GO WITH THE FLOW: OPTIMAL SAILBOAT RACING

CONNOR MCBRIDE, ETHAN PALENSKE, JACKSON POND

Abstract. In this paper we will try to model the optimal path in the historical M&M 100 Miler sailing race. Many forces impact optimal sailing, including wind direction, current, and waves, making this a challenging problem to model accurately. The primary questions we work to answer are what the simplest model is that rediscovers real sailing techniques, such as tacking, and how many parameters we can add into our model while still being able to solve it and get reasonable results. We solve the classic Zermelo Navigation Problem assuming constant boat velocity and fixed current drift vector field. Further iterations involve relying on wind instead of a motor, examining square vs. triangular sails, and adding land penalties. Our more accurate models were able to find a path that matched the speeds of real-world sailors from the year we got current and wind data.

1. Background

The historic M&M 100 Miler sailing race takes place annually in Lake Michigan, and despite its name, is about 42 nautical miles in length [Mar24]. While this is a family-centered race with no prizes or larger impact, the problem of optimizing sailing routes is an ancient one, and one that is growing in importance today.

Triangular sails were first used in the 2nd century, and since then, determining optimal routes for boats has been a key question not just for racing, but also for commerce, avoiding pirates, and catching commercial vessels that are avoiding you. Now that cargo vessels are beginning to explore adding sails to make their travel more efficient, optimal sailing is once again a key issue in global supply chain [Pag26].

In this project we work to solve a related problem by modeling the optimal route for the 100 Miler race. We started from the simplest case and then increased the model complexity from there.

The primary question this paper seeks to determine is what the simplest model is that re-discovers real world sailing techniques, particularly tacking and jibing. Similarly, we wanted to determine the most physically realistic model that could still be solved numerically and yield realistic results. Our code is available on our GitHub repository [MPP26].

Date: April 18, 2026.

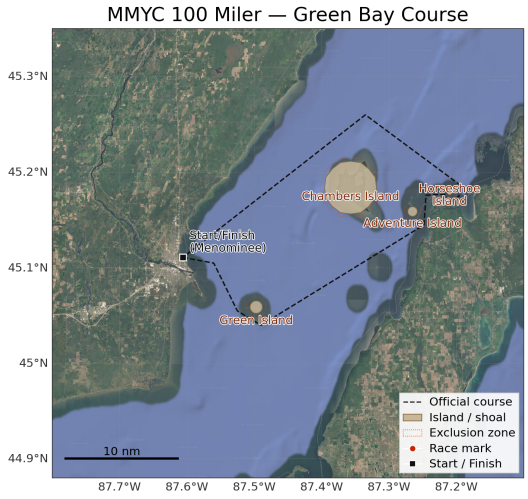

FIGURE 1. The M&M 100 Miler race course.

2. Mathematical Representations

In this section we go over the models we used to represent sailing dynamics.

2.1. **Model 1: Zermelo Navigation Problem.** We first solved the classical Zermelo Navigation Problem, which involves directing a motorboat moving at constant speed v through a body of water with a variable current, given by the vector field $\mathbf{w}$. The problem is to determine the path that minimizes the time it takes to get between two fixed points. In this case the control is the angle θ between the boat's heading and the x-axis. Our state $\mathbf{s}$ is given by

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

which are simply our x and y coordinates.

The control vector, or $\mathbf{u}$, is given by the x and y components of the fixed velocity v in the direction of θ, or

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = v \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}.$$

The velocity vector is the control vector plus the effects of the drift $\mathbf{w}$:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = v \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} + \begin{bmatrix} w_1(x,y) \\ w_2(x,y) \end{bmatrix}$$

This problem's constraints are fixed endpoints and a norm for our control $\mathbf{u}$:

$$\mathbf{s}(t_0) = \mathbf{s}_0, \quad \mathbf{s}(t_f) = \mathbf{s}_f, \quad \|\mathbf{u}\| = v.$$

Thus, the functional to be minimized is

$$\begin{aligned} J[u] &= \int_{t_0}^{t_f} 1\, dt \\ \text{subject to } \dot{\mathbf{s}} &= \mathbf{u}(t) + \mathbf{w}(\mathbf{s}(t)), \\ \mathbf{s}(t_0) &= \mathbf{s}_0, \\ \mathbf{s}(t_f) &= \mathbf{s}_f, \\ \text{and } \|\mathbf{u}\| &= v. \end{aligned}$$

2.2. **Model 2: Square-Sail Dynamics.** The next approach used simple sailing dynamics modeled on square-sailed boats. The assumption is that the boat moves in the same direction as the sail is facing, with speed proportional to the exposure of the sail to the wind. The dynamics are very similar to the previous problem, with the angle of the boat from the x-axis, θ, being our control, and ϕ being the angle of the wind in relation to the x-axis. The primary differences between this model and the previous one is that we no longer have the constraint of constant boat speed and the dynamics are slightly different:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = v \begin{bmatrix} \cos(\theta - \phi)\cos(\theta) \\ \cos(\theta - \phi)\sin(\theta) \end{bmatrix}.$$

Also, the constraint $\|\mathbf{u}\| = v$ becomes $\|\mathbf{u}\| = v|\cos(\theta - \phi)|$. Intuitively the $\cos(\theta - \phi)$ term is the dot product between the boat's heading and the wind's direction, dictating the exposure of the sail to the wind. The rest of the setup, including the functional, are identical to the previous one.

2.3. **Model 3: Triangular-Sail Dynamics.** Boats with triangular sails work very differently from boats with square sails. Instead of being pushed by the wind, triangular sails act similar to airplane wings: the wind blows faster over their front than their back, causing a pressure differential that *pulls* the boat in the direction the sail is facing. This allows boats with triangular sails to sail up to about 40° off of directly upwind. However, since the sail is assymetric, this also makes sailing directly downwind very unstable. So there are two no-go zones, sectors of headings, relative to the wind, toward which sailboats can't travel. When a sailor is traveling upwind or downwind, they zig-zag on either side of the no-go zone. Switching directions in this way is called *tacking* when traveling upwind, and *jibing* when traveling downwind.

This new dynamic can be neatly added to our model of Zermelo Navigation (Section 2.1) by making velocity a function of the wind direction and the boat's heading. We created a velocity function that matches sailing speeds respecting wind direction, wind velocity, and boat heading.

The cost function J is identical to previously, but the constraint that $\dot{x} = f$ is now given by

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = v(\theta(t),\phi(x,y))\begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix} + \begin{bmatrix}w_1(x,y)\\ w_2(x,y)\end{bmatrix}$$

Where $\phi(x,y)$ gives the x and y components of the velocity of the wind. $v(\theta(t),\phi(x,y))$ is a function of both wind and boat heading which returns the speed of the boat in the direction it is heading. Multiplying by $\cos(\theta)$ and $\sin(\theta)$ splits the velocity vector of the boat into its x and y components. Both $v(\theta,\phi(x,y))$ and $\mathbf{w}(x,y)$ have units $\frac{\text{deg}}{\text{hr}}$.

2.4. **Model 4: Acceleration and Land Penalties.** Each of the previous models assumed that wind and current directly determine a boat's velocity. However, it is more accurate to model these as changing the boat's *acceleration* instead.

To model this more accurate dynamic, we track two extra states:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}x\\ y\\ \dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ a(\theta(t),\phi(x,y))\cos(\theta)\\ a(\theta(t),\phi(x,y))\sin(\theta)\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \tilde{w}_1(x,y)\\ \tilde{w}_2(x,y)\end{bmatrix}$$

Where instead of determining the velocity with wind and heading dependent v, we determine acceleration with wind and heading dependent a. Both a and the $\tilde{w}(x,y)$'s have units $\frac{\text{deg}}{\text{hr}^2}$.

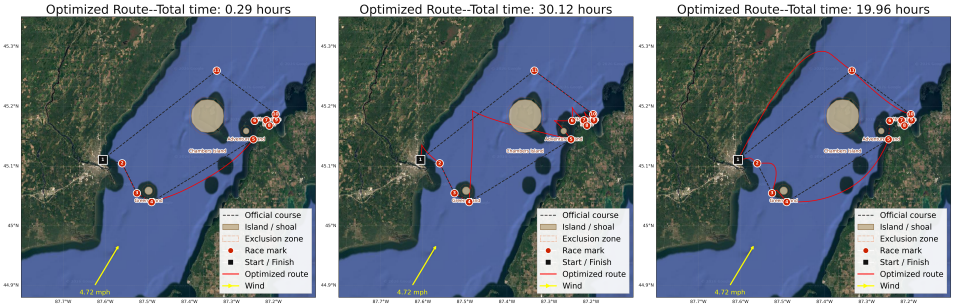

FIGURE 2. Optimized routes for the square-sail boat (left), triangular sail with velocity-based relation (middle) and acceleration-based (right) models. Since the square-sailed boat can't travel upwind, it can only sail the legs shown. The final leg of the velocity-relation model is not shown as it could not be computed.

3. SOLUTIONS

In this section we solve the optimal control problems from the previous section and explain the numerical methods we used to compute optimal paths.

3.1. **Models 1 & 2.** Under the dynamics we assumed in Section 2.1, our functional will be given by

$$\begin{aligned} J[u] &= \int_{t_0}^{t_f} 1\, dt \\ \text{subject to } \dot{\mathbf{s}} &= \mathbf{u}(t) + \mathbf{w}(\mathbf{s}(t)), \\ \mathbf{s}(t_0) &= \mathbf{s}_0, \\ \mathbf{s}(t_f) &= \mathbf{s}_f, \\ \text{and } \|\mathbf{u}\| &= v, \end{aligned}$$

For the sake of making future problems simpler to solve, we assumed that v is a function, represented by $v(\theta(t), \phi(x, y))$, indicating that the velocity of the boat is impacted by the boat angle and the wind angle. Here, $\phi(x, y)$ returns the horizontal and vertical components of wind velocity at the point (x, y). Similarly, we let the wind and current be variable here, enabling us to solve two approaches simultaneously. So, the Hamiltonian is given by

$$\begin{aligned} H &= \boldsymbol{\lambda}(t)^\top \left(v(\theta(t), \phi(x, y)) \begin{bmatrix} \cos(\theta(t)) \\ \sin(\theta(t)) \end{bmatrix} + \begin{bmatrix} w_1(x, y) \\ w_2(x, y) \end{bmatrix} \right) - 1 \\ &= \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix} \begin{bmatrix} v(\theta(t), \phi(x, y)) \cos(\theta(t)) + w_1(x, y) \\ v(\theta(t), \phi(x, y)) \sin(\theta(t)) + w_2(x, y) \end{bmatrix} - 1 \\ &= \lambda_1 (v(\theta, \phi) \cos(\theta) + w_1) + \lambda_2 (v(\theta, \phi) \sin(\theta) + w_2) - 1. \end{aligned}$$

So the costate evolution equations, and Pontryagin's Maximum Principle, will yield

$$\begin{aligned} \dot{x} &= v(\theta, \phi) \cos(\theta) + w_1(x, y) \\ \dot{y} &= v(\theta, \phi) \sin(\theta) + w_2(x, y) \\ \dot{\lambda}_1 &= -\lambda_1 \left(\frac{\partial v}{\partial x} \cos(\theta) + \frac{\partial w_1}{\partial x} \right) - \lambda_2 \left(\frac{\partial v}{\partial x} \sin(\theta) + \frac{\partial w_2}{\partial x} \right) \\ \dot{\lambda}_2 &= -\lambda_1 \left(\frac{\partial v}{\partial y} \cos(\theta) + \frac{\partial w_1}{\partial y} \right) - \lambda_2 \left(\frac{\partial v}{\partial y} \sin(\theta) + \frac{\partial w_2}{\partial y} \right) \\ \frac{\partial H}{\partial \theta} &= \lambda_1 \left(\frac{\partial v}{\partial \theta} \cos(\theta) - v \sin(\theta) \right) + \lambda_2 \left(\frac{\partial v}{\partial \theta} \sin(\theta) + v \cos(\theta) \right) = 0. \end{aligned}$$

The result is also subject to the boundary conditions

$$\mathbf{s_0} = \mathbf{s}(0) \text{ and } \mathbf{s_{t_f}} = \mathbf{s}(t_f),$$

with $H(t_f) = 0$ since there is no final time cost.

We implemented this using `scipy.integrate.solve_bvp`. In order to determine the optimal end time we treated it as a parameter and normalized the problem to go between the times 0 and 1.

The solution for the setup given in Section 2.2 is almost identical to the previous problem, just with the $\cos(\theta - \phi)$ terms accounted for; and indeed, we can account for that in the arbitrary velocity function used here. We were able to solve it similarly using `scipy.integrate.solve_bvp`.

3.2. **Model 3: Triangular-Sail Dynamics.** Given the dynamics described in Section 2.3, the Hamiltonian is given by

$$\begin{aligned} H &= \boldsymbol{\lambda}(t)^\top \left(v(\theta(t), \phi(x,y)) \begin{bmatrix} \cos(\theta(t)) \\ \sin(\theta(t)) \end{bmatrix} + \begin{bmatrix} w_1(x,y) \\ w_2(x,y) \end{bmatrix} \right) - 1 \\ &= \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix} \begin{bmatrix} v(\theta(t), \phi(x,y)) \cos(\theta(t)) + w_1(x,y) \\ v(\theta(t), \phi(x,y)) \sin(\theta(t)) + w_2(x,y) \end{bmatrix} - 1 \\ &= \lambda_1 (v(\theta, \phi)\cos(\theta) + w_1) + \lambda_2 (v(\theta, \phi)\sin(\theta) + w_2) - 1. \end{aligned}$$

So the costate evolution equations and $\frac{\partial H}{\partial u}$ are given by

$$\begin{aligned} \dot{\lambda}_1 &= -\frac{\partial H}{\partial x} = -\lambda_1 \left(\frac{\partial v}{\partial x}\cos(\theta) + \frac{\partial w_1}{\partial x} \right) - \lambda_2 \left(\frac{\partial v}{\partial x}\sin(\theta) + \frac{\partial w_2}{\partial x} \right) \\ \dot{\lambda}_2 &= -\frac{\partial H}{\partial y} = -\lambda_1 \left(\frac{\partial v}{\partial y}\cos(\theta) + \frac{\partial w_1}{\partial y} \right) - \lambda_2 \left(\frac{\partial v}{\partial y}\sin(\theta) + \frac{\partial w_2}{\partial y} \right) \\ \frac{\partial H}{\partial \theta} &= \lambda_1 \left(\frac{\partial v}{\partial \theta}\cos(\theta) - v\sin(\theta) \right) + \lambda_2 \left(\frac{\partial v}{\partial \theta}\sin(\theta) + v\cos(\theta) \right) = 0. \end{aligned}$$

Also, since there is no final time cost, we have the endpoint condition that

$$H(t_f) = 0$$

We implemented this formulation as a boundary value problem using `scipy.integrate.solve_bvp`. However, since the optimal solution discontinually switches between headings while going upwind or downwind, the solution has sharp corners which break `scipy.integrate.solve_bvp`, which expects smooth solutions.

To mitigate this issue, we switched to implementing this problem as an interior point optimization problem, and used the solver implemented by the `IPOPT` package, with Julia's `JuMP` optimization package. This solver essentially minimizes an objective function subject to equality and inequality constraints using a form of Newton descent. Our objective function was simply total time, and the $\dot{x} = f$ relation was implemented as a leapfrog Euler discretization of the two-state system, on each time-step of the solution. For $i = 1, \ldots, N-1$, we enforce the constraints:

$$\begin{aligned} (N-1)(x_{i+1} - x_i) &= T \cdot v(\theta_i, x_i, y_i)\cos(\theta_i) \\ (N-1)(y_{i+1} - y_i) &= T \cdot v(\theta_i, x_i, y_i)\sin(\theta_i) \end{aligned}$$

Where N is the number of timesteps, T is the final time, so $\frac{N-1}{T}$ is effectively $\frac{1}{\Delta t}$.

This implementation is, unsurprisingly, very finicky depending on initial guesses. Our initial guess is somewhat crude but works well. The path from beginning to end is initialized as a straight path, and θ starts a bit outside of the up-wind no-go zone on one side, and switches to the other halfway through the path, simulating a single tack. That initialization broke for legs of the race that do not require a change of direction (2, 3, 9, 10), and for those we initialized with a single heading. Leg 11, which requires a tack, broke the solver regardless of initialization. We tried tuning several hyperparameters such as mesh size, tolerance, and maximum iterations.

This implementation mostly worked well and handled the discontinuous nature of the solution effectively. However, initially, when traveling in a near-upwind or near-downwind direction, the solution switched rapidly between headings, which is an unrealistic route for a physical boat. As such we added a penalty term to the objective that penalized changes in heading.

3.3. **Model 4: Acceleration and Land Penalties.** We will again work out the details of Pontryagin's Maximum principle before discussing our implementation. With the $\dot{x} = f$ relation as described in Section 2.4, the Hamiltonian is given by

$$H = \lambda_1 \dot{x} + \lambda_2 \dot{y} + \lambda_3(a(\theta,\phi)\cos(\theta) + \tilde{w}_1) + \lambda_4(a(\theta,\phi)\sin(\theta) + \tilde{w}_2) - 1.$$

And so, with the Hamiltonian equations and by Pontryagin's Maximum principle, the costate evolution equations and $\frac{\partial H}{\partial \theta}$ are given by

$$\begin{aligned}
\dot{\lambda}_1 &= -\lambda_3 \left(\frac{\partial a}{\partial \phi}\frac{\partial \phi}{\partial x}\cos(\theta) + \frac{\partial \tilde{w}_1}{\partial x} \right) - \lambda_4 \left(\frac{\partial a}{\partial \phi}\frac{\partial \phi}{\partial x}\sin(\theta) + \frac{\partial \tilde{w}_2}{\partial x} \right) \\
\dot{\lambda}_2 &= -\lambda_3 \left(\frac{\partial a}{\partial \phi}\frac{\partial \phi}{\partial y}\cos(\theta) + \frac{\partial \tilde{w}_1}{\partial y} \right) - \lambda_4 \left(\frac{\partial a}{\partial \phi}\frac{\partial \phi}{\partial y}\sin(\theta) + \frac{\partial \tilde{w}_2}{\partial y} \right) \\
\dot{\lambda}_3 &= -\lambda_1 \\
\dot{\lambda}_4 &= -\lambda_2 \\
\frac{\partial H}{\partial \theta} &= \lambda_3 \left(\frac{\partial a}{\partial \theta}\cos(\theta) - a\sin(\theta) \right) + \lambda_4 \left(\frac{\partial a}{\partial \theta}\sin(\theta) + a\cos(\theta) \right) = 0.
\end{aligned}$$

And since there are still no endpoint penalties, we have that

$$H(t_f) = 0$$

The implementation of the model described in Section 2.4, however, is very similar to the previous. The only difference is the acceleration function a in place of the velocity function v. It was a little trickier since the acceleration applied to the boat by the wind depends not on the absolute wind-speed but on *apparent* wind speed, or the wind speed experienced by the boat. Otherwise the boat would continually accelerate forever on any fixed heading. We used the estimation that a racing sailboat would take 30

minutes on any given heading to reach maximum speed on that heading, and added the appropriate multiplicative factor.

As in the previous implementation, we incorporate the $\dot{x} = f$ relation via a standard leapfrog-style Euler discretization of the 4-state system. For $i = 1, \ldots, N-1$, we enforce the constraints:

$$\begin{aligned}
(N-1)(x_{i+1} - x_i) &= T\dot{x}_i \\
(N-1)(y_{i+1} - y_i) &= T\dot{y}_i \\
(N-1)(\dot{x}_{i+1} - \dot{x}_i) &= T \cdot a(\theta_i, x_i, y_i, \dot{x}_i, \dot{y}_i)\cos(\theta_i) \\
(N-1)(\dot{y}_{i+1} - \dot{y}_i) &= T \cdot a(\theta_i, x_i, y_i, \dot{x}_i, \dot{y}_i)\sin(\theta_i)
\end{aligned}$$

We used the same initial guess as for the previous model. This implementation still switches between directions but does so smoothly. The acceleration-relation was friendlier to the solver than the velocity-relation, and ran quickly without needing any alternate initializations.

3.4. **Land Penalties.** The interior point problem formulation made it very simple to add in coastline constraints with the `IPOPT` solver, which we used for models 3 and 4. We used WebPlotDigitizer to find points along the coastline, then polynomially interpolated to create functions that represented the coastlines. Then we added time-step-wise constraints like so: For $i = 1, \ldots, N-1$, we enforce the constraints:

$$\begin{aligned}
(y_i) &\leq \text{north}(x_i) \\
(y_i) &\geq \text{south}(x_i)
\end{aligned}$$

Where north() and south() are the interpolated functions representing the north and south coastlines, respectively.

For numerical solutions to the optimal control problem, we can implement different types of land penalties as different cost functions that penalize the boat for getting too close to land. Modeling each island as an ellipse and each coastline as a sequence of lines segments along the coast, possible choices for the two different land penalties include

$$C_{\text{island}}(x, y) = \frac{1}{((x - c_x)^2/r_x + (y - c_y)^2/r_y)^n} \quad \text{and} \quad C_{\text{coast}}(x, y) = \frac{1}{d^n},$$

where d is the calculated closest distance from the boat to the nearest point along the coastline interpolation. The widths and center of the island can be controlled with (c_x, c_y) and (r_x, r_y). The steepness of the penalty can be controlled with n.

While these constraints were simple to add to the implementations, they complicated an already messy nonlinear optimization problem, and made several of the legs locally infeasible for the initial guesses we used. This issue was especially pronounced for the version that related acceleration to wind and heading.

3.4.1. *Derivation of KKT Conditions with Land Constraints.* Let's suppose that we have land constraints for the islands and coastlines to prevent the ship from entering certain exclusionary zones around the land masses. We can define each exclusionary zone as its own inequality constraint, i.e., $\mathbf{h}(x, y) \preceq \mathbf{0}$. The modified Lagrangian becomes

$$\begin{aligned}\mathscr{L} &= H - \boldsymbol{\mu}(t)^\top \boldsymbol{h}(x, y) \\ &= \lambda_1 \dot{x} + \lambda_2 \dot{y} + \lambda_3(a(\theta, \phi)\cos(\theta) + \tilde{w}_1) + \lambda_4(a(\theta, \phi)\sin(\theta) + \tilde{w}_2) - 1 \\ &\quad - \boldsymbol{\mu}(t)^\top \boldsymbol{h}(x, y).\end{aligned}$$

The position costates then become

$$\begin{aligned}\dot{\lambda}_1 &= -\frac{\partial H}{\partial x} + \boldsymbol{\mu}(t)^\top \frac{\partial \mathbf{h}}{\partial x} \\ \dot{\lambda}_2 &= -\frac{\partial H}{\partial y} + \boldsymbol{\mu}(t)^\top \frac{\partial \mathbf{h}}{\partial y},\end{aligned}$$

where $-\frac{\partial H}{\partial x}$ and $-\frac{\partial H}{\partial y}$ are as derived before for the original position costates.

The velocity costates $\dot{\lambda}_3$ and $\dot{\lambda}_4$ remain unchanged as the inequality constraints don't depend on $\dot{x}$ or $\dot{y}$. The KKT conditions then become

$$\begin{gathered}\frac{\partial \mathscr{L}}{\partial \theta} = 0, \\ \mathbf{h}(x, y) \preceq \mathbf{0}, \quad \boldsymbol{\mu}(t) \succeq \mathbf{0}, \text{ and } \mu_j(t) h_j(x, y) = 0 \quad \forall j.\end{gathered}$$

Note that the stationary condition $\frac{\partial \mathscr{L}}{\partial \theta} = 0$ remains unchanged from the unconstrained situation as the constraints do not depend on θ.

4. Interpretation

While the simplest two models we worked with were not able to rediscover real sailing techniques, the more complicated ones were. By speciying exact waypoints throughout the race course to travel through, the boat is able to travel in a somewhat realistic race path.

4.1. **Comparisons of Results.** The solution to the Zermelo Navigation problem was more of a warm-up for the paper than an intended solution, so we don't compare it to the other models here.

As soon as we incorporated a relation between wind and heading that was similar to real racing dynamics, we found optimal paths that tacked or jibed once. With our model that related wind and heading to velocity directly, we got discontinuous, jagged paths with sharp corners. When we related wind and heading to acceleration, which is physical, we got curved, drifting paths that the boat followed, but still with a single tack or jibe. The simplest sailing approach did not discover tacking or jibing, since we made assumptions that made it impossible for the boat to go upwind.

4.2. **Potential improvements.** While we accounted for the impact of current, we did not account for the impact of waves on sailing. Similarly, we did not examine the impact of other racers, wind shadows (wind going slower after passing over land), bad weather, and friction. There are many paths that could be explored further, although at some point it becomes a problem of optimizing boats and not routes.

In reality, the fastest sailors do not follow paths like the ones that we found. Since wind shifts direction and speed, sailors switch direction more often both to take advantage of shifts and to account for potential future shifts. Additionally, right-of-way rules incentivize racers to switch directions more often both to stay out of the way for other racers and to force other racers to change their trajectory. However, if it were known that wind would be constant, as in our formulation, and there were only one sailboat, a single tack per race leg would be perfectly reasonable.

4.3. **Impact.** With continually growing concerns about the impact of fossil fuel use on the environment, and with the ever-present motivation to cut costs, the impact of cutting shipping emissions using sails cannot be overstated. One company, Vela, is already using a 100% wind-powered ship with triangular sails to transport cargo across the ocean faster than other shipping vessels. Not only does the prototype, a type of trimarian, cut down on emissions, it also is much quieter, less disturbing to wildlife than traditional shipping, and individual boats can cross the Atlantic 2-4 times faster than traditional cargo ships [Pag26].

As this technology grows more widespread, the problem of determining optimal sailing routes will once more be extremely important to global commerce, making markets more efficient and helping to combat global climate change.

5. FERPA Release

We give Dr. Jarvis and other instructors at BYU teaching ACME classes permission to share our project as an example of a good project in future classes they teach.

- Connor McBride
- Ethan Palenske
- Jackson Pond

References

[Mar24] Marinette & Menominee Yacht Club. 100 miler race charts, 2024. Accessed: March 27, 2026.

[MPP26] Connor McBride, Ethan Palenske, and Jackson Pond. vol4-second-project. `https://github.com/EthanPrimes/vol4-second-project`, 2026. GitHub repository, MIT license, accessed 2026-4-11.

[Pag26] Tom Page. This wind-powered cargo ship wants to cut emissions by up to 96% and deliver faster than conventional sea freight, 2026. CNN Climate. Published: February 12, 2026. Accessed: April 15, 2026.