Goal

Plan and updates

- Do prior research
  - I'm sure that someone has already found the ideal racing/paddling curve already, but is the solution easily findable online?
    - Try this paper
    - Longshot, but try this paper
    - Found it; we're dealing with Calculus of Variations, summarized near the end of this presentation
- Learn how to include your own Python functions
  - I'm gonna use modules (should've learned this before)
- Write the first two sections of the paper and draw some graphs with PowerPoint that explain how other paddlers vs you see the ideal paddling curve
- Create the program
  - It must do blank
  - It must do blank
  - The main program should be a short file; create function files to go in your folder, which will be "included" by the main program
- Blah
- Upload this sub-paper ("Ideal Paddling Power Curve") to GitHub, and update/organize my Github homepage to include mentions of it and my other programs

Section 1: State the Problem

In outrigger canoe races, paddlers aim to complete a course as quickly as possible, and they calculate strategies to achieve this. For example, paddlers and their coaches will theorize how to maximally use all muscle groups to achieve the strongest stroke.
Additionally, many paddlers adopt a deliberate technique when it comes to allotting their energy along the course. The usual strategy is as follows:

- *Begin the race with a surge of power to quickly accelerate the boat.*
- *For the majority of the course, maintain a constant speed by using less power now.*
- *At the end of the course, increase power and finish with a high velocity.*

This widely-adopted strategy is considered satisfactory for many paddlers, because it is mentally-encouraging, simple to remember, and makes easy the relinquishing of all remaining energy a paddler has (it can all be dumped at the finish without restraint).

However, I believe that this strategy is not the ideal way to allot a paddler's energy over the course. Before using rigorous math or modeling, I propose the three points:

- Hydrodynamic drag is not a constant force, but is known to increase quadratically with the moving object's velocity, so maintaining a relatively constant boat velocity over the course makes more sense than peaking it at any point (such as at the end, with a strong finish).
    - If drag was a constant force, then velocity peaks would not be especially punishing.
    - See the greatly-simplified mathematical model (calculus of variations) which involves a paddler who wants to minimize the total energy lost to drag over the course:
        - $Course\ distance\ =\ Constant\ =\ \int_{Starting\ time}^{Finish\ time} \frac{dx}{dt}\,dt = \int_{Starting\ time}^{Finish\ time} v(t)\,dt$
        - $Minimize\ this\ quantity$:
          $Drag\ energy\ =\ \int_{Starting\ line}^{Finish\ line} F_{drag}(x)\,dx$
          $= \int_{Starting\ time}^{Finish\ time} P_{drag}(t)\,dt$
          $= \int_{Starting\ time}^{Finish\ time} F_{drag} \cdot v(t)\,dt$
          $= \int_{Starting\ time}^{Finish\ time} friction\ coefficient\ \cdot\ v(t)^2 \cdot v(t)\,dt$
          $= \int_{Starting\ time}^{Finish\ time} friction\ coefficient\ \cdot\ v(t)^3\,dt$
        - This leads to the situation where $\int$**v(t)dt** is a constant but $\int$**v(t)³dt** must be minimized. Intuition tells me that the solution is **$v(t) = v_{ideal}$** , unvarying, the entire time, where v_{ideal} is a fixed velocity value. There will be no peaks by this model. This implies that velocity peaks have a cost and should be avoided.
- An especially-strong start does have benefit that could outweigh the drag cost; getting the boat up to a maintainable velocity quickly is important, since the boat starts at zero velocity and this will hurt that paddler's time.
- An especially-strong finish (having a high final boat velocity) leaves a large amount of kinetic energy stored in the boat. This is wasteful. While some kinetic energy must be sacrificed in a boat finish (you should not finish at a crawl, as this drags out your time), it seems unwise to make this amount especially large.
    - For example: instead of peaking boat speed at the end of a course, if a paddler produces a peak in the middle of the course, the extra kinetic energy of the boat has time to eventually be transferred into the fight against drag over remainder the course, helping the paddler's overall time.

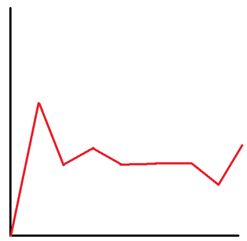
Section 2: Prior Research

B

Section 3: The Modeling Program

Basic Idea

Let's set the paddler's Energy to a fixed amount, and see how they dispense it over the course (Paddler Force).

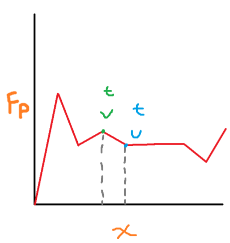


We will start with some graph of points. This is the Force vs. Distance graph. A linear fit is done between every pair of points.

★ The graph will be normalized (all y values scaled up or down by the same factor) so that the area under the curve is our fixed Energy value.

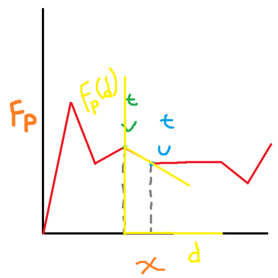
★ *Calculations* will be done piecewise for every graph region, so that we can find the total **Time** that the paddler requires to finish the course. According to this current graph.

The graph will be modified slightly (one of the points will be increased or decreased). Then, the above points are repeated. If the new **Time** is smaller than the old one, then this new graph becomes our new model graph.



The question is how to do those *calculations*. Take the green point on the graph, for example. We need to progress onto the blue point. This means using our current (green) values of ***time and velocity***, and updating them.

Basically, the paddler is traveling against a Drag Force which is changing, and also with a Paddler Force which is also changing (although, only linearly). Since the Drag Force is a tough time-based equation, we'll have to use velocity to calculate it and also vice-versa.



Using a new distance “d”, we can now model this step from green to blue. The yellow graph covers the space from green to blue **(the “yellow” region)**, and uses the $F_{Paddler}(d)$ equation.

Total Force = $F(d) = F_{Paddler} + F_{Drag} = (F_{P0} + A * d) - B * v(d)^2$

Boat Kinetic Energy = $K(d) = \frac{1}{2}mv(d)^2 = \frac{1}{2}mv_0^2 + \int_0^d F(d)dd$

Combine: $\frac{1}{2}mv(d)^2 = \frac{1}{2}mv_0^2 + \int_0^d (F_{P0} + A * d - B * v(d)^2)\, dd$

Rearrange: $v(d) = \sqrt{v_0^2 + \frac{2}{m}\int_0^d (F_{P0} + A * d - B * v(d)^2) dd}$

Simplify: $v(d) = \sqrt{v_0^2 + \frac{2 * F_{P0} * d}{m} + \frac{Ad^2}{m} - \frac{2B}{m}\int_0^d (v(d)^2) dd}$

A can be found from the F_P *vs. x* graph
B is the drag factor **(user must decide)** (perhaps ???)
m is paddler+boat mass **(user must decide)** (perhaps 50kg)
Area under the curve is total paddler energy **(user must decide)** (perhaps 50,000 J)
X , or x_f , is total distance of race **(user must decide)** (perhaps 250m)

I don’t know how to solve this equation. It looks like it might be analytically solvable, but I’ll use a numeric method instead. I’ll break this up into iterations; using the previous v(d) solution as the internal v(d) under the root, in order to find the new v(d).

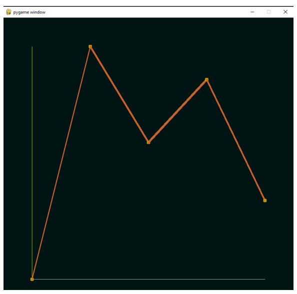
Once the entire portion is done (green → blue transition), we’ll have a set of v(d) points. Of course, this includes v(D), a.k.a. the blue velocity value, which is essential. However, we also need the blue time value. To find this, I’ll numerically (with tables values) integrate:

$$\int_0^D \frac{1}{v(d)} dd = T$$

This time value isn’t actually needed for any calculations until the end, when we sum them all up to find the paddler’s total time on the course.

Section 4: Find the Solution

A randomly-generated graph (Paddler Force vs. distance) will be used as the start graph, as such:

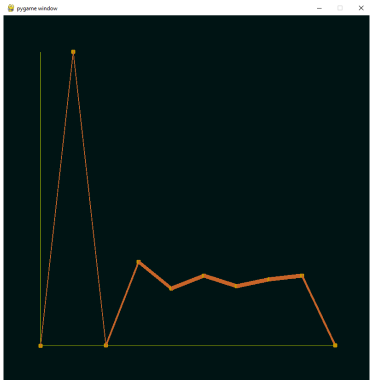


The program will calculate the amount of time that the paddler used to complete the course, according to calculations done from this graph's data.
Next, we will vary the points on the graph, while keeping the area underneath the curve constant (representing the paddler's total energy expenditure). We will find the total time calculated from the new graph.
We will keep varying the graph, one point at a time, until we reach a graph that seems ideal (has the lowest time).
This varying of one point at a time (by slightly increasing or decreasing its y value) will be called **tweaking**. Note that every time a point is tweaked upward, the rest of the graph's points naturally will be pushed downward so that the graph's total area remains the same. The reverse happens as points are tweaked downward.
We will know that we've reached an ideal solution when there are no points on the graph whose tweaking produces an improvement. While this solution is technically a *local solution*, I believe that the nature of the problem (and how our randomly-generated start graph will likely not resemble any edge cases or extreme graphs) makes it very likely that any local solution found will also be the one-and-only global solution to the problem.

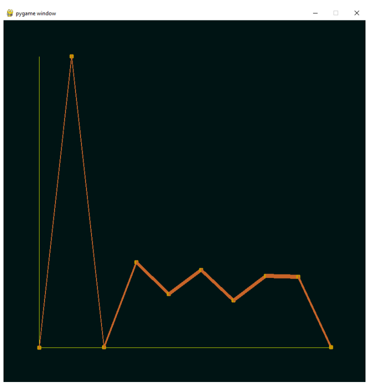
<u>The solution:</u>
With variables set to the following:

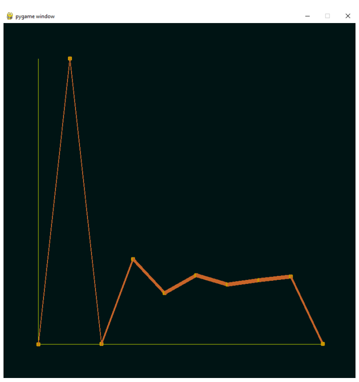
- Main function variables:
  - *dimension = 10* (points on the chart)
  - *tweak_ratio = 1.01*
  - *x_max = 1* (race distance)
  - *y_max = 5* (max power)
  - *factor = 0.26*
- yellow() function variables:
  - *m = 100* (paddler mass)
  - *B = 1000* (combined drag factor)
  - *steps = 300* (iterations per yellow region)

The result is 31.2 seconds. With the following graph:



The program takes about a minute to complete and doesn't seem to be sensitive to the first graph (which serves as a random-generated seed). When I run the program again, similar results occur:





While the results are relatively consistent, it's clear that **a greater number of points is needed** to fully understand what happens at the beginning and the end.
For now, the rough picture is that a paddler should explode at the race start for a short period of time, then sustain a constant pace, and cut power at the very last moment.
What is essential at this point is that I produce a **velocity vs. time** curve as well.

Section 5: Conclusion and Analysis

Section 6: Works Cited

[0] Author Name. Year. *Title.* Website or Company Name. URL.
[1] Drake Sorkhab. 2023. *Ideal Power Paddling Curve*. Github Page “35drake”.
https://github.com/35drake