Jack A. Wilkie*, Thomas Stieglitz and Knut Moeller

Real-Time Multirate Filtering of Digitized Torque Signals on Tiva Microcontroller using Fixed-Point Design with MATLAB

Abstract: Correct bone screw torque is critical for positive patient outcomes after orthopaedic surgery. Models of the screwing process have been developed to allow a smart screwdriver to optimise the insertion torque. Experimental data is required to test these models, so a test-rig has been developed. Accurate torque measurement is a key part of the test-rig. An FIR filter was designed for this torque signal, implemented on the test-rig, and compared theoretically and experimentally to a mean filter and to no filtering. The FIR and mean filters both performed well, with the FIR achieving better theoretical results, and the mean filter achieving better experimental results. Better understanding of the noise structure and potential signal distortion would be required to improve the FIR filter or to conclusively compare it against the mean filter, however both perform sufficiently well for this application.

Keywords: Finite impulse response filter, filter design, filter implementation, digital signal processing, real time computing, fixed-point

https://doi.org/10.1515/cdbme-2021-2183

1 Introduction

Bone screws are widely used in orthopaedic surgery to secure implants, and to support bones during natural healing. In both cases, it is critical that the connections are as resilient as possible. The correct torquing of the screws is a major factor in the success of the joint, as over-torquing will strip the threads and compromise strength [1], and under-torqued screws are susceptible to loosening over time [2]. While surgeons are highly skilled, it is easy to misjudge the torque, or incorrectly torque a screw [3].

Wilkie et al. [4] proposed the use of a smart screwdriver to automatically identify and enforce the optimal torque. Previous work has focused on developing and testing models of screw insertion for detecting the bone material properties [4]–[6], and for predicting the optimal torque from these properties [7]. To test these models, a test rig was developed [8]. The test rig had to collect torque and displacement data with a high accuracy and sample rate.

This paper discusses the acquisition and processing of the analogue torque signal by the microcontroller in the test rig. This will focus on the use of digital filtering to reduce the noise on the signal and allow oversampling. The filter design process and constraints will be discussed, as well as the implementation accounting for real-time processing constraints on the relatively low-power Tiva TM4C123GH6PM microcontroller.

2 Methods

The Tiva TM4C123GH6PM (Texas Instruments Inc.) is an 80 MHz ARM Cortex-M4F microcontroller. We will specifically make use of its 12-bit, 1 Msps ADC and µDMA controller.

2.1 Filter Design

The bandwidth of the torque signal is assumed to be 0-100 Hz. We use the full speed of the ADC; hence, the input sample rate is 1 MHz. We decided that the test rig would output data at 1000 Hz, hence we want to make sure any noise above this frequency is sufficiently filtered to prevent it aliasing during the down sampling/decimation. Any noise in the 100-1000 Hz range can be filtered offline later using more resource-intensive methods on a powerful desktop CPU. Choosing 1000 Hz gave a large transition band for the filter design from 100

***Corresponding author: Jack Wilkie:** Institute for Technical Medicine, Hochschule Furtwangen, Villingen-Schwenningen, Germany, e-mail: wij@hs-furtwangen.de
Thomas Stieglitz: Department of Microsystems Engineering, University of Freiburg, Freiburg, Germany
Knut Moeller: Institute for Technical Medicine, Hochschule Furtwangen, Villingen-Schwenningen, Germany

Hz to 1000 Hz, allowing prioritization of stopband attenuation, passband ripple, and computational complexity.

We had to choose between an FIR or IIR filter. In this case, because the output was only 1000Hz, the FIR convolution would only need to be calculated for every output sample, while an IIR filter would require many computations for every input sample. While the FIR convolution is more computationally demanding that the IIR iteration, the much lower rate made it favourable compared to IIR filtering, additionally, it would not require computationally expensive floating-point calculations.

The computational complexity of the FIR filtering is determined by the number of filter coefficients. Generally, sharper transition bands, higher stopband attenuation, narrower passbands (related to down sampling ratio here), and lower passband ripple will all increase the number of coefficients. To reduce the computational requirements, we used the ADC's built-in hardware averaging. We configured this as a 4-sample average, as it provided a good reduction in the computational complexity, while introducing minimal artefacts in the frequency response. Therefore, the input sample rate for the FIR filter was now 250 kHz.

The primary filter design was performed using the MATLAB "designfilt" function. We designed a low-pass filter for a sampling frequency of 250 kHz (ADC frequency after hardware averaging), with a passband frequency of 100 Hz (to match the assumed bandwidth of the data), and a stopband frequency of 1000 Hz (to match the planned output data rate). We selected the equiripple method, and specified the passband ripple as 1 dB, and the stopband attenuation as 60 dB. This gave us a type I (symmetric, odd-length) linear-phase FIR filter with 627 coefficients. All coefficients were positive.

2.2 Filter Implementation

The filter was implemented using fixed-point arithmetic to maintain accuracy while keeping the same speed as integer arithmetic, and to give more predictable accuracy and rounding compared to floating point. First, the filter coefficients were scaled to use the full dynamic range of a 16-bit unsigned integer; the highest coefficient was scaled to 65535 and the others were scaled proportionally. Generally, the filter was applied by convolution: for each output sample the 627 coefficients were multiplied elementwise by the previous 627 ADC samples and the sum of the products was accumulated. This used a 64-bit accumulator variable as a saturated ADC input would result in a 38-bit value, which would overflow a 32-bit integer. This accumulator was then scaled to fit within a 32-bit integer.

The 32-bit filtered value was then calibrated. This was done using an external precision power supply. This was also required as the torque sensor analogue output voltage ranged from 0-10 V, and a voltage divider was used to bring this into the microcontrollers 3.3 V range and the resistors were uncalibrated; this voltage calibration accounted for the combination of the voltage divider and ADC variations and corrected the gain from the filter. The external voltage was stepped from 1-9 V in 0.5 V steps, and the filtered value was recorded at each step. A linear fit was performed to get a multiplicative and additive offset for the calibration.

DMA was used to optimise the transfer of data from the ADC to the main memory. The DMA was programmed in a ping-pong mode. The ADC was free-running and produced new samples at 250 kHz (after 4x hardware averaging), when a sample was ready, the DMA module would copy this into a buffer in memory without processor intervention. Every 250 values, the DMA would switch to the next buffer, and trigger an interrupt to alert the program that data was ready to filter; this happened at 1000 Hz, so every time the interrupt was triggered, a value should be calculated from the filter, and output from the test rig.

Because of the 627 coefficients, the filter would need to process the last 3 buffers of 250 values, so 4 buffers of 250 samples were used to allow DMA writing to the last one without interfering with the filtering. As the data was only being output at 1000 Hz, the filtering and decimation could be combined into a single step to greatly reduce the computational load, as any filtered 250 kHz data points between the 1000 Hz samples would be discarded anyway.

Some care had to be taken in optimising the code, as there were only 80,000 processor cycles to process the convolution each time, which required a minimum of 627 multiplications, 627 additions, and even more control and data-access instructions; additionally, the processor must still be able to simultaneously perform other time-sensitive tasks, such as USB communications and motor control. The microcontroller was using a pre-emptive scheduler with FreeRTOS, and the long-running filtering task needed a lower priority to avoid blocking other fast tasks; hence, care had to be taken with other higher-priority tasks not to perform long-running operations which would cause the filtering task to fall behind and lose data.

2.3 Testing

The filter was compared to no filtering/a unit impulse filter (substitute coefficients with [1,0,0,…,0]), and a mean filter ([1,1,1,…,1]). The magnitude frequency responses of the filters were plotted, including a detailed plot of the response

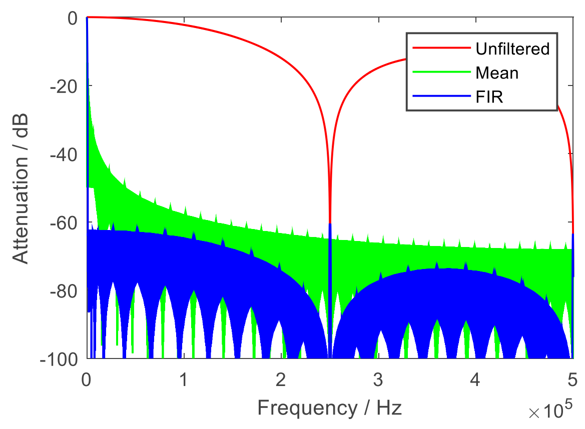

Figure 1: Theoretical frequency responses for the 3 different filtering methods tested.

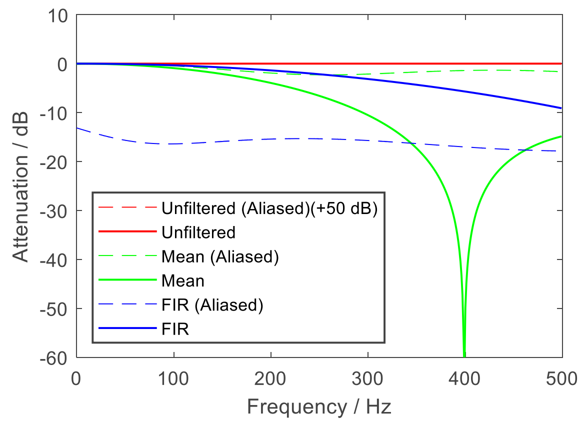

Figure 2: Theoretical frequency responses after decimation to 1000 Hz. Showing responses from signals <1000 Hz, as well as the sum of aliased responses >1000 Hz.

for the passband and stopband after decimation, accounting for aliasing.

The three filters were then evaluated on the test rig. For each filter, 10 seconds of data was collected while the torque sensor was unloaded and stationary. An FFT of the data was used to evaluate the noise spectrum after each filtering method.

3 Results

The theoretical frequency responses from the 3 tested methods are shown in figures 1,2, and 3. Figure 1 shows the entire response over the range of input frequencies. Figure 2 shows the response in the range of output frequencies, showing the directly mapped response from input components under 1000 Hz, and the aliased components summed from any input components over 1000 Hz. Figure 3 shows a detailed view of the responses in the passband.

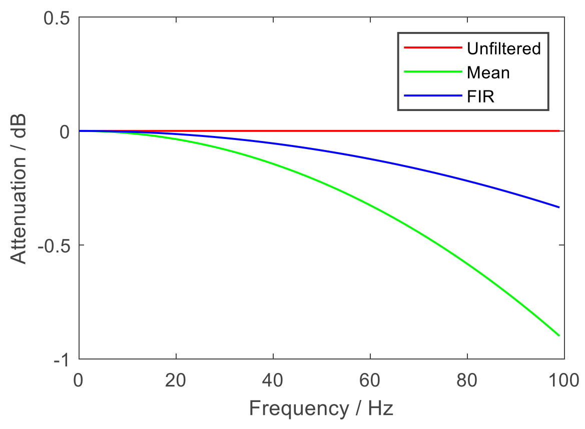

Figure 3: Theoretical frequency responses of the 3 different filtering methods in the passband.

The experimental data is shown in Figure 4. This shows the noise power spectrum using each of the different filtering methods while the sensor is unloaded and stationary.

4 Discussion

From the theoretical data, both a mean filter and the designed FIR filter should significantly reduce noise. Figure 1 shows that the attenuation of the designed FIR filter is notably better than that of a moving average filter across most of the spectrum, and generally has a much steeper transition band, however Figure 2 shows that the mean filter has more attenuation in the very low frequencies. More attenuation in the low frequencies may remove more noise, especially since random noise tends to be skewed towards lower frequencies, however it may also attenuate some on the desired signal; Figure 3 shows that the passband attenuation of the mean filter is higher than that of the FIR filter, however whether this small difference matters in practice is not clear. Figure 2 also shows that the FIR filter should reduce aliased higher-frequency noise approximately 15 dB more than the mean filter.

The experimental data in Figure 4 shows that both the mean and FIR filters significantly reduce noise in practice. However, the experimental results here contradict the theoretical predictions in Figure 2, with the mean filter appearing to have notably lower noise. This could be due to the extra low-frequency attenuation predicted for the mean filter, which would give it an advantage if the noise is significantly skewed towards the lower frequencies already. However, it should also be noted that this data is only considering the noise filtering of the filters and has not investigated the distortion of the desired torque signal, which may be higher for the mean filter due to the added low-frequency attenuation.

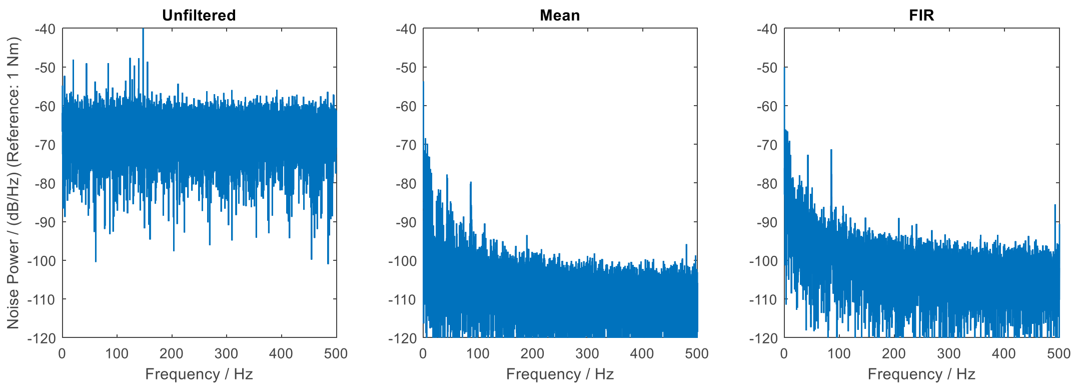

Figure 4: Experimental noise power spectrums from each filtering method.

It is difficult to say anything certain about the noise structure; from the unfiltered plot in Figure 4, the noise power distribution is relatively uniform over the 500 Hz bandwidth tested. However, the noise here is a combination of noise directly present in this low-frequency band, and any higher-frequency noise that has been aliased down during the sampling. It would be possible to develop a special version of the test-rig code specifically to analyse the high-sample-rate data from the ADC; this would give further insight into the structure of the noise present and may allow more optimal filters to be designed.

With more information, the design of the filter could be optimised by adjusting the desired attenuations/weightings for several different bands. However, the current FIR filtering methodology already reduces the noise power significantly compared to the unfiltered signal, and due to this, the limiting factor is more likely to be the torque sensor itself, rather than the analogue signal acquisition in the microcontroller. This work also suggests that a simple mean filter may have similar or better performance; if this is the case, it could significantly reduce the computational requirements of the filtering due to the highly optimised implementations possible for mean filters.

5 Conclusion

We developed an FIR filter using the filter design tools in MATLAB. This was then implemented into a real-time system on a TIVA TM4C123GH6PM microcontroller.

We found some differences between the predicted and actual performance of the FIR filter compared to a simple moving average filter; these were likely due to assumptions about the noise structure.

It may be possible to further optimise the filter after careful investigation into the noise structure, or simply substitute the complex FIR filter with a faster mean filter, however the implementation at present is likely more than sufficient for the application.

Author Statement

Research funding: This work was partially supported by grants “CiD” and “Digitalisation in the OR” from BMBF (Project numbers 13FH5E02IA and 13FH5I05IA). Conflict of interest: Authors state no conflict of interest.

References

[1] A. Feroz Dinah, S. C. Mears, T. A. Knight, S. P. Soin, J. T. Campbell, and S. M. Belkoff, ‘Inadvertent Screw Stripping During Ankle Fracture Fixation in Elderly Bone’, *Geriatr. Orthop. Surg. Rehabil.*, vol. 2, no. 3, pp. 86–89, May 2011, doi: 10.1177/2151458511401352.
[2] M. Evans, M. Spencer, Q. Wang, S. H. White, and J. L. Cunningham, ‘Design and testing of external fixator bone screws’, *J. Biomed. Eng.*, vol. 12, no. 6, pp. 457–462, Nov. 1990, doi: 10.1016/0141-5425(90)90054-Q.
[3] M. J. Stoesz, P. A. Gustafson, B. V. Patel, J. R. Jastifer, and J. L. Chess, ‘Surgeon Perception of Cancellous Screw Fixation’, *J. Orthop. Trauma*, vol. 28, no. 1, p. e1, Jan. 2014, doi: 10.1097/BOT.0b013e31829ef63b.
[4] J. Wilkie, P. D. Docherty, and K. Möller, ‘A simple screwing process model for bone material identification’, *Proc. Autom. Med. Eng.*, vol. 1, no. 1, Art. no. 1, Feb. 2020, doi: 10.18416/AUTOMED.2020.
[5] J. Wilkie, P. D. Docherty, and K. Möller, ‘Developments in Modelling Bone Screwing’, *Curr. Dir. Biomed. Eng.*, vol. 6, no. 3, pp. 111–114, Sep. 2020, doi: 10.1515/cdbme-2020-3029.
[6] J. Wilkie, P. D. Docherty, and K. Möller, ‘Model-based bone material property identification’, *- Autom.*, vol. 68, no. 11, pp. 913–921, Nov. 2020, doi: 10.1515/auto-2020-0083.
[7] J. Wilkie, P. D. Docherty, and K. Möller, ‘Stripping Torque Model for Bone Screws’, presented at the 11th IFAC Symposium on Biological and Medical Systems, Ghent, Belgium, In press.
[8] J. A. Wilkie and K. Möller, ‘Test rig for bone screw insertion modelling’, presented at the 15th Interdisciplinary Symposium Automation in Medical Engineering (AUTOMED 2021)., Basel, Switzerland, 2021.