DiffLoop: Tuning PID Controllers by Differentiating Through the Feedback Loop

Athindran Ramesh Kumar and Peter J. Ramadge

Princeton University
CISS 2021
arkumar@princeton.edu

February 25, 2021

PID controllers and wind-up compensation

PID tuning with anti-windup

- PID controllers[1] popular in industrial control, robotics
- Tuning PID parameters crucial
- Major source of non-linearity - actuator saturation
- Anti-windup for actuator saturation - **back-calculation**

Our work

- Focus on model-based tuning - both system and actuator models
- **Key idea** - solve the non-convex optimization with gradient descent
- Enabled by **automatic differentiation**

[1] Åström and Hägglund 1995.

Outline of our approach

1. Run simulation with current parameters
2. Compute cost function
3. Propagate gradients back through the models of actuator and system
4. Update parameters with gradient update
5. Repeat until convergence

AutoDiff tool - PyTorch[2]
Computation easily done in a modern CPU

[2] Paszke et al. 2017.

Related Work

Prior Work

Machine learning and PID tuning

- Black-box optimization - Genetic algorithm[3], Particle swarm optimization[4]
- Reinforcement learning[5]

Differentiable models

- Differentiate through to update model parameters or train controllers
- Success in various domains[6]

Ours - **Model-based** PID tuning with differentiable model

[3] Mitsukura, Yamamoto, and Kaneda 1997; Herrero et al. 2002.
[4] Chen 2007.
[5] Doerr et al. 2017; Lawrence et al. 2020; Shi et al. 2018.
[6] Chang et al. 2016; Degrave, Hermans, Dambre, et al. 2019; Avila Belbute-Peres et al. 2018.

Theoretical Standpoints

Non-convex optimization in control

- LQR, H^∞ controller design - policy gradient and gradient descent converge to global optima[7]
- Output feedback controller design less studied

Disturbance-feedback policies

- Introduced in online learning approach to control[8]
- Tight regret bounds

[7] Fazel et al. 2018; Zhang, Hu, and Basar 2020.

[8] Agarwal et al. 2019; Hazan, Kakade, and Singh 2020; Simchowitz, Singh, and Hazan 2020.

Disturbance Feedback for Anti-Windup Compensation

System Setup

Assume the system to be controlled is stabilizable and detectable

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{1}$$
$$y_t = Cx_t + e_t. \tag{2}$$

To model actuator saturation, modify (1) to:

$$x_{t+1} = Ax_t + B\text{sat}(u_t) + w_t \tag{3}$$

Back-calculation - The errors due to actuator saturation integrated and fed back[9]

[9] Åström and Murray 2010.

Back-calculation method

r_t - the reference signal to be tracked
P_t, I_t, D_t - **proportional**, **integral** and **derivative** components

$$P_t = k_p\,(r_t - y_t) \tag{4}$$
$$D_t = \alpha D_{t-1} + k_d \Delta y_t \tag{5}$$
$$I_{t+1} = I_t + k_i\,(r_t - y_t) + b(\mathrm{sat}(v_t) - v_t) \tag{6}$$
$$v_t = P_t + I_t + D_t \tag{7}$$
$$\mathrm{sat}(v_t) = \mathrm{clamp}(v_t, u_{\mathrm{low}}, u_{\mathrm{high}}). \tag{8}$$

Δ - difference operator, α - filter parameter
k_p, k_i, k_d, b - proportional, integral, derivative, back-calculation gains

Disturbance feedback and back-calculation

Start from linear state-space model with PID control
Append integral, derivative terms to state

$$i_{t+1} = \sum_{t'=1}^{t+1} x_{t'} = i_t + x_t \tag{9}$$

$$d_{t+1} = x_t - x_{t-1}. \tag{10}$$

Augmented state $X_t = [x_t; x_{t-1}; i_t]$

$$\begin{bmatrix} x_{t+1} \\ x_t \\ i_{t+1} \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ I & 0 & 0 \\ I & 0 & I \end{bmatrix} \begin{bmatrix} x_t \\ x_{t-1} \\ i_t \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u_t + w_t \tag{11}$$

$$Y_t = \begin{bmatrix} C & 0 & 0 \\ 0 & 0 & C \\ C & -C & 0 \end{bmatrix} \begin{bmatrix} x_t \\ x_{t-1} \\ i_t \end{bmatrix} + e_t. \tag{12}$$

PID controller design

Write equations more concisely as:

$$X_{t+1} = A^{'} X_t + B^{'} u_t + w_t \tag{13}$$

$$Y_t = C^{'} X_t + e_t, \tag{14}$$

where w_t and e_t defined appropriately

- Augmented system stabilizable, detectable
- PID controllers ($\alpha = 0$) expressed as $u_t = -KY_t$ for (13), (14)
- PID tuning - output feedback controller design (Open problem)

Actuator saturation as a disturbance

$w_t^a \triangleq B'(\text{sat}(u_t) - u_t)$ - denote the saturation error
Treat the saturation error as a disturbance:

$$X_{t+1} = A^{'} X_t + B^{'} \text{sat}(u_t) + w_t \tag{15}$$

$$= A^{'} X_t + B^{'} u_t + w_t^a + w_t. \tag{16}$$

Adversarial disturbances in online learning - use disturbance feedback[10]:

$$u = -KX_t - \sum_{l=1}^{h} K_d^{[l]} w_{t-l}. \tag{17}$$

Key Idea - if h is length of the simulation horizon and $K_d^{[l]} = K_d$ for all l, reduces to the **back-calculation method**

[10] Agarwal et al. 2019.

Disturbance feedback in episodic learning

Our work - focus on an episodic setting.
Introduce disturbance dynamics w_t^a:

$$w_t^a = \sum_{i=1}^{h} M^{[i]} w_{t-i}^a. \tag{18}$$

Augment the state further

$$Z_t = [X_t; w_t^a; w_{t-1}^a; w_{t-2}^a \ldots ; w_{t-h}^a].$$

Model disturbance to obtain disturbance feedback policies

$$Z_{t+1} = \begin{bmatrix} A' & I & 0 & 0 \\ 0 & M^{[1]} & M^{[2:h-1]} & M^{[h]} \\ 0 & I & I & 0 \end{bmatrix} Z_t + \begin{bmatrix} B' \\ 0 \\ 0 \end{bmatrix} u_t + w_t^r$$

$$Y_t^z = \begin{bmatrix} C' & 0 \\ 0 & I \end{bmatrix} Z_t + e_t^r$$

Optimization for controller tuning

The class of output-feedback controllers $u_t = -KY_t^z$:

$$\begin{aligned} u_t &= -K_c Y_t - K_d^{'} w_{t:t-h}^a \\ &= -K_c Y_t - K_d^{'} \left[\; M^{[1:h]}; \quad I \; \right] w_{t-1:t-h}^a \\ &= -K_c Y_t - K_d w_{t-1:t-h}^a. \end{aligned}$$

Encompasses disturbance-feedback and the back-calculation method
Tune K_c and K_d, gradient descent with:

$$\min_{K_c, K_d} \sum_{t=1}^{T} y_t^T Q y_t + u_t^T R u_t. \tag{19}$$

Numerical Simulations

Simulation Setup

Run simulations on linear systems with saturation

	Plant	Actuator limits	Step reference	Initial feedback gains
1	$P(s) = \frac{2e^{-0.02s}}{s-0.995}$	± 3.3	± 4	$k_p = 4,\ k_i = 10,$ $b = 0.5$
2	$P(s) = \frac{1}{(s+0.1)(s-0.1)}$	± 3.0	± 2.9	$k_p = 20,\ k_i = 2,$ $k_d = 5,\ b = 1$
3	$P(s) = \frac{(s+0.5)(s+0.3)}{(s+0.1)(s+0.2)(s+0.4)(s+0.6)}$	± 4.0	± 3	$k_p = 20,\ k_i = 8,$ $k_d = 10,\ b = 0.2$

Simulation setup

- Convert the continuous-time system into its discrete-time counterpart
- Generate 30 random reference signals - 20 for training, 10 for testing

Benchmark four controllers:

- Controller 1 - Initial PID controller
- Controller 2 - Initial PID controller with back-calculation
- Controller 3 - Optimized PID+back-calculation
- Controller 4 - Dynamic neural network controller

System 1 - $P(s) = \frac{2e^{-0.02s}}{s-0.995}$

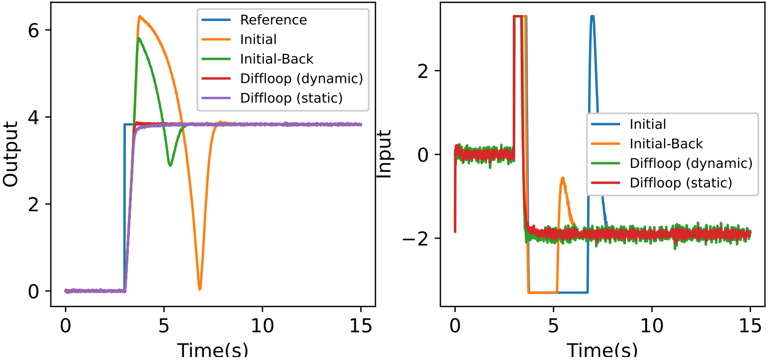


Figure: Performance of the four controllers on a test reference for system 1.

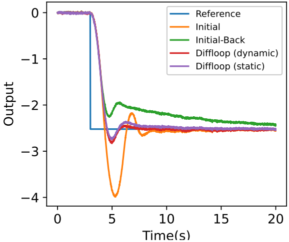
System 1

Squared error cost of the four controllers on system 1

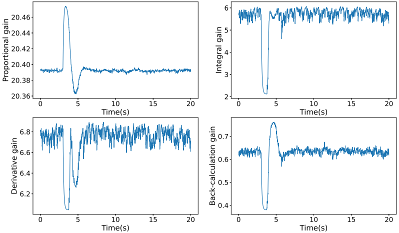
Method	Training cost	Test cost
Initial PI	304.4 ± 432.3	349.0 ± 503.3
Initial PI with backcalculation	178.2 ± 153.0	189.0 ± 164.8
PI+backcalculation optimized	110.2 ± 79.8	114.7 ± 82.3
Neural Net optimized	109.5 ± 79.9	114.0 ± 82.2

Optimized controllers dont show wind-up transients

System 2-$P(s) = \frac{1}{(s+0.1)(s-0.1)}$



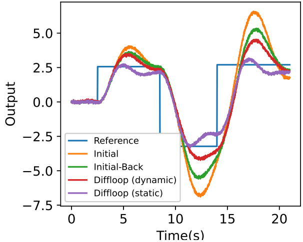
(a) System 2



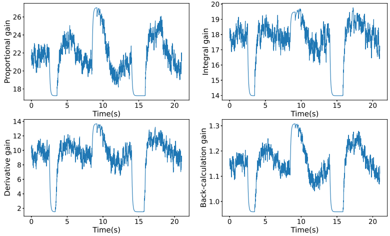
(b) System 2 feedback gains

Figure: (a)Output of the four controllers on a step input for system 2. (b) Variation of the feedback gains with time for the Dynamic PID controller.

System 3 - $P(s) = \frac{(s+0.5)(s+0.3)}{(s+0.1)(s+0.2)(s+0.4)(s+0.6)}$



(a) System 3



(b) System 3 feedback gains

Figure: (a)Output of the four controllers on a step input for system 3. (b) Variation of the feedback gains with time for the Dynamic PID controller.

Conclusions and Future Work

Summary

- Tuning PID using AutoDiff - simple and effective
- Relationship between disturbance feedback and back-calculation
- PID tuning as output feedback controller design

Future work

- Theoretical convergence properties
- PID tuning for non-linear robotics