Project 1, Part 2: Auto tuning of PID controller

María Fernanda Robles Hernández
20141698

IFT 6521 Dynamic Programming
Winter 2019
Emma Frejinger

1 Description of the problem

A proportional-integral-derivative controller (PID), is a system that allows to regulate the error of a result by means of feedback, with the objective of assimilating a reference value. The PID control consists of three parameters that define its behavior: the proportional value (Kp), the integral value (Ti) and the derivative value (Td), coexisting as in (1).

$$G_c = K_p + \frac{K_i}{s} + K_d s \tag{1}$$

The action of obtaining the values of the constants Kp, Ki and Kd is called tuning. There are several methods to tune a system, including Ziegler-Nichols (maximum gain method) and Cohen-Coon. These methods, are sometimes considered as tedious and imprecise, that's why in this project, we are looking to optimize.

To exploit the performance of a PID controller, we will explore the results obtained when self tuned when modeled as a stochastic sequential decision problem.

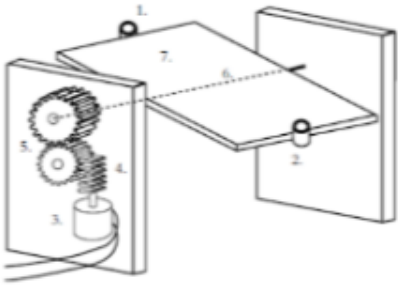

Figure 1: Track system

For example, in the Fig.1, we identify a system that needs to control its position, this system was analysed and roughly approximated to (2), this approximation is useful for simulations. Now, we can naively simulate the control of the system (Fig. 3) and stimulate it to obtain its reactions, the software used for this was Simulink from Matlab.

$$G_c = \frac{0.2608}{s^2 + 1.281s + 0.2615} \tag{2}$$

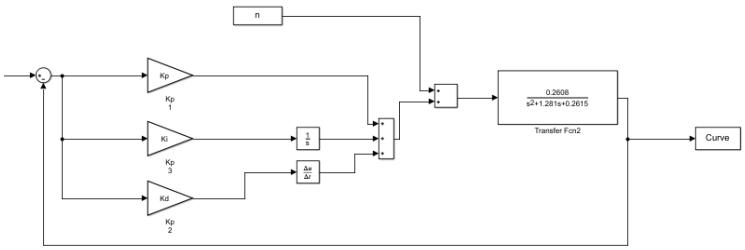

Figure 2: Control system

2 Mathematical formulation

2.1 Approximate method

For the approximate method we will use a rollout algorithm.

The states x_k are composed by the gains K_i, K_p and K_d, and the $curve$ resulting from their simulation. We can also define the system by its reference signal Va, and a general terminal cost approximation ($Tapprox$) denoted by (3).

$$Tapprox = \frac{Va}{2.9} \tag{3}$$

Our function to minimize is the expected value of the difference between the simulated $curve$ and Va at every state k ($Gdiff$), as seen in (4).

$$g_k(x_k, u_k) = \mathbf{E}[|Curve_k - Va|] \tag{4}$$

The control process will be defined by increasing one of this gains by a $step$ whose distribution is denoted by $U(0, 10)$. A second gain will be increased following a look-ahead policy, meaning that a mixure of two of the gains behavior of $Gdiff$ will be analyzed, being (5).

$$\tilde{J}_k = Gdiff_k \tag{5}$$

The $Gdiff$ will be a value constantly compared to find its optimal magnitude, if it's a minimum value, we approve the path of gain increment so far, if $Gdiff$ didn't improved, that gain increment will be discarded, being u_k and $x_{k+1} = f(x_k, u_k)$ defined.

Algorithm 1: Approximate method

```
Result: Gdiff, Ki, Kp, Kd
Ki, Kp, Kd, Va, Tapprox;
while While Tapprox > Gdiff do
    Increment gains sequentially, simulate and obtain Gdiff_k;
    if Gdiff_k < min(Gdiff) then
        Update gains
    else
        Discard gains
    end
end
```

2.2 Exact method

The exact method is a back-chain algorithm, modeled in a very similar way to the approximate method.

We use the same concepts of x_k, but g_k and J_k wont be approximate, which means, the obtained values will be absolutes.

The actions will be defined by increasing one of this gains by a $step$ whose distribution is denoted by $U(0, 10)$.

The $Gdiff$ will be a value constantly compared to find its optimal magnitude, if it's a minimum value, we approve the path of gain increment so far, if $Gdiff$ didn't improved, that gain increment will be discarded, being u_k and $x_{k+1} = f(x_k, u_k)$ defined.

3 Results and analysis

3.1 Exact method

This program takes an average 3.54 seconds to find results for the PID controller.

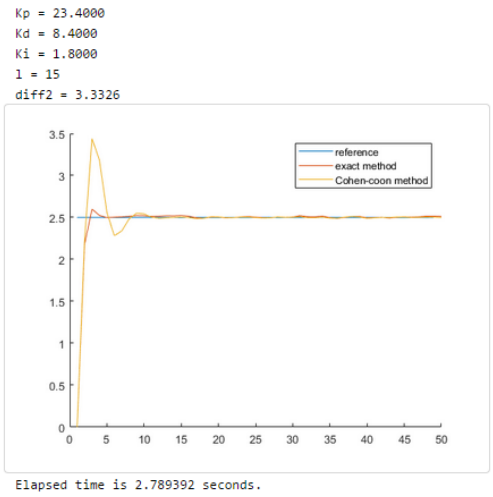

Figure 3: Response to the signal with different solution methods

As we can see in the Fig.3, the exact method presents better results that the classic Cohen-coon method. This results, do a faster path from the initial state to the reference value, considering also that the curve doesn't presents big peaks in the transition, showing graphically that it's good performance.

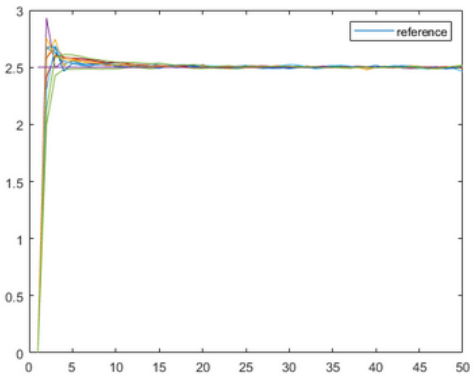

Figure 4: Variance in the exact method solutions

In the Fig.4 we can see how the exact method responds to the stochastic problem, presenting some variance in the results, but still keeping the best solution possible under the boundaries of $Tapprox$. We can also see, that the curves keep their shape, making this algorithm reliable.

3.2 Approximate method

This program takes an average of 5.56 seconds to find results for the PID controller. This algorithm takes more time to present an answer due to the fact that it needs more simulations for the look-ahead method.

In the Fig.5 we can see the results of one of the experiences. Here we note a better performance than the classic method of Cohen-coon, and similar to the exact method. But we can identify, that still the exact method had a more alike final $curve$ to the reference, in Table 1 we can see the comparative of their averages.

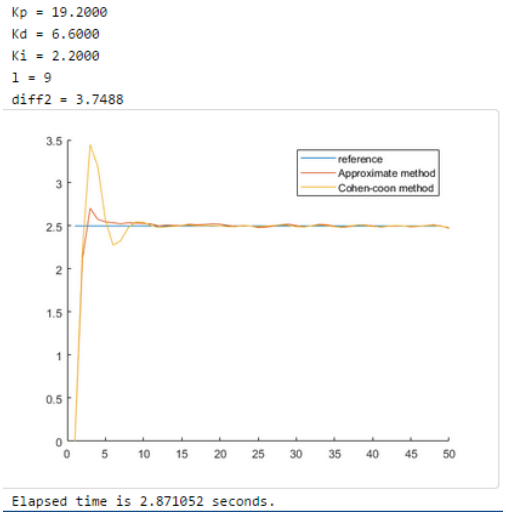

Figure 5: Response to the signal with different solution methods

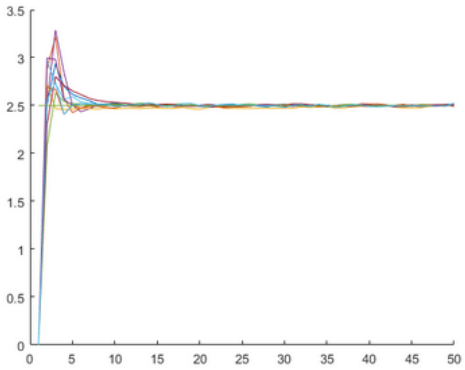

Figure 6: Variance in the approximate method solutions

In the Fig.6 we can see the results of multiple experiences. Here we note a relative big variance between the results, but still we can identify a tendency of shape and a successful reach of the boundary $Tapprox$.

The approximate method was faster to get closer to the boundary $Tapprox$ as seinf in the Table 1, but it would find a hard time by reaching it. I think it's because the behavior of the PID system and the transfer function (1) don't have

method	Gdiff	numer of steps
Exact	3.6	20.7
Approximate	4.8	16

Table 1: Average difference between $curve$ and reference value

a linear relationship to the change of the gains, so when the look-ahead method would took place, we would loose precision in the tuning.

4 Work process

Worked with the students Morris Fung and Ismael Martinez.

4.0.1 How to run the code:

1- Open the files "Approx2", "Exactmethod2", "Simulation".
2- Be sure that we are working on "Current folder" with the Proyect2 folder.
3- Run.

References

[1] Dimitri P. Bertsekas. (August 2010). Rollout Algorithms for Discrete Optimization: A Survey. Handbook of Combinatorial Optimization, Springer, 2013, 20. 4/8/2019, From MIT data base.

[2] Doerr et al. (8 Mar 2017). Model-Based Policy Search for Automatic Tuning of Multivariate PID Controllers. IEEE., 2017, 7. 4/08/2019, From arxiv data base.