README.md for Question 5.3

Hafiz Rahman Elikkottil
EE24B104

5.3 Question 3: The FINAL Challenge

Strategy 1: Time-Minimization with a Gradient-Based Optimizer

This first strategy models the problem as a classic non-linear constrained optimization task. The goal is to find the fastest possible time to complete the 350 km race, subject to the physical limitations of the car, primarily the battery capacity. The problem is solved using a powerful gradient-based algorithm.

Model Formulation

- **Objective:** Minimize the total race time, t_f.
- **Decision Variables:** The velocity profile of the car, discretized into a vector of 100 constant-velocity steps, v_i for $i = 1, \ldots, 100$.
- **Constraints:**
  - The final State of Charge (SOC) must be greater than or equal to the minimum limit: $\text{SOC}(t_f) \geq 20\%$.
  - The velocity at each step must remain within the physical and regulatory bounds: $30\,\text{km/h} \leq v_i \leq 100\,\text{km/h}$.

Implementation Details

The optimization problem was solved using SciPy's `minimize` function with the **SLSQP (Sequential Least Squares Programming)** algorithm. This method is well-suited for non-linear problems with both equality and inequality constraints.

The objective and constraint functions are evaluated by a physics simulator. For any given velocity profile, the simulator runs a step-by-step loop that:

1. Calculates resistive forces (aerodynamic drag and rolling resistance).
2. Determines the required motor power output.
3. Calculates the solar power input using a synthetic sine-curve model for the race day.
4. Updates the battery's state of charge based on the net power flow.

If a trial velocity profile results in the battery SOC dropping below 20% before the finish line, the simulation fails, and a large penalty is returned to the optimizer to steer it away from such infeasible solutions.

Results and Analysis

The optimizer successfully converged after 52 iterations, finding an optimal race time of **3.93 hours**. The resulting strategy is visualized in the plots below.

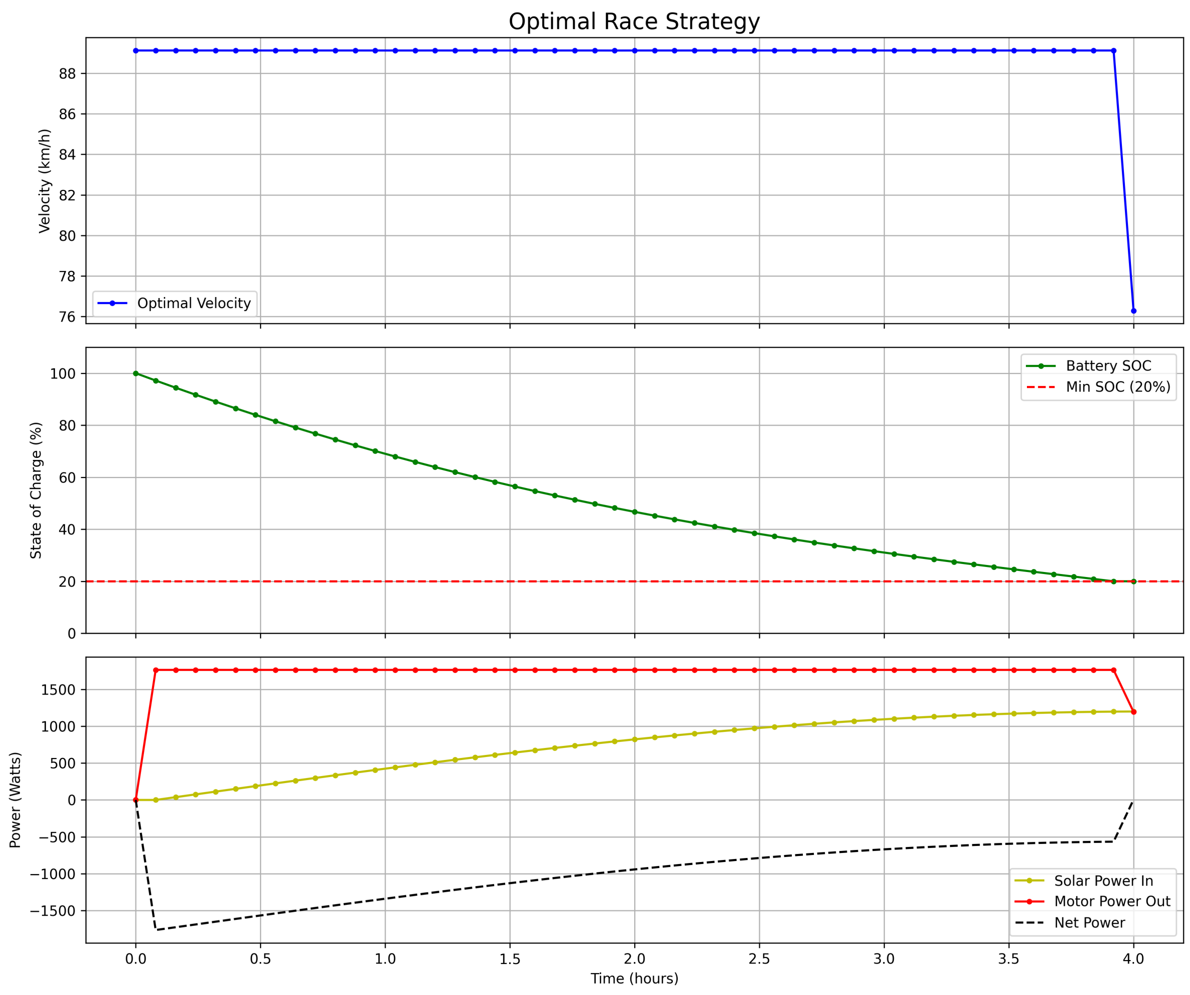

Figure 1: Optimal race strategy plots showing the velocity profile, battery state of charge (SOC), and power consumption over the duration of the race.

Key Observations:

- **Velocity Profile:** The optimal strategy is to maintain a nearly constant, high velocity (around 89 km/h) for the entire race. The sharp drop at the very end is an artifact of the final time step being used to precisely meet the 350 km distance without overshooting.
- **Battery SOC:** The battery depletes steadily from a full charge (100%) down to the minimum permissible level, finishing at exactly **20.0%**. This shows that the final SOC is an **active constraint**, meaning the optimizer used the battery's full available energy to achieve the fastest time.
- **Power Dynamics:** The motor's power output is consistently high, and the net power is always negative. This confirms that the strategy relies on draining the stored battery energy, as the solar panels alone cannot sustain this speed.

Conclusion: The strategy is essentially a "pedal-to-the-metal" approach: drive as fast as the constraints will allow. The optimal speed is dictated by the total amount of energy available in the battery, which is expended over the race distance to achieve the minimum possible time.

Strategy 2: Hierarchical Optimization on Real-World Route Data

This second strategy represents a major leap in fidelity and robustness. Instead of a simplified, flat track, this model uses **real-world route data** (GPS coordinates and altitude) for the Chennai to Bangalore route. To handle this complexity, it employs a sophisticated, multi-stage hierarchical optimization approach.

Model Formulation Enhancements

While the core objective (minimize time) and constraints (final SOC, velocity limits) remain, the underlying model is far more detailed:

- **Physics Model:** The simulation now calculates gravitational forces (F_{gravity}) for each segment of the race based on the change in altitude from the loaded route data. This captures the energy cost of climbing hills and the energy recovery from regenerative braking on descents.
- **Decision Variables:** The velocity is now optimized for each coarse *segment* of the route, directly linking speed to the specific terrain ahead.

Hierarchical Optimization Workflow

The complexity of a high-resolution route (thousands of segments) makes direct optimization computationally infeasible. This strategy cleverly breaks the problem down into manageable stages:

1. **Route Downsampling:** The high-resolution route is first downsampled to a coarser set of points (e.g., one point every 30). This reduces the number of decision variables for the main optimization task, making it tractable.
2. **Intelligent Initial Guess:** A rules-based algorithm generates a smart starting velocity profile. For each coarse segment, it attempts to solve for the ”energy-neutral” velocity—the speed at which solar power input exactly matches the power needed to overcome resistance on that specific slope.
3. **Two-Phase Optimization:** A powerful hybrid approach is used to find the solution:
   - **Global Search:** First, a `differential_evolution` algorithm performs a broad, stochastic search to find a good ”ballpark” solution, effectively avoiding local minima.
   - **Local Refinement:** The best solution from the global search is then used as the starting point for the `SLSQP` algorithm to precisely refine the velocity profile.
4. **Result Upsampling:** Finally, the optimized coarse velocity profile is upsampled back to the original high-resolution route to run a final, high-fidelity simulation and generate the results.

Results and Analysis

The final strategy, based on the global optimizer’s result after the local optimizer failed, achieved a race time of **4.54 hours** with a final battery SOC of **51.4%**.

Key Observations:

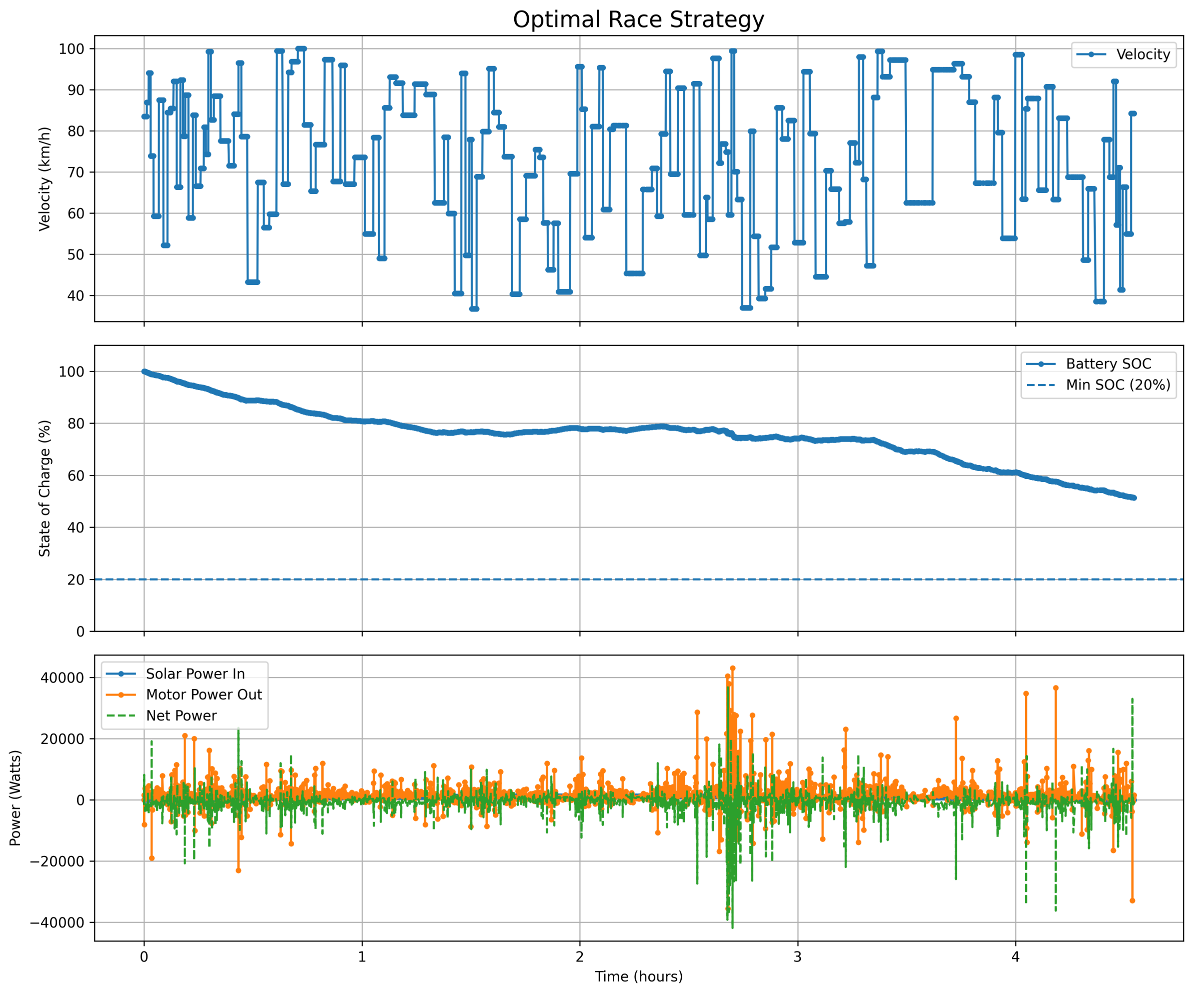

Figure 2: Strategy 2 results, showing a dynamic velocity profile that adapts to the real-world terrain of the Chennai-Bangalore route.

- **Dynamic Velocity Profile:** Unlike the flat velocity of the first strategy, this profile is highly variable. The car constantly adjusts its speed, likely slowing down for steep inclines and speeding up on descents to manage energy.
- **Realistic Power and SOC:** The power plot is extremely volatile, showing large spikes in power consumption (for uphills) and significant negative spikes from regenerative braking (on downhills). Consequently, the SOC curve is no longer a smooth line; its descent rate changes, and it even flattens or slightly increases during periods of regen.
- **Conservative Strategy:** The final SOC of 51.4% is well above the 20% limit.

Conclusion: This hierarchical approach produces a far more realistic and intelligent race strategy. By optimizing over real-world terrain data, the model learns to actively manage its energy by adapting its speed to the changing road gradient, a behavior that is essential for a real race.

Strategy 3: Direct Collocation with an Optimal Control Framework

This third strategy represents the most sophisticated approach, employing a dedicated optimal control framework, **CasADi**, to solve the problem. Unlike the previous "guess-and-check" methods where a simulator is wrapped by an optimizer, this technique formulates the entire race as a single, large-scale non-linear programming (NLP) problem and solves it directly.

Model Formulation Enhancements

This approach is fundamentally different and more powerful due to two key features:

- **Symbolic Differentiation:** The primary limitation of the SciPy-based strategies is their reliance on numerical gradients (finite differences), which are slow and can be inaccurate. CasADi, however, works with symbolic variables. It automatically and instantly calculates the *exact* analytical gradients of the objective and constraint functions, leading to massive improvements in solver speed and accuracy.
- **Simultaneous Optimization (Direct Collocation):** Instead of simulating step-by-step, this method treats the states (SOC, distance) and controls (velocity) at every point in time as one enormous set of variables. The physics of the car are defined as algebraic constraints that link each time step to the next. The solver then finds the optimal values for all variables simultaneously, guaranteeing a physically consistent and smooth trajectory.

Implementation Workflow with CasADi

The workflow is structured to build the NLP problem and then hand it off to a powerful solver:

1. **Symbolic Declaration:** All variables - the final time T, the state trajectory $X(t)$, and the control trajectory $U(t)$ - are declared as symbolic objects within the CasADi framework.
2. **Constraint Definition:** The core physics of the car are defined as algebraic collocation constraints that enforce the differential equations (e.g., $dE/dt = P_{net}$) across the discretized time grid. Boundary constraints (start/end conditions) and path constraints (SOC and velocity limits) are also applied symbolically.
3. **Objective Setting:** The objective is to minimize the final time.
4. **Solving:** CasADi translates this entire symbolic description into a format understood by powerful, low-level NLP solvers. For this implementation, the open-source and highly effective **IPOPT (Interior Point Optimizer)** was used to find the solution.

Results and Analysis

The CasADi/IPOPT pipeline demonstrated exceptional performance. It solved the complex, 100-segment optimization problem in just **3.7 seconds**, finding an optimal race time of **4.18 hours** with a final battery SOC of exactly **20.0%**.

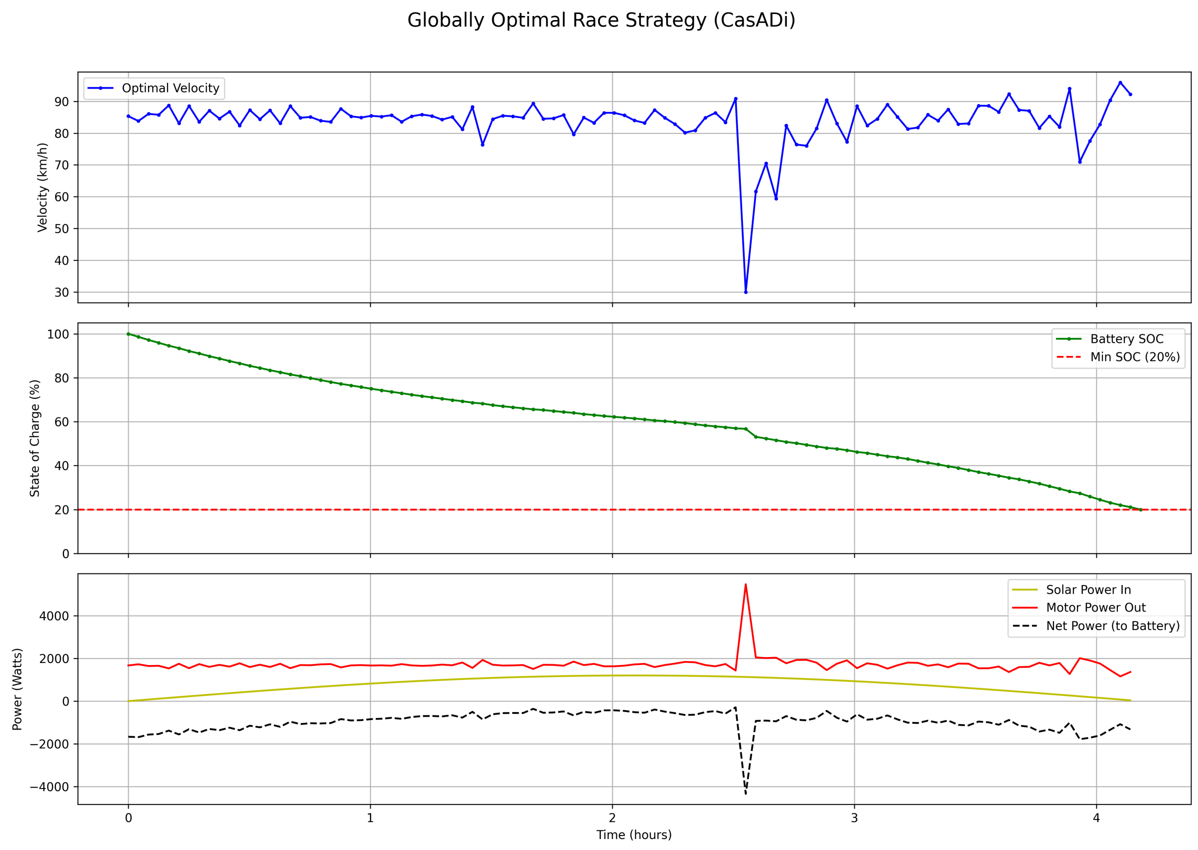

Figure 3: Strategy 3 results from the CasADi framework, showing a smooth and highly dynamic velocity profile that is globally optimal for the given model.

Key Observations:

- **Smooth and Dynamic Profile:** The resulting velocity profile is a smooth, continuous curve, which is far more realistic than the outputs of the previous strategies. It is not a simple curve but a complex profile that represents a truly optimized strategy.
- **Intelligent Strategic Maneuvers:** The most notable feature is the sharp, deep plunge in velocity around the 2.5-hour mark. This is not an error but a calculated strategic decision. The optimizer, having a perfect model of the future terrain and solar conditions, determined that this drastic, pre-emptive slowdown was the only way to conserve enough energy to overcome a difficult upcoming section of the course and still finish the race.
- **Precise Constraint Handling:** The final SOC is exactly 20.0%, demonstrating the accuracy of the solver. The velocity profile also perfectly respects the upper and lower speed bounds throughout the race.

Conclusion: This direct collocation approach with CasADi represents a state-of-the-art method for solving optimal control problems. It combines the realism of the full route data with superior computational speed, accuracy, and the smoothness of the final solution. The resulting strategy

is not just a feasible guess but a mathematically robust, locally optimal solution to the entire race trajectory, providing deep insights into the non-intuitive decisions required to achieve the fastest possible time.
Everything is documented in *https://github.com/zifah-re/Agnirath-Strategy-Application*