High-gain observers and Kalman filtering in hard real-time

Nicolas Boizot
SCE, LASSY, CSC, Université du Luxembourg
6 rue coudenhove-Kalergi, L-1953 Luxembourg
nicolas.boizot@uni.lu

Eric Busvelle
Le2i, Université de Bourgogne
Le2I, IUT, route des plaines de l'Yonne, F-89000 Auxerre
busvelle@u_bourgogne.fr

Juergen Sachau
SCE, LASSY, CSC, Université du Luxembourg
juergen.sachau@uni.lu

Abstract

In the framework of our research on high-gain observers for nonlinear systems we have reached the final development step consisting of a real-time implementation in Linux. The results presented in this paper were obtained using RTAI-lab and the Scilab/Scicos CACSD environment on a series-connected DC motor. We introduce the three algorithms we implemented as 1) the classic high-gain Luenberger, 2) the high-gain extended Kalman filter, and 3) our newly proposed adaptive-gain extended Kalman filter (AEKF). This article is a comprehensive description of our real-time applied mathematics experiment.

1 Introduction

Observer theory is a subset of control theory. In control theory, one wants to control system outputs using its actual state in order to reach a target or to optimize some cost function. In order to achieve this task, it is necessary to estimate the state of the process using measurements. The main goal of the observer is to provide an estimation of internal variables of the process using measurements and a dynamic model. If one consider linear systems the Luenberger observer and the Kalman filter naturally come to mind. Those two solutions and particularly the Kalman-Bucy approach have been adapted to the nonlinear case for which linearised equations of the dynamic model are used to obtain the extended Kalman filter or some sort of extended Luenberger observer. However the fact that this linearisation is made in a neighborhood of the estimated trajectory prevent us from producing general convergence results, in particular when the initial error between the actual state and its estimation is high. In the case of a small initial error the extended Kalman filter has been proven (in the deterministic setting) to be an exponentially converging observer (see[1] Thm.8 and [2] part 3.2.3). This result together with its good filtering properties in the presence of small noise (see Picard, e.g.[3]) indicates that this solution is a good local observer.

High-gain observers where introduced and their nice theoretical properties emphasised a long time ago, first in the form of observers with non-varying correction gain [4] and then as an adaption of the extended Kalman filter [5]. With time, a whole theoretical framework was established around the notions of uniform observability and high-gain observers showing how, starting from a given process, globally exponentially converging observers can be designed and implemented [6]. The structure or such observers relies on special ways of writing the system, the observability forms, and a modification of the tuning matrices used to design the observer by mean of a single parameter, θ, called the high-gain parameter. By choosing an appropriate value for this parameter, the user may obtain an observer whose estimation of the state converges (theoretically) to the real state of the system as quickly as desired. In practice, the situation in not as ideal since the price paid for a faster convergence is a higher sensibility to noise, which may render the estimation useless.

We therefore focused our work on the design of a globally exponentially converging observer having the following general behavior:

- when the estimation is close to the true state we want the observer to act as an extended Kalman filter (local convergence and noise rejection)
- when the estimation is far from the true state or is facing non-measured perturbations, the observer should evolve so as to reach the behavior of a high-gain observer (ensuring global convergence).

Since the extended Kalman filter corresponds to a high-gain Kalman filter for which $\theta = 1$, the precited behaviour will be reached when a proper way to adapt this parameter is defined. A first result in that direction is the *high-gain and non high-gain extended Kalman filter* proposed by Busvelle and Gauthier [2] where the parameter decays exponentially to one. Global convergence is reached when the initial value of θ is taken high enough and is not decreasing too fast. This system is made persistent by running multiples observers with their parameter θ re-initialised alternatively which ensures that at each time step one of the observer will have a high-gain behavior and another one will be close to an extended Kalman filter. The authors recently proposed another solution in which the high-gain parameter may increase or decrease depending on the estimation error ([7], [8]). The adaption is driven by a quantity denoted innovation whose computation implies the simulation of the model over a (small) time window (def.3). The computation of this quantity is time consuming compared to other calculations in the same task and a major problem when actually implementing it in a real-time setting. We therefore studied the possibility of a hard realtime implementation on a simple system, namely a series connected DC machine, having a representative time constant. This was done using the RTAI-lab computer aided control system design environment. In order to test one step further the accessibility of this tool we used the RTAI-Knoppix distribution proposed by Gianluca Palli [9].

2 Three observers

The three nonlinear observers we implemented are: the high-gain Luenberger observer (sometimes denoted as the classic high-gain observer), the high-gain extended Kalman filter, and the adaptive-gain extended Kalman filter. This choice was motivated by the increase in the computational effort that is needed to perform an update of the state estimates, with the computational time of the first observer being used as a reference for the two others.

When designing a high-gain observer, the model is assumed to be in an observability form. In the case of SISO systems, the observability canonical form is given by

$$\left\{ \begin{array}{lcc} \dot{x} & = & A(u).x + b(x,u) \\ y & = & C.x \end{array} \right. \tag{1}$$

where

$$A(u) = \begin{pmatrix} 0 & a_2(u) & 0 & ... & 0 \\ 0 & 0 & a_3(u) & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & a_n(u) & 0 \\ 0 & 0 & 0 & ... & 0 \end{pmatrix}$$

$$b(x,u) = \begin{pmatrix} b_1(x_1,u) \\ b_2(x_1,x_2,u) \\ b_3(x_1,x_2,x_3,u) \\ ... \\ b(x,u) \end{pmatrix}$$

$$C = \begin{pmatrix} a_1(u) & 0 & ... & 0 \end{pmatrix}$$

together with the following hypotheses:

- $a_i(.), i = 1, ..., n$ are positive smooth functions, bounded from above and from below as follows $0 < a_{min} < a_i(u) < a_{max}$
- b is a smooth, u-dependent vector field with components $b_i(.)$ that depend triangularly on x and are compactly supported.

Of course not all models will naturally appear under this normal form and some work will be needed to find an appropriate change of coordinates that will put it under the normal form. Readers interested in the topic of observability and normal forms may refer to [6], [8]. Some practical remarks concerning the assumptions made on the SISO observability form may be found in [2, section 2]. Keeping those notations and denoting the estimation of state by z we now give the following definitions.

Definition 1 *Given system 1, we define the high-gain Luenberger observer as the dynamical system*

$$\dot{z} \;=\; A(u).z + b(z,u) - K_\theta(C.z - y)$$

where $K_\theta = \theta\Delta_\theta K$, $\Delta_\theta = diag\{1, \theta, ..., \theta^{n-1}\}$ and K is such that $(A - KC) < 0$ in the sense of symmetric matrices.

Definition 2 *The extended Kalman filter is defined as a dynamical system composed of a set of two equations that are*

$$\left\{\begin{array}{rcl} \dot{z} & = & A(u).z + b(z,u) - PC^{'}R^{-1}(C.z - y) \\ \dot{P} & = & P(A(u) + b^*(z,u))^{'} + (A(u) + b^*(z,u))P... \\ & & -PC^{'}R^{-1}CP + Q_\theta \end{array}\right.$$

where $b^*(z,u) = \frac{\partial b}{\partial z}(z,u)$, $Q_\theta = \theta^2\Delta_\theta Q\Delta_\theta$. Q and R are symmetric definite positive matrices. In the stochastic setting, they respectively correpond to the covariance matrices of the state and measurement noise. The second equation of this observer is a dynamical version of the Riccati equation.

This next observer self-adapts to the perturbations it detects. This adaption should be driven by a quantity which measures such perturbations. Instead of using the estimation error on the measured state variables, we use innovation as it is defined below.

Definition 3 (Innovation) *Innovation is the quantity*

$$I(t) = \int_{t-d}^{t} \|y(s) - \hat{y}(s)\|^2 \, ds$$

where $y(s)$ corresponds to the measurements made over the time interval $[t-d; t]$ and $\hat{y}(s)$ is the solution over the time interval $[t-d; t]$ of the system

$$\begin{array}{rcl} \dot{\hat{x}} & = & A(u).\hat{x} + b(\hat{x}, u) \\ y & = & C\hat{x} \\ \hat{x}(t-d) & = & z(t-d) \end{array}$$

that is the simulation over a time window of length d of the model of the system with the estimation given by the observer at time t-d as the initial state.

The reason why we chose this quantity comes from a lemma [7] which states that, up to a multiplication by a constant, the innovation upper bounds the state error.

Definition 4 *We finally define the adaptive-gain extended Kalman filter as the three equations*

$$\left\{\begin{array}{rcl} \dot{z} & = & A(u).z + b(z,u) - PC^{'}R_\theta^{-1}(C.z - y) \\ \dot{P} & = & P(A(u) + b^*(z,u))^{'} + (A(u) + b^*(z,u))P... \\ & & -PC^{'}R_\theta^{-1}CP + Q_\theta \\ \dot{\theta} & = & \lambda\left[(1 - s(I)) + s(I)\theta_{max} - \theta\right] \end{array}\right.$$

where $Q_\theta = \theta\Delta_\theta Q\Delta_\theta$ and $R_\theta = \frac{1}{\theta}R$.

The meaning of this third equation is that the parameter θ will converge exponentially to a value that lies in the interval $[1; \theta_{max}]$ depending on the value $s(I)$ which is a function of the innovation. The function we use is the sigmoid:

$$s(I) = \frac{1}{(1 + e^{-\beta(I-m)})}, s(I) :]-\infty; +\infty[\to]0; 1[$$

where β is chosen to determine how fast the targeted parameters will change (i.e. as fast as possible or gradually) and m is used to discrime the values taken by the innovation only due to measurement noise from the influence of real pertubations, the only case against which we want the parameter to change.

The first observer is easy to implement as the task demands only the resolution of n differential equations corresponding to the number of variables that are to be reconstructed. The two other algorithms are more demanding in computational terms because in addition to the n differential equations, a Riccati equation has to be dynamically solved, which represents $n(n+1)/2$ more equations due to the symmetry of the matrix P. Finally the part we expect to be much more time consuming is the computation of the innovation because of the simulation (i.e. at each time step n differential equations are to be solved on a time interval of length d).

3 Apparatus

As stated in the introduction our goal is to use a high-gain observer to reconstruct the speed of a series-connected DC machine from the measures of the current (Fig.1).

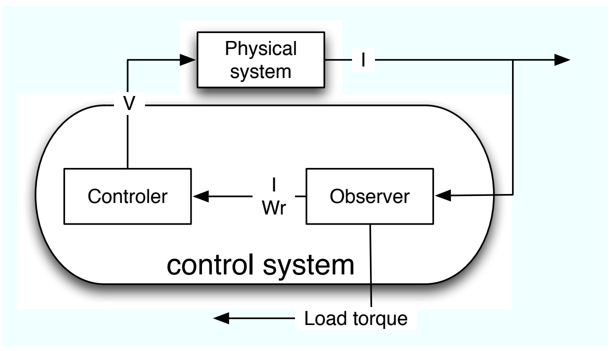

FIGURE 1: *General diagram of the testbed*

3.1 Testbed

Our testbed is composed of a DC motor from Lucas Nuelle (ref. SE2665-5C) coupled on one end with a tachometer and on the other end with a propeler. The tachometer (ref.2662-5U) is used only for a comparison of the estimated speed to the real one and to calibrate the mathematical model. The propeler (47.0 x 30.5 cm, together with a 5 ° pitch) attached to a 52mm center hub is from Aero-naut (ref.7234/97) and is expected to generate a significant resistive load torque [14]. Power is supplied by a DC source from Delta Electronika (SM-300-10D). Finally the physical system communicates with the control system by means of an I/O card from National Instruments (6024E-DACA): measurements of the current, voltages and speed are fed to the control system while set values for the current are delivered to the power supply (3 inputs/1 output).

The control system consist of a Dell PC (P.IV 3 GHz, 512 Mb DDR2 - SDRAM) running under a RTAI-knoppix distribution provided [9] with the following features:

- Linux kernel version 2.6.17 (SMP enabled kernel is available)
- RTAI version 3.4[10]
- Scilab-4.0/Scicos CACSD platform[13]
- comedi support[11]
- xrtailab graphic interface[12]

3.2 Modeling

The mathematical model of a series-connected DC machine presented in the litterature is the following one ([18]):

$$\begin{cases} \dot{I} &= \frac{1}{L}(V - R.I - Laf_1 I\omega_r) \\ \dot{\omega_r} &= \frac{1}{J}(Laf_2 I - T_{res}) \end{cases}$$

It is based on several assumptions although we did not keep the one of *ideal efficiency of the machine*. This is expressed by the use of two different mutal inductances (i.e. $Laf_1$and Laf_2). The expression T_{res} of the second equation is a resistive torque modeled as a viscous friction torque ($B\omega_r$) generated by the contacts inside the motor, a torque due to the propeler ($p\omega_r^{2.08}$ [14]), and an unknown perturbation load torque denoted T_l. This unknown variable is made identifiable by the addition of a third equation ($\dot{T_l}$=0) and only I is considered to be measured. This system is uniformly observable and therefore there is a change of coordinates that puts it under the SISO observability canonical form (see [8] for details). Calibration of the model is made once the parameters $R, L, B, J, L_{af1}, L_{af2}, p$ have been set to proper values. We followed the procedure:

- R represents the resistance of the motor we physically measured
- from data samples (voltage, current and speed) a first estimation is obtained with the least mean squares
- a nonlinear optimisation routine (simplex method) is used with the previuously found parameters estimation as starting point (the function to minimize associates the set of parameters to the distance between the data sample and a simulation of the model)
- remaining modeling errors are lessened by tuning the parameters around the peviously found values (those values probably correspond to a local minimum).

3.3 IMPLEMENTATION

In order to fully activate the real-time environment, we performed three steps:

1. the loading of the different RTAI and COMEDI modules is done following the commands indicated in[12]
2. the input/output signals are calibrated with a routine from Comedi. Since during this calibration big step impulses are sent to the motor, it is essential to safety that the power supply BE OFF. In addition to this a step is still active when the routine ends, therefore a (dummy) program should have been prepared in advance so that when run for a few seconds the signal will be set back to zero. Calling this program *end_of_load*, gives the following commands
   comedi_config /dev/comedi0 ni_pcimio
   comedi_calibrate –no-calibrate -S ni6024e.calibration
   *chmod 666 /dev/comedi**
   ./end_of_load -f 10
3. The first time we tried to get measurements using the A/D, D/A and FIFO block of the RTAI-lib palette in Scilab [12] we encountered a problem when tring to use more than one A/D channel (input of the control system). After some discussion with the RTAI community, it appeared that there was a small error in the file *rtai4_comedi_datain.sci* which has to be replaced with the code given in the annex [15]. The files to be replaced are located in the

virtual file system UNIONFS in the following repositories:
/UNIONFS/usr/src/rtai-3.4/rtai-lab/...
...scilab/macros/RTAI/
/UNIONFS/usr/local/scilab-4.0/macros/RTAI/
It is also necessary to recompile the file in the second repository with
scilab -comp rtai4_comedi_datain.sci

Since we can sample the I/O signal quickly enough compared to the physical system time constant we choose to implement the whole control strategy in the quasi-continuous setting which was straightforward following Fig.1 and [12], withstanding the observers. Their implementation was done using the *RTAICblock* bkock which appears in the RTAI-lib palette and is an adaption of the *Cblock2* block of the palette Others. The code developped for the Scicos simulation can then be reused for the realtime program up to a multiplication of the signals exchanged with the I/O card by a scaling factor. As its name implies the computational function of this block has to be written in C with a structure similar to Matlab S-function [19]. It appears from the definitions of the three observers that matrix mutliplications will be needed. We actually tried two solutions to solve this:

1. we used the routines already present in the Scilab code and may be called with respectively the header and the two functions
   #include <routines/machine.h>
   extern int C2F(dmmul)();
   extern int C2F(dmmul1)();
   Those two functions take the matrices A, B, C as inputs and output respectively $C = A * B$ and $C = C + A * B$. Computations are done with combinations of those two functions. In addition to this one has to consider that the Ricatti matrix appearing in the Kalman-like filters is square symmetric. Then for a $dim(n \times n)$ matrix, only $n(n+1)/2$ computations have to be done. A small program that trasforms the square matrix into a corresponding column vector and vice versa, has to be written.

2. it appears that the matrices used in the computations above have some particular structure (e.g. $A(u)$ is an upper diagonal matrix, Q, R are often choosen diagonal, $b^*(z, u)$ is lower triangular...). Consequently developping the equations on paper gives us simplified equations which will prevent the system from doing useless computations. The Ricatti equation will demand some effort but the other ones are straightforward.

In the case of the DC machine where the number of state equations is equal to 3, the second solution is definitely the one to use.

As shown in Fig.2, the implementation of the adaptive-gain extended Kalman filter has been devided into three parts: computation of the observer equations as for the high-gain Kalman filter, a time delay block, and computation of the innovation. This latter operation is done in discrete time mainly because

- the computation of $y(s)$ for $s \in [t-d; t]$ requires a simulation of the model which will implies a non-negligible amount of computational time
- computation of the integral (by a trapezoidal method) is easier to handle when data ($y(t)$) are updated at given and *apriori*-known sample times.

As in the case of matrix multiplication, the simulation of the model over a time window of length d may be done using an external routine. As a first atempt we used *lsode* which already exists in Scilab source code. Here again the *machine.h* header and a function call identical to the ones above are to be included (for more information on how to use this routine [16]). Unfortunately it appeared that unlike *dmmul* and *dmmul1*, the use of *lsode* was not supported by the realtime compiler. We then replaced it by a fourth order Runge-Kutta algorithm (e.g.[17]).

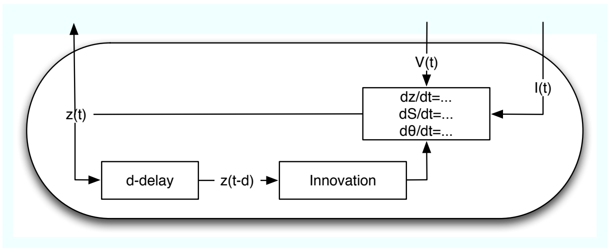

FIGURE 2: *Decomposition of the adaptive-gain extended Kalman filter*

4 Results

We implemented the three observers both in simulation and realtime compilation. Several simulations were done in order to find correct values for the high-gain parameters for the Luenberger observer and the Kalman filter. The parameters were estimated at a low speed (ω_r=180$rad.s^{-1}$) and the three observers tested according to the following scenario:

- the machine is fed 54 V for 30 seconds
- at t=30 the input voltage is switched to 42 V for another 30 seconds

- the voltage is then raised to 66 V and shut down after 30 seconds.

During each step, for a period of about 10 seconds a a non measured friction force perturbation is applied to the shaft of the motor: here appears the importance of low speed.

Fig.3 shows the estimations given by the Luenberger observer in open loop. The high-gain parameter was set to θ=2.5 and the application was ran with a 0.001 seconds sampling time.

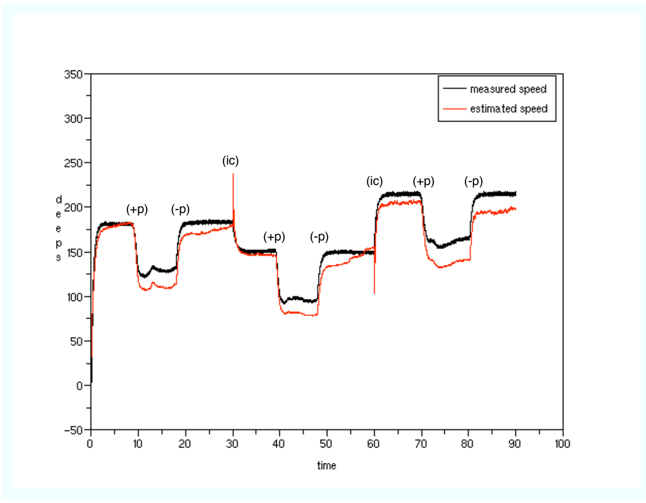

FIGURE 3: *Speed estimation using a high-gain extended Luenberger observer.*
(+p): begining of perturbation, (-p): end of perturabtion, (Ic): change of the input

The high-gain Kalman filter because of the additional computations needed to dynamically solve the Ricatti equation (representing $n(n=1)/2$ more equations when n is the dimension of the state vector) is often seen as very slow observer. Compared to a Luenberger filter this will actually be the case as ran it at a sample time of 0.01 seconds. However this observer is more efficient when dealing with systems for which the A matrix of the mathematical model (see 1) depends on the input. The results presented in Fig.4 shows both the results of the estimation of the speed delivered by an extended Kalman filter (θ=1) and its high-gain counterpart (θ=2.5). Since our perturbations are done by hand the only way to present such kind of graph is to run both those obsevers in parallel. We think that with some efforts to efficiently tune the code of the Kalman filter it could be possible to run it at 0.001 seconds. As expected the non high-gain filter is the slowest one. As the measurement noise was not that big it is difficult to distinguish the (bad) influence of a high-value of θ on the estimation but still it can be noticed in the time windows [5;10] and [25;30].

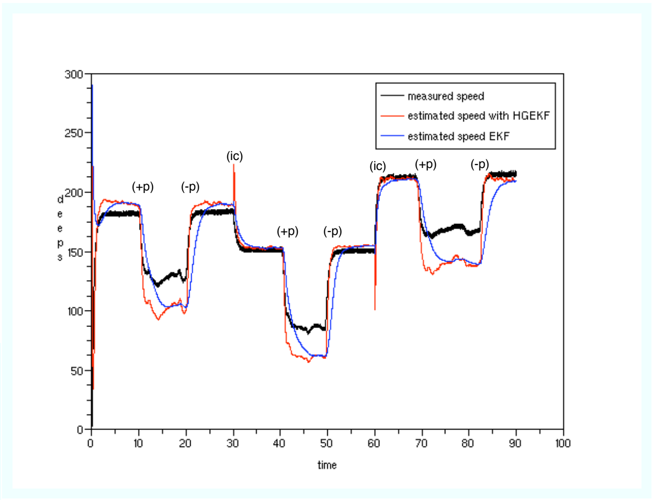

FIGURE 4: *Speed estimation using a high-gain extended Kalman filter*
(+p): begining of perturbation, (-p): end of perturabtion, (Ic): change of the input

The implementation of the adaptive-gain extended Kalman filter was a little bit more demanding than the previous ones, but in the end we managed to have it running in real-time with a 0.01 seconds sample time. The parameters specific to this observer were set to (see [7],[8] for details)

- θ_{max}=2.5, the value previously used for the fixed high-gain
- λ=500, θ will quickly reach its target value
- β=2000, θ will switch from 1 to θ_{max} almost immediately
- m_1=0.05, it determined by the choice we made for β
- m_2=$\sigma^2 * d$=0.004 where σ is the standard deviation of the measurment noise (0.2 is small)
- d=0.1 is the delay in the computation of the innovation
- Dt=0.01 is the sampling time of the innovation discrete block which will be updated at each time step of the process. It also means that the simulation needed to compute the innovation is made in d/DT=10 steps

Results of the estimation appear in Fig.5. Because this run was done with no other observer it is not so easy to compare it to others but still we see that when facing perturbations the observer has a speed of convergence comparable to the high-gain extended Kalman filter and presents a curve that is as smooth as the one of the extended Kalman filter.

If we take a look at Fig.6 we see that the parameter θ reacts 9 times during the experiment, i.e. once for every modification plus one extra time corresponding to a bad initialisation. Those reactions correspond to (measured) changes of the input voltage or non-measured changes in the load torque (perturbations). It seems regular that changes occur in the second situation but not in first one. In fact this is due to modeling errors because of which the innnovation may not vanish. This problem is solved by filtering the innovaton the following way: $\begin{cases} \dot{I}_f &= \alpha(I - I_f) \\ I &= I - I_f \end{cases}$ where α fixes the maximum time θ will stay at its maximum value.

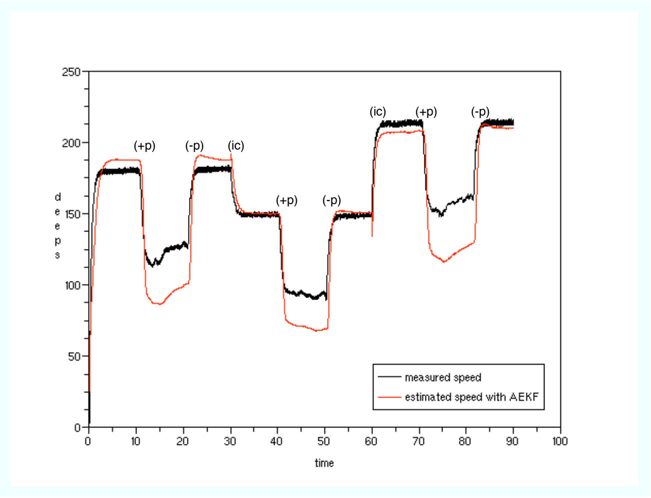

FIGURE 5: *Speed estimation using the adaptive-gain extended Kalman filter (+p): begining of perturbation, (-p): end of*

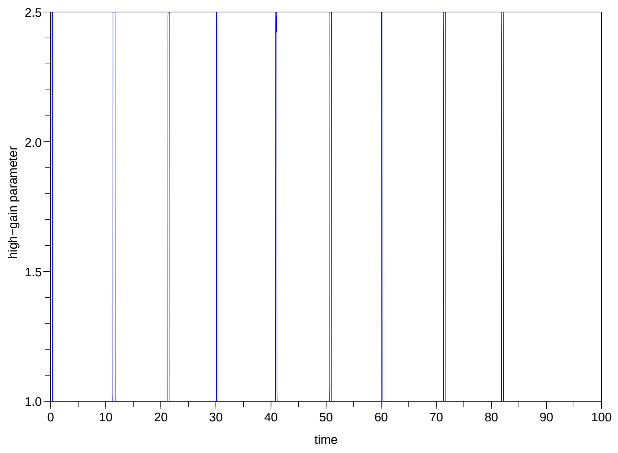

FIGURE 6: *Evolution of theta*

5 Conclusion

The main objective of this work was to show that the adaptive-gain extended Kalman filter can be implemented in a hard real-time setting. Keeping in mind that the system we used was small (three equations) we proved that this implementation was feasible. Some interesting points came out concerning modeling errors in particular the fact modeling errors may trigger the high-gain parameter of the adaptive-gain observer when no pertubation is occuring. In the future we foresee the use of the observers at a higher motor speed, to properly close the control loop as we planned in Fig.1, the usage of RTAI-lab to test the MIMO version of the adaptive-gain observer, and also the implementation of a small program that would automatically generate a C code to be copied in the *CRTAIblock* so as to ease future utilisation.

Acknowledgements

The first author really wants to thanks his colleague Kenneth D. Sebesta (at least i wrote your name properly) for his help during this long journey toward ”real life” application.

Annex [15]

```
function [x,y,typ] = rtai4_comedi_datain(job,arg1,arg2)
    x=[];y=[];typ=[];
    select job
    case 'plot' then
    exprs=arg1.graphics.exprs;
    ch=exprs(1)
    name=exprs(2)
    standard_draw(arg1)
    case 'getinputs' then
    [x,y,typ]=standard_inputs(arg1)
    case 'getoutputs' then
    [x,y,typ]=standard_outputs(arg1)
    case 'getorigin' then
    [x,y]=standard_origin(arg1)
    case 'set' then
    x=arg1
    model=arg1.model;graphics=arg1.graphics;
    exprs=graphics.exprs;
    while %t do
    [ok,ch,name,range,aref,exprs]=..
    getvalue('Set RTAI-COMEDI DATA block parameters',..
    ['Channel:';
    'Device:';
    'Range:';
    'Aref:'],..
    list('vec',-1,'str',1,'vec',-1,'vec',-1),exprs)
    if ~ok then break,end
    if exists('outport') then out=ones(outport,1), in=[], else out=1, in=[], end
    [model,graphics,ok]=check_io(model,graphics,in,out,1,[])
    if ok then
    graphics.exprs=exprs;
```

```
model.ipar=[ch;
range;
aref;
length(name);
ascii(name)'];
model.rpar=[];
model.dstate=[1];
x.graphics=graphics;x.model=model
break
end
end
case 'define' then
ch=0
name='comedi0'
range=0
aref=0
model=scicos_model()
model.sim=list('rt_comedi_datain',4)
if exists('outport') then model.out=ones(outport,1), model.in=[], else model.out=1, model.in=[], end
model.evtin=1
model.rpar=[]
model.ipar=[ch;
range;
aref;
length(name);
ascii(name)']
model.dstate=[1];
model.blocktype='d'
model.dep_ut=[%t %f]
exprs=[sci2exp(ch),name,sci2exp(range),sci2exp(aref)]
gr_i=['xstringb(orig(1),orig(2),["COMEDI A/D";name+" CH-"+string(ch)],sz(1),sz(2),"fill");']
x=standard_define([3 2],model,exprs,gr_i)
end
endfunction
```

References

[1] Dynamic Observers as Asymptotic Limits of Recursive Filters: Special Cases, 1988, Baras J. S., Bensoussan A., System & Control Letters, Vol.. 52, no 1, 7-15.

[2] High-gain and Non High-gain Observers for Nonlinear Systems, 2002, Busvelle E., Gauthier J-P., Contemporary Trends in Nonlinear Geometric Control Theory and its Applications, World Scientific, 233-256.

[3] Efficiency of the Extended Kalman Filter for Nonlinear Systems with Small Noise, 1991, Picard J., SIAM J. Appl. Math., 51, No 3, 843-885.

[4] A Simple Observer for for Nonlinear Systems, 1992, Gauthier J-P., Hammouri H., Othman S., IEEE Trans. Aut. Control, 37, 875-880.

[5] High-gain Estimation for Nonlinear Systems, 1992, Deza F., Busvelle E., Gauthier J.P., Rakotopara D., Systems & Control Letters 18, 25-299.

[6] Deterministic Observation Theory, 2001, Gauthier J-P., Kupka I., Cambridge university press.

[7] Adaptive-gain Extended Kalman Filter: Application to a Series-connected DC Motor, 2007, Boizot N., Busvelle E., Gauthier J-P., Sachau J., Conference on Systems and Control (CSC'07), Marrakech.

[8] Adaptive-gain Observers and Applications in Nonlinear Observers and Applications, 2007,Boizot N., Busvelle E., Lecture Notes in Control and Information Sciences, Springer.

[9] http://www-lar.deis.unibo.it/people/gpalli/

[10] www.rtai.org/

[11] www.comedi.org/

[12] RTAI-Lab tutorial: Scilab, Comedi, and real-time control, 2006, Bucher R., Mannori S., Netter T., www.rtai.org/RTAILAB

[13] Rapid Control Prototyping with Scilab/Scicos and Linux RTAI, 2004, Bucher R., Dozio L., in Scilab-2004, First Scilab International Conference.

[14] Hyperion Prop Talk, Connolly P., http://aircraft-world.com/prod_datasheets/hp/emeter/hp-proptalk.htm

[15] Personal communication from Roberto Bucher

[16] Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations, 1993, Radhakrishnan K., Hindmarsh A. C., LLNL report UCRL-ID-113855.

[17] Numerical Recipes: The Art of Scientific Computing, Third Edition, 2007, Cambridge University Press.

[18] Nonlinear Control of a Series DC Motor: Theory and Experiment, 1988, Mehta S., Chiasson J., IEEE transactions on idustrial electronics, Vol.45, No.1.

[19] Modeling and Simulation in Scilab/Scicos, 2006, Campbell S. L., Chancelier J-P., Nikoukhah R., Springer.