UNIVERSITY OF TRENTO - Italy
Information Engineering
and Computer Science Department

Master Degree in Computer Science

Laboratory of Applied Robotics

# Identification procedure of Lego Mindstorm Motors

Deliverable 1

Antonio Sciarretta, Davide Todeschi, Elena Tumanov

November 22, 2015

# 1. Introduction

## 1.1. General information about LEGO® MINDSTORMS® Education

LEGO® MINDSTORMS® Education is the next generation in educational robotics, enabling students to discover Science, Technology, Engineering and Mathematics in a fun, engaging, hands-on way. By combining the power of the LEGO building system with the LEGO MINDSTORMS Education technology, teams of students can design, build, program, and test robots. Working together on guided and open-ended engineering projects, the team members develop creativity and problem-solving skills along with other important mathematics and science knowledge. Students also become more skilled in communication, organization and research, which helps prepare them for future success in higher levels of schooling and in the workplace.

LEGO MINDSTORMS Education features an advanced 32-bit computer controlled NXT brick, Interactive Servo Motors, Sound, Ultrasonic and other sensors, Bluetooth communication and multiple downloading capabilities. The icon-based LEGO MINDSTORMS Education NXT Software is built on the LabVIEW™ software from National Instruments, an industry standard with applications in many engineering and research fields. [1]

## 1.2. nxtOSEK

nxtOSEK is an open source platform for LEGO MINDSTORMS NXT. nxtOSEK consists of device driver of leJOS NXJ C/Assembly source code,TOPPERS/ATK (Automotive Kernel, formerly known as TOPPERS/OSEK) and TOPPERS/JSP Real-Time Operating System source code that includes ARM7 (ATMEL AT91SAM7S256) specific porting part, and glue code to make them work together.

nxtOSEK can provide:

- ANSI C/C++ programming environment by using GCC tool chain.
- C API for NXT Sensors, Motor, and other devices.
- C++ API for NXT Sensors and Motor which include many third party sensors.
- TOPPERS/ATK provided real-time multitasking features proven in automotive industry.
- TOPPERS/JSP provided real-time multitasking features complied with Japan original open RTOS specification μITRON 4.0.
- Fast execution and less memory consumption (nxtOSEK program is executed natively on the ARM7 and nxtOSEK itself consumed totally just about 10Kbytes). [2]

## 1.3. ScicosLab

ScicosLab (http://www.scicoslab.org) is a free open-source software package for scientific computation. ScicosLab includes a fork of Scilab, based on Scilab 4, the modeling and simulation tool Scicos and a number of other toolboxes.

Scilab is an interpreted language specifically developed for matrix based numerical computations. It includes hundreds of general purpose and specialized functions for numerical computation, organized in libraries called toolboxes that cover such areas as simulation, optimization, systems and control, and signal processing. These functions reduce considerably the burden of programming for scientific applications. [3]

# 2. Collecting data from motor

## 2.1. NXT Brofist directory changes

At the beginning, we set command Set Power to return the current values of the counter from the motor (degrees).

In the server part, we sent an input step function and after the power, which was 50, we send more than 1000 commands for multiple times (from power 30 to 70).

Because Bluetooth connection is very slow, we chose the following way:

- We created a buffer in the client part, which has the capacity to store 1000 packets/commands (Set power commands).
- Stored simultaneously all set power commands and after elaborated all at once.
- Between the execution of two successive commands we set a wait time of 2 ms.
- After the execution, we sent back all the results.
- Set power instructions returned the current values.

We modified the packets to add the information about the time.

## 2.2. Collecting data through Bluetooth

Client side modifications:
Added Buffer to store a huge number of instructions. This let us to use Bluetooth only twice per experiment.
Implemented the interface to return also the current measure NXT timestamp.
Modified Set power commands, to return information about current counter and time.

Server side modifications:
Implement the interface to include also the current measure NXT timestamp.
Created Set power commands, which gave a step function with Power 50.

1. Send message from PC via Bluetooth to brick that define motor power(that determines the speed).
2. Waited for the elaboration.
3. Receive message from brick with time, input power, tachometer count.
5. Elaborated the speed.

$$Speed = (Counter - PreviousCounter) / (Time - PreviousTime) \quad (1)$$

6. Saved data in a .csv file that contains information received from the brick. We collected the following information: time, input power, tachometer count, speed (1)

# 3. Filtering data and estimating the parameters

To estimate the parameters we filtered the data using Chebyshev 2 filter and after this we estimated the parameters using a regular method proposed at the course.

## 3.1. Filtering

### 3.1.1. Chebyshev 2 filter

We used Chebyshev 2 filter of order 1 and a cut-off frequency of 0.04.
In the following figure fig. 1., you can see the comparison between Raw Data and Filtered Data for the Power 40.

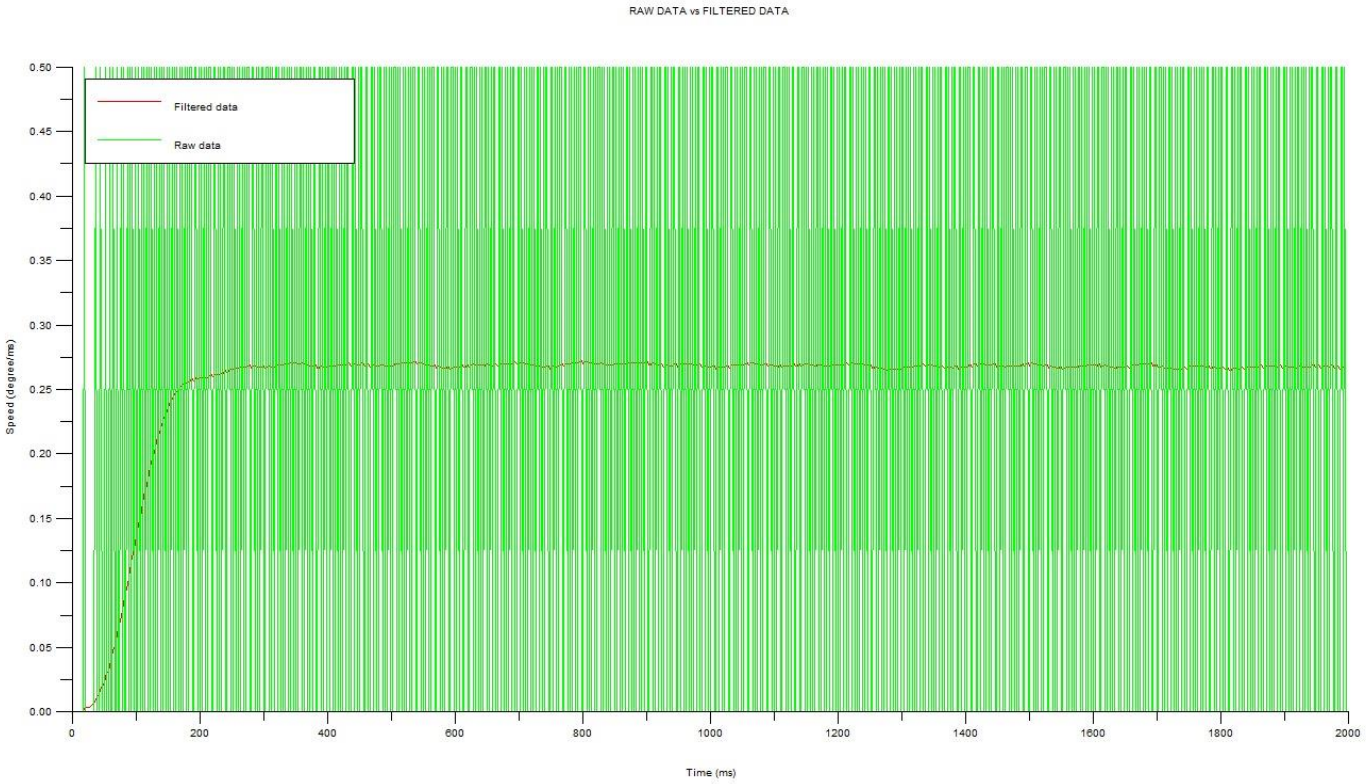


Fig.1 Raw Data and Filtered Data at Power 40

### 3.1.2. Chebyshev instead of Butterworth

Chebyshev equal ripple magnitude have a better rate of attenuation beyond the pass-band than Butterworth but is considerably more ringing in step response than Butterworth.

This filter response has the steeper initial rate of attenuation beyond the cutoff frequency than Butterworth.This advantage comes at the penalty of amplitude variation(ripple) in the pass-band. Unlike Butterworth and Bessel response, which have 3dB attenuation at the cutoff frequency,Cebyshev cutoff frequency is defined as the frequency at which the response falls below the ripple band. For even-order filters, all riple is above the dc-normalized passband gain response, so cutoff is at 0dB. For odd-order filters, all riple is below the dc-normalized passband gain response, so cutoff is at -(ripple) dB. For a given number of poles, a steeper cutoff can be achieved by allowing more pass-band ripple. The Chebyshev has more ringing in its pulse response than the Butterworth - especially for high-ripple designs.

Butterworth filter have some overshoot and ringing in step response. This filter has the flattest possible pass-band magnitude response. Attenuation is -3dB at the design cutoff frequency. Attenuation beyond the cutoff frequency is a moderately steep -20dB/decade/pole.The pulse response of the Butterworth filter has moderate overshoot and ringing.

The Butterworth filter is completely defined mathematically by 2 parameters: Cutoff frequency and number of poles. The Chebyshev filter has a third parameter: Passband Ripple.

## 3.2. Parameter Estimation

We needed to estimate 3 parameters q, $\omega_n$, $\xi$ using the following formulas:

$$q = \text{Last speed value} \quad (1)$$

$$\xi = \sqrt{\frac{\log(\text{overshot})^2}{\pi^2 + \log(\text{overshot})^2}} \quad (2)$$

$$\omega_n = \frac{\log(\frac{alpha}{100}) - \log(\frac{1}{\sqrt{1-\xi^2}})}{-\text{settling time} * \xi} \quad (3)$$

We started evaluation from Steady state values of the filtered functions (f - filtered function).

1. Steady state value estimation = k_est;

*k_est = f($)/input signal ($);*

2. We evaluated the damping factor (ξ), called xi_est.

*xi_est = sqrt (log (Overshoot) ^2) / (pi^2 + log (Overshoot) ^2));* (2)

Where: *Overshoot = abs (max (f) - f($) ) / abs (f(0) - f($));* (4)

3. Natural frequency ($\omega_n$) which is caled o_n_est.

*o_n_est = (log (alpha/100) – log (N_bar)) / (-xi_est * Settling time)* (3)

Where: *N_bar= 1/ sqrt (1-xi_est^2);*

*alpha= 3;*

4. Made the estimations of our system using the following estimation formula:

*G_est = k_est / (s^2/o_n_est^2 + 2*xi_est/o_n_est*s+1);* (4)

We evaluated all previous values for 40 times from Power 30 to Power 70 and we made the mean of the values.

Result values are:

k_est= 0. 0087755 (Steady state value)

xi_est= 0. 8209478 (Damping factor)

o_n_est= 0.0165151 (Natural frequency)

In the following figure fig.2 you can see the difference between Estimation and Test for Power of 40, 50 and 60.

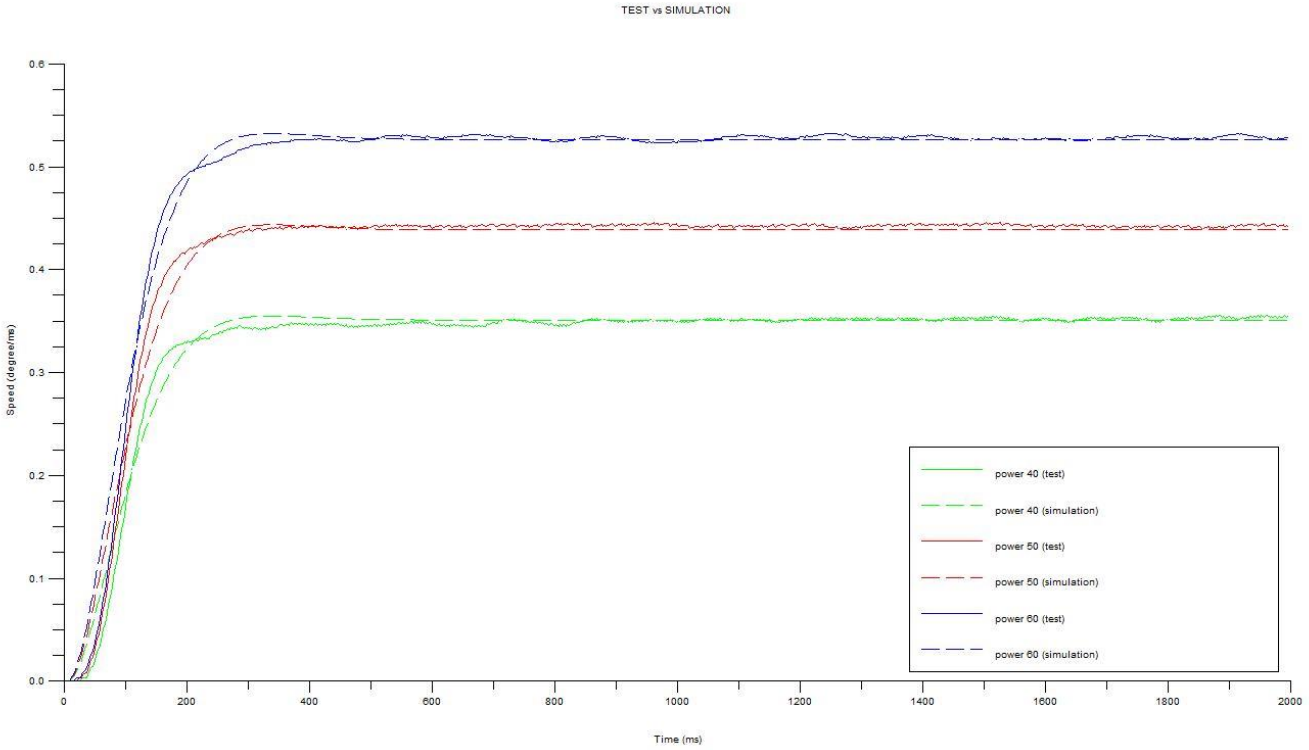


Fig. 2 Test vs Simulation (Power 40, 50, 60)

## 3.3. Performance indices to control system

A metric is needed to measure distances between observed and predicted data and, hence, to assess model fit. As a metric, we adopted some standard indices used as performance indices for control systems.

The indices formulas:

$$\text{Integral squared error (ISE)} = \int_0^T (y_m(t) - y(t))^2 dt; \quad (5)$$

$$\text{Integral absolute error (IAE)} = \int_0^T |y_m(t) - y(t)| dt; \quad (6)$$

$$\text{Integral time squared error (ITSE)} = \int_0^T t(y_m(t) - y(t))^2 dt; \quad (7)$$

$$\text{Integral time absolute error (ITAE)} = \int_0^T t|y_m(t) - y(t)| dt. \quad (8)$$

We tested our system three times with Power 40, Power 50 and Power 60.

Got the following errors calculating the average value of the above formulas ((5),(6),(7),(8)):

Power 40: AverageError = 557.45129

Power 50: AverageError = 998. 88904

Power 60: AverageError = 616.85092

Mean squared error:

Power 40: MSE = 0,08 degree/s

Power 50: MSE = 0.10 degree/s

Power 60: MSE = 0.13 degree/s

Used filter caused the big error at the beginning of the axis in the following figure because it have a big delay.

In the following figure fig.3 you can see the difference between Output Estimations and Input Estimations for Power of 40, 50 and 60.

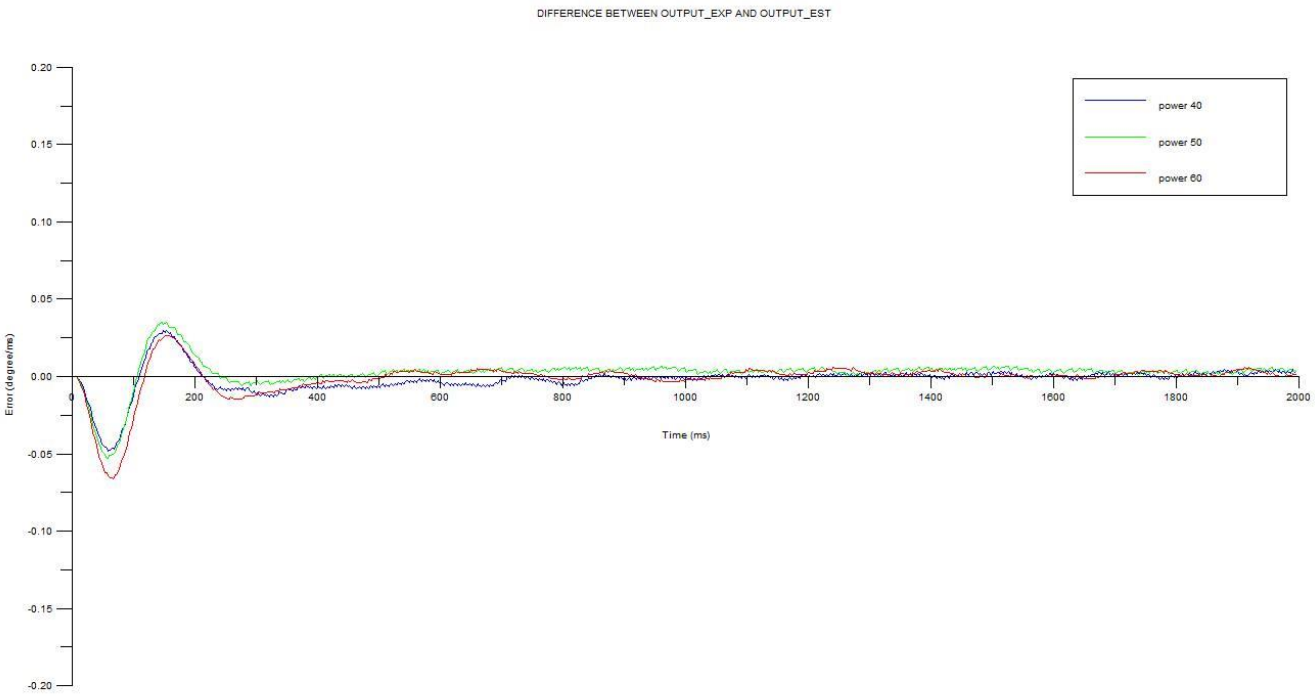


Fig. 3 Difference between Input and Output estimations

# 4. Conclusion

Our scope was to get and to identify parameters of the Lego NXT motors. Above we demonstrated our method of obtaining data from Lego Mindstorm motors and estimation of parameters from the motor data. Because of the quantity and quality of the collected data, it was possible to compare them to the estimated data. We used a Chenyshev filter to process the collected data and matched them to the estimated ones. With this filter, it was possible to have a better estimation than with butterworth, to receive plots that was easier to read.

Analysing Performance Indices for control systems we found acceptable received errors that were caused by used filter and by slow Bluetooth connection.

## References:


[1] http://www.generationrobots.com/media/Lego-Mindstorms-NXT-Education-Kit.pdf

[2] http://lejos-osek.sourceforge.net/whatislejososek.htm

[3] Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4, Stephen L. Campbell, Jean-Philippe Chancelier and Ramine Nikoukhah, Second Edition