Summer 8-6-2013

Oyster Sustainability Modeling as a Public Resource

Nathan A. Cooper
University of New Orleans, ncooper@uno.edu

Oyster Sustainability Modeling as a Public Resource

A Thesis

Submitted to the Graduate Facility of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Nathan Cooper

B.S. University of New Orleans, 2011

August, 2013

Acknowledgements

I would first like to thank the Louisiana Sea Grant program, the Louisiana Department of Wildlife and Fisheries, the National Fish and Wildlife Foundation, and The University of New Orleans for funding this project.

I would like to express my extreme gratitude to my advisor Dr. Mahdi Abdelguerfi, who has provided me with so many opportunities throughout my academic career. My deepest thanks goes out to Dr. Thomas Soniat for his friendly advice and expertise each day, helping a programmer to better understand the world of oysters. Also I would like to thank Dr. Shengru Tu, my advisor and coordinator, for his cheerful and always-helpful guidance. Of course, thanks to Susan Colley Theodosiou for her never-ending diligence and hard work.

A shout-out goes to my former brothers-in-code, Janak Dahal and Christian Chuindja Ngniah, who helped build this project from the ground up. And to Josh Gallegos, whose database expertise provided invaluable assistance. I know I'll be leaving the site in good hands.

I'd like to show my appreciation for everyone who helped us with the project. Thanks to Fareed Qaddoura, John Finigan, Jeanne Boudreaux, and all my wonderful professors from The University of New Orleans for their dedication, knowledge, and friendliness. Thanks to Patrick Banks, Mark Schexnayder, and everyone else at LDWF who were all a great pleasure to work with. And thanks to John Klinck, Eric Powell, Roger Mann, and all the other researchers involved, without whom this project would not be possible.

Finally, my warm thanks goes out to my mother Lisa Cooper, grandmother Frances Gregg, and the rest of my friends and family, who provided me with love and moral support throughout my studies. A special nod goes out to Christine Carey for her valuable insights and encouragement.

Thanks everyone!

Table of Contents

List of Tables

Tables

List of Figures

Abstract

A simulation algorithm based on biological references points proposed by Powell and Klink (2007) is implemented for predicting the total allowable catch of eastern oysters (*Crassostrea virginica*) from Louisiana's coast. The model accepts initial per-square-meter shell mass and oyster size distributions as input. Fishing effort is provided as fractions removed of each resource for each month of the season. The model outputs the expected remaining shell mass and harvests of sack and seed oysters after t discrete fishing months. Oyster mortality credits the shell budget, while fishing fractions debit oyster and shell resources. Surviving oysters grow larger along a time-dependent von Bertalanffy growth curve. Fishing fractions are chosen heuristically with the goal of minimizing shell loss. Input data is collected by the Louisiana Department of Wildlife and Fisheries in their annual stock assessment. The model is available as a public web resource at *www.oystersentinel.org*.

Oysters, *Crassostrea virginica*, Modeling, Information Technology, Simulation, Fishery

Oyster Sustainability Modeling as a Public Resource

1. Introduction

The abundance of the eastern oyster *Crassostrea virginica* in Louisiana's estuaries supports a thriving fishing industry. About 1.7 million acres constitute the public water bottoms where sack oysters (≥ 75 mm) are directly marketed and seed oysters (< 75 mm) removed for transplant onto private leases. The Louisiana Department of Wildlife and Fisheries (LDWF) is responsible for managing these grounds. However, they currently have no quantitative reference points for harvest sustainability (LDWF, 2010).

Powell and Klink (2007) proposed biological reference points based on the abundance of either oyster numbers (N) or substrate mass (S). In this model, the population and/or shell quantity remains constant with respect to time. That is,

$$\frac{dN}{dt} = 0 \qquad \text{Eq. 1}$$

or

$$\frac{dS}{dt} = 0 \qquad \text{Eq. 2}$$

Mann and Powell (2007) argue that the primary management goal is better captured by Eq. 2, as shell is more economical to conserve than to replace. Also, oyster populations are most likely more limited by the substrate upon which larvae attach and adults grow than by the number of oysters needed to reproduce and replace the population.

Powell, Klinck, and Soniat (2010) proposed a one-year forward prediction model in which the shell impact of a given fishing scenario is simulated, using local population data in conjunction with existing modeling techniques to arrive at quantitative estimations of shell loss. The model accepts initial per-square-meter shell mass and oyster size distributions as input. Fishing effort is provided as fractions removed of each resource for each month of the season. The model outputs the expected remaining shell mass and harvests of sack and seed oysters after t discrete fishing months. Oyster mortality credits the shell budget, while fishing fractions debit oyster and shell resources. Surviving oysters grow larger along a time-dependent von Bertalanffy growth curve. Fishing fractions are chosen heuristically with the goal of minimizing shell loss.

This thesis presents a formalization of the algorithm developed by Klinck, Powell, and Soniat in the original model (2011). It begins with a high-level description of the data aggregation techniques used as input for the model, followed by an abstract treatment of the model logic. It then describes the software techniques the author used to restructure and improve the model, making it a publically-accessible resource which draws its input from an online database. Finally, it describes some of the preliminary results obtained from its simulations and its goals as a management tool for the oyster industry.

2. Input Data

LDWF Stock Assessment

The LDWF provides data appropriate for model input in the form of an annual stock assessment. The LDWF conducts this assessment every summer across multiple Coastal Study Areas (CSA's). Before 2012, there were seven such individually-managed CSA's constituting the Louisiana public oyster grounds, depicted in the map below (Figure 1). In 2012, these were consolidated into five stations.

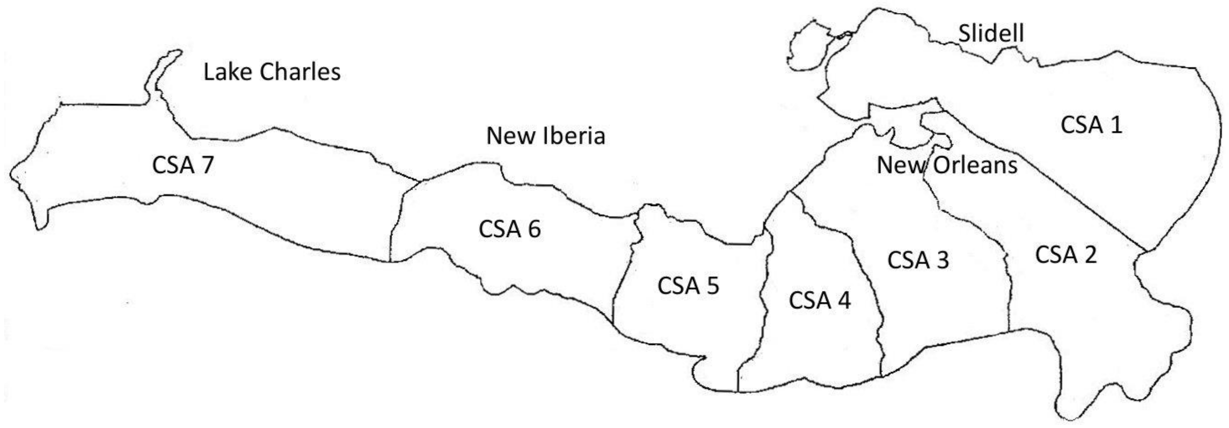
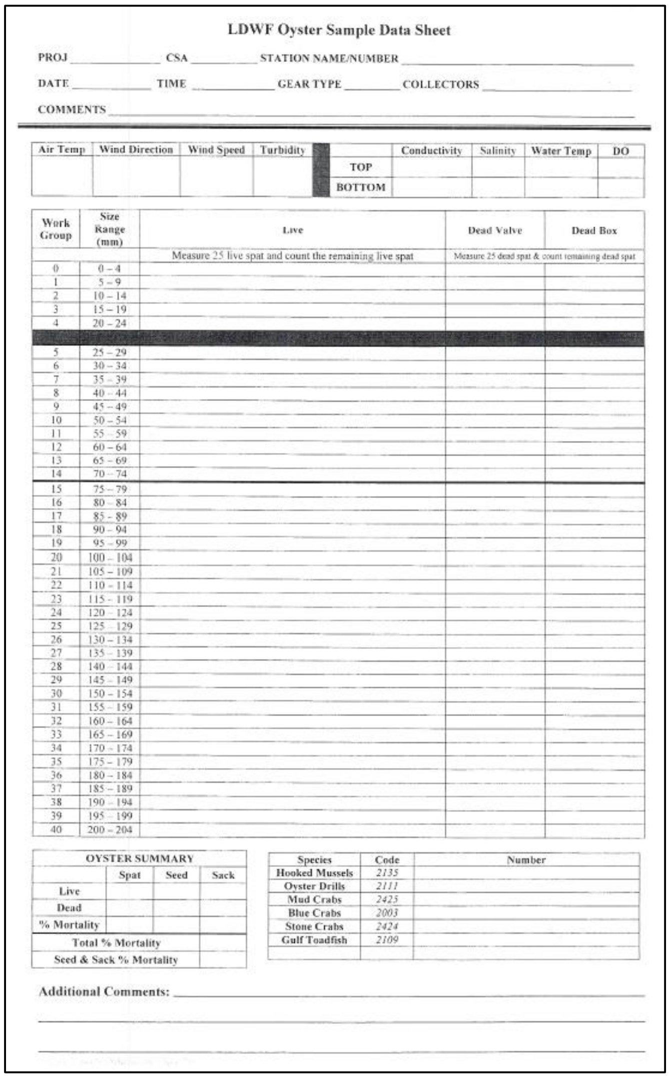


Figure 1. LWDF Louisiana Coastal Study Areas (CSA's).

Within each CSA are multiple stations, ranging in area from less than 10 acres to several hundred acres. During the stock assessment, each station is sampled at various replicate locations. Divers deploy 0.25 m^2 or 1.0 m^2 grids on the reef and remove oysters and cultch. Live and dead oysters are recorded separately and are measured based on the longest length across the shell with 1-mm precision. The oysters from each replicate are tallied on an official Sample Data Sheet (Figure 2). Each oyster in the replicate is counted and categorized into one of 41 different size classes, or *work groups*. Each size class spans 5 mm, allowing oysters from 0 to 204 mm to be recorded. Additional environmental conditions for each replicate (air temperature, wind direction, wind speed, turbidity, etc.) are recorded, along with a description of the cultch quality and counts of various non-oyster species encountered on the reef. Data in this format is available at least as far back as 1999, allowing a high size-resolution time series to be available as input for the model (LDWF 2010). As of the 2012 stock assessment, the cultch, or material constituting the reef bottom, is separated by type and weighted with 1-gram precision.



LDWF Oyster Sample Data Sheet

PROJ ____________ CSA __________ STATION NAME/NUMBER ______________________________

DATE ____________ TIME ____________ GEAR TYPE ________ COLLECTORS ____________________

COMMENTS ____

Air Temp	Wind Direction	Wind Speed	Turbidity		Conductivity	Salinity	Water Temp	DO
				TOP				
				BOTTOM				

Work Group	Size Range (mm)	Live	Dead Valve	Dead Box
		Measure 25 live spat and count the remaining live spat	Measure 25 dead spat & count remaining dead spat	
0	0 – 4			
1	5 – 9			
2	10 – 14			
3	15 – 19			
4	20 – 24			
5	25 – 29			
6	30 – 34			
7	35 – 39			
8	40 – 44			
9	45 – 49			
10	50 – 54			
11	55 – 59			
12	60 – 64			
13	65 – 69			
14	70 – 74			
15	75 – 79			
16	80 – 84			
17	85 - 89			
18	90 – 94			
19	95 – 99			
20	100 – 104			
21	105 – 109			
22	110 – 114			
23	115 – 119			
24	120 – 124			
25	125 – 129			
26	130 – 134			
27	135 – 139			
28	140 – 144			
29	145 – 149			
30	150 – 154			
31	155 – 159			
32	160 – 164			
33	165 – 169			
34	170 – 174			
35	175 – 179			
36	180 – 184			
37	185 – 189			
38	190 – 194			
39	195 – 199			
40	200 – 204			

OYSTER SUMMARY			
	Spat	Seed	Sack
Live			
Dead			
% Mortality			
Total % Mortality			
Seed & Sack % Mortality			

Species	Code	Number
Hooked Mussels	*2135*	
Oyster Drills	*2111*	
Mud Crabs	*2425*	
Blue Crabs	*2003*	
Stone Crabs	*2424*	
Gulf Toadfish	*2109*	

Additional Comments: ____

Figure 2. LDWF Oyster Sample Data Sheet.

The model aims to use this data by applying the no-shell-loss criterion (Mann & Powell, 2007). A wide range of simulation possibilities exist by focusing on various combinations of stations within a particular year. For example, simulating a single station can be used for narrow-scope area management, whereas simulating every station in a CSA together can be used to inform management about the entire region.

Oyster Data

For each included station, the model expects as input a size-to-count mapping for each populated work group. Replicate work group data must be averaged on a per-station basis in order to construct an abstract representative sample for each station included in the analysis. To accomplish this, a simple arithmetic mean is taken. More replicates per station generally imply more accurate results.

Consider the fictitious "Station 1" as an example. Station 1 is has a total area of 250 acres. Five replicate samples, A - E, were surveyed at various locations, three on 0.25 m^2 grids, and two on 1.0 m^2 grids,[1] as depicted in the table below (Table 1).

Table 1. Example replicate gear sizes.

Replicate	Gear Size (m^2)
A	0.25
B	1.00
C	0.25
D	1.00
E	0.25

Oyster data was sampled at each of the five replicates. This replicate data is shown on the following table (Table 2), with an empty entry indicating no oysters were found for that particular size group.

Table 2. Example work group data.

#	Group (mm)	Mid (mm)	A	B	C	D	E
0	0 - 4	2		2	1		5
1	5 - 9	7	2	1	4		1
2	10 - 14	12	1				2
3	15 - 19	17	1				
4	20 - 24	22	5				
5	25 - 29	27	2	1			1
6	30 - 34	32	1	1	1		2

[1] Grid sizes smaller than 1 m^2 are only used by LDWF for cultch plants with dense oyster populations to lessen the sampling effort. The example here uses assorted grid sizes to prevent loss of generality.

Assume only the size groups listed are under consideration. Note that for replicate D, the survey was conducted, but no oysters were found.

First, a live count of each work group must be computed for each station. A simple series of steps are performed to obtain this. Assume the model is processing a set S of included stations. For each station $s \in S$, let the total station area be A_s and let R_s be a collection of replicates in s. For each replicate $r \in R_s$, let a_r denote the gear area of the sample and let $n_{g,r}$ denote the live number of oysters in any work group $g \in G$. Then the station's estimated group-g oyster density, $O_{g,s}$ is given by the following formula (Eq. 3).

$$O_{g,s} = \frac{1}{|R_s|} \sum_{r \in R_s} \frac{n_{g,r}}{a_r} \qquad \text{Eq. 3}$$

Next, this density is used to compute the total estimated oyster count across the entire station, $N_{g,s}$, given by the formula below (Eq. 4), where K is a simple conversion from acres to square meters (approximately 4046).

$$N_{g,s} = O_{g,s} \times A_s \times K \qquad \text{Eq. 4}$$

Applying the last two equations for Station 1 yields the following results (Table 3).

Table 3. Example total predicted oysters.

g	Group (mm)	$O_{g,1}$	$N_{g,1}$
0	0 – 4	5.2	5,260,910
1	5 – 9	5.8	5,867,940
2	10 – 14	2.4	2,428,110
3	15 – 19	0.8	809,371
4	20 – 24	4.0	4,046,860
5	25 – 29	2.6	2,630,460
6	30 – 34	3.4	3,439,830

The length of oysters in each work group is taken to be the mid-value of the size range of that group (i.e. 0 – 4 mm oysters are considered 2 mm). It is this mapping of size and station to total oysters that provides a complete estimate of the oyster population for the model.

The older version of the model could not treat stations individually. Thus, a further calculation had to be made during the preprocessing stage whereby the included stations were combined into one. In this scenario, the total estimated oysters across all stations in $s \in S$, denoted N_g is given in the following formula (Eq. 5). However, in the current version of the model, this step is not necessary.

$$N_g = \sum_{s \in S} N_{g,s} \qquad \text{Eq. 5}$$

Cultch Data

Cultch, or the non-living reef environment of the oyster habitat, is a critical input parameter of the model. It is the manipulation of this resource that determines the fishing sustainability of the simulated region.

Prior to 2012, LDWF did not take quantitative sample measurements of shell or cultch, making it impossible to use station-specific values for the model's required input. In such cases, this value was roughly approximated by a single best-estimate density (g/m^2) value. However, during the 2012 stock assessment, detailed measurements of cultch were made, a policy which is expected to continue for future assessments. To accommodate the additional information, the author created a revised LDWF Oyster Sample Data Sheet.

For each replicate, surveyed cultch is separated by type and weighted with 1-gram precision. Along with the brown shell from dead oysters, measurements are recorded for limestone (frequently planted as artificial shell to improve reef conditions), concrete, mussel shell, and any other additional cultch types found. While the original model prototype did not distinguish between types of cultch, the improvements introduced by the author allow cultch types to be individually specified as input. However, the natural loss (dissolution and biodegradation) rates of each type of cultch is largely unknown and is the subject of ongoing investigations.

Consider the following data from the Station 1 example in the previous section (Table 4). Cultch types were measured for each replicate below and measured in grams. Note that a blank entry indicates that no cultch survey for that type was conducted, whereas a zero value indicates that the survey was made, but no cultch of that type was found. Here, replicate E did not perform *any* cultch measurements and replicate D failed to measure mussel weights.

Table 4. Example cultch data.

Cultch Types		**Mass (g)**				
#	**Type Name**	**A**	**B**	**C**	**D**	**E**
0	Brown Oyster Shell	1076	1,412	1045	4006	
1	Muddy Oyster Shell	301	2013	623	1099	
2	Brown Limestone	1,002	0	985	4012	
3	Mussels	0	205	0		

Assume the model is processing a set S of included stations and that we are interested in each type of cultch from a set T. For each station s, suppose $\delta_{t,s}$ is an estimate of the type-t shell density (in g/m^2) for that area. Let the total station area be A_s and let R_s be a collection of replicates in s. For each replicate $r \in R_s$, let a_r denote the gear area of the sample and $d_{t,r}$ be the amount of cultch of type t measured in grams, or undefined if no sample was made. Finally, let $w_{t,r}$ be defined by the below equation (Eq. 6).

$$w_{t,r} = \begin{cases} \frac{d_{t,r}}{a_r} if\ d_{t,r}\ is\ defined \\ \delta_{t,s}\ otherwise \end{cases} \quad \text{Eq. 6}$$

Thus, for the sample data provided above, all of replicate E and the *mussel* cultch type of replicate D would be estimated from the respective $\delta_{t,s}$ values.

With a density estimate provided for each replicate, the average is computed next. The station's estimated type-t cultch density, $C_{t,s}$ is given by the following formula (Eq. 7).

$$C_{t,s} = \frac{1}{|R_s|} \sum_{r \in R} w_{t,r} \qquad \text{Eq. 7}$$

Similarly to oysters, this density is used to compute the total estimated cultch mass across the entire station, $M_{t,s}$, given by the following formula (Eq. 8). It is this mapping of cultch type and station to cultch mass that is presented as input to the model.

$$M_{t,s} = C_{t,s} \times A_s \times K \qquad \text{Eq. 8}$$

The results of applying this logic to the data above is shown below (Table 5), with each $\delta_{t,s}$ value specified respectively.

Table 5. Total estimated cultch

t	Type Name	$\boldsymbol{\delta_{t,1}}$	$\boldsymbol{C_{t,s}}$	$\boldsymbol{M_{t,s}}$
0	Brown Oyster Shell	5,000	3,780.4	3,824,687,386
1	Muddy Oyster Shell	2,000	1,761.6	1,782,237,144
2	Brown Limestone	3,000	2992	3,027,051,280
3	Mussels	1,000	441	446,166,315

As was the case with oyster work groups, the older version of the model would require the station information to be collapsed out, namely by Eq. 9 below.

$$M_t = \sum_{s \in S} M_{t,s} \qquad \text{Eq. 9}$$

3. Model Logic

Introduction

The data sources from the previous section provide the necessary input to the model. With this information computed, the simulation can begin execution. The descriptions in this section are based on the author's analysis of the model documentation (Powell, Klinck & Soniat, 2011) and original Fortran program (Klinck, Powell, & Soniat 2011).

The diagram below (Figure 3) illustrates the basic logic of the simulation.

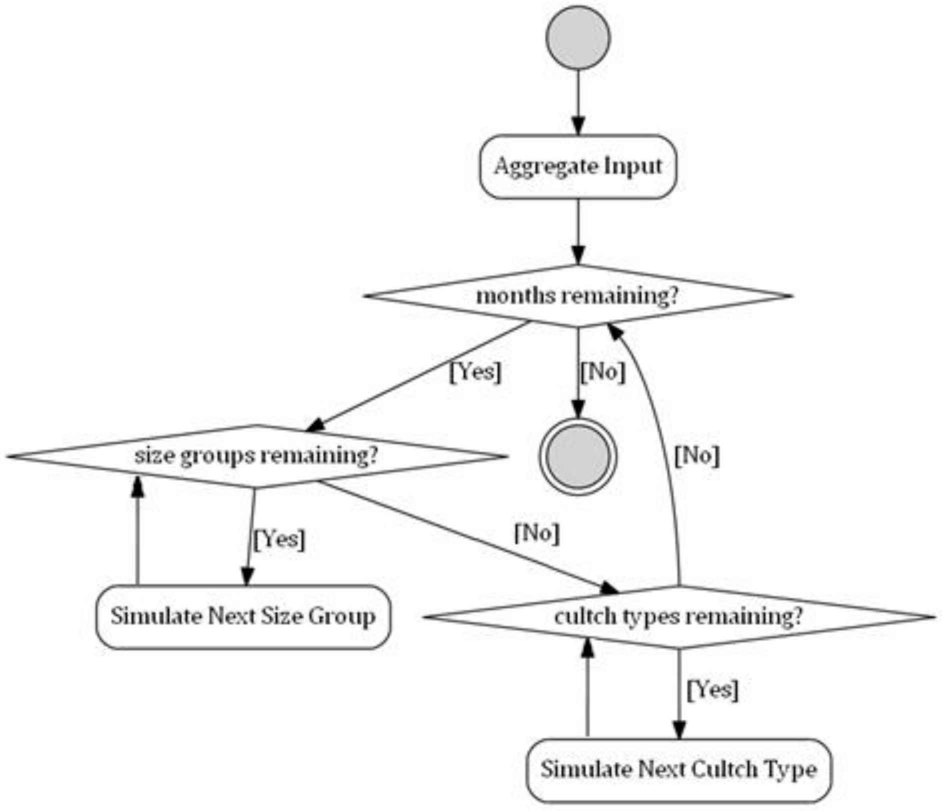


Figure 3. A flow chart of the model's high-level logic.

Note that the logic in this section applies to the original procedural versions of the model. The newer version, implemented as database functions, applies the same concepts, but operates on tables. This section will describe the pseudocode procedurally, while the SQL code itself is presented in the Implementation section.

The primary simulation routine consists of a high-level time loop, iterating for each month of the simulation. Typically, this means a 12-month forward prediction between stock assessments. Within each step, each oyster size group and each cultch type is simulated. Oysters of each size group are removed due to a time-dependent natural mortality rate. Additionally, a monthly fishing fraction is applied that removes oysters from each group. The sizes for each oyster are then increased along a time-dependent von Bertalanffy curve to simulate seasonal growth. The shell mass equivalent from the oysters dead from natural mortality during that month are credited to a specially-designated cultch type (usually *brown oyster shell*). Then for all cultch types, a monthly dissolution factor and a fractional fishing value debit the mass.

This process repeats until the end of the simulation. A running total of shell and oysters fished is kept updated for each iteration. By examining these results, the operator can fine-tune fishing rates to optimize for a desired termination condition (i.e. minimized loss of shell). Each stage in this process and the associated variables is discussed below.

Oyster Component

Oyster Mortality

Natural oyster mortality is simulated using a mortality fraction that generalizes oysters lost due to natural causes such as disease and predation. The associated mortality rate varies per month (tracking annual variation in temperate) using a trigonometric model. For month t, if m_{avg} is the month of average mortality, then this mortality rate $m(t)$ can be calculated for adult oysters ($\geq$ 25 mm) as:

$$m(t) = m_{0,A} + m_{1,A} \times \sin\left(2\pi \times \frac{t - m_{avg}}{12}\right) \qquad \text{Eq. 10}$$

and for juvenile oysters ($<$ 25 mm) as:

$$m(t) = m_{0,J} + m_{1,A} \times \sin\left(2\pi \times \frac{t - t_{avg}}{12}\right) \qquad \text{Eq. 11}$$

Here, $m_{0,A}$ and $m_{0,J}$ represent the mean annual mortality rate for adults and juveniles, respectively, while $m_{1,A}$ and $m_{1,J}$ represent the annual mortality rate maximum deviation for adults and juveniles, respectively. To calculate the number dead monthly, the resulting value for m can be substituted in an exponential decay equation, which is multiplied by *N*, the number of oysters in the work group, to obtain N_{dead}, the number of oysters dead for the given month, as shown by the equation below (Eq. 12).

$$N_{dead} = N \times (1 - \mathrm{e}^{-(\mathrm{m(t)}/12)}) \qquad \text{Eq. 12}$$

The mortality fraction using some representative values (which assumes juvenile mortality rates are higher than those of adult oysters) is depicted as a function of time in the graph below (Figure 4).

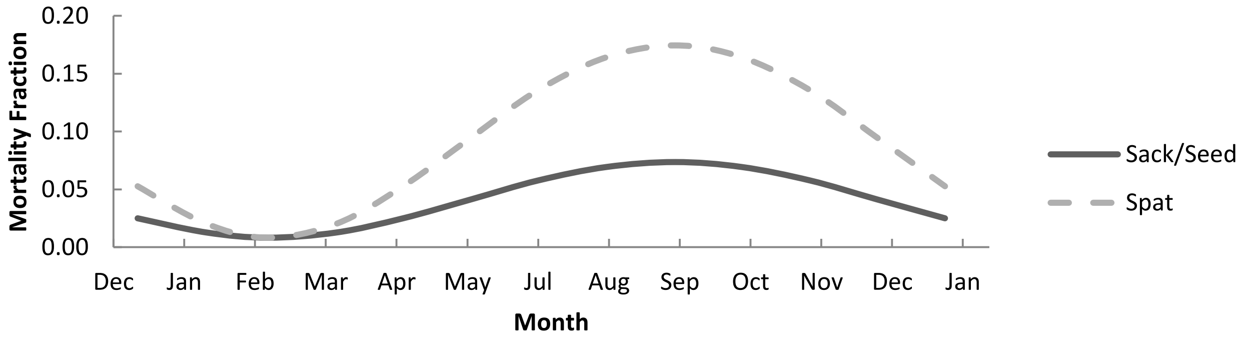


Figure 4. Typical mortality fractions for adult and juvenile oysters.

$\boldsymbol{r_{0,A}}$: 0.51, $\boldsymbol{r_{1,A}}$: 0.41, $\boldsymbol{r_{0,J}}$: 1.2, $\boldsymbol{r_{1,J}}$: 1.1, $\boldsymbol{t_{avg}}$: 6

Note that while the graph above (as well as other graphs in this section) is continuous, the model applies this fraction per-month in discretized units of time at the beginning of each month.

Oyster Growth

Oysters in the simulation grow steadily along a von Bertalanffy growth curve. The standard von Bertalanffy function allows animals to approach an asymptotic maximum length λ_∞ at annual rate k, given the age a_0 (in years) when length is zero. Specifically, the length λ of animals of age a (in months) is given by the equation below (Eq. 13).

$$\lambda(a) = \lambda_\infty \times \left(1 - \mathrm{e}^{-k\left(\frac{a}{12} - a_0\right)}\right) \qquad \text{Eq. 13}$$

The figure below illustrates such an age approximation for oysters (Figure 5).

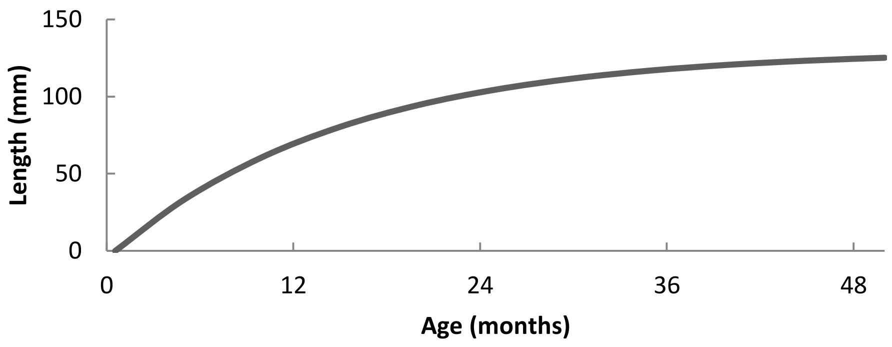


Figure 5. A typical von Bertalanffy growth curve.

$\boldsymbol{L_\infty}$: 130, $\boldsymbol{a_0}$: 0.045, $\boldsymbol{k}$: 0.8

A simple adjustment is made to this formula in the model to accommodate the slower growth oysters experience during winter months. The growth rate k is computed trigonometrically as a function of time. Thus, given a mean annual growth rate k_0, a growth rate maximum deviation k_1, and a time (as a fraction of a year) of average growth rate t_{avg}, the growth rate k at time t (in months) is expressed by the equation below (Eq. 14).

$$k(t) = k_0 + k_1 \times \sin\left(2\pi \times \left(\frac{t}{12} - t_{avg}\right)\right) \qquad \text{Eq. 14}$$

The figure below (Figure 6) illustrates how the growth rate k fluctuates with respect to time.

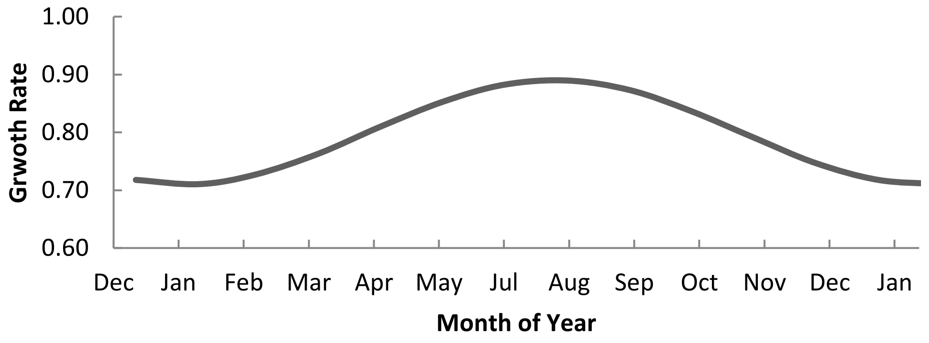


Figure 6. Typical values for k as a function of time over one year.

$\boldsymbol{k_0}$**:** 0.8, $\boldsymbol{k_1}$**:** 0.09, $\boldsymbol{t_{avg}}$**:** 0.4

The model transforms each work group's length by one month's growth increment at the end of every month. Given an oyster during month t with length λ_t, the growth rate $k(t)$, and a maximum length λ_∞, the new length after one month of growth λ_{t+1} can be calculated by the below equation (Eq. 15).

$$\lambda_{t+1} = \lambda_\infty - (\lambda_\infty - \lambda_t) \times \mathrm{e}^{(-k(t)/12)} \qquad \text{Eq. 15}$$

The graph below (Figure 7) illustrates a change in simulated oyster growth when the above formula is applied per-month.

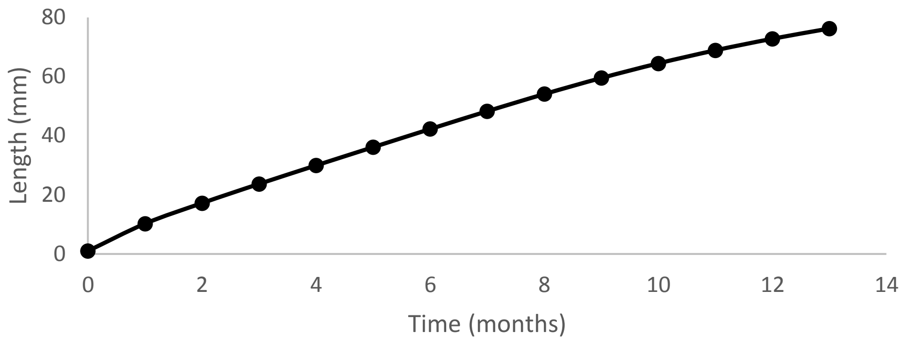


Figure 7. Typical simulated oyster growth starting in August at 1 mm.

$\boldsymbol{L_\infty}$: 130, $\boldsymbol{k_0}$: 0.8, $\boldsymbol{k_1}$: 0.09, $\boldsymbol{t_{avg}}$: 0.4

Sack Conversion

Oyster harvests are typically measured in sacks, 1.5-bushel quantities of oysters. In order to determine this amount, oyster length must be converted to volume. A simple power fit is used to determine the sack volume obtained from an oyster of a particular size. Given constants OPS_a and OPS_b, the number of sacks V of volume v (in liters) of *N* oysters of length λ can be approximated by the following formula (Eq. 16).

$$V(\lambda) = N \times \frac{52.85}{OPS_a \times \lambda^{OPS_b} \times v} \qquad \text{Eq. 16}$$

The result of an oysters-per-sack conversion model is scaled by the number of oysters present and volume of the sacks used in order to achieve this generalization. The graph below (Figure 8) illustrates the similar relationship between oyster length and the corresponding number per sack.

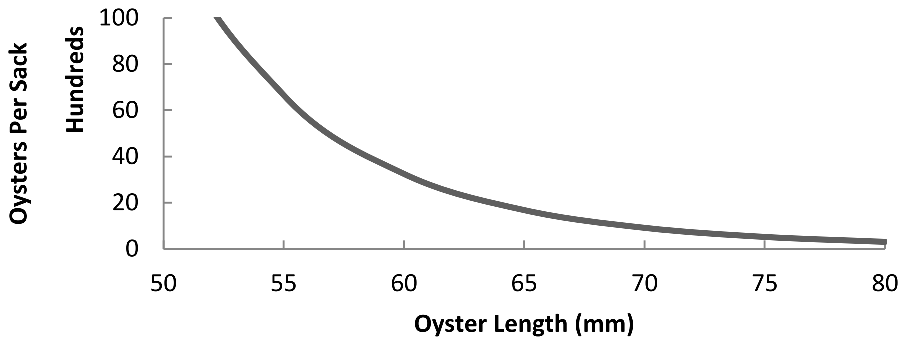


Figure 8. Typical sack volume for oysters of a particular size.

$\boldsymbol{N}$: 1, $\boldsymbol{OPS_b}$:1.767×10^8, $\boldsymbol{OPS_b}$: -2.926, $\boldsymbol{v}$: 52.85

Cultch Component

Shell Loss

Cultch on the reef is subject to environmental dissolution that occurs naturally. When reef particles eventually deteriorate to a fine enough grain, they are unsuitable for oyster recruitment (Gunter 1979). This is simulated by a simple fraction r_t associated with each cultch type, which represents the ratio of cultch that is lost each month. This amount is debited from the cultch stock at the beginning of each month. Thus, given M grams of a particular cultch type, the cultch lost to dissolution $M_{dissolved}$ is given by the following equation (Eq. 17).

$$M_{dissolved} = r_t \times M \qquad \text{Eq. 17}$$

Values for cultch loss are difficult to estimate; however, experiments on local reefs are being conducted to determine this parameter. Note that in the original model, this parameter was provided as an annual decay rate. This was converted to a fraction in the updated model for simplicity, since the percent removed is constant for each discrete month.

Cultch From Dead Oysters

Each oyster that dies naturally in place contributes its shell mass back into the reef environment. This is simulated by approximating the mass of each oyster that undergoes natural mortality and adding this to the cultch mass stock of a specially-designated cultch type representing dead oyster shell. A power model is used to perform this conversion. Given model variables M_a and M_b, the shell massed M_{gained} gained from N oysters of length λ can be approximated by:

$$M_{gained} = \mathrm{N} \times M_a \times \lambda^{M_b} \qquad \text{Eq. 18}$$

The relationship between length and contributed shell mass is demonstrated by representative values of M_a and M_b in the graph below (Figure 9).

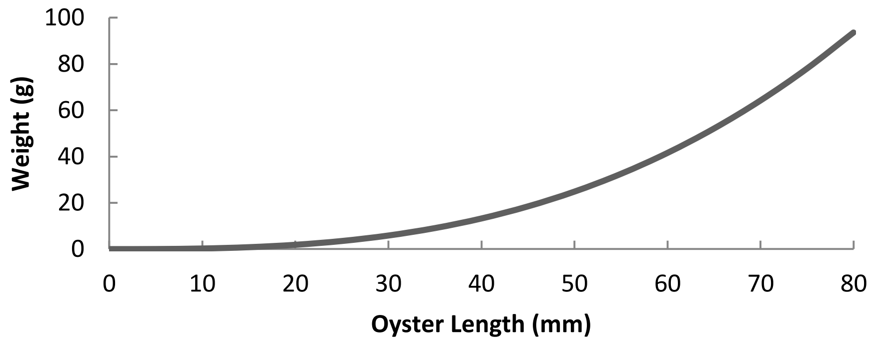


Figure 9. Typical contributed shell weights as a function of oyster length.

$\boldsymbol{M_a}$: 0.0004, $\boldsymbol{M_b}$: 2.8213

This relationship is also useful for determining the mass of live oysters still on the reef. The same formula can be used to track changes in live oyster weight, assuming a conversion factor from shell weight to whole oyster weight. This may be helpful when considering live biomass as a sustainability criterion.

Sack Conversion

Like oysters, shell mass must be converted into sacks. Given M grams of shell, the corresponding sack quantity V for sacks of volume v (in liters) is given by the following equation (Eq. 19).

$$V(M) = \frac{M}{d \times v \times p} \quad \text{Eq. 19}$$

The variables d and p above are specific to each cultch type. Specifically, d represents the estimated density of the shell (g/L) that carries out the conversion. The variable p is a packing coefficient, the ratio of occupied volume of shell to volume of cultch. The packing coefficient is required due to the loose packing of the cultch inside of the sacks (i.e. 1 is perfect packing, whereas lower values indicate more dispersed packing). The graph below (Figure 10) illustrates this linear relationship for typical values.

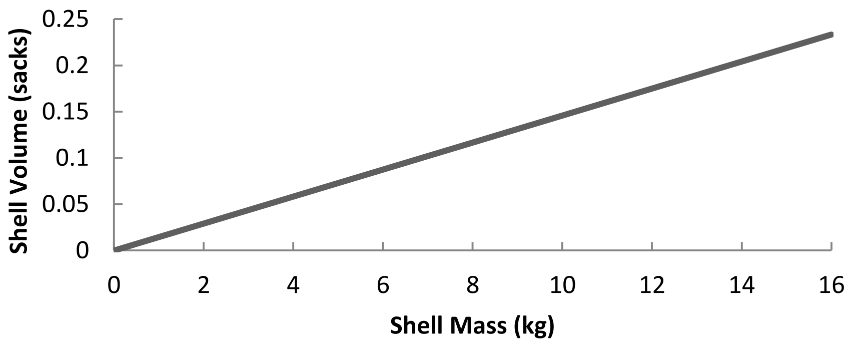


Figure 10. Typical shell mass-to-volume relationship

$\boldsymbol{d}$: 2200, $\boldsymbol{v}$: 52.85, $\boldsymbol{p}$: 0.59

Fishing

The user initializes the model with oyster fishing rates for each month of the simulation. These rates, R_{sack}, R_{seed}, and R_{cultch} are arrays of percentages of resources fished for sack oysters (≥ 75 mm), seed/spat oysters (< 75 mm), and cultch mass, respectively. Each of these is defined explicitly to account for the month's fishing activity. Oyster fishing is simulated by simply removing the corresponding fraction of the quantity of each oyster work group. Cultch fishing is simulated by removing the corresponding fraction of each cultch type's mass. Each cultch type is assumed to be fished in equal proportions.

Given N *seed*-sized oysters, the number fished $F_{sack,m}$ from the available population in month m is approximated by the equation below (Eq. 20).

$$F_{seed,m} = R_{seed,m} \times N \tag{Eq. 20}$$

Likewise, given N sack-sized oysters, the number fished $F_{sack,m}$ from the available population in month m is approximated by the equation below (Eq. 21).

$$F_{sack,m} = R_{sack,m} \times N \tag{Eq. 21}$$

Finally, given M grams of remaining shell, the shell mass fished $F_{cultch,m}$ from the available stock in month m is approximated by the equation below (Eq. 22).

$$F_{cultch,m} = R_{cultch,m} \times M \tag{Eq. 22}$$

Fishing from an oyster reef generally occurs in one of three ways (Soniat et al, 2012):

(1) sack oysters are directly harvested for the market,

(2) shell and seed oysters are fished for transplant to private leases, and

(3) shell, seed, and sack oysters are all harvested together for transplant to private leases.

Since sack oysters, seed oysters, and shell mass are handled independently, care must be taken to ensure that the fishing fractions entered represent realistic fishing scenarios. For example, for direct sack harvest represented by the first scenario, the incidental harvest of seed and shell is assumed to be minimal. When simulating the second scenario, seed and shell should be harvested at the same fraction. For the third scenario, all rates should be equivalent.

Because each month is discretized in the way previously described, the order in which fishing rates apply each month affects the resulting estimated sustainable harvest. An earlier version of the model treated each removal fraction independently to the month's initial stock. However, the newer version applies fishing fractions to the stock remaining after oyster mortality and shell dissolution. This has the precautionary effect of producing lower sustainable fishing estimates.

Model Pseudocode

The logic of the model described in the preceding section is now fully detailed in high-level pseudocode. In the current implementation, the model runs independently for each station and results are aggregated as needed for displaying the results. The algorithms described here are slightly simplified for the sake of clarity, omitting some minor calculated values that are sometimes important for interpreting the results.

Table 6 below describes the global parameters required, which are used by every station in the simulation. These parameters control the size classification of oysters, the sack capacity, and the months for which the model is run.

Table 6. Global model parameters.

Symbol	Description
$\boldsymbol{t_0}$	Initial month of simulation (1 = January, etc.).
$\boldsymbol{t_{max}}$	Number of months to run the simulation.
$\boldsymbol{v}$	Sack capacity (L / sack).
$\boldsymbol{\lambda_{sack}}$	Minimum sack oyster size (mm).
$\boldsymbol{\lambda_{seed}}$	Minimum seed oyster size (mm).

Table 7 describes each per-station parameter. These parameters can be specified separately for each station in the simulation. This allows a high degree of flexibility when dealing with oysters from different regions, In practice, however, simulations will probably treat most stations identically, excepting only a few stations in which different parameters must apply. For the older version of the model without station-specific input, the per-station parameters above are simply globals applied to the input as a whole.

Let s be the current station. The parameters N, L, and C in Table 7 are, respectively, the estimated oyster quantities ($N_{t,s}$), the associated mid-value lengths, and the estimated cultch masses per type ($M_{t,s}$), as described from the Model Data section. The remaining parameters correspond to the values described in the Model Logic section above.

Table 7. Per-station model parameters.

Symbol	Description
	Data Parameters
N	Array of initial oyster numbers.
L	Array of initial oyster sizes (mm), with indices corresponding to N.
$\boldsymbol{M}$	Array of initial cultch mass (g) of each cultch type.
	Fishing Parameters
$\mathbf{R}_{\boldsymbol{sack}}$	Array of monthly removal fractions of sack.
$\mathbf{R}_{\boldsymbol{seed}}$	Array of monthly removal fractions of seed/spat.
$\mathbf{R}_{\boldsymbol{cultch}}$	2D array of monthly removal fractions for each cultch type.
	Cultch Parameters
$\boldsymbol{NL}$	Array of monthly natural shell loss fractions for each cultch type.
$\boldsymbol{D}$	Array of shell densities ($g\ /\ L$) for each cultch type.
$\boldsymbol{P}$	Array of packing coefficients for each cultch type.
	Oyster Growth Parameters
$\boldsymbol{M_a}$	Mass model coefficient.
$\boldsymbol{M_b}$	Mass model exponent.
$\boldsymbol{OPS_a}$	Per-sack model coefficient.
$\boldsymbol{OPS_b}$	Per-sack model exponent.
$\boldsymbol{\lambda_\infty}$	Asymptotic maximum oyster size (mm).
$\boldsymbol{k_0}$	Mean annual growth rate.
$\boldsymbol{k_1}$	Growth rate freedom.
$\boldsymbol{t_{avg}}$	Time of average growth rate (years).
	Oyster Mortality Parameters
$\boldsymbol{m_{0,A}}$	Adult mean annual mortality rate.
$\boldsymbol{m_{1,A}}$	Adult annual mortality rate freedom.
$\boldsymbol{m_{0,J}}$	Juvenile mean annual mortality rate.
$\boldsymbol{m_{1,J}}$	Juvenile annual mortality rate freedom.
$\boldsymbol{m_{avg}}$	Month of average mortality ($1 =$ January, etc.).

For all pseudocode used in this section, assume a dynamically-typed language similar to JavaScript. The array indexing scheme used is informal.

GET_STAGE

This routine is used to classify oysters of a given length ($length$) in mm into SACK, SEED, or SPAT constants based on specified global threshold sizes λ_{sack} and λ_{seed}.

```
function GET_STAGE ($length$) {
    if ($length \geq \lambda_{sack}$)
        return SACK
    else if ($length \geq \lambda_{seed}$)
        return SEED
    else
        return SPAT
}
```

GET_MORTALITY_FRACTION

This routine computes the fraction of oysters dead due to natural causes at a given month ($time$) and of length ($length$) in mm. The fraction is based on an exponential model with a time-dependent rate that uses different constants for adult and juvenile oysters. At month m_{avg}, the mortality rate is equivalent to the respective mean rate r_0, but can vary seasonally by as much as $\pm r_1$.

```
function GET_MORTALITY_FRACTION ($length, time$) {
    if (GET_STAGE($length$) $\neq$ SPAT)
        var  $m \leftarrow m_{0,A} + m_{1,A} \times \sin\left(2\pi \times \frac{time - m_{avg}}{12}\right)$
    else
        var $m \leftarrow m_{0,J} + m_{1,J} \times \sin\left(2\pi \times \frac{time - m_{avg}}{12}\right)$
    return $1 - e^{-(\mathrm{m}/12)}$
}
```

GET_SACKS_PER_OYSTER

This routine calculates the conversion factor for oysters of a particular length ($length$) in mm for calculating sack volume from count. A simple polynomial model is used to perform this conversion using two constants, the OPS_a coefficient and the OPS_a exponent. Since this formula was derived from experiments using 1.5 bushel sacks, an additional scaling factor v is added that permits a global, user-defined sack capacity.

```
function GET_SACKS_PER_OSYTER (length) {
    return $\frac{52.85}{OPS_a \times length^{OPS_b} \times v}$
}
```

GET_SACKS_PER_GRAM

This routine calculates a linear conversion factor for cultch of type c for calculating sack volume from cultch mass. The parameterized density and packing coefficient is used to perform the conversion, and the global v variable is used to allow a user-defined sack capacity.

```
function GET_SACKS_PER_GRAM (c) {
    return $\frac{1}{D[c] \times P[c] \times v}$
}
```

GET_OYSTER_MASS

This routine relates the length of an oyster to its associated mass. A simple polynomial model is used to perform this conversion using two constants, the M_a coefficient and the M_b exponent. This function is used to estimate both the shell contribution of dead oysters and (by derivation) the mass of living oysters on the reef.

```
function GET_OYSTER_MASS (length) {
    return $M_a \times length^{M_b}$
}
```

GET_NEW_LENGTH

This routine is used to calculate the adjusted length of an oyster of a particular length ($length$) in mm after growth over the course of a given month ($time$) using a von Bertalanffy growth curve. The growth rate k is time-dependent, specified by an average rate k_0 at time $t_{avg}$with freedom of $\pm k_1$. As the oyster grows larger, it approaches an asymptotic maximum size, specified by λ_∞. In the case that the oyster already exceeds λ_∞, the original length is returned (in the case of exceptionally large input oysters).

```
function GET_NEW_LENGTH ($length, time$) {
    if ($length > l_\infty$)
        return $length$
    else {
        var $k \leftarrow k_0 + k_1 \times \sin\left(2\pi \times \left(\frac{time}{12} - t_{avg}\right)\right)$
        return $\lambda_\infty - (\lambda_\infty - length) \times \mathrm{e}^{-k/12}$
    }
}
```

GET_NET_SHELL_LOSS

This is the primary routine of the model. It takes all input parameters and performs the entire station simulation, returning the difference between initial and starting cultch types.

```
function GET_NET_SHELL_LOSS ($station$) {
    var t ← $t_0$
    var total_initial_shell ← 0
    for each (m in M)
        total_initial_shell ← total_initial_shell + M[m]
    var shell ← total_initial_shell
    var harvested_oyster_sacks ← 0
    var harvested_shell_sacks ← 0
    while (t – $t_0 < t_{max}$) {
        //Perform oyster simulation.
        var new_shell ← 0
        for each (g in N) {
            var dead ← GET_MORTALITY_FRACTION(t, L[g]) × N[g]
            new_shell ← new_shell + GET_OYSTER_MASS(L[g]) × dead
            if (GET_STAGE(L[g]) = SACK)
                var fished ← $R_{sack}$[t] × (N[g] – dead)
            else
                var fished ← $R_{seed}$[t] × (N[g] – dead)
            harvested_oyster_sacks ← harvested_oyster_sacks
                + fished × GET_SACKS_PER_OYSTER(L[g])
            N[g] ← N[g] – dead – fished
            L[g] ← GET_NEW_LENGTH(L[g],t)
        }
        //Perform cultch simulation.
        for each (m in M) {
            if (m = DEAD_OYSTER_SHELL_INDEX)
                var shell_created ← new_shell
            else
                var shell_created ← 0
            var shell_lost ←NL[m] × M[m]
            var shell_fished ← $R_{cultch}$[m,t]
                 × (M[m] + shell_created – shell_lost)
            var harvested_shell_sacks ← harvested_shell_sacks
                + shell_fished × GET_SACKS_PER_GRAM (m)
            M[m] ← M[m] – shell_lost + shell_created – shell_fished
            shell  ← shell – shell_lost + shell_created – shell_fished
        }
        t ← t + 1
    }
    return (total_initial_shell – shell)
}
```

The GET_NET_SHELL_LOSS function will be run for every station in the set of stations under consideration. Assume the per-station constants referenced above are valid for the current station being simulated.

Analysis

The bulk of the per-station code is contained within a month-iterating loop, which itself contains two independent loops iterating through the number of work groups and cultch types, respectively. Let $t = t_{max}$, $m = |M|$, and $n = |N|$ from the above pseudocode. Assuming the constant run-time of all basic arithmetic and indexing, the algorithm has a run time complexity of $\Theta(t \times (n + m))$, or, equivalently, $\Theta(t \times \max(n, m))$ (Cormen, 2009).

Furthermore, if running the simulation for each of $s = |S|$ stations, each with a maximum of c_{max} cultch types and g_{max} work groups, the total run-time will have a worst-case run-time complexity of $O(s \times t \times (g_{max} + c_{max}))$. In reality, the number of oyster work groups will exceed the number of cultch types in most cases. Therefore, the usual-case upper-bound run-time is $O(s \times t \times g_{max})$.

These bounds do not include the cost of performing the replicate aggregation for each station. Assuming a maximum per-station replicate count of r_{max}, this will introduce another worst-case $O(s \times r_{max} \times (g_{max} + c_{max}))$ preprocessing factor. However, since the replicate count is usually low, the usual-case upper-bound run-time above is expected to hold.

The SQL implementation of the model has a similar complexity. However, the parallelization and optimizations built into the database engine may decrease the run-time by a constant factor as determined by available CPU's, caching ability, and other factors.

Future Changes

While the current model version provides all the currently-desired functionality, the possibility of expanding it is still open. One of the most useful features would be the inclusion of an algorithm to obtain the appropriate fishing fractions automatically, within the desired precision. The current SQL implementation (see the following section) allows the user to specify fishing rates in relative terms. A simple iterative algorithm could be used to determine which additional fishing modifier would best meet certain sustainability criterion. Additional investigation into the algorithm itself could allow a more mathematically elegant solution to this problem. This is left as a future goal if such a feature is deemed useful.

Additionally, the extent to which oyster growth and mortality and cultch loss are dependent on temperature and salinity is not widely understood. As these relationships become better-known, the model can be automated to use environmental conditions from stock assessments to estimate those parameters.

4. Implementation

Introduction

The oyster sustainability model has undergone significant changes since its first conception. Originally a Fortran model, the author transcoded the model in Java, then subsequently rebuilt it as a MS SQL algorithm. This section describes various implementation details and a discussion on the choices that were made. It also demonstrates some of the front-end features the implementation provides.

Previous Implementations

Fortran Implementation

In March 2010, Dr. John Klinck coded the initial version of the sustainability model in Fortran based on a rough specification developed with Eric Powell and Thomas Soniat. Input was controlled via formatted text files specified by command line arguments. Output was generated as a text report. UNO undergraduate student Janak Dahal worked to link the model to LDWF stock assessment data. Due to the complexity of selecting and preparing this data for consumption by the model, a user interface was required. Using Microsoft Access, a stock assessment database was created, along with associated forms for selecting and aggregating data. However, interaction with this architecture was difficult since the output report had to be parsed manually to extract specific data at known offsets. The model itself was treated as a black box. It soon became evident that the model needed to be reconstructed in order to be more easily integrated into a user interface.

Java Implementation

The architecture of the stock assessment model was completely renovated beginning in May 2011. The Access databases containing sample data were exported into the more robust MS SQL format. The author transcribed the original Fortran code into Java. In doing so, several improvements were made. As the original program was mostly untested, several logical errors were discovered that could easily have resulted in invalid output. These included bugs in the main simulation loop that accumulated fishing statistics. Additionally, care was given to prevent program crashes. Edge-case input data (i.e. null or zero values) in the original model could have potentially resulted in division by zero or other fatal errors. These cases were addressed using conditional logic and exception-handling. The new model program was deployed as a REST (Representational State Transfer) web service on the Apache Tomcat platform. This service type can be accessed by any technology capable of sending HTTP requests and features can easily be exposed or hidden via simple configuration. The RESTful architecture was also particularly appealing as it allowed the model to remain agnostic to any future integration changes. The server accepted a single JSON-formatted string as input. This string was parsed into objects and used to instantiate Java Objects required by the model. The simulation was then performed and output sent back to the client, packaged in JSON format.

Current Implementation

Introduction

While the Java implementation provided a clear upgrade from the original prototype, it did not offer the flexibility required for large-scale interaction with the available data. It would also be clumsy for the web server to interact with the REST server if a larger load of data was required to be interchanged. To improve upon this, a second implementation was provided by the author, this time using MS SQL as the entire foundation. A new stock assessment database schema proposed by the author was implemented by undergraduate student Josh Gallegos, which made the data from the original Access database more structured. Gallegos also assisted by updating front-end forms, providing additional database implementation duties, and validating oyster data already entered.

A completely in-database implementation provides several advantages over the previous solutions. First, latency and throughput are improved, since the data requires no inter-process communication to receive and output the results. The algorithms further benefit from the parallelization, caching, and performance optimizations native to the database engine. Furthermore, it allows previous boiler-plate data management structures that were required in the original implementations to be handled by database engine itself, eliminating possible errors. Finally, it allows the logic to be expressed in a universal syntax. The syntax of MS SQL is well-known and allows future maintainers to focus on the logic rather than the designer's specific construction of data structures.

This implementation choice had very few disadvantages. The only significant consideration was that it is a departure from the original code. The first two implementations were traditional procedural algorithms with a front-end command-line interface. The database implementation introduces a dependency on the MS SQL environment and associated data tables. However, since the data and server environment have been stable for some time, this is not considered a significant detriment.

By transitioning to the pure SQL format, new features were able to be added. Simulations can be easily run per-station, so it is no longer necessary to aggregate all input into a single result. Also, results can be fine-tuned and adjusted with little expense.

Data Schema

The original oyster data schema was developed by Janak Dahal in MS Access from the original paper data forms. Dahal and the author migrated this to MS SQL via a direct format conversion in May of 2011. The schema remained relatively unchanged until the construction of the MS SQL model implementation, which was completed in mid 2013. During this time, the developers took the opportunity to improve the database, introduce additional constraints, and generalize some features.

The current scheme diagram for stock assessment data is depicted in the diagram below (Figure 11).

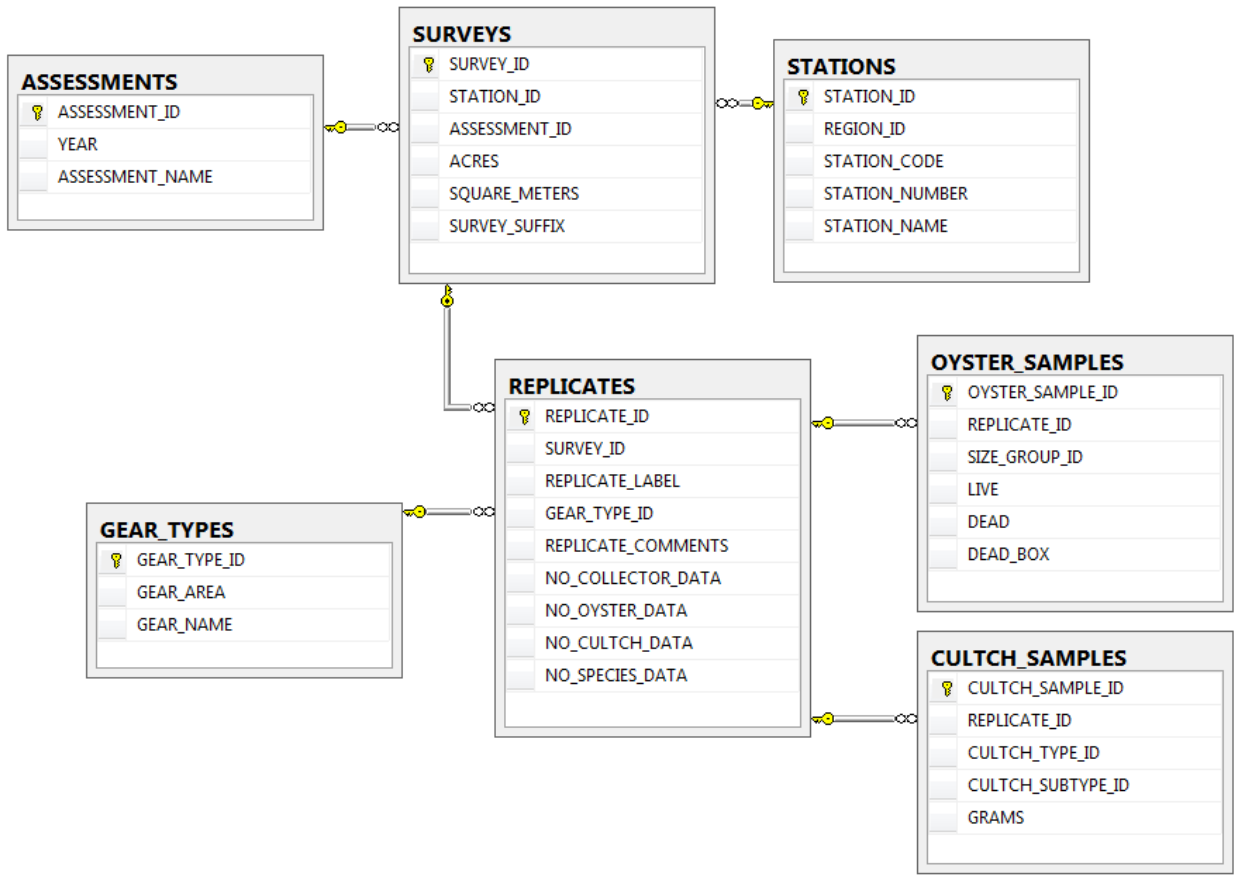


Figure 11. The data MS SQL database schema.

The ASSESSMENTS table is used to provide the highest-level separation of data samples. A new entry is created every year during the annual stock assessment. The SURVEYS table is used to group together replicates within a single station. This design was chosen so that stations could potentially be re-surveyed (as was the case in August 2012, when some stations had to be re-surveyed after Hurricane Isaac). Each SURVEYS entry is associated with a STATIONS entry. This abstracts the station information from the sampling initiative (a problem with the old schema, as station acreage may change annually with changes in policy).

The REPLICATES table contains entries associated with their respective surveys. For each replicate, an entry from GEAR_TYPE is associated. Finally, OYSTER_SAMPLES and CULTCH_SAMPLES provide the replicate data on a per-size-group and per-cultch-type basis, respectively.

Model Schema

The figure below (Figure 12) shows a simplified version of the MS SQL schema corresponding to the model configuration data.

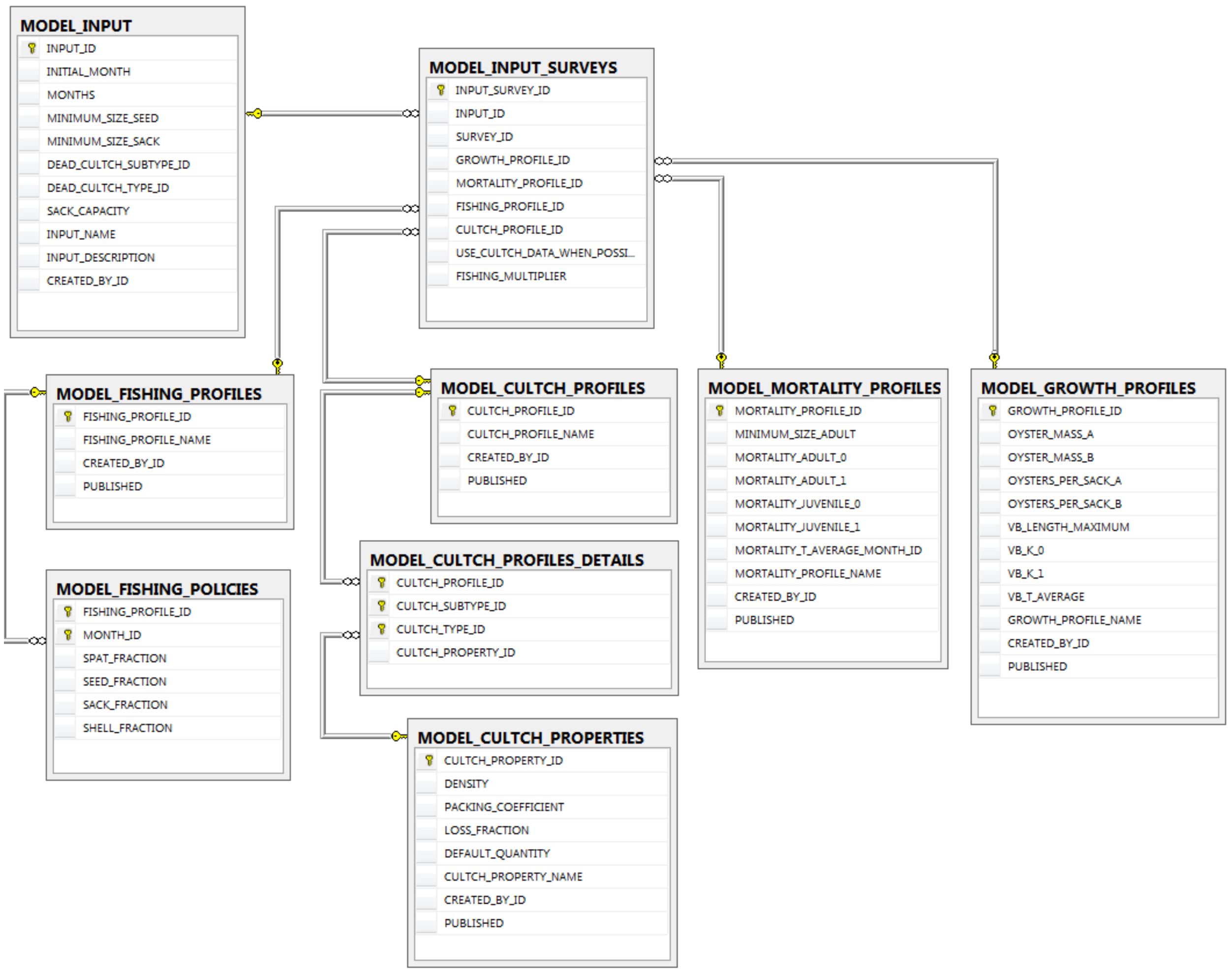


Figure 12. The model MS SQL database schema.

The MODEL_INPUT table allows users to enter the global variables for their simulation. One or more MODEL_INPUT_SURVEYS are associated with each MODEL_INPUT entry that specifies the SURVEYS entry the model will be run against. For each of these, different profiles can be configured. The per-station parameters described in the Pseudocode section are broken down into MODEL_GROWTH_PROFILES, MODEL_MORTALITY_PROFILES, MODEL_CULTCH_PROFILES, and MODEL_FISHING_PROFILES. This allows stations to potentially re-use profiles as deemed necessary.

Of particular interest is the FISHING_MULTIPLIER field in the MODEL_INPUT_SURVEYS table. This value is multiplied by the fishing fractions provided in the MODEL_FISHING_POLICIES table to obtain the effective fishing fractions. This allows the user to configure each fishing profile as independent proportions, then scale the values quickly when configuring each specific station.

Simulation Schemas

The figure below (Figure 13) describes the schema used for the data simulation. A separate stored procedure runs, acting on a single MODEL_INPUT table. This generates a new MODEL_OUTPUT entry. Information from the MODEL_INPUT_SURVEYS is copied over as a new MODEL_OUTPUT_SURVEYS entry so that MODEL_INPUT_SURVEYS data can be deleted and modified without affecting the output from a previous run.

The MODEL_OUTPUT_OYSTERS and MODEL_OUTPUT_CULTCH tables contain the simulation results. The simulation procedure updates the removal fractions and uses SQL's calculated column feature to perform the conversions automatically. Both tables track the resource's starting amount (count for oysters and mass for cultch) and the quantities removed by category. The simulation procedure then uses this information to determine the remaining quantity, which becomes the next month's initial amount. Thus, a station that runs for t months will require $t + 1$ rows of output per resource, as an additional final-results entry is always required.

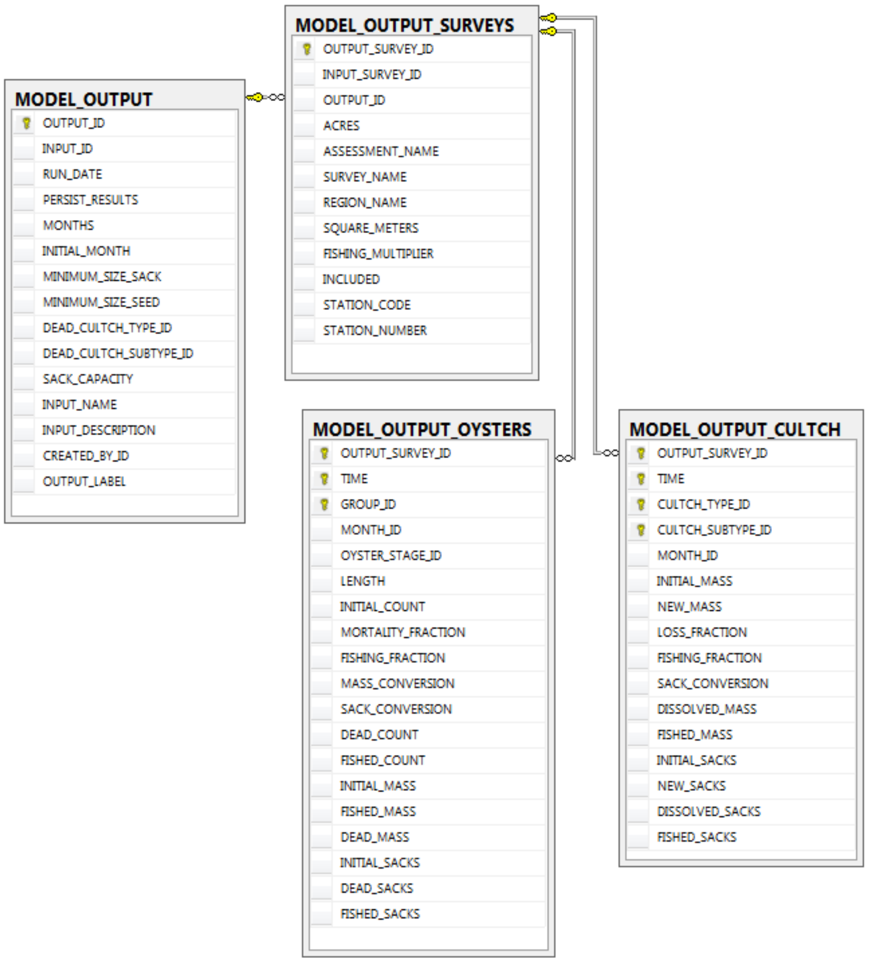


Figure 13. The model simulation MS SQL schema.

Procedures

The logic of the model is wrapped in a single procedure, RUN_MODEL, depicted in the sample on the following pages. For brevity and security reasons, the complete source is not included.

```sql
BEGIN TRANSACTION MODEL

    -- Perform cleanup of older models.
    DECLARE @cleanup_days int = 30;
    DECLARE @cleanup_delete_protected bit = 0;
    EXEC dbo.CLEAR_MODEL_OUTPUT @cleanup_days, @cleanup_delete_protected;

    -- Setup
    DECLARE @run_date datetime = SYSDATETIME();
    DECLARE @total_months int, @initial_month int;
    SELECT
        @total_months = MONTHS,
        @initial_month = INITIAL_MONTH
    FROM MODEL_INPUT WHERE MODEL_INPUT.INPUT_ID=@input_id

    -- Check input parameters.
    IF (@@ROWCOUNT != 1)
        BEGIN
            RAISERROR('The INPUT_ID provided is invalid.',15,1)
            ROLLBACK TRANSACTION MODEL
        END
    ELSE IF NOT EXISTS (SELECT * FROM MODEL_INPUT_SURVEYS WHERE MODEL_INPUT_SURVEYS.INPUT_ID=@input_id)
        BEGIN
            RAISERROR('There are no MODEL_INPUT_SURVEYS associated with the provided INPUT_ID.',15,1)
            ROLLBACK TRANSACTION MODEL
        END
    ELSE IF EXISTS (SELECT * FROM MODEL_OUTPUT WHERE MODEL_OUTPUT.OUTPUT_LABEL = @output_label)
        BEGIN
            RAISERROR('A MODEL_OUTPUT record already exists with the given OUTPUT_LABEL.',15,1)
            ROLLBACK TRANSACTION MODEL
        END
    ELSE
        BEGIN
```

```
-- Copy MODEL_INPUT parameters into a new MODEL_OUTPUT row to save the state of the input.
INSERT INTO MODEL_OUTPUT (
    MODEL_OUTPUT.RUN_DATE,
    MODEL_OUTPUT.PERSIST_RESULTS,
    MODEL_OUTPUT.INPUT_ID,
    MODEL_OUTPUT.MINIMUM_SIZE_SEED,
    MODEL_OUTPUT.MINIMUM_SIZE_SACK,
    MODEL_OUTPUT.DEAD_CULTCH_TYPE_ID,
    MODEL_OUTPUT.DEAD_CULTCH_SUBTYPE_ID,
    MODEL_OUTPUT.INITIAL_MONTH,
    MODEL_OUTPUT.MONTHS,
    MODEL_OUTPUT.INPUT_NAME,
    MODEL_OUTPUT.INPUT_DESCRIPTION,
    MODEL_OUTPUT.SACK_CAPACITY,
    MODEL_OUTPUT.CREATED_BY_ID,
    MODEL_OUTPUT.OUTPUT_LABEL
)
SELECT
    @run_date,
    0, /*false*/
    INPUT_ID,
    MINIMUM_SIZE_SEED,
    MINIMUM_SIZE_SACK,
    DEAD_CULTCH_TYPE_ID,
    DEAD_CULTCH_SUBTYPE_ID,
    INITIAL_MONTH,
    MONTHS,
    INPUT_NAME,
    INPUT_DESCRIPTION,
    SACK_CAPACITY,
    CREATED_BY_ID,
    @output_label
FROM MODEL_INPUT WHERE MODEL_INPUT.INPUT_ID=@input_id

-- Determine the new OUTPUT_ID of the MODEL_OUTPUT row we just created.
DECLARE @output_id int = SCOPE_IDENTITY();
```

```
-- Copy the state of each MODEL_INPUT_SURVEYS into new MODEL_OUTPUT_SURVEY rows.
INSERT INTO MODEL_OUTPUT_SURVEYS (
    MODEL_OUTPUT_SURVEYS.OUTPUT_ID,
    MODEL_OUTPUT_SURVEYS.INPUT_SURVEY_ID,
    MODEL_OUTPUT_SURVEYS.ACRES,
    MODEL_OUTPUT_SURVEYS.ASSESSMENT_NAME,
    MODEL_OUTPUT_SURVEYS.SURVEY_NAME,
    MODEL_OUTPUT_SURVEYS.REGION_NAME,
    MODEL_OUTPUT_SURVEYS.FISHING_MULTIPLIER,
    MODEL_OUTPUT_SURVEYS.STATION_CODE,
    MODEL_OUTPUT_SURVEYS.STATION_NUMBER
)
SELECT
    @output_id,
    MODEL_INPUT_SURVEYS.INPUT_SURVEY_ID,
    SURVEYS.ACRES,
    CONVERT(nvarchar(4),ASSESSMENTS.YEAR) + N' ' + ASSESSMENTS.ASSESSMENT_NAME,
    STATIONS.STATION_NAME + CASE WHEN
            SURVEYS.SURVEY_SUFFIX IS NULL THEN '' ELSE ' (' + SURVEYS.SURVEY_SUFFIX + ')' END,
    REGIONS.REGION_NAME,
    MODEL_INPUT_SURVEYS.FISHING_MULTIPLIER,
    STATIONS.STATION_CODE,
    STATIONS.STATION_NUMBER
FROM
    MODEL_INPUT_SURVEYS
    INNER JOIN SURVEYS ON SURVEYS.SURVEY_ID=MODEL_INPUT_SURVEYS.SURVEY_ID
    INNER JOIN STATIONS ON STATIONS.STATION_ID=SURVEYS.STATION_ID
    INNER JOIN ASSESSMENTS ON ASSESSMENTS.ASSESSMENT_ID=SURVEYS.ASSESSMENT_ID
    INNER JOIN REGIONS ON REGIONS.REGION_ID=STATIONS.REGION_ID
WHERE
    MODEL_INPUT_SURVEYS.INPUT_ID=@input_id
```

```
        -- Run the actual simulation.
        DECLARE @time int = 1, @month int = @initial_month;
        WHILE (@time <= @total_months+1)
            BEGIN
                -- Run the oyster simulation, getting data from the previous output rows.
                INSERT INTO MODEL_OUTPUT_OYSTERS (
                    OUTPUT_SURVEY_ID,
                    TIME,
                    MONTH_ID,
                    GROUP_ID,
                    OYSTER_STAGE_ID,
                    LENGTH,
                    INITIAL_COUNT,
                    MORTALITY_FRACTION,
                    FISHING_FRACTION,
                    MASS_CONVERSION,
                    SACK_CONVERSION
                )
                SELECT * FROM dbo.SIMULATE_OYSTERS(@output_id, @time, @month)

                -- Run the cultch simulation, getting data from the previous output rows.
                INSERT INTO MODEL_OUTPUT_CULTCH (
                    OUTPUT_SURVEY_ID,
                    TIME,
                    MONTH_ID,
                    CULTCH_TYPE_ID,
                    CULTCH_SUBTYPE_ID,
                    INITIAL_MASS,
                    NEW_MASS,
                    LOSS_FRACTION,
                    FISHING_FRACTION,
                    SACK_CONVERSION
                )
                SELECT * FROM dbo.SIMULATE_CULTCH(@output_id, @time, @month)

                -- Go to the next month.
                SET @time = @time + 1;
                SET @month = ((@month) % 12) + 1;
            END
            COMMIT TRANSACTION MODEL

        END

    -- Return the new OUTPUT_ID that was generated.
    RETURN @output_id;
```

For housekeeping purposes, a stored procedure REMOVE_OLD_ENTRIES clears away any previous results older than 30 days that are not flagged as PERSISTED. RUN_MODEL then copies the necessary MODEL_INPUT_SURVEYS data into new MODEL_OUTPUT_SURVEYS in order to retain their values across future edits. It then iterates through every month of the simulation. Calls to table-valued functions SIMULATE_OYSTERS and SIMULATE_CULTCH are then made, which abstract the per-month changes for each station's resources.

For the first month of the simulation, the model must first collect the data from the stored stock assessment data. Each month thereafter may be run simply by examining the previous entries. The code for the latter cases is presented below.

First, oysters are simulated as follows:

```
DECLARE @last_month int = ((@month - 2) % 12) + 1;
INSERT INTO @RESULTS
SELECT
      OLD.OUTPUT_SURVEY_ID,
      @time,
      @month,
      OLD.GROUP_ID,
      dbo.GET_OYSTER_STAGE(MINIMUM_SIZE_SEED,MINIMUM_SIZE_SACK,OLD.NEW_LENGTH),
      OLD.NEW_LENGTH,
      OLD.REMAINING,
      CASE WHEN @time=MONTHS+1 THEN 0 ELSE dbo.GET_MORTALITY_FRACTION(
            M.MORTALITY_JUVENILE_0,M.MORTALITY_JUVENILE_1,M.MORTALITY_ADULT_0,M.MORTALITY_ADULT_1,
            M.MINIMUM_SIZE_ADULT,M.MORTALITY_T_AVERAGE_MONTH_ID,OLD.NEW_LENGTH,@month) END,
      CASE WHEN @time=MONTHS+1 THEN 0 ELSE OLD.FISHING_MULTIPLIER * dbo.GET_FISHING_RATE(
            MINIMUM_SIZE_SEED,MINIMUM_SIZE_SACK,
            F.SPAT_FRACTION,F.SEED_FRACTION,F.SACK_FRACTION,OLD.NEW_LENGTH) END,
      dbo.GET_OYSTER_MASS(OYSTER_MASS_A, OYSTER_MASS_B,OLD.NEW_LENGTH),
      dbo.GET_OYSTER_SACKS(OYSTERS_PER_SACK_A, OYSTERS_PER_SACK_B, SACK_CAPACITY,OLD.NEW_LENGTH)
FROM (
      SELECT
            MODEL_OUTPUT_SURVEYS.OUTPUT_ID,
            MODEL_OUTPUT_SURVEYS.FISHING_MULTIPLIER,
            G.OYSTERS_PER_SACK_A,
            G.OYSTERS_PER_SACK_B,
            G.OYSTER_MASS_A,
            G.OYSTER_MASS_B,
            MODEL_INPUT_SURVEYS.MORTALITY_PROFILE_ID,
            MODEL_INPUT_SURVEYS.GROWTH_PROFILE_ID,
            MODEL_INPUT_SURVEYS.FISHING_PROFILE_ID,
            MODEL_OUTPUT_OYSTERS.OUTPUT_SURVEY_ID,
            MODEL_OUTPUT_OYSTERS.GROUP_ID,
            MODEL_OUTPUT_OYSTERS.INITIAL_COUNT - MODEL_OUTPUT_OYSTERS.FISHED_COUNT -
                        MODEL_OUTPUT_OYSTERS.DEAD_COUNT as REMAINING,
            dbo.GET_NEW_OYSTER_SIZE(G.VB_LENGTH_MAXIMUM,
                  dbo.GET_GROWTH_FACTOR(G.VB_K_0,G.VB_K_1,G.VB_T_AVERAGE,@last_month),
                        MODEL_OUTPUT_OYSTERS.LENGTH) as NEW_LENGTH
      FROM
            MODEL_OUTPUT_OYSTERS
            INNER JOIN MODEL_OUTPUT_SURVEYS ON                    q
                  MODEL_OUTPUT_OYSTERS.OUTPUT_SURVEY_ID=MODEL_OUTPUT_SURVEYS.OUTPUT_SURVEY_ID
            INNER JOIN MODEL_INPUT_SURVEYS ON
                  MODEL_INPUT_SURVEYS.INPUT_SURVEY_ID=MODEL_OUTPUT_SURVEYS.INPUT_SURVEY_ID
            INNER JOIN MODEL_GROWTH_PROFILES as G ON
                  MODEL_INPUT_SURVEYS.GROWTH_PROFILE_ID=G.GROWTH_PROFILE_ID
      WHERE MODEL_OUTPUT_OYSTERS.TIME=(@time-1) AND MODEL_OUTPUT_SURVEYS.OUTPUT_ID=@output_id
      ) as OLD
      INNER JOIN MODEL_MORTALITY_PROFILES AS M ON OLD.MORTALITY_PROFILE_ID=M.MORTALITY_PROFILE_ID
      LEFT JOIN MODEL_FISHING_POLICIES AS F ON
            OLD.FISHING_PROFILE_ID=F.FISHING_PROFILE_ID AND F.MONTH_ID=@month
      INNER JOIN MODEL_OUTPUT ON MODEL_OUTPUT.OUTPUT_ID=OLD.OUTPUT_ID
```

Then, cultch is simulated, with the DEAD_MASS quantities from the oyster simulation being applied to the cultch whose CULTCH_SUBTYPE_ID and CULTCH_TYPE_ID corresponding to dead cultch.

```
INSERT INTO @RESULTS
SELECT
      OLD.OUTPUT_SURVEY_ID,
      @time,
      @month,
      OLD.CULTCH_TYPE_ID,
      OLD.CULTCH_SUBTYPE_ID,
      OLD.REMAINING_MASS,
      CASE WHEN OLD.CULTCH_SUBTYPE_ID=DEAD_CULTCH_SUBTYPE_ID AND OLD.CULTCH_TYPE_ID=DEAD_CULTCH_TYPE_ID
            THEN DEAD_MASS ELSE 0 END,
      CASE WHEN @time=MONTHS+1 THEN 0 ELSE OLD.LOSS_FRACTION END,
      CASE WHEN @time=MONTHS+1 THEN 0 ELSE COALESCE(OLD.FISHING_MULTIPLIER*F.SHELL_FRACTION,0) END,
      OLD.SACK_CONVERSION
FROM (
```

```sql
      SELECT
            MODEL_OUTPUT_SURVEYS.OUTPUT_ID,
            MODEL_OUTPUT_SURVEYS.FISHING_MULTIPLIER,
            MODEL_OUTPUT_SURVEYS.INPUT_SURVEY_ID,
            MODEL_OUTPUT_CULTCH.OUTPUT_SURVEY_ID,
            MODEL_OUTPUT_CULTCH.CULTCH_TYPE_ID,
            MODEL_OUTPUT_CULTCH.CULTCH_SUBTYPE_ID,
            MODEL_OUTPUT_CULTCH.LOSS_FRACTION,
            MODEL_OUTPUT_CULTCH.SACK_CONVERSION,
            MODEL_OUTPUT_CULTCH.INITIAL_MASS - MODEL_OUTPUT_CULTCH.DISSOLVED_MASS -
                  MODEL_OUTPUT_CULTCH.FISHED_MASS + MODEL_OUTPUT_CULTCH.NEW_MASS AS REMAINING_MASS
      FROM
            MODEL_OUTPUT_CULTCH
            INNER JOIN MODEL_OUTPUT_SURVEYS ON
                  MODEL_OUTPUT_SURVEYS.OUTPUT_SURVEY_ID=MODEL_OUTPUT_CULTCH.OUTPUT_SURVEY_ID
      WHERE MODEL_OUTPUT_CULTCH.TIME=(@time-1) AND MODEL_OUTPUT_SURVEYS.OUTPUT_ID=@output_id
) as OLD INNER JOIN (
      SELECT
            MODEL_OUTPUT_SURVEYS.OUTPUT_SURVEY_ID,
            SUM(DEAD_MASS) as DEAD_MASS
      FROM MODEL_OUTPUT_OYSTERS
      INNER JOIN MODEL_OUTPUT_SURVEYS ON
            MODEL_OUTPUT_SURVEYS.OUTPUT_SURVEY_ID=MODEL_OUTPUT_OYSTERS.OUTPUT_SURVEY_ID
      WHERE MODEL_OUTPUT_SURVEYS.OUTPUT_ID=@output_id AND MODEL_OUTPUT_OYSTERS.TIME=@time
      GROUP BY MODEL_OUTPUT_SURVEYS.OUTPUT_SURVEY_ID
) as DEAD_OYSTERS ON DEAD_OYSTERS.OUTPUT_SURVEY_ID=OLD.OUTPUT_SURVEY_ID
INNER JOIN MODEL_INPUT_SURVEYS ON MODEL_INPUT_SURVEYS.INPUT_SURVEY_ID=OLD.INPUT_SURVEY_ID
LEFT JOIN MODEL_FISHING_POLICIES AS F ON
      MODEL_INPUT_SURVEYS.FISHING_PROFILE_ID=F.FISHING_PROFILE_ID AND F.MONTH_ID=@month
INNER JOIN MODEL_OUTPUT ON MODEL_OUTPUT.OUTPUT_ID=OLD.OUTPUT_ID
```

Performance

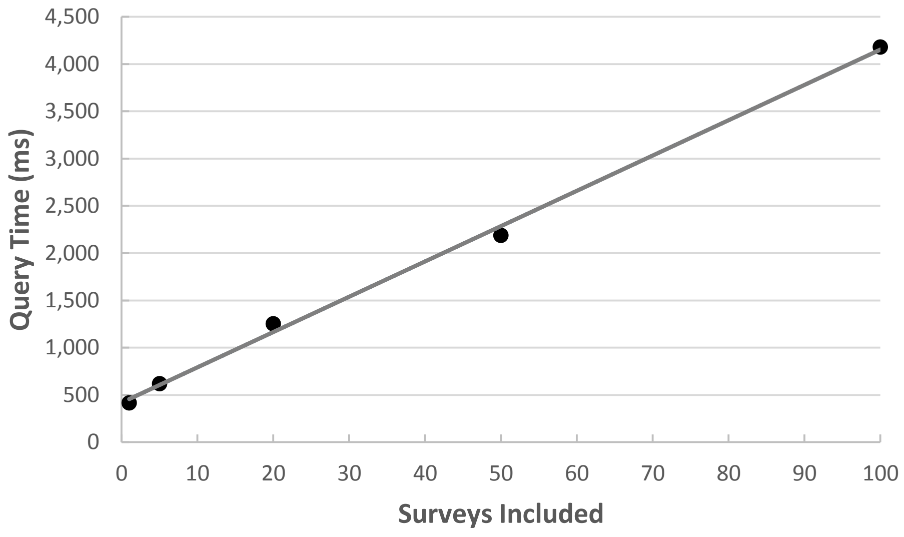


Figure 14. RUN_MODEL procedure performance.

The RUN_MODEL procedure was benchmarked by running the query against a series of MODEL_INPUT values directly from the SQL Server Management Studio. The procedure was run directly from the Dell server hosting the production database. For each trial, the SQL cache was turned off and a simple timer was setup using the CURRENT_TIMESTAMP and DATEDIFF built-in functions. Simulations were performed using 1, 5, 20, 50, and 100 surveys from the 2012 stock assessment. Each simulation ran for 12 months using standard parameters (discussed in the Simulations section). Each survey had low to moderate amounts of oyster and cultch data, with 1 to 5 replicates each. The cultch profile used for each survey only considered brown oyster shell. Three trials were performed for each simulation, with the average of the three taken as the canonical speed. The graph above (Figure 14) shows the results of this benchmarking. Note that in practice, the resulting times will likely be considerably lower, due to the caching ability of MS SQL.

The results indicate that the model scales linearly for station quantities within the average-case range. A linear regression using Microsoft Excel showed an R^2 value of 0.9979. This is consistent with the algorithm analysis performed in the previous section. However, it is possible that surveys with an outlying number of replicates or cultch and oyster samples may result in skewed performance. If performance becomes in issue in practice, investigations should be made to optimize the algorithm.

Data Aggregation

Once the simulation has been run, monthly data is available for each of the stations in the simulation. Since this data is overwhelming in its raw format, SQL views were provided by the author that aggregate this view for easy digestion by the user. A feature of this method is that aggregation within a single simulation can be filtered on a per-station basis. The MODEL_OUTPUT_SURVEYS table is outfitted with a 1-bit INCLUDED column. By introducing predicates that filter out non-INCLUDED surveys, the user may adjust the scope of the simulations results to whatever degree is necessary.

An abundance of such aggregation views exist, so only one sample will be provided here. The MODEL_CULTCH_SUMMARY view offers a high-level overview of each station. It provides the starting and ending cultch quantities, specified in mass and volume, the percent difference between them, as well as other interesting information. Again, note that a significant amount of associated code has been omitted for brevity.

```
SELECT
    STARTING.OUTPUT_ID,
    /*Mass*/
    SUMMARY.TOTAL_NEW_MASS,
    SUMMARY.TOTAL_FISHED_MASS,
    SUMMARY.TOTAL_DISSOLVED_MASS,
    SUMMARY.TOTAL_NEW_MASS/AREA.TOTAL_SQUARE_METERS as NEW_MASS_PER_SQUARE_METER,
    SUMMARY.TOTAL_FISHED_MASS/AREA.TOTAL_SQUARE_METERS as FISHED_MASS_PER_SQUARE_METER,
    SUMMARY.TOTAL_DISSOLVED_MASS/AREA.TOTAL_SQUARE_METERS as DISSOLVED_MASS_PER_SQUARE_METER,
    STARTING.INITIAL_MASS/AREA.TOTAL_SQUARE_METERS as INITIAL_MASS_PER_SQUARE_METER,
    ENDING.REMAINING_MASS/AREA.TOTAL_SQUARE_METERS as REMAINING_MASS_PER_SQUARE_METER,
    STARTING.INITIAL_MASS as TOTAL_INITIAL_MASS,
    ENDING.REMAINING_MASS as TOTAL_REMAINING_MASS,
    dbo.GET_PERCENT_DECREASE(STARTING.INITIAL_MASS,ENDING.REMAINING_MASS) as PERCENT_MASS_LOSS,
    (STARTING.INITIAL_MASS - ENDING.REMAINING_MASS) as NET_MASS_LOSS,
    /*Sacks*/
    SUMMARY.TOTAL_DISSOLVED_SACKS,
    SUMMARY.TOTAL_FISHED_SACKS,
    SUMMARY.TOTAL_NEW_SACKS,
    SUMMARY.TOTAL_NEW_SACKS/AREA.TOTAL_SQUARE_METERS as NEW_SACKS_PER_SQUARE_METER,
    SUMMARY.TOTAL_FISHED_SACKS/AREA.TOTAL_SQUARE_METERS as FISHED_SACKS_PER_SQUARE_METER,
    SUMMARY.TOTAL_DISSOLVED_SACKS/AREA.TOTAL_SQUARE_METERS as DISSOLVED_SACKS_PER_SQUARE_METER,
    STARTING.INITIAL_SACKS/AREA.TOTAL_SQUARE_METERS as INITIAL_SACKS_PER_SQUARE_METER,
    ENDING.REMAINING_SACKS/AREA.TOTAL_SQUARE_METERS as REMAINING_SACKS_PER_SQUARE_METER,
    STARTING.INITIAL_SACKS as TOTAL_INITIAL_SACKS,
    ENDING.REMAINING_SACKS as TOTAL_REMAINING_SACKS,
    dbo.GET_PERCENT_DECREASE(STARTING.INITIAL_SACKS,ENDING.REMAINING_SACKS) as PERCENT_SACKS_LOSS,
    (STARTING.INITIAL_SACKS - ENDING.REMAINING_SACKS) as NET_SACK_LOSS
FROM
    MODEL_AREA as AREA
    INNER JOIN STARTING ON STARTING.OUTPUT_ID=AREA.OUTPUT_ID
    INNER JOIN ENDING ON ENDING.OUTPUT_ID=STARTING.OUTPUT_ID
    INNER JOIN MODEL_OUTPUT ON MODEL_OUTPUT.OUTPUT_ID=STARTING.OUTPUT_ID
    INNER JOIN SUMMARY ON SUMMARY.OUTPUT_ID=STARTING.OUTPUT_ID
```

Additional views partition cultch by type, which is useful when considering multiple cultch types for sustainability. Similar views present oyster data, both as a whole and per-stage. These views, in conjunction with the routines above, various check constraints, computed columns, helper functions, and maintenance procedures, constitute the entirety of the current implementation. The remaining section is dedicated to the front-end view of the model.

Interactive Site

Web Environment

Drupal, a free and open-source PHP-based CMS, was used as a framework for the web site containing the front-end user interface for the model. It is currently running on the Apache HTTP server as a PHP application. Drupal's Form API allowed fast and simple form generation, which was necessary to properly allow for the many input variables the model requires. Gallegos provided the implementation to the new model input tables. The author provided the forms required to view the results of a simulation. A new Dell server was setup to host these services at the UNO Department of Computer Science. The site aims to completely replace the previous incarnation of Oyster Sentinel, and is now up and running with new and migrated content from the previous site at *www.oystersentinel.org*. This section describes the model configuration interface. Details on the data entry forms, implemented by Dahal and Gallegos, are omitted. For security purposes, the site code is not presented in detail; rather, a high-level treatment is provided.

Security

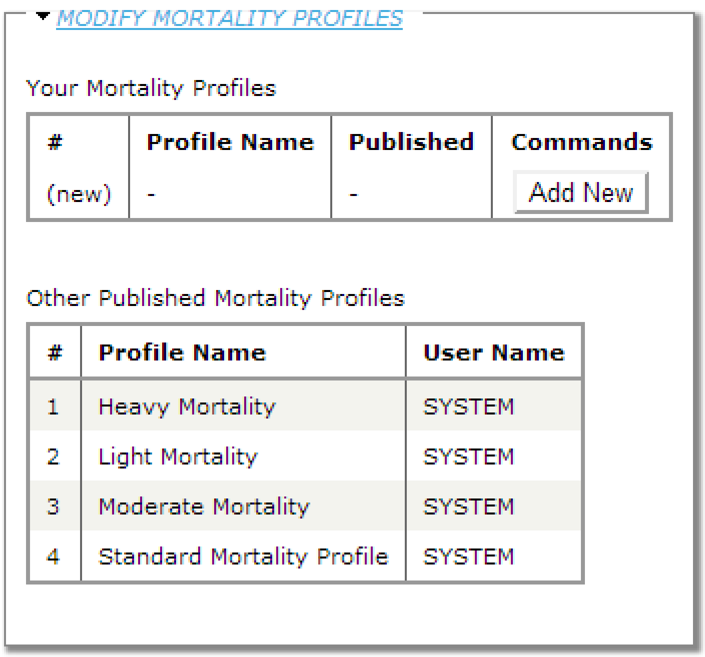
The hierarchical data provided by LDWF required a custom level of security and access privileges. A simple system was implemented in Drupal that allows separation of user control into CSA's, known in the site as *regions*. Each user that logs in can be granted access to zero or more of these regions, allowing them to overwrite stock assessment data and save model presets under region-specific namespaces. This separation prevents users from different CSA's from accidentally overwriting data belonging to one another. Special users called *managers* can delegate access to each CSA via a custom user access page. Managers also have the ability to delete data, create new stock assessments, and organize new stations.

Besides these custom features, the site utilizes several other typical security methods. Passwords are hashed automatically by Drupal. The database is not listening on a public port. Site errors result in PHP execution termination without any debugging information. Drupal's Form API provides server-side validation with post-processing that prevents XSS attacks, input modification, and other malicious techniques. SQL statements are properly bound as PDO objects, reducing the threat of query injection. A CAPTCHA service helps prevent automated logon attempts. Finally, regular backups are made in the event of accidental overwrites or catastrophic failure.

Presentation

The forms used for data input and viewing were designed to be as simple as possible. The stock assessment data entry forms were made to closely mimic the physical paper forms used by the LDWF. Access to the model itself is divided into three stages: (1) Model Profiles, (2) Model Setup, and (3) Model Results.

First, the Model Profiles section (designed by Gallegos) allows users to configure profiles for growth, mortality, cultch, and fishing, as previous described. For example, the figure below (Figure 15) demonstrates the user's view of modifying or deleting growth profiles. Clicking "Edit" will allow the use to modify each field corresponding to that profile. The "Add New" button allows the creation of new profiles.


MODIFY MORTALITY PROFILES

Your Mortality Profiles

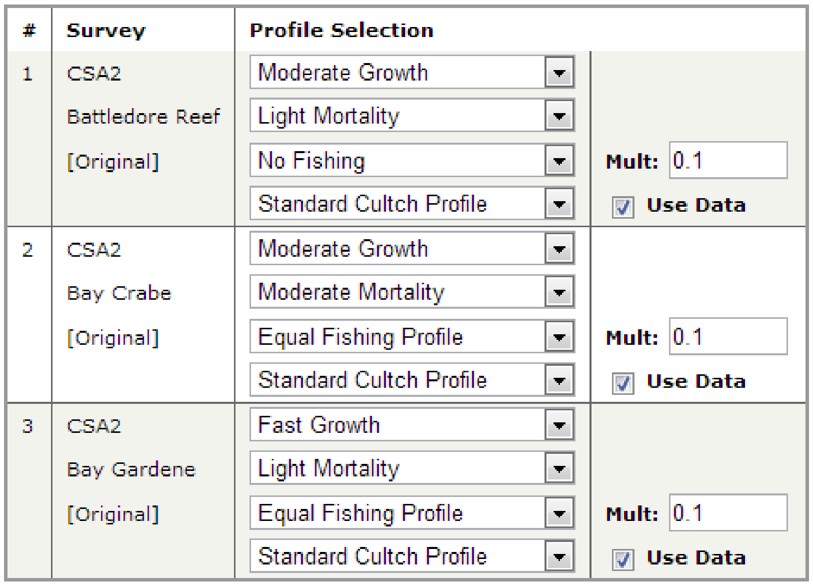
#	Profile Name	Published	Commands
(new)	-	-	Add New

Other Published Mortality Profiles

#	Profile Name	User Name
1	Heavy Mortality	SYSTEM
2	Light Mortality	SYSTEM
3	Moderate Mortality	SYSTEM
4	Standard Mortality Profile	SYSTEM

Figure 15. The growth profile management page.

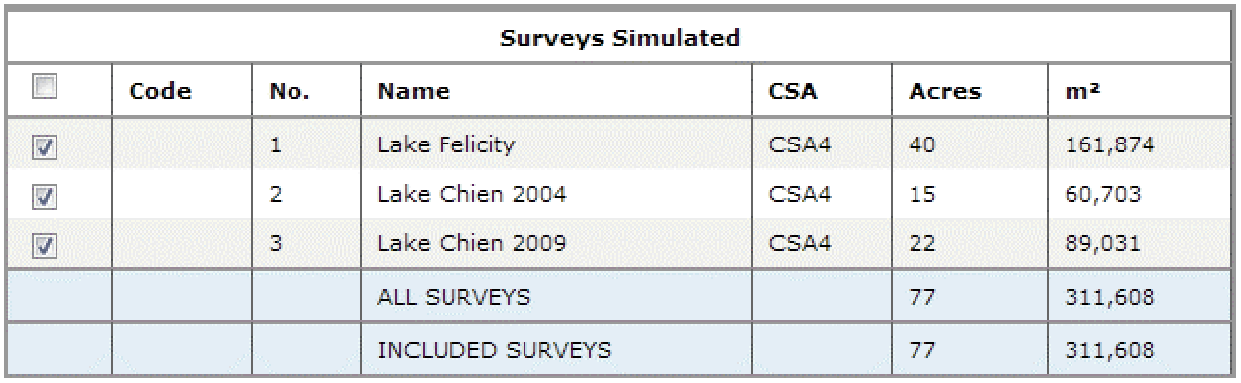
Second, the Model Setup page allows the user to step through the process of running a model. After entering global variables described in the previous section, the user may assign each station profiles that correspond to the simulation being performed. The figure below illustrates how this is done (Figure 16).



#	Survey	Profile Selection	
1	CSA2 Battledore Reef [Original]	Moderate Growth Light Mortality No Fishing Standard Cultch Profile	Mult: 0.1 ✓ Use Data
2	CSA2 Bay Crabe [Original]	Moderate Growth Moderate Mortality Equal Fishing Profile Standard Cultch Profile	Mult: 0.1 ✓ Use Data
3	CSA2 Bay Gardene [Original]	Fast Growth Light Mortality Equal Fishing Profile Standard Cultch Profile	Mult: 0.1 ✓ Use Data

Figure 16. Survey profile assignment.

Finally, the user may view the results of the simulation. The Model Report form provides access and management of the results of previously runs. The figure below (Figure 17) illustrates how specific surveys may be filtered out of the view and data aggregation.

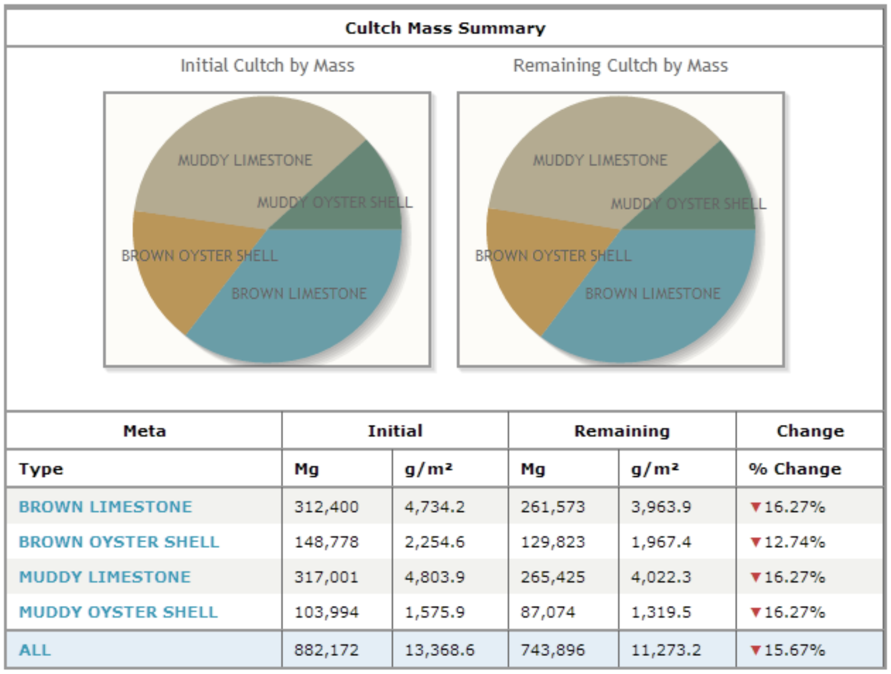


Surveys Simulated						
☐	**Code**	**No.**	**Name**	**CSA**	**Acres**	**m²**
✓		1	Lake Felicity	CSA4	40	161,874
✓		2	Lake Chien 2004	CSA4	15	60,703
✓		3	Lake Chien 2009	CSA4	22	89,031
			ALL SURVEYS		77	311,608
			INCLUDED SURVEYS		77	311,608

Figure 17. Survey filtering form.

The user may then view data in both summary and timeline views. The mass and volume of both oysters and cultch are presented in analogous tables. The *jqplot* JavaScript library was used to present the user with a view graphical representation of the data. This specific library was chosen because it is client-side (easing the load on the server), well-documented, and utilizes the commonly-available HTML canvas interface. For older browsers which do not support the canvas element, this option is not available.

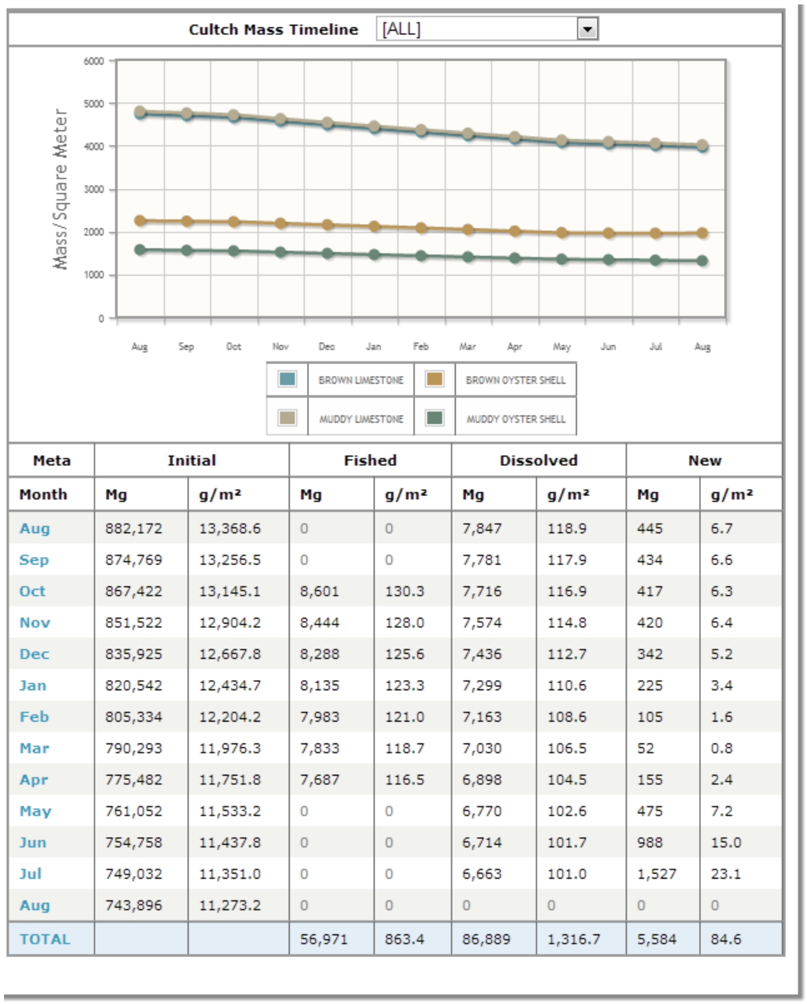
The figure below (Figure 18) demonstrates the summary view for cultch mass. Here, pie charts display how each cultch type contributes to the total cultch present. The amounts of each type at the start and end of the simulation are then shown in the table below.


Cultch Mass Summary

Meta	Initial		Remaining		Change
Type	Mg	g/m²	Mg	g/m²	% Change
BROWN LIMESTONE	312,400	4,734.2	261,573	3,963.9	▼16.27%
BROWN OYSTER SHELL	148,778	2,254.6	129,823	1,967.4	▼12.74%
MUDDY LIMESTONE	317,001	4,803.9	265,425	4,022.3	▼16.27%
MUDDY OYSTER SHELL	103,994	1,575.9	87,074	1,319.5	▼16.27%
ALL	882,172	13,368.6	743,896	11,273.2	▼15.67%

Figure 18. The cultch mass summary display.

The optional timeline view displays the month-to-month changes to each resource. The figure below (Figure 19) demonstrates the timeline view for cultch mass. The line graph shows the relative quantity of the resource for each month. The table below it shows how the resource was affected, with color-coded arrows denoting the loss or gain of each. A drop-down selection menu above the timeline allows the user to filter between different oyster stages and cultch types (depending on the display). Selecting "[ALL]" allows all resources to be combined into a single view.



Meta	Initial		Fished		Dissolved		New	
Month	Mg	g/m²	Mg	g/m²	Mg	g/m²	Mg	g/m²
Aug	882,172	13,368.6	0	0	7,847	118.9	445	6.7
Sep	874,769	13,256.5	0	0	7,781	117.9	434	6.6
Oct	867,422	13,145.1	8,601	130.3	7,716	116.9	417	6.3
Nov	851,522	12,904.2	8,444	128.0	7,574	114.8	420	6.4
Dec	835,925	12,667.8	8,288	125.6	7,436	112.7	342	5.2
Jan	820,542	12,434.7	8,135	123.3	7,299	110.6	225	3.4
Feb	805,334	12,204.2	7,983	121.0	7,163	108.6	105	1.6
Mar	790,293	11,976.3	7,833	118.7	7,030	106.5	52	0.8
Apr	775,482	11,751.8	7,687	116.5	6,898	104.5	155	2.4
May	761,052	11,533.2	0	0	6,770	102.6	475	7.2
Jun	754,758	11,437.8	0	0	6,714	101.7	988	15.0
Jul	749,032	11,351.0	0	0	6,663	101.0	1,527	23.1
Aug	743,896	11,273.2	0	0	0	0	0	0
TOTAL			56,971	863.4	86,889	1,316.7	5,584	84.6

Figure 19. The cultch mass timeline display.

Performance

The Model Results page is timed for efficiency evaluation using two timers. The first timer ("Server") is performed in PHP, evaluating the time elapsed to execute the Drupal module that collects input, initiates the SQL result aggregations, and generates JavaScript for rendering the *jqplot* charts. The second timer ("Page") is performed in JavaScript, starting when browser executes the first body scripts loaded by Drupal and ending with the *document.ready* handler call.

The same simulations described in the RUN_MODEL benchmark earlier were viewed to gauge the speed of the Model Results page. The results of the Server and Page timers are shown. Each page was refreshed three times after initially loading the page once (for more consistent browser caching) and the average of the three trials was used. The trials were also run in two modes. In “Min” mode, timelines and graphs were turned off. In “Full” mode, both timelines and graphs were both enabled. The browser used was Google Chrome Version 27.0.1453.116 m, running on a Dell Inspiron One with Windows 7 and a 2.40 GHz AMD Athlon II X4 610e processor.

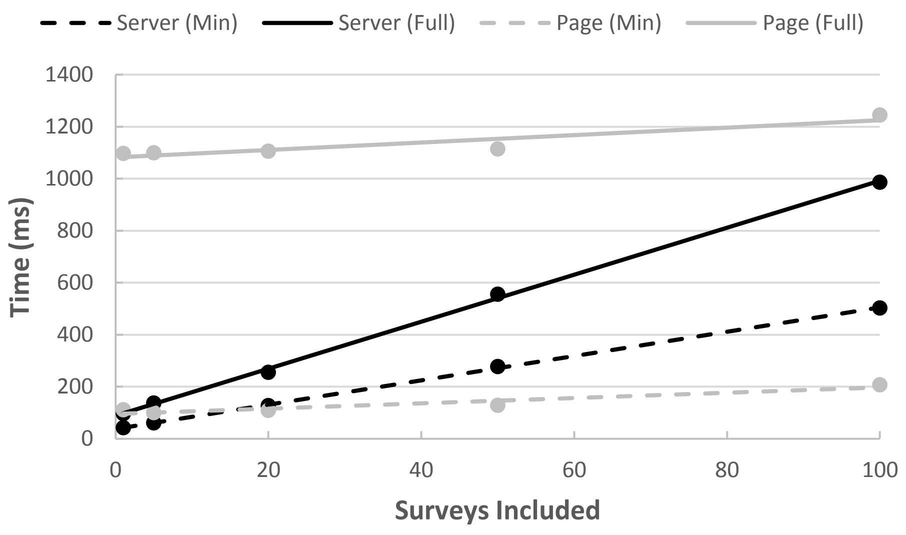


Figure 20. Model Results server and page benchmarks.

Microsoft Excel was used to perform a linear regression for each of the four experiments. Again, we see a clear linear trend in performance. The Server timers demonstrate a steeper slope, as the SQL queries executed for aggregation must work with increasingly more rows as the number of surveys increase. The Page timers demonstrate the gentler slope since the page renders only slightly more data with each increase of surveys. However, the Full Page timer performed slower than any other result due to the additional burden of constructing the HTML canvas graphs.

Note that the trend suggests that for greater than 100 surveys, the Server costs may soon overtake the Page costs due to the large amount of required data processing. This is not likely a problem, as it would be uncommon to run the model against more than 100 surveys at once.

Future Goals

The current implementation provides a robust solution for interacting with oyster data and performing simulations. However, future demand may require more features to be added. For instance, users are often interested in the sum of dead cultch mass and live oyster mass, which gives a good approximation of the reef total shell mass. A system which incorporates the automation of this computation along with useful filtering abilities may be useful.

5. Simulations

The Java version of the model was used to perform various preliminary simulations for the Louisiana Department of Wildlife and Fisheries. The model was parameterized with values from a variety of existing sources. This section briefly describes the results obtained.

Parameterization

Since the model is quite sensitive to its input parameters, much effort was invested in providing it the most accurate arguments possible. The parameters from the Model Logic section are repeated below (Table 8) with the values used for simulations listed. Recall that the Java version of the model treated the entire simulated region as a single entry, so per-station parameters are global variables in this table. Also recall that only brown oyster shell was used; thus, cultch parameters are no longer arrays. The remaining entries are unchanged.

Table 8. Simulation parameter values.

Symbol	**Value**
	General Parameters
$\boldsymbol{t_0}$	8
$\boldsymbol{t_{max}}$	12
$\boldsymbol{v}$	52.859
$\boldsymbol{\lambda_{sack}}$	75
$\boldsymbol{\lambda_{seed}}$	25
	Cultch Parameters
$\boldsymbol{NL}$	0.008
$\boldsymbol{D}$	2200
$\boldsymbol{P}$	0.59
	Oyster Growth Parameters
$\boldsymbol{M_a}$	4×10^{-4}
$\boldsymbol{M_b}$	2.8213
$\boldsymbol{OPS_a}$	1.767×10^{8}
$\boldsymbol{OPS_b}$	-2.926
$\boldsymbol{\lambda_\infty}$	151
$\boldsymbol{k_0}$	0.8
$\boldsymbol{k_1}$	0.09
$\boldsymbol{t_{avg}}$	0.4
	Oyster Mortality Parameters
$\boldsymbol{m_{0,J}}$	1.2
$\boldsymbol{m_{1,J}}$	1.1
$\boldsymbol{m_{0,A}}$	0.51
$\boldsymbol{m_{1,A}}$	0.41
$\boldsymbol{m_{avg}}$	6

The t_0 value is set to 8, as the LDWF stock assessment is performed around August. t_{max} is set to 12, as a one-year forward prediction is desired until the next year's stock assessment. The v parameter is set to 52.859, as fishermen typically bag oysters into 1.5-bushel sacks. The λ_{sack} parameter is set to 75, the minimum length of a harvest-size oyster. λ_{seed} is set to 25, the minimum size seed as defined by the LDWF (LDWF, 2010).

Powell et al. (2012) estimated the density D of cultch to be 2.2 $\mathrm{kg/L}$. Loose shell was packed into a container of known volume. The weighed mass was then divided by D and the container's volume to arrive at the packing coefficient P of 0.59 (Powell, Klinck, & Soniat, 2010).

Shell loss was determined to be approximately 9.5% per year (Powell et al., 2006). By solving for the monthly loss fraction, the NL parameter was set accordingly to 0.008.

The length-to-weight correlation was derived using local sampling data (Eberline, 2012) and a power regression was used to determine the M_a and M_b parameters as 4×10^{-4} and 2.8213, respectively. This yields a fair approximation, with an R^2 value of 0.8441. The figure below (Figure 21) provides a representation of this data.

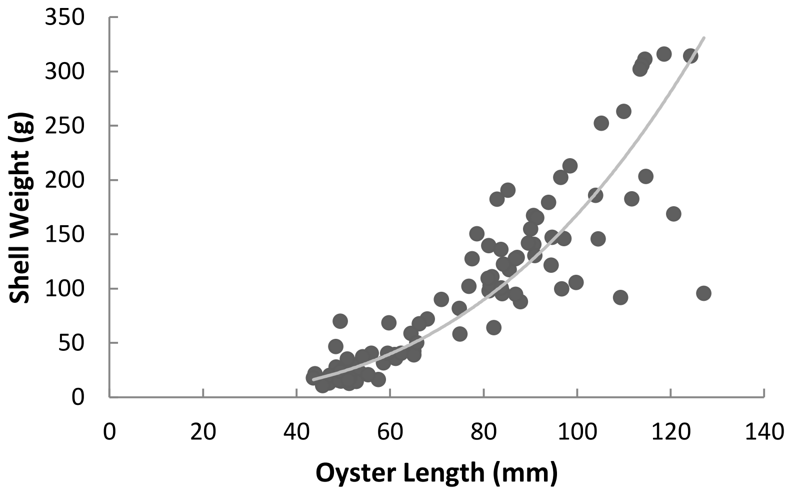


Figure 21. Shell-to-weight plot used to derive OPS parameters.

The oysters-per-sack relationship is well-established. Louisiana oyster data (Hopkins, 1950) was used to arrive at the OPS_a and OPS_b parameters as 1.767×10^8 and -2.926, respectively. Growth rates k_0 and k_1 of were decided by Powell, Klinck, & Soniat. when developing the model, based on spat growth estimates (2010). A particularly high λ_∞ value of 151 was used to simulate fast oyster growth, with a correspondingly low seasonal variation.

Informed by previous mortality analyses of adult oysters (Mackin, 1961, Owen, 1955) and juvenile oysters (Powell et al., 2009), the mortality parameters $m_{0,J}$, $m_{1,J}$, $m_{0,A}$, and $m_{1,A}$ were set to 1.2, 1.1, 0.51, and 0.41, respectively. The m_{avg} parameter was set such that mortality peaks in September and is least in March (Mackin, 1961).

Experiments

Retrospective Analysis

The Java implementation was run against CSA 2 stock assessment data from 1999 to 2009 and a retrospective analysis was made (Soniat et al., 2012). In these simulations, sustainability was decided to be a total reef mass (live oyster shell mass + dead shell) loss not exceeding a magnitude of $\pm$5%, with no loss of dead shell allowed.

Due to the absence of measured cultch samples, an assumption of the initial density was required. In these simulations, the cultch densities were set to 5,000 $\mathrm{g/m^2}$ (Mann et al., 2009) for each station, indicating healthy reefs. The resulting sustainable harvest proved quite sensitive to this parameter. Lower initial cultch density would result in higher estimates of sustainable harvests since fewer oysters would need to remain and die in place to maintain the smaller cultch amount. This realization was a factor in encouraging cultch measurements during future stock assessments.

Data was compared against the current oyster stock and harvest reports from each annual Stock Assessment Report. The figure below (Figure 22) shows the stock, in sacks, from 1999 to 2009. These numbers were used as a basis of comparison for the results obtained by the model.

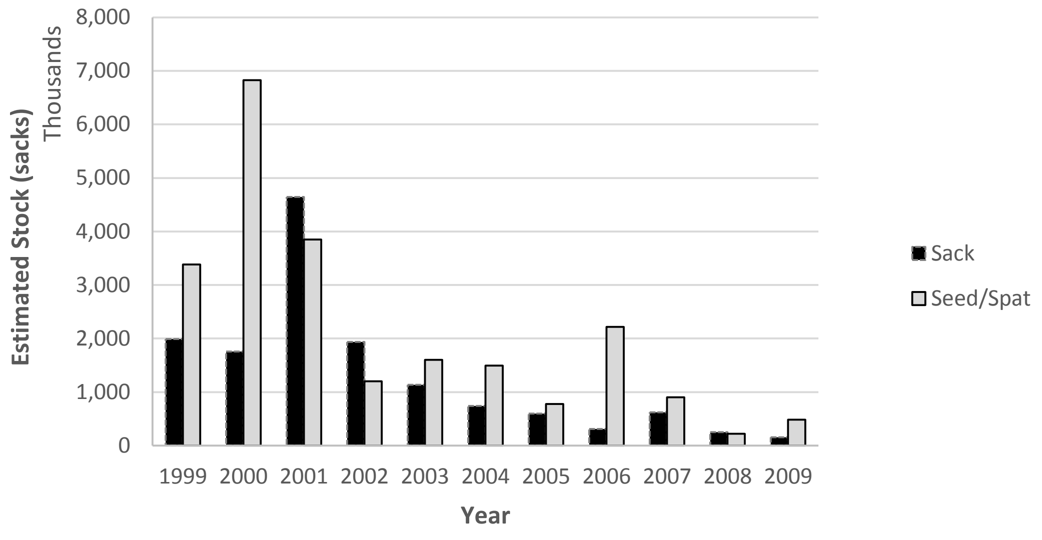


Figure 22. LDWF estimated sack and seed/spat stocks.

The results of the simulations are shown in the table below (Table 9). Quantities are in sacks unless otherwise noted. A * denotes division by zero, when the simulation did not allow any fishing. Note that for some months, the reported fishing was within the threshold of simulated sustainability. Most noticeably, in 2000, only 13.2 of sack oysters were fished than what the model allowed. However, as the time series progresses, the allowed fishing becomes lower, with some years allowing for no fishing at all when oyster numbers were particularly low.

LDWF Sustainable Management

In order to determine the effectiveness of using the model as a basis for management, a trial fishing experiment was decided upon during a 2012 stock assessment workshop at The University of New Orleans. Hackberry Bay, a public oyster station in CSA 3, was chosen as the location to test the accuracy of the model. After running simulations, 7,000 sacks of shell and seed and 4,700 sacks of sack oysters were deemed sustainable. A recommendation was made that boats be monitored daily until the limit is reached, when fishing would be shut down for the season. The results of the experiment should be available after the 2013 stock assessment. The new data will be reviewed and compared against the model-predicted values.

Provided the results are successful, the LDWF will seek to use the model as a quantitative tool to assist in the certification of the Louisiana oyster industry as sustainable.

Future Simulations

Additional simulations should be performed in an effort to improve the model. As the availability of cultch data increases, simulations should be performed that compare model-predicted shell impact with actual surveyed shell impact. Each parameter should be scrutinized further against multiple stations. These comparisons will help determine causes of error and reveal the strengths and weaknesses of the model for proper analysis and future advancements.

Table 9. Results of retrospective analysis simulations.

Year	Sack Fishing					Seed Fishing				
	Estimated Stock	Actual Harvest	Harvest/ Stock (%)	Simulated Harvest	Harvest/ Sim (%)	Estimated Stock	Actual Harvest	Harvest/ Stock (%)	Simulated Harvest	Harvest/ Sim (%)
1999	1,989,508	700,617	35.2	2,012,018	34.8	3,382,040	138,056	4.1	372,441	37.1
2000	1,754,696	403,374	23.0	3,049,175	13.2	6,828,812	201,560	3.0	816,468	24.7
2001	4,642,996	844,898	18.2	3,065,531	27.6	3,851,418	318,490	8.3	385,332	82.7
2002	1,937,488	704,284	36.4	1,004,700	70.1	1,201,982	281,766	23.4	176,001	160.1
2003	1,134,036	286,963	25.3	751,139	38.2	1,598,908	626,320	39.2	87,736	713.9
2004	741,188	535,936	72.3	634,174	84.5	1,497,112	391,072	26.1	86,670	451.2
2005	598,628	271,271	45.3	234,559	115.7	778,146	58,270	7.5	0	*
2006	308,986	183,355	59.3	794,925	23.1	2,215,294	221,134	10.0	53,6576	41.3
2007	619,124	278,580	45.0	414,042	67.3	902,068	347,170	38.5	340,485	102.0
2008	248,786	265,581	106.8	0	*	221,502	154,006	69.5	0	*
2009	156,900	167,614	106.8	476,887	35.1	483,524	165,376	34.2	354,295	46.7

6. Conclusion

This paper presented a formalization of the algorithm proposed by Powell and Klinck. It first described the data sources obtained from The Louisiana Department of Wildlife and Fisheries and detailed how this data was extrapolated into the final input. An abstract description of each component of the model was then described and presented in pseudocode. Next, the implementation history was reported, with emphasis on the data modeling and web presentation of the front-end available at *www.oystersentinel.org*. Finally, some initial applications of the model were discussed, summarizing a retrospective analysis and experiments made in association with LDWF.

The wealth of accurate historical data provided by LDWF lends itself very well to the modeling initiative; however, more simulations need to be performed and analyzed before the effectiveness of the current model can be fully ascertained. Future improvements are expected to add more functionality to the model, which will hopefully result in a functional metric for sustainable oyster harvest in Louisiana.

References

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009) *Introduction to algorithms* (3rd ed.). Cambridge, Mass: MIT Press.

Eberline, B. S. (2012). *Population dynamics of the eastern oyster in the northern Gulf of Mexico.* M.S. Thesis, Louisiana State University, Louisiana State University.

Gunter, G. (1979). Studies of the southern oyster borer, *Thais haemostoma*. *Gulf Research Reports, 6*, 249-260.

Hopkins, S. H. (1950). The inter-relationship of weight, volume, and linear measurements of oysters and the number of oysters per Louisiana sack measure. (pp. 15). College Station, TX: Texas A&M Research Foundation.

Klinck, J. M., Powell, E. N., & Soniat, T. M. (2011) OysterShell.F90 [Fortran source code]. Retrieved May 23, 2011.

LDWF. (2010). *Oyster stock assessment report of the public oyster areas in Louisiana seed grounds and seed reservations*. Baton Rouge, LA: The Louisiana Department of Wildlife and Fisheries.

Mackin, J. G. (1961). Oyster disease caused by *Dermocystidium marinum* and other microorganisms in Louisiana. *Publication of the Institute of Marine Science, 7*, 132-299.

Mann, R., & Powell, E. N. (2007). Why oyster restoration goals in the Chesapeake Bay are not and probably cannot be achieved. *Journal of Shellfish Research, 26*, 905-917.

Mann, R., Southworth, M., Harding, J. M., & Wesson, J. A. (2009). Population studies of the native eastern oyster, *Crassostrea virginica*, (Gmelin, 1791) in the James River, Virginia, USA. *Journal of Shellfish Research, 28*(2), 193-220. doi: 10.2983/035.028.0203

Owen, H. M. (1953). Growth and mortality of oysters in Louisiana. *Bulletin of Marine Science of the Gulf and Caribbean, 3*, 44-54.

Powell, E. N., & Klinck, J. M. (2007). Is oyster shell a sustainable estuarine resource? *Journal of Shellfish Research, 26*(1), 181-194. doi: 10.2983/0730-8000(2007)26[181:iosase]2.0.co;2

Powell, E. N., Klinck, J. M., Ashton-Alcox, K., Hofmann, E. E., & Morson, J. M. (2012). The rise and fall of *Crassostrea virginica* oyster reefs: the role of disease and fishing in their demise and a vignette on their management. *Journal of Marine Research, 70*(2-3), 505-558.

Powell, E. N., Klinck, J. M., Ashton-Alcox, K. A., & Kraeuter, J. N. (2009). Multiple stable reference points in oyster populations: implications for reference point-based management. *Fishery Bulletin 107*, 133–147.

Powell, E. N., Klinck, J. M., & Soniat, T. M. (2010). *Model Documentation*. Unpublished manuscript.

Powell, E. N., Kraeuter, J. N., & Ashton-Alcox, K. A. (2006). How long does oyster shell last on an oyster reef? *Estuarine, Coastal and Shelf Science, 69*(3-4), 531-542. doi: 10.1016/j.ecss.2006.05.014

Soniat, T. M., Klinck, J. M., Powell, E. N., Cooper, N., Abdelguerfi, M., Hofmann, E. E., . . . Quaddoura, F. (2012). A shell-neutral modeling approach yields sustainable oyster harvest estimates: a retrospective analysis of the Louisiana state primary seed grounds. *Journal of Shellfish Research, 31*(4), 1103-1112.

Vita

The author was born in Hattiesburg, Mississippi. He obtained his Bachelor’s degree in computer science from The University of New Orleans in 2011. After graduation, he worked as a graduate assistant under Dr. Thomas Soniat of the UNO biology department, working on the Oyster Sentinel project as part of his computer science thesis.