Application of Kalman Filter for Intelligent Built–in Torque Sensor of Harmonic Drives

Hamid D. Taghirad

CIM-TR-97-06 April 1997

Centre for Intelligent Machines
McGill University
Montréal, Québec, Canada

Postal Address: 3480 University Street, Montréal, Québec, Canada H3A 2A7
Telephone: (514) 398-6319 Telex: 05 268510 FAX: (514) 398-7348
Email: cim@cim.mcgill.ca

Abstract

A harmonic drive is a compact, light–weight and high–ratio torque transmission device which has almost zero backlash. Its unique performance features has captured the attention of designers in many industrial applications, especially in robotics. Despite widespread industrial application of harmonic drives, the torque control of this system has not been fully addressed. In this thesis the robust torque control of harmonic drive system is examined in detail.

In order to measure the transmitted torque of a harmonic drive and for the purpose of a torque feedback scheme, an intelligent built–in torque sensing technique is developed in this thesis. Specifically, strain–gauges are mounted directly on the flexspline and therefore no extra flexible element is introduced into the system. To have maximum sensing accuracy, four Rosette strain gauges are employed using an accurate positioning method. Kalman filter estimation is employed to cancel the torque ripples, oscillations observed on the measured torque and caused mainly by gear teeth meshing. A simple forth order harmonic oscillator proved sufficient to model these torque ripples. Moreover, the error model is extended to incorporate any misalignment torque. By on–line implementation of the Kalman filter, it is shown that this method affords a fast and accurate way to filter torque ripples and misalignment torque from torque measurements.

Application of Kalman Filter for Intelligent Built–in Torque Sensor of Harmonic Drives

Résumé

Un entraînement harmonique est un mécanisme léger, de rapport de transmission de couple élevé et qui n'a pratiquement pas de jeu. Ses performances uniques attirent l'attention de concepteurs dans diverses applications industrielles, particulièrement en robotique. Cependant, malgré la prépondérance dans l'industrie d' entraînements harmoniques la commande du couple du système n'a pas encore fait l'objet d'étude complète. Dans cette thèse nous examinons la commande robuste du couple des systèmes à entraînement harmonique en détails.

Ainsi pour mesurer et utiliser le couple transmis d'un entraînement harmonique dans un schéma à rétroaction, nous développons une technique intélligente de détection du couple. Dans celle–ci on monte des extensomètres directement sur l'élément flexible, et donc il n'y a pas d'autre élément flexible introduit dans le système. Afin de minimiser les erreurs de lecture, nous employons quatre jauges de contrainte utilisant une méthode précise de positionnement. Un filtre de Kalman est utilisé pour réduire les ondulations du couple, les oscillations observées sur le couple mesuré sont dûes principalement au maillage de l'engrenage. Un oscillateur harmonique du quatrième ordre a suffit pour modéliser de façon précise les ondulations du couple. De plus par l'implémentation en temps réel du filtre de Kalman, on montre que cette méthode constitue un moyen à la fois rapide et précis de filtration des ondulations du couple et des erreurs d'alignement dûes aux mesures.

Acknowledgements

I wish to extend my sincere gratitude and appreciation to my research advisor, Dr. Piérre R. Bélanger, for his superb guidance throughout my research studies. His penetrating insight and rigour were crucial to the successful completion of this thesis. I would like to express my appreciation to A. Helmy for his involvements in the experiments as well as and my fellow students at the Centre for Intelligent Machines, especially Kaouthar Benameur and Mojtaba Ahmadi for many interesting and enjoyable discussions. Special thanks to my officemates Michael Glaum for his careful review of the manuscript, and Mohammed Seddik Djouadi for his assistance in the French translation of the abstract. I am grateful to the machine shop staff of the Department of Electrical Engineering, especially Don Pavlasek for his valuable suggestions and assistance toward developing the torque sensor and the harmonic drive testing stations. The assistance of the Photonic System Group of Electrical Engineering Department for equipment employed in the development of the torque sensor is gratefully acknowledged.

Contents

List of Figures

List of Tables

1. Introduction

Developed in 1955 primarily for aerospace applications, harmonic drives are high–ratio and compact torque transmission systems. Every harmonic drive consists of the three components illustrated in Figure 1. The wave generator is a ball bearing assembly with a rigid, elliptical inner race and a flexible outer race. The flexspline is a thin–walled, flexible cup adorned with small, external gear teeth around its rim. The circular spline is a rigid ring with internal teeth machined along a slightly larger pitch diameter than those of the flexspline. When assembled, the wave generator is nested inside the flexspline, causing the flexible circumference to adopt the elliptical profile of the wave generator, and the external teeth of the flexspline to mesh with the internal teeth on the circular spline along the major axis of the wave generator ellipse.

If properly assembled, all three components of the transmission can rotate at different but coupled velocities on the same axis. To use the harmonic drive for speed reduction, the wave generator is mounted on the electric motor shaft, and the output is conveyed either through the flexspline while the circular spline is fixed or through the flexspline while the circular spline is fixed. In the latter case, by rotation of the wave generator the zone of gear-tooth engagement is carried with the wave generator major elliptical axis. When this engagement zone is propagated 360° around the circumference of the circular spline, the flexspline which contains fewer teeth than the circular spline, will lag by that fewer number of teeth relative to the circular spline. Through this gradual and continuous engagement of slightly offset teeth, every rotation of the wave generator moves the flexspline a small angle back on the circular

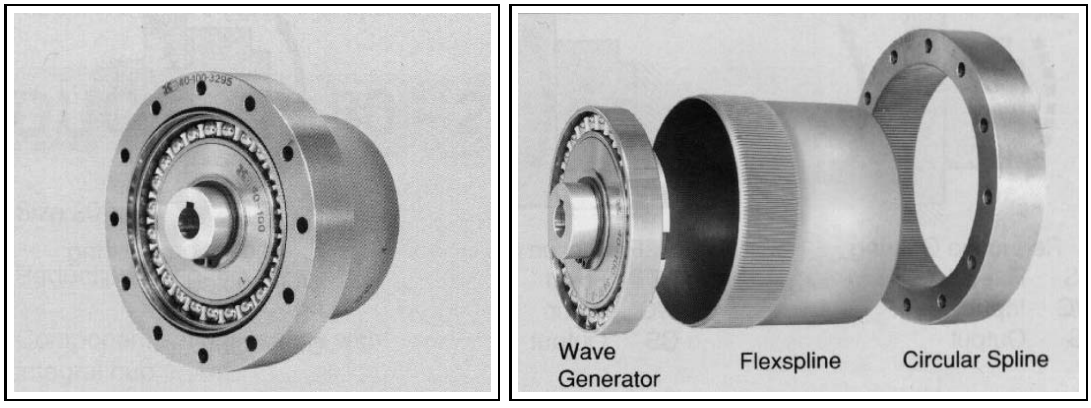

Figure 1. Harmonic drive components

spline, and through this unconventional mechanism, gear ratios up to 320 : 1 can be achieved in a single transmission.

The harmonic drive exhibits performance features both superior and inferior to those of conventional gear transmissions. Its performance advantages include high–torque capacity, concentric geometry, lightweight and compact design, zero backlash, high efficiency, and back drivability. Harmonic drive systems suffer however, from high flexibility, resonance vibration, friction and structural damping nonlinearities [11]. In numerous robotic control techniques such as feedback linearization, computed torque method and some adaptive control schemes, the actuator torque is taken to be the control input [6, 9, 8]. This can be only accomplished through torque feedback at each joint of the robot [12].

In order to apply torque feedback on the robot joint, it is necessary to measure the transmitted torque through the actuator transmission mechanism. Conventionally, torque sensor are placed in the output transmission line of the robot [5, 10]. However, for a harmonic drive transmission, which has an elastic element, the flexspline, there is no advantage to add an additional compliant element and thereby reduce the joint stiffness. In this paper we are using the idea of a built-in torque sensor for harmonic drives as first proposed by Hashimoto in 1989 [2]. This method proved to be an economical and effective way of torque sensing for harmonic drives in our setup, as claimed by Hashimoto et al. [3]. In our testing station a Wheatstone bridge of four Rosette strain–gauges is utilized to sense the torsional torque transmitted through the flexspline. A practical an accurate method is proposed to mount the strain–gauges on the flexspline in order to minimize any radial or circumferential misplacement of each strain–gauge.

One important characteristic of harmonic drive torque transmission, as observed in free motion experiments, is a high frequency oscillation in the output torque signal. These oscillations named torque ripples whose principal frequency of oscillation (in rad/sec) is twice the motor velocity (in rad/sec), are mainly caused by harmonic drive gear meshing vibration. A small fraction of the torque ripples are caused by the non-ideal torque measurement, because of the direct attachment of strain gauges on the flexspline. Since the flexspline has an elliptical shape, strain gauges mounted on the flexspline are subjected to unwanted strain caused by the elliptical shape. We show in this paper, however, that using four Rosette strain–gauges, and using an accurate method to mount the strain–gauges, will reduce the amplitude of the torque ripple to a minimum. Moreover, the dependence of the frequency content of the torque ripples on the velocity makes it possible to model them as a simple harmonic oscillator, and

to employ a Kalman filter to predict and filter them from the torque measurement. If only low–frequency torque control is desired [12], the high–frequency ripples may be removed by estimating them via Kalman filtering. This is more efficient than simple low–pass filtering, because it uses the known structure of the torque signal.

A fourth–order harmonic oscillator error model is proposed in this paper to characterize both the fundamental and first-harmonic frequency content of the torque ripple. Using this model for the torque ripples, a prediction–type Kalman filter algorithm is applied to estimate the torque ripples. The performance of the on–line implemented Kalman filter for torque ripple cancellation is shown to be quite fast and accurate.

The Kalman filter is used not only to estimate the torque ripples, but also to cancel any mechanical misalignment torque signature on the measured torque [15]. Many torque sensors exhibit the limitation of being sensitive to the torques applied on the direction perpendicular to their axis of measurement. In our setup after repeated use of the harmonic drive system for different experiments, a similar torque signature was observed on the measured torque. After examining the system accurately, the source of this torque signature is found to be the misalignment of the harmonic drive shaft and the load. The frequency of misalignment torque(in rad/sec), as it can be intuitively identified from its source of generation, is found to be exactly the same as the output shaft velocity (in rad/sec). Therefore, by adding another block to the harmonic oscillator model we managed to estimate the misalignment component of the measured torque. By using the extended model for the torque ripple and misalignment torque together, it is shown that the filtered torque cancels the torque ripples and misalignment torque quite accurately. The experimental results obtained from combining the original ideas introduced in this paper with robust torque controller design given in [12, 13], proves that built–in torque sensors are viable and economical means to measure harmonic drive transmitted torque and to employ them for torque feedback strategies.

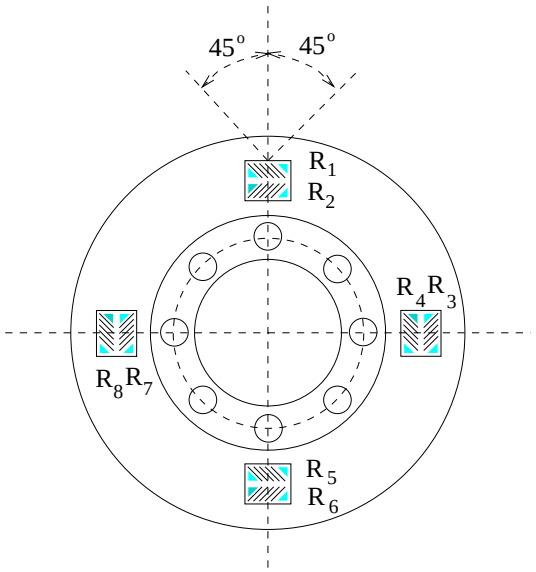

FIGURE 2. Position of four Rosette strain–gauges on the flexspline diaphragm

2. Built–In Torque Sensor

As illustrated in Figure 2 four Rosette strain–gauges are mounted on the diaphragm part of the flexspline. A Rosette stain–gauge consist of two separate strain–gauges perpendicularly mounted on one pad. For a clockwise torque exerted on the flexspline illustrated in Figure 2 strain–gauge R_1 is under compression while strain–gauge R_2 is under tension. Similarly all odd–indexed strain–gauges illustrated in Figure 2 are under compression, while the others are under tension. Thus, a Wheatstone bridge of strain-gauges as illustrated in Figure 3 can transduce the torsion into a difference voltage. The reason why Rosette strain-gauges are necessary for harmonic drive built in torque sensor, as explained in [3], is to compensate for the elliptical shape of the flexspline. If the strain caused by applied torque is named ϵ_t, while the strain caused by the elliptical shape of flexspline is called ϵ_w, Hashimoto et al. [4] illustrated that the strain applied to strain-gauge R_1, and R_2 are

$$\begin{cases} \epsilon_1 = \epsilon_t + \epsilon_w \\ \epsilon_2 = -\epsilon_t + \epsilon_{w'} \end{cases} \tag{1}$$

where $\epsilon_{w'}$ is assumed in [3] to be a sinusoidal modulation of ϵ_w. Thus,

$$\epsilon_1 - \epsilon_2 = 2\epsilon_t + \Psi_0 \sin(2\beta) \tag{2}$$

In order to cancel the modulation $\Psi_0 \sin(2\beta)$ and to detect the actual torsional strain ϵ_t the information from strain–gauges R_3, and R_4 is necessary. These strain–gauges

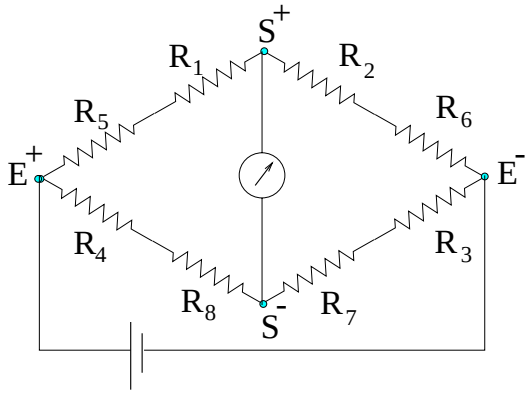

FIGURE 3. Strain–gauges Wheatstone bridge network

are located at an angle of 90^o from Strain–gauges R_1 and R_2, and therefore:

$$\epsilon_3 - \epsilon_4 = 2\epsilon_t + \Psi_0 \sin(2\beta - \pi) \tag{3}$$

Therefore, a Wheatstone bridge, constructed from strain–gauges R_1 to R_4 is sufficient to produce a difference voltage proportional to the torsional strain ϵ_t as following

$$\begin{aligned} E_{out} &= \frac{K}{4}(\epsilon_1 + \epsilon_3 - \epsilon_2 - \epsilon_4)E_{sup} \\ &= K\epsilon_t E_{sup} \end{aligned} \tag{4}$$

in which K is the gauge factor, and E_{out} and E_{sup} denote output and supply voltages, respectively.

Although, only two Rosette strain–gauges are sufficient to extract the torsional strain, two other Rosette strain–gauges are introduced to maintain symmetry, and to minimize the effect of positioning error. Sensing inaccuracies are caused by radial, circumferential, and angular positioning error. Radial positioning error occurs when the gauges are placed on different radii from the center of the flexspline. As shown in Figure 2 even without any radial misplacement of the strain–gauges, radial error exists, since strain–gauges R_1 and R_2 are placed at different radii. However, by using four Rosette strain–gauge, as illustrated in Figure 2, this error will be compensated by strain–gauges R_5, and R_6 which are located in the reverse position.

Circumferential positioning error occurs when two Rosette strain–gauges are mounted in an angle different than 90^o from each other. This positioning error introduces more sensing inaccuracies than radial positioning error [4, 14]. Angular positioning error occurs when a Rosette strain–gauge is not mounted perpendicular to the flexspline's axis of rotation. This positioning error also introduces sensing inaccuracy similar to the circumferential positioning error. To minimize the positioning error, we propose

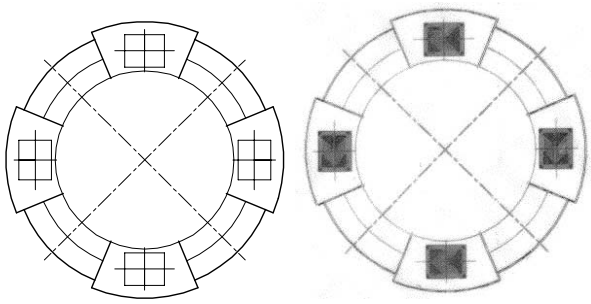

FIGURE 4. Proposed transparent film for accurate strain–gauge placement

a method using a specially–designed transparent film for the strain–gauges placement. As illustrated in Figure 4, an accurate drawing of the strain–gauge placement positions is printed on a transparent film using a laser printer with the finest possible lines. Then the strain-gauges are placed on the transparent film using a microscope. By this means all positioning errors are reduced to a minimum, and as examined in our testing station to other placement methods, the sensing inaccuracy is minimized. When the strain–gauges are mounted on the transparent film as illustrated in Figure 4, the transparent film is placed on the flexspline and the strain–gauges are cemented on the surface. The final configuration of the wired strain–gauges are illustrated in Figure 5. The harmonic drive used in the testing station and illustrated in Figure 5 is from RHS series of HD systems, with gear ratio of 100:1, and rated torque of 40 Nm. The flexspline diameter is 46 mm, and its length is 49 mm. To amplify the output signal, a variable range amplifier with gain 1000 is used, while it generates 10 volts DC voltage as input voltage to the Wheatstone bridge.

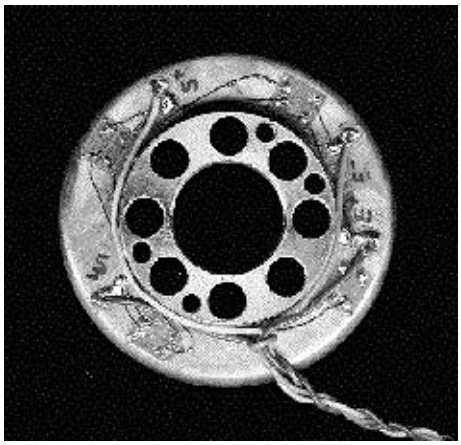

FIGURE 5. Harmonic drive built–in torque sensor

3. Torque Sensor Calibration

To examine the dynamics of the torque sensor and to calibrate it, we locked the harmonic drive wave generator, and applied a known torque on the flexspline. The locking device is a simple shaft resembling the motor shaft, which can be fixed to the ground. The output torque is applied using arm and weight method. The torque calibration details of loading and unloading some weight on the arm, in both direction. This experiment is repeated at different flexspline positions, to check the position dependence of the torque sensor. Also low-range and full-range loading experiments are tested at each position to evaluate the nonlinearity of the torque sensor. A typical result for full-range loading experiment are shown in Figure 6. These data are best fitted by a straight line using least–squares approximation. The estimated torque sensor gain for each experiment is calculated from the slope of this line. However, this gain is deemed acceptable, only if it is consistent for other experiments. By consistency we mean a statistical measure, namely *the ratio of the standard deviation to the average value of estimated parameter for different experiments* [11]. If this measure is small, we have a good consistency for different experiments, and the final gain is obtained from the average value of the estimated gain for different experiments. The final gain is obtained by this method for eight experiments, and the results are

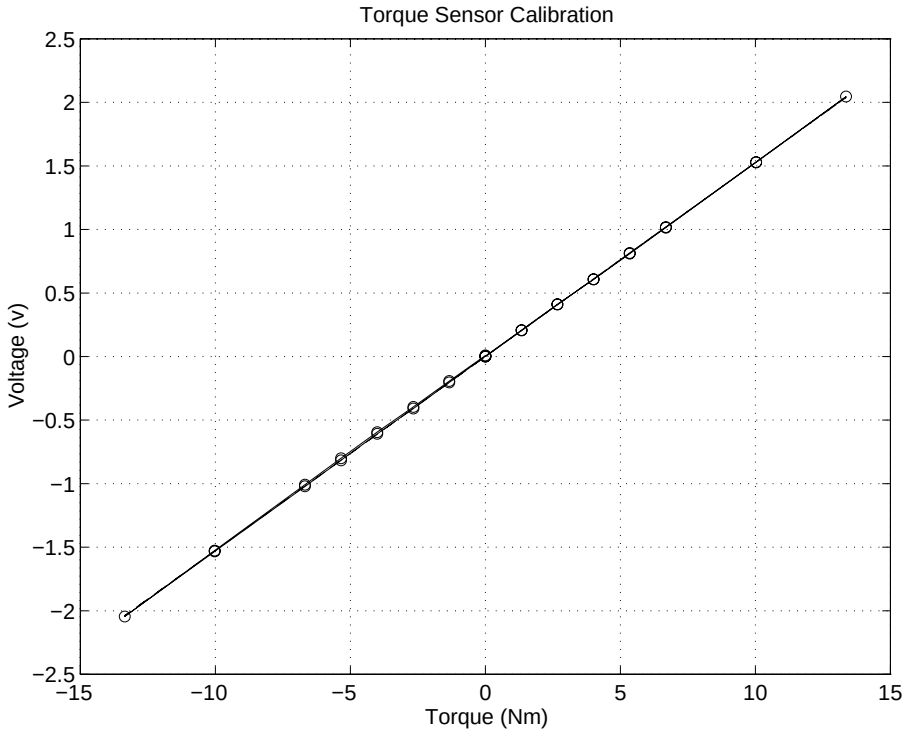

Figure 6. Torque sensor calibration results for low-range loading

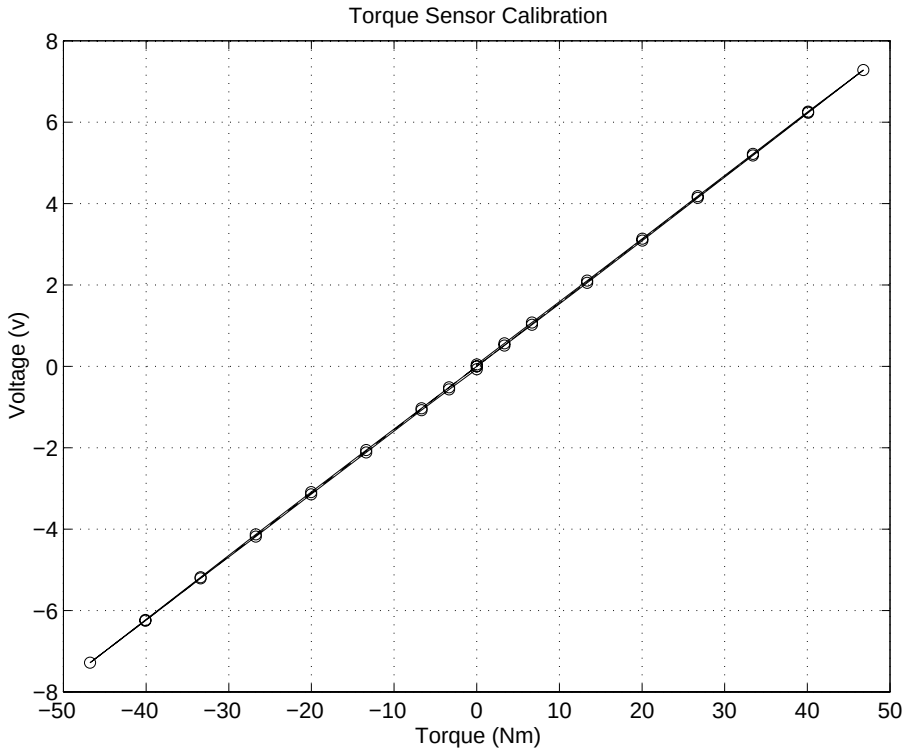

FIGURE 7. Torque sensor calibration results for full-range loading

given in Table 1, where the gain is 6.6 Nm/volts, with a low consistency measure of 2%. This illustrates that the torque sensor is quite linear, and it is not position dependent. Moreover, the torque sensor gain is consistent for different operating conditions.

TABLE 1. Torque sensor calibration results

Torque Sensor Gain	Consistency Measure
6.60 Nm/Volts	2%

4. Torque Ripple Compensation

One important characteristic of harmonic drive torque transmission as observed in free motion experiments, is a high frequency oscillation in output torque signal (See torque curve in Figure 8). These oscillation, named torque ripples, were also observed by other researchers [1]. Torque ripples are caused mainly by harmonic drive gear meshing vibration. Harmonic drive gear meshing vibration introduces a real torque oscillation which can be observed in the end effector motions of robots using harmonic drives and even sensed by hand when back–driving the harmonic drive. Its principal frequency of oscillation (in rad/sec) is twice the motor velocity (in rad/sec), for the gear teeth in harmonic drives are meshing in two zones. A small fraction of the torque ripples are caused by the non-ideal torque measurement, because of the direct attachment of strain gauges on the flexspline. Since the flexspline has an elliptical shape, strain gauges mounted on the flexspline are subjected to unwanted strain caused by the elliptical shape. Hashimoto [3], proposed using at least two pairs

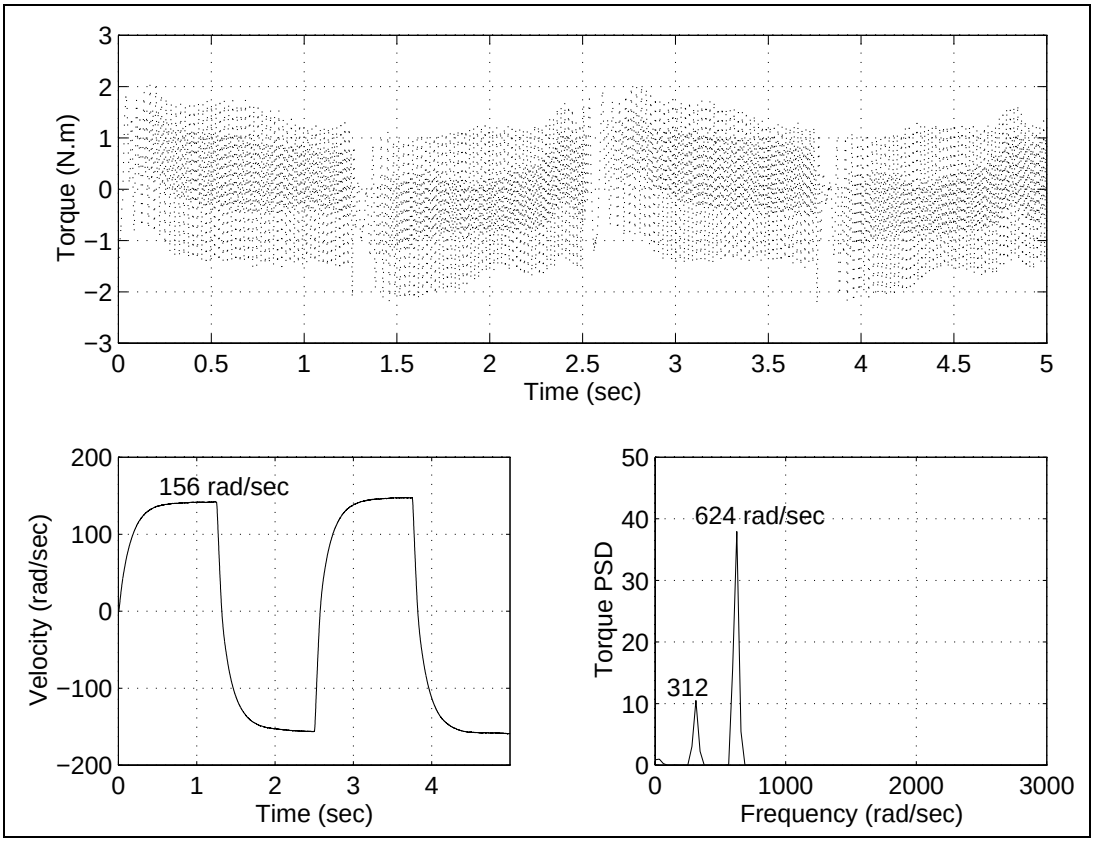

Figure 8. Measured Torque and velocity and the power spectrum of the measured torque with peaks at multiples of the velocity

of Rosette strain gauges to compensate for this unwanted strain. However, ideal compensation is possible only if there is no positioning error of strain–gauges. As explained in § 2 it has been shown however, that using four Rosette strain–gauges, and using an accurate method to mount the strain–gauges will reduce the amplitude of the torque ripple to a minimum. Unfortunately the frequency of torque ripples (in rad/sec) introduced by the non-ideal behavior of the sensor is also twice the motor speed (in rad/sec), since the major axis of the ellipse is travelling twice as fast as the wave generator. This make it impossible to discern the true ripples caused by the gear meshing vibration from that caused by non-ideal measurement. As illustrated in Figure 8, the power spectrum of the measured torque plotted for the time interval 0.8 to 1.1 sec when the velocity is almost flat and about 156 rad/sec shows two peaks at 312, and 624 rad/sec. This confirms the existence of the fundamental frequency of the oscillation as twice the velocity and shows the significance of the next important first-harmonic frequency of four times the velocity. The dependence of the frequency content of the torque ripples on the velocity makes it possible to model them as a simple harmonic oscillator, and to employ a Kalman filter to predict and filter them from the torque measurement. If only low–frequency torque control is desired [12], the high–frequency ripples may be removed by estimating them via Kalman filtering. This is more efficient than simple low–pass filtering, because it uses the known structure of the torque signal.

A fourth–order harmonic oscillator error model can characterize both the fundamental and first-harmonic frequency content of the torque ripple [16]. This can be represented by the following time–varying discrete state space form:

$$\begin{cases} \mathbf{x}(k+1) &= \left[\begin{array}{c|c} \Phi_1(k) & \mathbf{0} \\ \hline \mathbf{0} & \Phi_2(k) \end{array}\right] \mathbf{x}(k) + \mathbf{w}(k) \\ \\ y(k) &= [\ 1 \quad 0 \quad 1 \quad 0\ ]\mathbf{x}(k) + v(k) \end{cases} \tag{5}$$

where

$$\Phi_i(k) = \begin{bmatrix} \cos(\omega_i(k)T_s) & \sin(\omega_i(k)T_s) \\ -\sin(\omega_i(k)T_s) & \cos(\omega_i(k)T_s) \end{bmatrix} \tag{6}$$

in which T_s is the sampling period, and $\omega_1(k) = 2\dot{\theta}(k)$, $\omega_2(k) = 4\dot{\theta}(k)$, and $\dot{\theta}(k)$ is the motor shaft velocity in rad/sec at time step k, hence $\Phi_i(k)$ is a time–varying matrix adapting with changes in the velocity. Moreover, y is the torque ripple, which needs to be observed from the torque measurement by having a crude model for the

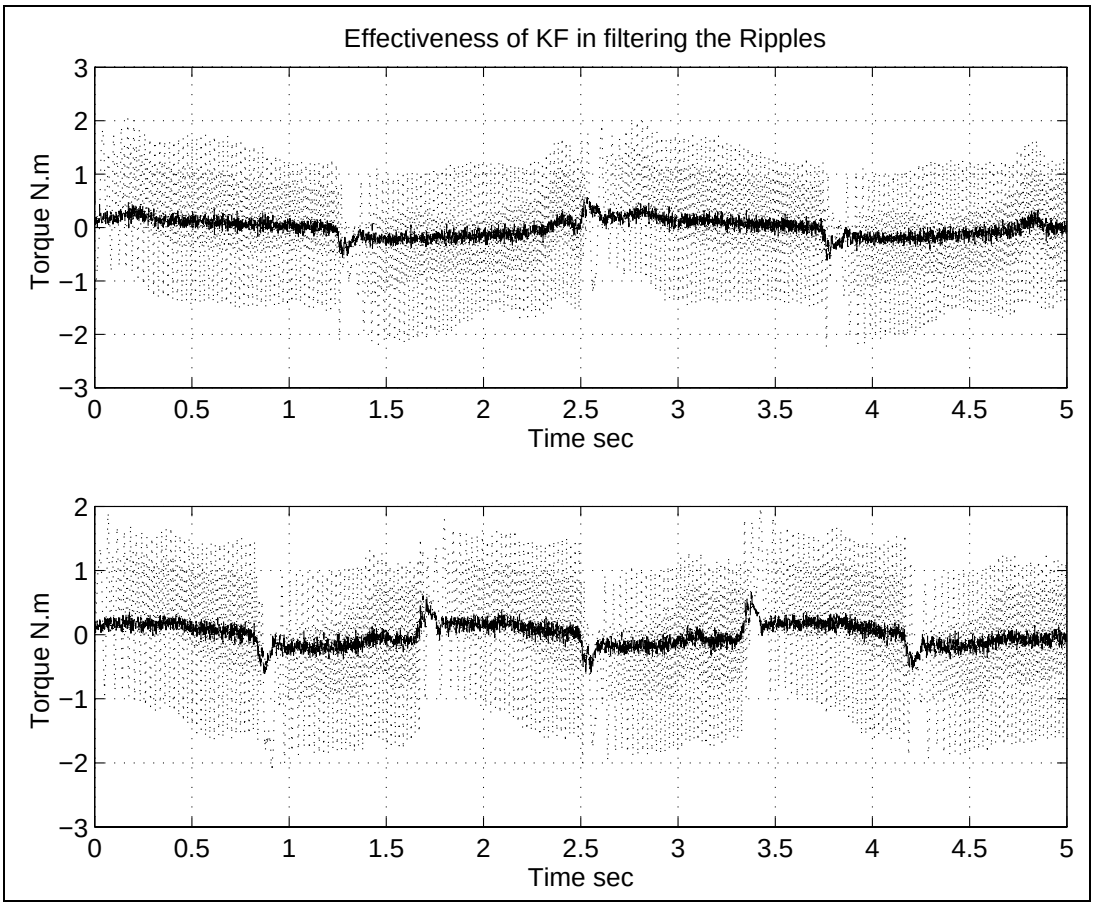

FIGURE 9. Kalman filter performance to cancel torque ripples. Dotted: Measured torque, Solid: Kalman filtered torque

expected torque. If measured torque is indicated by T_{meas}, and the expected torque is indicated by T_{exp} then torque ripple $y(k) = T_{meas}(k) - T_{exp}(k)$, while the difference between true torque and the crude estimate of the expected torque is encapsulated by the measurement errors in $v(k)$ in Equation 5. Therefore, no accurate model for the expected torque T_{exp} is necessary and hence, for free motion experiments the expected torque can be estimated simply by the inertial part of the output torque which is the load inertia multiplied to the load acceleration. The first two elements of the state $\mathbf{x}$ consists of the component of the torque ripple due to the fundamental frequency ω_1, and its derivative, while the next two elements are those components of torque ripple due to the first-harmonic frequency ω_2, and its derivative. Therefore, the total torque ripple is calculated by adding the first and third element of the state. $\mathbf{w}(k)$ characterizes the other frequency components of the torque ripples which will not be estimated in this model.

Using the fourth–order harmonic oscillator model for the torque ripples, a prediction–type time–varying Kalman filter algorithm is applied to estimate the torque ripples [7]. A Prediction–type estimate is computationally faster than a current–estimate type of Kalman filter, and therefore preferable for online implementation. Assuming that the process noise $\mathbf{w}(k)$ and measurement noise $v(k)$ are zero–mean Gaussian white and have covariances defined by $\mathbf{Q}$ and R as:

$$E\{\mathbf{w}(k)\mathbf{w}^T(k)\} = \mathbf{Q} \quad , \quad E\{v(k)v^T(k)\} = R \tag{7}$$

the states estimates is calculated using Kalman filter formulation [7] as following:

$$\hat{\mathbf{x}}(k+1) = \Phi(k)\hat{\mathbf{x}}(k) + \mathbf{K}_e(k)\left[\mathbf{y}(k) - \mathbf{C}(k)\hat{\mathbf{x}}(k)\right] \tag{8}$$

in which the Kalman filter gain $\mathbf{K}_e(k)$ will be updated from,

$$\mathbf{K}_e(k) = \Phi(k)\mathbf{P}(k)\mathbf{C}^T(k)\left[R(k) + \mathbf{C}(k)\mathbf{P}(k)\right]\mathbf{C}^T(k)\Big]^{-1} \tag{9}$$

$$\mathbf{P}(k+1) = \mathbf{Q}(k) + \left[\Phi(k) - \mathbf{K}_e(k)\mathbf{C}(k)\right]\mathbf{P}(k)\Phi^T(k) \tag{10}$$

Since the measurement signal $y(k)$ is a scalar, its covariance matrix $R(k)$ is also a scalar, and no matrix inversion is required for online implementation of the Kalman filter gain given in Equation 9. Figure 9 illustrates the performance of the Kalman filter implemented on line with a sampling frequency of 1 kHz to estimate and filter the torque ripples for two typical experiments, in which $\mathbf{Q} = 10^3\ \mathbf{I}_{4\times4}$ and $R = 1$. The performance of the Kalman filter for torque ripple cancellation is shown to be quite fast and accurate.

5. Misalignment Torque Compensation

The Kalman filter can be used not only to estimate the torque ripples, but also to cancel any mechanical misalignment torque signature on the measured torque. Many torque sensors exhibit the limitation of being sensitive to the torques applied on the direction perpendicular to their axis of measurement. In our setup after repeated use of the harmonic drive system for different experiments, a similar torque signature was observed on the measured torque. Figure 10 illustrates a simple experiment in which the harmonic drive is driven by a constant velocity but the measured torque exhibit a sinusoidal trend. The expected torque, the solid line in Figure 10, is constant after a short time of acceleration, but the measured torque, dotted lines, displays a sinusoidal behavior. After examining the system accurately, the source of this torque signature is found to be the misalignment of the harmonic drive shaft and the load. By disassembling the system and carefully reassembling it, the peak to peak amplitude of this misalignment signature was reduced from 10 N.m to less than 2 N.m. However, in practice it is quite expensive, and probably infeasible to perfectly align all the moving components. Fortunately, the sinusoidal feature of this misalignment torque makes it possible to accurately estimate them with an error model. The frequency of misalignment torque(in rad/sec), as it can be intuitively identified from its source of generation, is exactly the same as the output shaft velocity (in rad/sec). Therefore, adding another block to the harmonic oscillator model (given in Equation 5) of the system with frequency $\omega_3(k) = vel(k)/(\text{Gear Ratio})$, will estimate the misalignment component of the measured torque. Using this sixth order model for the torque ripple and misalignment torque together, Figure 11 illustrates the Kalman filter performance in cancelling those elements for two typical experiments.

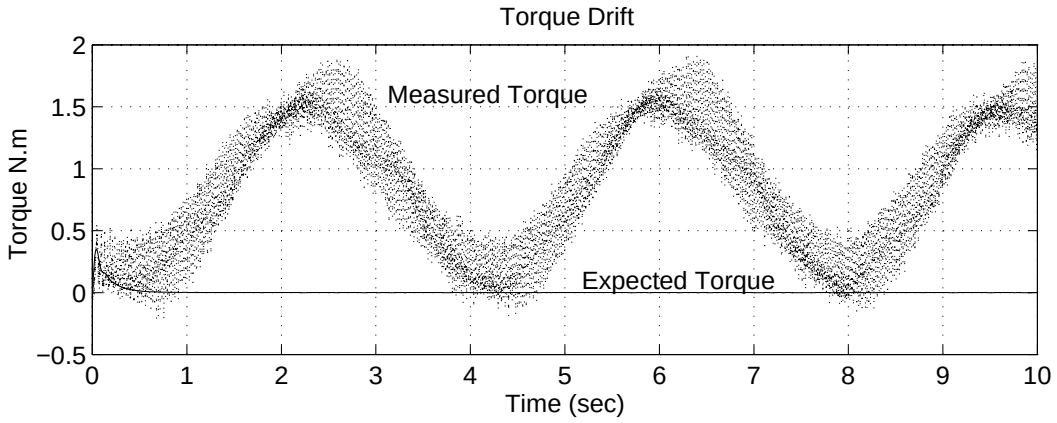

Figure 10. Misalignment torque signature on the measured torque

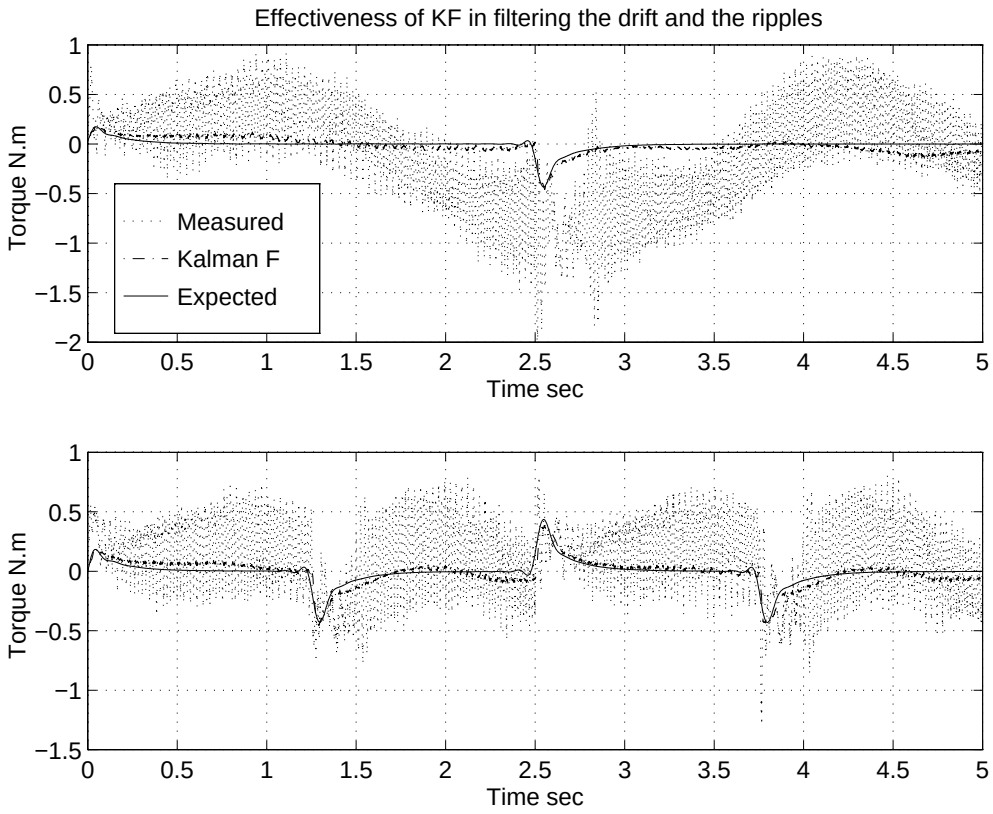

FIGURE 11. Kalman filter performance to cancel torque ripples, and misalignment torque for two typical experiments

The Kalman filtered torque is shown to cancel the torque ripples and misalignment torque quite accurately. These results are obtained using an online Kalman filter implementation on the system with sampling frequency of 1 kHz. In contrary to low–pass filtering which is incapable of extracting the misalignment torque from the measurements, Kalman filter estimation proved to be fast and accurate to filter both torque ripples and misalignment torque. Moreover, it is a reliable method for different operating ranges, and therefore, preferable for torque feedback.

6. Conclusions

In this paper the built–in torque sensor for harmonic drive systems as first proposed by Hashimoto is examined in detail. By this method strain–gauges are directly mounted on the flexspline, and therefore, no extra flexible element is introduced into the system. To have minimum sensing inaccuracy, four Rosette strain gauges are employed using an accurate method of positioning. An accurate drawing of the strain–gauge placement positions is printed on a transparent film, and the strain-gauges are placed on the transparent film using a microscope. Then the transparent film is accurately placed on the flexspline, and the strain–gauges are cemented on the surface. It is shown that employing this method reduces the positioning error to a minimum. Calibration of the torque sensor shows that the sensor is performing linearly and the torque readings are not dependent to the position of the flexspline.

One important characteristic of harmonic drive torque transmission, as observed in free motion experiments, is a high frequency oscillation in the output torque signal. To cancel these oscillation named torque ripples from the measured torque, Kalman filter estimation is employed. Due to the dependence of the frequency content of the torque ripples on the wave generator velocity, a simple forth order harmonic oscillator proved to accurately model the torque ripples. The performance of Kalman filter to cancel the torque ripples from torque measurements is shown to be very fast and accurate. Moreover, the error model is extended to incorporate any misalignment torque signature. By on line implementation of the Kalman filter incorporating a sixth order model, it is shown that this method is a fast and accurate way to filter torque ripples and misalignment torque, and hence, this intelligent built–in torque sensor is preferable for torque feedback.

References

[1] I. Godler, K. Ohnishi, and T. Yamashita. Repetitive control to reduce speed ripple caused by strain wave gearing. *IECON Proceedings*, 2:1034–38, 1994.

[2] M. Hashimoto. Robot motion control based on joint torque sensing. *Proceeding of IEEE International Conference on Robotics and Automation*, pages 256–261, 1989.

[3] M. Hashimoto, Y. Kiyosawa, H. Hirabayashi, and R.P. Paul. A joint torque sensing technique for robots with harmonic drives. *Proceeding of IEEE International Conference on Robotics and Automation*, 2:1034–1039, April 1991.

[4] M. Hashimoto, Y. Kiyosawa, and R.P. Paul. A torque sensing technique for robots with harmonic drives. *IEEE Transaction on Robotics and Automation*, 9(1):108–116, Feb 1993.

[5] R. Hui, N. Kircanski, A. Goldenberg, C. Zhou, P. Kuzan, J. Weircienski, D. Gershon, and P. Sinha. Design of the iris facility - a modular, reconfigurable and expandable robot test bed. *Proceeding of IEEE International Conference on Robotics and Automation*, 1:155–160, 1993.

[6] S. Nicosia and P. Tomei. On the feedback linearization of robots with elastic joints. *Proceeding of IEEE Conference on Decision and Control*, 1:180–185, 1988.

[7] K. Ogata. *Discrete - time control systems*. Prentice Hall, Englewood cliffs, N.J., 1995.

[8] M.W. Spong. Adaptive control of flexible joint manipulators. *Systems & Control Letters*, 13(1):15–21, Jul 1989.

[9] M.W. Spong, J.Y. Hung, S. Bortoff, and F. Ghorbel. Comparison of feedback linearization and singular perturbation techniques for the control of flexible joint robots. *Proceedings of the American Control Conference*, 1:25–30, 1989.

[10] D. Stokic and M. Vukobratovic. Historical perspectives and state of the art in joint force sensory feedback control of manipulation robots. *Robotica*, 11(2):149–157, Mar-Apr 1993.

[11] H.D. Taghirad and P.R. Belanger. An experimental study on modelling and identification of harmonic drive systems. *Proceeding of IEEE Conference on Decision and Control*, 4:4725–30, Dec 1996.

[12] H.D. Taghirad and P.R. Belanger. Intelligent torque sensing and robust torque control of harmonic drive under free–motion. *To be presented in the 1997 IEEE International Conference on Robotics and Automation*, April 1997.

[13] H.D. Taghirad and P.R. Belanger. Robust torque control of harmonic drive under constrained motion. *To be presented in the 1997 IEEE International Conference on Robotics and Automation*, April 1997.

[14] H.D. Taghirad, P.R. Belanger, and A. Helmy. An experimental study on harmonic drive. *Technical Report Submitted to the International Submarine Engineering Ltd., Port Coquitlam, BC, Canada*, http://www.cim.mcgill.ca/~ taghirad, 1996.

[15] H.D. Taghirad, A. Helmy, and P.R. Belanger. Intelligent built-in torque sensor for harmonic drive system. *Submitted to the IEEE Transactions on Instrumentation and Measurement*, Dec 1996.

[16] H.D. Taghirad, A. Helmy, and P.R. Belanger. Intelligent built-in torque sensor for harmonic drive system. *To be presented in the 1997 IEEE Instrumentation and Measurement Technology Conference*, May 1997.

Appendix A. Software Code Listings

A.1. **Torque Ripple Estimation by Kalman Filter.**

```
/*-----------------------------------------------------------------------
 *  FILE: free_kalman.c
 *
 *  Calling Sequence:
 *
 *  Function: 1- C30 node "C" program to respond to C30V timer-generated
 *    interrupts using timer 0 and sample a2d inputs and the pulse count
 *    from the quadrature decoder.
 *            2- encoder pulse count acqusition.
 *            3- Torque ripples filtered by Kalman Filter
 *
 *  Revision Information:
 *       Date      By   Changes Made....
 *
 *       26/02/92  ptd  Cleaned up and documented.
 *       11/03/92  ptd  Expanded from 2 channels to four.
 *       24/02/94  ptd  DT140X analog input boards; float functionalities.
 *       01/11/95  hdt  Butterworth filter for torque
 *       18/04/96  hdt  Differentiator Filter for acceleration
 *       18/04/96  hdt  Kalman filter estimation for torque ripples
 *
  ********************************************************************** */

#include "DT140X.h"         /* Analog-to-digital converter definitions */
#include <math.h>                     /* Sine-wave generator functions */
#include "node_map.h"                  /* C30 node address definitions */
#include "stypes.h"                         /* Boolean type definition */
#include "servo_refgen.h"      /* Servo reference generation functions */
#include "servo_conv.h"            /* Servo variable value definitions */

#define NUM_CHANNELS 4
#define NUM_MISC_FIELDS 1
#define FIELD_QTY  (NUM_CHANNELS+NUM_MISC_FIELDS)
                                      /* Number of fields per sample */
#define NUM_A2D_CHAN    DT1401_A2D_CHANNELS
#define SIGN(X)         ((X) <  0  ? -1  :  1 )

int *time_sample_ptr = (int *) 0xc23000;        /* Sample data storage */
short *mbox = (short *) 0xe0;                              /* Mail boxes */
float *parm_ptr =  (float *) 0xc000c0;  /* Parameter passing mailboxes */
float *sample_ptr =  (float *) 0xc03000;        /* Sample data storage */
float Namplitude = 0.0, Vamplitude = 0.0;
float Nbias = 0.0, Vbias = 0.0;
float theta = 0, freq = 0 ; /* Angle and angular frequency of function */
float rise = 0.0;
float fall = 0.0;
float duty = 1.0;        /* duty cycle ratio for some reference inputs */

void c_int10();
float Diff_filter();
```

```
float Kalman();

/* Constant needed for Subroutines */
int kal_count = 1;
float v_1=0.0, a_1=0.0;
float volt2Nm = 6.6, volt2radps = 56.2565;
float J_load = 1.725e-2, Gain = 100;
float T_ripple = 0.0;

DT140XChan
  str_ch = DT140X_CH3,
  end_ch = DT140X_CH6;
int numchannels = NUM_CHANNELS;              /* number of a2d channels */
int wavetype = 0;                      /* no wave generation by default */
int samplecount = 0;                                  /* sample count */
float sample_period = 0.001;               /* Sample period in decimal */
int time_samplecount = 0;

/* ************************************************************** */
void main()
{
  Bool Samples_Ready = True;
  int channel, pause, numsamples;
  float t1period = 514.0;            /* max. allowable Timer period in */
                                                        /* seconds */
  copy(parm_ptr,   &Nbias, 1, 0x0000, 21);               /* GRAM->NRAM */
  copy(parm_ptr+2, &sample_period, 1, 0x0000, 21);       /* GRAM->NRAM */
  copy(parm_ptr+3, &numsamples, 1, 0x0000, 21);          /* GRAM->NRAM */
  copy(parm_ptr+4, &Namplitude, 1, 0x0000, 21);          /* GRAM->NRAM */
  copy(parm_ptr+5, &wavetype, 1, 0x0000, 21);            /* GRAM->NRAM */
  copy(parm_ptr+6, &freq, 1, 0x0000, 21);                /* GRAM->NRAM */
  copy(parm_ptr+7, &duty, 1, 0x0000, 21);                /* GRAM->NRAM */

/* Initialize the hardware "my addition" */

 xvme085_init();      /* Prototype board encoder interface initialized */
 init_c30v_bus();                    /* C30 node and buses initialized */
 c30_led_on();                      /* Port configuration reset, mode 0 */
 t0_schedule_intr(c_int10, sample_period);  /* CPU interrupt level 0x9 */
 enable_ie_int3();                /* CPU external interrupt #3 enabled */

  Vbias = PC2V_linear_conv( Nbias );
  Vamplitude = PC2V_linear_conv( Namplitude );
  setup_dac_csr();
  write_dac0( Vbias );

  /* Allow for mech. time constant */
  for (pause=0; pause < 0xffff; pause++);

  /* setup a2d converter; sample rate & mapping */
  DT140XInit(str_ch, end_ch);
  t1_schedule_intr(c_int10,sample_period);
  enable_intr();
  t1_start();

  while( samplecount < numsamples  ){}
  disable_intr();
```

```
  copy(&time_samplecount,(parm_ptr + 10), 1, 0x0000, NRAM_2_GRAM);
  copy(&Samples_Ready,(parm_ptr +1),1,0x0000, 16); /* NRAM->GRAM */
  write_dac0(0.0);
  c30_led_sflash();
}
/* ********************** Misc Sub_Routines ************************** */

void c_int10()
{
  int *t1_counter_reg = (int *) T1_CNTR_REG;
  /* timemark = blank dummy     */
  int  channel, timecnt, timemark = 0xffffff0f;
  /* current set of resolved values */
  float a2dsamples[NUM_A2D_CHAN];
  float Tacho, MotorCurrOut, Torque, Torque_f, ref_fl;
  float vel, acc, T_meas, T_mod, Delta_T, out;
  short pcsample;
  short EncoderPulses;
  int Nref;

  t1_stop();
  timecnt = *t1_counter_reg;
  t1_start();
  copy( PB_LAT_RD_ADDR,&pcsample,1,0xca00,VME_2_NRAM);

  DT1401Read( str_ch, end_ch, a2dsamples);

  Tacho        = a2dsamples[DT140X_CH4];
  MotorCurrOut = a2dsamples[DT140X_CH5];
  Torque       = a2dsamples[DT140X_CH6];

  vel          = Tacho * volt2radps;  /* Calibration volt -->  rad/sec */
  acc          = Diff_filter( vel);   /* Differentiation of velocity   */
  T_meas       = Torque * volt2Nm ;   /* Calibration volt --> Nm       */
  T_mod        = J_load * acc / Gain; /* Evaluatind Modelled Torque    */
  Delta_T      = T_meas - T_mod;      /* Torque Error                  */
  Torque_f     = T_meas - T_ripple;    * Final Filtered Torque         */
  T_ripple     = Kalman (Delta_T, vel, sample_period);
                                  /* Finding torque ripple for next
                                       iteration: Perdiction-Type KF */
  ref_fl=reference_generation(wavetype);/*  floatV for MAX-100 servo   */
  write_dac0( ref_fl );

/* writing the wanted variables into Gram for further use */
    copy(&T_meas,       sample_ptr++,1,0x0000, NRAM_2_GRAM);
    copy(&vel,          sample_ptr++,1,0x0000, NRAM_2_GRAM);
    copy(&MotorCurrOut, sample_ptr++,1,0x0000, NRAM_2_GRAM);
    copy(&Torque_f,     sample_ptr++,1,0x0000, NRAM_2_GRAM);

/*********************************************************************/

  /* Store the interpulse time data */
  copy(&timecnt,time_sample_ptr++,1,0x0000,NRAM_2_GRAM); /* NRAM->GRAM */
  /* Process and store the pulse count data */
  pcsample &=  0x0000ffff;         /* Avoid hi/lo duplication of short */
  copy(&pcsample, sample_ptr++,1,0x0000, NRAM_2_GRAM);   /* NRAM->GRAM */
```

```
  EncoderPulses = pcsample;
  copy(&timemark,time_sample_ptr++,1,0x0000,NRAM_2_GRAM);
  time_samplecount = time_samplecount + 3;
  ++samplecount;
  enable_intr();
} /* c_int10() */

/********************** Differentiator Filter ************************
 *
 * Discretized Differentiotor Filter H(s) =  s / (0.005 s + 1)
 * Disretized by tustin method for Ts = 1 ms
 *
 * Function Calls : a = Diff_filter (v)
 *
 *    input :       v = velocity     (rad/s)
 *    output:       a = acceleration (rad/s/s)
 *********************************************************************/

float Diff_filter (v)
float v;
{
  float nd[2]={
    181.818181818182,
   -181.818181818182
  };
  float dd[2]={
    1.00000000000000,
   -0.81818181818182
  };
  float a;

  a=nd[0]*v + nd[1]*v_1 - dd[1]*a_1;
  v_1 = v; a_1 = a;

  return(a);
}
/* ***************************** Kalman ****************************
 *
 *       Kalman Filter estimation for HD Torque ripples
 *       function calls
 *
 *       x = KALMAN ( y , v , T)
 *
 *            y : T_meas - T_mod    ( your estimate of the torque error)
 *            v : velocity (rad/sec)
 *            T : Sampling Period (sec)
 *            x : The estimate of the torque ripple
 *
 *      Algorithm:
 *
 *     A second order harmonic oscillator, with two harmonics of
 *     oscillation equal to twice and for times velocity is the model
 *     of the torque ripples. Time varying Kalman Filter is considered
 *     to estimate the torque ripple.
 *
 *      Last Update : April 18, 1996
 *      By: Hamid D. Taghirad
 *
```

```
**************************************************************************/

float Kalman( Delta_T, velocity, sample_period)

float Delta_T;
float velocity;
float sample_period;

{
  int  i, j, k;
  float om1, om2, K[4], PC[4], pPC[4], CPC;
  float phi [4][4], Pp[4][4], KC[4][4], p_KC[4][4], gb[4][4];
  float x1[4], Torque_ripple;
  float x[4], C[4], R, P[4][4], Q[4][4];

  /* Matrix Initialization */

  if (kal_count == 1)    /* only true for first iteration */
  {
    R = 1.0;
   for(i=0; i<4; ++i)
   {
     x[i] = 0.0;
     C[i] = 0.0;
     for(j=0; j< 4; ++j)
       {
         if ( i == j )
         {
           P[i][j] = 1.0;
           Q[i][j] = 1000.0;
                     }
         else
         {
           P[i][j] = 0.0;
           Q[i][j] = 0.0;
                     }
       }
   }
    x[0] = x [2] = Delta_T;
    C[0] = C [2] = 1.0;

  ++kal_count;
  }
  /* Two harmonics of the Torque ripples * Sample Period */
  om1 = 2.0 * velocity * sample_period;
  om2 = 2.0 * om1;

  /* System Model of Harmonic Oscillator 4th order */
  for(i=0; i< 4; ++i)
  {
    for(j=0; j<4; ++j)
      phi[i][j]=0.0;
  }
  phi[0][0] = cos(om1);
  phi[0][1] = sin(om1);
  phi[1][0] =-sin(om1);
  phi[1][1] = cos(om1);
```

```
  phi[2][2] = cos(om2);
  phi[2][3] = sin(om2);
  phi[3][2] =-sin(om2);
  phi[3][3] = cos(om2);

/* ****************************** 1 ***************************** */
/* Gain update K */

  /* Matrix Multiplication PC(4x1) = P(4x4) * C'(4x1) */
  for ( i=0; i < 4; ++i)
  {
    PC[i] = 0.0;
    for(j=0; j < 4; ++j)
      PC[i] += P[i][j] * C[j];
  }

  /* Matrix Multiplication pPC(4x1) = phi(4x4) * PC(4x1) */
  for ( i=0; i < 4; ++i)
  {
    pPC[i] = 0.0;
    for(j=0; j < 4; ++j)
      pPC[i] += phi[i][j] * PC[j];
  }

  /* Matrix Multiplication CPC(1x1) = C(1x4) * PC(4x1) */
    CPC = 0.0;
    for(j=0; j < 4; ++j)
      CPC += C[j] * PC[j];

  /* Final Gain Calculation K(4x1) = phi P C' / ( R + C P C')
   * Note that (R + C P C') is scalar --> NO Matrix inversion
   */

  for ( i=0; i < 4; ++i)
    K[i] = pPC[i] / ( R + CPC);

/* ****************************** 2 ***************************** */
/* P Update */


  /* Matrix Multiplication Pp(4x4) = P(4x4) * phi'(4x4) */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
      {
        Pp[i][j] = 0.0;
        for(k = 0;  k < 4 ; ++k)
          Pp[i][j] += P[i][k] * phi[j][k];
      }
  }

  /* Matrix Multiplication KC(4x4) = K(4x1) *C(1x4) */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
        KC[i][j] = K[i] * C [j];
```

```
  }

  /* Matrix Addition  p_KC(4x4) = ( phi(4x4) - KC(4x4)) */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
        p_KC[i][j] = phi[i][j] - KC[i][j];
  }

/* Matrix Multiplication gb(4x4) =(phi - K C)P phi'=p_KC(4x4)*Pp(4x4) */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
      {
        gb[i][j] = 0.0;
        for(k = 0;  k < 4 ; ++k)
          gb[i][j] += p_KC[i][k] * Pp[k][j];
      }
  }

  /* Final formula for P(4x4) = Q + (phi - K C) P phi' = Q + gb */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
        P[i][j] = Q[i][j] + gb[i][j];
  }

/* ****************************** 3 ****************************** */
/* x Update, and output generation */


/* Matrix Multiplication x1(4x1) = ( phi -KC ) * x = p_KC(4x4) * x(4x1) */
  for ( i=0; i < 4; ++i)
  {
    x1[i] = 0.0;
    for(j=0; j < 4; ++j)
      x1[i] += p_KC[i][j] * x[j];
  }
/* Matrix Multiplication x(4x1) = (phi -KC) x+K y=K(4x1)*y(1x1)+x1(4x1) */
  for ( i=0; i < 4; ++i)
     x[i] = K[i] * Delta_T + x1[i];

  /* Output generation y = C x = x(1) + x(3)  */
  Torque_ripple = x[0] + x[2] ;

  return(Torque_ripple);
}

/* -- end of file free_kalman.c --*/
```

A.2. **Torque Ripple and Misallignment Torque Estimation.**

```
/* ***************************** Kalman ******************************
 *
 *       Kalman Filter estimation for HD Torque ripples
 *       function calls
 *
 *       x = KALMAN ( y , v , T)
 *
 *            y : T_meas - T_mod    ( your estimate of the torque error)
 *            v : velocity (rad/sec)
 *            T : Sampling Period (sec)
 *            x : The estimate of the torque ripple
 *
 *      Algorithm:
 *
 *      A second order harmonic oscillator, with two harmonics of
 *      oscillation equal to twice and for times velocity is the model
 *      of the torque ripples. Time varying Kalman Filter is considered
 *      to estimate the torque ripple.
 *
 *
 *      Last Update : April 18, 1996
 *      By: Hamid D. Taghirad
 *
*************************************************************************/

float Kalman( Delta_T, velocity, sample_period)

float Delta_T;
float velocity;
float sample_period;

{
  int  i, j, k;
  float om1, om2, K[4], PC[4], pPC[4], CPC;
  float phi [4][4], Pp[4][4], KC[4][4], p_KC[4][4], gb[4][4];
  float x1[4], Torque_ripple;
  float x[4], C[4], R, P[4][4], Q[4][4];

  /* Matrix Initialization */

  if (kal_count == 1)    /* only true for first iteration */
  {
    R = 1.0;
   for(i=0; i<4; ++i)
   {
     x[i] = 0.0;
     C[i] = 0.0;
     for(j=0; j< 4; ++j)
       {
         if ( i == j )
         {
           P[i][j] = 1.0;
           Q[i][j] = 1000.0;
                     }
         else
```

```
        {
          P[i][j] = 0.0;
          Q[i][j] = 0.0;
                    }
      }
   }
    x[0] = x [2] = Delta_T;
    C[0] = C [2] = 1.0;
    Q[2][2] = Q[3][3] = 5e-2;

  ++kal_count;
  }


  /* Two harmonics of the Torque ripples * Sample Period */
  om1 = 2.0 * velocity * sample_period;
  om2 = velocity * sample_period / 100.0; /* 100 == HD Gear ratio */

  /* System Model of Harmonic Oscillator 4th order */
  for(i=0; i< 4; ++i)
  {
    for(j=0; j<4; ++j)
      phi[i][j]=0.0;
  }

  phi[0][0] = cos(om1);
  phi[0][1] = sin(om1);
  phi[1][0] =-sin(om1);
  phi[1][1] = cos(om1);

  phi[2][2] = cos(om2);
  phi[2][3] = sin(om2);
  phi[3][2] =-sin(om2);
  phi[3][3] = cos(om2);


/* ****************************** 1 ***************************** */
/* Gain update K */

  /* Matrix Multiplication PC(4x1) = P(4x4) * C'(4x1) */
  for ( i=0; i < 4; ++i)
  {
    PC[i] = 0.0;
    for(j=0; j < 4; ++j)
      PC[i] += P[i][j] * C[j];
  }

  /* Matrix Multiplication pPC(4x1) = phi(4x4) * PC(4x1) */
  for ( i=0; i < 4; ++i)
  {
    pPC[i] = 0.0;
    for(j=0; j < 4; ++j)
      pPC[i] += phi[i][j] * PC[j];
  }

  /* Matrix Multiplication CPC(1x1) = C(1x4) * PC(4x1) */
```

```
    CPC = 0.0;
    for(j=0; j < 4; ++j)
      CPC += C[j] * PC[j];

  /* Final Gain Calculation K(4x1) = phi P C' / ( R + C P C')
   * Note that (R + C P C') is scalar --> NO Matrix inversion
   */

  for ( i=0; i < 4; ++i)
    K[i] = pPC[i] / ( R + CPC);

/* ****************************** 2 ****************************** */
/* P Update */


  /* Matrix Multiplication Pp(4x4) = P(4x4) * phi'(4x4) */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
      {
        Pp[i][j] = 0.0;
        for(k = 0;  k < 4 ; ++k)
          Pp[i][j] += P[i][k] * phi[j][k];
      }
  }

  /* Matrix Multiplication KC(4x4) = K(4x1) *C(1x4) */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
        KC[i][j] = K[i] * C [j];
  }

  /* Matrix Addition  p_KC(4x4) = ( phi(4x4) - KC(4x4)) */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
        p_KC[i][j] = phi[i][j] - KC[i][j];
  }

/* Matrix Multiplication gb(4x4) =(phi - K C)P phi'=p_KC(4x4)*Pp(4x4) */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
      {
        gb[i][j] = 0.0;
        for(k = 0;  k < 4 ; ++k)
          gb[i][j] += p_KC[i][k] * Pp[k][j];
      }
  }

  /* Final formula for P(4x4) = Q + (phi - K C) P phi' = Q + gb */
  for ( i=0; i < 4; ++i)
  {
    for(j=0; j < 4; ++j)
        P[i][j] = Q[i][j] + gb[i][j];
```

```
  }

/* ****************************** 3 ***************************** */
/* x Update, and output generation */


/* Matrix Multiplication x1(4x1) = ( phi -KC ) * x = p_KC(4x4) * x(4x1) */
  for ( i=0; i < 4; ++i)
  {
    x1[i] = 0.0;
    for(j=0; j < 4; ++j)
      x1[i] += p_KC[i][j] * x[j];
  }
/* Matrix Multiplication x(4x1) = (phi -KC) x+K y=K(4x1)*y(1x1)+x1(4x1) */
  for ( i=0; i < 4; ++i)
     x[i] = K[i] * Delta_T + x1[i];

  /* Output generation y = C x = x(1) + x(3)  */
  Torque_ripple = x[0] + x[2] ;

  return(Torque_ripple);
}
/* ---------------------- End of Module ------------------------------ */
```

Center for Intelligent Machines, McGill University, 3480 University St., Montréal (Québec) H3A 2A7, Canada, *Tel.* : (514) 398-8202
E-mail address: taghirad@cim.mcgill.ca