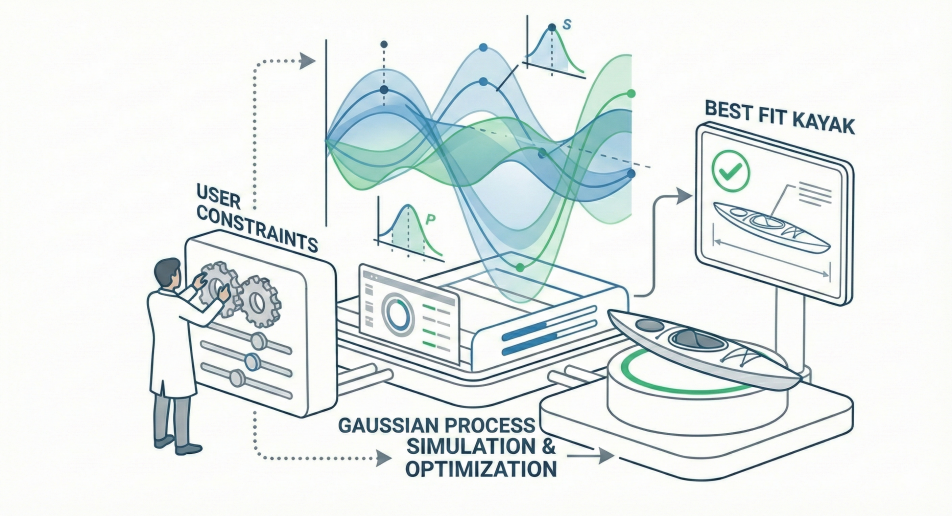
Kayak Hull Optimisation

Max Carroll (mc2372), Muhammad Ayain Fida Rana (mafr2), Kilian Schulz (ks2218)
University of Cambridge

Abstract

In this report we lay out a tool for use in whitewater kayak design. We use a combination of Gaussian Processes to model kayak hull stability and buoyancy over static fluid simulations, performing Bayesian Optimisation to find an optimal kayak hull design for *user-defined* priorities via an interface. To do this we give a way to parametrise and constrain the design space of kayak hulls which we optimise over. Further, this tool can be extended to other types of boats with differing hull parametrisation frameworks, opening the door to exploration of static stability and buoyancy for general boat hull design.



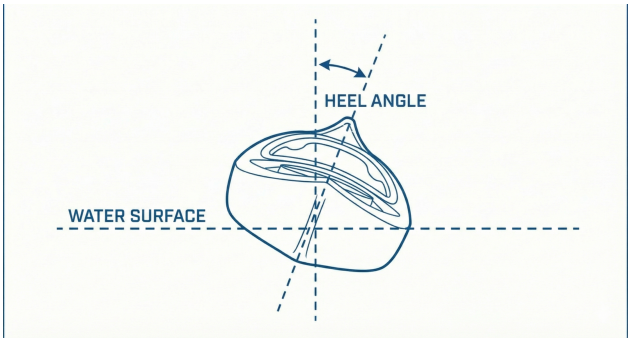
1 Introduction

Due to the lack of financial interest in the field, and the secretive arms race between the different manufacturers, there is an immense scarcity of published research in the field of kayak design. This becomes even more obvious when looking specifically at whitewater kayaks. While classic boat design techniques should in theory apply also to kayak design, they have not yet been formalised, and a search on google scholar for the term “whitewater kayak design” returns zero results. This reports aims at laying a foundation for the field, both addressing the lack of data, approaches to buoyancy and stability simulations, as well as modeling and optimising based on those results in a configurable manner. For this, we have introduced a program to parameterise the concept of a whitewater kayak, simulated different kayaks in a fluid simulation, and then fit and optimise a Gaussian Process based on that data. The code and simulation framework are publicly available at

2 Motivation & Background

The forces acting upon a kayaker are relentless; currents, standing waves, and hydraulic features are major challenges to the balance of even the most experienced kayakers. Other than the skill of the kayaker, an integral factor here is the

Figure 1: Hull tilted at a heel angle relative to the water.



correct design of the boat, in order to stay on the water. As such, our kayak design model will consider stability and buoyancy. Steerability is left to future work (Section 6.2.3).

However, stability and buoyancy are multi-faceted, not just a single number. We might care about different elements of stability in differing situations. For example, we could optimise to allow the hull to tip as far as possible yet still return to an upright without capsizing, or we could maximise the energy required to tip the boat until it naturally capsizes. Our system accounts for **7** different stability/buoyancy metrics, whose relative weighting is configurable by the user; we call these the **user priorities**.

2.1 Static Stability

To simplify, we consider only *static stability*, in particular the righting moment curves. This involves considering a kayak hull under a steady state fluid situation and measuring how the righting moment changes (angular force which naturally attempts to rotate the hull back to an upright or capsized position). This righting moment is the *instantaneous response* measured when a hull is tilted to a specific heel angle, as illustrated in Figure 1.

On the other hand, *dynamical stability* considers how the hull evolves over time. For example, how the hull oscillates over time after being disturbed (such as by waves). While this is important to design for in whitewater kayaking, it is a fundamentally different problems requiring a drastically differing framework.

2.1.1 Metrics

Consider a static stability curve f conceptually demonstrated in Figure **??**. We may measure various useful elements of stability from the curve to avoid having to directly compare curves. Of which, we choose to consider:

Final Stability (root_f): When the righting moment becomes negative (first root), the hull naturally rotates in the same direction as the tilt, causing the hull to capsize. Hence, this also known as tipping/capsizing point.

Diminishing Stability ($\max(f)$): Initially, the further you tilt the hull the harder it will try to push you back. However, after the point of diminishing stability (maximum) the righting moment will decrease. Intuitively, a constant tilting force less than the Diminishing Stability will not capsize the boat.[1]

Overall Stability ($\int_0^{\mathrm{root}_f} f$): The energy required to capsize the boat.

Righting Energy ($\int_{\mathrm{root}_f}^{\pi} f$): The energy required to un-capsize the boat. (This metric should be minimised.)

Initial Stability ($f'(0)$): The 'stiffness' of the hull. That is, how quickly the boat responds to the tilting, effectively how aggresively the hull tries to self-right for small heel angles.

2.2 Reserve Buoyancy

Reserve buoyancy is the extra weight that can be added to the hull before it sinks. *It is important to note that this is not constant over heel angles* for non-watertight hulls (e.g. unsealed kayaks), as a tilted hull will flood at a different draught (waterline). Hence, also represented as a function over heel angles, typically making an 'n-shape' or upside-down 'v-shape' (see Figure 4), from which we can extract 2 metrics:

[1] When starting from the inital (flat) state.

Initial Buoyancy: Reserve buoyancy of the hull when flat. Typically also the maximum.

Average Buoyancy: Average reserve buoyancy of the hull over all heel angles.

3 Model Framework

3.1 Interaction Flow

The interaction flow of our tools consists of the following steps, of which some only need to be performed once (e.g. data generation, initial training):

1. Generate simulation data from hulls described by our built-in hull parameterisation framework. Typically these satisfying a configurable set of constraints allowing only *realistic* hulls to be considered.
2. Train the model on the data.
3. Provide a (possibly differing)[2] set of constraints to optimise the hull over.
4. Specify user priorities weighting each of the 7 aforementioned stability/buoyancy metrics.
5. Returns hull parameters which optimises (maximises) a weighted score incorporating the user priorities. (This optimal hull can also be viewed directly.)

3.2 Components

The model consists of multiple distinct components and Gaussian Processes working together, briefly introduced here and explained further in latter sections.

The Simulator: Simulates hull meshes to produce righting moment and reserve buoyancy for a given heel angle.

Hull Parameterisation: Describes and generates hull meshes from arrays of parameters / features. Hence, a hull mesh can be generated from parameters and then simulated by the simulator.

Hull Constraints: Constraints on the hull parameters. These are essential for constraining the optimiser for both efficiency and practical reasons. For example, the knowledge that a hull should be large enough to fit a human inside and must be longer than it is wide to avoid severely inhibiting steerability[3].

User Priorities: The weightings. and in particular constraint-determined[4] normalisation factors on how to balance each metric (whose un-normalised values have differing units and scales).

The Aggregator: Two Gaussian Processes to predict righting moments and buoyancies over hull parameters and heel angles. For each constant set of hull parameters, the 7 metrics are extracted by performing Bayesian Optimisation on the corresponding heel-righting moments or heel-buoyancy curves involving a *composite* of 7 acquisition functions. The 7 optimised metrics are aggregated according to the User Priorities.

The Optimiser: Treating the Aggregator itself as a simulation (from hull parameters to scores). A third Gaussian Process optimises the hull parameters to maximise the score.

4 System Description

This section describes the parametric hull representation, mesh generation, and simulation framework, and aggregator used to evaluate kayak hull designs.

4.1 Parametric Hull Representation

When we sought to optimize hull designs, a key challenge was choosing a geometric representation that was sufficient to explore variations in design space while remaining low-dimensional and maintaining physical realism.

Therefore, we adopt a parametric hull representation that encodes prior knowledge about realistic kayak geometries, allowing us to systematically explore key design degrees of freedom. The parameterisation is organized into five main categories: material properties, global dimensions, cross-sectional shape, rocker profile, and cockpit geometry, as defined in Table 1.

[2] Note: currently unsupported in the UI, but possible in the Python interpreter.

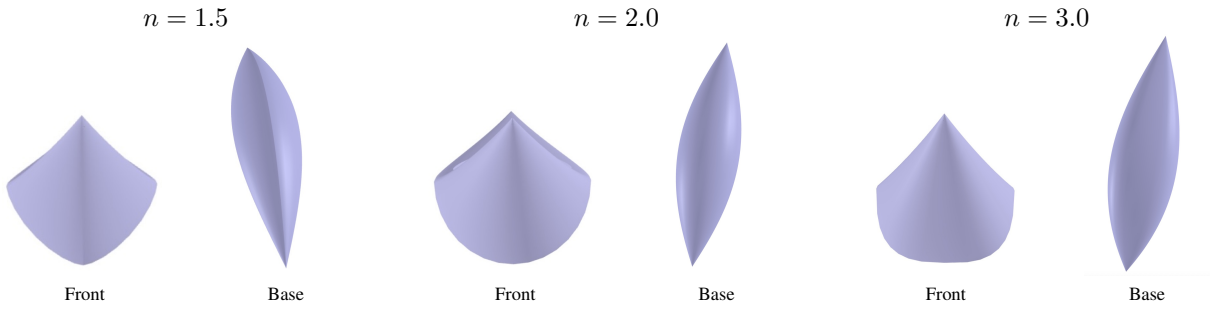
[3] Which, of course, is not yet considered by our model.

[4] See extended description.

Table 1: Hull parameter definitions

Category	Parameter	Symbol	Description
Material	Hull material density	ρ_{hull}	density of hull material
	Hull thickness	t	thickness of the hull shell
Global	Hull length	L	Overall length of the hull from bow to stern
	Maximum beam	B	Maximum width of the hull cross-section
	Hull depth	D	Maximum vertical depth of the hull
Cross-section	Super-ellipse exponent	n	Controls roundness of the cross-sectional shape
	Beam position	p_B	Longitudinal position[5] of maximum beam
Rocker	Bow rocker	r_b	Vertical curvature of the hull centreline near the bow
	Stern rocker	r_s	Vertical curvature of the hull centreline near the stern
	Rocker position	p_r	Longitudinal Position of minimum rocker
	Rocker exponent	n_r	Controls how curvature is concentrated toward hull ends
Cockpit	Cockpit opening	–	Boolean flag indicating presence of cockpit opening
	Cockpit length	L_c	Length of the cockpit opening
	Cockpit width	W_c	Width of the cockpit opening
	Cockpit position	p_c	Longitudinal position of cockpit center

Figure 2: Lower values of n produce sharper, more V-shaped profiles with reduced wetted surface area, while higher values yield fuller, flatter-bottomed sections with greater initial stability.



This parameterisation enables us to balance expressiveness with physical constraints when defining hull geometry. For example, the rocker profile controls the curvature in the bottom of the hull along its length and captures the trade-off between tracking and maneuverability. Similarly, the cross-sectional shape parameters model the bottom design, which in turn influences speed, stability, and turning.

4.2 Mesh Generation

We then transformed the parametric representation into a watertight triangular mesh that serves as the geometric representation of a hull used by the simulation framework. The generation begins with discretising the hull into stations uniformly distributed from bow to stern. At each of these points, we then compute the local width and depth using a sinusoidal taper function, which in turn produces natural kayak lines that avoid discontinuities in curvature.

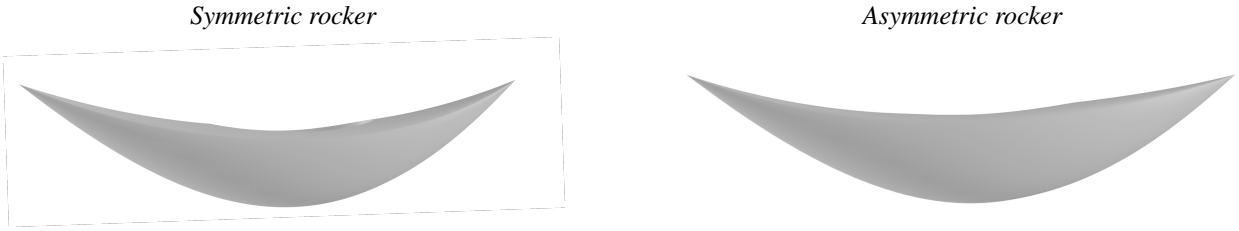
Cross-sectional geometry. Hull cross-sections are defined using a super-ellipse parameterisation,

$$\left|\frac{y}{B/2}\right|^n + \left|\frac{z}{D}\right|^n = 1, \tag{1}$$

where the exponent n controls cross-sectional shape. Varying n allows us to capture a wide range of realistic hull profiles, from sharp to flat-bottomed shapes (Figure 2).

[5] measured along hull length (0.0=bow, 1.0=stern).

Figure 3: Side-view hull profiles illustrating symmetric and asymmetric rocker configurations.



Rocker profile. The longitudinal curvature (rocker) of the hull is parameterized independently for the bow and stern as a smooth vertical displacement along the hull length according to:

$$z_{\text{rocker}}(x) = \begin{cases} r_b \left(1 - \frac{x}{p_r L}\right)^{n_r}, & x \le p_r L, \\ r_s \left(\frac{x - p_r L}{(1-p_r)L}\right)^{n_r}, & x > p_r L, \end{cases} \tag{2}$$

where the exponent n_r controls how gradually the rocker rises toward the hull ends, with higher values producing a more pronounced upturn near the tips. Separate rocker parameters for the bow and stern allow these regions to be lifted independently, as illustrated in Figure 3.

Hollow hull construction. A key design choice in our representation is to model the kayak as a hollow structure rather than a solid object in order to enable realistic estimation of hull mass, centre of gravity, and buoyancy. We do so by generating two meshes: an outer hull and an inner hull offset inward by the thickness parameter t. The final mesh is constructed via boolean difference:

$$\mathcal{M}_{\text{hull}} = \mathcal{M}_{\text{outer}} \setminus \mathcal{M}_{\text{inner}} \tag{3}$$

resulting in a watertight hollow structure with mass derived from surface area, thickness, and hull material density, ready for subsequent fluid simulation. Cockpit opening, if indicated by the bool, is introduced by subtracting a parameterized volume from the top deck of the hull's mesh.

Mesh resolution. Mesh resolution is treated as a fixed fidelity parameter controlling the trade-off between accuracy and computational cost and kept separate from hull parameters.

4.3 Simulation Framework

While there exist tools to analytically calculate righting moment curves from meshes, we do not know of any which simulate non-watertight hulls. That is, hulls which upon being tilted to a certain angle will have segments flooded by water. As such, we had the rather complex task to create such a simulator from scratch, solved by making use of mesh boolean differences and a combination of bisection and brute-force optimisation techniques, though the details are not relevant to this report. Equally, tools that calculate buoyancy for non-watertight hulls, so we also use the simulator for this.

We note that this simulation does *not* account for fluid flow, but that our framework is sufficiently generic to instead use a steady state or even (some) dynamical fluid simulation as a simple drop-in replacement for the current 'analytic' simulator (see 6.2.3).

A typical curve is shown in figure 4 with 4 corresponding graphics at at differing points in **??**.

However, what is relevant is the general shape of these curves. The key domain knowledge being that:

- As the x-axis describes rotations: Both curves are *exactly* periodic over 2π.
- As the hulls are symmetric across the length axis: Buoyancy is symmetric and Righting Moments are anti-symmetric.
- When the hull floods: There will be discontinuities in both curves, with the righting moment curve potentially have a jump at this continuity.
- Reverse buoyancy is constant while the hull is flooded.

This suggests that this (heel) dimension of the (Aggregator) GPs should be described by some combination of a periodic kernel and a matérn kernel (to account for the jumps at discontinuities over righting moments), encoded with (anti-)symmetry. Further to the discontinuity arguments, the buoyancy curve will also likely do better with a matérn element (vs RBF) when learning the constant buoyancy while flooded.

Figure 4: Typical Righting Moment and Reserve Buoyancy Curves

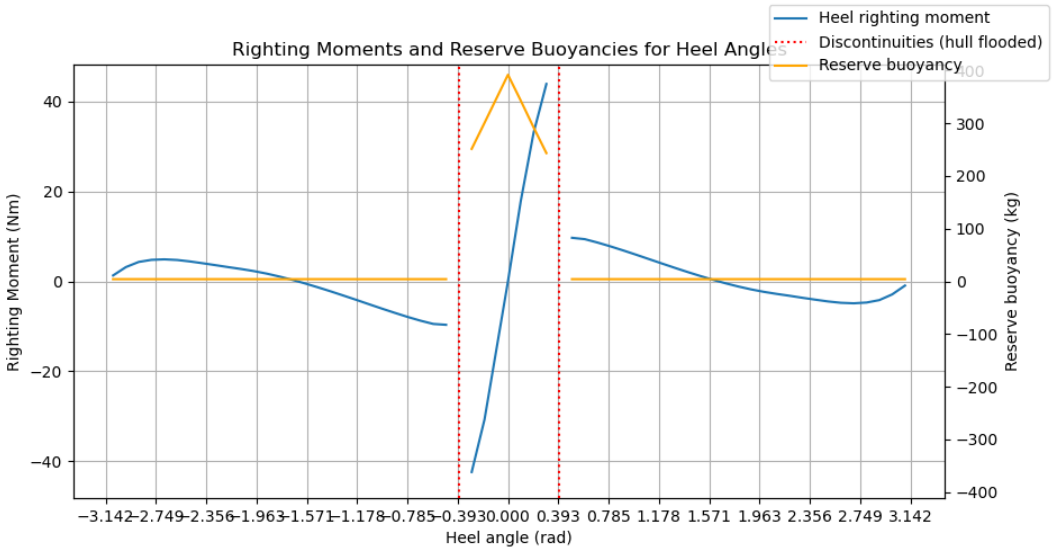
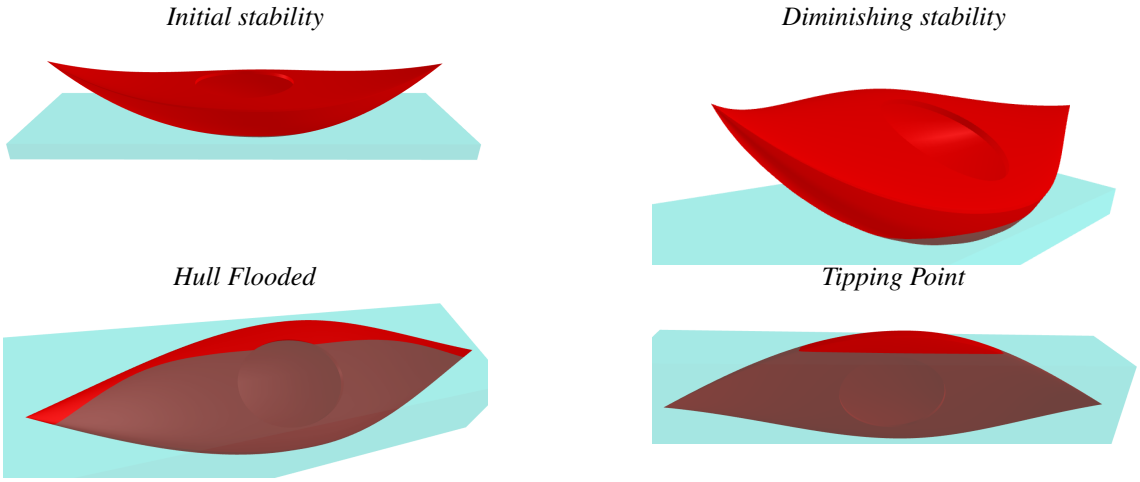


Figure 5: Simulation samples.



4.4 The Aggregator

The aggregator aims to score a hull (described by its parameters) over all possible heel angles according to the 7 weighted user priorities. To do so we use two GPs, which return righting moments and buoyancies respectively for hulls at (any) heel angle. These are trained on data from the previous simulator using matérn kernels over the hull parameters and a domain-suggested periodic kernel (product of matérn and periodic, encoded with (anti-)symmetry) over heel angles.

Using these GPs we perform Bayesian Optimisation to extract the seven user priority metrics from the righting moments / buoyancy curve. But, how do we choose where to sample to improve the accuracy of our output score? We solve this problem with seven separate acquisition functions, one for each metric, then select which to use based on the user weights and sample further data. This means that the aggregator will only explore the parts of the curve *relevant* to the user's priorities.

Our stopping criterion gives computation budgets[6] to each metric according to the user priorities, and avoids sampling from the acquisition function for a metric that is out-of-budget. Further, initial buoyancy requires only one sample (buoyancy at 0), so is stopped immediately after.

[6]Simulation cost specified by the iterations performed in the numerical solvers within the simulator.

The following (un-normalised) acquisition functions were used:

Tipping Point: Simply use the probability density that the GP predicts 0, *but* only consider the region after the point of diminishing stability (estimated from the mean curve).

Diminishing Stability: Expected Improvement (for maximisation).

Overall Stability: GP Variance if in the range 0 to the estimated tipping point (from the mean curve).

Righting Energy: GP Variance if from the estimated tipping point until π.

Initial Stability: Sample as close to zero as possible in the grid size being used.

Initial Buoyancy: Sample at 0.

Overall Buoyancy: GP Variance.

Derived from the following intuition: integrals can be made more accurate by decreasing the variance in our curves. Expected improvement is well known to accurately find maxima. Roots have a direct probabilistic interpretation, which we augment to direct *away* from the origin.

4.4.1 Metric Normalisation

We may then estimate each of the 7 metrics independently from the resulting mean curve in the GP. However, we wish to combine this into a single score. We can account for user priorities easily by their weights. The bigger difficulty is that each metric is in different units (energy, radians, Nm, kg, etc.) and magnitudes (overall stability » tipping point), so we would like to normalise each metric based on some sort of reference value, to enforce some degree of fairness. We allow this to be set manually, but note an appealing methodology for creating informed normalisation factors.

Suppose we are optimising a hull under a set of constraints, but don't yet know how to normalise. For each metric, we could run the simulation with a user priority of 1 for the corresponding metric and 0 for the others. This returns an optimal hull and un-normalised score maximising metric with maximum freedom, e.g. the most extreme tipping point possible if we didn't also have to factor in buoyancy, overall stability etc. This un-normalised score can be used as a normalisation constant. Then each normalised metric will have a meaning (in percent), representing how close to the maximum (under free priorities).

4.5 Score Optimiser

We make use of a generic GP-based optimiser to optimise hull parameters using the aggregator as an objective function. The configuration of this optimiser is automated; exploiting domain knowledge on how hull parameters (plus user priorities) affect scores is left to future work (i.e. increase beam will generally increase the tipping point metric).

5 Kernel Comparisons

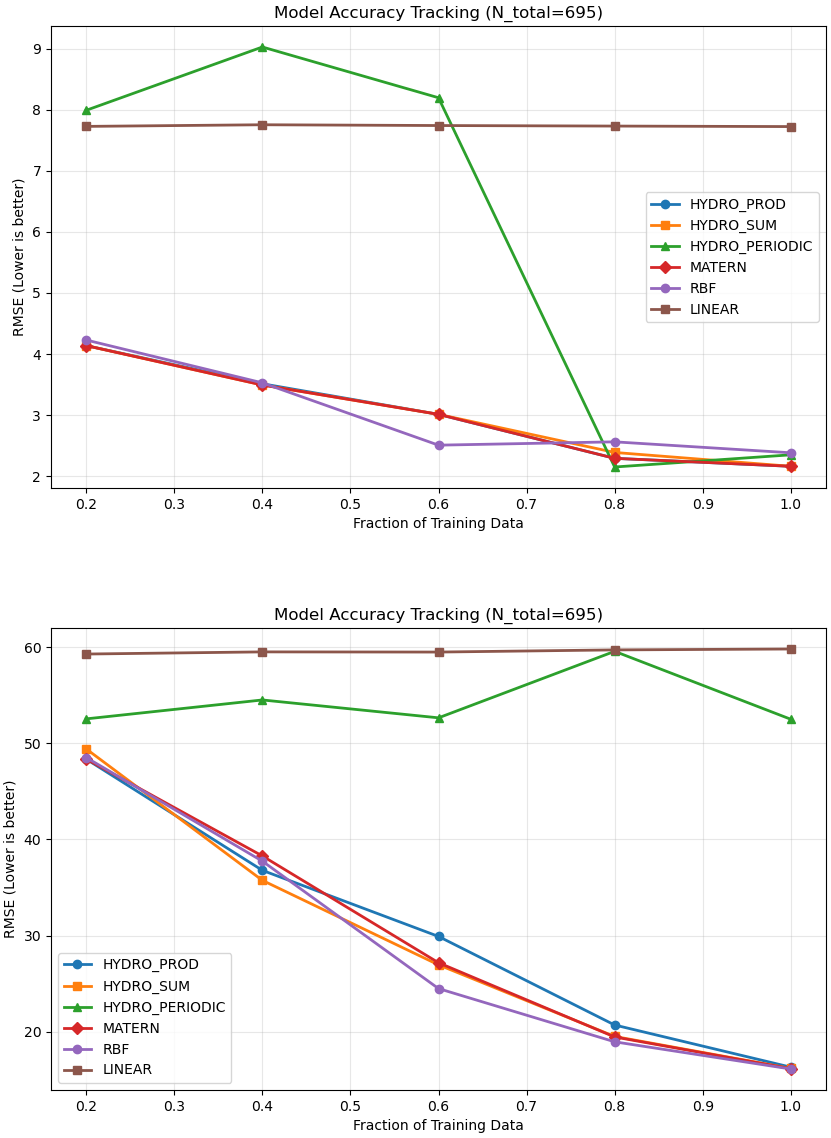
We explore our hypothesis that using a combined periodic matérn kernel with symmetry should more accurately represent our function space along the heel angles dimension. We do this by training the model on larger and larger random segments of the data for a variety of kernel options and tracking the root mean square error (RMSE). We consider (on the heel dimension only, with hull parameter dimensions all defined by matérn52 kernels, except for in the RBF & LINEAR kernel case):

- `HYDRO_PROD`: A product of a periodic and matérn52 kernel.
- `HYDRO_SUM`: A sum of a periodic and matérn52 kernel.
- `HYDRO_PERIOD`: A pure periodic kernel. (heel dimension only)
- `MATERN`: matérn52 kernel.
- `RBF`: RBF kernel. (including for hull params)
- `LINEAR`: Linear kernel. (including for hull params). Intended to serve as a baseline.

As a sanity check the linear kernel performs very badly. While the periodic kernel and RBF kernels are worse than the matérn kernels, likely due to the discontinuities. Of particular note is the periodic kernel in the buoyancy curve where it is likely struggling to learn the fact that the buoyancy is constant (flat) while the hull is flooded.

Unexpectedly however, there is little difference between the three kernels which have a matérn component, suggesting that the matérn kernel trains to become the dominant feature in both the `HYDRO_PROD` and `HYDRO_SUM` cases. Equally,

Figure 6: Kernel comparisons for righting moment (upper) vs reserve buoyancy (lower)



it could be that the dominant factor in the RMSE for these cases is not from the heel angle dimension, and instead from the hull parameters (in which all 3 options use a pure matérn kernel). Future work could train some restricted GPs over constant hull parameters to isolate the RMSE.

6 Discussion

6.1 Optimal Hull Design

While there probably isn't enough nuance to our project for a whitewater kayak company to take our program and simply start generating their next line of kayaks, we did show an effective and useful way to generate kayak hulls according to user needs. The techniques we employed, are similar to other companies and their boat design [Keys et al., 2025]. We found unique ways to cope with our lack of data, given the very high dimensional space in which we were

running our optimisation. Using our framework, and substituting whichever part one cares about most, can give future kayak designers a well defined structure of the process.

6.2 Further Work & Applications

6.2.1 Simulation

The simulation is purely analytic, but could easily be extended to performing fluid simulations, adding fluid flow direction and speed as a parameter. This would additionally allow the modelling of hydrodynamicity (drag), enriching the trade-offs between hull parameters.[7] You would need to run the fluid simulation to a steady state, either by fixing the position of the hull or a dynamic simulation applying correcting forces to hold it at a given heel angle, to measure the righting moment (more accurate, but more unstable).

6.2.2 Parametrisation

Our parametrisation is intended to be interchangeable, any system that can generate meshes from parameters can be substituted, with the (generic) optimisation process still being valid. There is some obvious future work here such as more parameters that control the kayak hull explicitly.

Of particular note, you could consider a real kayak design and parameterise just a small portion of it, for example a parameter could describe just a V-shape ridge at the bottom of the hull. Then the user could optimise something the width of the ridge keeping the rest of the hull constant.[8] This approach is likely of more practical use in industry, reframing the problem as 'testing and optimising specific design ideas', without needing to build physical kayaks.

If one were to take it one step further, there is also the possibility of having kayak hulls and training a Variational Autoencoder on it in order to find an minimal set of parameters that best models the boat shape and then optimize those in the process.

6.2.3 Kernels

For our particular hull parametrisation we are guaranteed to have at most two discontinuity.[9] It might be possible to encode this directly by having two smooth periodic/RBF kernels applied over the range before and after the discontinuity (but unrelated across the gap)[10], and parameterise or explicitly search for the discontinuity (flooding) location.

Further both buoyancy and righting moment have discontinuities in exactly the same location, but we treat them as separate GPs, meaning this relation cannot be learned.

Finally, we have exact periodicity, but the combined kernels (`HYDRO_PROD`, `HYDRO_SUM`) don't actually enforce this (they just have a periodic component). Using a pure periodic kernel with a change-point as suggested above would provide for this.[11]

Predicting Normalisation Constants: Calculating each normalisation constant requires a full optimisation loop for each set of contraints. We could try and predict this by building a gp on constraints to normalisation constants.

Steerability: Our system does not account for steerability in any capacity, meaning the results are likely unsuitable for whitewater kayaking, despite their optimised stability and buoyancy.

6.3 Observations

Testing on different optimisation objectives seems to show the model working properly and realistically. Optimising only for tipping point for example outputs a relatively wide hull, with a tipping point at a 91° degree heel angle. The optimisation process respects our constraints and doesn't exceed them. Most created hull shapes look feasible and stable. Even some constraints that we did not explicitly specify were learned by the model: For example, even though our parametrization allows for cockpits bigger than the beam (width of the kayak),[12] the gaussian process quickly learned this to not be a stable shape and has not been the output of any of our optimisation runs yet. This shows, that within the constraints of our limited complexity in simulation, parametrisation and opmitimisation, our program performs well and has learned the basics of kayak design.

[7] i.e. wide boats generally improve both stability and buoyancy, but *worsen* hydrodynamicity.

[8] Of course, this requires retraining in general.

[9] Technically four, for the symmetric ones. The two unique ones are for 'flooding' and 'unflooding' (rare).

[10] Or a more smooth relation rather than a strict change-point, to account for inaccuracies.

[11] Note: the change point would also need to be replicated over each period.

[12] Causing the hull to have holes!

References

Dustin Keys, Khang Minh Phan, and Hamid Sadat. Conditional gaussian processes for wave design in ship hydrodynamics considering multiple events. *Journal of Marine Science and Technology*, dec 2025. ISSN 1437-8213. doi: 10.1007/s00773-025-01103-w. URL https://doi.org/10.1007/s00773-025-01103-w.