Implementation and validation of a Data-Driven Fast Measurement Method

Gael Mukendi Tshisuaka

***Abstract*— In this project, a method for predicting the steady-state value of a measured temperature from a small number of observations is implemented and validated. In contrast with other methods, the one presented in this paper does not need system identification and computes directly the quantity of interest. The results of both simulations and real-life experiments show that the method speeds-up effectively the measurement process as expected. In this paper, method is presented for a first order system but can be easily extended to multi-order system. Matlab is used for the implementation of the algorithms and the Lego Mindstorm setup is used as testbed.**

Keywords: steady-state, system identification, finite difference, observations.

I. INTRODUCTION

The motivation of this project is the following problem :

"According to Newton's law of cooling, an object of higher temperature than its environment cools at a rate that is proportional to the difference in temperature. A thermometer reading $21°C$, which has been inside a house for a long time, is taken outside. After one minute the thermometer reads 15° C; after two minutes it reads 11° C. Using Newton's law of cooling and the thermometer readings, find the outside temperature."[5]

II. SOLUTION TO THE MOTIVATION PROBLEM

The Newton's law of cooling is defined as:

$$\frac{dy(t)}{dt} = k(y(t) - T_E), \tag{1}$$

where:

- $y(t)$ is the sensor temperature at time t;
- T_E is the temperature of the environment;
- k is a negative constant that depends on thermometer and environment.

The equation (1) is a first order non-homogeneous differential equation. The solution of this equation is:

$$y(t) = T_E + (y_0 - T_E)e^{kt} \tag{2}$$

The details about the resolution of this equation are given in the appendix A. In this equation y_0 is the initial temperature, *i.e* $y_0 = y(0)$.

Using the given data, we can write the following two equations for two time instants:

$$y(t_1) = T_E + (y_0 - T_E)e^{kt_1}$$

$$y(t_2) = T_E + (y_0 - T_E)e^{kt_2},$$

where $y(t_1)$ and $y(t_2)$ are respectively 15 °C and 11°C. We will first compute e^{kt} then we will deduce T_E. Solving these two equations with respect to T_E lead to

$$\frac{y(t_1) - T_0 e^{kt_1}}{1 - e^{kt_1}} = \frac{T(t_2) - T_0 e^{kt_2}}{1 - e^{kt_2}} \tag{3}$$

Recall that $t_1 = \frac{1}{60}h$ and $t_2 = \frac{1}{30}h$. Let assume this change of variable:

$$e^{k/30} = p^2 \tag{4}$$

Thus, the equation becomes:

$$\frac{15 - 21p}{1 - p} = \frac{11 - 21p^2}{(1-p)(1+p)}$$

With $p \neq 1$.
What leads to $p = -2/3$.
Substituting this value in equation (2), we obtain $T_E = 3$°C.

Hereafter, we will solve a more general problem in which we are given a sequence of output observations $(y(1), \dots, y(T))$ and we will have to find the input $\overline{u}$.

III. STATE-SPACE MODEL

From the dynamics of the system described in the equation (1), we can derive the state-space model as following:
Let's $y(t) = x$, so that $\frac{dy(t)}{dt} = \dot{x}$, which leads to:

$$\dot{x} = kx - kT_E$$
$$y = x$$

Adopting the usual notation (x for the state, u for the input and y for the output), we have:

$$\begin{cases} \dot{x} = kx - ku \\ y = x \end{cases} \tag{5}$$

We recognize the state-space model (a, b, c, d) with $a = k$, $b = -k$, $c = 1$ and $d = 0$. We will use this state-space model to simulate trajectories in MATLAB in order to simulate and validate the method used in this paper.

The naive estimator uses the device's measurement without any processing, i.e, $\widehat{y} = y$.

IV. MODEL-FREE METHOD

The naive estimator uses the device's measurement without any processing, i.e, $\widehat{y} = y$. In order to reduce the transient time when using the naive estimator (the direct device's measurement), *i.e* to speed-up the measurement process, we will use the model-free method [1]. The purpose of this method is to avoid the system identification and to compute directly the quantity of interest $\overline{u}$.

To apply this method, first the signal difference for each output measurement is computed:

$$\Delta y = (y(2) - y(1), y(3) - y(2), ..., y(T) - y(T-1)), \quad (6)$$

where T is the maximum number of samples.

Next, we derive the block Hankel matrix as follows:

$$\mathscr{H}(\Delta y) = \begin{pmatrix} \Delta y(1) & \Delta y(2) & \dots & \Delta y(n_{max}) \\ \Delta y(2) & \Delta y(3) & \dots & \Delta y(n_{max+1}) \\ \vdots & & \ddots & \vdots \\ \Delta y(T - n_{max}) & \dots & \dots & \Delta y(T) \end{pmatrix}$$

As we have a first order system (we choose $n_{max} = 1$)[1], the block-Hankel Matrix is reduced to:

$$\mathscr{H}(\Delta y) = \begin{pmatrix} \Delta y(1) \\ \Delta y(2) \\ \vdots \\ \Delta y(T-1) \end{pmatrix} \quad (7)$$

.

To compute the quantity of interest $\overline{u}$, we have to solve the linear system of equations:

$$[1_{T-1} \otimes G \;\; \mathscr{H}] \begin{pmatrix} \overline{u} \\ l \end{pmatrix} = \begin{pmatrix} y(2) \\ \vdots \\ y(T) \end{pmatrix}$$

As the dc-gain $G = 1$, the equation becomes:

$$\begin{pmatrix} 1 & \Delta y(1) \\ 1 & \Delta y(2) \\ \vdots & \vdots \\ 1 & \Delta y(T-1) \end{pmatrix} \begin{pmatrix} \overline{u} \\ l \end{pmatrix} = \begin{pmatrix} y(2) \\ \vdots \\ y(T) \end{pmatrix} \quad (8)$$

.

Where $\overline{u}$ is the predicted value of the input in steady-state and l is a non relevant variable.

Let A, m and z be:

$$A = \begin{pmatrix} 1 & \Delta y(1) \\ 1 & \Delta y(2) \\ \vdots & \vdots \\ 1 & \Delta y(T-1) \end{pmatrix}, m = \begin{pmatrix} y(2) \\ \vdots \\ y(T) \end{pmatrix}, z = \begin{pmatrix} \overline{u} \\ l \end{pmatrix} \quad (9)$$

The equation has thus the standard form $Az = m$. The equation (8) is a least-square problem and can be solved as follows: Let's multiply both sides by A^T to write:

$$z = (A^T A)^{-1} A^T m \quad (10)$$

For the case $n = 1$, it is possible to do a few computations (see the appendix B for details), the value of interest, $\overline{u}$ is:

$$\overline{u} = \frac{1}{D} \sum_{i=1}^{T-1} (\Delta y(i))^2 \sum_{i=1}^{T-1} y(i+1) - \frac{1}{D} \sum_{i=1}^{T-1} \Delta y(i) \sum_{i=1}^{T-1} \Delta y(i) y(i+1) \quad (11)$$

[1]In fact, n_{max} is by definition (see [1]) an upper bound for the order n of the system (here n = 1), therefore $n_{max} \geq 1$.

where $D = (T-1)\sum_{i=1}^{T-1} (\Delta y(i))^2 - \left(\sum_{i=1}^{T-1} (\Delta y(i)\right)^2$

The formula (11) is equivalent to (10) for $n = 1$, it can be used to implement and validate the method in a software in which there are not all operations on matrices, such as inverse (for example in C). This formula can also be put directly in a digital signal processor (Lego NXT).

From equation (8) it can be seen that if we have only one sample of the output measurement $y(1)$, Δy is not possible to calculate and there is no equation to solve. If we have two output measurements $y(1)$ and $y(2)$, there is only signal difference $\Delta y(1) = y(2) - y(1)$, *i.e*, one equation for two unknowns, which leads to an infinity of solutions. For three output measurement, we have exactly two equation for two unknowns. From four measurements or more, we have more equations than unknowns and the estimations of unknowns can be done in the least-squares estimation sense (equations (10) and (11)).

The algorithm of the model-free method is:

- **input**: output observations $y(1), \dots, y(T)$

1) compute the signal difference Δy
2) solve the problem (8).

- **output**: $\widehat{u}$

V. Validation of the Model-Free Method

A. Simulation without noise

For the simulation, the state-space model of equation (5) is used in order to simulate some trajectories. First, we do simulation for exact data, then we will add some noise to see its influence on the method. The data are collected over 60 seconds with sampling rate of 1 Hz. We set $k = -0.4$, the initial temperature (initial state) $x(0) = -1$ and the steady-state value is $4°$. The results in Figure 1 show that the data-driven fast measurement method (DDFM) achieves better performance than the naive estimator which means that the measurement process has been improved as we expected. In Figure 1 and following Figures of this paper, the naive estimator is in blue and the DDFM in red.

B. Simulation with noise

Next, we add some noise to the exact data above to see how the method is affected. We assume that the noise is white, Gaussian (normal distribution) and zero mean. To see the influence of the noise on the DDFM method, the average error is used:

$$e = \frac{1}{N} \sum_{i=1}^{N} \|\overline{u} - \widehat{u}_i(t)\|_1, \quad (12)$$

where N is the number of independent noise realizations, $\widehat{u}$ is the estimation of $\overline{u}$ using the data $(y(1), \dots, y(T))$ and $\|.\|_1$ is the 1-norm. In our experiments, we took $N = 100$. The noise variance is set to 0.02. The results are shown in Figure 2. The performance is measured according to the average error: the smallest average error the best performance. It can be seen

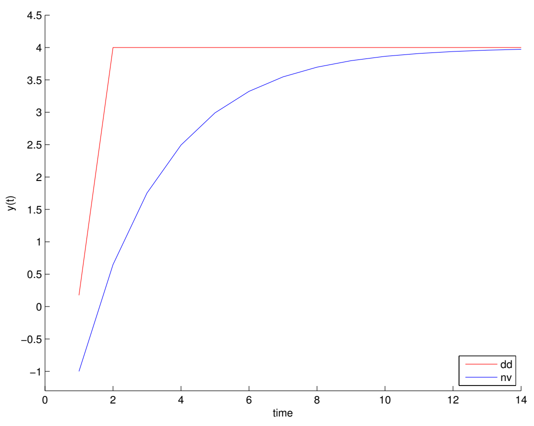

Fig. 1: Result for exact data

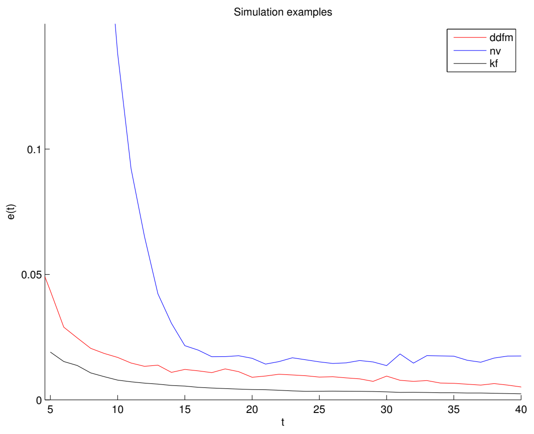

Fig. 2: Average error for different methods

that performance of the DDFM is once again better than the naive estimator. The black line in the Figure 2 is the result of the Kalman filter method. As it can be seen, it has better performance than DDFM. In fact, the Kalman filter has "more information" than the DDFM:

"it uses the knowledge of the process dynamics of the system." [2][1]

The study of the Kalman filter is out of the scope of this paper, it is shown only for illustration purposes. A full explanation of the algorithm of the Kalman filter is done in [5] and [2].

C. *The impact of the noise variance*

In the previous experiment, we have set the noise variance to 0.02. When the noise variance increases, the experiments show that the average error for all methods increases also. However, the performance of DDFM is better than the naive estimator if the noise variance s is such that $s \leq 0.2$. When $s \geq 0.3$, the performance of DDFM is slightly degraded. In fact, we can see in Figure 4 that at the first 5 measurements, the error of DDFM is higher than the one of the naive estimator. However, the average error of DDFM tends to zero and the one of the naive estimator does not. Figures 3 ($s = 0.2$) and 4 ($s = 0.7$) illustrate these observations.

Noise variance	Naive	DDFM	Kalman filter
0.02	e = 0.01	e = 0.002	e = 0.001
0.2	e = 0.17	e = 0.04	e = 0.019
0.7	e = 0.5	e = 0.11	e = 0.08

TABLE I: Average error

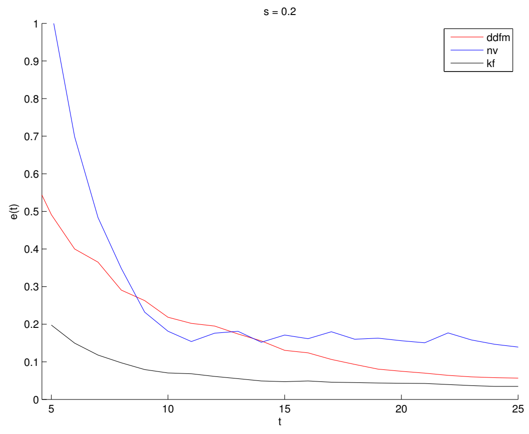

Fig. 3: impact of the noise variance

Note that the Kalman filter still have the best performance. Table I shows the value of the average error for different noise variances.

D. *Real temperature measurements*

For the validation of the method in real temperature measurements, we use data taken with NXT and EV3 Lego bricks. The results for different experiment are shown in Figure 5. The green line is the result obtained with the DSP, this result matches perfectly with the one obtained with Matlab.

Table II gives more information about different measurement processes of Figure 5.

Here again, Figure 5 show that the Model-Free method achieves better performance that the naive estimator. However, if we compare Figure 1 to Figure 5 we see that

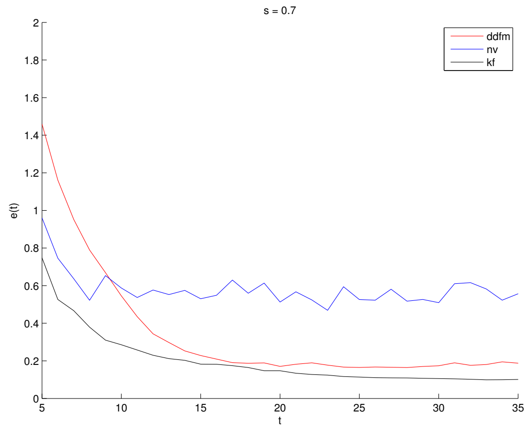

Fig. 4: impact of the noise variance

Experiment	Figure	initial temp	final temp	sampl. time
1	up-l	27.8	68.9	0.3s
2	up-r	24.88	69	0.3s
3	down-l	20	43.4	0.3s
4	down-r	25.26	44.1	0.3s

TABLE II: Experiments information

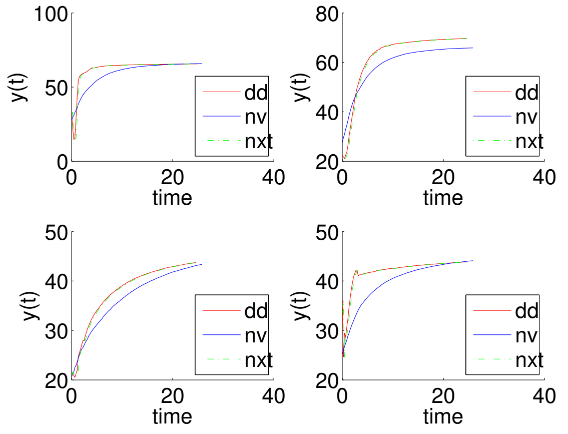

Fig. 5: Real measurements

the performance of the DDFM is not as good as in the simulation.

"This is caused by the combination of quantization errors and use of finite differences." [1].

E. Impact of the noise

As in the simulation, we will add noise noise to the data and observe how the method if affected. The experiments show that the method is efficient when the noise variance is less than 0.2. From $s = 0.3$, the performance of the method is deteriorated. Figures 6, 7 and 8 illustrate this observations.

F. Implementation of the algorithms

For the implementation and validation of the method, Matlab is used. The algorithms are principally based on the formula (11). However, for the Kalman filter used in the simulation set up, we used the code based

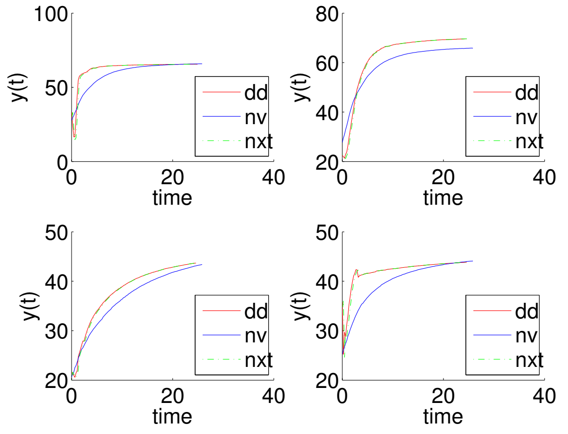

Fig. 6: impact of noise, $s = 0.01$

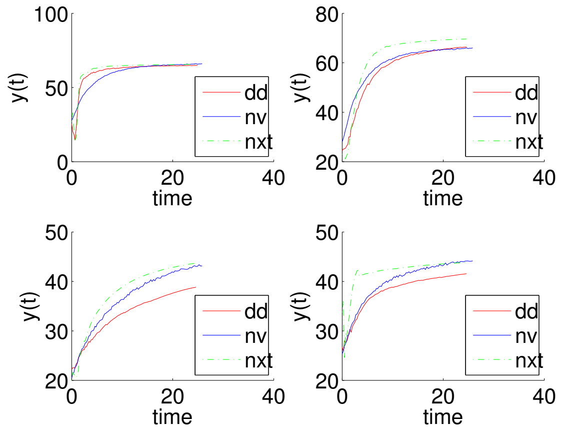

Fig. 7: impact of noise, $s = 0.2$

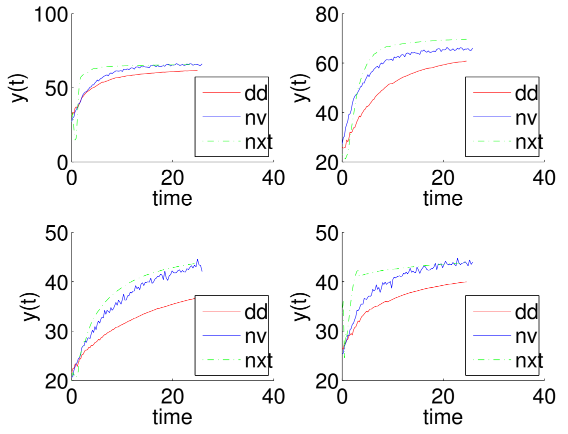

Fig. 8: impact of noise, $s = 0.45$

on the work of Ivan Markovsky and can be found at `http://homepages.vub.ac.be/~imarkovs/sensor-cep-experiments.html`. However, we adapted it for the first order system used in this paper.

The table below summarizes the algorithms used in this paper.

The input "sys" in the Kalman filter method is the state-space model and v is the noise variance. For the naive estimator, we have just to plot the output observations as said in section IV. The simulation experiments are in the file "*test_methods.m*" and the real measurements experiments are in the file "$testddfm.m$".

Method	Implementation	Input(s)
DDFM using eq. 11	true_ddfm	y
Kalman filter	stepid_kf	y, sys, v

TABLE III: Matlab functions

VI. Conclusion

The implementation and validation of a method to improve the measurement devices for first order systems is done. The required mathematical computations and algorithms are presented. The method has been tested in the case of temperature measurement and the results met the expectations. Matlab was used for algorithms implementation, the NXT and EV3 Lego bricks were used to collect data. In both case of noiseless data and noisy data, the DDFM method achieves better performance than the naive estimator and thus speeds-up the measurement process. In both simulations and real measurements, a high variance has as consequence to degrade the method. Several experiments show that the performance of the method is reduced in real-life measurements due to the use of finite difference and the quantization errors.

Appendix

A. Resolution of differential equation (1)

We consider the following equation:

$$\frac{dT(t)}{dt} = k(T(t) - T_E). \tag{13}$$

We solve this equation using the variation of the constant method, see [6].

Let first consider the homogeneous differential equation $\frac{dT(t)}{dt} = kT(t)$. It comes:

$$\begin{aligned} \frac{dT(t)}{T(t)} &= kdt \\ \Rightarrow \int \frac{dT(t)}{T(t)} &= k\int dt \\ \Rightarrow T(t) &= Ce^{kt}, \end{aligned}$$

where C is an arbitrary constant. We assume now that $C = C(t)$ (the main machinery of the variation of the constant method), therefore:

$$\begin{aligned} y(t) &= C(t)e^{kt} \\ \Rightarrow \frac{dy(t)}{dt} &= C'(t)e^{kt} + kC(t)e^{kt} \end{aligned}$$

Introducing the latter in (13) yields:

$$C(t) = T_E e^{-kt} + C_1$$

The constant C_1 is determined by the initial condition $T(0) = T_0$ (see section I). Finally, we have:

$$y(t) = T_E + (T_0 - T_E)e^{kt}$$

B. The solution of equation (8)

Using the equation (10), we have:

$$\begin{pmatrix} \bar{u} \\ l \end{pmatrix} = \left(\begin{pmatrix} 1 & \Delta y(1) \\ 1 & \Delta y(2) \\ \vdots & \vdots \\ 1 & \Delta y(T-1) \end{pmatrix}^T \begin{pmatrix} 1 & \Delta y(1) \\ 1 & \Delta y(2) \\ \vdots & \vdots \\ 1 & \Delta y(T-1) \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 & \Delta y(1) \\ 1 & \Delta y(2) \\ \vdots & \vdots \\ 1 & \Delta y(T-1) \end{pmatrix}^T \begin{pmatrix} y(2) \\ \vdots \\ y(T) \end{pmatrix}$$

$$\begin{pmatrix} \bar{u} \\ l \end{pmatrix} = \begin{pmatrix} T-1 & \sum_{i=1}^{T-1} \Delta y(i) \\ \sum_{i=1}^{T-1} \Delta y(i) & \sum_{i=1}^{T-1} (\Delta y(i))^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{T-1} y(1+i) \\ \sum_{i=1}^{T-1} y(i+1)\Delta y(i) \end{pmatrix}$$

Recall that the invert of a $2x2$ matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is given by $\frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

Therefore, the right side of the equation becomes:

$$\frac{1}{D}\begin{pmatrix} \sum_{i=1}^{T-1} (\Delta y(i))^2 & -\sum_{i=1}^{T-1} \Delta y(i) \\ -\sum_{i=1}^{T-1} \Delta y(i) & T-1 \end{pmatrix} \begin{pmatrix} \sum_{i=1}^{T-1} y(1+i) \\ \sum_{i=1}^{T-1} y(i+1)\Delta y(i) \end{pmatrix},$$

where $D = (T-1)\sum_{i=1}^{T-1} (\Delta y(i))^2 - \left(\sum_{i=1}^{T-1} (\Delta y(i)\right)^2$.

Finally the value of interest $\bar{u}$ is

$$\bar{u} = \frac{1}{D}\sum_{i=1}^{T-1} (\Delta y(i))^2 \sum_{i=1}^{T-1} y(i+1) - \frac{1}{D}\sum_{i=1}^{T-1} \Delta y(i) \sum_{i=1}^{T-1} \Delta y(i) y(i+1)$$

References

[1] I. Markovsky *Comparison of adaptive and model-free methods for dynamic measurement. IEEE Signal Proc. Letters, 22:1094-1097* 2015.

[2] I. Markovsky *An application of system identification in metrology. Control Engineering Practice, 43:85-93* 2015 Random House, N.Y.

[3] D. G. Luenberger *Introduction to Dynamical Systems: Theory, Models and Applications. John Wiley* 1979: International Secretariat, Institute of Pacific

[4] John C. Hansens *Power Programming: Robotics in C* Second Edition.

[5] Greg Welch, Gary Bishop *An Introduction to the Kalman Filter* . 2001

[6] Robert Marik. *First order linear differential equation Variation of constant Interactive tests* July 2006