the latitude allowed by the parser system in accepting abbreviations, aliases, and substitutions, the user should not have much difficulty in entering commands correctly. However, should he make a mistake in entering a command, the parser generates a detailed error message (as opposed to the message "ERROR, RE-ENTER INPUT" produced by many systems) that tells him which part of the command is invalid and the reason why it is invalid. For many errors, the user is prompted to re-enter only the portion of the command that is invalid and the remainder of the command does not have to be re-entered.

The parser interface removes much of the tedious work involved in implementing an application system that requires a complex command language. The parser can be installed quickly and easily because almost no changes need be made to the source code for the parser. Because of the work performed by the parser generator in determining how each command must be parsed, the parser routine is able to parse each command very efficiently.

3. THE SIMULATOR PROGRAMS

Two fish growth programs are available to the user. They are referred to as the 'single-fish' program and the 'multi-tank' program.

3.1 The Single-fish Program

This simulator uses a deterministic differential growth equation model to simulate the growth of a single (representative) fish without density, biological or system constraints. Specifying tank or system information is thus unnecessary. The program is designed to permit the user to explore the properties of the underlying model (which is also the basis of the multi-tank model) in response to various sets of initial conditions and rearing conditions. Rearing condition variables are: species (or strain), food type, feeding rate (either percent or level) and temperature. Initial conditions are the initial weight and time interval increment (DT or WITH time), reporting interval (BY time) and elapsed time (FOR time). Species and foodtype are generally chosen in response to prompts and then the user will generate tables of results by using the three main commands to SET rearing conditions, RUN the model, and LIST various growth consequences. For example:

```
SET INITIALWEIGHT=(.05,.1)
SET TEMPERATURE=(8,10,12) FEEDLEVEL=(.5,.75,1.0)
RUN FOR 4 WEEKS BY 1 WEEK WITH DT=1 DAY
LIST INITIALWT, FINALWT, INSTGROWTH, FEEDPERCENT
```

will produce a table of the variables in the LIST command that result from the first week's growth using daily (but unreported) updating. The table has 18(=3x3x2) lines, one for each possible combination of rearing conditions. Initial weight is then automatically re-assigned the final weight and the table is regenerated for the second week's growth, and so on. The differential equation for growth is based on that given by Sparre (1976). Sparre's model includes an explicit relationship between feeding percent and feeding level so that the user can specify ration in absolute or percentage amounts or by feeding level. These conversions are recomputed every update interval (DT), as are oxygen consumption and ammonia production rates as these change with increasing fish weight. The model in fact has an analytic solution for fixed feeding level, but it is not realistic to use this as it would imply continuous adjustment of feeding percent as the fish grow. When feedweight or feedpercent is specified, the updates provide a finite approximation to continuously changing feeding level. The results turn out to be fairly insensitive to the choice of DT, provided DT is kept fairly small, generally less than 1 week, but possibly smaller at very high growth rates. Auxiliary equations (currently also from Sparre) predict oxygen consumption and ammonia excretion over the growth interval, but we are in the process of revising these equations (see McLean 1979 and Paulson 1980 for recent reviews). We are also revising Sparre's model, which was developed for open ponds at relatively low temperatures, to give more realistic temperature responses. Sparre's model gives monotone increasing growth rates as temperature increases (for fixed rearing conditions other than temperature), but

salmonids generally show an optimum growth rate at around 16 degrees centigrade, dropping off steeply above that, and more slowly below that. The nature of the response is highly species-specific, but most salmonids can grow at temperatures in the range 5-20 degrees. The parameters of the growth model are species and food-type specific; these are provided by matching the species and food name to labelled entries in the species file. Originally we had also hoped to have a food-type file that would permit the conversion of one type of food into equivalent units of a standard food, but fish physiologists have convinced us that this hope may be illusory. Species file entries are thus each specific to a particular food and strain, and some entry must match both the user-supplied names before simulation can proceed; that entry then supplies the parameters for the growth and auxiliary equations.

3.2 Model Calibration

Entries to the species file require calibration (fitting) of the model to experimental data. The data is generated in carefully planned experiments over a factorial combination of rearing conditions. Variation in growth response at each level of conditions is used for weighting the mean response and for fitting an auxiliary equation for variation about the mean required by the multi-tank programs. However, calibration is largely a process of finding those parameters that make the predicted growth at those levels most closely correspond to the observed mean growth. Some of the parameters can be found by using weighted multiple linear regression on transformed growth and level variables. Particularly useful are experiments at maximum ration and at starvation since each of these causes some of the growth parameters to drop out of the model (see Sperber et al. 1976). However the remaining parameters will have to be chosen by non-linear optimisation; i.e. a directed iterative search for the parameters that minimize the distance between the observed mean response and the predicted responses. The single-fish program factorial tabulations have been designed to provide those predicted responses in the same form as actual mean responses. We are in the process of developing programs that will automate this calibration process as much as possible, by extracting data from the database, checking it for completeness; formatting it into tables, and invoking the single-fish routines and regression routines to carry out the job. As we shall see, there are many other auxiliary equations that must also be fitted before the higher level multi-tank models can be used.

3.3 The Multi-tank Program

This simulator is intended to allow the user to simulate, in a single session, every significant action and observation that could be carried out at a real or hypothetical production facility over an extended period of time. Here the population is structured by weight classes and by tanks. The underlying model is a second-order Markovian model: Markovian in that the current state of the system, but not its history prior to the current time, determines its future course; it is second-order in that it accounts not only for average growth but the variability (in fact the entire distribution) about that average. The model is nevertheless still entirely deterministic, in that we make no use of random numbers or variables.

For a group of fish, all of identical initial weight at time t, grown for a known time (dt) under identical rearing conditions, the mean final weight (at time t+dt) can be computed from the single-fish model. An auxiliary equation predicts the variance as a function of dt and mean final weight w(t+dt). This completely specifies the final weight distribution if one assumes a Normal distribution, which can reasonably be assumed over short growth intervals. However, the multi-tank program does not keep track of individual fish, nor the complete distribution of groups of fish, but only of counts of fish per weight class. This can be obtained by integrating the Normal distribution in each weight class j, from say UB(j-1) to UB(j) where UB is the upperbound of the j-th weight class. This gives $p(j|w)$, the probability that the fish are in class j given that they began at weight w. The numbers of fish out of n, in each class are, in expectation, $n(j)=n.p(j|w)$. The $p(j|w)$ are multiplied by a further auxiliary equation to model survival, as a function of initial weight and dt, giving $n(j) = p'(j|w) = n.s(w,dt).p(j|w)$ where s is the survival function. The n(j) are, in fact, the expected values of observations from a multinomial distribution with parameters n and $p'(j|w)$ and this implies independence of growth and constant assignment probabilities for all fish. There is some evidence (Thorpe 1977) that this may not be so; it shows up in bimodality in the distribution of fish grown together over several months, and there is some controversy as to whether it is due to genetic variation, or behavioural interactions (e.g. dominance) among the fish. It is to avoid such problems that hatchery management tries to keep the weights of fish within tanks fairly homogeneous. This practice also keeps the growth model accurate without the need to model dependence and non-Markovian effects.

The $p'(j|w)$ do not completely specify the growth dynamics, because we do not record the weight w for all fish at time t, but only have a count n(i) of the number of fish in class i at time t. The distribution of weights within classes also contributes to the variation in the final weight distribution. Sparre (1976) solves this problem by assuming the n(i) fish are uniformly distributed within their weight class and integrates out initial weight w(i) over UB(i-1) to UB(i) in the equation for $p'(j|w(i))$ to produce the transition probabilities p(i,j), the proportional contribution per fish in class i at time t to class j at time t+dt. The sum of the elements in row i of this matrix can be seen to give the average survival rate of class i fish. We found Sparre's approach too expensive to implement, as the transition matrix is large and must frequently be recomputed, and both the uniform and Normal integrations must be done numerically. We replaced the uniform assumption by a two point discrete distribution within class i at wm(i)+g and wm(i)-g where wm(i) is the midpoint weight of class i, (UB(i) + UB(i-1))/2, and g is chosen to give the same within class variance as a uniform distribution. Thus the resultant distribution, induced by growth of the n(i) class i fish, is the mixture of

two equally weighted multinomial distributions. One of the numeric integrations has been replaced by the sum of two (vector) terms, giving a row of the transition matrix. Multiplying the vector of initial counts n(t) by the transition matrix T(p(i,j) | dt,R) for specific rearing conditions R, gives the final expected structure n(t+dt).

This procedure works well if the class boundaries are fairly narrow. We have implemented it using some 50 background classes where class width is never more than 10% of the lower class bound. It produces growth runs, under fixed rearing conditions starting with fish all in one weight class, which are very close in expectation to the predicted mean weight from the single-fish model using the same rearing conditions and initial weight. This correspondence is vital if the calibrations of the single fish model are to have any relevance to the multi-tank simulations. The user does not want to see the census in every tank for every background class however. We have therefore allowed the user to define up to 20 foreground classes whose cut-points align with a subset of the background classes. Whenever a census is displayed, the background counts are aggregated up and displayed rounded to integers. For accuracy, background class counts are kept as reals. The user can also display the background census if he wishes. The foreground cutpoints can be RESET at any time and all simulations are invariant to the current foreground definition.

The multi-tank model permits much more than simple growth runs under fixed rearing conditions. The user is prompted to specify the level of constraints he wishes imposed, the number of tanks, and his foreground class definitions. For each tank, he specifies a label (tank name) and, if necessary, a type which refers to tank configuration parameters (volume, filter capacity, flow capacities, re-oxygenation parameters, etc.) stored in a tank-type file similar to the species file. He is then prompted for the species, food and initial (by weight class) vector of fish loadings. After the initial prompts, the system issues the command prompt which is the current period (P=) and time (T=), both initially 0. Period is a counter of the number of run iterations that have been reported (BY increments in the RUN command). The user can now SET rearing conditions (including changes to species in empty tanks) such as temperature and feeding regimen (by feed weight, percent or level) for each tank. He can then RUN FOR a certain elapsed time, saving results BY given intervals WITH updating every DT time units. Results for the census and conditions in each tank are saved out to a file or to the database at each BY period. When the RUN is done, the updated period and time prompt is displayed and the user can LIST census, growth and related variables in some or all of the tanks at the current time. He can then ADD new fish into tanks, MOVE fish between tanks, provided he does not attempt to mix fish of different species, and REMOVE fish from tanks, all by numbers or percentages by weightclass. These movements can be done by reference to either foreground or background classes. The rules for how a given number of fish in foreground are distributed over background will not be given here, but the ability to specify foreground classes makes the movement commands more succinct and powerful. Now the user can again SET rearing conditions, or use LIST to see the effects of his movements and SETs, and then RUN again. In this way, the user simulates an actual or hypothetical strategy of interventions and conditions. If the simulation runs into infeasible loadings or environmental conditions, the user can BACKUP to a previous period and try a different strategy. One can, in fact, BACKUP and ADVANCE freely through the saved tank structures file, and this will re-locate the user to some previously generated time and re-establish the conditions, at that time. He can then try another strategy from that point, using SET, MOVE, RUN etc. Depending on whether he is saving to a stand-alone flat file or to the database, a RUN may or may not obliterate all saved structures after the current time. Detailed instructions for using the simulators are given in Arnason et al. (1981).

Levels of multi-tank modelling. To help the manager cope with the difficulties of planning intervention strategies that will keep within biological and system constraints, we are permitting use of the multi-tank programs at one of 5 different constraint levels. A sixth level, that keeps track of the costs and benefits of interventions, is planned. The 5 constraint level models are listed below in order of increasingly strict biological constraints. These levels are also in decreasing order of the directness of control over the environmental conditions provided by the interventions which are at his disposal at this level. (Recall that interventions are variables which are always immediately and independently manipulable by the manager.) Thus finding feasible strategies becomes more difficult at higher levels. The multi-level system permits him to find promising strategies quickly at a lower constraint level and then to test if they are viable at a higher constraint level, or conversely, to find additional or less direct-acting interventions that will keep conditions within the higher level constraints while preserving a lower level growth strategy.

Having specific growth models and auxiliary models for environmental effects permits a more explicit definition of rearing conditions and biological constraints. Rearing conditions are those variables which are needed to completely define the growth, death and auxiliary equations for a given level model. The biological constraints are the range of environmental conditions within which the rearing condition variables give valid model predictions and acceptable growth and survival rates according to user-set limits. A variable, such as oxygen in the tank water, may enter only into the biological constraint equations at one level, but at a higher level, may become a rearing condition variable where its effect on growth must be explicitly modelled. The range of values over which the model is valid remains part of the biological constraints. When biological constraints are exceeded, the user is warned and the RUN terminated at the last feasible (BY) time.

Of the following levels, 1 is fully implemented, 2 and 3 are partly implemented and 4 and 5 have been allowed for in the program and command structure but are not yet started.

1. The Basic (unconstrained) model: This model is analogous to the single-fish model. The user must specify for each tank a valid species/foodtype that exists on the species file. Tank names, but not types, are specified merely as identifiers of individual tanks for use in movement and SET commands. The rearing conditions are feeding rate and temperature and these are also the permissible interventions (in addition to movements). Death rate is independent of rearing conditions. This model calculates, in response to interventions, the new census, total weight, instantaneous growth rate, and keeps track of total food fed, oxygen consumed and ammonia produced on a tank by tank basis.

2. The Tank Effects model: The user must now specify a valid tank-type for each tank so that tank volume and filter parameters are known. He must SET recirculation and makeup rates but temperature can be either directly set or indirectly set by setting air temperature and makeup temperature. No restrictions are placed on flow rates. Tank levels for oxygen and ammonia concentrations are computed and these can be checked to see if they are within biological constraints given the current density, mean fish weight, growth and feeding rate. Warnings are issued, but the rearing conditions are as for the Basic model and deaths, within the constraints, are still independent of rearing conditions.

3. The Tank Constraints model: The user has the same level of intervention control as in level 2 but now oxygen and density become rearing conditions in that they affect the death rate. There is some evidence (M. Papst, pers. comm.) that they should also begin to affect growth rate long before there is an appreciable effect on deaths. The biological constraints are adjusted to operate on the death rate instead of on oxygen level directly. Ammonia levels are still checked directly. Recirculation and make-up rates must be within maximum tank capacities. Declines in filter efficiency will be modelled and backwash interventions must be made. If temperature is specified directly, it must be in the range that can be delivered by the make-up water and will determine the make-up percentage needed to achieve that temperature. If any constraint is exceeded, simulation stops.

4. The System Effects model: This model will compute the total simultaneous demands made by all the tanks on the physical plant. A system file will specify volumes available in the holding tanks and maximum replacement rates. Temperatures in the holding tanks will be set by the user, and volumes and rates can be exceeded by the fish tank settings with only a warning.

5. The System Constraints model: Here, demands on the system that exceed the specifications in the system file entry will halt simulation. In addition, water and air temperatures and availability of heated water will not be under direct user control, but will vary according to seasonal averages. Simulations must therefore begin at a user-specified absolute date (e.g. January 1) within the year.

We are only just beginning to develop the many auxiliary equations needed to model environmental effects and to establish biological constraint equations. A second year's cycle of production runs at the Rockwood hatchery and a set of special calibration runs will permit us to judge the validity and relevance, for these purposes, of the many models and relationships that have been put forward in the massive literature on fish growth, physiology and tank filtration systems. Initially, we will use literature values and the experience of hatchery personnel to set constraints very conservatively. As the data in the database and the means to manipulate it and report it increase, the constraints can be fine tuned to permit more daring strategies.

4. THE DATABASE SYSTEM

The database system programs have received as much attention as the simulator programs. They include programs to edit hatchery data and load it to a structured database; batch programs to create, structure or restructure, and preserve backups of, the database; batch reporting programs to produce large, time ordered, collated reports on database contents; and an on-line QUERY program for interactive interrogation and display of database contents. Perhaps the most interesting program feature is a database interface, which facilitates the development of applications programs which need to access the database.

4.1 Data Collection and Entry

Hatchery data forms have been designed in collaboration with hatchery management personnel. Currently, data is transcribed to these forms from an existing paper records system, but we shortly expect to revise the forms and develop regularised procedures for what data is to be collected, and when, and then switch to direct recording on the data forms. These procedures are beneficial in their own right as they tend to formalize the data collection process, help to ensure data is complete, correct, encoded according to uniform and well documented standards. The very process of designing the forms and procedures forces hatchery managers and database analysts to clarify the purposes to which the data will be put and to ensure that all associated measurements are made so that those purposes can be met. For example, to assess the effect of movement and flow interventions on environmental variables it is vital that the latter variables be recorded just before the intervention and just after equilibrium conditions are re-established (usually within the time taken for one tank turn-over of water). In general, the proper time sequencing of all records is crucial, but this can be difficult to ensure with irregular procedures. The following data forms are currently used.

1. Group form: this identifies a given production cycle at a specific hatchery. It is not necessary to identify system configurations and constraints, but simply to identify a system (hatchery) name that exists on the system file (like the species

and tank-type files, whose entries are created by the system developers).

2. Tank forms: identifying each of the tanks by name and tank type (which must exist on the tank-type file).

3. Species in tank form: identifies the date, tank, and species name whenever a new species is placed into an (empty) tank. The database will not permit mixing of different strains or species in a tank.

4. Intervention forms: records the time, tank and new settings whenever feeding regimen (foodtype, method or amount), flow rates (recirculation or makeup), or backwash frequency are altered.

5. Environmental Variables Forms: record oxygen levels, pH, temperature, and metabolite levels (ammonia, nitrate, nitrites) in tank, filter outflow, and recirculation inflow water. Air temperature is also recorded.

6. Census forms: these are the most complicated as they may specify varying amounts of detail about the number and size distribution of fish in a tank at a given time. It may give only a total count, or an estimated count and mean weight per fish based on a small (25-100 fish) random subsample, and may include length data on individual fish in the subsample. The latter data provide the only means by which an actual weight distribution can be inferred (after transforming to weights using a log-linear weight-length relationship). Census data rarely contains a complete census, except when an empty tank is initially loaded with fingerlings. It should be noted that real census data is by no means as complete or as conveniently summarized as a multi-tank simulation census. With some care to ensure data completeness, and a good deal of "massaging", the weightclass distribution can be estimated from the census data in the database.

7. Movement forms: are a census form for the fish being moved with the time, from-tank, and to-tank specified as well. (ADDs are from special tanks called BUY or NEW and REMOVEs are to SELL or DESTROY). A special null census is recorded for the from-tank if the move has emptied out the tank. In other cases, it can be difficult to infer if a random or a size selective sample was taken from the from-tank, but movement forms always report, at least, the total count and total weight of fish moved.

8. Death forms: record the tank, fish length, and removal time of individual dead fish.

The differences between real and simulated data, mentioned here, are relevant to the discussion of the simulator/database interaction discussed in Section 5.

Once data forms are key punched, they are run against a batch program that does range and type checks on all fields. When this program runs without error, the data cards are sorted so that group forms and tank forms precede all other forms. These database segments must be created first as the database load program rejects forms that refer to tanks for which database segments do not exist. The load program is run against the sorted input data and writes out an error log for any forms that need corrections. The level of consistency checks on the data is still fairly rudimentary. We do not check for correct time ordering and associations among segments (e.g. that a census is present for each intervention), nor that census or movement data are consistent with past census, death and movement records for that tank (e.g. that a movement segment does not remove more fish than are actually in the tank). Many of these checks will be possible when an interactive UPDATE database loading program is developed.

4.2 Database Structure and Interface

The fish farm database is manipulated using IBM's Information Management System (IMS) database system. The IMS database is quite complex, involving logical relationships and special indicators in the segments. All of the information from the input forms is stored in IMS segments, with each segment containing the information about a particular entity (for example, a tank, a temperature change, etc.). The organisation of the segments in the database is shown in Fig. 1.

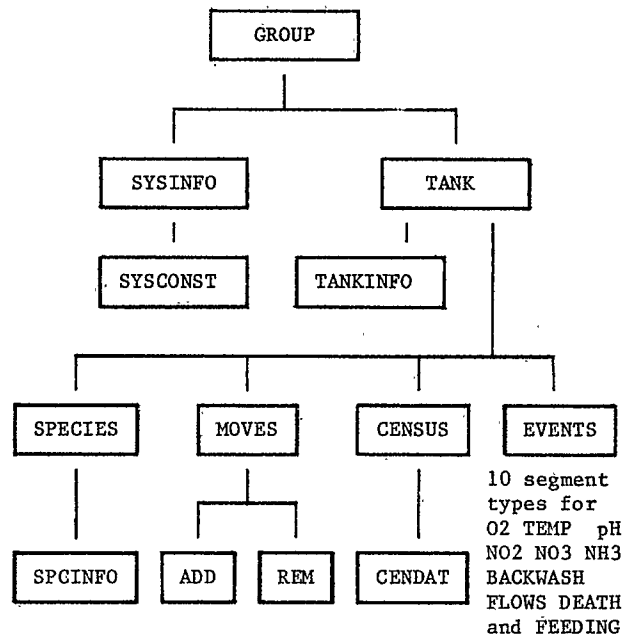

FIGURE 1

The GROUP segment contains the description of an experiment at a hatchery, including the date on which the experiment was started, and any constraints that apply to the hatchery. Subordinate to each GROUP segment is one TANK segment for each tank being used in the experiment. The TANK segment identifies the tank and contains any constraints that apply to that tank. Subordinate to each TANK segment are the segments that describe the contents of the tank and the events that have occurred during the experiment. The database routines automatically insert the BUY, SELL, NEW, and DESTROY tank segments. A SPECIES segment defines the type of fish in the tank and the major

characteristics of that species. The remaining segments store information directly off the corresponding input forms. All of the segments subordinate to the TANK segment (except TANKINFO) are stored in date-time order so that accessing the segments provides a picture of what happened in the tank as the experiment progressed. The MOVEMENT (MOVES in Fig. 1), ADD, and REMOVE (REM) segments require extra processing by IMS because they participate in an IMS logical relationship. Whenever fish are removed from (or added to) a tank, the corresponding number of fish must be added to (or removed from) another tank in the group. This processing is required in order to keep the database consistent and to simplify the processing in the application programs.

The complexity of the IMS database used to store the fish farm information necessitated the inclusion in the system of a routine that would simplify access to the database. This routine, called the database interface, controls all access to the database: any application program that requires information stored in the database issues a request to the database interface and the interface then issues the appropriate command to IMS. The interface was used not only to make it easier to extract information from the database but also to provide data management facilities that are not provided by IMS.

All information that is required to issue a request to the interface is included in a data structure that the application programmer copies into the application program from a system library. This data structure defines the format of the segments that can be accessed and also contains command and key fields that are set in the application program, and status fields that are interrogated by the application program. The application programmer sets the command field for each type of segment to be processed (more than one segment type can be processed with each request), and, if a particular segment is to be retrieved randomly, sets the key field of the segment. Special commands were defined to permit the application programmer to retrieve the first segment with a key greater than or the first segment with a key less than the specified key since much of the processing is in date-time order and the application programmer may not know the exact time (field) of the desired segment. This is particularly important for the QUERY programs described in the next section.

The system, tank and species information segments indicated in Fig. 1 are not, in fact, stored in the IMS database, but in the flat (random access) files which we have been calling the system configuration, tank-type and species files. This permits easier modification of the information, and reduces the complexity of the IMS database since the information in the files could not have been stored there without significant redundancy. Nevertheless, the programmers' view of the database is as if it were structured as in Fig. 1. The interface handles the access to these files and the programmer does not need to be concerned with how they are structured or where they are stored, even if these should change.

The use of the database interface has made the development of the fish farm system much easier because the application programmers have been able to concentrate on the high-level manipulation of the information in the database instead of having to be concerned with the low-level details of how the information had to be accessed using IMS or the flat files. Some additional ways in which the interface simplified the writing of application programs are given in Sections 4.3 and 5. The database interface reduces the time and coding required to implement the database functions of the fish farm system and to ensure that they are operating correctly. A more detailed discussion of database interfaces is given by Scuse (1981).

4.3 Database Reports

Information from the database can be generated using batch programs for large printed reports or using an on-line QUERY program for selectively exploring the database and producing short comparative reports.

There are 2 main batch programs written using EASYTRIEVE/IMS (Pansophic Systems Inc.). The first is a dump of the complete database in storage order that is convenient for checking the information fields against the original records. Thus the information fields are listed, in the following nested order (slowest changing variable first): for every group, for every tank, for every segment type, from first to last time. The second, and most useful report, is the within-tank merged information report, which produces a listing of the following form, for each tank and with times in increasing order:

TIME	SPECIES	MOVE	CENSUS	EVENT	ETYPE
date		data	data	(removes to)	
date			data	(empty tank)	
date	data			(new species)	
date		data	data	(added)	
date				data	FEED
date				data	02
date			data		
date				data	FEED

Now, for the first time, the manager can see a log of everything that happened in a tank, in the order it happened or was reported. He can see how water quality variables change in response to tank loadings and other interventions (feeding, flow, temperature). He can easily identify "growth-sets": time intervals when a group of fish are subject to given rearing conditions during which no fish are added to or removed from the tank, apart from deaths. With adjustments for any deaths, the census and feeding data within such intervals can be used to compute actual growth rates and conversion ratios (kg. food fed/kg. growth produced). The merged reports also facilitate comparisons between tanks or groups and will be altered to permit comparisons of SAVEd simulation results within a group (see Section 5).

The on-line QUERY program has a DISPLAY command that produces a succinct table of contents of the database (group and tank names and descriptive comments). A very powerful LIST command then

permits the user to specify a group or set of groups, a tank or set of tanks (which must be common to the groups if more than one group is specified) and a time or range of times, and finally a list of variable names. Currently, all the variable names identify various segment information fields. A table of these variable values is then given for all groups, tanks, and times. However, QUERY also remembers your "position" (group, tank, time) in the database, and this allows the user to use LIST sequentially, in an exploratory way, by specifying the TIME keyword within the LIST command as NEXT, PREVIOUS, FIRST, LAST, etc. Features in the database interface support the multi-positioning (if more than one tank is specified), the backwards and forward searches within segments, and merges among segments required to do this. Presently, we will also add variable names for derived variables (e.g. growth rate and conversion ratio), and the action routines to compute them from database entries at 2 or more times, checking that these times are within a "growth set". The LIST command within QUERY will then produce information in every way comparable to the LIST command within the multi-tank simulator, but with allowance for the additional complications of real data. It will then be a powerful investigative tool for monitoring growth performance, and extracting the data needed to fit and validate the various auxiliary equations of the models.

5. INTERACTION BETWEEN DATABASE AND SIMULATORS

Despite the already proven value of the database and simulation systems as independent entities, neither system is adequate, by itself, for developing optimal management procedures for fish farms. No amount of actual data, or generated reports, can predict the effect of untried strategies, while the simulators' predictions are of little value unless the growth models and auxiliary equations are fitted to, and validated against, real data. There are two main applications where the simulators must access the database. First, it is desireable to SAVE simulated interventions and results (censuses, etc.) to the database; and secondly, one would like to be able to extract real initial census and rearing conditions from the database, and then use subsequent real interventions to drive the simulators. We will describe what can be achieved by these applications, and how we are going about achieving them. As a preliminary, it should be pointed out that such applications would be very much easier to implement if simulated and real data were identical in form and quality. They are not, and this creates formidable difficulties that must be resolved by compromises from both ends: by altering the applications programs to make them cope with less than complete (real) data, and by developing more stringent protocols on the way actual data is collected to ensure it is more complete.

5.1 Saving Simulation Results to the Database

There are considerable advantages to saving a complete record of the results of a simulation run to the database, above and beyond what can be saved without using the database. Even without the database, a session can be saved and/or regenerated. This is because the multi-tank programs maintain an echo and a prompts file. The echo file lists every user input and system response and can be directed to the printer at the end of a session to preserve a hardcopy of the complete session at the terminal. The prompts file contains a list of every prompt response and command entered by the user. It can be used to regenerate the session automatically until the end-of-file is hit, at which point input is sought from the terminal. The user can then resume the session where he left off in the earlier session when the prompts file was saved. (This is particularly useful when the session terminates abnormally; the user simply edits out the last few lines of the prompts file and then uses it to regenerate the session up to the point where the error occured.) Variations on a past strategy can be explored by resuming the session in this way and using BACKUP to locate to a past period and re-RUNning from that point. As we pointed out in Section 3.3, this obliterates the previous simulation from that point on and no direct comparisons can be made between the variations. Saves to the database do not have this consequence (within limits) so that one can bring to bear the full power of the database reporting system to produce comparative reports, although this must be done in a separate batch report or QUERY session. Second, saves to the database are not in the form of a sequence of prompts, commands and system responses (as are the echo and prompts files), but are formatted into structures, passed to the database interface, and stored as database segments whose structure and organisation are (virtually) indistinguishable from real data. Thus comparisons between real and simulated data are also possible. Some of the database and interface design features that make this possible will now be discussed.

One of the design objectives of the database was to achieve non-redundant storage of real and simulated data, where these might share common movement and event segments (i.e., for interventions) but possibly different census segments and event segments for environmental variables (i.e., system responses). This has obvious applications for validating simulated runs against equivalent real runs. Similarly one might wish to store variations on a basic simulation strategy, where these might share common intervention and response segments up to some period and then diverge. Two features help to achieve this. The first is that the simulators share with the database, the species, tank-type and system configuration files. This ensures that whenever two or more real and/or simulated segments refer to the same entry in any one of these files, they are guaranteed to provide the same information. The database interface handles these references automatically without the applications programmer needing to know anything about how they are stored. Secondly, each segment (except the GROUP and TANK segments - see Fig. 1) contains a 4-byte simulation mask enabling the interface to determine which simulations the segments belong to. This mask field has 32 bits, the rightmost indicating whether or not the segment represents real data, the remaining 31 bits representing simulations (variations) 1 to 31.

Now, when in response to a prompt, the user indicates he wishes to SAVE to the database, he is asked if he wishes to create a new GROUP segment.

If he does, he supplies the group and tank information and the simulation number is set to 1. Multi-tank simulation now proceeds exactly as described in Section 3.3, but all interventions and results are stored out to the database as segments. We also allow full use of BACKUP and ADVANCE. If at any point, however, he does a RUN from a period other than the last, the simulation number is incremented by 1. The past history segments shared with the previous simulation number are indicated by simply flipping to ON the simulation bit corresponding to the new simulation number for all segments of the old simulation up to that period. This is handled automatically by the interface. From this period on, the two simulations will have distinct segments, even if some of these happen to coincide. Groups created in this way have no real data, except possibly for real species, tank-types and system configurations.

A FETCH command permits the user to locate to an existing group in the database, automatically positioning him to the last period of the highest simulation number. Comparison reports of simulation variations and reversion to lower simulation numbers cannot be done from the simulate programs. Implementation of the features described to this point are near completion. Those that follow are just being begun.

5.2 Using Real Data to Drive the Simulator.

The structure of the database and the properties of the interface make this a fairly straightforward application, but problems arise from inadequacies in the real data. In principal the procedure will work as follows. The user is prompted for the name of an existing group containing real data. If it exists and has real data in it (the group segment contains a single bit flag for indicating this without the need to check for segments with the simulation bit 0 turned on), the user is prompted with the time of the first and last census and asked to choose a starting time between the two. The system must then search back from that time to find the preceding census, and the relevant conditions and system settings that pertain at those times, depending on the simulation level in force. For example, if the Basic model is in use, only the species-in-tank, tank-temperature, feeding regimen, and census segments are needed to establish initial conditions. The tank segments are needed only to inform the user of the number of tanks and their names. If the simulation is at the Tank Effects or Tank Constraints level, the oxygen and metabolite concentrations, and flow rate conditions must be determined, as well as each tank-type. The database interface is now set up to handle this backward search, even though IMS, which is used to structure the database, does not support backward processing. The structuring of the database, by date-time order within tanks for every segment type also facilitates this, and the extraction of interventions.

Having established initial conditions, the system then determines which segments are relevant interventions at this level of modelling (e.g., feeding, temperature and movements for the Basic model) and gets the next segment of each of these for every tank. The interface is set up to handle this with very little coding on the part of the applications programmer. The first of these interventions is presented to the user and he has the chance to modify it before implementing it. Implementation involves doing a RUN up to the time of the next intervention (if this time is not negligible), carrying out the intervention (e.g., a MOVE or SET), and storing out the relevant simulated response variable segments (e.g., CENSUS, EVENT segments of Fig. 1). This process is then repeated for the next intervention. Real tank censuses would also be considered as interventions for this purpose so that one could compare each one with the corresponding simulated census as one went along. However, more complete comparisons would be done using the reporting features of the database system (Section 4.3). When the user terminates this process, or comes to the end of the group segment data, he can either continue simulating (to plan future strategies or alternative strategies after the termination point), or use BACKUP to relocate back to an earlier time, and then explore (as described in the previous subsection) for alternatives that might have improved on the real strategy. This provides a local optimisation feature by means of gaming.

The difficulties in doing this are not inherent in the mechanics of accessing the database or driving the simulator, but in the inadequacies and ambiguities of the real data. Extensive checks must be done to ensure that the real data are complete enough to provide all the initial conditions and interventions required at the desired level of simulation. As we mentioned in Section 4.1, not all census and movement segments give reasonably complete weight distribution information. Those that don't will have to be handled differently from those that do; the former by additional prompting or extrapolation, the latter by massaging to reconstruct a distribution over the standard background weightclasses of the simulator. To prevent having to do this repeatedly, the census segment has been given a flag to indicate this has been done and a child segment (CENDAT in Fig. 1) for storing the count vector. Derivation of the vector requires the use of (length-weight relationship) parameters which are currently stored in the species file. However, these parameters are not really invariant even within strains, and may have to be refitted from the census data within each group. There are also inconsistencies between real and simulated time that will have to be resolved. Simulated interventions can be carried out instantaneously, but real interventions are spaced out in time. Often the recorded time intervals are more notional than real, intended to preserve the actual ordering of events rather than their actual time of occurrance. This creates problems in determining whether, for example, given census and rearing condition segments for the same tank are close enough in time to be considered contemporaneous. Such problems can only be resolved by refining the data gathering and encoding protocols.

6. CONCLUSIONS

The validity of this system as a management tool depends strongly on the accuracy with which the growth equations and the auxiliary equations (for environmental effects and biological constraints)

reflect reality. For that reason we have put considerable facilities at the disposal of the manager for examining his real data and making comparisons between comparable simulated and real outcomes. With use, the manager will soon get a feel for the reliability of the models' predictions. We have only just put in place most of the database and database/simulator interaction facilities, so it is a bit too soon to report on the adequacy of our models and calibrations. This will be the main focus of our work over the next year, and can proceed more quickly given this system support. Nevertheless, the system becomes useable even at this stage because of the implementation of successively higher levels. The manager can use his experience, and the feedback from the database, to confine his simulations to subsets of the rearing conditions or to lower levels of models (which use fewer auxiliary equations) that appear to produce valid results. For example, before the growth model was modified to reflect optimal temperature, simulations at above 14 or 15 degrees gave unrealistically high growth rates. The Basic model was still useable and useful, but only for lower temperature simulations. Similarly, if the Tank Constraints model seems to give unrealistic or restrictive biological constraints, one can revert to the Tank Effects model until one's experience with the model, and the refinement of the model itself, increases.

One serious and unresolved limitation in the use of this system for managing real commercial hatcheries is the possible need to recalibrate for every different food type. The variation between and within commercially available foods is the subject of current research at Rockwood. The degree to which differences might affect growth rate and fish physiology (e.g. ammonia excretion rates) is being evaluated. Should it be necessary, our system makes it very much easier to recalibrate, but there is still a considerable cost in time and effort to do so.

Finally, the system has potential for a number of applications not yet discussed in this paper. Once a cost level model is implemented, and the database is modified to record costs (and benefits from sales), one can begin to think about forming an objective function and using dynamic programming methods for formal optimisation. Sparre (1976) has shown that while global optimisation is not feasible, local optimisation about a promising initial strategy can lead to significantly improved production strategies. Our system is structured in a way that would easily permit extension to include such optimising capabilities.

ACKNOWLEDGMENTS

This work has been supported for two years by a subvention grant from the Canada Department of Fisheries and Oceans. Algas Resources Ltd. has provided reseach funds for the biological research associated with system development. The Natural Sciences and Engineering Research Council has provided operating grants to A.N.A. and D.H.S.

We wish to acknowledge the considerable contribution to the system development made by some unusually bright and productive students: Robert Day for work on the calibration and tank effects programs, Bruce Jones for general project co-ordination and development of the database QUERY system, Mike Karasick for the parser/scanner system, Gary Karasiuk for developing the database creation, interface and batch reporting programs, Norm Kozusko for the SAVE/FETCH routines integrating simulators and database, and Gerard Meszaros for extensions to the simulator programs.

This system could not have been developed without the constant help and guidance of the people in the FWI Aquaculture project and the Rockwood Hatchery. Of these, special thanks are due to Burton Ayles who initiated and guided the project, and most of all, to Michael Papst, who gave generously of his time and extensive knowledge of fish and hatchery behaviour. We thank Irmgard Wiebe for help with text entry of this paper.

REFERENCES

Arnason A.N., C.J. Schwarz and G.G. Meszaros (1981), FISHDAMS: A database management system and growth simulator for optimal planning of a commercial-type fish rearing facility, Version 2, Technical Report, Computer Science Department, University of Manitoba, Winnipeg, Canada, September, 70 p.

Karasick, M. (1981), EASYPARSE: A generalized, easy-to-use parser interface for user-oriented systems, Technical Report, Computer Science Department, University of Manitoba, Winnipeg, Canada, in prep.

Kiviat, P.J., R. Villanueva and H.M. Markowitz (1975), SIMSCRIPT II.5 Programming Language, C.A.C.I., Los Angeles, 384 p.

McLean, W.E. (1979), A Rearing Model for Salmonids, Ph.D. Thesis, University of British Columbia, Vancouver, 134 p.

Paulson L.J. (1980), Models of Ammonia Excretion for Brook Trout (Salvelinus fontinalis) and Rainbow Trout(Salmo gairdneri), Can.J. Fish. Aquat. Sci., Vol. 37, pp. 1421-1425.

Scuse, D.H. (1981), Database Interfaces, submitted to: Australian Computer Journal.

Sparre, P. (1976), A Markovian Decision Process Applied to Optimization of Production Planning in Fish Farming, Medd. Danm. Fisk-og Havunders., Vol. 7, pp. 111-197.

Sperber, O., J. From and P. Sparre (1976), A Method to Estimate the Growth Rate of Fishes as a Function of Temperature and Feeding Level Applied to Rainbow Trout, Medd. Danm. Fisk-og Havunders., Vol. 7, pp. 275-317.

Thorpe, J.E. (1977), Bimodal Distribution of Length of Juvenile Atlantic Salmon (Salmo salar) under Artificial Rearing Conditions, J. Fish Biol., Vol. 11, pp. 175-184.