Modifications to SPM

The items below are suggested modifications to the SPM code to allow further flexibility and uses. It is split between items which are now in the SPM manual and those that still need to be defined.

# Bug fixes

*All the following cases are based on tests on the 1x1 example that is in the directory* examples/1x1/

## Covariance matrix

Introduce an alternative method hessian inversion … the basic one is a bit basic, and there may be better ones ALISTAIR TO LOOK INTO.

Remove covariance option from DE\_Solver (or else implement it!)

## MCMCs

MCMC code needs implementing. But currently calling spm –m errors out and crashes. Might be related to the thread code?

With spm –m we get,

Convergence was successful

terminate called after throwing an instance of 'std::string'

This application has requested the Runtime to terminate it in an unusual way.

Please contact the application's support team for more information.

We might want to return a ‘not yet implemented’ message in the meantime.

TODO: Ironically, it was throwing a "not yet implemented" exception that wasn't being properly caught in it's thread. Scott corrected this. There is still an issue with the thread not terminating properly. Scott to investigate this

## Threads

Cannot recall if threads have been implemented as per spm –t n. Can we check, fix as required. Let me know and also what text to add to the manual, as we give the command line argument but provide no help to the user as to what this means or why use it (and I cannot recall what its purpose was!).

Threads are semi-implemented. But there is still a significant amount of work to do for these. Originally I was aiming to thread-pool the minimisers to allow concurrent theories to be tested at once. The way the numerical\_differences calculates the gradient can be threaded. The DESolver can be completely threaded because generations only rely on the current best candidate and it's not too problematic if this changes during a cycle. The main idea behind the threads was to devise a way to use the MCMC in a threaded way. At the moment threads are ignored.

TODO: Lets remove threads at this stage.

TODO: ALISTAIR update the manual to remove reference to threads, and remove the documentation on the -t command line argument.

## Estimating vectors and values within vectors

Currently we can estimate variables and elements of a vector, e.g., @process[mylabel].proportions(0). We need to enable estimation of the entire vector in one go, e.g., @process[mylabel].proportions.

We should also implement the ability to estimate a bunch of them in one go, i.e., @process[mylabel].proportions(1-5)

Plus we should use natural counting for the element vectors in estimate blocks, not count from 0 – THIS BIT IMPLEMENTED

## Implement phased estimation

Phased estimation (see manual) needs implementing (or else revise the manual!)

## Simulations

SPM runs only 1 iteration with the spm -s n command. Needs to run n of them. NOW UPDATED

FILENAMES TO DO

If a file name is specified in the simulate\_observations report, it needs to append the simulation number, i.e.

@report CAA-1998-1

type simulated\_observation

observation CAA-1998-1

file\_name simulated\_observations

overwrite False

the filename should be written as below, where each file contains the output from a single simulation iteration. Hence, the appended number ranges from 1 to n. We can also be a bit clever and pad the number with leading zeros so that they all sort nicely.

simulated\_observations.001

simulated\_observations.002

simulated\_observations.003

etc.

## Long double error

Similar to the previous BOOST error when returning long doubles: the FMM\_Minimiser is returning the value 1.13076e-317 instead of the true value. The error message is,

Numerical\_differences: Linear step size too small (1.13076e-317)

Unclear convergence, maybe local minimum

See line 311 of FMM.cpp

cerr << FMM\_SMALL\_LINEAR\_STEP\_SIZE << " (" << dLambda << ")" << endl;

# AddED to the SPM manual and needing implementation

## Validation of likelihoods

Just as layers have a validation, some likelihoods require a validation process:

* Lognormal likelihood can only take positive layer values
* Multinomial have one error value per year only
* Lognormal can have one error value per class per year

Note that layers were not allowed to have a zero value. That has been changed to be allowed, but would provide issues if used for lognormal likelihoods.

*This has been implemented: Needs some additional checks and a bit of code modification. See validation functions in the observation classes for TODOs.*

## Derived quantities

Needs to implement object children: the argument ‘Type’ is not yet implemented (the current type implemented is abundance. Need a biomass one as well). Derived layers (see later) will probably follow the same format as derived quantities. When these work we will implement derived layers. See BH\_recruitment for possible accessor functions

Derived quantities need to be implemented during initialisations. During initialisations, for each initialisation phase, we can either (i) calculate and keep a value for every year of the initialisations (keeping these in a vector) or (ii) only keep the last value calculated by the initialisation phase (and do not bother calculating it for every year in the meantime). The latter is intended to be a time saving feature for the program.

Hence, for the @derived quantity we need to tell it what to do. Suggest a subcommand, say, ‘initialisation\_action?” that defaults to keep only last one unless specified as keep\_all (these names need to be improved!)

Pts: (these probably need more explanation ‘cause they are just dumps from ADs head)

* The offset ‘counts back’, and can count back through the initialisations when required,
  + but can only count back via 1 ‘phase’, and always takes to first one if the offset is larger than the elements to count back
* We will need to introduce time steps into initialisation phases
* Derived quantities will need to define the time step during each initialisation phase when the derived quantity is calculated

## Proportion time step

Observations allow the use of a proportion time step argument. This allows the observation to be evaluated part-way through the time step by taking the weighted average of the expected value from the start of the time step (the end of the previous time step) and the end of the time step.

Currently the code implements the solution as if this value is 1 (a good default).

We need to keep a copy of the world at the start of the first time step in the first year, and then overwrite this with a copy of the world at the end of every subsequent time step (once we have calculated out observation expected values using the weighted sum calculation).

Observations will need to calculate the expected value from the partition at the start of the time step, and then at the end of the time step, then use these two values to calculate the weighted sum. Note is proportion=0 or 1, then we can just use one or other, and don’t need to calculate both.

There may be a better method for this, but I suggest we use this to start… we could use the existence of an observation to tell us when to save a copy of the partition, hence reducing the model overhead. Note we would never need to do this during an initialisation.

## Biomass

The relevant formulas are described in the SPM manual, sections 4.10 and 4.11. This requires a size at age function, and a size-weight function. Both these functions have been implemented.

The subcommand age\_size command needs to be added to @model

We will enforce a relationship for each category in the model, but we will need to check that this exists for any category where we derive a size or biomass. Biomass and biomass density layers need to be implemented using these functions (section 4.4). Biomass derived quantiles also need to be applied; these are a single value that is the sum of a biomass layer.

THIS HAS BEEN IMPLEMENTED: SCOTT TO CHECK CODE FOR EFFICIENCY (and other errors)

## Mixed observations

Change of notation to + to denote “adding” of categories… see manual for observation=proportions\_at\_age

## Updated report arguments to cover multiple years

For

Report\_type = partition - IMPLEMENTED

Report\_type = layer – NOT IMPLEMENTED YET

Report\_type = selectivity – NOT IMPLEMENTED YET

Report\_type = layer\_derived\_view - IMPLEMENTED

modify argument **year** to **years** so a single report can be used to return a range of years or all years, rather than each report just doing one year at a time

## Recruitment

Details of the Beverton Holt (BH\_recruitment) recruitment formula are in the SPM manual, section 4.7.1. The recruitment process needs to define also steepness and YCS for the BH type recruitment. They are not yet fully implemented.

The C++ classes for both exist, but they need checking and additional work to function correctly. In particular, these rely on derived quantities and derived layers to be fully implemented.

Two options are defined:

* BH\_recruitment: recruitment in each cell is on the basis of the total recruitment over a large number of cells (for example, the SSB is a derived quantity over some specific area) and applied to some cells according to a layer (e.g. a recruitment layer), or
* local\_BH\_recruitment: recruitment in each cell is on the basis of the spawning stock biomass of that individual cell only (for example, a biomass layer); remember to note in the manual that this recruitment layer can be further submitted to dispersion through a movement function.

Formulas for calculations are given in the manual, section 4.7.1 (eqns 4.8, 4.9)

For BH\_recruitment

@process Recruitment

type BH\_recruitment

r0 5000000

steepness 0.75

categories immature # Define the categories into which recruitment occurs

SSB SSB\_TOA # a derived quantity that gives the SSB for the recruitment calculation

B0 SSB\_TOA element 1 # a value of the derived quantity that gives B0 for the recruitment calculation

Layer MyRecruitmentlayer # Name of the layer used to determine where recruitment occurs

Age 1 # Define the age within each category that receive recruitment (see constant recruitment)

proportions 1.0

standardise\_years 1994-2004

YCS\_years 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009

YCS 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

SSB\_yearclass\_offset 1

R0, steepness can be estimated or fixed, and all the “free” YCS are estimated. If standardise\_years is defined, then the YCS used in the formula is YCS/mean(sum(YCS[in range standardise\_years])), otherwise they are used as defined.

For local\_BH\_recruitment

@process Recruitment

type local\_BH\_recruitment

r0 5000000 # multiples the cell value in r0\_layer by this to get actual recruitment R0 for each cell

steepness 0.75

categories immature # Define the categories into which recruitment occurs

SSB\_layer MySSBLayer # a numeric layer that gives the SSB in each cell in each year for the recruitment calculation

B0\_layer MySSBLayer element 1 # a value of the derived layer that gives the SSB for the recruitment calculation

r0\_layer MyRecruitmentlayer # Name of the layer used to determine the relative cell by cell r0

Age 1

proportions 1.0

standardise\_years 1994-2004

YCS\_years 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009

YCS 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

SSB\_yearclass\_offset 1

R0, steepness can be estimated or fixed, and all the “free” YCS are estimated. If standardise\_years is defined, then the YCS used in the formula is YCS/mean(sum(YCS[in range standardise\_years])), otherwise they are used as defined. Note if r0\_layer has a zero or negative value, then recruitment in that cell is zero.

# Additional changes (once we’ve sorted the above)

## meta layers (this needs a bit more thought .. and we’ll implement once we’ve implemented derived layers and layer math)

These are defined in the SPM manual (section 4.4 item 8). Meta layers are layers indexed by year and applied by year. The meta-layer class is already defined (see section 8.6.8), which is in effect a three dimensional layer. Need to add the type of layer (numeric or categorical).

From the description it seems like the meta-layer is defined as a list of the names of the individual layers, see example below. It might be useful to have another option to define the meta-layer as defining layer, list the data as a layer, with data lines for each year.

Also required is how to apply these to calculations. Everywhere a layer is used, if it is a meta-layer then the year-specific layer should be retrieved prior to being applied.

Interpolation of layers is not allowed: all years defined must be contiguous otherwise throw an error. Extrapolation method is requested in the arguments, whereby the user defines the initialisation and prediction layers.

**Note**: Fishing layers (for example) can be defined as meta-layers or as individual layers. We need to make sure the code can cope with either, whereby the layer is applied, or if a meta layer, the appropriate year is applied. This will impact the @process function.

Below an example as it is currently described

@layer Fishing

initialisation No\_Fishing

prediction Future\_Fishing

years 1998 1999

layers Fishing\_1998 Fishing\_1999

Could also be defined as follows

@layer Fishing

initialisation No\_Fishing

prediction Future\_Fishing

Years 1998 2005

data 0 0 234 0 111 0

data 0 0 0 500 10 0

… #as many rows as rows in the model

## Derived layers and derived meta layers

There is a need for more generic derived layers, with specific mathematics formula to obtain these layers, and parameters which can be estimated if needs be. Current derived layers include abundance layers or biomass layers (not yet implemented – but will be).

Two types of derived layers need to be defined: as derived layers, or as derived meta-layers (see section 4.4 of the manual where this will be defined).

* Derived layers are calculated at a specific user-defined time step and years. If no years are defined, the data overwrites the previous calculation. For example B0 is derived once only, at the last initialisation phase (defined under year); but a mortality dependent layer is calculated every year (including in the initialisation phases) and only the latest calculation is kept in memory.
* Derived meta-layers are calculated every year of the model, and the results for each year saved in a single meta-layer. Derived meta layers will be calculated for the initialisation phases as well, and the information stored in the meta-layer.

Two new layer types need to be created, a used-defined calculated layer termed a derived\_layer or derived\_meta\_layer. It will include the layers to use, calculation, etc. Parameters for the formula can come from other layers, other categories, and / or specifically defined parameters. These parameters can be then estimated in the estimation section as any other parameter. Layer calculations can also be nested in each other.

Note that we currently define that derived layers are calculated or determined at the time a calculation is required. We propose to determine the time step when the layer calculation occurs, and then preserve this value until updated.

For example B0 is defined from the SSB biomass layer saved as a derived layer. SSB is firstly defined.

@layer SSB

type biomass

timestep one

categories mature mature\_tag

selectivities 1 1

age\_size\_weight TOA\_asw

The derived layer B0 is simply SSB at initialisation phase, here it is multiplied by a layer which defines your stock limit for example (called SSB\_area). If the timestep is hard-coded in both, then there needs to be a check that they do agree.

@layer B0\_TOA

type derived\_layer

years initialisation-phase3

timestep one

layers SSB BBS\_area

formula SSB \* SSB\_area

Density dependent mortality base on diet electivity can also be defined using a derived layer. In the example below the predation of toothfish on grenadier as opposed to eelcod depends on the preference for the two species tempered by their availability in each cell. The layer defined below can then be used as a layer for mortality of grenadier due to predation by toothfish.

@layer MortalityDep

type derived\_layer

timestep one

layers B\_WGR B\_ELC

parameters ElecTW ElecTE # user defined labels that become estimable parameters

values 0.75 0.25

formula ElectTW\*B\_WGR/(B\_WGR+B\_ELC) # # code resolves string, and hence applies calculation.

The actual electivities of toothfish for grenadier can be estimated as follows.

@estimate layer[MortalityDep].ElecTW

lower\_bound 0.01

upper\_bound 0.99

prior Uniform

Layers, parameters and values are optional, but need to be defined if used in the calculation. If parameters is used, values is also needed, and of the same size as parameters. Layers can be any layer known to the model. Years are optional in which case the layer gets calculated every year including initialisation.

The following math functions would need to be implemented, and include parentheses

+ - \* / exp log(e) sqrt ^2 pow(a,b) cos sin Z

All would be functions that operate on every cell in a layer independently. Z is the zeroFun implemented elsewhere in our code.

See separate document for some math parsers.

## MCMC

Implement MCMCs

## Projections

Remove projections from the model (or at least from the users point of view … we should leave the code as we may want these in the future)

# To be further developed and defined in the SPM manual

*This section outlines thoughts about new and changed functionality. Its just a holding place for these ideas and should not be implemented.*

## Implement CVs on size-weight relationship

Implement CVs on size-weight relationship.

## Problems with meta-layers

We need to think carefully when we should allow meta-layers and when we shouldn’t. For example meta-layers under biomass events might be misleading if only applied for specific years when it should be all years.

Maybe have meta-layers used instead of layers as default, but with specific safeguards or cases where you can’t.

## Entering layer data

At the moment we require a “data” subcommand for every line of SPM to input layer data. Is it worth adopting the form used in randomstation and cala to have a ‘begin\_data’ and ‘end’data’ style command? Then the subcommand is not required on each new line anymore…?. Note one difference between the programs is that we are inputting a matrix of values by row… and not a table of data with column headings.

## Tags as indicators of movement only

The aim of this new implementation is to use tags as informative of movement only, based on individual tag recaptures only. A new observation data class is created which relates to individual event matching. Data is input with a line per recapture event, with the relevant information to the model (in the order specified).

@observation Tag\_match

type event\_match

categories immature\_tag mature\_tag spawning\_tag

# year\_release year\_recapture cell\_release cell\_recapture age\_recapture selectivity\_recapture selectivity\_recapture selectivity\_recapture

data 2004 2009 r3-c4 r3-c4 12 FishingSel FishingSel FishingSel

…

Each tag recapture will be associated with temporary layers, created in the year of release. The layers will be calculated as such

* At year\_release,
  + Estimate the proportion of that fish in each category
  + Create one layer per category with in cell\_release the value of the proportion of that fish in that category
  + Age of that fish is age\_recapture + year\_recapture - year\_release.
* Between year\_release +1 and year\_recapture:
  + For each category, apply the relevant processes in the model (e.g. maturation, movement, mortality)
  + Update the layers to the new values of distribution, moving fish between layers (categories) and cells
* At year of recapture, once the layer has been updated
  + Apply fishing selectivity to each layer
  + For each layer, calculate the probability of being in the recapture\_cell as the value in that cell divided by the sum of the value in all the cells. Add each value to the total likelihood
  + Delete the layers