Report - Assignment 2
COMP765 (Fall 2017)
Prof. David Meger
Github URL[1]

Shabir Abdul Samadh
ID: 260723366
Date: 12/15/2017

Cart-pole balancing

The objective of the assignment was to write a **P**roportional-**I**ntegral-**D**ifferential (`PID`) controller to balance an inverted pendulum on a cart whilst keeping the cart position at the origin. In addition to this it was also required to experiment with a parameter tuning algorithm (other than the *manual-tuning* process) to obtain best PID gains for the controller. A complex-controller was also to be experimented with from a given set of choices.

Implementation

0.1 PID Control

I have implemented the PID system for the case of balancing the pendulum with the initial state being upright. The implementation uses a combination of two different PID controllers: `PID for the pendulum` & `PID for the cart`. The PID system for the pendulum takes into account the *angle* & the *angular-velocity* of the pendulum whilst the PID controller for the cart uses the *velocity* & the *distance* of the cart from the origin. The proportional gains *(for both controllers)* are subjected against the positional errors: the distance from origin in the case of the cart and the deviation-angle from its intended balance position ($\pi - radians$) for the pendulum. The cumulative value of these positional-errors are used to generate the integral gain for the system. Moreover, the velocity terms for both the cart and the pendulum were utilized as inputs for the differential gain.

The pendulum's angle (θ) was passed into the PID control function in its $\cos(\theta)$ and $\sin(\theta)$ values. Hence, according to the position of the pendulum, these values varied in a non-uniform manner. It was required to re-scale them to a range such that the positional error (of the pendulum) calculated using θ reflected the **magnitude** and the **direction** of the force to be applied on the cart. Thus, I converted the input-angle to be within a range between $(0 - 2\pi)rad$ based on the cos and sin values of the angle in different quarters of the coordinate system. This conversion ensured that a positive-error *(indicating positive force)* was required when the pendulum was on the right side of the cart and vice-versa when it was on the left. It also ensured that the magnitude of the force was higher for pendulum states in the lower quarters of the plane. The cart's positional error was deduced by subtracting the distance of the cart from the origin *(received as an input to the function)* from `ZERO`. Any state to the right of the origin demanded a negative force backwards and vice-versa when it was to the left of the origin.

The differential errors were calculated directly by subtracting the the velocities of the two entities (`cart and pendulum`) from `ZEOR`. However, when using this error term (with the *gain* parameter) in the controller it was noted that the direction of the pendulum's differential error had to be converted. This was because the pendulum's velocity was calculated positive to the anti-clockwise direction. When the pendulum is having a positive velocity *(imagine the pendulum on the top-right quarter)* the optimal control on the cart must be positive to project it

[1]https://github.com/Shabirmean/Assignment2

towards the intended upright position. The differential error, when calculated as a difference between ZERO and a positive velocity gives a negative value. This results in a force pulling the cart further towards the left. This is evaded by inverting the sign of the differential error of the pendulum.

Two global variable were maintained to keep track of the cumulative errors of each of the two control entities in the system. This cumulative error along with the learned gain was used to produce the integral gain for the two PID systems. The three `(P,I,D)` controls for the two entities *(the cart and the pendulum)* were summed separately to produce the final control parameter per PID system. Finally, the difference between the control value for the pendulum's PID system and the cart's PID system ($PID_{pendulum} - PID_{cart}$) was output as the optimal control parameter for the system.

0.2 Twiddle parameter learning

Upon setting up the PID system, some values were manually varied to find suitable gains (K_P, K_I, K_D) for the P, I, D components. It was noticed that even the smallest values for the integral gain K_I *(for both PID systems)* caused the system to lose balance. Hence the K_I term was kept `ZERO` throughout the manual tuning process. Upon, manually experimenting with various combinations for the gain terms, I was able to observe a **close-to-equilibrium** state when the K_P terms were 100 and 50 for the pendulum and the cart respectively. I stopped the manual tuning at this point and switched to `Twiddle` to further optimize and find the best possible gain combination.

The twiddle algorithm was run with the initial parameters being the learned values *(via manual tuning)* of $K_P^{cart} = 50$ and $K_P^{pendulum} = 100$. Initially twiddle was used to only tune the parameters for $PID_{pendulum}$. Then, with the learned parameters for the pendulum controller, it was re-run for PID_{cart}. The final learned parameters for both the controllers were as follows:

- $K_P^{cart} = 50.77308$, $K_I^{cart} = 0.0$, $K_D^{cart} = 2.144549$
- $K_P^{pendulum} = 100.02169$, $K_I^{pendulum} = 0.0$, $K_D^{pendulum} = -0.951134$

The twiddle algorithm was run until a tolerance of 0.02 was reached for the sum of the 6 parameter scaling values *(which were set to* 1 *at the beginning)*. The **best-cumulative-error** recorded for the task (over 100000 iterations) with these parameters was 420.91.

Learning process & outcomes

I started implementing the controller based on the basic idea behind PID control systems. Initially, I addressed the task as a single PID control problem and implemented the error terms *(proportional, integral & differential)* as a combination of both: the state variables of the pendulum and the cart. However, this resulted in a very unstable system. Even with numerous combinations of the parameter variables I was not able to observe any reasonable control. Then, upon talking to some of my colleagues I realized that the problem had to be approached as a system of 2 separate PID controllers. This allowed me to come up with a more stable solution. With the splitting of the controllers into two parts, I had to also verify that each of the error terms for the two PIDs were adjusted appropriately.

In terms of using `Twiddle` for parameter optimization, it seemed to be a long and time-consuming process. However, by starting with a set of parameters already tuned manually, I was able to speed-up the process. In addition, I realized that the twiddle algorithm at each step allows for a complete cycle of N iterations of the control system. Then compares the error for that cycle against the best error observed so far before tuning the parameter scale factor. I tweaked the algorithm and the `applyController()` function to remove all redundant iterations. I passed in the current `best_error` parameter to the `applyController()` function and at each iteration I compared the current `cumulative_error` of this cycle against the `best_error`. If the `cumulative_error` had already grown past the `best_error`, then I prematurely terminate the current cycle and allow twiddle to tune the parameters and the parameter scaling values. This allowed me get rid of many unnecessary iterations. The *integral gain* seemed to have no-effect and without the above workaround, the twiddle algorithm would be running a complete cycle of 100000 iterations for the *integral* term without any purpose.

Final Output

The controller is managed by the combined output of the PID systems of both the cart and pendulum. With the learned parameters with `Twiddle` the system reached the required equilibrium. I also tried to use the same twiddle based parameter search to attempt to balance the pendulum starting from the downward facing situation. The search went on for almost 12 hours. The resulting parameters could only balance the pendulum *(from the downward-facing position)* nondeterministically. It was failing almost all the time and was successful very randomly once in a while. Upon, further investigating I realized that the use of PID control alone will not suffice to achieve this task and a more complex energy shaping controller is required.