

Explanation of Mathematical Methods Behind the First Half of the Project

AUTHOR

Emma Bowen

Introduction

What mathematical methods will I explain in this document?

- Gaussian processes
- Bayesian History Matching
- Kriging
- Expected Improvement
- Augmented Expected Improvement
- Knowledge Gradient
- K-Means Clustering

Context

We are building on the paper “Using history matching to speed up management strategy evaluation grid searches”. This paper is looking to find the Harvest Control Rule parametrised by F_{target} and $B_{trigger}$ that maximises the median long-term catch whilst keeping the risk below 0.05 for a single-stock fishery. The paper does not consider fleet dynamics. We have an objective function in the paper which combines the risk and the catch and so this is the function we want to maximise to get our maximal median catch with the constraint of keeping the risk below 0.05.

To do this, the paper takes a Bayesian History Matching (BHM) approach. Firstly, we sample our first eight points which are spaced evenly throughout the sample space to get some initial data. Then, for each round we do the following:

- We set up or update the Gaussian Process (GP) to model the risk and the GP to model the median catch
- We can then use the risk as a threshold so that we only consider the values of F_{target} and $B_{trigger}$ that have risk below 0.05
- We use the GP which is modelling the median catch to get the value for the median 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 low probability of being higher than the current best median catch)
- We select 8 plausible points to sample in the next round

We repeat this process until there is only one plausible point left and then we will accept this as being the F_{target} and $B_{trigger}$ that maximise the median catch whilst keeping the risk below 0.05.

Now, we will look at the mathematics behind the methods above and also at the acquisition functions of Expected Improvement, Augmented Expected Improvement and Knowledge Gradient which I added to the original code.

General theory

Gaussian Processes

A Gaussian Process F has a mean function μ_0 and a covariance function $\text{cov}_0(x_i, x_j)$. We can then evaluate the covariance function $\text{cov}_0(x_i, x_j)$ for every pair x_i, x_j where $i, j \in \{1, \dots, n\}$ to find the covariance matrix $\Sigma_{1:n}$. Then, F is a probability distribution over our objective function f with the property that, for any given collection of points x_1, \dots, x_n , the marginal probability distribution on $F(x_{1:n}) = (F(x_1), \dots, F(x_n))$ is given by (Frazier 2018):

$$F(x_{1:n}) \sim N((\mu_0(x_{1:n}), \Sigma_{1:n})) \quad (1)$$

where

$$\mu_0(x_{1:n}) = (\mu_0(x_1), \dots, \mu_0(x_n))$$

We choose a covariance function such that inputs that have nearby points that have been evaluated have a more certain output than points that are further away from the points that have been evaluated (Spence 2025). This is equivalent to saying that if for some x, x', x'' in the design space we have $\|x - x'\| < \|x - x''\|$ for some norm $\|\cdot\|$, then $\text{cov}_0(x, x') > \text{cov}_0(x, x'')$ (Frazier 2018).

We use a GP to emulate the objective function because it is much cheaper to evaluate than our objective function. We can calculate $F(x)$ for any x in the design space as our estimate of $f(x)$ based on our current beliefs. This is true even for the evaluated points x_1, \dots, x_m as the emulator is fitted to these points (Spence 2025).

Maximum Likelihood Estimation

When using GPs to emulate our objective function, we need to be able to estimate the coefficients of the objective function using the data we gain from evaluating our points x_1, \dots, x_n (Frazier 2018).

We can do this as follows. Firstly, we let the vector η represent the hyperparameters that give us μ_0 and cov_0 . Then, given the observations $f(x_{1:n}) = (f(x_1), \dots, f(x_n))$, we calculate the likelihood of these observations under the prior given η which is denoted as $p(f(x_{1:n})|\eta)$ and modelled by Equation 1. Lastly, we set $\hat{\eta}$ to the value that maximizes this likelihood Frazier (2018):

$$\hat{\eta} = \text{argmax}_{\eta} p(f(x_{1:n})|\eta)$$

Kriging

Kriging is a Bayesian statistical method for modelling functions (Frazier 2018). Again, let f be the objective function and we focus on the design space $X := \{x_1, \dots, x_n\}$. Now, if we have evaluated n points such that we have $f(x_{1:n})$ and want to evaluate x_{n+1} we let $k = n + 1$ in Equation 1. Then, we can compute the conditional distribution of $F(x_{n+1})$ given $f(x_{1:n})$ using Bayes' rule:

$$F(x_{n+1})|f(x_{1:n}) \sim N(\mu_n(x_{n+1}), \sigma_n^2(x_{n+1})) \quad (2)$$

where:

$$\mu_n(x_{n+1}) = \text{cov}_0(x_{n+1}, x_{1:n})(\Sigma_{1:n})^{-1}(f(x_{1:n}) - \mu_0(x_{1:n})) + \mu_0(x_{n+1})$$

$$\sigma_n^2(x_{n+1}) = \text{cov}_0(x_{n+1}, x_{n+1}) - \text{cov}_0(x_{n+1}, x_{1:n})(\Sigma_{1:n})^{-1}\text{cov}_0(x_{1:n}, x_{n+1})$$

where:

$$\text{cov}_0(x_{n+1}, x_{1:n}) = (\text{cov}_0(x_{n+1}, x_1), \dots, \text{cov}_0(x_{n+1}, x_n))$$

This conditional distribution $F(x_{n+1})|f(x_{1:n})$ is called the posterior probability distribution for x_{n+1} . We can calculate this distribution for every point in the design space X . This results in a new GP F_n with a mean vector and covariance kernel that depend on the location of the unevaluated points, the locations of the evaluated

and covariance kernel that depend on the location of the unevaluated points, the locations of the evaluated points $x_{1:n}$, and their values $f(x_{1:n})$ (Frazier 2018). So, we can update our GP every round based on the new points we have evaluated.

Bayesian History Matching

Let x be a point in the sample space. We begin with some uncertainty about our objective function $f(x)$ (Spence 2025). However, we can make probabilistic statements such as:

$$P(f(x) > a) = \int_a^{\infty} P(f(x)) df(x)$$

Once we evaluate another point x' where $x' \neq x$, we are able to use Bayes' Theorem improve our integral to:

$$P(f(x) > a | f(x')) = \int_a^{\infty} P(f(x) | f(x')) df(x)$$

We now let $a = \max\{f(x_1), \dots, f(x_n)\}$ where n is the number of points we have evaluated so far. For the first round, $n = 8$ but as the rounds increase we make sure to include all previous points of the objective function that have been evaluated.(Spence 2025) We remove the point x if:

$$P(f(x) > a | f(x_{1:n})) = \int_a^{\infty} P(f(x) | f(x_{1:n})) df(x) < \varepsilon \quad (3)$$

for a small $\varepsilon > 0$ until no plausible points remain. Then, the optimum will be x^* such that:

$$f(x^*) = \max\{f(x_{1:n})\}$$

as our index n counts the number of points we have evaluated throughout the whole simulation.

Expected Improvement

The first type of acquisition function we will look at is Expected Improvement (EI). Suppose we have sampled the points x_1, \dots, x_n and observe the values $f(x_{1:n})$. Then, if we were to return a solution at this point, bearing in mind we observe the objective function f without noise and we can only return points we have already evaluated, we would return $f_n^* = \max\{f(x_1), \dots, f(x_n)\}$ (Frazier 2018). Imagine we then consider evaluating another point x_{n+1} to get $f(x_{n+1})$. We can then define the Expected Improvement as:

$$EI_n(x_{n+1}) := E[F(x_{n+1}) | f(x_{1:n}) - f_n^*]^+$$

where $[F(x_{n+1}) - f_n^*]^+$ is the positive part of $[F(x_{n+1}) - f_n^*]$. This acquisition function is relatively easy to optimise and many different methods have been developed for doing this (Frazier 2018).

There is another expression for $EI_n(x_{n+1})$:

$$EI_n(x_{n+1}) = [(\mu_n(x_{n+1}) - f_n^* \cdot \Phi(Z)) + (\sigma_n(x_{n+1}) \cdot \phi(Z))]^+ \quad (4)$$

where again the notation $[\cdot]^+$ means the positive part and where:

$$Z = \frac{\mu_n(x_{n+1}) - f_n^*}{\sigma_n(x_{n+1})}$$

[Equation 4](#) can be gained from [Equation 1](#) by setting $k = n + 1$ and then studying the distribution of $F(x_{n+1}) - f_n^*$. However, we can also consider it as a version of Equation (15) from (Jones, Schonlau, and Welch 1998) where we first flip the signs as we are focused on the maximisation case and then set $f_{min} = f_n^*$, $\hat{y} = \mu_n(x)$ and $s = \sigma_n(x)$.

Auamended Expected Improvement

This was included to help make the method perform better for noisy functions which will make it more generally applicable ([Huang et al. 2006](#)). To deal with these noisy observations, a change was proposed to standard EI function as detailed below. This change seems mostly to have been justified by empirical performance ([Letham et al. 2018](#)).

By adjusting Equation (12) found in ([Huang et al. 2006](#)) to our own notation, we get that:

$$AEI_n(x_{n+1}) = E_n[F(x_{n+1})|f(x_{1:n}) - f_{eb}^*]^+ \left(1 - \frac{\sigma_{obs}}{\sqrt{\sigma_n^2(x_{n+1}) + \sigma_{obs}^2}} \right)$$

where σ_{obs} is the standard deviation of the noise variable set by the user and $\sigma_n(x)$ is the standard deviation of GP F at the n^{th} iteration, as used beforehand. We have also changed f_n^* to f_{eb}^* which is the highest predicted mean at any sampled point so far so that we take into account that the uncertainty in our observations could cause a large spike ([Huang et al. 2006](#)).

Knowledge Gradient

We remove the assumption of EI that we have to return a pre-evaluated point as our best point ([Frazier 2018](#)). This allows us to do some different computations to the ones in EI. We also now start by saying that the solution we would choose if we have to stop sampling after n points would be the point in the design space with the largest $\mu_n(\cdot)$ value, where $\mu_n(\cdot)$ is the mean vector of the posterior probability distribution after n iterations. We call this maximum x_n^* and then can say that $F(x_n^*)$ is random under the posterior distribution and has the mean vector after sampling $f(x_{1:n})$ of:

$$\mu_n^* := \mu_n(x_n^*) = \max_x \mu_n(x)$$

where x is any point in the sample space ([Frazier 2018](#)).

Then, we imagine that we are now allowed to sample a new point x_{n+1} . We get a new posterior distribution at the point x which we can calculate using [Equation 2](#) by replacing x_{n+1} with x and $x_{1:n}$ with $x_{1:n+1}$ to include our new observation. This will have the posterior mean function $\mu_{n+1}(\cdot)$ and the conditional expected value for $F(x_n^*)$ changes to be:

$$\mu_{n+1}^* := \max_x \mu_{n+1}(x)$$

So, we can see that the increase in the conditional expected value of $F(x_n^*)$ by sampling the new point x_{n+1} is ([Frazier 2018](#)):

$$\mu_{n+1}^* - \mu_n^*$$

While this quantity is unknown before we sample x_{n+1} we can calculate its expected value given our observations x_1, \dots, x_n . The Knowledge Gradient for sampling at a new point x in the design space is defined as ([Frazier 2018](#)):

$$KG_n(x) := E_n[\mu_{n+1}^* - \mu_n^* | x_{n+1} = x] \quad (5)$$

where again E_n indicates the expectation taken under the posterior distribution at the n^{th} iteration. We would sample the point x with the largest $KG_n(x)$ as our next point ([Frazier 2018](#)).

The easiest way to calculate the KG is via simulation. This can be done by simulating one possible value for $f(x_{n+1})$. Then, we calculate μ_{n+1}^* and subtract μ_n^* . We iterate this process many times so that we can find the average of $\mu_{n+1}^* - \mu_n^*$ and this allows us to estimate $KG_n(x)$ ([Frazier 2018](#)). This process, or calculating [Equation 5](#) directly from the properties of the normal distribution, both work well in discrete, low dimensional problems which is the situation we are in for the first half of the project ([Frazier 2018](#)).

Alternatively, we can calculate μ_{n+1} using the formula below ([Ungredda, Pearce, and Branke 2022](#)):

$$\mu_{n+1}(x) = \mu_n(x) + \frac{\text{cov}_n(x_{n+1}, x)}{\text{var}_n(x_{n+1}) + \sigma_{\text{obs}}^2} (F(x_{n+1}) - \mu_n(x_{n+1})) \quad (6)$$

where σ_{obs} is a noise variable which can be determined by the user ([Ungredda, Pearce, and Branke 2022](#)). From the GP for median catch, we get $\text{cov}_n(x_{n+1}, x)$ and the standard deviation σ_{obs}^2 .

Kmeans process for selecting multiple points

Combining a clustering method with an acquisition function was an idea I had early on in the project. I was then able to find literature on the subject, including using kmeans clustering.

We have now been able to determine which points are possible based on the probability that their median catch is higher than the current maximum median catch (using Bayesian History Matching) and the probability that their risk is less than 0.05. These points will be called the Possible Space, PS . We have then assigned a value to each point in PS using one of the acquisition functions above. Now, we want to decide which 8 points are best to evaluate next.

For sampling only one point next, this would be very simple as you would take the point with the highest value assigned by the acquisition function ([Frazier 2018](#)). However, we want to sample 8 points next so that we continue the pattern set up in the original paper and we want a good trade-off between exploration and exploitation ([Spence 2025](#)),([Azimi, Fern, and Fern 2010](#)).

So, we use the `kmeans` function which is part of the stats package in R (which is automatically loaded into an R session). This function uses the algorithm from Hartigan and Wong, 1979 by default([Hartigan and Wong 2018](#)), ([RDocumentation 2025](#)). The k-means clustering method is the most commonly used due to its simplicity compared to other clustering algorithms([Kodinariya, Makwana, et al. 2013](#).)

This algorithm creates k clusters (groups of points) such that the points within each cluster have the sum of squares to the centre of their cluster smaller than it would be to the centre of any other cluster ([RDocumentation 2025](#)). It starts by defining k centroids which should be placed as much as possible far away from each other ([Kodinariya, Makwana, et al. 2013](#)). Then, we take each point in the space and associate it to the nearest centroid. We stop when every point has been assigned to a centroid ([Kodinariya, Makwana, et al. 2013](#)). At this point, we re-calculate k new centroids as the centers of the clusters created by the previous step. This may result in some points changing clusters ([Hartigan and Wong 2018](#)). We repeat this process until no points change clusters ([Hartigan and Wong 2018](#)),([Kodinariya, Makwana, et al. 2013](#)).

However, before the algorithm can start, we must specify how many clusters we want ([Kodinariya, Makwana, et al. 2013](#)). This can be difficult in many cases ([Kodinariya, Makwana, et al. 2013](#)). In our case, it is relatively simple as we know how many points we want to sample next and so we set this to be the number of clusters. We then run the algorithm on PS to form the clusters. Then, we pick the point with the highest value of the acquisition function from each cluster to sample in our next round. This allows us to search for viable points by looking in the Possible Space but also to keep the points we are going to sample spread out so that we can balance exploitation and exploration more effectively ([Azimi, Fern, and Fern 2010](#)).

Application of theory in my project

Set up the Gaussian Processes

We are focusing on maximising the objective function from the paper which is given below ([Spence 2025](#)):

$$f(\theta) = I_{[0.95,1]}(P(B(\theta) > B_{\text{lim}})) \times C(\theta)$$

where $C(\theta)$ is median long term catch, $B(\theta)$ is long-term SSB and:

$$I_{[0.95,1]}(x) = \begin{cases} 1 & \text{if } x \in [0.95, 1] \\ 0 & \text{otherwise} \end{cases}$$

is an indicator function.

To do this, Spence uses two GPs where the risk GP models $\ln(risk)$ and the catch GP models $\ln(median\ catch)$ ([Spence 2025](#)). Maintaining the notation from the Spence 2025 paper, we use m_1 for the mean function of the catch GP and m_2 for the mean function of the risk GP ([Spence 2025](#)):

$$\begin{aligned} m_1(\phi) &= \beta_{1,0} + \beta_{1,1}(\ln(\phi_1 + 0.1)) + \beta_{1,2}(\ln(\phi_1 + 0.1))^2 + \\ &\quad \beta_{1,3}(\ln(\phi_1 + 0.1))^3 + \beta_{1,4}(\phi_2 \ln(\phi_1 + 0.1)) + \beta_{1,5}\phi_2 \\ m_2(\phi) &= \beta_{2,0} + \beta_{2,1}\phi_1 + \beta_{2,2}\phi_2 + \beta_{2,3}\phi_1\phi_2 \end{aligned}$$

where all of the above $\beta_{s,t}$ for $s \in \{1, 2\}$ and $t \in \{1, 2, 3, 4, 5\}$ are coefficients to be found through maximum likelihood estimation and:

$$\phi_1 = \frac{F_{target} - 0.1}{0.4} \quad \text{and} \quad \phi_2 = \frac{B_{trigger} - 110000}{90000}$$

where we rescale for numerical stability in the GP ([Spence 2025](#)).

Our covariance function c for both GPs is the variance σ_i^2 (which is acting as a scalar) times the Ornstein-Uhlenbeck correlation function ([Spence 2025](#)):

$$r_i(\phi, \phi', \delta_i) = \exp\left(-\frac{|\phi_1 - \phi'_1|}{\delta_{i,1}} - \frac{|\phi_2 - \phi'_2|}{\delta_{i,2}}\right)$$

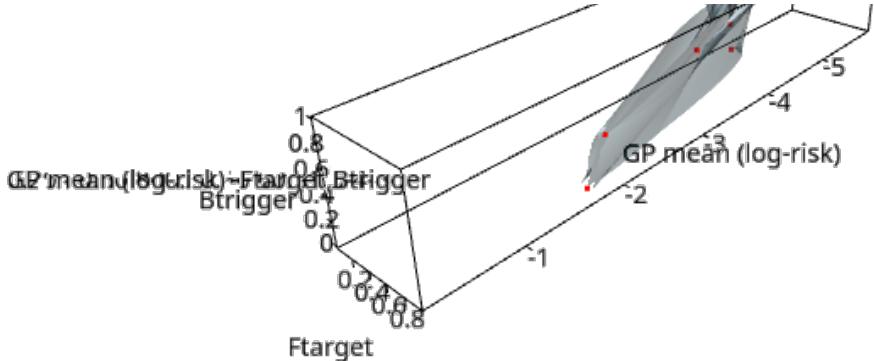
where $\delta_{i,1}$ and $\delta_{i,2}$ are the length scales for each of the ϕ_1 and ϕ_2 terms respectively ([Williams and Rasmussen 2006](#)).

We need to sample our first round of eight points before setting up the GPs so that we have enough data to estimate all of the coefficients in our GPs ([Jones, Schonlau, and Welch 1998](#)). Note that until we have sampled sixteen points, we set the prior of the catch GP to be the same as the risk GP, $m_2(\phi)$, except with different coefficients ([Spence 2025](#)). This is because we need to estimate the coefficients for the mean function for the catch GP and the length scales for the covariance function c from the same data ([Jones, Schonlau, and Welch 1998](#)), ([Roustant, Ginsbourger, and Deville 2012](#)). For mathematical stability, these functions should have less than eight coefficients between them in our first round as we are only sampling eight points per round due to computational limitations encountered at the time of the Spence 2025 paper ([Jones, Schonlau, and Welch 1998](#)), ([Spence 2025](#)). However, after our first round we reset the mean function for the catch GP to $m_1(\phi)$ ([Spence 2025](#)).

We can set up Gaussian Processes (GPs) as described above in R using the DiceKriging package and use maximum likelihood estimate to get the hyperparameters of $m_1(\phi)$, $m_2(\phi)$ and c for our GPs ([Roustant 2025](#)), ([Roustant, Ginsbourger, and Deville 2012](#)). Then, we use them to predict the $\ln(median\ catch)$ and $\ln(risk)$ at every point in the design space. We can then exponentiate these results where needed. We are building our GPs with \ln of the values we want because this helps us generate better predictions ([Huang et al. 2006](#)).

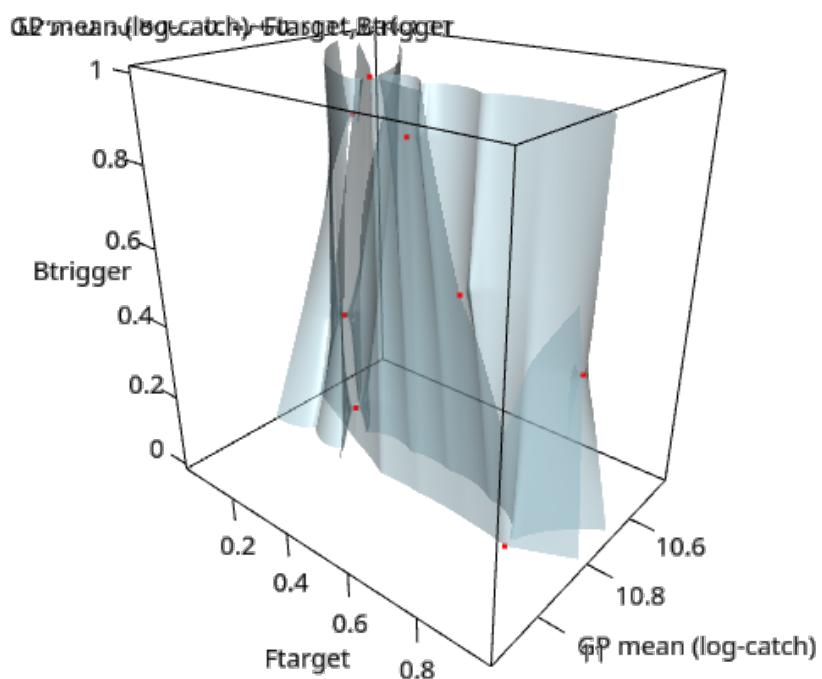
We have been able to visualise the GPs after the first round in 3D:





The middle plane is the mean of the risk GP, whereas the bottom and top planes represent the lower and upper bounds of the 95% confidence interval respectively. The planes meet at the evaluated points, which are the red dots on the diagram. The scales are odd due to the re-scaling of F_{target} and $B_{trigger}$ and fitting our GPs to log of the values we want throughout the code which helps to keep the GP stable ([Huang et al. 2006](#)).

We can then also do this for the catch GP:



Excluding implausible points

Recall that we have the GP for $\ln(\text{median catch})$ which is modeling the $C(\theta)$ part of the objective function. As the first half of the objective function f is an indicator function, we can focus on maximising $C(\theta)$ whilst meeting the threshold stated by the indicator function.

Firstly, we enforce the threshold in the indicator function by calculating $P(\ln(\text{risk}) \leq \ln(0.05))$, which is equivalent to the precautionary threshold $P(\text{risk} \leq 0.05)$. We calculate this by making predictions for our risk at each point using our risk GP and finding the probability that these predictions for risk are below $\ln(0.05)$. This probability can be found using [Equation 2](#). Then, we exclude any points with $P(\ln(\text{risk}) \leq \ln(0.05)) \leq \varepsilon = 0.0001$ by setting their value for the acquisition function equal to 0. This means they will not be chosen as a point to sample in the next round.

To maximise $C(\theta)$, we use Bayesian history matching to speed up the process by removing any points that are implausible according to [Equation 2](#).

implausible according to [Equation 3](#).

This full process ensures that only points that meet the precautionary threshold and are plausible according [Equation 3](#) can be selected.

Deciding on next point to sample

We have three different acquisition functions we have investigated using here.

Expected Improvement

We use the [Equation 4](#) expression and calculate the EI for every point in the sample space, setting those that don't meet the precautionary threshold to zero as explained above.

Augmented Expected Improvement

As our situation has no noise, in our code we are still using f_n^* ([Spence 2025](#)). Thus, we use the equation:

$$AEI_n(x_{n+1}) = EI_n(x_{n+1}) \left(1 - \frac{\sigma_{obs}}{\sqrt{\sigma_n^2(x_{n+1}) + \sigma_{obs}^2}} \right)$$

calculate the AEI for every point in the sample space. We again make sure to set any points not meeting the precautionary threshold to have AEI = 0.

Knowledge Gradient

We estimate the KG from 100 simulations and update the KG each round using [Equation 6](#). As in the other acquisition functions, we make sure to set the value to zero for any point in the sample space not meeting the precautionary threshold.

Kmeans process

To then make sure that the next points we sample are spread out across the sample space, we use the Kmeans process ([Azimi, Fern, and Fern 2010](#)). We create eight clusters and select the point with the highest value of the acquisition function in each cluster. These are the points we will sample in the next round.

Updating our GPS

In our second round, we need to update our GPs with new data ([Spence 2025](#)). We do this by adding in the data for the points that have been newly evaluated this round. Hence, we can do the calculations from the Kriging section to update the GP with a new mean vector and covariance kernel for our next round. For the catch GP, we move on to using $m_1(\phi)$ from the second round onwards because by the time we create this GP we have sampled 16 points([Spence 2025](#)).

Now, we repeat the full process described in the Application of theory in my project section until there are no points with a $KG > 0$. Then, the optimal point is the x^* that has the highest catch out of the precautionary points. In our case, x^* is the F_{target} and $B_{trigger}$ that will give the highest median long-term catch whilst following the precautionary principle ([Spence 2025](#)).

References

- Azimi, Javad, Alan Fern, and Xiaoli Fern. 2010. "Batch Bayesian Optimization via Simulation Matching." In *Advances in Neural Information Processing Systems*, edited by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2010/file/e702e51da2c0f5be4dd354bb3e295d37-Paper.pdf.
- Frazier, Peter. 2018. "A Tutorial on Bayesian Optimization." <https://doi.org/10.48550/arXiv.1807.02811>.

- Hartigan, J. A., and M. A. Wong. 2018. "A k-Means Clustering Algorithm." *Journal of the Royal Statistical Society Series C: Applied Statistics* 28 (1): 100–108. <https://doi.org/10.2307/2346830>.
- Huang, D., Theodore Allen, William Notz, and Ning Zheng. 2006. "Global Optimization of Stochastic BlackBox Systems via Sequential Kriging Meta-Models." *Journal of Global Optimization* 34: 441–66. <https://doi.org/10.1007/s10898-005-2454-3>.
- Jones, Donald, Matthias Schonlau, and William Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions." *Journal of Global Optimization* 13: 455–92. <https://doi.org/10.1023/A:1008306431147>.
- Kodinariya, Trupti M, Prashant R Makwana, et al. 2013. "Review on Determining Number of Cluster in k-Means Clustering." *International Journal* 1 (6): 90–95.
- Letham, Benjamin, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. 2018. "Constrained Bayesian Optimization with Noisy Experiments." <https://arxiv.org/abs/1706.07094>.
- RDocumentation. 2025. <https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/kmeans>.
- Roustant, Olivier. 2025. <https://www.rdocumentation.org/packages/DiceKriging/versions/1.6.1>.
- 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>.
- Ungredda, Juan, Michael Pearce, and Juergen Branke. 2022. "Efficient Computation of the Knowledge Gradient for Bayesian Optimization." <https://arxiv.org/abs/2209.15367>.
- Williams, Christopher KI, and Carl Edward Rasmussen. 2006. *Gaussian Processes for Machine Learning*. Vol. 2. 3. MIT press Cambridge, MA.