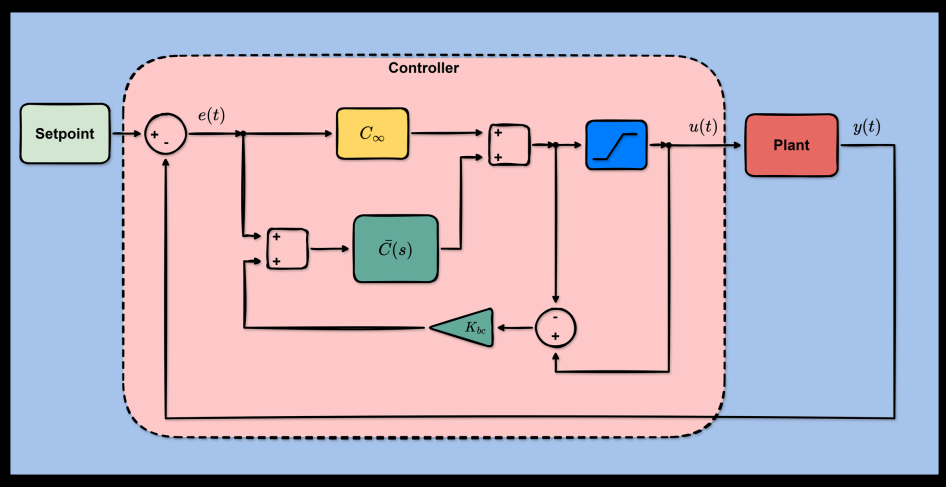
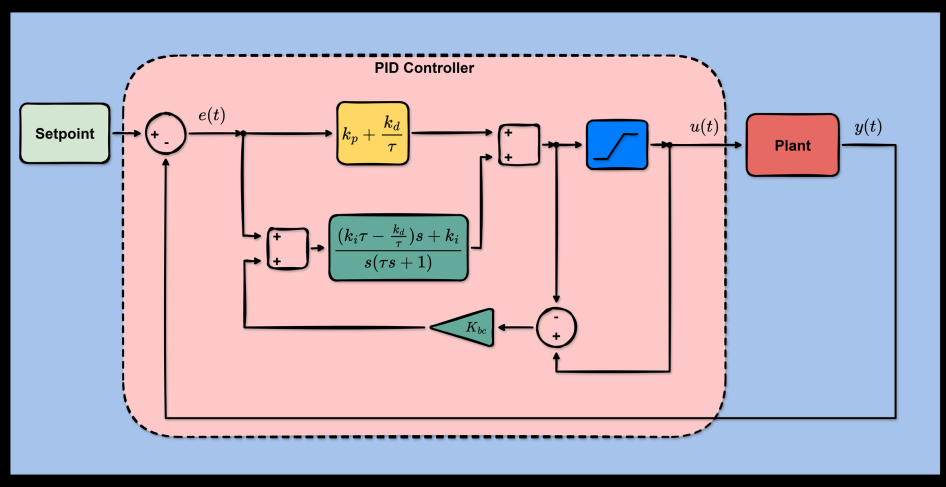
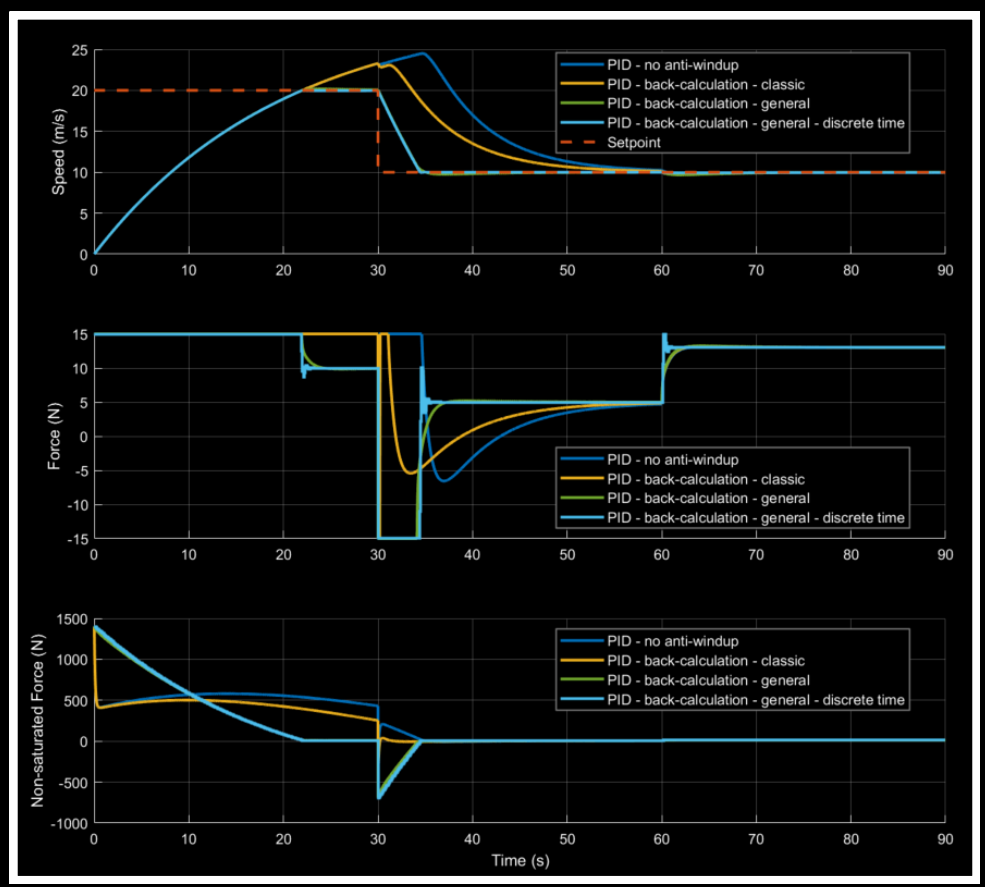
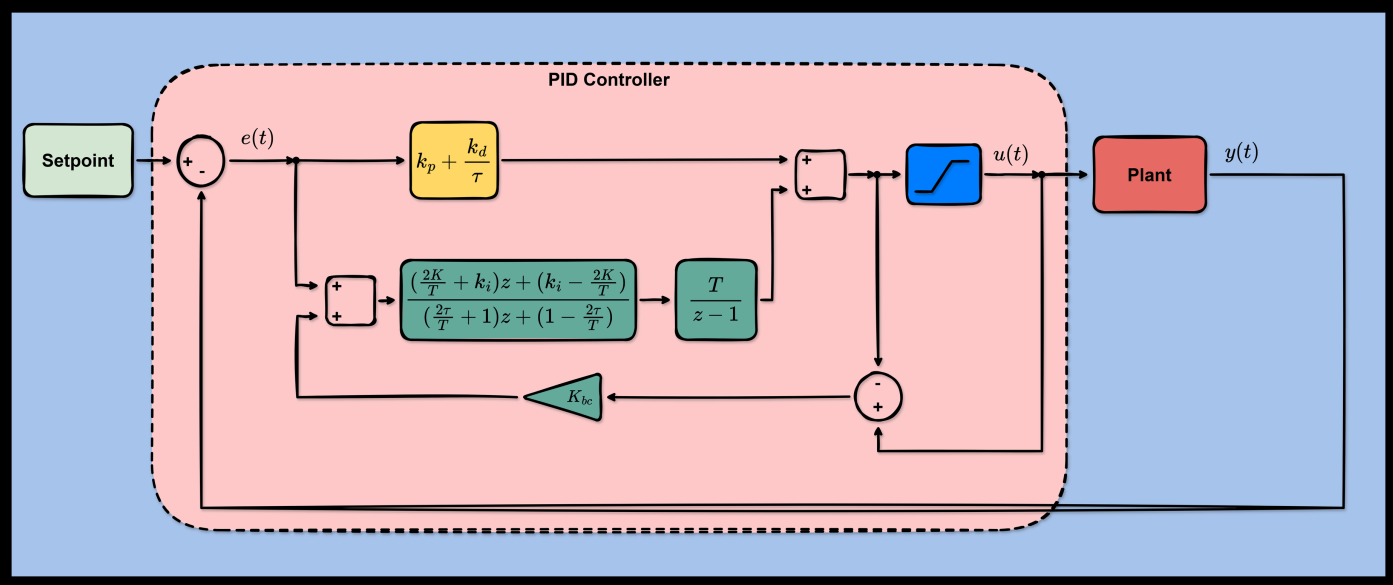
PID Anti-Windup General Back-Calculation Discretisation









Code / Model

https://github.com/simorxb/back-calculation

The Problem: Integral Windup

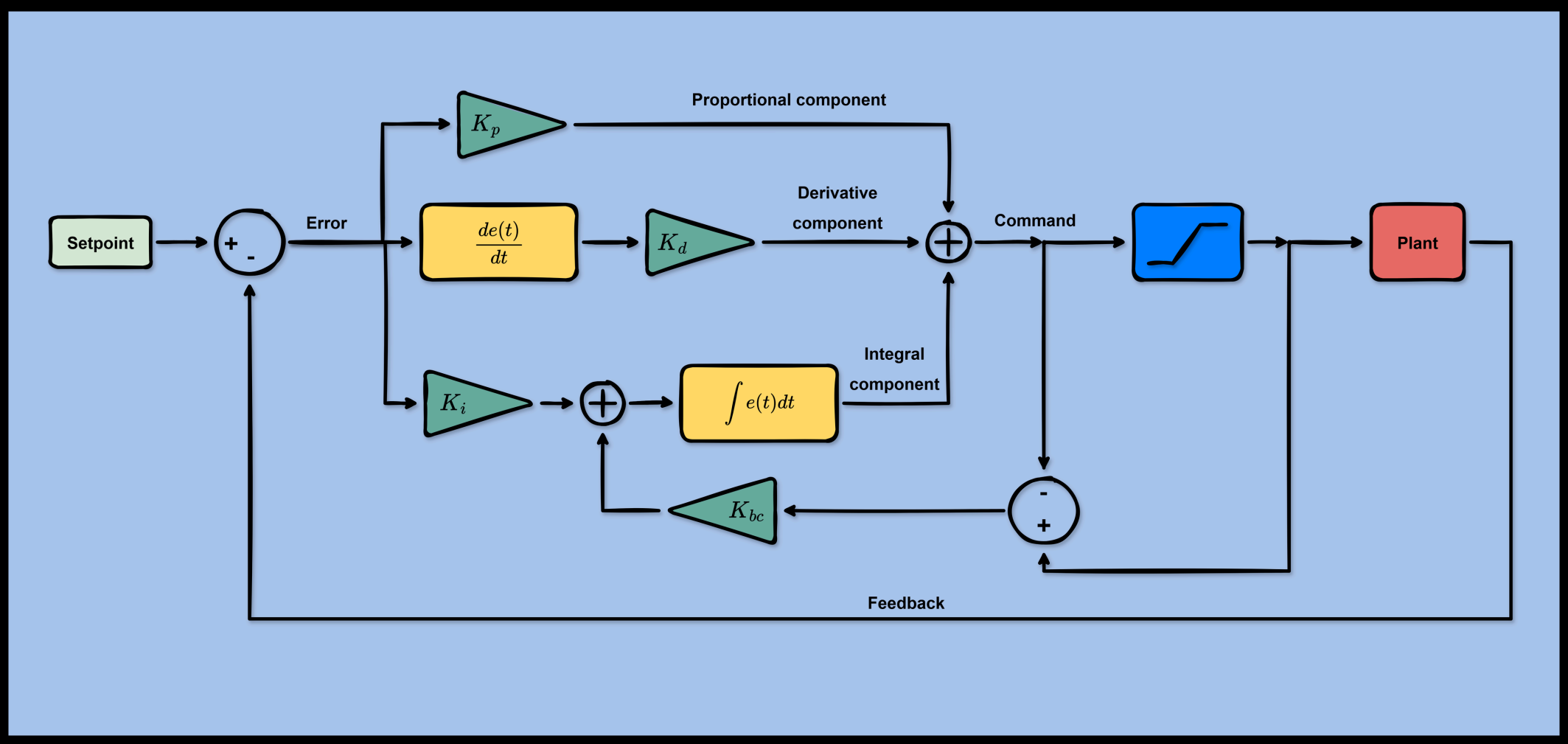
Integral windup is a phenomenon that happens when the actuator is limited in amplitude and slew rate. In these cases the integrator (if present in the control loop) can build up to large values, causing unwanted behaviours.

When the input exits the limited condition, the large value will take some time to recover and cause a large transient.

Back Calculation Classic Implementation

The classic back calculation anti-windup implementation adds a second input to the integrator in the PID, equal to the difference between the saturated command and the unsaturated command multiplied by a gain K_{bc}.

In the case of upper saturation, the more the command increases, the more negative the second input becomes, resulting in a decrease in the integral. The same mechanism, reversed, is true for a lower saturation when the command decreases.



Back Calculation General Implementation

The classic implementation is incorrect if the controller has poles in addition to the integrator (i.e., a pole at the origin).

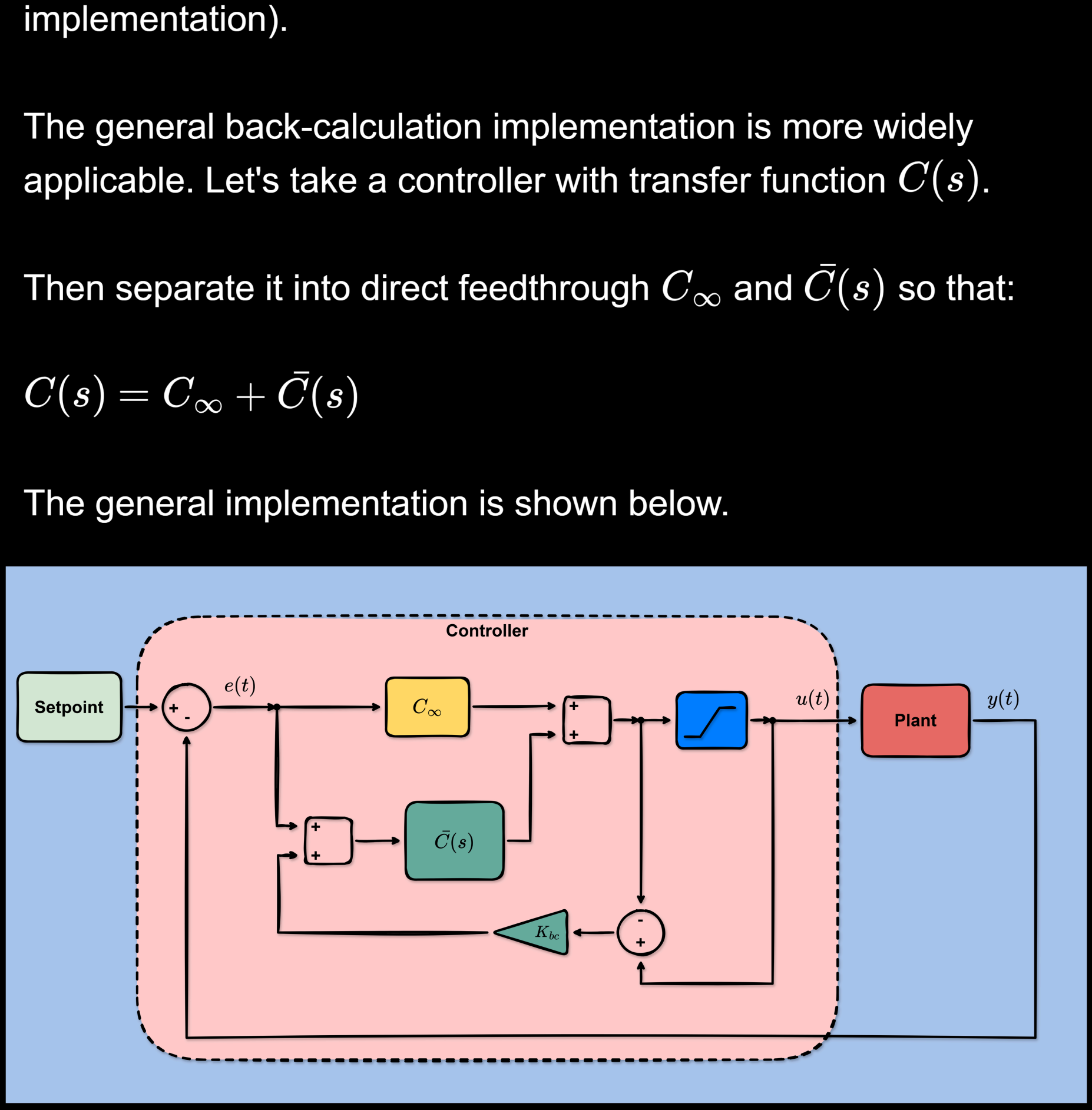
For example, the classic back-calculation implementation is incorrect for the PID with filtered derivative (a very common PID implementation).

The general back-calculation implementation is more widely applicable. Let's take a controller with transfer function $C(s)$.

Then separate it into direct feedthrough C_∞ and $\bar{C}(s)$ so that:

$$C(s) = C_\infty + \bar{C}(s)$$

The general implementation is shown below.



Back Calculation General Implementation PID

Considering the PID controller, we have:

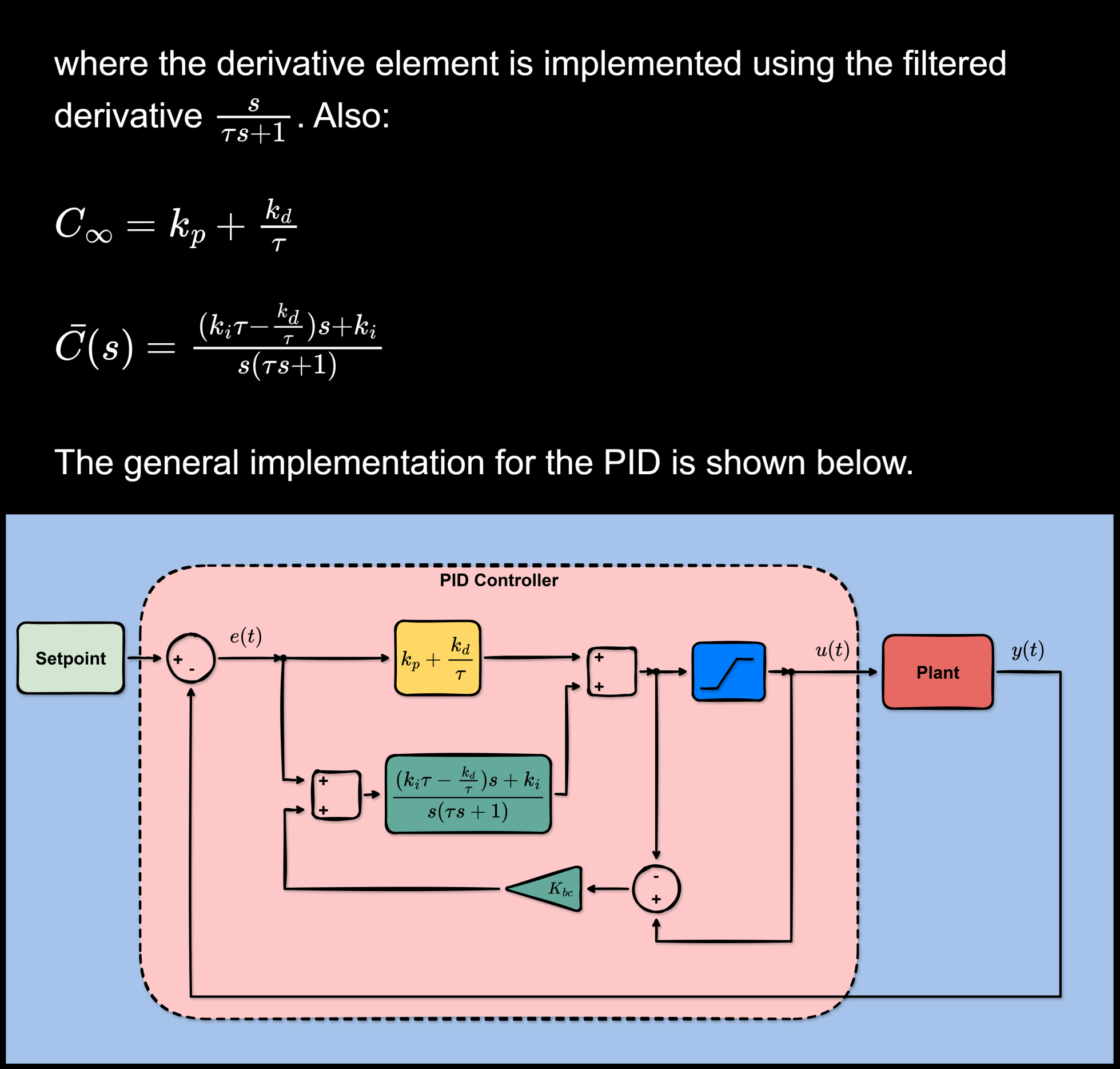
$$C(s) = k_p + \frac{k_i}{s} + k_d \frac{s}{\tau s+1} = \frac{s^2(k_p\tau+k_d)+s(k_i\tau+k_p)+k_i}{s^2\tau+s}.$$

where the derivative element is implemented using the filtered derivative $\frac{s}{\tau s+1}$. Also:

$$C_\infty = k_p + \frac{k_d}{\tau}$$

$$\bar{C}(s) = \frac{(k_i\tau-\frac{k_d}{\tau})s+k_i}{s(\tau s+1)}$$

The general implementation for the PID is shown below.



Back Calculation General Implementation PID – Discretisation

For convenience, we define:

$$K = k_i\tau + \frac{k_d}{\tau}$$

Now we have:

$$\bar{C}(s) = \frac{Ks+k_i}{s(\tau s+1)}$$

$\bar{C}(s) = \bar{C}_1(s)\bar{C}_2(s)$, with $\bar{C}_1(s) = \frac{Ks+k_i}{\tau s+1}$ and $\bar{C}_2(s) = \frac{1}{s}$

Using Tustin transformation ($s \leftarrow \frac{2}{T}\frac{z-1}{z+1}$) we have:

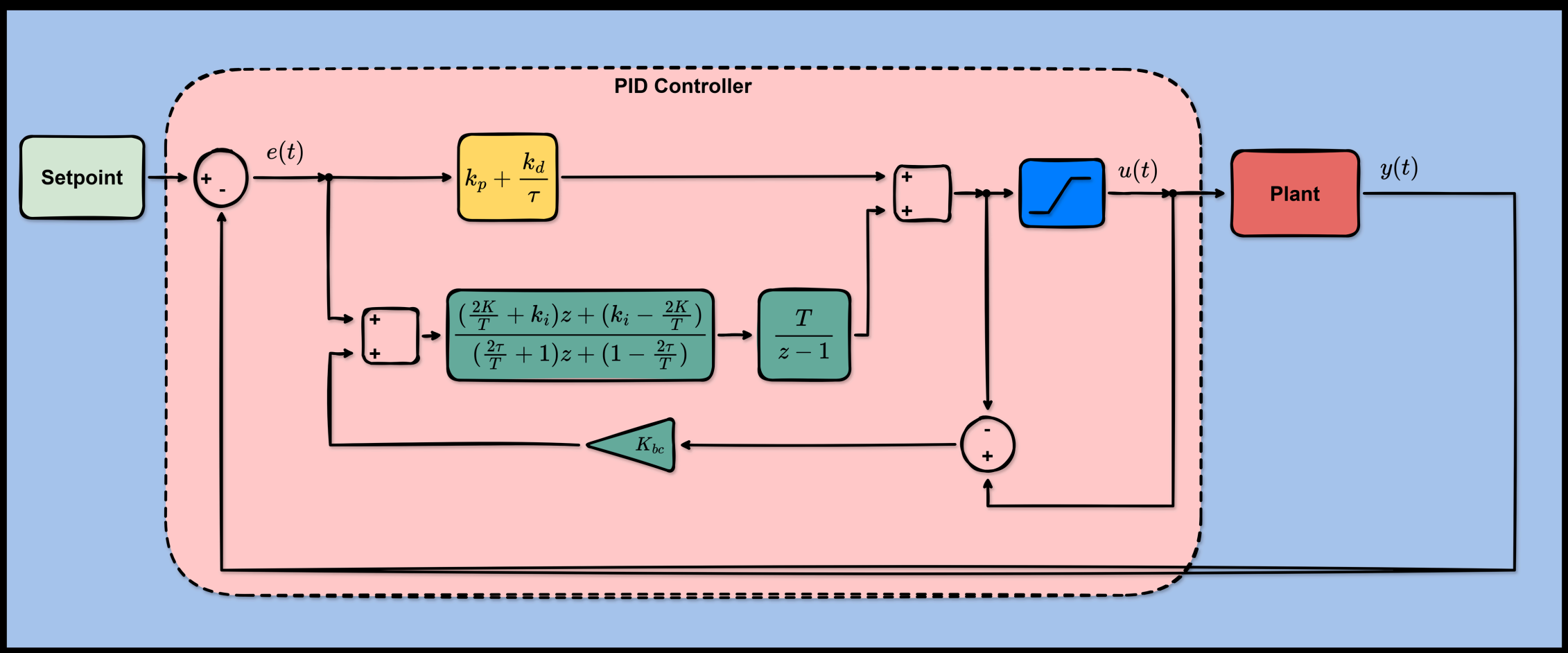
$$C_1(z) = \frac{(\frac{2K}{T}+k_i)z+(k_i-\frac{2K}{T})}{(\frac{2\tau}{T}+1)z+(1-\frac{2\tau}{T})}$$

while for $C_2(s)$ we use forward Euler to avoid algebraic loops:

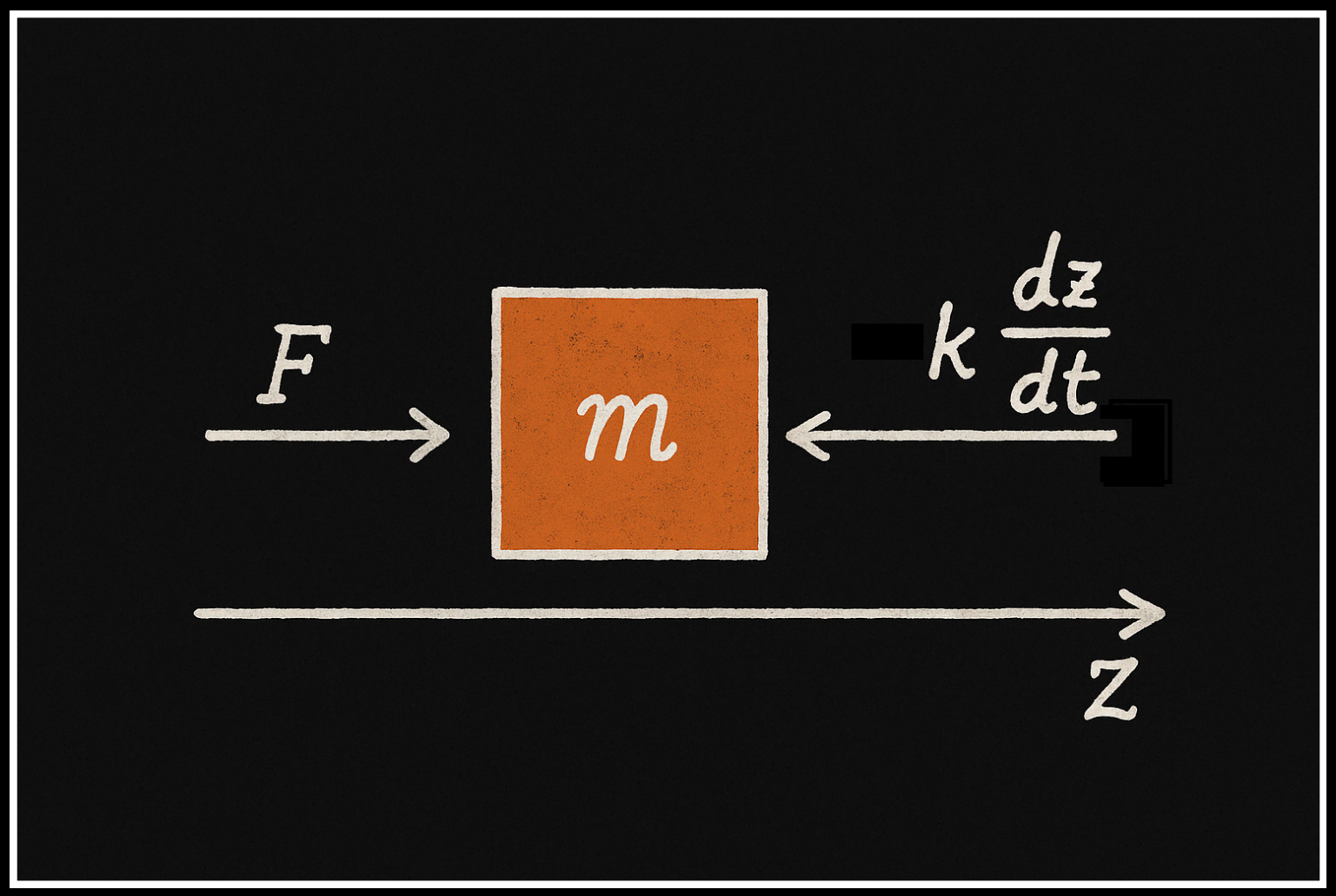
$$C_2(z) = \frac{T}{z-1}.$$

And we are ready for the discretised general implementation for the PID.

Back Calculation General Implementation PID – Discretisation



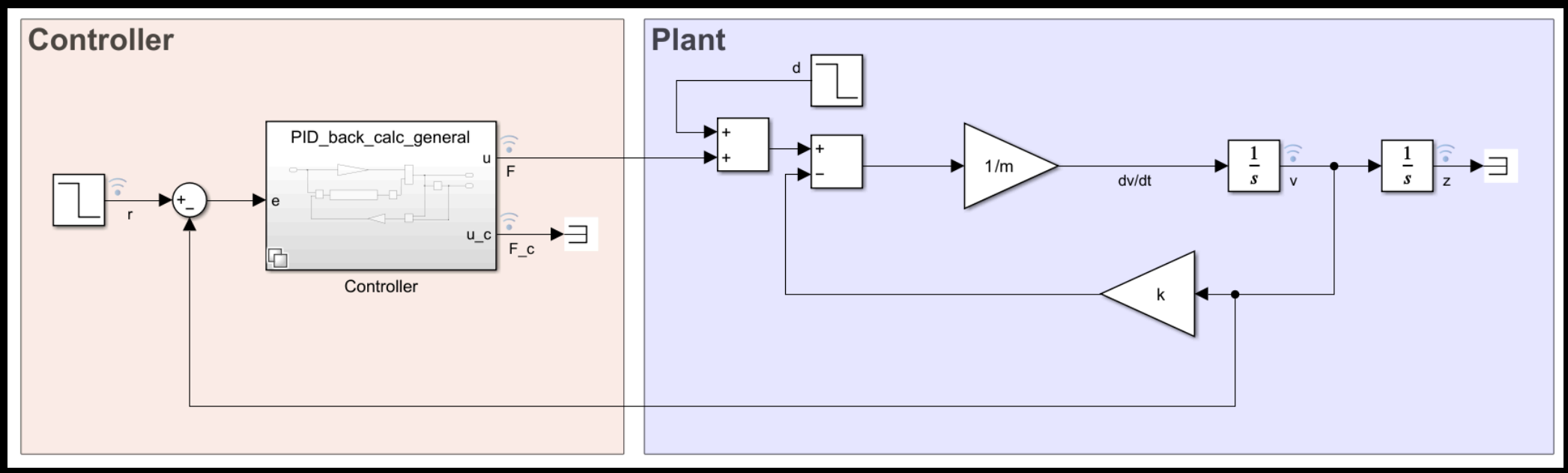
Plant



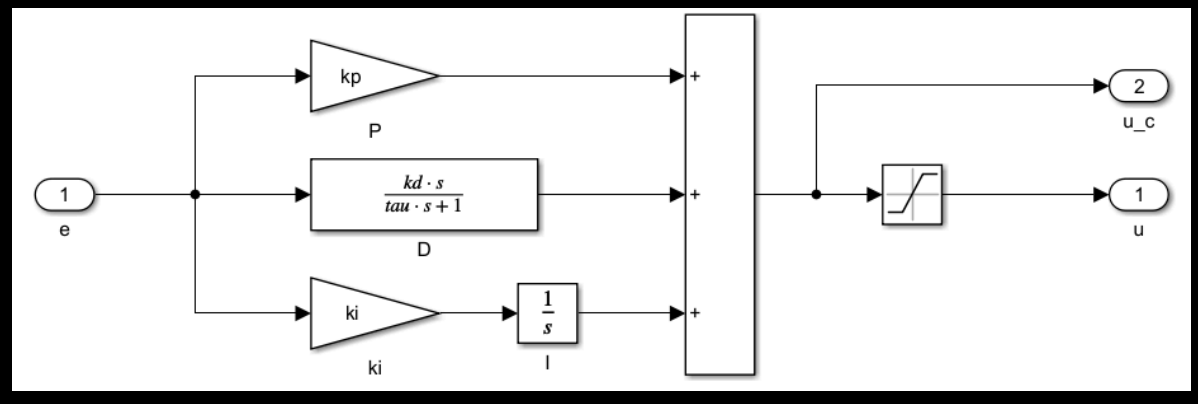
$$m\frac{d^2z(t)}{dt^2} = F - k\frac{dz(t)}{dt}$$

$$m = 10kg$$
$$k = 0.5\frac{Ns}{m}$$

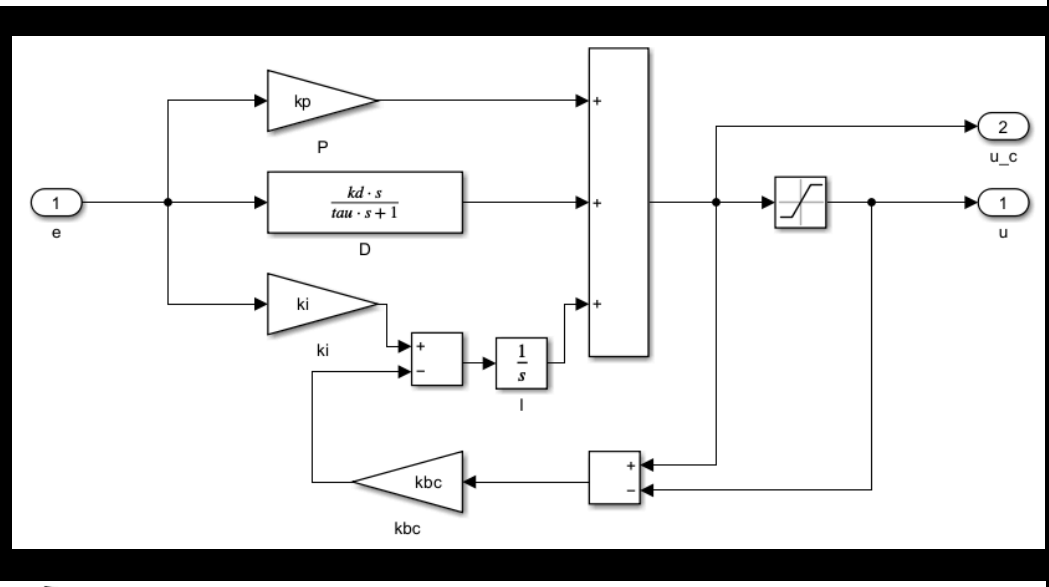
Simulink



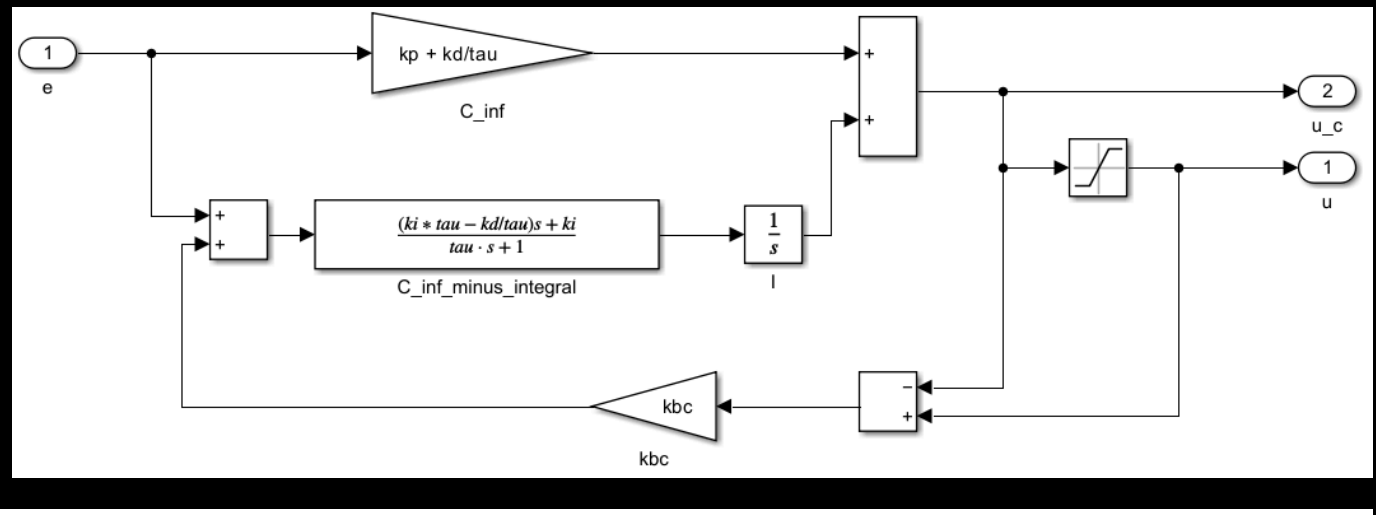
PID
No Anti-windup



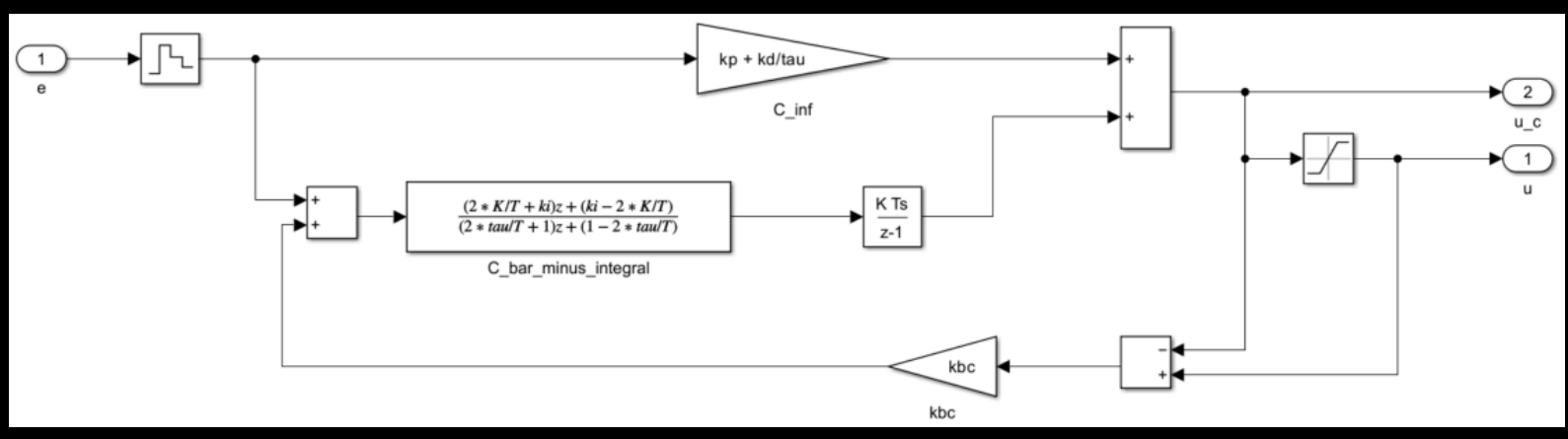
PID
Back Calculation



PID
General
Back Calculation



PID
General
Back Calculation
Discretisation



MATLAB Code Initialisation

```matlab
%% Plant

% System parameters
m = 10;
k = 0.5;

% Controller parameters
kp = 20;
ki = 3;
kd = 5;
tau = 0.1;
kbc = 1/(kp+kd/tau);
F_max = 15;
F_min = -15;
K = ki*tau + kd/tau;
T = 0.1;
```

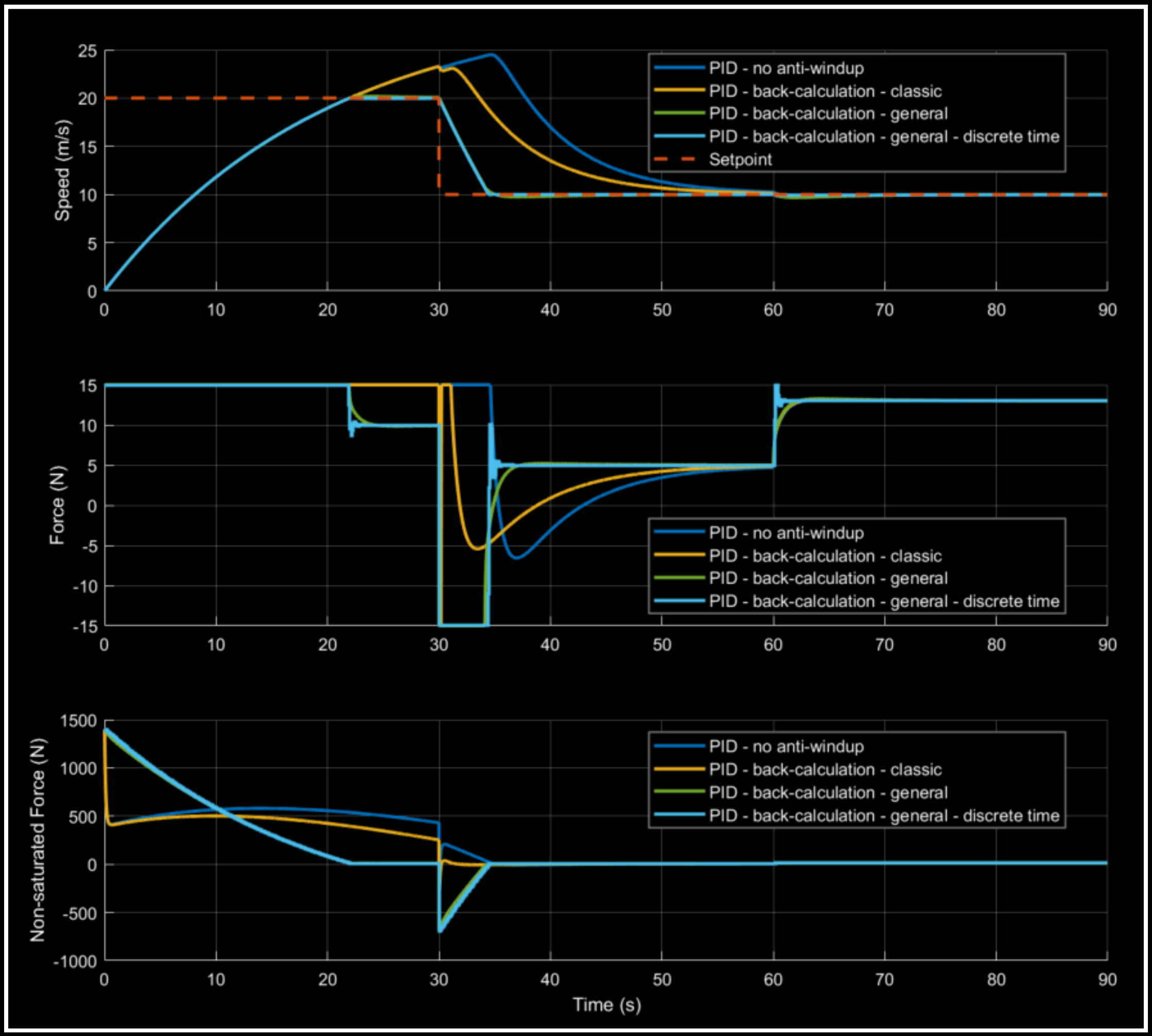
MATLAB Code
Run Simulations

```matlab
%% Run simulations

% Define configurations
configs = {
    struct('name', 'PID - no anti-windup', 'ctrl_type', 0, 'color', '#0072BD');
    struct('name', 'PID - back-calculation - classic', 'ctrl_type', 1, 'color', '#EDB120');
    struct('name', 'PID - back-calculation - general', 'ctrl_type', 3, 'color', '#77AC30');
    struct('name', 'PID - back-calculation - general - discrete time', 'ctrl_type', 4, 'color', '#4DBEEE')
};
results = struct([]);

% Loop through configurations
for i = 1:length(configs)
    ctrl_type = configs{i}.ctrl_type;

    % Run the model with the ith configuration
    simOut = sim("speed_control");

    % Access the signals from out.logsout
    results(i).r = simOut.logsout.get('r').Values;
    results(i).F = simOut.logsout.get('F').Values;
    results(i).F_c = simOut.logsout.get('F_c').Values;
    results(i).z = simOut.logsout.get('z').Values;
    results(i).v = simOut.logsout.get('v').Values;

end
```

MATLAB Code Plot Results

```matlab
%% Plot results
figure('Color', 'k');

% First subplot: Speed
ax1 = subplot(3,1,1);
hold on;
for i = 1:length(configs)
    plot(results(i).v.Time, results(i).v.Data, 'LineWidth', 2, 'DisplayName', configs{i}.name, "Color", configs{i}.color);
end
plot(results(1).r.Time, results(1).r.Data, '--', 'LineWidth', 2, 'DisplayName', 'Setpoint', "Color", "#D95319");
legend('TextColor', 'w', 'Color', 'k', 'EdgeColor', ...
    [0.5 0.5 0.5], 'LineWidth', 1, 'FontSize', 10, 'Location', 'best');
hold off;
grid on;
ylabel('Speed (m/s)');
ax1.Color = 'k';
ax1.GridColor = 'w';
ax1.GridAlpha = 0.3;
ax1.XColor = 'w';
ax1.YColor = 'w';

% Second subplot: Force (saturated)
ax2 = subplot(3,1,2);
hold on;
for i = 1:length(configs)
    if configs{i}.ctrl_type == 4
        stairs(results(i).F.Time, results(i).F.Data, 'LineWidth', 2, 'DisplayName', configs{i}.name, "Color", configs{i}.color);
    else
        plot(results(i).F.Time, results(i).F.Data, 'LineWidth', 2, 'DisplayName', configs{i}.name, "Color", configs{i}.color);
    end
end
legend('TextColor', 'w', 'Color', 'k', 'EdgeColor', ...
    [0.5 0.5 0.5], 'LineWidth', 1, 'FontSize', 10, 'Location', 'best');
hold off;
grid on;
ylabel('Force (N)');
ax2.Color = 'k';
ax2.GridColor = 'w';
ax2.GridAlpha = 0.3;
ax2.XColor = 'w';
ax2.YColor = 'w';

% Third subplot: F_c (non-saturated force)
ax3 = subplot(3,1,3);
hold on;
for i = 1:length(configs)
    if configs{i}.ctrl_type == 4
        stairs(results(i).F_c.Time, results(i).F_c.Data, 'LineWidth', 2, 'DisplayName', configs{i}.name, "Color", configs{i}.color);
    else
        plot(results(i).F_c.Time, results(i).F_c.Data, 'LineWidth', 2, 'DisplayName', configs{i}.name, "Color", configs{i}.color);
    end
end
legend('TextColor', 'w', 'Color', 'k', 'EdgeColor', ...
    [0.5 0.5 0.5], 'LineWidth', 1, 'FontSize', 10, 'Location', 'best');
hold off;
grid on;
ylabel('Non-saturated Force (N)');
xlabel('Time (s)');
ax3.Color = 'k';
ax3.GridColor = 'w';
ax3.GridAlpha = 0.3;
ax3.XColor = 'w';
ax3.YColor = 'w';

% Link x axes
linkaxes([ax1, ax2, ax3], 'x');
```

Simulation

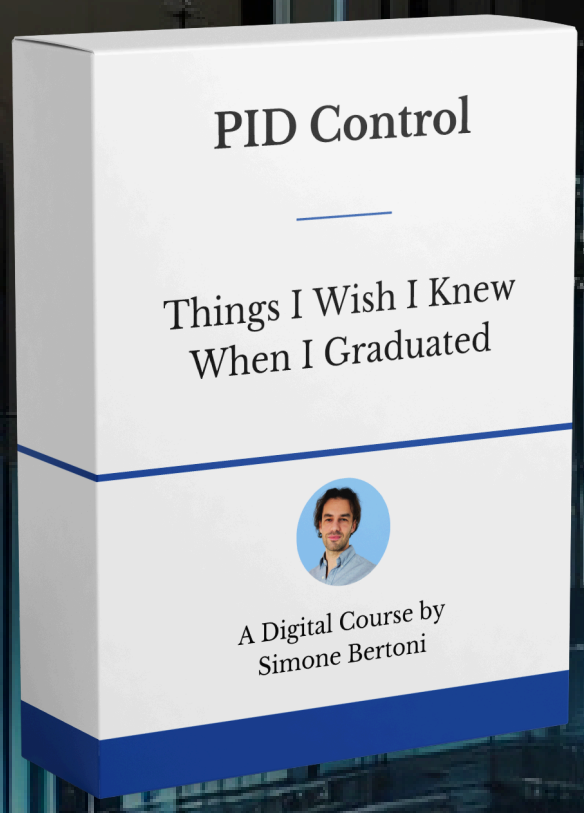


Note: a -8 N disturbance on the input force to the plant is applied at time 60 seconds.

PID Controller Course

https://simonebertonilab.com

SPECIAL DISCOUNT



15% OFF

LIMITED TIME

Overall, a great course. worth recommending.

★★★★★ *May 7, 2024*

The course is exciting and engaging, covers most of the concepts and all the explanations are clear and easy to understand. it would be great if there was more explanation on the calculations and recommendations for additional resources. Overall excellent course, and I got to learn a lot.

Vikram Kolupula ✔ Verified

★★★★★

Very helpful and practical

Yoav Golan

I enjoyed this course very much. I learned a lot of practical knowledge in a short time. Simone is very clear and teaches well, thank you! In the future, I would be very interested if Simone added a course with more subjects, such as cascading controllers, rate limiting, and how the controllers look in actual code. Thanks again!

★★★★★

Intuitive and Practical

Ranya Badawi

Simone's explanation of PID control was very intuitive. This is a great starter course to gain a fundamental understanding and some practical knowledge of PID controllers. I highly recommend it. For future topics, I'd be interested in frequency response, transfer functions, Bode plots (including phase/gain margin), Nyquist plots, and stability.

Understand the control theory

★★★★★ *April 28, 2024*

I think the most important thing is to understand the meaning behind the mathematical formula. I guess this is the mission of Simone in this course and from my point of view he fully achivied this target. I hope to see in the future other courses (e.g advanced controls) structered in the same way with the same passion and examples.

Thank you Simone. Show less

Emidio ✔ Verified

★★★★★

Very good sharing of experience

Romy Domingo Bompart Ballache

I have background in control system for power electronics, I see every lesson very useful.