PID Control and Auto-Tuning in the VentCon Pressure Control System

VENTREX

June 16, 2025

Abstract

This document describes the implementation of the PID (Proportional-Integral-Derivative) control system and auto-tuning procedures in the VentCon pressure control system. The document details the core control algorithm, anti-windup mechanisms, hysteresis compensation, and the relay-based auto-tuning method with multiple tuning rules for different performance requirements.

Contents

1 Introduction

The VentCon2 system is designed to provide precise pressure control using a solenoid valve operated via PWM (Pulse Width Modulation). The system employs a digital PID controller to maintain the desired pressure setpoint by dynamically adjusting the valve opening based on real-time pressure measurements. To simplify the tuning process, an automated self-tuning procedure based on the relay method is implemented.

2 PID Control Implementation

2.1 Basic PID Theory

The PID control algorithm calculates the control output $u(t)$ based on the error $e(t)$ between the desired setpoint and the measured process variable [7]:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{1}$$

Where:

- K_p is the proportional gain
- K_i is the integral gain
- K_d is the derivative gain
- $e(t) = \text{setpoint} - \text{measured value}$

2.2 Implementation in VentCon2

The VentCon2 system uses the PID_v2 library for Arduino which implements a discrete form of the PID equation. The controller is configured with the following parameters:

```
// PID initialization
PID pid(&pressureInput, &pwmOutput, &settings.setpoint,
        settings.Kp, settings.Ki, settings.Kd, DIRECT);
pid.SetMode(PID::Automatic);
pid.SetOutputLimits(0, pwm_max_value);
pid.SetSampleTime(SAMPLE_TIME); // 10ms (100Hz)
```

Listing 1: PID Configuration

The controller updates at a frequency of 100Hz, providing excellent responsiveness while maintaining processing efficiency. The PID output is then mapped to the valve's effective operating range (50% to 90% duty cycle), as the valve exhibits non-linear behavior at lower duty cycles.

2.3 Anti-Windup Implementation

The system implements an advanced anti-windup mechanism that prevents integral windup in two key situations:

1. When the valve is below its minimum effective operating range (50% duty cycle)
2. When the valve is above its maximum effective operating range (90% duty cycle)

The anti-windup mechanism works by temporarily switching the PID controller to manual mode and back to automatic when these conditions are detected, effectively resetting the integral component:

```
if (settings.antiWindup)
{
  float pidPercent = (pwmOutput / pwm_max_value) * 100.0;

  if ((pidPercent < VALVE_MIN_DUTY && pwmOutput >
    previousOutput) ||
      (pidPercent > VALVE_MAX_DUTY && pwmOutput >
    previousOutput))
  {
    // Reset the PID to prevent integral accumulation
    pid.SetMode(PID::Manual);
    pid.SetMode(PID::Automatic);
  }
}
```

Listing 2: Anti-Windup Implementation

2.4 Hysteresis Compensation

Solenoid valves typically exhibit hysteresis behavior, where the opening and closing characteristics differ. To compensate for this, the system implements a direction-based hysteresis compensation mechanism:

```
if (settings.hysteresis)
{
  bool currentlyIncreasing = pressureInput > lastPressure;

  if (currentlyIncreasing != pressureIncreasing)
  {
    if (!currentlyIncreasing)
    {
      // When changing direction from increasing to decreasing
      pidPercent += settings.hystAmount;
    }
    else
    {
      // When changing direction from decreasing to increasing
      pidPercent -= settings.hystAmount;
    }

    pressureIncreasing = currentlyIncreasing;
  }
}
```

Listing 3: Hysteresis Compensation

This approach helps overcome the "stiction" effect in solenoid valves, providing more consistent control regardless of whether the pressure is increasing or decreasing.

3 Auto-Tuning Procedure

3.1 Relay Method Theory

The auto-tuning implementation uses the relay method developed by Åström and Hägglund [1], which is a variation of the Ziegler-Nichols frequency response method [2]. This approach is widely used in industrial applications due to its reliability and ease of implementation [8]. Instead of increasing the proportional gain until sustained oscillation occurs (which can be difficult and potentially unsafe), the relay method:

1. Forces the system into controlled oscillation using a relay (on/off) controller
2. Measures the resulting oscillation amplitude and period
3. Calculates the "ultimate gain" (K_u) and "ultimate period" (T_u)
4. Applies tuning rules to determine the optimal PID parameters

The ultimate gain is calculated as:

$$K_u = \frac{4d}{\pi a} \tag{2}$$

Where:

- d is the relay amplitude (AUTOTUNE_RELAY_HIGH = 90%)
- a is the resulting oscillation amplitude in the process variable

3.2 Relay Method Visualization

Figure 1 illustrates the relay method auto-tuning process, showing how the relay output forces controlled oscillation in the pressure system and how the key parameters are measured.

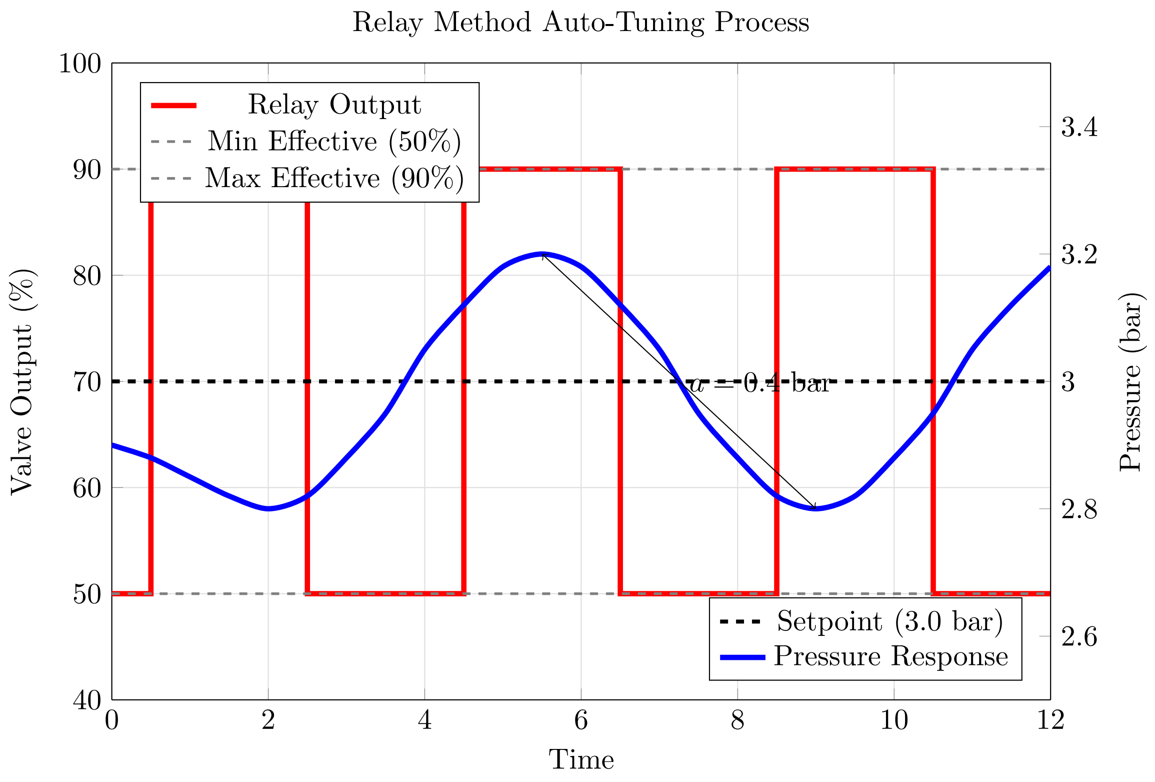

Figure 1: Relay method auto-tuning illustration showing the square wave relay output (red) switching between the valve's effective operating range of 50-90% and the resulting sinusoidal pressure response (blue). The ultimate period T_u and oscillation amplitude a are measured to calculate the PID parameters.

Key observations from the relay method:

- **Relay Output**: Switches between 50% (minimum effective opening) and 90% (maximum effective opening) based on pressure relative to setpoint

- **Pressure Response**: Shows sinusoidal oscillation around the setpoint with phase lag relative to the valve output
- **Ultimate Period** (T_u): Measured as the time between consecutive transitions (e.g., 4.0 seconds in the example)
- **Oscillation Amplitude** (a): Peak-to-peak pressure variation (e.g., 0.4 bar in the example)
- **Phase Relationship**: The pressure response lags behind the valve output due to system dynamics

The ultimate gain is then calculated as: $K_u = \frac{4\times(90\%-50\%)}{\pi\times 0.4 \text{ bar}} = \frac{4\times 40\%}{\pi\times 0.4} = \frac{160}{1.257} = 127.3$

3.3 Implementation in VentCon2

The auto-tuning process in VentCon2 works as follows:

1. The system sets the target pressure to a predefined setpoint (3.0 bar)
2. A relay controller alternates the valve between:
   - 50% (minimum effective opening) when pressure exceeds the setpoint
   - 90% of effective valve opening when pressure falls below setpoint - noise band
3. The system records the cycle times and amplitudes for a configurable number of cycles (default 10) or until timeout
4. The ultimate gain and period are calculated from the averaged measurements
5. PID parameters are calculated using the selected tuning rule

An important implementation detail is the handling of the valve's nonlinear behavior. The relay amplitude (90%) refers to the actual valve opening percentage rather than the raw PWM value. When applying the relay output in the high state, the system accounts for the valve's characteristics by mapping this percentage to the appropriate PWM value:

```
// When in high state:
float mappedDuty = VALVE_MIN_DUTY + (AUTOTUNE_RELAY_HIGH /
    100.0) * (VALVE_MAX_DUTY - VALVE_MIN_DUTY);
uint32_t pwmValue = (uint32_t)((mappedDuty / 100.0) *
    pwm_max_value);
ledcWrite(PWM_CHANNEL_MOSFET, pwmValue);

```

```
// When in low state:
ledcWrite(PWM_CHANNEL_MOSFET, 0);  // Completely closed
```

Listing 4: Auto-Tuning Relay Output

This ensures the actual valve behavior matches the theoretical model used for parameter calculation.

3.4 Configurable Auto-Tuning Parameters

The VentCon2 system provides several configurable parameters to optimize the auto-tuning process for different system dynamics:

- **Minimum Cycle Time (MIN_CYCLE_TIME)**: Fixed at 1000ms (1 second). This parameter prevents false cycle detection due to noise and should be appropriate for most pneumatic systems.
- **Number of Cycles (AUTO_TUNE_CYCLES)**: Fixed at 10 cycles. This provides good averaging for reliable parameter calculation.
- **Noise Band (AUTOTUNE_NOISE_BAND)**: Fixed at 0.1 bar to prevent noise-triggered oscillations.
- **Timeout**: Fixed at 3 minutes for safety.

These parameters can be adjusted using serial commands:

- `TUNE CYCLE n` - Set minimum cycle time in milliseconds
- `TUNE CYCLES n` - Set number of cycles to collect

3.5 Cycle Detection and Data Collection

The auto-tuning algorithm uses a state machine to detect oscillation cycles:

```
// Transition from low to high output when pressure exceeds setpoint
if (!autoTuneState && pressureInput > settings.setpoint)
{
    if (now - lastTransitionTime > MIN_CYCLE_TIME)
    {
        // Record cycle data and switch to low output
        cycleTimes[currentCycle] = now - lastTransitionTime;
        cycleAmplitudes[currentCycle] = maxPressure - minPressure;
        currentCycle++;

        Serial.printf("Cycle %d: Period=%.2fs, Amplitude=%.2f bar (Max=%.2f, Min=%.2f)\n",
                      currentCycle, (now - lastTransitionTime)/1000.0,
                            cycleAmplitudes[currentCycle-1],
    maxPressure, minPressure);
    }
}
```

Listing 5: Cycle Detection Logic

The system tracks the maximum and minimum pressure values during each cycle to calculate the oscillation amplitude accurately.

3.6 Multiple Tuning Rules

The VentCon2 system implements four different tuning rules to accommodate different control requirements [8]:

Table 1: PID Tuning Rules [12]

Tuning Rule	K_p	K_i	K_d
Z-N Classic [2]	$1.5K_u \cdot VRC$	$3.0K_u/T_u \cdot VRC$	$0.18K_uT_u$
Balanced response with moderate overshoot			
Z-N Aggressive	$2.0K_u \cdot \text{aggr} \cdot VRC$	$4.5K_u/T_u \cdot VRC$	$0.25K_uT_u$
Faster response with potential increased overshoot			
Tyreus-Luyben [3]	$1.2K_u \cdot VRC$	$1.0K_u/T_u \cdot VRC$	$0.35K_uT_u$
Reduced overshoot, slower response			
Pessen Integral [4]	$2.2K_u \cdot VRC$	$5.0K_u/T_u \cdot VRC$	$0.3K_uT_u$
Fast setpoint tracking			

Where Z-N stands for Ziegler-Nichols, "aggr" is the aggressiveness factor (default 2.0) that can be adjusted from 0.5 to 2.0, and VRC is the Valve Range Compensation factor (2.5x for 50-90% effective range).

The system defaults to the Ziegler-Nichols Aggressive rule with an aggressiveness factor of 2.0, as this provides a good balance between response speed and stability for most applications.

3.7 Valve Range Compensation

A critical aspect of the VentCon2 auto-tuning implementation is the valve range compensation. Since the valve only operates effectively in the 50-90% duty cycle range (40% total range instead of 100%), the calculated PID parameters are scaled by a compensation factor:

$$\text{Valve Range Compensation} = \frac{100\%}{\text{VALVE\_MAX\_DUTY} - \text{VALVE\_MIN\_DUTY}} = \frac{100\%}{90\% - 50\%} = 2.5 \tag{3}$$

This compensation is applied to the proportional and integral gains to account for the reduced effective control range, ensuring the PID controller can achieve the same performance despite the valve limitations.

3.8 Adjustable Aggressiveness

A unique feature of the VentCon2 auto-tuning system is the adjustable aggressiveness factor, which provides an additional layer of control over the tuning outcome:

- Values from 0.5 to 1.0 provide more conservative control with less overshoot
- The default value of 2.0 provides aggressive response for fast pressure control
- Values from 1.5 to 2.0 generate very aggressive control for applications requiring minimal response time

This factor primarily affects the proportional and integral gains, allowing fine-tuning without changing the fundamental tuning rule.

```
// Example of how aggressiveness affects parameter calculation
newKp = 2.0 * Ku * tuningAggressiveness *
    valveRangeCompensation;
newKi = 4.5 * Ku / Tu * valveRangeCompensation;
```

Listing 6: Aggressiveness Implementation

3.9 Auto-Tuning Workflow

The typical auto-tuning workflow for a user is:

1. Select the desired tuning rule (optional): `TUNE RULE n`
    - 0: Ziegler-Nichols Classic (balanced)
    - 1: Ziegler-Nichols Aggressive (faster response) - Default
    - 2: Tyreus-Luyben (minimal overshoot)
    - 3: Pessen Integral (fast setpoint tracking)
2. Adjust aggressiveness if needed (optional): `TUNE AGGR x` (0.5-2.0, default 2.0)
3. Start the auto-tuning process: `TUNE START`
4. Monitor the cycle detection progress via serial output
5. Wait for the process to complete (typically 1-3 minutes)

6. Review the calculated parameters and tuning statistics
7. Accept or reject the parameters: `TUNE ACCEPT` or `TUNE REJECT`

Users can view available tuning rules and the current settings with the command `TUNE RULES`.

3.10 Minimum Cycle Time Guidelines

Proper selection of the minimum cycle time is critical for successful auto-tuning:

- **Fast pneumatic systems**: 100-500ms - Systems with small volumes and fast valves
- **Medium pneumatic systems**: 500-2000ms - Typical pressure control applications (default range)
- **Slow pneumatic systems**: 2000-5000ms - Large volume systems or slow valve response
- **Hydraulic systems**: 1000-3000ms - Generally faster than pneumatics but with higher inertia
- **Thermal systems**: 3000-10000ms - Very slow response systems

To estimate the appropriate cycle time:

1. Manually test system response: `PWM 0` then `PWM 70`
2. Observe the time for pressure to change significantly
3. Set MIN_CYCLE_TIME to 2-3 times this response time

4 Performance Considerations

4.1 Tuning Rule Selection

- **Ziegler-Nichols Classic**: Provides a balanced response, but may exhibit significant overshoot
- **Ziegler-Nichols Aggressive**: Faster response time at the cost of potentially increased overshoot
- **Tyreus-Luyben**: More conservative, with reduced overshoot but slower response
- **Pessen Integral**: Emphasizes setpoint tracking with fast recovery from disturbances

4.2 Comparative Step Responses

Figure 2 illustrates the typical step responses achieved with each tuning rule. These theoretical response curves demonstrate the relative differences in performance characteristics.

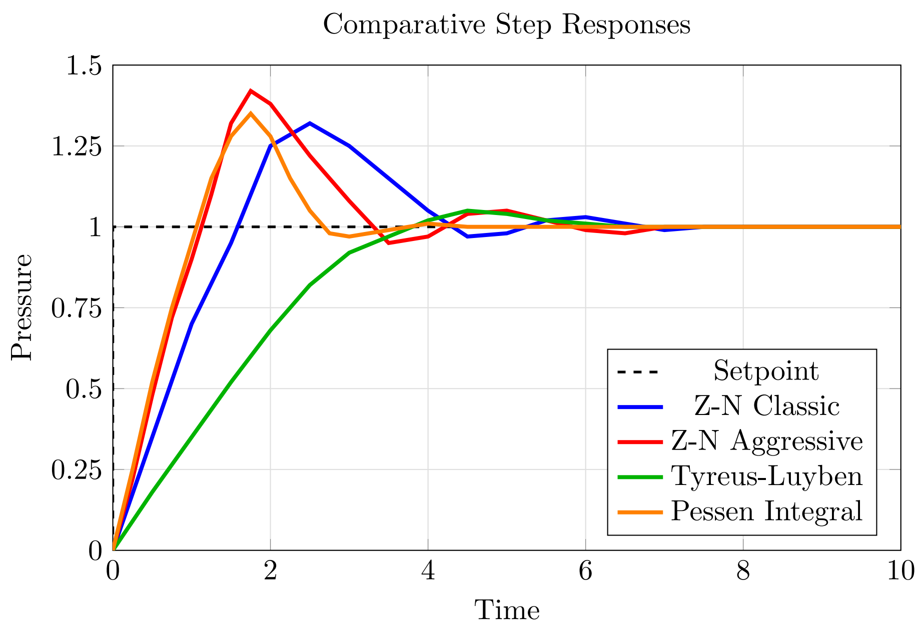

Figure 2: Typical step responses for different tuning rules applied to the pressure control system. The plot shows how each rule affects rise time, overshoot, and settling characteristics.

The plot illustrates several important characteristics:

- **Z-N Classic (blue)**: Shows a moderate rise time with significant overshoot (approximately 32%) and some oscillation before settling.
- **Z-N Aggressive (red)**: Demonstrates the fastest rise time with the highest overshoot (approximately 42%) but can take longer to fully settle due to more pronounced oscillations.
- **Tyreus-Luyben (green)**: Exhibits the slowest rise time but with minimal overshoot (approximately 5%), making it ideal for applications where overshoot must be minimized.
- **Pessen Integral (orange)**: Provides a fast rise time with moderate overshoot (approximately 35%) but settles more quickly than the Ziegler-Nichols methods, making it well-suited for setpoint tracking applications.

The aggressiveness factor in the Z-N Aggressive method can shift its response curve between the Classic and Aggressive profiles shown here, providing flexible tuning options.

4.3 Practical Recommendations

1. Start with the default (Ziegler-Nichols Aggressive, aggressiveness=2.0)
2. If the response is too sluggish:
   - First try increasing aggressiveness: `TUNE AGGR 1.5` or `TUNE AGGR 2.0`
   - If still insufficient, try Pessen Integral rule: `TUNE RULE 3`
3. If overshoot is problematic:
   - First try reducing aggressiveness: `TUNE AGGR 0.8`
   - If still insufficient, switch to Tyreus-Luyben: `TUNE RULE 2`
4. Enable anti-windup (`AW ON`) for all tuning methods to improve recovery from saturation
5. For valves with significant stiction, enable hysteresis compensation (`HYST ON`)
6. After auto-tuning, fine-tune individual parameters if necessary:
   - To reduce overshoot: reduce Kp or increase Kd
   - To eliminate steady-state error: increase Ki
   - To reduce oscillation: increase Kd or reduce Kp

4.4 Limitations

The auto-tuning method has several limitations to be aware of:

- Works best for linear or approximately linear systems
- May not produce optimal results for highly non-linear systems
- Performance can be affected by noise (mitigated with AUTOTUNE_NOISE_BAND)
- Requires a stable starting point for best results
- External disturbances during tuning can affect accuracy

5 Conclusion

The VentCon2 system implements a sophisticated PID control system with advanced features like anti-windup and hysteresis compensation. The automated tuning procedure based on the relay method provides an accessible way to optimize the control parameters for specific applications without requiring in-depth knowledge of control theory [5].

The multiple tuning rules offer flexibility to balance between response speed, overshoot, and disturbance rejection based on the specific requirements of the application [6] [9] [10] [11].

6 References

References

[1] Åström, K.J. and Hägglund, T. (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20(5):645–651.

[2] Ziegler, J.G. and Nichols, N.B. (1942). Optimum settings for automatic controllers. *Transactions of the ASME*, 64:759–768.

[3] Tyreus, B.D. and Luyben, W.L. (1992). Tuning PI controllers for integrator/dead time processes. *Industrial & Engineering Chemistry Research*, 31(11):2625–2628.

[4] Pessen, D.W. (1954). A new look at PID-controller tuning. *Journal of Basic Engineering*, 76:494–501.

[5] Åström, K.J. and Hägglund, T. (1995). *PID Controllers: Theory, Design, and Tuning.* Instrument Society of America, Research Triangle Park, NC, 2nd edition.

[6] O'Dwyer, A. (2009). *Handbook of PI and PID controller tuning rules.* Imperial College Press, London, 3rd edition.

[7] Wikipedia (2023). PID controller. https://en.wikipedia.org/wiki/PID_controller [Online; accessed November 2023]

[8] Wikipedia (2023). PID controller tuning. https://en.wikipedia.org/wiki/PID_controller#Loop_tuning [Online; accessed November 2023]

[9] Control Guru (2023). PID control and auto-tuning resource site. https://controlguru.com/ [Online; accessed November 2023]

[10] MathWorks (2023). PID Tuning Algorithms. https://www.mathworks.com/help/control/ug/pid-tuning-algorithms.html [Online; accessed November 2023]

[11] Control Engineering Practice (2023). PID Control Tuning: Principles and Applications. https://blog.opticontrols.com/archives/344 [Online resource; accessed November 2023]

[12] National Instruments (2023). PID theory explained. https://www.ni.com/en-us/innovations/white-papers/06/pid-theory-explained.html [Online; accessed November 2023]