Explanation of Mathematical Methods Behind the Second Half of the Project

AUTHOR
Emma Bowen

Introduction

What mathematical methods will I explain in this document that have not been explained in Explanation of Mathematical Methods from First Half of the Project?

- The basics of the MixME model
    - Creating the operating model
    - Building the MixME input object
    - Running the simulation
    - Analysing the results of the simulation
    - HCR parameters
    - Additions or differences in the shortcut model
- Calculating the risk
- Parallelisation

HIGHLIGHT MORE THAT FTARGETS ARE STAYIING THE SAME THROUGHOUT THE SIMUALTION - found in MixME wiki Fixed fishing mortality management strategy

Context

The aim of this half of the project is to apply the methods from the first half of the project to mixed fisheries using the MixME R package. I have used the methods from the first half of the project to write code that follows a very similar process, but where the aim is now to find the Ftarget for each stock that is precautionary but maximises total catch over the years 2030 to 2039. I set this goal because MixME is designed to run projections into the future, and looking at catch in the long term will minimise the chance that a simulation where a stock fails is chosen (Pace et al. 2025a),(Pace 2024). The 20 year projection from 2020-2039 was kept consistent with examples on the MixME documentation and follows guidelines from ICES to create long-term projections based on the biology of the stocks (Pace 2024),(ICES 2019a). It also follows ICES guidelines to only calculate catch and risk for the last ten years for long-term projection, to allow time for a recovery period (ICES 2019a).

I focused on the datasets from the Fixed fishing mortality management strategy example (`mixedfishery_MixME_om`) and the Exploring simulation outputs example (`mixedfishery_MixME_input`) which are both in the MixME documentation (Pace 2024).

However, for the second dataset `mixedfishery_MixME_input` I was given a shortcut method by a researcher at Cefas which takes a more direct approach to the simulation. The code for these datasets is `Optimising_ftarget_in_MixME_mult_points_parallel.R` and `Two_stocks_Optimising_ftarget_in_shortcut_model.R` respectively.

Both of these datasets have two stocks (cod and haddock) and two fleets (Pace 2024). I was told by the same researcher at Cefas that the stocks are North Sea cod and Celtic Sea haddock, but using citations I can only back up that they are Atlantic cod and haddock (Pace 2024),(ICES, n.d.),(ICES, n.d.). This allows me to use the same methods as in the first half of the project by replacing F_{target} with F_{cod} and $B_{trigger}$ with F_{had}, where F_{cod} and F_{had} are the fishing mortalities for cod and haddock respectively.

For this half of the project, we have switched to modelling SSB directly instead of calculating the risk. This is due to the simulation being deterministic as both datasets only have one iteration and the noise for this is pre-calculated (Pace et al. 2025b). These conditions simplify the simulation but unfortunately mean that we cannot use the standard ICES definition of risk to calculate $Risk = P(SSB < B_{lim})$ (Pace et al. 2025b),(ICES 2019a). Instead, we extract $\min(SSB)$ for each sampled point from the years 2030-2039 in the simulation and then model these as a GP (Spence 2025),(ICES 2019a). This allows us to predict the distribution of $\min(SSB)$ values at each point in the design space (Spence 2025). For each point, we then see how many of these predicted values fall below B_{lim} to calculate $P(\min(SSB) < B_{lim})$ at that point (Spence 2025). We do this for each stock. This method of calculation satisfies the ICES precautionary standard (ICES 2019a).

Due to the change in stocks being modelled, I have decided to keep the GP prior for catch modelled by the `~.^2` function in R for every round of the optimisation process. Despite the more complicated prior used in later rounds in `case_study8.R` being designed to approximate yield curves, it may not be appropriate in a mixed fisheries context where the catch of one species is affected by the catch of another (Spence 2025),(Pace et al. 2025a), (Ulrich et al. 2012). Leaving the GP more general avoids mis-specification, ensuring that the GP can be appropriately fitted to the points (Williams and Rasmussen 2006).

The process for the first half of the project adapted to our new situation is outlined below. We take a Bayesian History Matching (BHM) approach (Spence 2025). Firstly, to get some initial data, we randomly sample our first set of $n - 1$ points, where n is the number of cores the system we are on has (Spence 2025). Then, for each round we do the following:

- We set up or update the Gaussian Processes (GPs) to model the $\min(SSB)$ from 2030-2039 for each stock and the GP to model the total catch from 2030-2039
- We can then use the B_{lim} for each stock as a threshold so that we only consider the values of F_{cod} and F_{had} that have $P(min(SSB) > B_{lim}) > 0.05$ for all of the last ten years of the simulation (2030-2039)
- We use the GP which is modelling the total catch to predict the value for the total catch at every point in the sample space, which will have some uncertainty
- We use BHM to remove any points that are implausible (that have a probability of less than 0.01% of being higher than the current best total catch)

0.01% of being higher than the current best total catch)

- We use the Knowledge Gradient (KG) acquisition function to select $n-1$ plausible points to sample in the next round, as per the discussion in Deciding which acquisition function is best

We repeat this process until there is only one plausible point left and then we will accept this as being the F_{cod} and F_{had} that maximise the catch whilst keeping the SSB above B_{lim}.

Calculating the risk

MixME defines risk as the proportion of iterations in each year where SSB falls below B_{lim} (Pace 2024). However, due to only having one iteration in each of my datasets, I have experimented with using a different method to measure the risk (Pace 2024).

In contrast to the first half of the project, we now calculate the risk using the SSB. We have described it above briefly but will go into more detail here.

We should quickly note before our calculations that our new sample space is as below:

```
# Define Sample Space as discrete with 0.02 increments
dat <- data.frame(expand.grid(
  Fcod = seq(0.0, 0.6, by=0.02),
  Fhad = seq(0.0, 0.6, by=0.02)
))
```

This range ensures that we are able to model stock collapse for cod by modelling values above F_{lim} and that we can model stock recovery for cod by including very low values (ICES 2019c). It also allows us to model unsafe fishing for haddock by modelling values above F_{MSY} (ICES 2019b). Recalling that cod is the choke stock for both of our datasets, this sample space is appropriate (Pace 2024). - MAYBE FIND MORE JUSTIFICATION, I.E. SOMEHTING SAYING HOW TO SET UP THESE SMAPLE SPACES?

We also have a B_{lim} for each stock, taken from ICES advice (ICES 2020b),(ICES 2020a):

```
Blim_cod <- 107000
Blim_had <- 9227
```

which are both measured in tonnes.

Firstly, we extract the $\min(SSB)$ for each stock from the result of the simulation we have run using the tracking object (Pace 2024),(Pace et al. 2025a):

```
## // Extract the min ssb at the last ten years of the simulation for both stocks //
    # Picking up long term SSB values to see if they dip below Blim at any point
    ssb_cod_data <- c(res$tracking$cod$stk["SB.om", ac(2030:2039)])
    ssb_had_data <- c(res$tracking$had$stk["SB.om", ac(2030:2039)])

    # Getting minimum ssb during this time for each stock to model with GPs
    ssb_cod_min <- min(ssb_cod_data, na.rm = TRUE)
    ssb_had_min <- min(ssb_had_data, na.rm = TRUE)
```

```
ssb_had_min <- min(ssb_had_data, na.rm = TRUE)
```

We then put these results into the GP we will use to model $\min(SSB)$ for each stock:

```
gp_cod_ssb <- km(~.^2,design=runs[,c("Fcod","Fhad")],estim.method="MLE",
                 response = ssb_cod_min,nugget=1e-12*var(ssb_cod_min)+1e-15,
                 covtype = "exp")
gp_had_ssb <- km(~.^2,design=runs[,c("Fcod","Fhad")],estim.method="MLE",
                 response = ssb_had_min,nugget=1e-12*var(ssb_had_min)+1e-15,
                 covtype = "exp")
```

These GPs have remained modelled by the `.^2` function, the same as in the GP for risk in the first half of the project, as this is quite general and avoids mis-specification (Williams and Rasmussen 2006),(Roustant, Ginsbourger, and Deville 2012). They have also kept the same estimation method, nugget and kernel (Williams and Rasmussen 2006). - JUSTIFY MORE

Next, we predict the $\min(SSB)$ for every point in the sample space:

```
pred_ssb_cod <- predict(gp_cod_ssb, newdata = dat, type = "SK")
pred_ssb_had <- predict(gp_had_ssb, newdata = dat, type = "SK")
```

We then calculate $Risk = P(min(SSB) < B_{lim})$ for each stock:

```
# Get the probability that sbb =< Blim for cod and haddock
pssb_cod <- pnorm(Blim_cod, pred_ssb_cod$mean, pred_ssb_cod$sd + 1e-12)
pssb_had <- pnorm(Blim_had, pred_ssb_had$mean, pred_ssb_had$sd + 1e-12)
```

Then, we set the KG of any unsafe points to be zero so that they will not be chosen as points to be sampled in the future:

```
# Loop over candidate points
  for (i in seq_len(m)) {
    # set to 0 for any unsafe points - any points with
    # probability that ssb < Blim > 0.05 in the years 2030-2039
    # i.e. the run is not precautionary in these years
    if (pssb_cod[i] > 0.05 || pssb_had[i] > 0.05) {
      kg[i] <- 0
      next
    }
```

We can say `ssb < Blim` here instead of `ssb =< Blim` because we are sampling from a continuous normal distribution and so they are equivalent.

By ensuring that $P(\min(SSB) < B_{lim}) < 0.05$ we ensure that $P(SSB < B_{lim}) < 0.05$ for every year in the long term forecast (2030-2039). This guarantees that the maximum annual risk in this period remains below the 5% threshold, which is exactly what is required to meet the ICES precautionary standard due to their definition of $Prob3$ in (ICES 2019a). This means that this risk calculation could be used in policy documents to set official catch limits in countries that have agreed to this standard (ICES 2019a).

The basics of the MixME model

Going to write down everything and then remove areas Gustav said I don't need to worry about.

- Explaining the tracking object as this how I get catch ?? - MAYBE IN A SECTION SUMMARISING FULL CODE PROPCESS?

Starting F_{cod} and F_{had}

We started with $F_{cod} = 0.28$ and $F_{had} = 0.353$ based upon the values given in the Fixed fishing mortality management strategy example from the MixME wiki (Pace 2024). These were drawn from the North Sea Cod and Celtic Sea Haddock advice published in 2020 respectively (ICES 2021),(ICES 2020b).

Creating the Operating model

We first create the operating model `mixedfishery_MixME_om` which contains the true data for the stocks and fleets in the dataset (Pace et al. 2025a). We have data for North Sea Cod from 1963-2019 and for Celtic Sea Haddock from 1993-2019 (Pace 2024). For the stocks, this includes the numbers, natural mortality, stock mean individual weight and proportion mature split by age and for fleets this includes landing numbers, landing mean individual weight, discard numbers, discard mean individual weight, selectivity and catchability (Pace 2024)**. - DISCUSS CATCHABILITY?**

In both of our files, this is loaded with the dataset (Pace 2024). This true data is generated using standard age-structured equations to model the dynamics of the fleets and the stocks (Pace 2024). For every year, it has the catch (in terms of landings and discards) from each fleet and the survivors from that year (Pace 2024). It also contains a stock-recruitment model and recruitment is done at the beginning of each time step (Pace 2024).

The steps for fully assembling the operating model for input into the MixME model are:

1. Estimate historic quota-share for the two fleets
2. Project stocks n years into the future
3. Calculate numbers of both stocks in initial year
4. Generate an observation error model

Estimate historic quota-share for the two fleets

Firstly, we use `mixedfishery_MixME_om` to determine the quota share of the catch for each fleet. This is done by assuming for each stock that the quota share corresponds to the proportional share of landings and is carried out by the `calculateQuotashare` function (Pace 2024).

```
out <- calculateQuotashare(stks = mixedfishery_MixME_om$stks,
       flts = mixedfishery_MixME_om$flts, verbose = TRUE)
```

Project stock n years into the future

To carry out our 20 year projection, we need to extend the stock and fishery structures forward from 2019 into 2039. There are three categories of parameters that are not estimated dynamically and so need to be extended (Pace 2024). These are:

1. All stock parameters, landings and discards mean individual weights and landed fraction
2. Catchability and catch selection
3. Quota-share

We project the parameters in each of these categories from 2019-2039 using the average from the last three years using the `stfMixME` function (Pace 2024).

```
out <- stfMixME(mixedfishery_MixME_om, method = "yearMeans", nyears = 20,
       wts.nyears = 3, sel.nyears = 3, qs.nyears = 3, verbose = TRUE)
```

- JUST SET THIS UP OR ACTUALLY PROJECT? DO HAVE HISTORIC DATA TO DO THIS FROM

Calculate numbers of both stocks in initial year

To be able to do our projections of catch, we need to know the number of each stock in each age class at the beginning of the first projection year, 2020. This requires us to do a 1-year short term forecast from our starting point of 2019 using the FLasher package (Pace 2024).

```
# initial projection year
iy = 2020

# set an arbitrary effort-based target for each fleet as we only want the
# values at the beginning of 2020
ctrlArgs <- lapply(1:length(mixedfishery_MixME_om$flts), function(x) {
            list(year = iy,quant = "effort",
            fishery = names(mixedfishery_MixME_om$flts)[x],value = 1)
})

# This matrix maps which fleets catch which stocks
ctrlArgs$FCB <- makeFCB(biols = mixedfishery_MixME_om$stks,
                        flts = mixedfishery_MixME_om$flts)

# Generate effort-based FLasher::fwd forecast control
flasher_ctrl <- do.call(FLasher::fwdControl, ctrlArgs)

# Simulate one year of fishing to get starting conditions right
omfwd <- FLasher::fwd(object = mixedfishery_MixME_om$stks,
                      fishery = mixedfishery_MixME_om$flts,
                      control = flasher_ctrl)

#Update the operating model with projected population numbers
mixedfishery_MixME_om$stks$had@n[,ac(iy)] <- omfwd$biols$had@n[,ac(iy)]
mixedfishery_MixME_om$stks$cod@n[,ac(iy)] <- omfwd$biols$cod@n[,ac(iy)]
```

We set an arbitrary forecast target here because the 2020 numbers are calculated from existing numbers (rather than a defined target) (Pace 2024). - ? FIND OUT MORE

Creating the Observation Error Model

This is another important component of the MixME model (Pace 2024). We create the observation error model `stk_oem` where we apply pre-sampled noise to the catch from each fleet (which we obtained earlier from the `mixedfishery_MixME_om` object) (Pace 2024). We generate future stock and management advice from this object (Pace 2024). - MAY NEED TO REFERENCE MORE COMPLICATED TUTORIAL FOR FULL EXPLANATION OF THIS

```
# Create a list of FLStock objects from the FLBiol objects in
# the operating model
stk_oem <- FLStocks(lapply(mixedfishery_MixME_om$stks, function(x) {

  # for each fleet, find which catch element corresponds
  # to the current stock
  catch <- sapply(mixedfishery_MixME_om$flts, function(y)
           which(names(y) %in% name(x)))

  # get catches form each fleet and convert to FLStock
  xx <- as.FLStock(x, mixedfishery_MixME_om$flts,
                   full = FALSE, catch = catch)

  # specifies fishing mortality is measured as instantaneous
  # fishing mortality rate
  units(xx@harvest) <- "f"

  # remove excess data as we don't observe stock biomass or stock
  # numbers-at-age directly in reality forces us to simulate
  # these using only the available data
  stock.n(xx)[] <- NA
  stock(xx)[]   <- NA

  # return the converted object
  return(xx)
}))
```

Build the MixME input object

We then use this in the `makeMixME` function to make the MixME input object, which can then be used to run the simulation. We have five arguments here. Two of these are where we input our Operating Model and our Observation Error model. The next two specify how many time steps (in our scenario, years) it takes for the management advice to be enacted and what type of management we are using respectively (Pace 2024). The last argument - NOT DESCRIBED YET

```
  input <- makeMixME(om = mixedfishery_MixME_om, catch_obs = stk_oem,
                     management_lag = 0, management_type = "fixedF",
                     parallel = FALSE)
```

Running the simulation

We update some of the management and simulation settings and then run the simulation (Pace 2024).

```
# Cod and haddock catches occur at the start of the year so timing=0
input$oem@args$catch_timing$cod <- 0
input$oem@args$catch_timing$had <- 0

# Define the age range for calculating average fishing mortality
# (Fbar) for both stocks
input$oem@observations$stk$cod@range[c("minfbar","maxfbar")] <- c(2,4)
input$oem@observations$stk$had@range[c("minfbar","maxfbar")] <- c(3,5)


# Set the target fishing mortality for both stocks as
# the next point decided by the algorithm
input$ctrl_obj$hcr@args$ftrg$cod <- f_cod
input$ctrl_obj$hcr@args$ftrg$had <- f_had

## Update fbar ranges
input$args$frange$cod <- c("minfbar" = 2, "maxfbar" = 4)
input$args$frange$had <- c("minfbar" = 3, "maxfbar" = 5)


#RUN MIXME SIMULATION
res <- runMixME(om  = input$om, oem = input$oem,
                ctrl_obj = input$ctrl_obj, args = input$args)
```

It can be seen above that we have updated the arguments for when catches occur, the age range for calculating average fishing mortality in all the places it is needed and the target fishing mortality we are using for this round of the simulation (Pace 2024).

It is important to note that the F_{cod} and F_{had} we choose remain constant throughout the simulation (Pace 2024). - SURELY IN SUPP OR PAPAER TOO?

Analysing results of the simulation

The `tracking` object records summary statistics for the modelled stock and fleet dynamics, as well as metrics describing the observed state of the system by the management procedure. It also contains simulation performance and diagnostics statistics, which is what we will focus on here (Pace 2024).

We firstly check for management advice failure (Pace 2024):

```
## Check for advice failure
apply(res$tracking$iterfail, 1, mean)
```

Then we can also check for effort optimisation failure and the message given if there was any failure (Pace 2024):

```
## Check for effort optimisation failure
```

```
res$tracking$optim

## Check for effort optimisation message
res$tracking$message
```

Effort optimisation failure can be sued to see if we are over-fishing the stocks to a point that they will go extinct (Pace 2024).

Another result we can see is the over-quota catches (**mixMEwiki?**):

```
## Check maximum overshoot of the quota by fleets
max(res$tracking$overquota, na.rm = TRUE)
```

We can also check the quota uptake using the below (Pace 2024):

```
## Check quota uptake for cod and haddock
res$tracking$uptake["cod",,,1]
res$tracking$uptake["had",,,1]
```

Lastly, we can check which stock is the choke stock using (Pace 2024):

```
## Check choke stock
res$tracking$choke
```

HCR parameters

We add B_{lim} into the HCR to help us calculate the risk later on (Pace 2024). It is helpful for calculating it in our current way, or the way MixME defines it as: "the proportion of iterations (read replicates) in each year where stock spawning biomass falls below the biological limit reference point (B_{lim})" (Pace 2024).

```
## Define Blim for each stock - required for way
#calculating risk right now
hcrpars <- list(cod = c(Blim = 107000), had = c(Blim = 9227))

# Update the control object with the new HCR parameters
res$ctrl_obj$phcr <- mseCtrl(args = list(hcrpars = hcrpars))
```

ARE THERE OTHER hcrpars ALREADY DEFINED?

Additions or Differences in the Shortcut Model

NEED CITATIONS FROM SOMEWHERE BUT MAY BE VERY DIFFICULT AS NOT PUBLIC YET

This is the `Two_stocks_Optimising_ftarget_in_shortcut_model.R` file. We use the same data here as in the `Optimising_ftarget_in_MixME_mult_points_parallel.R` file, except that we have now also loaded in the Operating Model and Observation Error Model with the data (**MixMewiki?**).

Zero-catch advice

CHECK DEFINETLY NOT IN OLD METHOD

In this simulation, we have a condition meaning we return zero-catch advice if the SSB is below B_{lim} in the year after the advice year. We are checking the year after the advice year due to our management lag of one year. Firstly, we identify any simulation where $SSB < B_{lim}$ and re-run these simulations specifically targeting B_{lim}:

```
## Find iterations where SSB at end of TAC year is < Blim
  belowBlim <- which(TACyr_ssb < attr(ctrl, "Blim"))

## Go through model fits - where SSB < Blim, forecast to target Blim
  if (length(belowBlim) > 0) {

    ## define forward control targetting Blim
    targ <- matrix(0,nrow=fwd_yrs+1, ncol = niter)
    targ[1,] <- fsq
    # target is now Blim
    targ[2,] <- c(attr(ctrl,"Blim"))
    targ[3,] <- ctrl@iters[,"value",]

    ctrl_blim <- fwdControl(list(
      year  = c(ay,ay+mlag,ay+mlag+1),
      #focus on biomass in the 2nd year to force us to end at
      # Blim instead if we were below
      quant = c("fbar","ssb_end","fbar"),
      value = c(targ)))

    #rerun simulation focusing on Blim to find the Ftarget
    # that is appropriate
    stk_blim <- FLasher::fwd(stk0, sr = sr0, control = ctrl_blim)#
```

However, if this still fails, we identify these runs in the `zeroTAC` variable and manually set the total annual catch (TAC) to zero.

```
## Find iterations with zero TAC advice
# (some stocks may be in such bad health that we can't catch any)
zeroTAC <- ssb(stk_blim)[,ac(ay+mlag+1)] < attr(ctrl,"Blim")

# set any in zeroTAC to zero manually
TAC[zeroTAC]   <- 0

}
```

Combined, this looks like:

```
## Find iterations where SSB at end of TAC year is < Blim
belowBlim <- which(TACyr_ssb < attr(ctrl, "Blim"))

## Go through model fits - where SSB < Blim, forecast to target Blim
if (length(belowBlim) > 0) {
```

```
  ## define forward control targetting Blim
  targ <- matrix(0,nrow=fwd_yrs+1, ncol = niter)
  targ[1,] <- fsq
  # target is now Blim
  targ[2,] <- c(attr(ctrl,"Blim"))
  targ[3,] <- ctrl@iters[,"value",]

  ctrl_blim <- fwdControl(list(
    year  = c(ay,ay+mlag,ay+mlag+1),
    #focus on biomass in the 2nd year to force us to end at
    # Blim instead if we were below
    quant = c("fbar","ssb_end","fbar"),
    value = c(targ)))

  #rerun simulation focusing on Blim to find the Ftarget
  # that is appropriate
  stk_blim <- FLasher::fwd(stk0, sr = sr0, control = ctrl_blim)

  ## Find iterations with zero TAC advice
  # (some stocks may be in such bad health that we can't catch any)
  zeroTAC <- ssb(stk_blim)[,ac(ay+mlag+1)] < attr(ctrl,"Blim")

  ## update TAC
  TAC[belowBlim] <- round(c(catch(stk_blim)[,ac(ay+mlag)])[belowBlim],3)
  # set any in zeroTAC to zero manually
  TAC[zeroTAC]   <- 0
  }
```

`ICES_HCR` function

This is a new way to create the HCR.

```
## === Define a custom harvest control rule ===
ICES_HCR <- function (stk, args, hcrpars, tracking) {

#' @param stk contains information about fish stocks, e.g. age
#' @param args contains useful arguments such as ay and mlag
#' @param hcrpars contains the parameters of the HCR, e.g. Ftrgt, Btrigger, Blim
#' @param tracking contains the current tracking object and
# is currently returned unchanged

  ## Extract year arguments

  # ay is the current assessment year
  ay <- args$ay
  # mlag is the management lag in years
  mlag <- args$management_lag
  # this is the number of iterations
  ni <- dims(stk)[["iter"]]


  # SHORT-TERM FORECAST
  ## Carry out short-term forecast to get ssb at the beginning of the advice year
```

```
## historical geomean to estimate recruitment for the stock
# (prevents big spikes in recruitment having a large effect)
sr0 <- as.FLSR(stk, model = "geomean")
# propagate sr0 params to match number of simulations
sr0@params <- FLCore::propagate(sr0@params, iter = ni)

# hardcoded using different starting years for different stocks
# when estimating recruitment
if(stk@name == "cod") stk_n <- window(stk@stock.n, start = 2015)
if(stk@name == "had") stk_n <- window(stk@stock.n, start = 1993)

# We get the row of data for the youngest fish and calculate
# the geometric mean for that row
# We do this by taking a log before calculating the average,
# as this reduces the effect of large values,
# and then exponentiating
sr0@params[] <- exp(yearMeans(log(stk_n[1,])))


## find the first year with any data
minyr <- dims(stk@stock[!is.na(stk@stock.n)])$minyear
## remove any years before this
stk0  <- window(stk, start = minyr)

## extend stock object by one year
## automatically fills in parameters by averaging over the last 3 years
stk0 <- FLasher::stf(stk0, 1)

# FLasher::fwd will throw an error if there are NAs in weights in future years.
# I need to zero these if associated numbers are zero
FLCore::discards.wt(stk0)[FLCore::discards.n(stk0) == 0] <- 0
FLCore::landings.wt(stk0)[FLCore::landings.n(stk0) == 0] <- 0


## find status quo F slightly differently for each stock
# cod and whiting average last three years, haddock average last year only
if(stk@name == "cod") fwd_yrs_fsq <- -2:0
if(stk@name == "had") fwd_yrs_fsq <- 0

# estimated fishing mortality for the forward year for each stock
fsq <- yearMeans(fbar(stk)[,as.character(fwd_yrs_fsq+ay-mlag)])

## FLasher cannot handle fsq=0
fsq <- ifelse(fsq==0, 0.001,fsq)

## Construct forward fishing mortality with same number of rows as simulations
# Will overwrite the second row later
targ <- matrix(0,nrow=2, ncol = ni)
targ[1,] <- fsq
targ[2,] <- fsq

# Wrappper that constructs the fwdControl object, setting fbar
```

```
  # to values in the targ matrix
  ctrl0 <- fwdControl(list(
    year  = c(ay,ay+mlag),
    quant = "fbar",
    value = c(targ)))

  ## project stock forward to get ssb in the advice year
  stk_fwd <- FLasher::fwd(stk0, sr = sr0, control = ctrl0)


  # HCR

  ## Extract the parameters and propogate to each simulation
  Ftrgt <- propagate(FLPar(hcrpars["Ftrgt"]), ni)
  Btrigger <- propagate(FLPar(hcrpars["Btrigger"]), ni)
  # The Blims we put into hcrpars are propagated here
  Blim <- propagate(FLPar(hcrpars["Blim"]), ni)

  ## calculate F multiplier

  # calculate what ratio of Btrigger we are at for each simulation
  # - 1 means we're at Btrigger
  status_Btrigger <- tail(ssb(stk_fwd), 1)/Btrigger
  # find which simulations are above Btrigger
  pos_Btrigger <- which(status_Btrigger > 1)
  # set Ftarget to the ratio of Ftarget that corresponds to status_Btrigger
  Fmult <- status_Btrigger
  # cap this ratio at 1
  Fmult[, , , , , pos_Btrigger] <- 1

  ## calculate new F target
  Ftrgt <- Ftrgt * Fmult
  # make most of the control object for output
  ctrl <- fwdControl(list(year = ay + mlag, quant = "fbar", value = Ftrgt))

  ## Attach Blim for use in implementation
  attr(ctrl, "Blim") <- Blim

  #return the control object and tracking object
  return(list(ctrl = ctrl, tracking = tracking))
}
```

`forecast_fun` function

This function projects forwards into the next year to set an appropriate TAC. We call this for every year of the simulation and it is similar to the short term forecast we used in the last file.

```
#' Advice implementation using FLasher
#'
#' Carry out a short-term forecast using Flasher
#' to transform advised fishing mortality into an advised catch target.
#'
#' @param stk Object of class \code{FLStock} containing observed stock
```

```
#'            information including commercial catch data, individual mean
#'            weights and biological parameters.
#' @param tracking Tracking object
#' @param args Additional arguments
#' @param forecast Logical. Should a short-term forecast be carried out? Defaults
#'                 to \code{TRUE}
#' @param fwd_trgt Character. Catch or effort target to use during forecast.
#'                 Defaults to 'fsq'.
#' @param fwd_yrs Integer. The number of years to forecast. Defaults to 1.
#' @param fwd_yrs_average Integer vector. The historical data years over which
#'                        biological parameter are averaged for use during
#'                        forecast. Defaults to -3:-1.
#' @param fwd_yrs_rec_start Integer. Starting historical year from which to sample
#'                          projection period recruitment during forecast.
#' @param fwd_yrs_sel Integer vector. The historical data years over which
#'                    catch selection-at-age is averaged for use during forecast.
#'                    Defaults to -3:-1.
#' @param fwd_yrs_lf_remove Integer Vector. ... Defaults to -2:-1.
#' @param fwd_splitLD Logical. Defaults to \code{TRUE}


# === Creating forecast function ===
forecast_fun <- function(stk, tracking, ctrl,
                         args,  # contains ay (assessment year) and management lag
                         fwd_trgt = c("fsq", "hcr"), # fish to status quo (fsq) in
                         # intermediate year and then apply hcr in final calculation
                         fwd_yrs = 2,                # number of years to add
                         # - have a space for the year inbetween to do intermediate
                         # calculations and then for the year we want a quota for
                         fwd_yrs_fsq = -2:0,         # years used to calculate fsq
                         fwd_yrs_average = -3:0,     # years used for averages
                         fwd_yrs_rec_start = NULL, # recruitment -
                         # null uses the entire history to estimate
                         fwd_yrs_sel = -3:-1,        # selectivity of equipment
                         # - average form 3 years ago to 1 year ago
                         fwd_yrs_lf_remove = -2:-1,  # calculate landings to discard
                         # ratio from 2 years ago to 1 year ago
                         fwd_splitLD = TRUE)         # Whether to calculate landings
                         # and discards separately
{
  ## get current assessment year
  ay <- args$ay

  ## get management lag
  mlag <- args$management_lag

  ## checking number of iterations
  niter <- dim(stk)[6]

  ## geomean to estimate recruitment for the stock
  sr0 <- as.FLSR(stk, model = "geomean")
  sr0@params <- FLCore::propagate(sr0@params, iter = niter)

  if (!is.null(fwd_yrs_rec_start)) {
```

```r
if (!is.null(fwd_yrs_rec_start)) {
  stk_n <- window(stk@stock.n, start = fwd_yrs_rec_start)
} else {
  stk_n <- stk@stock.n
}
sr0@params[] <- exp(yearMeans(log(stk_n[1,])))

## remnove years before first data year
minyr <- dims(stk@stock[!is.na(stk@stock.n)])$minyear # min year where data exists
stk0  <- window(stk, start = minyr)

## extend stock object and fill with assumptions calculated based on
# variables defined in function call use fwd_yrs+1 because we need to
# see if the stock crashes at the start of the year after the advice year
stk0 <- FLasher::stf(stk0, fwd_yrs+1)

# FLasher::fwd will throw an error if there are NAs in weights in future years.
# I need to zero these if associated numbers are zero
FLCore::discards.wt(stk0)[FLCore::discards.n(stk0) == 0] <- 0
FLCore::landings.wt(stk0)[FLCore::landings.n(stk0) == 0] <- 0

## find status quo F for each stock
fsq <- yearMeans(fbar(stk)[,as.character(fwd_yrs_fsq+ay-mlag)])

## FLasher cannot handle fsq=0
fsq <- ifelse(fsq==0, 0.001,fsq)

## Construct matrix with rows as years and columns as iterations as before
# and fill it with the Ftarget values
targ <- matrix(0,nrow=fwd_yrs+1, ncol = niter)
targ[1,] <- fsq
targ[2,] <- ctrl@iters[,"value",]
targ[3,] <- ctrl@iters[,"value",]

# year sets the timeline, quant = fbar says we are using fishing mortality
ctrl0 <- fwdControl(list(year  = c(ay,ctrl@target$year,ctrl@target$year+1),
                         quant = "fbar", value = c(targ)))

## project stock forward
stk_fwd <- FLasher::fwd(stk0, sr = sr0, control = ctrl0)

## Extract catch target for the advice year (3 decimal places)
# by looking at the result object
TAC <- round(c(catch(stk_fwd)[,ac(ay+mlag)]),3)

## Get SSB at the end of the advice year (or at the start of the year after?)
TACyr_ssb <- c(ssb(stk_fwd)[,ac(ay+mlag+1)])

## Find iterations where SSB at end of TAC year is < Blim
belowBlim <- which(TACyr_ssb < attr(ctrl, "Blim"))

## Go through model fits - where SSB < Blim, forecast to target Blim
if (length(belowBlim) > 0) {
```

```
    ## define forward control targetting Blim
    targ <- matrix(0,nrow=fwd_yrs+1, ncol = niter)
    targ[1,] <- fsq
    # target is now Blim
    targ[2,] <- c(attr(ctrl,"Blim"))
    targ[3,] <- ctrl@iters[,"value",]

    ctrl_blim <- fwdControl(list(
      year  = c(ay,ay+mlag,ay+mlag+1),
      #focus on biomass in the 2nd year to force us to end
      # at Blim instead if we were below
      quant = c("fbar","ssb_end","fbar"),
      value = c(targ)))

    #rerun simulation focusing on Blim to find the Ftarget that is appropriate
    stk_blim <- FLasher::fwd(stk0, sr = sr0, control = ctrl_blim)

    ## Find iterations with zero TAC advice (some stocks may be in such
    # bad health taht we can't catch any)
    zeroTAC <- ssb(stk_blim)[,ac(ay+mlag+1)] < attr(ctrl,"Blim")

    ## update TAC
    TAC[belowBlim] <- round(c(catch(stk_blim)[,ac(ay+mlag)])[belowBlim],3)
    # set any in zeroTAC to zero manually
    TAC[zeroTAC]   <- 0
  }

  ## Construct fwd control object, which we measurre using catch not fbar now
  ctrl0 <- fwdControl(list(year = ay + mlag, quant = "catch", value = TAC))

  # return the same objects as before
  return(list(ctrl = ctrl0, tracking = tracking))
}
```

Setting forecast arguments

This is how we find the average values for the parameters from the historical data in this simulation. It is similar to the `stfMixME` function we used beforehand, but here we can set it up separately for each stock.

```
# Set Forecast/ISYS Arguments
# Using similar settings to Cod/Had and those defined in ICES_HCR function for whiti
input$ctrl_obj$isys@args$isysList <- list(
    cod = list(fwd_trgt = c("fsq", "hcr"), fwd_yrs = 2, fwd_yrs_fsq = -2:0,
               fwd_yrs_average = -2:0, fwd_yrs_rec_start = 2015,
               fwd_yrs_sel = -3:-1),
    had = list(fwd_trgt = c("fsq", "hcr"), fwd_yrs = 2, fwd_yrs_fsq = 0,
               fwd_yrs_average = -2:0, fwd_yrs_rec_start = 1993,
               fwd_yrs_sel = -3:-1)
)
```

Parallelisation

This process has been set up so that it is easy to run in parallel. We are able to load in the data we need once and then run the simulations for each point we have chosen to sample in the round in parallel. As they run, each process only gathers the values we need and removes the rest of the results of the simulation to save memory. Here, the values we need are the $min(SSB)$ for cod, the $min(SSB)$ for haddock and the total catch of cod and haddock all over the years 2030-2039.

Then, we collect these results and set up the GPs for each of them. After this, we run through the process of removing any implausible points and selecting the points to sample in the next round in a very similar way to the first half of this project. We iterate this process until the optimal point(s) are found. - TACKLE WHEN I AM ABLE TO TACKLE WHY I AM GETTING MULTIPLE POINTS AS AN ANSWER (LIKELY DUE TO HAVING SAME FCOD AND COD BEING THE CHOKE?)

The process above makes up one run of the script. To ensure that we are consistently receiving the same point(s) as our optimal point(s), we want to follow on from the original paper and run this script 1000 times (Spence 2025). This requires a new kind of parallelisation. We were able to accomplish this by creating an array of jobs which will run when there is free space on the Viking HPC used by the Univeristy of York (York, n.d.). This allowed us to submit one job that would run the script 1000 times, making it a lot easier to do this than having to submit 1000 separate jobs (York, n.d.). - SAY ABOUT MY 50 THROTTLE AS WELL?

We will now briefly explain the code for this. Firstly, we have the `doOne` function which runs the simulation for one of the points we have chosen to sample that round (Hofert and Mächler 2016). Note that we must first load in the libraries so that the worker has access to them, similarly to how we must provide the `input_data` and `run_id`. Also note that we remove the large result object (`res`) once we have the values we need to return.

```
# This is the function that will eventually run on each core
doOne <- function(run_id,input_data) {

  #Fixing could not find functions error in runners by loading libraries - DO NOT RE
  library(FLCore)
  library(FLFishery)
  library(mse)
  library(stockassessment)
  library(MixME)
  library(DiceKriging)

  # Retrieve the specific inputs for this run ID
  # We use the 'frozen' input_data table and pick the row matching run_id
  this_Fcod <- input_data$Fcod[run_id]
  this_Fhad <- input_data$Fhad[run_id]

  # Run the simulation
  res <- obj_func(f_cod = this_Fcod, f_had = this_Fhad, mixedfishery_MixME_om = mixe

  # Get Catch directly from the 'tracking' object
```

```
  catch_cod <- sum(res$tracking$cod$stk["C.om", ac(2030:2039)], na.rm = TRUE)
  catch_had <- sum(res$tracking$had$stk["C.om", ac(2030:2039)], na.rm = TRUE)
  # Calculate total catch
  total_catch <- catch_cod + catch_had

  ## // Extract the min ssb at the last ten years of the simulation for both stocks
    # Picking up long term SSB values to see if they dip below Blim at any point
    ssb_cod_data <- c(res$tracking$cod$stk["SB.om", ac(2030:2039)])
    ssb_had_data <- c(res$tracking$had$stk["SB.om", ac(2030:2039)])

    # Getting minimums to model with GPs
    ssb_cod_min <- min(ssb_cod_data, na.rm = TRUE)
    ssb_had_min <- min(ssb_had_data, na.rm = TRUE)

    # Safety Ctahc for if the simulation failed or produced all NAs
    if(is.infinite(ssb_cod_min) | is.na(ssb_cod_min)) ssb_cod_min <- 0
    if(is.infinite(ssb_had_min) | is.na(ssb_had_min)) ssb_had_min <- 0

  # free up memory now that we have the data we need from res
  rm(res)

  # Set up variable to return
  to_return <- c(Fcod = this_Fcod,Fhad = this_Fhad,SSBCod = ssb_cod_min,SSBHad = ssb

  # Return the result
  return(to_return)
}
```

We currently run this on `n_cores` which is one less than the number of cores we can detect the computer has as we needed to keep our computer running during initial testing.

```
n_cores <- parallel::detectCores() - 1
```

For each round, we are sampling `2 * n_cores` points. We are able to apply this `doOne` function to each of these points using the `doMclapply` function (Hofert and Mächler 2016):

```
# Run all the points in parallel, one at a time one each core
result <- doMclapply(points_to_run,doOne = doOne,cores = n_cores)
```

This means we can run each `doOne` on a different core and the first argument provides the data (Hofert and Mächler 2016). Unfortunately, this method does not work on Windows because it relies on forking (Hofert and Mächler 2016).

Further areas for Development

Moving to a more complicated dataset

Mention the one in the MixME paper.

Doing a proper risk calculation

I now meet precautionary, so would it be much of an improvement to use probabilities instead?

References

Hofert, Marius, and Martin Mächler. 2016. “Parallel and Other Simulations in r Made Easy: An End-to-End Study.” *Journal of Statistical Software* 69 (4): 1–44. https://doi.org/10.18637/jss.v069.i04.

ICES. 2019a. “Workshop on Guidelines for Management Strategy Evaluations (WKGMSE2),” January. https://doi.org/10.17895/ices.pub.5531.

———. 2019b. “Haddock (Melanogrammus aeglefinus) in divisions 7.b–k (southern Celtic Seas and English Channel),” June. https://doi.org/10.17895/ices.advice.4785.

———. 2019c. “Cod (Gadus morhua) in Subarea 4, Division 7.d, and Subdivision 20 (North Sea, eastern English Channel, Skagerrak),” November. https://doi.org/10.17895/ices.advice.5640.

———. 2020a. “Cod (Gadus morhua) in Subarea 4, Division 7.d, and Subdivision 20 (North Sea, eastern English Channel, Skagerrak),” June. https://doi.org/10.17895/ices.advice.5891.

———. 2020b. “Haddock (Melanogrammus aeglefinus) in divisions 7.b–k (southern Celtic Seas and English Channel),” October. https://doi.org/10.17895/ices.advice.5897.

———. 2021. “Benchmark Workshop on North Sea Stocks (WKNSEA),” April. https://doi.org/10.17895/ices.pub.7922.

———. n.d. “ICES-FishMap Cod.” https://www.ices.dk/about-ICES/projects/EU-RFP/EU%20Repository/ICES%20FIshMap/ICES%20FishMap%20species%20factsheet-cod.pdf.

Pace, Matthew. 2024. “MixME Wiki.” https://github.com/CefasRepRes/MixME/wiki.

Pace, Matthew, José Oliveira, Simon Fischer, and Paul Dolder. 2025a. “MixME: An r Package to Simulation-test Fisheries Management Robustness to Mixed-fisheries Interactions.” *Methods in Ecology and Evolution* 16 (February): 698–706. https://doi.org/10.1111/2041-210X.70005.

———. 2025b. “"Supplementary Material a: Conditioning a Multi-Stock, Multi-Fleet Operating Model for Celtic Sea Cod, Haddock and Whiting"," February. https://besjournals.onlinelibrary.wiley.com/action/downloadSupplement?doi=10.1111%2F2041-210X.70005&file=mee370005-sup-0001-Supinfo1.pdf.

Roustant, Olivier, David Ginsbourger, and Yves Deville. 2012. “DiceKriging, DiceOptim: Two r Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization.” *Journal of Statistical Software* 51 (1): 1–55. https://doi.org/10.18637/jss.v051.i01.

Spence, Michael A. 2025. “Using History Matching to Speed up Management Strategy Evaluation Grid Searches.” *Canadian Journal of Fisheries and Aquatic Sciences* 82: 1–14. https://doi.org/10.1139/cjfas-2024-0191.

Ulrich, Clara, Douglas C. K. Wilson, J. Rasmus Nielsen, Francois Bastardie, Stuart A. Reeves, Bo S. Andersen, and Ole R. Eigaard. 2012. “Challenges and Opportunities for Fleet- and Métier-Based Approaches for Fisheries Management Under the European Common Fishery Policy.” *Ocean & Coastal Management* 70: 38–47. https://doi.org/https://doi.org/10.1016/j.ocecoaman.2012.06.002.

Williams, Christopher KI, and Carl Edward Rasmussen. 2006. *Gaussian Processes for Machine Learning*. Vol. 2. 3. MIT press Cambridge, MA.

York, University of. n.d. “Viking Documentation.” https://vikingdocs.york.ac.uk/index.html.