DESCRIPTION OF AN INDIVIDUAL BASED FISHERIES SIMULATION OPERATING MODEL (IBM)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Jeremy McKenzie*

*Nokome Bentley*

*Richard Bian*

04-04-2016 Ver 1.1

[1. What is an IBM? 3](#_Toc455649424)

[2. IBM Specifications 4](#_Toc455649425)

[2.1. Core dynamics 4](#_Toc455649426)

[2.2. Spatio-temporal structural definitions 5](#_Toc455649427)

[2.2.1. Within time-step model process order 5](#_Toc455649428)

[2.2.2. Within time-step individual fish process order 6](#_Toc455649429)

[2.3. IBM run phases 6](#_Toc455649430)

[2.4. Model population dynamics 7](#_Toc455649431)

[2.4.1. Spatial and temporal structure 7](#_Toc455649432)

[2.4.2. Recruitment 7](#_Toc455649433)

[2.4.3. Growth 7](#_Toc455649434)

[2.4.4. Natural mortality 8](#_Toc455649435)

[2.4.5. Maturity and sex processes 8](#_Toc455649436)

[2.4.6. Movement 8](#_Toc455649437)

[2.4.7. Gear selectivity 8](#_Toc455649438)

[2.4.8. Catch removals (harvest) and tag recovery 9](#_Toc455649439)

[2.4.9. Tag release 10](#_Toc455649440)

[2.5. Individual fish attributes 10](#_Toc455649441)

[2.6. Model speed and parallelisation potential 11](#_Toc455649442)

[2.7. Population Scaling Options 11](#_Toc455649443)

[2.7.1. Model individual scaling 11](#_Toc455649444)

[2.7.2. Agent based scaling 11](#_Toc455649445)

[3. IBM component functions + test and validation results 13](#_Toc455649446)

[3.1. Individual growth path model validation 13](#_Toc455649447)

[3.2. Harvest and tag recovery process routines and functions 15](#_Toc455649448)

[3.2.1. Harvest process 15](#_Toc455649449)

[3.3. An Agent based modelling approach incorporating tagging 18](#_Toc455649450)

[3.4. Casal model SSB estimates from “true” SNA 1 3 area IBM 18](#_Toc455649451)

[3.4.1. Model specifications 18](#_Toc455649452)

[3.4.2. Model fits 19](#_Toc455649453)

[4. References 21](#_Toc455649454)

[5. APPENDICES 21](#_Toc455649455)

# What is an IBM?

This document provides a technical description of an Individual Based Model (IBM) software tool for building fish stock catch history models. The main use of this model is for evaluating the performance of age or length-based stock assessment models and for building “real-world” construct models for use in Management Strategy Evaluation (MSE).

The IBM modelling tool described in this document was specifically built to be the operating model component of Management Strategy Evaluation for the Northern New Zealand snapper stocks (SNA 1 & SNA 8). It is capable of simulating the complex spatial and mixing dynamics evident in these stocks and for generating observational data required by the current assessment models, specifically: mark-recapture (tagging); commercial catch at-age and length, CPUE indices, and recreational harvest estimates. Although capable of complex configurations simple parameterisations are also supported. We believe the base IBM software has a wide range of management applications beyond MSE, for example the exploration of optimum yield, i.e. yield per recruit (YPR), or testing stock assessment model performance. The base model c++ code resides in the public domain and can be downloaded from the following repository: https://github.com/trophia/sna1 .

The main advantage of IBMs is they a free for formal partition space limitations of more standard models. Partitions such as length, age, spatial location, growth dynamics, tag status, are all defined as attributes of the individual fish. The frequency of fish in the IBM for any point and any desired partition structure is simply derived by summing across fish attribute classes, e.g. length frequency of fish in area A that are tagged. At each time step the fate of each fish is governed by defined probabilities in accordance to their inherent/unique set of attributes (e.g. move, grow, or caught by a fishery).

The disadvantage of IBMs is they can be computationally slow when the total number fish in the model exceeds 100,000. A common approach to avoid memory and speed issues is to have each individual fish unit represent a group of fish that share the same attributes and fates (e.g. location age length, sex, growth). Scalars (weight) are then used to transform model outputs and observations to reflect “real world” biomass, catches, tag recoveries, and harvest targets.

Advantages:

* True length and age based model
* Retains attributes of a particular cohort of fish through time
* Allows incorporation of greater partition space than other modelling approaches
* Efficient at modelling complex spatial complexity and movement

Disadvantages:

* Number of individuals in the model computationally limiting (i.e. > 100,000)
* May need to use scalers to model large populations individuals being representative of a group of fish that share the same fates and attributes
* Speed issues means IBMs not easily used for estimation.

Main uses:

* As an operating model for assessing the performance of more “tradition” age or length based assessment models (estimators) and as a “real-world” construct in Management Strategy Evaluation simulation modelling.

# IBM Specifications

## Core dynamics

The stock universe is populated by individual fish that are born, grow move, are tagged and die across the model "universe" spatio-temporal partition space. The status of the stock and its spatial distribution can be derived for any time step by summing the individuals.

Harvest is removed pursuant to a number of defined fisheries. These fisheries are allocated quota for each spatio-temporal partition with exploitation subject to quota, MLS, and gear selectivity constraints. Tags are also recovered by the harvest process with tag detection rules applying.

Tags are released in specified spatio-temporal partitions in accordance with a given selectivity and initial survival rules

Individual movement across the spatio-temporal partition space is governed by predefined movement matrices or advection dispersion rules the underlying dynamics being either Markovian or Home Fidelity.

A recruitment process governs the number of fish recruiting to specific spatio-temporal partition space. The individual recruits coming from a defined length frequency distribution all being the same initial recruited age e.g. 1 year old, 3 months post settlement etc.

The model dynamics are governed by over-all population processes parameterised as per a "typical" stock assessment population model, i.e. mean recruitment, natural mortality, mean growth, total harvest rules, gear selectivities, discard mortalities etc. The recruitment process can made to confirm to any defined stock recruit dynamic. Similarly, natural morality rules can be simple or complex (e.g. age-area based), like-wise mean growth. Movement dynamics are defined at the population level and again can be defined in a number of complex ways. A critical requirement is the net individual fish dynamics must confirm to required (specified) population dynamics. For example the mean growth rates of all fish in the IBM should be consistent with the expected growth model parametrisation.

## Spatio-temporal structural definitions

**Class: Stock-management-area-partition**

This is a stock spatial management unit it is the unit we are most interested in and for which quotas are set; stock assessments undertaken and management decisions made.

**Class: Region-partition**

This is the base spatial resolution unit in the model i.e. all model all harvest and movement processes act only at this spatial level. The stock Area is thus made up of one or more regional cells such that stock or area partition statistics are derived by summing across regions.

**Class: Annual cycle**

The IBM model time cycle is always annual and most reported statistics are reported as end-of-year.

**Class: Time step**

The annual cycle can be divided up into 1 or more discrete time-steps (e.g. season, month day etc). The fates of all fish in the model are recalculated in each time-step, therefore the number of annual time-steps is a direct multiplier on model speed.

### Within time-step model process order

The key model processes occurring with each time step are as follows:

*Individual fish update process:*

* Update individual fish attributes (requires model to step through every live fish)

*Harvest and tag recovery process*

* Accumulate current model population statistics (e.g. numbers a length/age by area etc) and derive length-base harvest probabilities by gear and region
* Harvest process: random draw to the required number of harvested fish from the population and recover tags.

*Tagging process* (process occur occurs during a designated tag release time step)

* Accumulate current model population statistics (e.g. numbers a length/age by area etc) and derive length-base tagging probabilities by gear and region
* Tagging process: random draw to the required number of tagged fish from the population.

*Model output process*

* Model output process: Accumulate end of time step model population statistics (e.g. numbers a length/age by area etc) and generate required output statistics includes observational data for input to and estimator model.

### Within time-step individual fish process order

1. Recruitment (at initial age = 0)
2. Ageing (being: initial age + time\_step increment time)
3. Maturation (size
4. Migration (in accordance with either Markovian or HF rules)
5. Growth (amount of growth occurring, i.e. growth increment, over time\_step)
6. Natural mortality (may be specific the Sub-area-spatial-partition the fish now resides)
7. Anthropogenic mortality other than fishing (may be specific the Sub-area-spatial-partition the fish now resides)
8. Harvest induced mortality and tag recovery (specific the Sub-area-spatial-partition the fish now resides)
9. Tag release events (specific the Sub-area-spatial-partition the fish now resides)

* release mortality (specific to capture method)
* reset growth rate (specific to capture method)
* Tag loss rate (specific to tag type)

## IBM run phases

1. Burn in period: necessary to achieve virgin equilibrium conditions
2. Catch history period: A catch history is applied to achieve the required MSE starting point conditions.
3. MSE period: time steps over which the MSE simulations takes place.

## Model population dynamics

Like most fisheries models the IBM is mostly specified by population level parameterisations (e.g. mean growth, mean recruitment, natural and fishing mortality) and parameters and functions are required to specify model population dynamics, e.g. R0 (mean recruitment), M (natural mortality), growth model and parameters (e.g. vB model).

It is possible (desirable) to introduce complexity or variability into the IBM simulation process at the population level. Examples being: temporal change in R0, growth and natural mortality.

For the IBM to “work” as intended introduced stochasticity at the individual level must average or sum to be consistent with expected higher level parameterisations, trends and patterns.

### Spatial and temporal structure

Fundamental to the model structure these are expressed in the model as location and age individual fish attributes.

### Recruitment

The model is specified as to mean recruitment (R0) it also can accommodate rules as to how this might vary and change through the history of the fishery, examples being:

* Any desired stock recruit relationship and or dispensation rules
* A specified vector of year class strengths (YCS)
* Function/rules can be used to construct any desired systematic tend or change in R0 (e.g. regime shift simulation)

R0 is the 0 year-old fish entering the model in time step *x* of annual cycle *y*. However, after the new recruits are added they and all other fish in the model are incremented in age by adding the time-step time value to their current age; i.e. for new recruits their age becomes 0 + time-step. So the effective age at recruitment is 0 + time-step, e.g. if there is only one annual time step R0 = the number of 1 year-olds entering the population.

### Growth

The growth function specifics how much to increment the length of a fish over the given time-step duration. The model is currently configured for von Bertalanffy growth (linear increment) but other models or growth rules could easily be incorporated.

Growth stochasticity at the population level is achieved through random selection of growth parameters at the individual fish level, i.e. each fish has its own growth path (i.e. an individual’s growth path that is set at recruitment and growth for the individual after that is deterministic. The model is able to accommodate fish changing their growth paths as a consequence of say tagging or moving to a different area. Effectively the IBM is capable of simulating simple and complex growth dynamics, and is able to replicate the effect of local or regional growth disturbance through time.

Uses a VB (linear increment) approach where *K* and *Linf* are generated as lognormal random deviates each fish *y* is assigned its own growth path as specified by *Ky*  and *Linfy* (Appendix 1).

The change in length (i.e. increment) for a fish of length over is given by:

Length weight conversions are currently achieved by way of a simple function. Again the model is capable of introducing stochasticity on this at the individual level.

### Natural mortality

Like growth natural mortality is set at the population level and this currently implemented as a single parameter but the use of more complexity age specific mortality functions is possible. Like growth, stochasticity can be introduced at the population level, with every fish having a slightly different natural mortality probability.

### Maturity and sex processes

Again, rules for this defined at the population level with the option to introduce stochasticity at the individual level. Currently maturity is implemented as a simple age-based ogive; being a snapper model there is no sex partition (i.e. sex is not set as an individual fish attribute).

### Movement

The rules and processes that govern movement are step at the population level and both home fidelity and Markovian dynamics are possible. Movement is specified using *n* x *n* transition matrices where n is the number of regional-partitions in the model. Also like growth stochasticity can be introduced by assigning each fish a random variate of the population level movement matrix. i.e. each fish moves in accordance to its own movement matrix which is fixed but specific to that fish.

### Gear selectivity

Gear selectivity is defined at the population level as functions (ogives) or matrices. The effect of selectivity is implemented at the individual level being the probability of the individual fish of a given length being captured by specific gear.

Three specific selectivity curves need to be defined for each gear in order to accommodate harvest, discard mortality and tag selection dynamics in the model (**Figure 1**):

1. *Capture selectivity:* governs the total proportion of the population vulnerable to the gear: i.e. determines how many fish in total are caught by a particular gear for a given TACC (used primarily for the tag release and recovery process)
2. *Landed selectivity:* selectivity of the retained or landed portion of the catch i.e. selectivity of the observed landed catch (TACC).
3. *Discard selectivity:* Off-set of the total selectivity, governs the mortality of fish caught then discarded by the gear (unseen mortality) the amount of offset being proportional to the discard mortality rate. Note actual total motility is defined by combining this curve with the landed selectivity curve.



**Figure 1: Selectivity ogives used by the IBM to implement; tag selection; harvest mortality; discard mortality**

### Catch removals (harvest) and tag recovery

Harvest (fishing mortality) can be specified either as a TACC or as an exploitation rate (F). These need to be assigned by the user specific to gear time-temp and regional-strata. Effectivity the removal or harvest “rules” govern how a given management strategy is implemented in the model. Harvest is implemented at the individual level as the probability of a given fish in a given region at a given time being in a given catch to a given method.

Tag recovery is also implemented in the IBM harvest cycle, in accordance to tag detection rules and the amount of catch scanned; this effected as 1 probability being:

P[tag recovered] = P[tag in catch]\*P[catch scanned]\*P[tag detected]

### Tag release

Tagging is functionally implemented in the IBM as harvest but a run as separate processes so as to allow the tag release design to differ from the harvest strategy. Tagging occurs after harvest in the model to allow for tag mixing (through movement) to occur, i.e. tagged fish cannot be recovered in the same time step they are released. Instead of harvest a prescribed number of tags are allocated to each gear to each region in a given time step. Tagging at the individual level is governed by the probability of a given fish in a given region at a given time being tagged by a given method which has been allocated a given number of tags.

## Individual fish attributes

Individual fish in the model are defined by their attributes some of which are updated in each time step. In current model fish have the following stored attributes:

* current age
* maturity (age dependant)
* sex (not used)
* current region
* recruitment region (only used in home fidelity movement)
* individual growth path parameters
* individual movement probability matrix
* current length

Tagged fish attributes (discarded if tag is lost)

* Tag-number
* Tag release region (only recorded when fish is tagged)
* length at time of tagging
* time-step, region and method of harvest (only recorded if fish is harvested)

## Model speed and parallelisation potential

The fish update process is main time limiting step, however, because the fates of all individual fish are independent it is relatively straight forward to parallelise this aspect of the code. The current IBM code has been set up to allow multi-threading (parallelisation). Without parallelisation the current practical limit on a single processor is approximately 100,000. Using NIWA’s super computer the practical limit increases to around 30 million.

## Population Scaling Options

Two options for increasing the effective number of fish in the model so as to achieve equivalence to a typical stock e.g. 100,000 -> 50,000,000 are:

1. Individual scaling;
2. Agent based modelling.

### Model individual scaling

The IBM model gets round the problem of having to create more than 100,000 fish to represent a typical fish stock like SNA 1 through the use of scaling. In real terms this means that the “individual” fish in the model is not a true individual but a representation of a group of *n* fish that all share the same attributes and fates e.g. get caught by the same method in the same region in the same time-step.

Having fewer than the “true” number of fish in the IBM reduces its precision particularly in tagging scenarios where model tags instead represent units of *x* tagged fish not individual tags. The overall effect of scaling is that stochasticity (variability) across the model partition space may be overstated. This is unlikely to be an issue with less complex models but will increase as complexity increases i.e. as the number of model individuals per model partition reduces.

### Agent based scaling

The main difference with the agent-based scaling to standard scaling is that the unit number of fish can change through time in response to fishing pressure or tagging. In some models when the number of fish represented by an agent gets small the model will combine agents with like characteristics. The price of more complex agent-based model dynamics is increased computational overhead. In some applications the additional accounting required to keep track of agent and their individual number sequences may result in little or no computation time savings over a true individual based model.

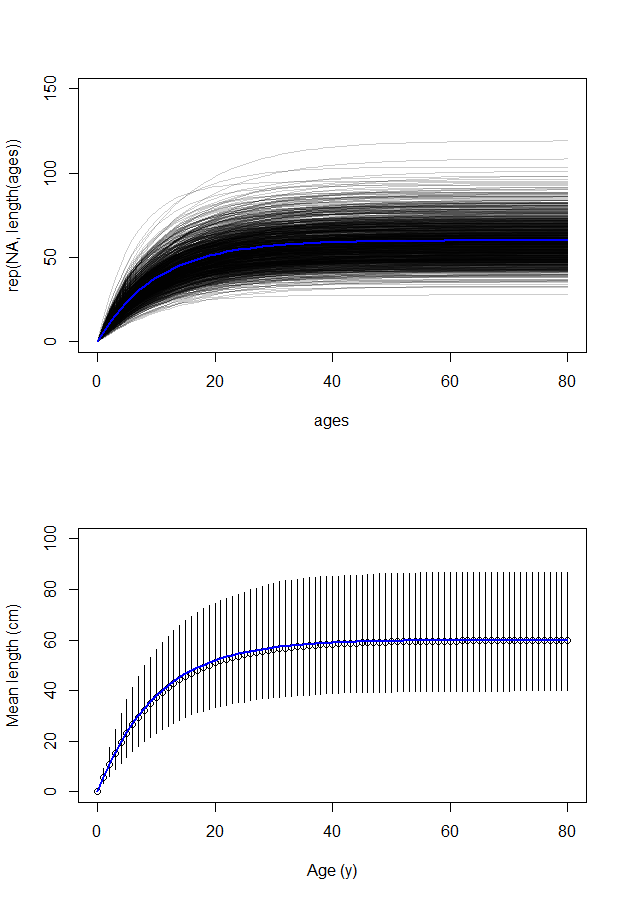
We believe the strongest justification in using an agent-based approach over basic scaling is when tagging needs to be a feature of the model. In tagging scenarios typically the number of tags recovered from the population relative to its population size is low, e.g. order hundreds-thousands relative to millions - 10s of millions in the pop. Scaling applied at the tag recovery level may significantly limit the “effective” resolution power of the observed number of model tag recoveries, e.g. given a scale factor of 50 a recovery of 500 implied tags will have an effective estimation power equivalent to that of only 10 true individual tags. The presence of fine scale partitioning in the model where the recovery expectation is only 1 or two tags per partition will exacerbate the problem as scaling is likely to introduce significant bias into the simulations.

A full description and simulation test results for an agent based approach incorporating tagging is given in Section

# IBM component functions + test and validation results

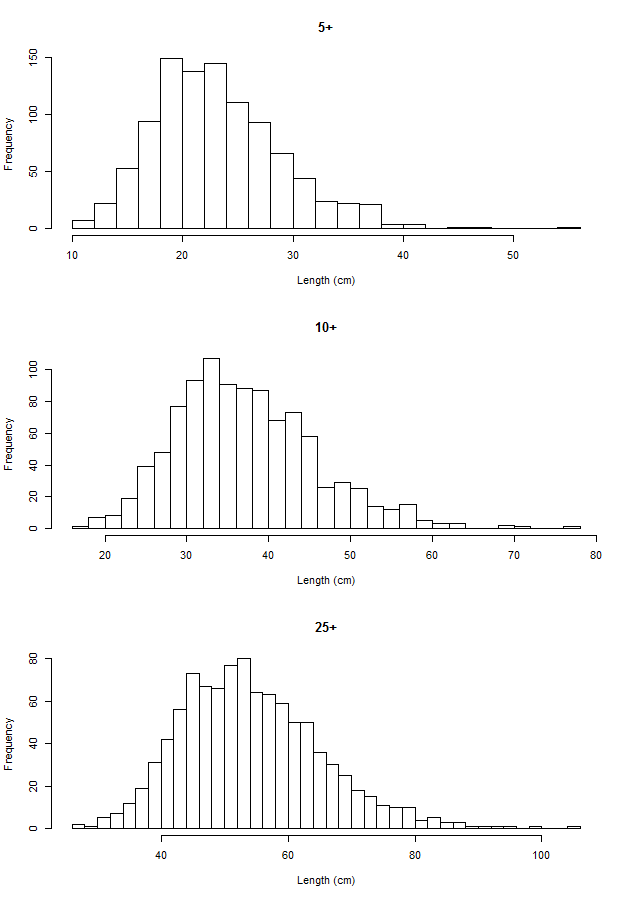
## Individual growth path model validation

One thousand individual growth paths were generated randomly from a log-normal distribution of *K* and *Linf* with a cv of 0.2 and respective means 0.1 and 60. The model mean length at-age as derived from the individual bootstrap growth paths corresponded highly to the actual vB curve (*K* = 0.1 and *Linf* = 60 Figure 2).



**Figure 2: Individual growth paths from 1000 *K* and *Linf* random draws (top graph). Blue line shows true vB curve for *K* = 0.1 and *Linf* = 60. Dots (bottom graph) are the boot-strap means vertical lines are bootstrap 95% confidence intervals.**

The log normal pattern was evident in the model generated length distributions by age (Figure 3).



**Figure 3: Model length frequencies for ages 5, 10, and 25 as derived from the 1000 random vB growth paths**

## Harvest and tag recovery process routines and functions

### Harvest process

This process applies a quota allocation or exploitation-rate as a random removal of individuals from the population in accordance with gear selectivity and discard survival definition rules. The process occurs after the IBM has implement growth, movement, and natural mortality at the individual level i.e. after stepping through all the individual fish.

The general process of how the IBM removes a fixed TAC is as follows:

1. Sum across all individual fish to obtain a population length-frequency for a given area
2. Apply the appropriate total mortality selectivity function for a given fishing method to the population to derive the “expected” proportional length frequency of all fish taken by that method in the time-step.
3. Via a length-weight relationship work out the total number of fish required in the above MLS portion of the method catch total length frequency required to achieve the designated TAC thus also derive the discard-dead length frequency.
4. Apply a random draw process to remove the required number individual fish from the population from a given length-bin.

Having to step through all the individual fish a second time to apply a capture probability is avoided by simply taking a random draw of the “exact” number of required fish from the population from within each length bin without replacement.

The process from removing fish pursuant to an exploitation rate largely as above except the designated F is applied as a multiplier to selectivity curve which is the applied to the available population lf to derive the expected length frequency of the catch.

The catch removal process has two key issues to resolve:

1. Derivation of multiple method catch length frequencies when these methods are fishing simultaneously.
2. What to do when the designated catch exceeds the available catch

The issue of how to apply fishing mortality from multiple fisheries is a classic stock assessment modelling problem for which anyone has yet to find a truly unbiased solution. The way our IBM implements natural and multiple fishery mortalities is likely introduce small levels of bias but to no greater degree than most other fisheries stock assessment and simulation models. Our approach is to derive the expected length frequency of each method catch independently in accordance with the process described above, then randomly draw without replacement the exact predetermined number of fish each method requires from each length bin.

The application of a fixed harvest (as opposed to fishing mortality rate) has potential to remove all fish from a given area of a given length from the model (i.e. when TAC area length >= Population area length). In traditional SA estimation models this contingency is typically dealt with through the application of likelihood penalties such that the estimation process “avoids” parameterisations that cause the fishery to crash. A different approach is needed for dealing with over-exploitation occurrences in simulation modelling. Basically there are two model options when the designated catch is higher than the available catch:

1. Stop the simulation and alert the user that the TAC cannot be taken
2. Take out less than the specific catch and generate a warning that this has occurred.

Modelling options to deal with instances of over-harvest get progressively more complex under multiple fishery scenarios and when overfishing is occurring relatively to only one or two length classes.

**Scenario 1 over-harvest by one method across all length bins**

Presupposing the user does not want to allow fishing to collapse the stock, then under the simple scenario the user sets a cap on maximum proportional remove i.e. Umax.

**Scenario 2 combined over-harvest by 2 or more methods**

Under this scenario overfishing occurs at the level of the combined catch from two or more methods whereas the individual fishery catches do not there is a second option to prevent the cumulative catch exceeding Umax using the self-limiting properties of the Baranov catch equation.

Let UmaxA be the annual proportion of the stock (N0) removed by method A

Let UmaxB be the annual proportion of the stock (N0) removed by method B

Then the instantaneous fishing rate (FA) of fishery A is given by

The annual catch taken by fishery A (Acatch) can thus be derived as

Likewise the instantaneous fishing rate (FB) of fishery B is given by

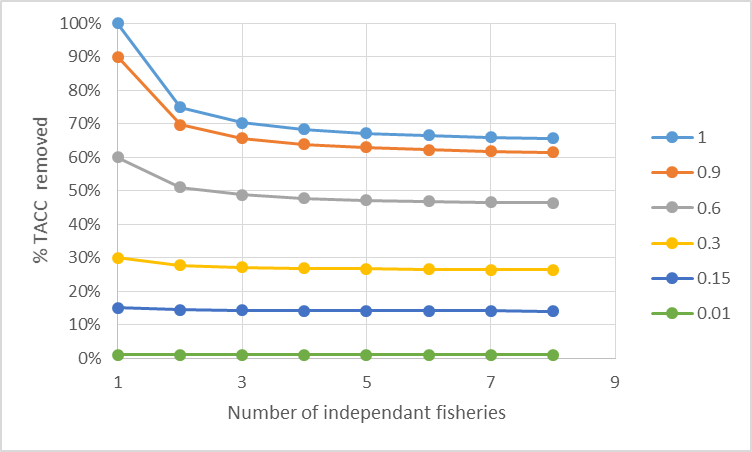
The annual catch taken by fishery B (Bcatch) can thus be derived as

Using the Baranov catch formulation the total annual catch from fisheries A & B operating at the same time is given by

The A proportion of the catch being

Under the Baranov catch equation the derived annual catch for a single fishery with a given fixed F is always less if other fisheries are operating in the same area at the same time.

Therefore using the Baranov formulation is a way of avoiding over-exploitation in the simulations. The limitation is that the TACC is never actually taken even when the Umax limits are low (i.e. <0.15; Figure 4)



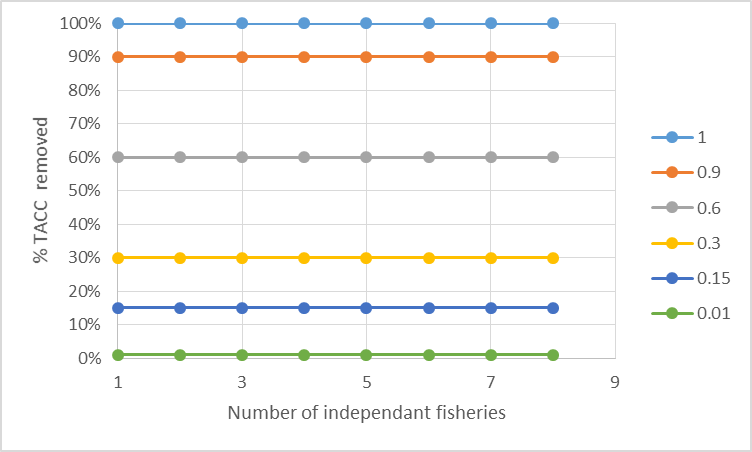
**Figure 4: Reduction in % TACC removed by model via the Baranov catch equation as number of fisheries increases relative to varying Umax removal limits.**

Not using the Baranov ensures the full TACC for each fishery is taken in the model up to the maximum limit of Umax as follows (Figure 5).

When

Then

and



**Figure 5: Reduction in % TACC removed by model via setting fishing removals proportional to Umax**

## An Agent based modelling approach incorporating tagging

Each agent is defined as a consecutive number sequence these numbers correspond to each of the “true” individual fish it represents. Population processes including tagging are implemented as a series of random number draws from the true total population number an agent gets selected if any one of its sequence numbers is chosen in the draw, obviously the more fish an agent represents the greater the probability of it being selected.

### Agent-based tagging process

The agent-based tagging approach we propose involves splitting out tagged fish from agents as true individuals. The tagged fish takes all agent attributes but after which is “free” to act independently. If an agent losses one or more fish to tagging the number of fish the agent represents is decremented but the loss amount. The tagging process will randomly draw a number from the true total population this number will fall within the sequence range of only one agent. Regardless of the tagged fish actual number in the agent’s sequence of numbers it gets reassigned the agents last or highest sequence value. The agent is then resequenced its overall total reduced by one. The tag resequencing process has no effect on the number sequences of any of the other agents in the population, as the splitting off of a tagged fish and resequencing is only changing the tagged agent numbering the tagged fish now accounting for the last number in the old agent sequence.

### Agent-based harvest and tag recovery process

After tagging the model population is made of a number of true individual tagged fish (represented in the pop as a single sequence number) and a given number of agents representing variable sequences of identical fish dependent upon the number of tagged fish drawn from their original recruited value.

Due to the presence of individual tagged fish in the population harvest removals and associated tag recovery are implemented in two phases as follows.

1. Tag recovery harvest:

Once the number of fish removals from the population is defined the first of two ‘lottery’ process is then implemented. Harvest number draws that fall into non-tagged agents are ignored only the selected tagged fish are removed.

1. Untaged fish harvest:

The harvest target is then decremented by the actual number of tagged fish selected in phase one. A series of repeat random draws are then drawn from only the untagged agent component of the population until the harvest target is reached. Ideally if the final draw selects an agent representing more fish than is required to meet the harvest target the agent should be split returning to the sea the surplus component. The resulting increasing level of agent fragmentation and associated computational overhead is unlikely to warrant doing this it being probably simpler just to increase the “known” catch removals accordingly.

### Harvest process steps with examples

1. Set population size (e.g. 100)
2. Set agent maximum size (e.g. 15)
3. Divide population into agents and assign unique number sequences to each.

The last agent may have less the maximum agent number so as not to exceed the stated population size. Example, agent sequence numbers based on a population of 100 and max agent size 15 note the last agent has only 10 fish.

agent number S\_seq E\_seq tag

1 15 1 15 0

2 15 16 30 0

3 15 31 45 0

4 15 46 60 0

5 15 61 75 0

6 15 76 90 0

7 10 91 100 0

1. Set number of fish to tag x (e.g. 10)
2. Randomly draw x sequence numbers without replacement from population.
3. Create x individual tagged fish from agent sequences resequencing and reducing agent sizes accordingly. Example of agent sequences after tagging 10 random fish

agent number S\_seq E\_seq tag

1 13 1 13 0

2 1 14 14 1

3 1 15 15 1

4 14 16 29 0

5 1 30 30 1

6 13 31 43 0

7 1 44 44 1

8 1 45 45 1

9 15 46 60 0

10 14 61 74 0

11 1 75 75 1

12 12 76 87 0

13 1 88 88 1

14 1 89 89 1

15 1 90 90 1

16 9 91 99 0

17 1 100 100 1

1. Set harvest number y (e.g. 50)
2. Randomly draw y sequence numbers without replacement from population.
3. Remove from population only the tagged fish selected.

E.g. 6 tags recovered

agent number S\_seq E\_seq tag

3 1 15 15 1

5 1 30 30 1

7 1 44 44 1

11 1 75 75 1

13 1 88 88 1

15 1 90 90 1

1. Reduce harvest target by the number of tag recoveries (e.g. 50-6=44)
2. Randomly draw and remove agents until untagged harvest total reached (may require splitting last agent to remove exactly the required number of fish.
3. Combine remaining tagged and untagged agents and resequence.

agent number S\_seq E\_seq tag

1 1 1 1 1

2 1 2 2 1

3 1 3 3 1

4 1 4 4 1

5 14 5 18 0

6 14 19 32 0

7 12 33 44 0

8 6 45 50 0

The Petersen population estimate based on the above simulation gives

Number of tags release = 10

Amount of catch taken = 50

Number of tags recovered = 6

Petersen population estimate using the Chapman bias correction gives = 79

### Effect of scalar magnitude on population estimate precision and model run times.

The R agent-based tag simulation routine given in Appendix 2 was used to explore the time and estimate precision trade-offs. One thousand bootstrap model runs were undertaken for each scalar scenario; estimate precision (root mean error CV [RMCV]) are given for each (Table 1).

The level of precision and bias on the model Peterson population estimates appear to be largely independent on the choice of agent scalar value even when the scalar chosen was equivalent to 99% of the population (Table 1). Simulation model runtimes increased exponentially as the scalar value approached unity (i.e. a true individual based model) (Table 1).

**Table 1: Relationship between model runtime and estimate precision and the relative magnitude of the agent scalar based on 1000 replicate bootstrap model runs.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **agent max** | **True pop** | **Tags** | **catch** | **estimate mean** | **bias** | **RMCV** | **boots** | **runtime (minutes)** |
| 1 | 100 | 10 | 50 | 100.58 | 0.01 | 0.41 | 1000 | 4.9 |
| 5 | 100 | 10 | 50 | 99.85 | 0 | 0.32 | 1000 | 0.6 |
| 10 | 100 | 10 | 50 | 100.71 | 0.01 | 0.35 | 1000 | 0.37 |
| 15 | 100 | 10 | 50 | 102.22 | 0.02 | 0.4 | 1000 | 0.32 |
| 25 | 100 | 10 | 50 | 99.5 | -0.01 | 0.35 | 1000 | 0.26 |
| 50 | 100 | 10 | 50 | 101.6 | 0.02 | 0.39 | 1000 | 0.25 |
| 99 | 100 | 10 | 50 | 99.26 | -0.01 | 0.31 | 1000 | 0.24 |

## Casal model SSB estimates from “true” SNA 1 3 area IBM

### Model specifications

An IBM with similar spatial structure productivity dynamics and catch history to SNA 1 was constructed. The model comprised three stock areas comparable to East Northland (ENLD, Hauraki Gulf (HAGU), and Bay of Plenty (BPLE) but did not incorporate movement between stock areas. Four harvest methods specified in the model were: Bottom Long Line (BL); Bottom Trawl (BL), Danish Seine (DS) and recreational line (REC). The model covered the period 1900 to 2017 with similar method-area catch histories to those used in the 2013 SNA 1 assessment. The model stock area B0 values in tonnes were: ENLD 100,000; HAGU 200,000; BPLE: 100,000. The model was configured to allow variable individual growth but there was no variation on year class strength. The models main parameter values are given in Table 1. Gear selectivities were length-based and the same in all areas as was growth (Table 1).

**Table 2: IBM fixed parameters**

|  |  |  |
| --- | --- | --- |
| Natural mortality | *m* | 0.075 y-1 |
| Stock-recruit steepness (Beverton & Holt) | *h* | 0.85 |
| Proportion mature |  | 0 for ages 1–4, 0.5 for age 5, 1 for ages > 5 |
| Length-weight [mean weight (kg) = *a* (length (cm))b] |  | *a* = 4.467 × 10-8, *b* = 2.793 |
| von Bertalanffy growth parameters |  |  |
|  | *L∞* | 60.0 |
|  | *k* | 0.1 y-1 |
| Coefficient of variation on vB length at age | *cv* | 0.2 |
| Selectivity Long Line {Length based double normal} | *a, σL, σR* | 30.47, 2.76, 1000 |
| Selectivity Bottom Trawl {Length based double normal} | *a, σL, σR* | 29.39, 2.35, 29.15 |
| Selectivity Danish seine {Length based double normal} | *a, σL, σR* | 31.63, 3.13, 20.54 |
| Selectivity Rec {Length based double normal} | *a, σL, σR* | 30.11, 1.97, 15.27 |
| MLS |  | recreational 30 commercial 25 |

Observational data generated by the IBM as input to a similar configured casal model were:

* Three longline relative abundance indices (one for each area) covering the entire history of the model.
* Proportional catch at-age observations for all methods in all areas over the entire history of the model.

The casal estimator model was aged-based but otherwise structural identical to the IBM and configured with the same fixed parameters given in Table 1 with the exception of the selectivity parameterisations.

The casal model inputs were:

* Catch history
* IBM generated long line abundance indices
* IBM Catch at-age observations
* IBM multinomial error on catch at-age

Estimated in the Casal model were:

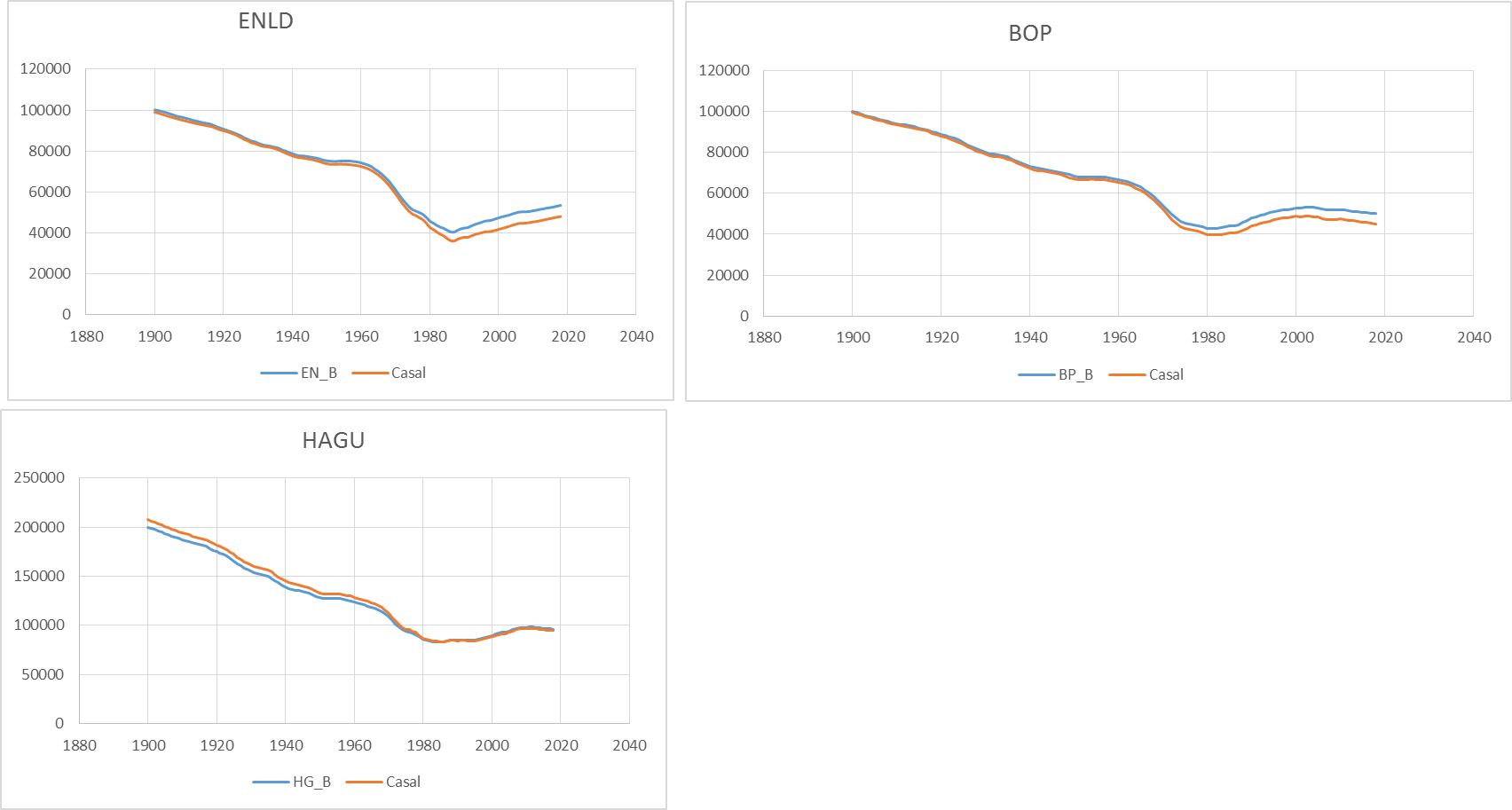
* B0 for each stock (3 parameters)
* Age-based selectivities for each method (double normal; 12 parameters)
* Catchability (qs) for the abundance indices (3 parameters)

### Model fits

The Casal model B0 estimates were very close to the “true” IBM values and casal likewise achieved very good predictive match to the IBM historical spawning stock biomass (SSB) trajectory (Table 2; Figure 4).

**Table 3: CASAL estimates of B0**

|  |  |  |
| --- | --- | --- |
|  | IBM B0 | CASAL estimate |
| East Northland | 100000 | 98984 |
| Hauraki Gulf | 200000 | 207605 |
| Bay of Plenty | 100000 | 99442 |



**Figure 6: Comparison of IBM and Casal spawning stock biomass trajectories for each SNA 1 simulated area**

Some possible reasons why the two model did not match more closely are:

* Casal selectivity is aged-based where-as the true process is length-based this may have resulted in less-than-perfect casal model predictions of catch at-age
* The order casal applies mortality in the annual time-step differs from the IBM sequence.
* There are significant computational differences in the way CASAL uses the vB curve to derive mean length at age (ergo biomass).

We intend to look further into the reasons for these slight differences however, the over-all result from the first preliminary test is very encouraging and we believe the IBM is basically performing well as measured against a modelling tool of known integrity, i.e. CASAL.

# References

Francis, R.I.C.C. (1988) Are growth parameters estimated from tagging and age-length data comparable?

*Canadian Journal of Fisheries and Aquatic Sciences* 45, 936 – 942

# APPENDICES

**Appendix 1: Model implementation of the VB growth increment model**

Uses a VB (linear increment) approach

*K* and *Linf* are generated as lognormal random deviates each fish *y* is assigned its own growth path as specified by *Ky*  and *Linfy*

Derivation of *K* ~ (lognormal) assumes K comes from a lognormal distribution with mean = *K* and variance = *v* = (*K*\*cv)2.

In log space log(k) come from a normal distribution with mean = 'u' and standard deviation 's' calculated as follows:

Likewise *log(Linf)~(Normal)*

The expected growth increment for a fish of length L with growth path *Ky* and *Linfy* over unit time t is given by the linear increment formulation of the vB (Francis 1988)



Where:

= change in length (increment) over unit time t for a fish of given length

m = rate of change of *L* (

c = increment for a fish of length = 0 the intercept of =

The change in length (i.e. increment) for a fish of length over is given by

**Appendix 2: R-code for agent-based tagging estimator simulations**

# Routine to simulate agent-based tagging process

# J. McKenzie

#############

# FUNCTIONS #

#############

# RMSD

RMSD<- function(x,y){

CV<-round(sqrt( sum((x-y)^2)/length(y))/x,2)

bias<-round((mean(y)/x)-1,2)

return(cbind('true'=x,'est\_mean'=round(mean(y),2),'bias'=bias,'RMCV'=CV,'obs'= length(y)))

}

# Resequence

reseq<- function(x){

seq\_ref<- 1

for(i in 1:length(x[,1])){

x[i,3]<-seq\_ref

seq\_ref<-seq\_ref+x[i,2]

x[i,4]<-seq\_ref-1

}

x[,1]<-c(1:length(x[,1]))

return(x)

}

# point sequence

pointer<- function(x,y){

point<-NULL

for(i in x){

point<-c(point,grep(1,(y[,3] <= i) \* (y[,4] >= i)) )

}

point<-point[order(point)]

point<-aggregate(point,list(point),length)

colnames(point)<-c('point','freq')

return(point)

}

################

# Main Routine #

################

# Population size

pop = 3000

# maximum agent scaler value

mscalr = 50

# Number of tags

tags = 100

# Number of catch

catch = 527

boot<- 1000

#######################

RunDet<-list()

RunDet[c('pop','mscalr','tags','catch','boot','runtime','stats')]<-c(pop,mscalr,tags,catch,boot,0,0)

pop\_est<-NULL

startTime<-Sys.time()

for(bt in 1:boot){

# trap silly mscalar values

if(mscalr >= pop ) mscalr <- pop-1

if(mscalr <= 0 ) mscalr <- 1

#### Create agent matrix

agent\_vec<- rep(mscalr,as.integer(pop/mscalr))

if(pop %% mscalr > 0) agent\_vec<-c(agent\_vec,pop %% mscalr)

agent\_mat<-as.data.frame(cbind('agent'=1:length(agent\_vec),'number'=agent\_vec,

'S\_seq'=(c(0,cumsum(agent\_vec)[1:(length(agent\_vec)-1)]))+1,

'E\_seq'=((c(0,cumsum(agent\_vec)[1:(length(agent\_vec)-1)]))+1)

+(agent\_vec-1), 'tag'=0))

##########################################

# Tagging sequence

tagseq<-sample(c(1:pop), tags, replace = FALSE, prob = NULL)

point<-pointer(tagseq,agent\_mat)

for(i in 1:length(point[,1])){

for(j in 1:point[i,2]){

if(agent\_mat[point[i,1],2]==1){

agent\_mat[point[i,1],5]= 1

}else{

zz<-agent\_mat[point[i,1],]

zz[,3]<-zz[,4]

zz[,5]<-1

zz[,2]<-1

agent\_mat[point[i,1],4]<- agent\_mat[point[i,1],4]-1

agent\_mat[point[i,1],2]<- agent\_mat[point[i,1],2]-1

agent\_mat<-rbind(agent\_mat,zz)

}

}

}

agent\_mat<-agent\_mat[order(agent\_mat[,4]),]

agent\_mat[,1]<-seq(1,length(agent\_mat[,1]),1)

pop<- sum(agent\_mat[,2])

tagsum<- sum(agent\_mat[,5])

#############################

# Take catch routines

#############################

# catch tag as individual catch untagged as agent

# recover tags

catseq<-sample(c(1:pop), catch, replace = FALSE, prob = NULL)

point\_tag<-pointer(catseq,agent\_mat)

catch\_tag<-NULL

del\_vec<-NULL

for(i in 1:length(point\_tag[,1])){

if(agent\_mat[point\_tag[i,1],5]==1){

catch\_tag<-rbind(catch\_tag,agent\_mat[point\_tag[i,1],])

del\_vec<-c(del\_vec,-1\*point\_tag[i,1])

}

}

norectags<- sum(catch\_tag[,5])

agent\_mat<-agent\_mat[del\_vec,]

# split out uncaught tagged fish

Tags<- agent\_mat[grep(1,agent\_mat[,5]),]

untagged<-agent\_mat[grep(0,agent\_mat[,5]),]

untagged<-reseq(untagged)

############

# untagged catch removal

cat\_untag<- catch-norectags

while(cat\_untag>0){

catseq<-sample(sum(untagged[,2]), 1, replace = FALSE, prob = NULL)

point<-pointer(catseq,untagged)

cat<-untagged[point[1,1],2]

if(cat\_untag>=cat)

{

cat\_untag<-cat\_untag-cat

untagged<-untagged[-point[1,1],]

untagged<-reseq(untagged)

}else{

untagged[point[1,1],2]<-untagged[point[1,1],2]-cat\_untag

untagged[point[1,1],4]<-untagged[point[1,1],4]-cat\_untag

untagged<-reseq(untagged)

cat\_untag<-0

}

}

agent\_mat<-reseq(rbind(Tags,untagged))

#stats

tagrec<-sum(catch\_tag[,5])

pop\_est<- rbind(pop\_est,cbind('pop\_est'= ((tags+1)\*(catch+1)/(tagrec+1))-1, 'tagsrec'=tagrec))

# Petersen estimator with Chapman bias correction

pop\_est<-pop\_est[order(pop\_est[,2]),]

} #endboot

endTime<-Sys.time()

runTime<-endTime-startTime

############# #

# simulation stats output table

#############

RunDet[['runtime']]<-runTime

RunDet[['stats']]<-RMSD(pop,pop\_est[,1])

RunDet