# 1 Kalman Filter for Current Measurement in PID Control System

## 1.1 Algorithm Description

This Kalman filter implements a one-dimensional recursive filter designed for current measurement filtering in electronic load systems. The filter provides optimal state estimation by combining predictions with noisy measurements, reducing measurement noise while maintaining system responsiveness.

**Algorithm 1** One-Dimensional Kalman Filter for Current Measurement

**Require:** $z(k)$: noisy measurement, $Q$: process noise covariance, $R$: measurement noise covariance
**Ensure:** $\hat{x}(k)$: filtered estimate
Initialize: $\hat{x} \leftarrow 0$, $P \leftarrow 1.0$, $K \leftarrow 0$, $initialized \leftarrow false$
Set noise parameters: $Q \leftarrow 0.001$, $R \leftarrow 0.1$ {Default values}
**while** new measurement available **do**
  **if** $\neg initialized$ **then**
    $\hat{x} \leftarrow z(k)$ {Initialize with first measurement}
    $initialized \leftarrow true$
    **return** $\hat{x}$
  **end if**
  {— Prediction Step —}
  $\hat{x}^-(k) \leftarrow \hat{x}(k-1)$ {State prediction (constant model)}
  $P^-(k) \leftarrow P(k-1) + Q$ {Error covariance prediction}
  {— Update Step —}
  $K(k) \leftarrow \frac{P^-(k)}{P^-(k)+R}$ {Kalman gain calculation}
  $\hat{x}(k) \leftarrow \hat{x}^-(k) + K(k) \cdot [z(k) - \hat{x}^-(k)]$ {State update}
  $P(k) \leftarrow [1 - K(k)] \cdot P^-(k)$ {Error covariance update}
  **return** $\hat{x}(k)$
**end while**

## 1.2 Integration with PID Controller

The Kalman filter is seamlessly integrated into the PID control loop to provide filtered current measurements:

## 1.3 Mathematical Formulation

The discrete-time Kalman filter for current measurement can be expressed as:

**Algorithm 2** PID Controller with Kalman Filtered Measurements

**Require:** $r(k)$: current setpoint, $z(k)$: raw current measurement
**Ensure:** $u(k)$: DAC output voltage
Initialize Kalman filter with parameters $(Q, R)$
Initialize PID parameters: $K_p$, $K_i$, $K_d$
**while** control loop active **do**
  $z_{raw}(k) \leftarrow$ read_sensor() {Get raw current measurement}
  **if** Kalman filter enabled **then**
    $y(k) \leftarrow$ KalmanFilter.update($z_{raw}(k)$) {Apply filtering}
  **else**
    $y(k) \leftarrow z_{raw}(k)$ {Use raw measurement}
  **end if**
  $e(k) \leftarrow r(k) - y(k)$ {Calculate error}
  $u(k) \leftarrow$ PID_compute($e(k)$, $y(k)$) {PID calculation}
  convert_output($u(k)$) {Output to DAC}
**end while**

### 1.3.1 State Model

For current measurement filtering, we use a simple constant model:

$$x(k) = x(k-1) + w(k-1) \quad \text{(State equation)} \tag{1}$$
$$z(k) = x(k) + v(k) \quad \text{(Measurement equation)} \tag{2}$$

where:

- $x(k)$ is the true current at time $k$
- $z(k)$ is the noisy measurement
- $w(k-1) \sim \mathcal{N}(0, Q)$ is the process noise
- $v(k) \sim \mathcal{N}(0, R)$ is the measurement noise

### 1.3.2 Recursive Equations

**Prediction Step:**

$$\hat{x}^-(k) = \hat{x}(k-1) \tag{3}$$
$$P^-(k) = P(k-1) + Q \tag{4}$$

**Update Step:**

$$K(k) = \frac{P^-(k)}{P^-(k) + R} \tag{5}$$
$$\hat{x}(k) = \hat{x}^-(k) + K(k)[z(k) - \hat{x}^-(k)] \tag{6}$$
$$P(k) = [1 - K(k)]P^-(k) \tag{7}$$

## 1.4 Parameter Tuning Guidelines

### 1.4.1 Process Noise Covariance (Q)

- **Small Q (0.001-0.01)**: Assumes current changes slowly, provides smoother output
- **Large Q (0.1-1.0)**: Allows for rapid current changes, less filtering
- **Recommended**: Start with $Q = 0.001$ for stable current applications

### 1.4.2 Measurement Noise Covariance (R)

- **Small R (0.01-0.1)**: Trusts measurements more, faster response
- **Large R (0.5-2.0)**: Filters out more noise, slower response
- **Recommended**: Start with $R = 0.1$ and adjust based on measurement quality

### 1.4.3 Tuning Strategy

1. **High noise environment**: Increase $R$ to $0.5 - 1.0$
2. **Fast response needed**: Decrease $R$ to $0.05 - 0.1$
3. **Very stable system**: Decrease $Q$ to $0.0001 - 0.001$
4. **Dynamic system**: Increase $Q$ to $0.01 - 0.1$

## 1.5 Performance Characteristics

### 1.5.1 Advantages

- **Optimal estimation**: Minimizes mean squared error under Gaussian assumptions
- **Real-time capability**: Recursive implementation suitable for embedded systems
- **Adaptive**: Kalman gain automatically balances prediction vs. measurement
- **Memory efficient**: Only requires storing previous state and covariance

### 1.5.2 Computational Complexity

- **Time complexity**: $O(1)$ per update (constant time)
- **Space complexity**: $O(1)$ memory usage
- **Suitable for**: Real-time embedded applications with limited resources