World Academy of Science, Engineering and Technology
International Journal of Mechanical and Mechatronics Engineering
Vol:10, No:8, 2016

Application of Fourier Series Based Learning Control on Mechatronic Systems

Sandra Baßler, Peter Dünow, Mathias Marquardt

International Science Index, Mechanical and Mechatronics Engineering Vol:10, No:8, 2016 waset.org/Publication/10005018

Abstract—A Fourier series based learning control (FSBLC) algorithm for tracking trajectories of mechanical systems with unknown nonlinearities is presented. Two processes are introduced to which the FSBLC with PD controller is applied. One is a simplified service robot capable of climbing stairs due to special wheels and the other is a propeller driven pendulum with nearly the same requirements on control. Additionally to the investigation of learning the feed forward for the desired trajectories some considerations on the implementation of such an algorithm on low cost microcontroller hardware are made. Simulations of the service robot as well as practical experiments on the pendulum show the capability of the used FSBLC algorithm to perform the task of improving control behavior for repetitive task of such mechanical systems.

Keywords—Climbing stairs, FSBLC, ILC, Service robot.

I. Introduction

THE paper deals with the application of ILC-methods to improve motion control of mechatronic systems. The investigated ILC-methods where applied on two processes. The first process is a transport robot for supporting elder and disabled people. We consider the stair climbing problem as one example for recurring control tasks that should be adapted to the environmental conditions by ILC-methods. A requirement for these robots is that they have to be affordable. This paper is therefore addressed to the implementation of control algorithms in particular ILC on low cost hardware for service robots which perform repetitive motions during their work. Fig. 1 shows different configurations of such a robot which is intended for transporting goods. This robot is capable of climbing stairs which is a difficult task for the control system of the robot since it requires high torque dynamics and there are also changing environmental conditions and possibly limits in the drive performance. However, because of the repetitive character of the motion the control behavior can be improved by using ILC-Methods. The algorithm has to be suitable for low cost hardware with limited computation time and storage capacity. The second process is a propeller driven pendulum. This system was chosen as a realtime test bench because of its largely unknown non-linear process behavior.

FSBLC as a parametric iterative learning control method can satisfy the requirements given above. In Section II, the ILC-Method is briefly introduced. We describe the benefits and explain the use of a FSBLC-structure in a PD-control system. In Section III, the two application problems are explained. For the real time investigations we used an available propeller driven pendulum system. With respect to control engineering, the requirements for a repetitive starting, flying and landing procedure for this system are very similar to the stair climbing problem. Some facts about the implementation on low cost hardware are provided in Section IV. In Section V, some results of the investigation are discussed.

S. Baßler, P. Dünow and M. Marquardt are with the University of Applied Sciences Technology, Business and Design, Wismar, Germany (e-mail: sandra.bassler@hs-wismar.de).

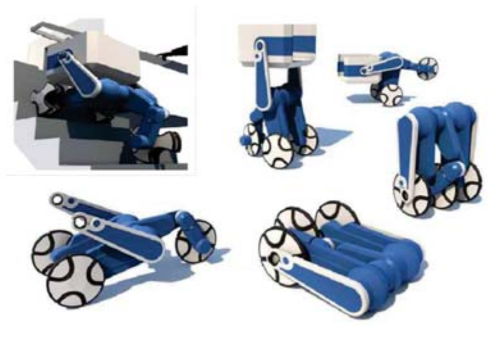
Fig. 1 Design study of a stair climbing service robot

II. Parametric Iterative Learning Control

Since the early eighties a variety of ILC methods was developed, but as mentioned we set the requirement that the control algorithms should be applicable for low cost hardware. Most of the ILC methods require storage for the entire trajectory of the manipulated variable. This leads to high increase in memory demand with increasing length and number of the trajectories which have to be learned. Parametric ILC methods avoid this problem by learning only a few parameters. This could be for example a number of Fourier coefficients of the manipulated variable trajectory, which will be transformed in the trajectory at the beginning of a new cycle. Another benefit of these methods is that the trajectory is filtered by parametrization, whereas other methods need an additional filter algorithm to disable the learning of higher frequencies, which often cause divergent behaviour. An introduction and overview about parametric ILC-methods can be found in [1] and [4], whereas Fourier series based methods are described in [3], [5] and [2]. On the one hand there are inversion based methods like secant method [3] or Model-Less Inversion-Based Learning Control (MIIC) [2]. On the other hand a method that learns the feed forward using the manipulated variable calculated by a simple PD controller is introduced in [5] for tracking control of nonlinear SISO systems. The implementation of this method, which is called FSBLC with PD controller, on problems described above is investigated in the following.

International Science Index, Mechanical and Mechatronics Engineering Vol:10, No:8, 2016 waset.org/Publication/10005018

A. FSBLC with PD Controller

Fig. 2 shows the control scheme of FSBLC with a PD controller, whereby the lower part is a simple control loop with a PD controller designed by standard methods. The upper part in Fig. 2 represents the learning algorithm that adapts the feed forward after each cycle. $y_d(t)$ is the desired output, $y(t)$ the actual output of the system and $e(t)$ the control error. The system input $u(t)$ is equal to the sum of the PD controller output $u_{PD}(t)$ and the estimate of the optimal feed forward $u_f(t)$. The learning algorithm determines the estimate of the optimal feed forward as follows. All samples of one cycle of the manipulated variable $u_{PD}(t)$ are stored in a memory $\mathbf{u}_{PD}$ to calculate a specified number of Fourier coefficients $\mathbf{U}_{PD}$ at the end of the cycle. These Fourier coefficients are multiplied with the learning gain γ and added to the Fourier coefficients of the previous cycle $\mathbf{U}_k$.

$$\mathbf{U}_{k+1} = \mathbf{U}_k + \gamma \mathbf{U}_{PD,k} \tag{1}$$

This results in the Fourier coefficients of the estimated optimal feed forward $\mathbf{U}_{k+1}$, which are used to determine the feed forward for the next cycle $\mathbf{u}_{k+1}$ by inverse Fourier transform.

B. Parametrization

The learning gain γ has to be chosen between zero and one. A higher gain leads to a faster convergence rate of the tracking error, whereas a smaller gain should be used to ensure stability. One drawback of a higher learning gain is learning non repetitive disturbances faster, which causes an additional control error in the next cycle without this disturbance. The number of Fourier coefficients has to be chosen considering the frequency components of the desired output trajectory and the system bandwidth. By using only selected frequency components the divergence causing frequencies can be suppressed and furthermore this can reduce computation time and storage requirements. In case of a large time delay each frequency component can be phase shifted to compensate for the delay like described in [6] for tracking control of a belt-driven system.

III. Process Description

A. Process 1

For a first investigation a model that includes the most significant problems of climbing stairs is required. Therefore we consider a robot with special wheels shown in Fig. 1 respectively the stair climbing wheel in Fig. 3 and make some simplifications.

We assume that there is a direct driven stair climbing wheel with a single arm perpendicular to the ground like a stabilized inverted pendulum shown in Fig. 4. Such a stabilized pendulum can easily be modelled with SimMechanics, however, the stair climbing is hardly implementable with this tool. To overcome this problem we estimate the additional torque requirement caused by a step and add this to the SimMechanics model. Considering the wheel in Fig. 4 respectively in Fig. 5 there can be seen that the required torque jumps up as soon as the wheel contacts the next step and has a minimum after a 90° rotation, since a simple rolling movement follows until the next step is contacted. In the following we determine the required torque caused by the gravitational force and neglect the moment of inertia, Coriolis and centrifugal force and friction, since this experiment is intended to investigate how effective the FSBLC with PD control can learn to handle unknown but recurrent torque requirements so we need only a rough estimate of the required torque trajectory. As well known the torque M [7] is calculated with

$$M = \mathbf{l} \times \mathbf{F} = |\mathbf{l}| \cdot |\mathbf{F}| \cdot sin(\gamma) \tag{2}$$

where $\mathbf{l}$ is the vector between the rotary axis and the point where the force is applied, $\mathbf{F}$ is the force, in this case the gravitational force, which is calculated as follows

$$F_g = m \cdot g \tag{3}$$

and γ is the angle between $\mathbf{l}$ and $\mathbf{F}$

$$\gamma = arcsin\left(\frac{r \cdot sin(\beta)}{l}\right). \tag{4}$$

This results in

$$M = r \cdot F_g \cdot sin(\beta). \tag{5}$$

The simulation experiment, which is shown in Fig. 6, is implemented in Matlab Simulink. A PD controller is used to control the velocity of the wheel that is simulated with SimMechanics by using torque as manipulated variable. The wheel has a radius of 183 mm and a weight of 2.2 kg, whereas the arm has a length of 800 mm and a weight of 4.5 kg. The task is learning to climb 3 stairs. Thereby Stateflow is used on the one hand to control the input of the additional torque that simulates the steps and on the other hand it controls the FSBLC, that means for example enabling the learning algorithm while climbing stairs and disable it for driving.

In this example, a learning of each separate step is also possible, since the stairs have the same dimensions and the distance between the rotary axis and the centre of mass l has the same trajectory for each step. However, considering a robot shown in Fig. 1, this trajectory is different at the beginning or end of the stairs. Furthermore the stairs could have different dimensions, therefore the entire stair should be learned as one trajectory.

The simulation experiment is performed as described before whereby the parameters of the FSBLC are a learning gain of 0.7 and the number of learned frequency components is 23. A higher number of frequency components causes divergent behaviour. The interval time is 4 s, this includes climbing the 3 stairs. After the stairs the wheel drives on a flat surface for 2 s, before the next stair climbing interval starts. Fig. 7 shows the simulation sequence for two cycles. It is assumed that the stairs are climbed with a constant velocity so that the additional torque can be implemented as a time depended trajectory. Of course there is no constant velocity and the torque is rather angle depended, but this requires some enhancements in future work (see Section VI). The velocity is regulated by the PD controller and should be constant $0.222\frac{rev}{s}$ also between the stair climbing intervals.

International Science Index, Mechanical and Mechatronics Engineering Vol:10, No:8, 2016 waset.org/Publication/10005018

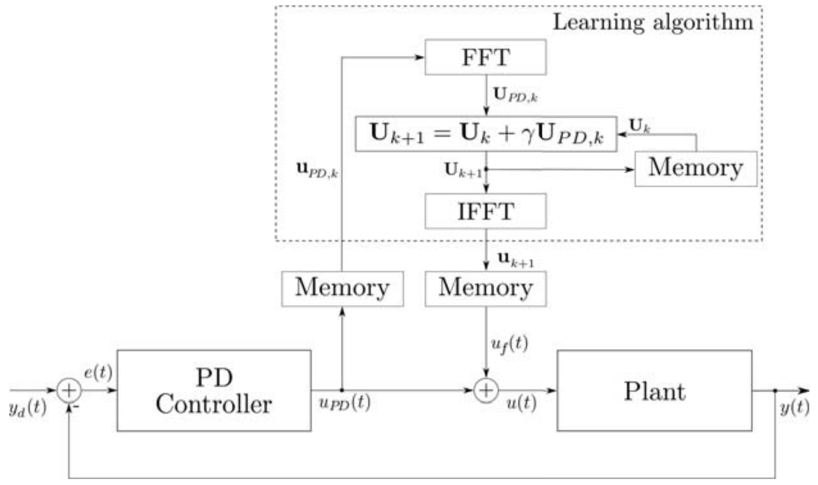

Fig. 2 PD-Controller with FSBLC [5]

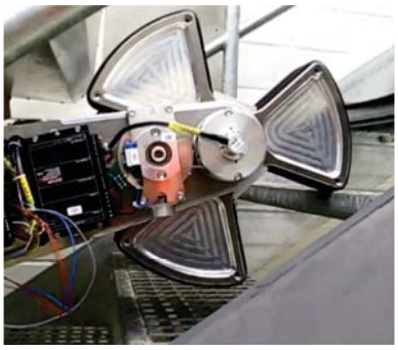

Fig. 3 Stair climbing wheel

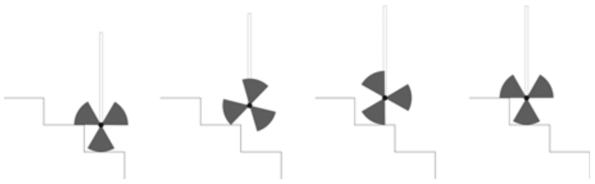

Fig. 4 Stair climbing sequence

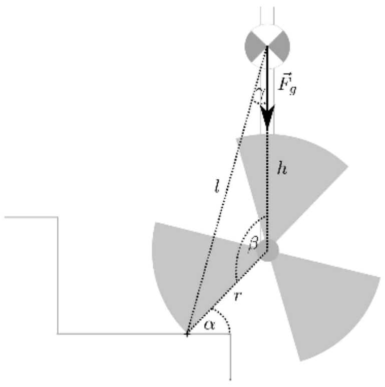

Fig. 5 Stair climbing wheel

B. Process 2

For the practical investigations we used an available propeller driven pendulum connected to an xPC Target system for control, which can be seen in Fig. 8. On the right side there is a propeller generating thrust perpendicular to the pendulum and on the left side a movable counterweight is fastened. The manipulated variable is the voltage of the propeller motor and the output is the angle of the pendulum. The task is that the system has to fly some maneuvers that means in this example:

1) starting
2) flying
3) landing.

Fig. 11 shows the desired trajectories for each maneuver. The management of these trajectories and the FSBLCs is done via Stateflow. With respect to control engineering, the requirements for this tasks are very similar to the stair climbing problem, since this is also a nonlinear process with an angle depending required torque. In contrast to the simulated process we have the influence of friction and some non repetitive disturbances like noise. There, an FSBLC with PD controller is used to improve following the trajectories of the maneuver. The learning gain is 0.5 and the first 10 frequency components are learned whereas the sampling time is 10 ms.

IV. Comments on Implementation

As already mentioned the algorithm has to be suitable for running on low cost hardware. For this reason it is implemented on an STM32F446RE microcontroller and the computation time is measured for different trajectory lengths and numbers of frequency components. The STM32F446RE is based on an ARM® Cortex®-M4 32-bit RISC core operating at a clock frequency of up to 180 MHz [10] and costs approximately 10 €. The most complex part of this algorithm is the calculation of the frequency components respectively the transform of the frequency components in a trajectory. These calculations can be performed using a discrete Fourier transform matrix $\mathbf{A}$ to calculate the Fourier transform

$$\hat{\mathbf{X}} = \mathbf{x} \cdot \mathbf{A} \tag{6}$$

and the inverse discrete Fourier transform matrix $\mathbf{A}_i$ to determine the inverse Fourier transform [8].

$$\hat{\mathbf{x}} = \hat{\mathbf{X}} \cdot \mathbf{A}_i \tag{7}$$

International Science Index, Mechanical and Mechatronics Engineering Vol:10, No:8, 2016 waset.org/Publication/10005018

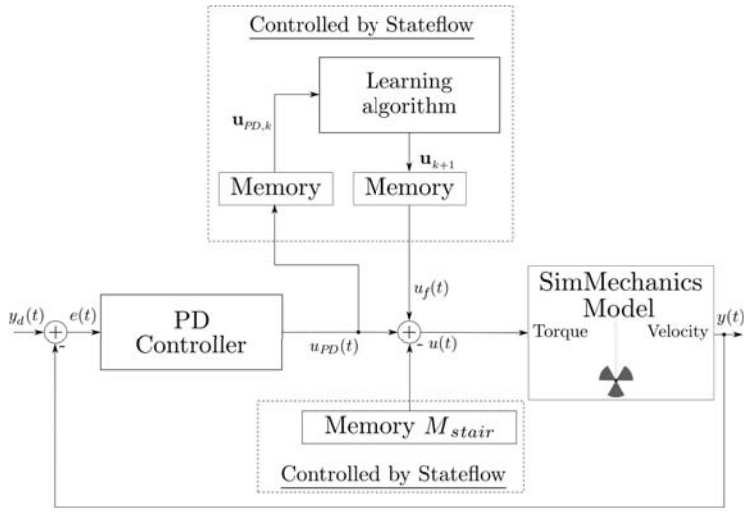

Fig. 6 Simplified Simulink model

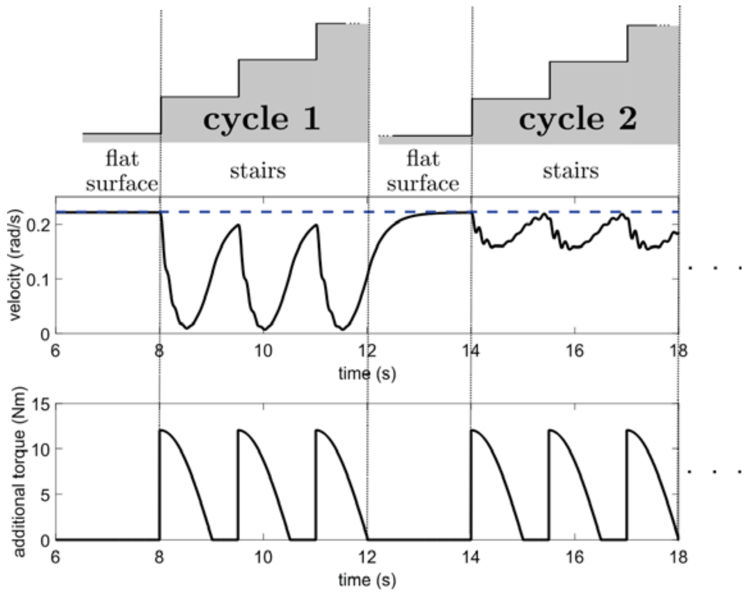

Fig. 7 Simulation sequence

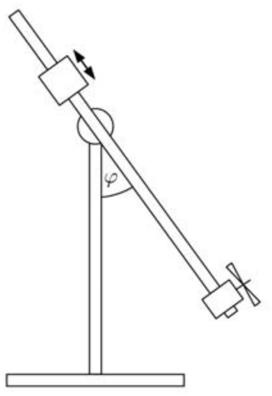

Fig. 8 Propeller driven pendulum

Both matrices can be calculated for example with Matlab by using the dftmtx function ($\mathbf{A} = dftmtx(n)$) respectively

$$\mathbf{A}_i = \frac{conj(dftmtx(n))}{n}$$

where n is the vector length [9]. These could be large square matrices with complex values, since the dimension depends on the trajectory length, which leads to a large number of multiplications and additions respectively a high computation time. Thus we need some assumptions. The trajectory $\mathbf{x}$ which should be transformed consists of only real values, therefore the result of the inverse transform $\hat{\mathbf{x}}$ is also real, since

$$\hat{\mathbf{x}} = \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{A}_i$$

and

$$\Im \{\mathbf{A} \cdot \mathbf{A}_i\} = \mathbf{0},$$

only the transform matrices and therefore the frequency components $\hat{\mathbf{X}}$ are complex. Thus we can calculate the real

International Science Index, Mechanical and Mechatronics Engineering Vol:10, No:8, 2016 waset.org/Publication/10005018

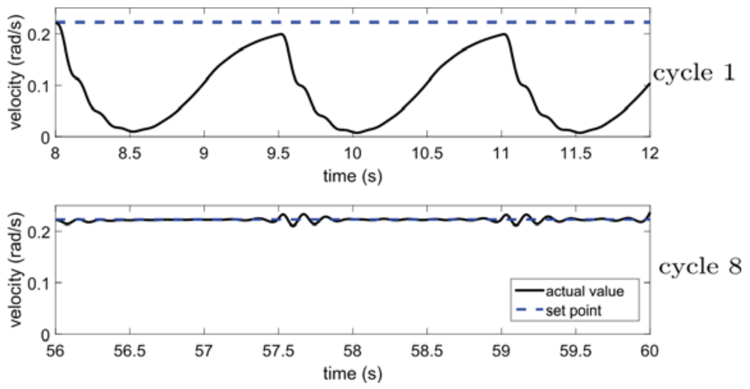

Fig. 9 Simulation results of cycle 1 and cycle 8

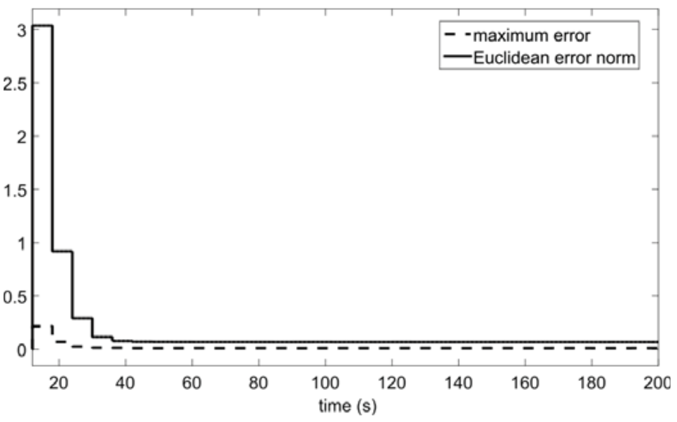

Fig. 10 Euclidean error norm and maximum error (simulation)

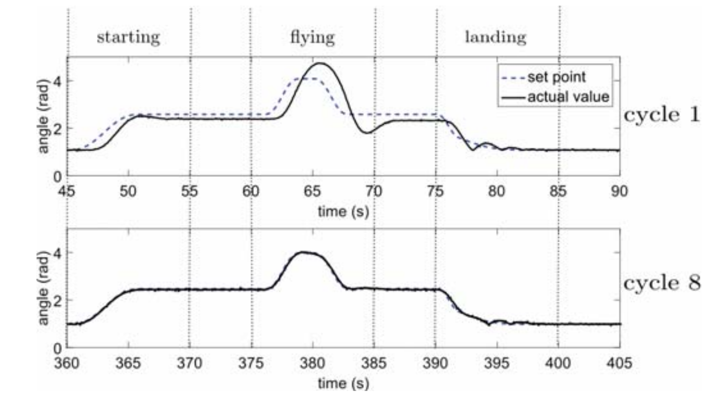

Fig. 11 Propeller driven pendulum results of cycle 1 and cycle 8

part and imaginary part separately, that means $\hat{\mathbf{X}}$ and $\hat{\mathbf{x}}$ can be calculated as follows

$$\Re\left\{\hat{\mathbf{X}}\right\} = \mathbf{x} \cdot \Re\left\{\mathbf{A}\right\} \tag{8}$$

$$\Im\left\{\hat{\mathbf{X}}\right\} = \mathbf{x} \cdot \Im\left\{\mathbf{A}\right\} \tag{9}$$

$$\hat{\mathbf{x}} = \Re\left\{\hat{\mathbf{X}}\right\} \cdot \Re\left\{\mathbf{A}_i\right\} - \Im\left\{\hat{\mathbf{X}}\right\} \cdot \Im\left\{\mathbf{A}_i\right\}. \tag{10}$$

Furthermore, $\hat{\mathbf{X}}$ is symmetrical that means the second half is complex conjugate to the first half (except the direct component)

$$\hat{\mathbf{X}} = \left[\begin{array}{cccccc} a_0 & a_1 + jb_1 & a_2 + jb_2 & \ldots & a_2 - jb_2 & a_1 - jb_1 \end{array}\right]$$

and the corresponding elements of $\mathbf{A_i}$ are also complex conjugate to each other. Thus we can multiply the frequency components (except the direct component) by two and omit the second part. As a result we only need half the number of columns of $\mathbf{A}$ respectively of rows of $\mathbf{A}_i$, plus one for the direct component. As already mentioned only a few frequency components are mostly required to learn, which reduces the number of required columns respectively rows, too. For example, if we like to learn the first l frequency components (including the direct component) of a vector $\mathbf{x}$

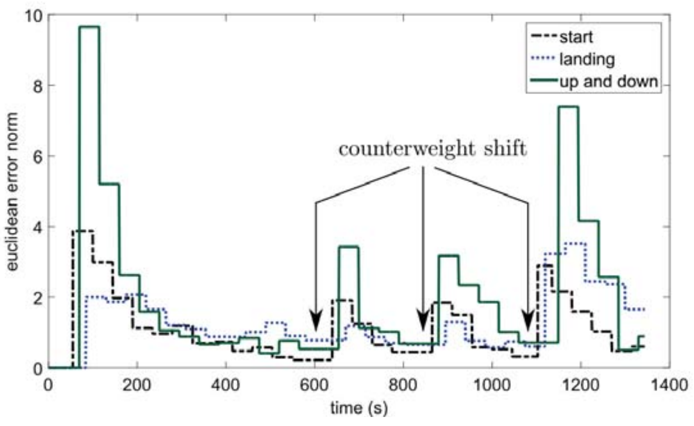

Fig. 12 Propeller driven pendulum Euclidean error norm

International Science Index, Mechanical and Mechatronics Engineering Vol:10, No:8, 2016 waset.org/Publication/10005018

with n elements, we have twice a multiplication of a vector with n elements and a $n \times l$ matrix for the Fourier transform resulting in two vectors with l elements (one vector for the real part and imaginary part) i.e. in pseudo code:

$$\begin{aligned}
&\mathbf{A} = dftmtx(n), && \mathbf{A} \in \mathbb{C}^{n\times n}\\
&\mathbf{g} = diag([1, 2\cdot ones(1, l-1)]), && \mathbf{g} \in \mathbb{R}^{l\times l}\\
&\mathbf{X}_{re} = \mathbf{x}\cdot real(\mathbf{A}(:, 1:l))\cdot \mathbf{g}, && \mathbf{x} \in \mathbb{R}^{1\times n}, \mathbf{X}_{re} \in \mathbb{R}^{1\times l}\\
&\mathbf{X}_{im} = \mathbf{x}\cdot imag(\mathbf{A}(:, 1:l))\cdot \mathbf{g}, && \mathbf{x} \in \mathbb{R}^{1\times n}, \mathbf{X}_{im} \in \mathbb{R}^{1\times l}
\end{aligned}$$

For the inverse transform we need two multiplications of a vector with l elements with a $l \times n$ matrix resulting in two vectors with n elements, which are added together to get $\hat{\mathbf{x}}$.

$$\begin{aligned}
\mathbf{A}_i &= \frac{conj(dftmtx(n))}{n}, && \mathbf{A}_i \in \mathbb{C}^{n\times n}\\
\hat{\mathbf{x}} &= \mathbf{X}_{re}\cdot real(\mathbf{A}_i(1:l, :))\\
&\quad - \mathbf{X}_{im}\cdot imag(\mathbf{A}_i(1:l, :)), && \hat{\mathbf{x}} \in \mathbb{R}^{1\times n}
\end{aligned}$$

The learning algorithm can be calculated in a separate task at the end of the cycle, thus it has not be executed in one sample. However, the calculation of $\hat{\mathbf{x}}$ has to be real-time capable, since more than one trajectory respectively the frequency components of these should be stored, for example for climbing stairs with different loads. Therefore the frequency components have to be read at the beginning of the cycle and to be transformed in $\hat{\mathbf{x}}$. Nevertheless only the value for the actual sample s has to be calculated. Thus the frequency components $\hat{\mathbf{X}}$ have to be multiplied only with the corresponding column of $\mathbf{A_i}$ in each sample.

$$\begin{aligned}
\hat{x}_s &= \mathbf{X}_{re}\cdot real(\mathbf{A}_i(1:l, s))\\
&\quad - \mathbf{X}_{im}\cdot imag(\mathbf{A}_i(1:l, s)), \qquad \hat{x}_s \in \mathbb{R}^{1\times 1}, s = 1...n
\end{aligned}$$

V. Results and Discussion

A. Process 1

In Fig. 9 the simulation results of the stair climbing process are shown whereby in the upper part the velocity set point and the actual value of the first cycle can be seen. There is a large control deviation caused by the additional torque which simulated the stairs, so that the wheel almost stops. The lower part of Fig. 9 presents the result of cycle 8 with a significant improvement of the control deviation. As shown in Fig. 10 both the Euclidean error norm and the maximum error converge to a final value, which is nearly the result of cycle 8.

Summarized the simulation experiment verified that FSBLC with PD controller is a well suitable method for learning the compensation of unknown torque requirements.

B. Process 2

Fig. 11 shows the practical investigation results of FSBLC with PD controller applied on the described propeller driven pendulum. Comparing the result of cycle one, where is no influence of the FSBLC at all and only the PD controller is working, with cycle 8, both shown in Fig. 11, a significant improvement of trajectory tracking of the 3 maneuvers can be seen. The Euclidean error norm in Fig. 12 indicates convergent learning, even for changed system parameters i.e. for a shifted center of mass due to moving the counterweight on the left side of the pendulum. Practical investigations confirm the effectiveness of FSBLC with PD controller in compensating unknown repetitive torque requirements. Additionally, a certain degree of robustness was experimentally verified due to the convergent learning after changing the system parameters.

C. Time Measurement Study

The described parts of the algorithm are implemented with trajectories of different lengths and different numbers of frequency components that have to be learned on the microcontroller for a time measurement study. The results are presented in Table I, where l is the number of frequency components, n is the trajectory length, t_1 is the computation time of the Fourier transform and t_2 is the time for calculating the inverse Fourier transform for one sample. There can be seen that the algorithm is suitable also for longer trajectories and higher numbers of frequency components, however, the higher frequency components are mostly omitted, due to generation of divergent behaviour. Furthermore, it is important to choose a suitable sample time, since it determines the trajectory length and thus the computation time. The maximum

International Science Index, Mechanical and Mechatronics Engineering Vol:10, No:8, 2016 waset.org/Publication/10005018

frequency that can be determined is half the sampling frequency and the resolution is the reciprocal interval time.

TABLE I
COMPUTATION TIME

Frequency components l	Trajectory length n	$t_1(ms)$	$t_2(\mu s)$
	100	0.3191	
5	500	1.581	1.88
	1000	3.191	
	100	0.6402	
10	500	3.181	2.68
	1000	6.362	
	100	1.4505	
30	500	7.282	6.6
	1000	14.505	

VI. Conclusion and Future Work

The simulation as well as the practical experiment and the comments on implementation verified FSBLC with PD controller as an appropriate way to improve the stair climbing problem. Future work will focus on investigation of the angle dependency of the additional torque for climbing stairs and under circumstances the development of methods for learning an angle dependent feet forward. Another aspect is the implementation of a supervision that disables or adapts the learning in case of divergent behaviour.

References

[1] Baqué, H., *Zyklische Regelung von Mehrgrößenprozessen*, Aachen: Shaker, 1999.
[2] Teng, K. and Tsao, T., *A comparison of inversion based Iterative Learning Control Algorithms*. American Control Conference (ACC), 2015, pp. 3564-3569.
[3] Manabe, T. and Miyazaki, F., *Learning control based on local linearization by using DFT*. IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems, 1991, pp. 639-646.
[4] Schmidt, F., *Analyse und Entwurf iterativ lernender Regelungen*, Aachen: Shaker, 1996.
[5] Huang, W., *Tracking control of nonlinear mechanical systems using Fourier series based learning control*, 1999.
[6] Zhu, Y., Zuo, W. and Cai, L. *Tracking control of a belt-driving system using improved Fourier series based learning controller*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 881–886.
[7] Hering, E. and Martin, R. and Stohrer, M., *Taschenbuch der Mathematik und Physik*, Berlin, Heidelberg: Springer-Verlag, 2009.
[8] Neubauer, A., *DFT - Diskrete Fourier-Transformation*, Wiesbaden: Vieweg+Teubner Verlag, 2012.
[9] http://de.mathworks.com/help/signal/ref/dftmtx.html.
[10] http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00141306.pdf.