# Initial State Estimates

First step pull initial estimates from survey data.

* TrawlData5mmbin.m
* HabCamData5mmbin.m

## TrawlData5mmbin

The original script was computing a density in count / m^2

Detect = 0.4

T2M2 = 1 / (0.3048^2 \* 6076.12\*8) = 1/4515.9

density = count \* T2M2 / Detect)

First mistake in this same script:

I misplaced the detect variable and ended up with it in the numerator.

Now fixed

towArea\_sqm = 4516.;

countPerSqm = 1.0 / (towArea\_sqm \* detect);

density = sum(sizes) \* countPerSqm

Fortunately, this did not factor in as there is no data for the year 2022 in the dredge files.

## HabCamData5mmbin

Second mistake. Carried this detect variable into HabCamData5mmbin script when computing density. I was computing a count divided by SQM and then multiplying by 0.4. This was showing up in the initial estimate data file, bin5mm2022AL.csv. This is why the estimates seemed small.

This is now fixed by removing from computation, detect is not necessary for HabCam data.

The initial state, state(1:ngrids, 1:nsizeClasses) is read in from this file, bin5mm2022AL.csv. The units are scallops / m^2, or just count/m^2

# Recruitment Estimates

Recruitment estimates are likewise pulled from the survey data, both Dredge and Habcam. A combined script is used for this given that HabCam does not need to be adjusted for rock information. This script is named PullOutProcessRecruitData.m. Additional processing is performed to align the recruit geo-position with the survey geo-position which is done via NearestNeighborRecInterp.m

## Pull Estimates from Survey Data

### Dredge Survey

This script first considers the DredgeFile information. For each entry the script sums the count information from 3cm to 6cm, inclusive. Then converts this to a count density in the same manner as initial state data, i.e. count / 4516

As with the initial state the script compensates for rock and depth using the same shapefile, ShapeFiles/Shellfish\_Strata.shp. The data is kept in internal memory data array as:

MDredge = [decYr(:), lat(:), lon(:), utmx(:), utmy(:), depth(:), recA(:)];

### HabCam Survey

Now consider HabCam data. Unlike dredge data, it was simpler to first convert the survey data to a count density by simply dividing the survn colum by the area column.

survn = survn ./ sqm; % convert count to density

And then take the sum from 3cm to 6cm inclusive. Likewise this is stored in internal memory data array as:

MHC=[decYr(:), lat(:), lon(:), utmx(:), utmy(:), depth(:), recM2(:)];

### Pull Out Desired Years

The above arrays are combined for further processing

M = [Mdredge;MHC];

The data is then filtered for each desired year and written to a separate CSV file:

Data/RecruitsYYYYDN.csv

This csv file has the header:

header='decimal year,latitude,longitude,utm x,utm y,bottom depth(m),recruits per sq m';

## Align Recruit Estimates with Selected Data

Aligning the recruit geo-position with the survey geo-position is done in NearestNeighborRecInterp.m. This script works with the data files generated by PullOutProcessRecruitData and aligns the survey grid to a reference year that coincides with the start year of the simulation. NearestNeighborRecInterp reads in the recruit estimate that matches the start year, e.g. 2022 and AL

Data/Recruits2022AL.csv

and uses this as a reference to aligns each of the remaining recruit estimate years to that file using a nearest neighbor approach. These realigned files are written out to:

RecruitEstimates/RecruitEstimateDNYYY.csv

It is these data files that the GeoSAMS simulation reads in to determine the recuitment. These data files are randomly selected with a weighted function that favors the more recent years.

# GeoSAMS

For each timestep, 1 to number of years \* timeStepPerYear.

In the following

* ‘:’ is short for 1:numSizeClasses

## Compute Fishing Effort

Fishing effort is dependent on many other parameters

* Exploitable scallops in count per square meter, or explScallops\_psqm
* Exploitable biomass in grams per square meter, or explBiomass\_gpsqm
  + Which is dependent on exploitable scallops at size in count per square meter, explScallopsAtSize\_psqm
* Total Catch
  + Which is dependent on landings by weight, landingsByWght\_grams
    - Which is dependent on landings by size, landingsAtSize
      * Which is dependent on fishing mortality, Fmort
        + Which is dependent of landing per unit effort, lpue

Thus computing these parameters in reverse order

1. explScallops\_psqm
2. explScallopsAtSize\_psqm
3. explBiomass\_gpsqm
4. lpue
5. Fmort
6. landingsAtSize
7. landingsByWght\_grams
8. totalCatch
9. Finally, fishingEffort

### Compute exploitable scallops in count/m^2, explScallops\_psqm

explScallops\_psqm(loc) = dot\_product(selectivity(loc, :), state(loc,:))

### Compute exploitable scallops at size, count/m^2, explScallopsAtSize\_psqm

For given grid loc

explScallopsAtSize\_psqm(:) = selectivity(loc,:) \* state(loc,:)

### Compute exploitable biomass in g/m^2, explBiomass\_gpsqm

explBiomass\_gpsqm(loc) =   
dot\_product(explScallopsAtSize\_psqm(:), weightGrams(loc,:))

### Compute Landings Per Unit Effort, landings/day, LPUE

LPUE needs additional computations based on explBiomass\_gpsqm, explScallops\_psqm

* Mean exploitable weight, meanExplWght\_g
* explBiomassInTow\_g

#### Compute mean exploitable weight

meanExplWght\_g = explBiomass\_gpsqm / explScallops\_psqm

#### Compute exploitable biomass in tow, g

explBiomassInTow\_g = explBiomass\_gpsqm \* towArea\_sqm  
 = explBiomass\_gpsqm \* 4516

#### Compute LPUE

LPUE = min(lpuePPD, lpueLimitPPD)

##### Compute Pound Per Day, lpuePPD

lpuePPD is a piecewise linear relationship between exploitable biomass and lpuePPD.

lpueSlope, lpueSlope2, and lpueIntercept are dependent on MA or GB regions

A = lpueSlope2 \* explBiomassInTow\_g

B = lpueSlope \* explBiomassInTow\_g + lpueIntercept

lpuePPD = min(A,B)

##### Compute Limit Pound Per Day, lpueLimitPPD

maxPerDay: max number of scallops that can be shucked in a day

lpueLimitPPD = maxPerDay \* meanExplWght\_g / gramsPerPound

### Compute fishing mortality, Fmort

#### Compute mortality constant, Cmort

fishingMort is a configuration parameter typically set to 0.4

alphaMort is a configuration parameter typically set to 1.0

Cmort = fishingMort \* sum(explBiomass\_gpsqm(1:ngrids)) /  
 dot\_product(explBiomass\_gpsqm(1:ngrids), LPUE(1:ngrids)^alphaMort)

#### Compute initial estimate of Fmort(1:ngrids)

Fmort(1:ngrids) = Cmort \* LPUE(1:ngrids)^alphaMort

#### Compute final Fmort(1:ngrids)

if grid is closed: Fmort = 0.0

else no change to Fmort

Unless grid is in a special access area then use Fmort for that special area.

### Compute landingsAtSize(:)

(1.0 - exp(-F \* delta\_time)) \* state \* grid\_area\_sqm \* selectivity, that is

At each grid location, loc

landingsAtSize(:) = (1.0 - exp(-Fmort(loc) \* delta\_time)) \* state(loc, :)   
\* gridArea\_sqm \* selectivity(loc, :)

### Compute landingsByWght\_grams(loc)

(1.0 - exp(-F \* delta\_time)) \* dot\_product(selectivity \* state \* grid\_area\_sqm, weight), that is

landingsByWght\_grams(loc) = dot\_product( landingsAtSize(:), weight\_grams(loc,:))

### Total Catch

totalCatch = sum(landingsByWght\_grams)

### Combine Terms

if grid(loc) is closed OR explScallops\_psqm == 0.0: fishingEffort(loc) is 0.0

else

E = sum( explBiomass\_gpsqm(1:ngrids)^2 / explScallops\_psqm(1:ngrids) )

fishingEffort = ( explBiomass\_gpsqm(loc) \* totalCatch / explScallops\_psqm(loc) )  
/ ( E \* gridArea\_sqm)

## Evaluate Growth

For each grid, loc, evaluate growth in Time To Grow

1. Compute natural mortality based on current state\_vector, state(loc, :)
2. Adjust population state\_vector based on von Bertalanffy growth
3. Adjust population state\_vector based on recruitment
4. Compute overall mortality
5. Apply mortality and compute new state\_vector

### Compute natural mortality based on current state

#### Natural Juvenile Mortality, naturalMortJuv

naturalMortJuv is dependent on the recruit size classes. Therefore, find recruits the number of recruits at the given grid location

recruits = sum(state(loc, 1:max\_rec\_ind)) \* gridArea\_sqm / (10.\*\*6)

if recruits > 1400

GB: naturalMortJuv = max( naturalMortAdult , exp(1.226\_dp \* log(recruits) - 10.49\_dp ))

MA: max( naturalMortAdult , exp(1.093\_dp \* log(recruits) - 9.701\_dp) )

else

GB: naturalMortJuv = 0.2

MA: 0.25

#### Collect terms

length\_0 = GB: 70, MA: 65

mortAlpha = 1 - 1 / ( 1 + exp( - (shellLengths(:) - length\_0) /10.\_dp ) )

naturalMortality(:) = mortAlpha(:) \* naturalMortJuv + (1.0 - mortAlpha(:)) \* naturalMortAdult

### Adjust population state\_vector based on von Bertalanffy growth

state(loc, :) = matmul(G(:, :), state(loc, :))

### Add Recruitment, count/m^2

If not first year, and if in recruitment period for the year, typically Jan1 to Apr10 then

state(:) = state(:) + recuitment(:) / recrPeriod

### Compute Overall Mortality

***incidental*** and ***discard*** are configuration parameters

discard(:) = Set\_Discard(shellLen(:), selectivity(loc, :), cullSize, ***discard***, gridIsClosed)

if ((length .gt. cull\_size) .or. is\_closed) then

Set\_Discard = 0.0

else

Set\_Discard = discard \* selectivity

endif

M(:) = naturalMortality(:) + fishingEffort \* ( selectivity(:) + ***incidental*** + discard(:) )

### Apply mortality and compute new state\_vector

state(loc, :) = state(:) \* (1.0 - deltaTime \* M(:))

## Additional Output Parameters

These parameters are computed at each timestep

### Abundance, number of scallops

abundance(loc) = sum(state(loc,:)) \* gridArea\_sqm

### Biomass, in g/m^2

bms(loc) = dot\_product(state(loc,:), weight\_grams(loc,:)

### Exploitable Biomass, in metric tons

ebms\_mt(loc) = explBiomass\_gpsqm(loc) \* gridArea\_sqm / 1e6

### Landings by Number

landingsByNum(loc) = sum(landings\_at\_size(:))

### Outputs

The following data is written at each timestep to files named   
Results/Lat\_Lon\_Surv\_OOOO\_DN.csv, where OOOO is one of the following and DN is domain name. These files are used to show the results before interpolation, that is at the survey grid, ngrids.

* ABUN, count / m^2: abuncance(1:ngrids)
* BIOM, g/m^2: bms(1:ngrids)
* EBMS, metric tons: ebms\_mt(1:ngrids)
* FEFF, unitless: fishingEffort(1:ngrids)
* FMOR, unitless: Fmort(1:ngrids)
* LAND, number of scallops: landingsByNum(1:num\_grids)
* LNDW, grams : landingsByWght\_grams(1:num\_grids)
* LPUE, pound per day : lpue(1:num\_grids)
* RECR, count / m^2 : recruits(1:num\_grids)

This same data is written to X-Y coordinates to files in the format  
Data/X\_Y\_OOOO\_DN\_YYYY\_[MA|GB] for interpolation to the region grids.

# UK Interpolation

Skipping the detail for now. The UK Grid Manager

* Reads in the region of interest from Grids/[MA|GB]xyzLatLon.csv
* Reads in the survey data for interpolation for each year of interest from:  
  Data/X\_Y\_OOOO\_[AL|MA|GB]\_YYYY\_[MA|GB]
* Produces the output file:  
  Results/Lat\_Lon\_Grid\_OOOO\_[AL|MA|GB]\_StartYear\_StopYear.csv

## Output Limit

UK Interpolation will limit the output for each desired parameter based on the maximum value found in the growth simulation parameter data. This limit is determined by the function limitz. The function is the limit Z. Grid is a data structure that holds the simulated growth data.

* grid%field: The output is in the member variable
* grid%z: depth of output
* fpeak: max value from input data

do j=1,ngrids

if (grid%lon(j) > ma\_gb\_border) then ! GB

zc = 70.\_dp

w = 50.\_dp

a = 0.01\_dp

kappa = 20.\_dp

else

zc = 60.\_dp

w = 33.\_dp

a = 0.01\_dp

kappa = 10.\_dp

endif

grid%field(j) = max( grid%field(j), 0.\_dp )

fmax= fpeak \* (a + exp(-((grid%z(j) - zc) / w )\*\*kappa))

grid%field(j) = min( grid%field(j) , fmax )

enddo

**~~Potential issue here: Keston mentioned that there is a software switch to limit magnitude of the output to the max of the input. Where is this variable? I believe you referred to this as x0Max in original code.~~**

# GUI

For the SortByRegion frame we have been reviewing, specifically BIOM

## Sort Data

For each data point in Results/Lat\_Lon\_Grid\_OOOO\_[AL|MA|GB]\_StartYear\_StopYear.csv

* Sort by year
* Check if data point position Lat/Lon is within a given region
  + If so accumulate the data  
    accumParamData[row][col] += paramData[col]  
    where row is region of interest, col is year and paramData is what was read from file
  + Also keep track of how many data points were added  
    countData[row][col] += 1

## Display Results

Again looking at just BIOM, the GUI converts the accumulated data from g/m^2 to metric tons. From the shapefile we know the area of the region and it is given in Km^2. To convert g/m^2 to mt is a simple multiplication: g/m^2 \* Km^2

* BIOM is in g/m2
* mt = 1e6 g
* g/m2 \* Km2 \* (1e6 m2/Km2) \* (mt / 1e6 g)
* mt = g/m2 \* Km2
* accumParamData[row][col] = accumParamData[row][col] \* shape[row].areaKm2   
  / countData[row][col]

*~~Keston pointed out that I should be taking the mean of this data point~~*

* *~~display data = accumParamData[row][col] \* areaKm2 / countData[row][col]~~*

*~~This does reduce the overall value but not the trend.~~*