C++ Implementation

Pieter P

Simplifying the Difference Equation

Recall the Simple Moving Average difference equation:

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i] \tag{1}$$

A naive approach would be to implement the difference equation directly: keeping the last $N-1$ inputs, and calculate the sum on each iteration, calculating $N-1$ additions at each time step.
However, we can do much better if we notice how only two terms of the sum change each time:

$$\begin{aligned}
y[n+1] &= \frac{1}{N} \sum_{i=0}^{N-1} x[n+1-i] \\
&= \frac{1}{N} \left(x[n+1] + \sum_{i=1}^{N-1} x[n+1-i] \right) \\
&= \frac{1}{N} \left(x[n+1] + \sum_{i=1}^{N-1} x[n+1-i] + x[n+1-N] - x[n+1-N] \right) \\
&= \frac{1}{N} \left(x[n+1] + \sum_{i=1}^{N} x[n+1-i] - x[n+1-N] \right) \\
&= \frac{1}{N} \left(x[n+1] + \sum_{i=0}^{N-1} x[n-i] - x[n+1-N] \right) \\
&= y[n] + \frac{1}{N} \Big(x[n+1] - x[n+1-N] \Big)
\end{aligned}$$

We can now define the sum $S[n]$ as follows:

$$\begin{aligned}
S[n] &\triangleq N \cdot y[n] \\
&= \sum_{i=0}^{N-1} x[n-i] \\
\Leftrightarrow \; y[n] &= S[n]/N
\end{aligned}$$

The difference equation then becomes:

$$S[n+1] = S[n] + x[n+1] - x[n+1-N] \tag{2}$$

To update the sum, each iteration now requires only one addition and one subtraction, as well as some housekeeping to remember the previous inputs. To get the output $y[n]$, a division by N is needed.

C++ Implementation

We can now implement Equation 2 directly, and we'll use a rounding division instead of truncating the quotient. Note that this rounding operation is valid for unsigned integer types only.
The previous inputs $x[n-i]$ are kept in a circular buffer.

SMA.cpp

```cpp
#include <stdint.h>

template <uint8_t N, class input_t = uint16_t, class sum_t = uint32_t>
class SMA {
  public:
    input_t operator()(input_t input) {
        sum -= previousInputs[index];
        sum += input;
        previousInputs[index] = input;
        if (++index == N)
            index = 0;
        return (sum + (N / 2)) / N;
    }

    static_assert(
        sum_t(0) < sum_t(-1),  // Check that `sum_t` is an unsigned type
        "Error: sum data type should be an unsigned integer, otherwise, "
        "the rounding operation in the return statement is invalid.");

  private:
    uint8_t index             = 0;
    input_t previousInputs[N] = {};
    sum_t sum                 = 0;
};
```

Arduino Example

As a basic example, you can use this filter for smoothing analog inputs on microcontrollers. Keep in mind that an exponential moving average filter is often more appropriate than a simple moving average filter. The SMA uses much more memory, and is much slower than the EMA. The exponential impulse response of the EMA may be better as well.
You can find an Arduino example using an EMA here.

```cpp
template <uint8_t N, class input_t = uint16_t, class sum_t = uint32_t>
class SMA {
  public:
    input_t operator()(input_t input) {
        sum -= previousInputs[index];
        sum += input;
        previousInputs[index] = input;
        if (++index == N)
            index = 0;
        return (sum + (N / 2)) / N;
    }

    static_assert(
        sum_t(0) < sum_t(-1),  // Check that `sum_t` is an unsigned type
        "Error: sum data type should be an unsigned integer, otherwise, "
        "the rounding operation in the return statement is invalid.");

  private:
    uint8_t index             = 0;
    input_t previousInputs[N] = {};
    sum_t sum                 = 0;
};

void setup() {
  Serial.begin(115200);
  while (!Serial);
}

const unsigned long interval = 10000; // 100 Hz

void loop() {
  static SMA<20> filter;
  static unsigned long prevMicros = micros();
  if (micros() - prevMicros >= interval) {
    int rawValue = analogRead(A0);
    int filteredValue = filter(rawValue);
    Serial.print(rawValue);
    Serial.print('\t');
    Serial.println(filteredValue);
    prevMicros += interval;
  }
}
```