AMSI VACATION RESEARCH SCHOLARSHIPS 2022–23

SET YOUR SIGHTS ON RESEARCH THIS SUMMER

Optimal Control in Stochastic Hydrodynamic Models

Michael Nefiodovas
Supervised by Dr Thomas Stemler
University of Western Australia

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

1 Abstract

This project addresses the problem of predicting and recommending the path of a rower on the water based on wind and current data. The approach uses a force balance equation and graph algorithms to model the movement of the rower, and a Bayesian model to estimate the distribution of errors. The proposed model demonstrates promising potential in predicting the path of a rower and offering suggestions for improving performance, as suggested by the model's formulation and validation procedures. The project's significance lies in its contribution to the field of sports science and engineering, providing a novel and effective method for predicting and recommending rowing paths based on environmental data.

2 Introduction

2.1 Background information

Ocean rowing is a highly demanding sport that involves rowing across vast stretches of open ocean for weeks or even months. The success of an ocean rowing journey heavily depends on the management of supplies, energy levels, and the unpredictable ocean weather conditions [4, 9, 3]. The effect of winds and currents in the ocean significantly impact the success of an ocean rowing journey, and taking advantage of them can improve the chances of success because the rower is able to exert less energy on their journey. There are many algorithms for using wind and current data to find shortest paths on the ocean, however most of these algorithms are designed for large vessels such as cargo ships. We present a pathfinding algorithm specifically designed for the use-case of ocean rowing. Such an algorithm could help ocean rowers find efficient paths through the ocean, ultimately decreasing the time it takes for them to get to their target.

Ocean rowing is a popular sport however there are few computational tools used by these rowers to help them plan their routes. Typically, heuristic based methods are applied (e.g. rowing generally towards the target day-by-day). The disadvantage of this approach is that they don't take full advantage of the ocean currents and wind in their journey.

While there exists some pathfinding algorithms for larger ships [10], these algorithms can't be readily applied to small boats like those used in ocean rowing because smaller vessels require precise and adaptive routing to ensure safety and success. Additionally, large vessels are typically confined to shipping lanes which are preset known safe routes whereas ocean rowing is done in the open ocean with fewer safeguards. Therefore, there is a clear need for an algorithm specifically designed to help ocean rowers take advantage of the winds and currents when they traverse the ocean.

This research aims to address this need by developing a pathfinding algorithm that can recommend optimal routes for ocean rowers based on real-time oceanographic data. The algorithm will be based on a time-dependent shortest path model that discretizes the ocean domain, allowing for more precise and adaptive routing. In addition, we will develop a Bayesian model to estimate the potential inaccuracies and uncertainties in the pathfinding model's recommendations during deployment. By doing so, we aim to provide a valuable tool for

ocean rowers that can help increase safety, improve efficiency, and promote the growth of ocean rowing as a sport.

Figure 1: Ocean row boat. Notice the small size relative to other boats. Photographer unknown [11]

2.2 Problem statement

Although ocean rowing has become increasingly popular as a sport, few tools currently exist to aid ocean rowers in planning their routes and capitalizing on favorable winds and currents. While pathfinding algorithms have been developed for larger ships, they do not always translate to smaller boats, like those used in ocean rowing. This is due to the need for more precise and adaptive routing to ensure the safety and success of ocean rowers. Additionally, existing algorithms may not consider the potential inaccuracies and uncertainties in oceanographic data used to generate route recommendations. Hence, there is an evident requirement for a pathfinding algorithm tailored to help ocean rowers optimize their routes by utilizing the winds and currents in the ocean, while also taking into account the uncertainties in the data.

The aim of this research is to fill this gap by developing a pathfinding algorithm that can suggest the most efficient routes for ocean rowers based on real-time oceanographic data. The algorithm will rely on a time-dependent shortest path model that discretizes the ocean domain, providing a more accurate and adaptive routing system. Moreover, we will create a Bayesian model to assess the potential inaccuracies and uncertainties in the pathfinding model's recommendations during deployment. Our goal is to provide ocean rowers with a valuable tool that increases safety, improves efficiency, and advances the growth of ocean rowing as a sport.

2.3 Objectives

The main objectives of this research are:

- To develop a pathfinding algorithm that can optimize ocean rowing routes based on oceanographic data.

- To develop a Bayesian model to estimate the potential inaccuracies and uncertainties in the pathfinding model's recommendations during deployment.
- To provide a valuable tool for ocean rowers that can help increase safety, improve efficiency, and promote the growth of ocean rowing as a sport.

By achieving these objectives, this research can contribute to the development of better tools and resources for ocean rowers, ultimately improving the safety and success of ocean rowing expeditions while promoting the growth of the sport.

2.4 Methodology

To achieve the objectives of this research, we will employ a mixed-methods approach that combines quantitative data analysis with qualitative feedback from experienced ocean rowers.

We will begin by collecting oceanographic data from reliable sources, to use as input for our pathfinding algorithm. We will then discretize the ocean domain and convert the problem to a time-dependent shortest path problem [8]. This will allow us to develop a rudimentary model of ocean path planning that can provide recommendations for optimal routes based on the oceanographic data.

In addition to the pathfinding algorithm, we will also develop a Bayesian model to estimate the potential inaccuracies and uncertainties in the model's recommendations during deployment. This will involve collecting data on the accuracy and uncertainty of oceanographic data used as input for the algorithm, as well as feedback from ocean rowers who have used the algorithm during real-world expeditions.

2.5 Contributions

The main contributions of this research are:

- The development of a pathfinding algorithm that can optimize ocean rowing routes based on real-time oceanographic data while taking into account the unique challenges of this sport. This algorithm represents a significant improvement over existing pathfinding algorithms for ocean rowing, as it accounts for factors such as changing oceanographic conditions, currents, and winds.
- The development of a Bayesian model to estimate the potential inaccuracies and uncertainties in the pathfinding model's recommendations during deployment. This model provides a valuable tool for ocean rowers, as it allows them to better understand and anticipate potential inaccuracies in the algorithm's recommendations and make informed decisions accordingly.

3 Statement of Authorship

- PhD student Rick de Kreij developed the system of equations to be solved for the force-balance which is used to determine the rower's direction, this is mentioned at the end of subsection 4.1. He also provided

code which included how to extract and manipulate the dataset.

- Ivica Janekovic provided example code for another project using a Time Dependent Dijkstra which was used as a reference.
- Professor Phil Watson oversaw and supervised the project and provided guidance on practical requirements.
- Dr Thomas Stemler supervised the project and proofread the report.

4 Methodology

4.1 Discretisation process

Available wind and current hindcast datasets typically provide a sparse lattice of estimated historical measurements of wind and current values. The first step of the algorithm will require a refinement of these measurements which will either increase the distance between lattice measurement points or decrease it. Suppose we specify these new distances between vertex points with parameters Δ_x, Δ_y.

To perform this refinement process, we first convert the original lattice to a continuum, $F : \mathsf{R}^2 \quad \mathsf{R}^+ \; ! \; \mathsf{R}^n$, by a bilinear interpolation process. To estimate the value at an arbitrary position (a, b), we consider the four lattice vertices which bound the point (a, b), call these $\mathbf{x}_i = (x_i, y_i)$ for $i = 1, 2, 3, 4$ and call them "top left", "top right", "bottom left" and "bottom right" respectively. We can ignore the edge case on the boundary where there won't be four bounding points as we just shrink the domain so that there is always 4. We then calculate $F((a, b), t)$ using the following formula:

$$F((a,b),t) = \frac{\sum_{i=1}^{4} F(\mathbf{x}_i, t) \mathit{j} x_i \quad a \mathit{jj} y_i \quad b \mathit{j}}{\sum_{i=1}^{4} \mathit{j} x_i \quad a \mathit{jjj} y_i \quad b \mathit{j}}$$

This simply weights the values at each vertex $\mathbf{x_i}$ by how close it is to the point (a, b).

The next step is to then select a discretisation of this vector field F. This is easy to do in principle, simply selecting values $\Delta_x, \Delta_y \quad 0$ and forming the lattice $L = \mathit{f}(0{+}k\Delta_x, 0{+}l\Delta_y) : k, l \; \mathit{2} \; \mathsf{N}_0 \mathit{g}$ results in a corresponding discretisation of F where we then associate a vector to each point in the obvious way: $\bar{F} : L \quad \mathsf{R}^+ \; ! \; \mathsf{R}$ mapping $\bar{F}(\mathbf{x}, t) = F(\mathbf{x}, t)$.

Our goal is now to convert this to a graph structure (V, E). It will be natural to consider the vertex set, V, of our graph as the set of locations L and it will also be convenient to index the entries as (l, k). To construct the edge set E we have a couple options: connect adjacent vertices with in their Von Neumann neighborhood (connecting only directly adjacent vertices) or connecting vertices with their Moore neighborhood (adjacent vertices and diagonals). As this is a model of ocean navigation, it is natural for a vessel to move diagonally and so the Moore neighborhood is selected here.

The next task is to calculate the edge weights. In the graph the edge weights will correspond to the time it takes the rower to travel from one vertex location to the other. We therefore need to estimate the time it

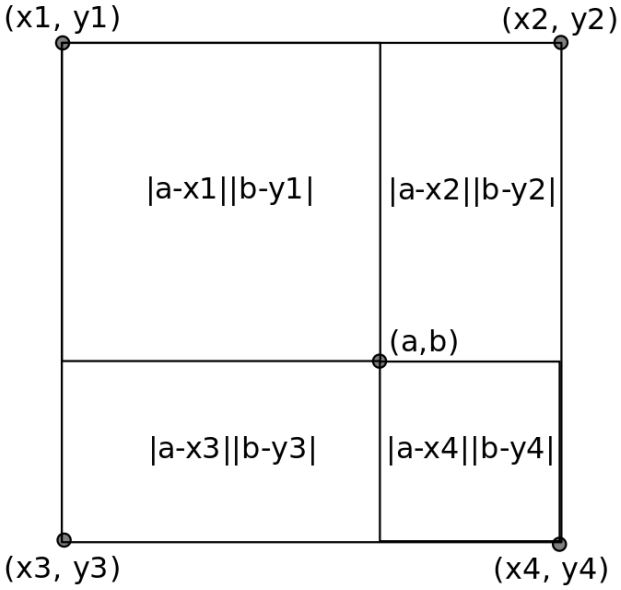

Figure 2: Representation of how the bilinear interpolation works. The value at the point (a,b) is equal to a weighted average of the function values at each of the corner points proportional to the size of the corresponding rectangle.

will take the rower to travel between two adjacent vertices. This time will depend on two factors: the pair of vertices which are being travelled between, and the time at which the vessel starts this traversal.

For sake of brevity we restrict our attention to the case of a single external force acting on the vessel during the traversal, for example the effect of wind. Suppose we have two vertex points $a, b \; 2 \; V$ with an edge from a to b: $(a, b) \; 2 \; E$.

Initial location	a
Destination	b
wind velocity at position a at time t	$F(a, t)$
Thrust force of vessel	$jF_r j$

Table 1: Known variables in the calculation.

Ideal thrust angle	θ
Resulting vessel velocity magnitude	$jv_b j$

Table 2: Unknown variables in the calculation.

If we assume that the vessel has zero acceleration we can solve the force balance equation:

$$\mathbf{0} = \mathbf{F_r}(\theta) + \mathbf{F_{a}}_{jj}(\theta, jv_b j)$$

where $\mathbf{F}_{ajj}$ is the wind friction force parallel to the vessel. It is calculated as

$$\mathbf{F}_{ajj} = \alpha_a(\mathbf{e}_{\mathbf{Fa}} \quad \mathbf{e}_\mathbf{r})j\mathbf{v}_\mathbf{a} \quad \mathbf{v}_\mathbf{b}j^2\mathbf{e}_\mathbf{r}$$

Where α_a is the air friction coefficient, $\mathbf{e}_{\mathbf{Fa}}$ is the direction of the friction force, $\mathbf{v}_\mathbf{a} \quad \mathbf{v}_\mathbf{b}$ is the relative velocity of the wind with the boat and $\mathbf{e}_\mathbf{r}$ is the direction the boat is rowing.

We solve the force balance equation numerically with in SciPy, an open-source Python library used for scientific computing and technical computing [12].

We can then calculate the time to travel based on the velocity magnitude, since we can divide the distance between a and b by the vessel speed jv_bj to calculate the time of travel between the vertices. This is the weight of the edge (a, b) at a given time t.

4.2 Pathfinding algorithm

We now have a graph (V, E) and time dependent weights $w : E \quad \mathsf{R}^+ \; ! \; \mathsf{R}^+$. Intuitively, the vertices are the locations the vessel can be at any time and the edge weights correspond to the amount of time it would take to travel between two locations taking into account the wind forecast at that location.

Our objective is to find the shortest path on this graph. This is an example of a time dependent shortest path problem (TDSP) [6].

The Dijkstra algorithm is a shortest path algorithm in a weighted graph with non-negative edge weights. We will apply this algorithm to the graph made in the discretisation process. In order to account for the time-varying nature of the ocean conditions, we modify the algorithm slightly.

The modified algorithm works as follows: Associate with each vertex i a label D_i which is the shortest path found to vertex i so far. Also associate another label q_i which is the predecessor to vertex i in the shortest known path to i from the source.

Algorithm 1 Time Dependent Dijkstra

```
D[s]  0
while jFj 6= jVj do
    v  arg min_{u∈F} D[u]
    F = F [ fvg
    for (v, y) 2 f(v, y) : (v, y) 2 E, a = vg do
        if D[v] + w((v, y), D[v]) < D[y] then
            D[y] = D[v] + w((v, y), D[v])
            q[y] = v
        end if
    end for
end while
```

The Dijkstra algorithm is a way to find the shortest path between two nodes in a graph. It achieves this

by assigning a "distance" value to each node, indicating the distance from the starting node to that node. At the beginning, the distance value for the starting node is set to zero, while the distance values for all the other nodes are set to infinity.

Now for the next step: the algorithm selects the node with the smallest distance value, which hasn't yet been visited. What happens next is called "relaxation," where the algorithm updates the distance values for all the neighboring nodes based on the weights of the edges connecting them. These weights can represent various factors such as distance, time, or any other relevant parameter. In our case, the edge weights represent time.

The algorithm then iterates through all the nodes, updating their distance values until the destination node is reached. And if that's not enough, it continues iterating until all the nodes have been visited.

The time-dependent Dijkstra algorithm [8] works by taking into account the time-dependent edge weights when computing the shortest path. Each edge is assigned a weight, which represents the time it takes to travel along that edge. The weight of an edge may vary depending on the time of day, the day of the week, or other factors. In our case, the edge weights would be the calculated travel time discussed earlier.

The algorithm then maintains a priority queue of vertices, where the priority of a vertex is given by the value $D[u]$. In the same way as Dijkstra's algorithm, the priority of the source vertex is set to 0, and the priorities of all other vertices are set to infinity. The algorithm then repeatedly selects the vertex with the lowest priority from the queue and relaxes all of its outgoing edges.

During relaxation, which is the step where the $D[y]$ entry is updated, the algorithm checks if the path to the neighbor vertex can be improved by going through the current vertex. If it can be improved, then the value $D[y]$ is updated.

The algorithm continues this process until all shortest paths have been found and all the vertices have been explored.

4.3 Bayesian model

All models are wrong [1]. In order to estimate the accuracy of the pathfinding model we've made, we identified two major sources of error: distance errors resulting from the rower overshooting the target, and angular errors resulting from the final destination not being in-line with the target. If our model was perfect, then this distance and angle should be zero after every stage of activity. To predict the distribution of these error parameters, we developed a Bayesian model. The goal of this model is to estimate the distribution of the errors so that the rower can make more informed decisions while on the journey. Practically speaking, this will allow the expectation management at different scenarios throughout the journey. By understanding the probability distribution of the errors we can know when alternative approaches need to be employed (e.g. deferral to professional meteorologists or navigators).

We model these error values as random variables:

$$\theta_i \quad \text{Von-Mises}(\mu_0, \kappa) \tag{1}$$

$$\epsilon_i \quad N(0, \sigma^2) \tag{2}$$

$$\mu_0 = \pi \tanh(\beta_0 \varphi_i) \tag{3}$$

Where θ 2 [π, π) is the angular error experienced in session i, ϵ_i 2 R is the relative distance error experienced on day i and φ_i 2 [π, π) is the angle of the wind velocity vector relative to the direction the rower is travelling. Equation (3) expresses that the mean of the von-mises distribution may depend on the direction of the wind on a given day, i.e. the rower may be pushed further off-course when there is more perpendicular component to the wind.

We also write the prior distributions on κ, σ^2 and β_0:

$$\kappa \quad \text{cauchy}(0, 200) \tag{4}$$

$$\sigma \quad \text{half-cauchy}(0, 100) \tag{5}$$

$$\beta_0 \quad N(0, 100) \tag{6}$$

These priors were chosen because they capture our understanding about the physical scale of the parameters in our application.

When using the model, we will receive a dataset D consisting of realisations of the values θ_i, ϵ_i and φ_i for $i = 1, \ldots, K$ where K is the number of steps so far.

Our objective is to infer the posterior probability density functions:$P(\kappa|D)$, $P(\sigma|D)$, $P(\beta_0|D)$. To do this we can use bayes rule for analytical results: $P(\kappa|D) = \frac{P(D|\kappa)P(\kappa)}{P(D)}$. The issue is in calculating $P(D)$ as you need to perform an intractable integral. Algorithms such as the No U-turn Sampler (NUTS) [7] allow us to empirically sample from $P(\kappa|D)$ without directly computing the denominator. The program Stan uses an implementation of NUTS to perform Bayesian analysis and was used here to implement this model.

4.4 Data sources

For our project, we used the Global Ocean Wind L4 Reprocessed 6 hourly Observations v6.0 dataset for the wind data [5]. This dataset provides information on wind components such as meridional and zonal wind, wind stress, and wind/stress curl and divergence. It also includes error estimates.

The dataset was originally made by combining wind data from a variety of different sources, such as satellites, to get a complete picture of the wind patterns on the ocean surface.

4.5 Software and tools

I used several tools for developing and implementing my project. The primary tool I used was Python, which I utilized for writing the algorithms. Additionally, I used Scipy [12] for solving the force balance equation and

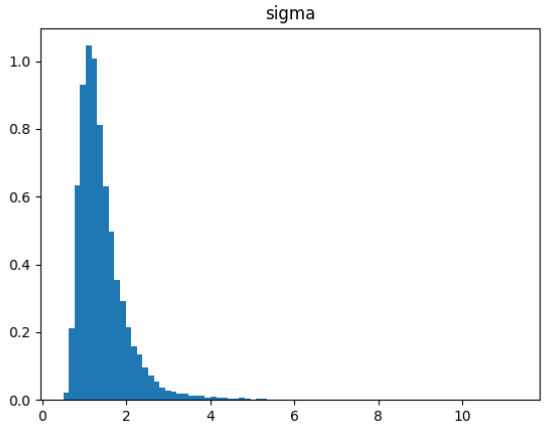

Figure 3: Posterior distribution of σ using synthetic data after running Stan.

Stan [2] for implementing the Bayesian model. These tools proved to be highly effective in facilitating the successful completion of my project.

4.6 Summary

To summarise, the project was split into three parts: firstly; reformulating the problem as a shortest path problem on a graph, secondly; recognising the problem is solvable using a time dependant Dijkstra algorithm and finally; developing the Bayesian error modelling system.

5 Discussion and Conclusion

In conclusion, we have developed a system for predicting and recommending the path of a rower on the water based on wind and current data. The model uses a force balance equation to calculate the velocity and travel time of the rower. This was achieved by discretising the domain into a graph and running a classical graph algorithm. We also developed a Bayesian model to estimate the distribution of errors in the model, taking into account both distance and angular errors.

Our work has significant implications for the field of rowing and water sports, providing coaches and athletes with a powerful tool for optimizing their performance. By integrating wind and current data, our system can help rowers make informed decisions about their routes on the water, avoiding unfavorable conditions and maximizing their speed and endurance.

In addition, our approach of using force balance equations and graph algorithms can be applied to other water sports and activities, such as sailing and kayaking. Since our model can be applied to many scenarios it makes it a valuable contribution to the wider field of sports science and engineering.

Furthermore, our use of Bayesian modeling to estimate the distribution of errors allows us to understand when our model is performing well and when it is performing poorly. This modeling technique can be applied

to other prediction models in a wide range of fields, allowing for more rational decision making.

Our team has successfully designed a system that predicts and recommends the optimal path for rowers based on wind and current data. The core of our model lies in a force balance equation that accurately calculates the rower's velocity and travel time. To achieve this, we discretised the domain into a graph and applied a classical graph algorithm. But that's not all, we also went one step further and developed a Bayesian model that estimates the distribution of errors in the model. Our model takes into account both distance and angular errors, ensuring accuracy and precision in its recommendations.

5.1 Limitations

Although our system provides a valuable tool for predicting and recommending rowing paths, there are several limitations and assumptions that should be noted.

In our model, we assume that rowers follow a straight line path between two vertices. This neglects many different effects such as variation in rowing speed, variation in wind velocity, etc.

Another assumption is that we assume in our Bayesian model that the error distributions are normally distributed. This is almost certainly false in general (due to the existence of major outlier events). Heavier-tailed distributions could be used here to better capture the true nature of errors in the model.

5.2 Future Directions

There are many directions we could take our work. The first is that we could add more physical considerations to the model. For example, we currently don't model the impact of wave height on the vessel's travel which is likely a factor. In that regard, it may also be useful for developing a principled system for identifying what factors should be included and which factors are acceptable to neglect. Importantly, you wouldn't want to spend excess time modelling features which only have a mild or negligible impact on the predictions.

It would also be interesting to setup a streaming system so that the model is ingesting real-time data from the vessel. This would allow the route to be updated in close to real-time. Due to the long time horizon on these rows, such a system is probably not extremely important as day-to-day decisions don't have a major impact on the outcome of the row.

More general analysis could be done on specifically how different environmental factors impact the time of the shortest path. For instance, looking at how variations in the wind speed across the ocean impact the length of the trip. It would also be interesting to investigate the impact of macroscopic factors such as differences between el niño and la-niña years on the shortest path.

In summary, there are many additional considerations which could be made to probe the model further or to incorporate more complex physical features (such as the impact of waves).

5.3 Conclusion

Our project has contributed to the field of sports science and engineering by providing a novel and effective method for predicting and recommending rowing paths based on environmental data. By combining physical principles with computational methods, we have developed a practical and useful tool that can improve the performance of rowers and water sports enthusiasts. Our system has important practical applications, as it can be used by coaches and athletes to plan their routes on the water and make informed decisions about when and where to row. Moreover, our approach of using a force balance equation and graph algorithms to model rowing on water can also be applied to other water sports and activities. Overall, our work represents a step forward in the field of sports science and engineering, and we believe that it has the potential to inspire further research in this area.

6 Acknowledgements

I would like to specifically acknowledge the help and support of Rick de Kreij in the development of the project, his help and support has been crucial to the project's continuation. I would also like to more generally acknowledge the support from members such as Professor Phil Watson at the UWA Oceans Graduate School of Research who helped develop the problem statement and provided supervision during the project.

References

[1] George E. P. Box. "Science and Statistics". In: *Journal of the American Statistical Association* 71.356 (1976), pp. 791–799. DOI: 10.1080/01621459.1976.10480949. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1976.10480949. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1976.10480949.

[2] Bob Carpenter et al. "Stan: A probabilistic programming language". In: *Journal of statistical software* 76.1 (2017).

[3] Translated by Content Engine LLC. "Rowing across the Atlantic to save it: The giant challenge of Ocean Cats". eng. In: *CE Noticias Financieras* (2022).

[4] "Doctors Adrift hit halfway mark in Indian Ocean rowing challenge". eng. In: *M2 Presswire* (2017).

[5] European Union-Copernicus Marine Service. *Global Ocean Hourly Reprocessed Sea Surface Wind and Stress from Scatterometer and Model*. en. 2019. DOI: 10.48670/MOI-00185. URL: https://resources.marine.copernicus.eu/product-detail/WIND_GLO_PHY_L4_MY_012_006/INFORMATION.

[6] Michel Gendreau, Gianpaolo Ghiani, and Emanuela Guerriero. "Time-dependent routing problems: A review". eng. In: *Computers operations research* 64 (2015), pp. 189–197. ISSN: 0305-0548.

[7] Matthew D. Hoffman and Andrew Gelman. *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*. 2011. DOI: 10.48550/ARXIV.1111.4246. URL: https://arxiv.org/abs/1111.4246.

[8] Wu Jigang et al. "Algorithm for Time-dependent Shortest Safe Path on Transportation Networks". In: *Procedia Computer Science* 4 (2011). Proceedings of the International Conference on Computational Science, ICCS 2011, pp. 958–966. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2011.04.101. URL: https://www.sciencedirect.com/science/article/pii/S1877050911001591.

[9] "Oxford Brookes University supports Royal Navy submariners' epic 3,000-mile ocean rowing challenge". eng. In: *M2 Presswire* (2023).

[10] Ülkü Öztürk, Melih Akdağ, and Tarık Ayabakan. "A review of path planning algorithms in maritime autonomous surface ships: Navigation safety perspective". eng. In: *Ocean engineering* 251 (2022), pp. 111010–. ISSN: 0029-8018.

[11] Unknown. *Woodvale 2 pair ocean rowing boat*. [Online; accessed Feb 25, 2023]. URL: https://images.squarespace-cdn.com/content/v1/5f4a7253d483dc72d272bb00/e06622d0-6d32-4faf-a437-8553452c93f0/Boat+-+Phoenix.jpg?format=2500w.

[12] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.