International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:3, No:5, 2009

A Tutorial on Dynamic Simulation of DC Motor and Implementation of Kalman Filter on a Floating Point DSP

Padmakumar S., Vivek Agarwal, and Kallol Roy

***Abstract*—**With the advent of inexpensive 32 bit floating point digital signal processor's availability in market, many computationally intensive algorithms such as Kalman filter becomes feasible to implement in real time. Dynamic simulation of a self excited DC motor using second order state variable model and implementation of Kalman Filter in a floating point DSP TMS320C6713 is presented in this paper with an objective to introduce and implement such an algorithm, for beginners. A fractional hp DC motor is simulated in both Matlab® and DSP and the results are included. A step by step approach for simulation of DC motor in Matlab® and "C" routines in CC Studio® is also given. CC studio® project file details and environmental setting requirements are addressed. This tutorial can be used with 6713 DSK, which is based on floating point DSP and CC Studio either in hardware mode or in simulation mode.

***Keywords*—**DC motor, DSP, Dynamic simulation, Kalman Filter.

NOMENCLATURE

R_m	Armature resistance in Ohms;
L_m	Inductance in *mH,*
K_b	Back EMF constant (Volt-sec/Rad)
K_t	Torque constant (Nm/A)
J_m	Rotor inertia (Kg m^2)
B_m	Mechanical damping factor
K_k	Kalman gain
P_k	Posterior co-variance
$P_{(k)}^-$	Prior co-variance
Q	State noise
R	Measurement noise.

I. INTRODUCTION

THIS paper presents a tutorial on dynamic simulation of DC motor on a Digital Signal Processor (DSP). The 32 bit floating point DSP processors are readily available in the market and have high computing power and performance. This has helped in implementing some of the computationally complex observer algorithms such as Kalman filter which offers excellent estimates of states even in the presence of system and measurement noise. Sensor-less estimation of induction motor speed and GPS tracking are just a few examples, where these observer algorithms have been successfully used.

S. Padmakumar and Kallol Roy are with the BARC, Mumbai, India 400085 (e-mail:spadma@iitb.ac.in; kallol.roykallol@gmail.com).

Vivek Agarwal is with the Department of Electrical Engineering, IIT Bombay, Mumbai-400076, India (e-mail:agarwal@ee.iitb.ac.in).

The objective of this paper is to acquaint a beginner with what all is involved in the design and implementation of such systems.

As an example, a self excited fractional hp DC motor is modeled using Matlab® and the simulation results are presented for open loop time response of the state variables. The measurements are then corrupted by super imposing a noise, which follow a Gaussian distribution. A Kalman filter is implemented in Matlab® to observe the states from the corrupted measurement signal. A comparison of the noisy measurement signal and the estimated states through Kalman Filter is presented. The same model is implemented in TI® floating point DSP C6713 and the results including the Kalman Filter responses are also presented. The Matlab® code for all the simulations and the Code Composer Studio® project file including the required header files are available for download, from a link provided in Appendix-I.

II. DC MOTOR MODEL IN MATLAB®

A second order, linear state space model of DC motor is considered for this study. The two states considered are the armature current (I_a) and the speed (ω) of the DC motor. The input supply to the self excited DC motor is 12 V and the outputs are same as states.

A. Continuous Time State Variable Model of DC Motor

$$A = \begin{bmatrix} -R_m / L_m & -K_b / L_m \\ K_t / J_m & B_m / J_m \end{bmatrix} \quad (1)$$

$$B = \begin{bmatrix} 1 / L_m \\ 0 \end{bmatrix} \quad (2);\ C = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad (3) \text{ and } D = [0] \quad (4)$$

The continuous time state variable model, which is used for this work has the system matrices as given by (1-4).

$$A = \begin{bmatrix} -8655.462 & -98.825 \\ 21962.616 & -1.121 \end{bmatrix};\ B = \begin{bmatrix} 4201.680 \\ 0 \end{bmatrix} \quad (5)$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \text{ and } D = [0] \quad (6)$$

Numerically, the system matrices are as given by (5-6). Continuous time state (I_a and ω) response in Matlab for an input of 12 V DC supply is as shown in Fig. 1.

It can be observed that the starting current is of the order of 5 amps, which is taken as 1 p.u and no load current is 0.25

amps and the steady state speed is 500 rpm, which is again scaled to 1 p.u.

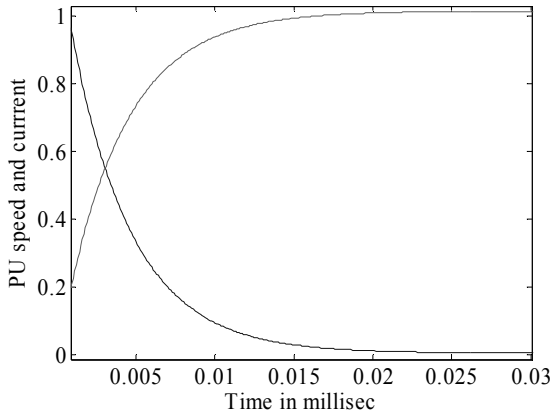

Fig. 1 Current and speed (scaled) response of DC motor in Matlab®

B. Discrete Time DC Motor Model

The continuous time state space model is discretised with a sampling time of 100 microsec, using the Matlab® function 'c2d'. Discrete time state variable DC motor model is as shown by equation (6) and is obtained by Matlab® function c2d.

$$A_d = \begin{bmatrix} 0.4146 & -0.0066 \\ 1.4643 & 0.9916 \end{bmatrix}; B_d = \begin{bmatrix} 0.2802 \\ 0.3521 \end{bmatrix}; C_d = [1\ 0] \qquad (9)$$

The discrete time simulation is as given in Fig. 2 which shows per unit combined current and speed response.

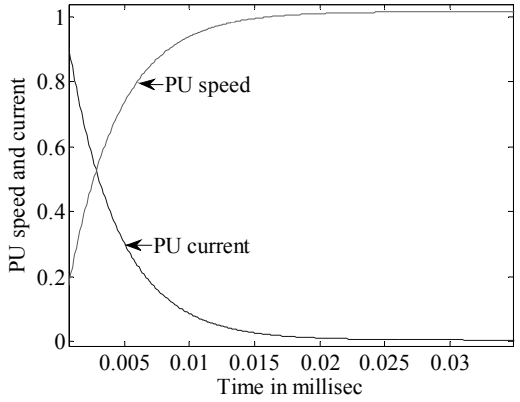

Fig. 2 Discrete time estimate of current and speed

If the sampling time is increased to 2 milliseconds, the system matrices in discrete time domain are as shown below:

$$A_d = \begin{bmatrix} 0.0189 & -0.0072 \\ 1.606 & 0.6139 \end{bmatrix}; B_d = \begin{bmatrix} 0.308 \\ 16.34 \end{bmatrix}; C_d = [1\ 0] \qquad (10)$$

By reducing the sampling time, the system may become non-observable, as can be seen in Fig. 2, where the current estimate is not accurate. The peak value attained is only 0.82 p.u. as against the actual 1 p.u. value.

III. Application of Kalman Filter for DC Motor

Kalman Filter is an optimal stochastic state observer for estimation of states which can take care of model uncertainties called system or process noise and measurement uncertainties or measurement noise [2].

$$\bar{X}_{(k+1)} = A_d X_{(k)} + B_d U_{(k)} \qquad (11)$$

$$\bar{P}_{(k+1)} = A_d P_{(k)} A_d^T + Q \qquad (12)$$

$$K_{(k)} = \bar{P}_{(k)} C_d^T \left[(C_d^T \bar{P}_{(k)} C_d + R) \right]^{-1} \qquad (13)$$

$$\hat{X}_{(k)} = \bar{X}_{(k)} + K_{(k)} [Z_{(k)} - (C_d X_{(k)})] \qquad (14)$$

$$P_{(k)} = (I - K_k C_d) \bar{P}_{(k)} \qquad (15)$$

It consists of a set of prediction equations (11-12) and a set of correction equations (13-15). The Kalman filter algorithm flowchart is as per Fig. 3. It starts with initialization of states and initialization of process and measurement co-variance matrices. The process co-variance matrix has the dimension of the '*A*' matrix of the system and the measurement co-variance matrix has a dimension of the measurement or observation matrix. In this particular case, '*Q*' is taken as 1e-13 and '*R*' is taken as 10. This is based on an intuitive approach, with apriory knowledge of the system [3].

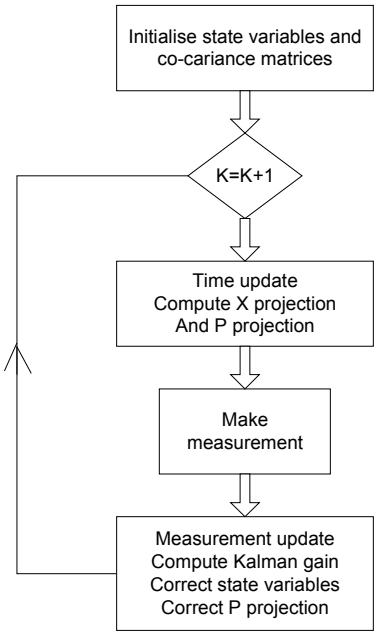

Fig. 3 Kalman filter flowchart

The DC motor is modeled in an open loop mode and the time response and output data set is generated. The measurement so obtained is corrupted by a white noise of mean zero, which can be generated using the Matlab® function

'randn'. With this corrupted measurement, the Kalman filter can estimate the correct amplitude of the state variable. Fig. 4 shows a corrupted measurement and Fig. 5 shows the estimated current and speed, in presence of noisy observations. The Matlab® code used in this tutorial is available for download through the link provided.

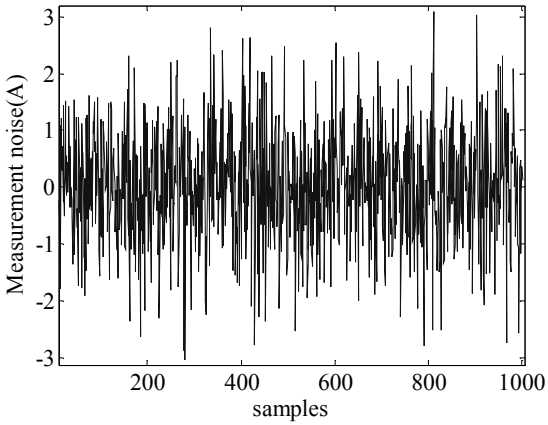

Fig. 4 Measured current (corrupted by noise)

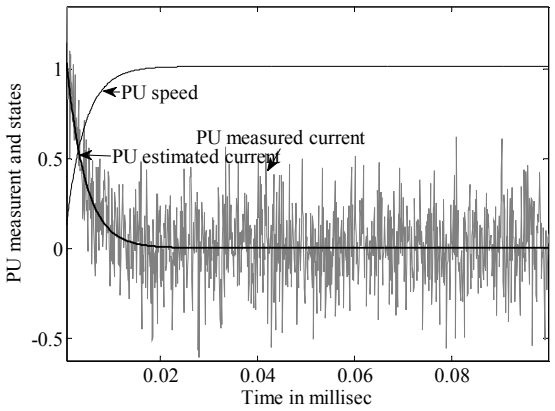

Fig. 5 Estimated current and speed from noisy measurements

IV. DC Motor Simulations on 6713 DSP

DC motor modeling and simulations on a floating point DSP is discussed in this section. Spectrum digital® DSK based on TI® TMS320C6713 processor is used for simulations. This DSK is based on 32 bit floating point processor, with both built in flash of 512 kB and 16 MB SDRAM. Besides, it has other additional features which include a codec and a built in one channel 8 bit ADC and DAC. This DSK works on a single 5 volts power supply, with 4 number of on board LEDs and a set of DIP switches. These DIP switches can be used for simulating limited external events, and the LEDs can be programmed to indicate any specific conditions based on the outcome of the computations. The programming in done through USB interface and the development platform for this product is code composer studio® from TI®. Fig. 6 gives the photo of DSK [4].

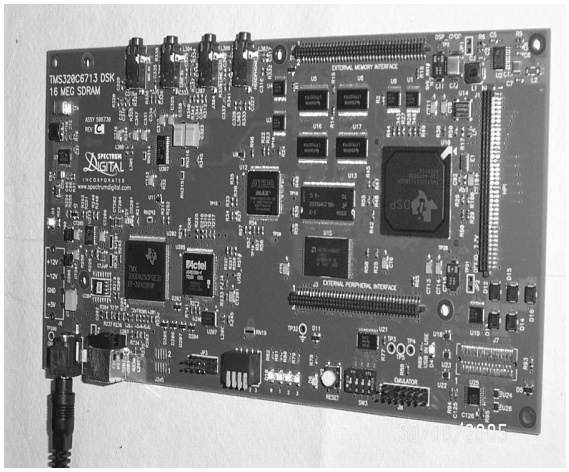

Fig. 6 DSK 6713 (courtesy: Spectrum Digital®)

The essential pre requisites are the code composer studio and the DSK. Initial training can be had on the CC studio itself, as there is a device simulator mode available, which can simulate the results exactly as the DSK would have given. During the setup, the specific DSK (6713) is to be selected for the simulator to function properly. The CC studio [5] provides an integrated development environment, primarily based on project concept. A project file will have many types of files including system, GEL, header, library and the user programs, which is written in either 'C' or assemble language. User has to correctly select a list of system files pertaining to the specific processor, its library files and the header files. Simple tasks are provided in CC studio to start with. After this, the project menu allows the user to compile individual files or build the while project with all the included files to generate a 'COFF' or an 'OUT' file, which can be loaded to the processor's flash and executed. The following are the steps to work on a DSK 6713 environment. The pre-requisites are also given (first two steps). The project file can also be downloaded from the link provided.

1. A DSK 6713 kit (Spectrum Digital).
2. Code composer studio (An evaluation version is available at TI® site for download) is to be installed in the PC, preferably in default 'C' drive.
3. The DSK can be power up and connected to a windows based PC through a USB port.
4. Driver files are installed properly for the DSK.
5. It is advisable to load both DSK 6713 and the device simulator at the setup stage, so that even if the DSK is not available readily, the simulator can be used.
6. On starting CC studio, the DSK will be detected initially and no error messages will be given if everything is properly installed. An interactive pop up menu will turn on in case of any issue, which can be ignored, so that the simulator can be started.
7. The project file, which is provided in the link can be downloaded and copied to my project folder within CC studio.

8. Check all the project dependencies and compile through 'Build all' option. Select 671X option in compiler if it doesn't work properly.
9. The project will be compiled and an 'out' file will be generated by the compiler, with no errors and warnings.
10. From the 'File' menu, load this program, which is usually residing in the default project folder, under the debug sub-folder, with an '.out' extension.
11. The assembly code will appear in the active right pane. From the debug menu, give a 'run' command to execute the assembly code of DSP. After successful execution, test passed message will appear and various parameters can be viewed through 'view' menu.
12. Check the main program to and familiarize with variables such as 'observ' for noisy observations, 'cur' and 'spd' for current and speed etc. From the view menu and the graph option (time/frequency) the trend of these variables can be seen by providing the variable name, the number of data (251) and the format of data (32 bit floating point format).
13. CC studio can handle both simulation of 6713 processor to a great extend and the DSK itself. The selection of the type of environment is to be given during starting of the program.

Fig. 7 gives the listing of the main 'C' file, which is separately given in Appendix-II also. Typical project environment for the dc motor simulation is shown in Fig. 8, which gives a list of files required to be included in the project environment. All the associated files are provided in the project file. The native compiler of CC studio environment support 'C' and 'C++'.

All the functions and main programs are to be written in 'C++' or assembly language for it to compile properly. Matlab® can handle the matrix operations easily, but for a DSP system development, all the functions and programs are to be developed in 'C' or 'C++' only.

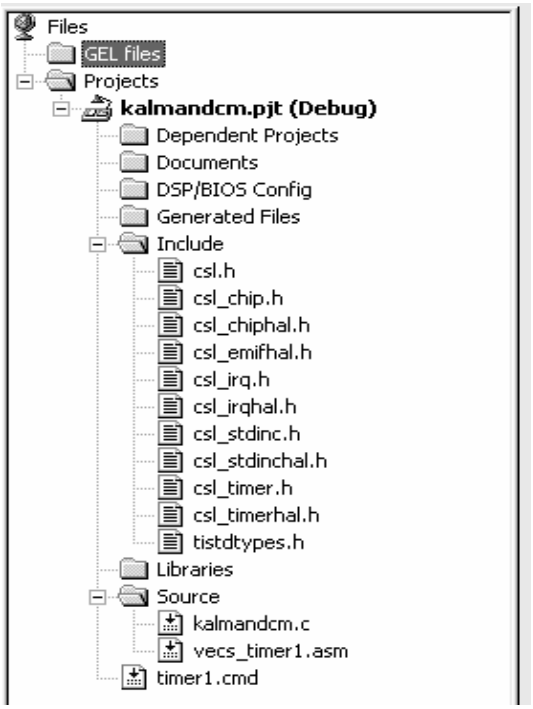

Fig. 7 List of associated files required for compilation

The code for dc motor simulations and the Kalman filter code are provided in the annexure-1. Fig. 8 shows the corrupted measurements provided to Kalman filter through a data set and Fig. 9 shows the current estimation by Kalman filter in DSP and Fig. 10 shows both the estimates as computed by the recursive Kalman filter iterations.

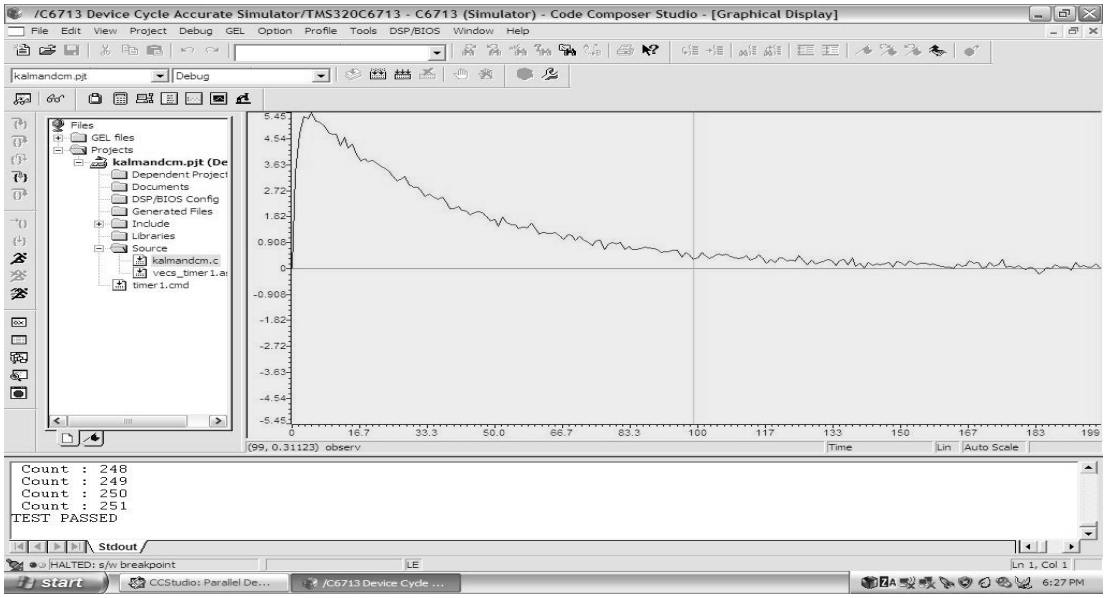

Fig. 8 Measured current, which is corrupted by noise

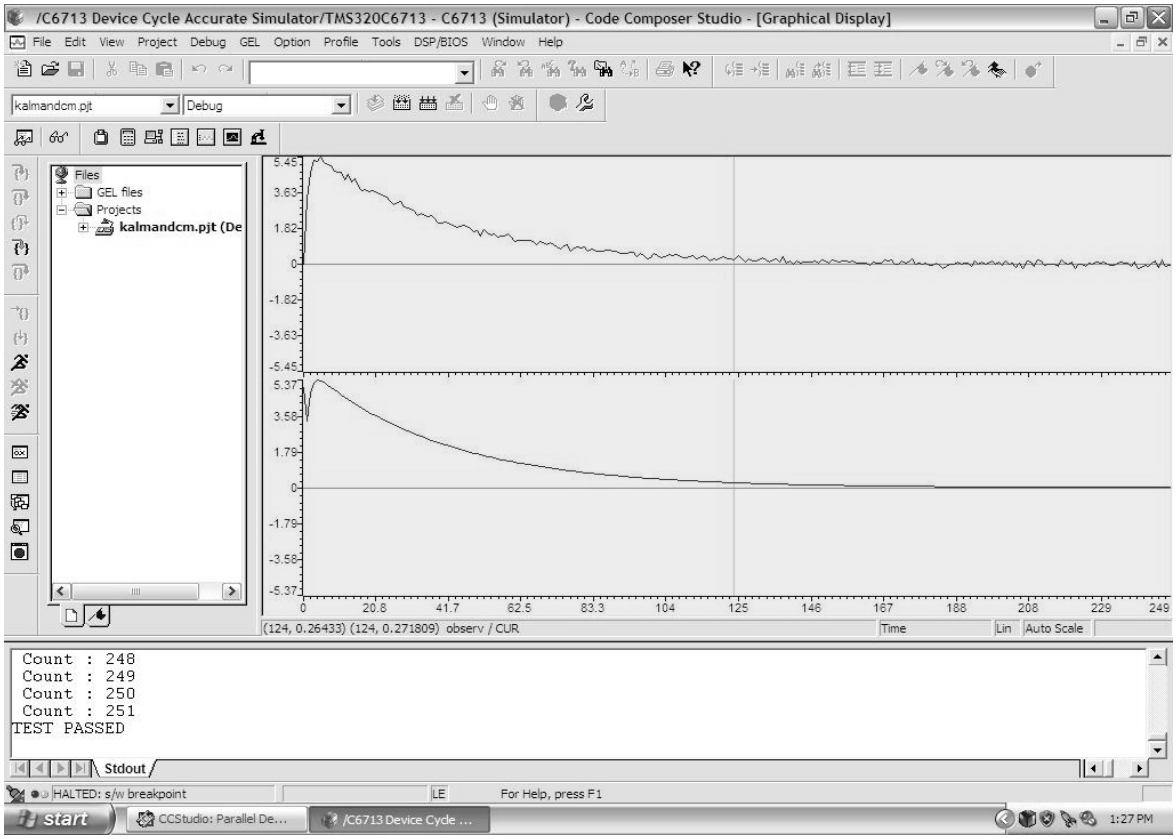

Fig. 9 Estimated and actual current of dc motor in DSP

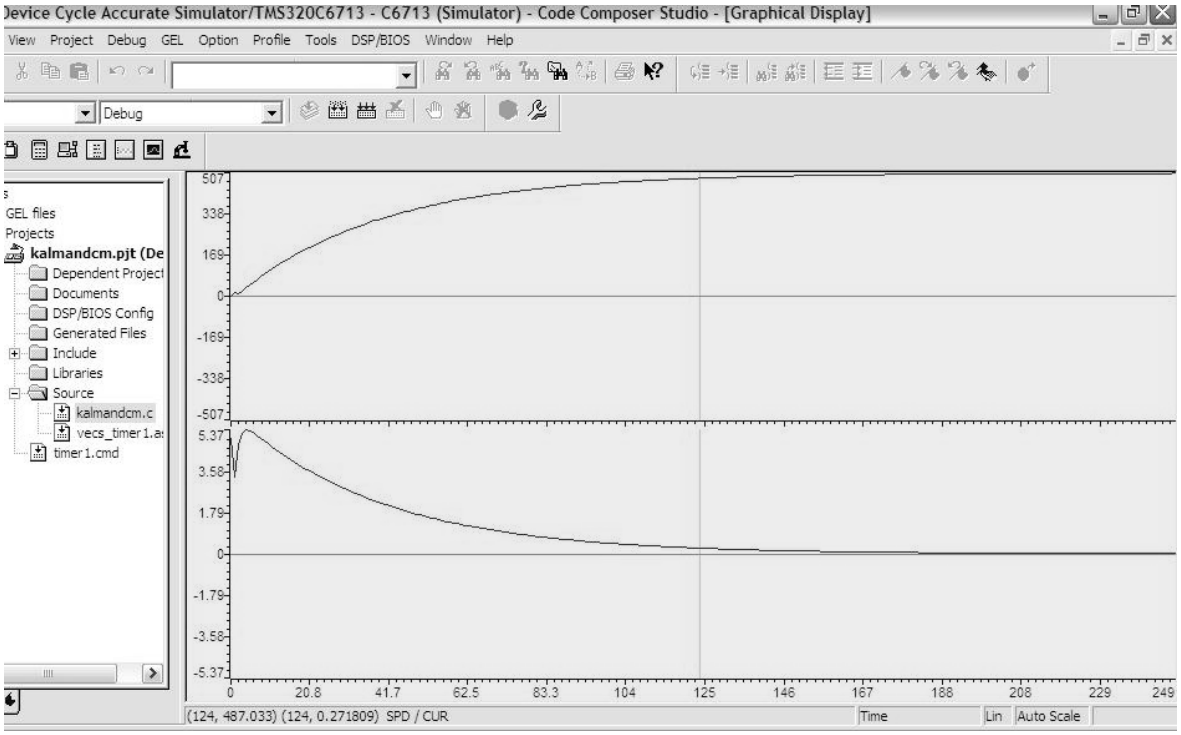

Fig. 10 Estimated speed and current by Kalman filter (DSP)

V. Concluding Remarks

A tutorial on DC motor modeling and simulations in Matlab® and implementation of Kalman filter for estimation of states from corrupted or noisy measurement is presented. A step by step approach for the implementation of the same in a 32 bit floating point TI® TMS320C6713 based DSP and the results are also provided. This tutorial gives details of how to implement dynamic model simulations in discrete domain and a stochastic filter implementation both in hardware and simulation mode for a floating point DSP. The associated codes are available for download, which are expected to be helpful for the beginners, to get an experience with a floating point DSP.

Appendix-I
Web links for **code** download

1. www.iitb.ac.in/student/~padmakumar/dcm_matlab.zip
2. www.iitb.ac.in/student/~padmakumar/dcm_ccsproj.zip

Appendix-II
'C' routine for dc motor simulation and Kalman filter in Code Composer studio

```
void TimerEventHandler(void) {

 /* process timer event here */

 cnt++;

AT[0][0]=A[0][0];AT[0][1]=A[1][0];AT[1][0]=A[0][1];AT[1][1]=A[1][1]
;
Xhatproj[0][0] = A[0][0] * X[0][0] + A[1][0] * X[0][1] + B[0][0] * 12.0;
Xhatproj[0][1] = A[0][1] * X[0][0] + A[1][1] * X[0][1] + B[0][1] * 12.0;
AP[0][0]=A[0][0] * P[0][0] + A[1][0] * P[0][1];
AP[1][0]=A[0][0] * P[1][0] + A[1][0] * P[1][1];
AP[0][1]=A[0][1] * P[0][0] + A[1][1] * P[0][1];
AP[1][1]=A[0][1] * P[1][0] + A[1][1] * P[1][1];
APAT[0][0]=AP[0][0] * AT[0][0] + AP[1][0] * AT[0][1];
APAT[1][0]=AP[0][0] * AT[1][0] + AP[1][0] * AT[1][1];
APAT[0][1]=AP[0][1] * AT[0][0] + AP[1][1] * AT[0][1];
APAT[1][1]=AP[0][1] * AT[1][0] + AP[1][1] * AT[1][1];

Pproj[0][0]= APAT[0][0] + Q[0][0];
Pproj[1][0]= APAT[1][0] + Q[1][0];
Pproj[0][1]= APAT[0][1] + Q[0][1];
Pproj[1][1]= APAT[1][1] + Q[1][1];

INVCPCT = 1.0/(Pproj[0][0] + R);
PprojCT[0][0]= Pproj[0][0];
PprojCT[0][1]= Pproj[0][1];
K[0][0]= PprojCT[0][0] * INVCPCT;
K[0][1]= PprojCT[0][1] * INVCPCT;

innov =(observ[cnt]- C[0][0] * Xhatproj[0][0]);

xhat[0][0]= Xhatproj[0][0] + K[0][0] * innov;
xhat[0][1]= Xhatproj[0][1] + K[0][1] * innov;

KC[0][0]=1.0-K[0][0]; KC[1][0]=0.0;
KC[0][1]= -K[0][1]; KC[1][1]=1.0;

P[0][0]= Pproj[0][0] * KC[0][0] + Pproj[1][0] * KC[0][1];
P[1][0]= Pproj[0][0] * KC[1][0] + Pproj[1][0] * KC[1][1];
P[0][1]= Pproj[0][1] * KC[0][0] + Pproj[1][1] * KC[0][1];
P[1][1]= Pproj[0][1] * KC[1][0] + Pproj[1][1] * KC[1][1];

CUR[cnt]= xhat[0][0];
SPD[cnt]= xhat[0][1];
INNV[cnt]=innov;
X[0][0]=xhat[0][0];
X[0][1]=xhat[0][1];

 /* Exit from the program when certain count is reached */
 if (cnt > TIMER_CNT)
 {
    TIMER_pause(hTimer1);
    TIMER_close(hTimer1);
    printf("\nTEST PASSED\n");
    exit(0);
 }
 printf("\n Count : %3d ",cnt);
```

Appendix-III
DC motor data used for the code

R_m	2.06 Ω, Armature resistance;
L_m	0.238 *mH,* Inductance
K_b	0.02352, Back EMF constant (Volt-sec/rad)
K_t	0.0235, Torque constant (Nm/A)
J_m	1.07e-6, Rotor inertia (Kg m^2)
B_m	12e-7, Mechanical damping factor

References

[1] Simulink® 'Simpowersystem' documentation, pp 2.87-2.97
[2] Grewal M. S and Andrews A. P, 'Kalman Filtering theory and practice', book by Englewood Cliffs, Prentice Hall, 1993.
[3] Peter. S Maybeck,' Stochastic models-estimation and control' Vol. 1, Academy press, New York, 1982.
[4] Spectrum Digital product catalog for DSK 6713:
[5] CC studio programmer's manual by TI®

Padmakumar.S has graduated from Kerala University in 1985, completed his post graduation from IIT Bombay in 1987 and is undergoing Ph.D program at IIT Bombay. He is presently working as Engineer in charge of electrical systems refurbishment cell, research reactor maintenance division, BARC. He is associated with electrical systems maintenance of nuclear research reactors.

Vivek Agarwal received the bachelor's degree in Physics from St. Stephen's college, Delhi University, India. He then obtained an integrated master's degree in Electrical Engineering from Indian Institute of Science, Bangalore. Subsequently he pursued a Ph.D. degree in the dept. of Electrical and Computer Engineering, University of Victoria, Canada. After obtaining the Ph.D. degree, he briefly worked for Statpower Technologies, Burnaby, Canada as a research engineer. In 1995 he joined the department of Electrical Engineering, Indian Institute of Technology-Bombay, where he is currently a Professor.

Kallol Roy completed his B.Tech (Electrical) from NIT, Calicut in 1984. He obtained his M.Tech. from IISc., Bangalore in 1990 and PhD (Systems and Control) from IIT, Powai, Mumbai in 2000. He has been Post-Doctoral Fellow at University of Alberta, Canada, from April 2005 to March 2006. His expertise is in the Maintenance of Research Reactors, especially Planning, Retrofitting and Development of Predictive Maintenance tasks.