Towards a Newer Toolbox for Computer Aided Polynomial Design of Sampled-Data Systems

Rudy Cepeda Gomez, Bernhard P. Lampe*

Institute of Automation, University of Rostock, 18051 Rostock, Germany; **bernhard.lampe@uni-rostock.de*

Simulation Notes Europe SNE 25(2), 2015, 79 - 84
DOI: 10.11128/sne.25.tn.10294
Received: August 15, 2015 (Selected ASIM STS 2015 Postconf. Publ.); Accepted: August 20, 2015;

Abstract. This work presents recent steps taken in order to update the DIRECTSD toolbox for MATLAB. Thistoolbox realizes recently developed polynomial methods for the analysis and optimal design of sampled-datasystems. Once released, the version under development will be compatible with the newest versionsof MATLAB and includes the possibility of working with both SISO and MIMO systems. The text describesthe last published version, an interim version currently running and the updates planned for a future stablerelease. Some usage examples obtained with the interim version are also presented.

Introduction

Sampled-data systems are systems in which a digital computer controls a continuous-time plant. This class of systems is widely used in almost any industry. Due to the periodic sampling, sampled-data systems are periodically time-varying systems, so that the standard methods for time invariant systems cannot be applied. Classical approaches to the design of controllers for such systems consider either a continuous time synthesis of a controller followed by its discretization, or a discretization of the plant followed by a controller synthesis in the discrete time domain. Both approaches are approximations. Modern approaches to the design of optimal controllers are based on so-called direct design methods, which take into account the continuous time behavior of the system without approximations.

The widely known *lifting* technique [1] allows the transformation of some hybrid optimization problems to equivalent problems for time-invariant discrete systems. However, the dimension of the systems becomes infinite.

This methodology was implemented in MATLAB as the Sampled-Data Control toolbox [2, 3]. The lifting technique, however, is not applicable to some important cases, e.g., to systems with arbitrary time delays.

An alternative for the direct design of sampled-data systems is the frequency domain approach based on the parametric transfer function concept [4, 5], which allows to apply polynomial methods for analysis and design of sampled-data control systems. This approach has some advantages over the lifting technique: it can be used even when the continuous plant has time delays and it also allows to obtain the structure and order of optimal controllers.

The DIRECTSD toolbox, in its initial version [6], was designed to implement polynomial methods in the case of SISO systems. Its development generated a MIMO version [7], which uses the theory presented in [5]. Its latest version [8] was a mature project: albeit it focused on SISO systems, it included options to use either polynomial methods or the lifting technique to solve the problem.

In addition, a comprehensive set of examples and demos has been integrated into the MATLAB help explorer.

The development of the DIRECTSD toolbox, however, halted with its version 3.0. After almost tenyears since its last revision, the code became obsolete, mainly because of the change in the object-oriented programming (OOP) model introduced in MATLAB release 2012.

During the last year, an effort has been made within the Institute of Automation at the University of Rostockto update the code and to have, once again, afunctional toolbox. The steps taken in this directionand the current development status of the DIRECTSDversion 4.0 are presented in this paper.

1 Structure of DIRECTSD 3.0

The DIRECTSD toolbox is currently in its version 3.0, which is available in [9]. The current code uses OOP to define a new class to create and manipulate polynomial objects. The class`poln`and its methods are one of the most important parts of the toolbox. To model continuous and discrete linear time invariant (LTI) systems, the standard objects from the Control Systems Toolbox of MATLAB, i.e. `tf`, `zpk`, or`ss`, are used. Some functions for these objects were overloaded in order to change their behavior.

The toolbox provides functions to perform analysis and design of sampled-data systems using both the frequency domain methods presented in [4] and the lifting technique [1]. Due to some unsolved numerical robustness problems encountered during the development of the MIMOtoolbox only SISOcontrollers are considered in this version.

As the OOP model of MATLAB changed with version 7.6 (Release 2008a), the syntax of class `poln` made it obsolete. This change also removes the support of overloaded functions, making the methods newly defined for the standard objects unusable.

2 Interim: DIRECTSD 3.5

The first steps taken towards the newer version were aimed to recover a basic level of functionality while the new developer got familiarized with the structure of the code and its theoretical background. The result of this work is an interim version, which is being used as a basis for further developments.

For this intermediate version, designated as DIRECTSD 3.5, the class `poln` was rewritten to make it compatible with the new OOP model. To recover the functionality of the overloaded functions, new classes were defined as subclasses of the standard MATLAB classes. For example, for the `tf` class a subclass called `sdtf` was defined. In this way, the class `sdtf` inherits all the properties and methods of a standard `tf` object replacing the old methods with those that were overloaded in previous versions. Albeit this solution required extensive editing of the codes, it was the most practical way to replace the overloaded functions.

Besides the recovery of the basic functionality some minor extensions were added to the toolbox. The most important is the possibility of using a first order hold with systems affected by a time delay.

Since this version is considered as an interim while the newer version is under development, we have no plans to make it publicly available. However, any interested reader may contact the authors to request a copy.

The following subsections present some examples of usage obtained with DIRECTSD 3.5.

2.1 H_2-Optimization of a system with delay using Zero and First Order Hold

This case study consists in solving the H_2 optimization problem for a sampled-data system affected by a time delay, using two different hold devices. The objective of the H_2optimization is to find the transfer function $C(\zeta)$ of the LTI digital controller described in the complex variable $\zeta = z^{-1} = e^{-sT}$, such that the minimum value of the mean variance of the output $\varepsilon(t)$ is obtained. This mean variance is defined as

$$\bar{d}_\varepsilon = \frac{1}{T}\int_0^T d_\varepsilon\,(t)dt \qquad (1)$$

where $d_\varepsilon(t) = d_\varepsilon(t+T)$is the instantaneous variance of the signal $\varepsilon(t)$ [4].

We would like to compare the results (the order of the controller and the optimal cost) using each hold device.

The system under study is shown in Figure 1. It consists of a continuous plant $P(s)$, a digital controller $C(\zeta)$, a hold device $H(s)$, and a pure delay representing time needed for computation, communic,ation or transport. The plant is taken as $P(s) = 1/s(s+1)$, and a sampling period $T =$1sec is used. A value of $\tau = 0.1$sec is set for the delay.

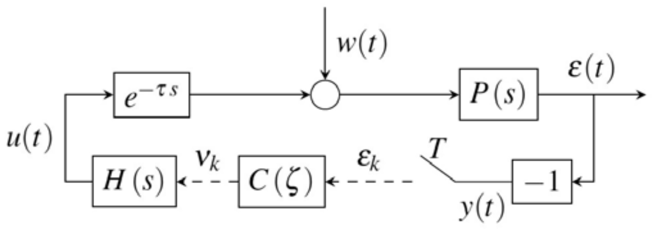

Figure 1. Structure of the system.

The two options considered for the hold device are a zero order hold (ZOH) for which

$$u(t) = \nu_k \text{ for } kT \le t < (k+1)T, \qquad (2)$$

and a first order hold (FOH) for which

$$u(t) = \nu_k + (t - kT)\frac{\nu_k - \nu_{k-1}}{T} \tag{3}$$

for $kT \leq t < (k+1)T$.

In order to use the DirecSD toolbox, the system must be converted into the standard form of MIMO sampled-data systems as shown in Figure 2.

Since in our case we have only a SISO system, all matrices and signals are scalar quantities.

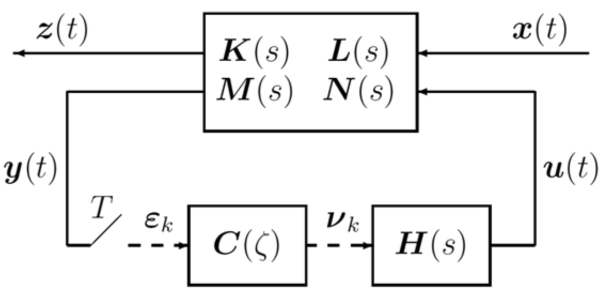

Figure 2. Standard sampled-data system.

Finally, we configure the scalar blocks

$$K(s) = P(s), \qquad L(s) = P(s)e^{-\tau s},$$
$$M(s) = -P(s), \qquad N(s) = -P(s)e^{-\tau s}.$$

The commands needed to run the example are as follows.

At first, let us set-up the system:

```
>>T = 1; %Sampling Period
>>H0 = tf(1,[1 0]); %Zero-order hold
>>H1(:,:,1,1)=ss(1);%First-order hold
>>H1(:,:,2,1)=ss(1);%First-order hold
>>P = zpk([],[0 -1],1); %Plant
>>tau = 0.1;
>>Fdelay = tf(1);
>>Fdelay.iodelay = tau;
>>Pdelay = P*Fdelay;
>>sys = [P Pdelay; -P -Pdelay];
```

The last line creates a rational matrix, which represents a system in the standard form. The functionsdh2 is used to synthesize the controllers and to find the optimal cost.

The commands issued are

```
>>[Kzohd, err0d] = sdh2(sys,T,[],H0);
>>[Kfohd, err1d] = sdh2(sys,T,[],H1);
```

For the ZOH the toolbox reports the controller

$$C_0(\zeta) = \frac{-193.39(\zeta - 3.964)}{(\zeta + 171.1)(\zeta + 1.143)} \tag{4}$$

with a minimum cost of $\bar{d}_{\varepsilon 0} = 0.6003$. On the other hand, using the FOH, we obtain the controller

$$C_1(\zeta) = \frac{200(\zeta - 3.819)}{(\zeta + 118.4)(\zeta - 2.705)(\zeta + 0.9415)} \tag{5}$$

with an optimal cost $\bar{d}_{\varepsilon 1} = 0.6538$.

These results show that in this particular case the introduction of a FOH does not show any advantage over the use of a ZOH. Besides the higher order of the reported controller, the performance index worsens when the FOH is used. While a first order interpolator would give smoother results, here due to the necessary causality oft he real hold element, the FOH becomes an extrapolator. This fact explains the above results. To have a more decisive comparison, Figure 3 presents the inter-sample variance$d_\varepsilon(t)$for both cases.

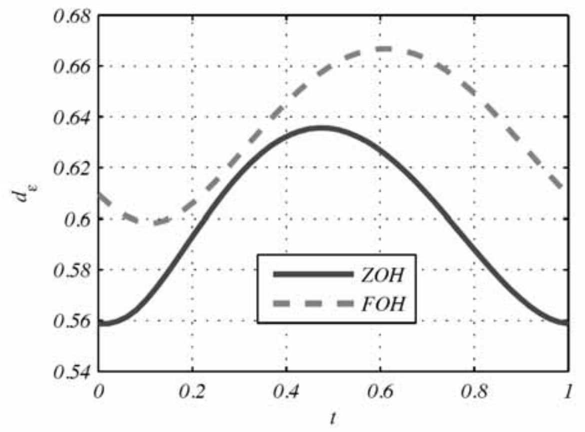

Figure 3. Comparison of the inter-sample variances for the cases with ZOH and FOH.

To obtain this plot, we used the function `sdh2norm` which finds either the mean variance or the variance at a certain point within the sampling interval.

For this particular case we issued the following commands to DIRECTSD 3.5

```
>>t = linspace(0,T,50);
>>errs0d = sdh2norm(sys,Kzohd,t,H0);
>>errs1d = sdh2norm(sys,Kfohd,t,H1);
```

in order to obtain the variance at 50 equidistant points within the sampling interval. The results were then plotted to obtain the picture observed in Figure 3.

2.2 H$_2$-Optimization of a generic system with two delays

This example considers the H_2optimization of a closed loop sampled-data system, as shown in Figure 4.

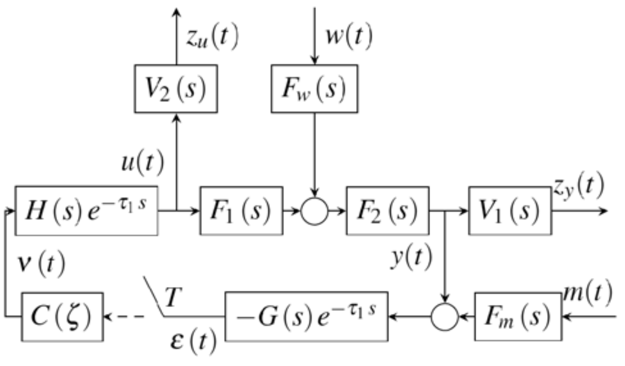

Figure 4. Structure of the system.

This is the most general case for a single loop system with a SISO controller.

For optimization purposes, the output of the system is taken as

$$\boldsymbol{z}(t) = \begin{bmatrix} z_y(t) \\ z_u(t) \end{bmatrix} \tag{6}$$

and the cost function to be minimized is

$$J = \bar{d}_z = \bar{d}_{zy} + \bar{d}_{zu} \tag{7}$$

In order to use the DIRECTSD toolbox, the system must be transformed into the standard form of Figure 2.

Realizing that the input of the system is

$$\boldsymbol{x}(t) = \begin{bmatrix} w(t) \\ m(t) \end{bmatrix}, \tag{8}$$

we obtain

$$\boldsymbol{K}(s) = \begin{bmatrix} V_1(s)F_2(s)F_w(s) & 0 \\ 0 & 0 \end{bmatrix} \tag{9}$$

$$\boldsymbol{L}(s) = \begin{bmatrix} V_1(s)F_2(s)F_1(s)(t)H(s)e^{-\tau_1 s} \\ V_2(s)H(s)e^{-\tau_1 s} \end{bmatrix} \tag{10}$$

$$\boldsymbol{M}(s) \\ = -[G(s)F_2(s)F_w(s)e^{-\tau_2 s} \quad G(s)F_m(s)e^{-\tau_2 s}] \tag{11}$$

$$\boldsymbol{N}(s) = -G(s)F_2(s)F_1(s)H(s)e^{-(\tau_1+\tau_2)s}. \tag{12}$$

Consider a simple case in which the disturbance and measurement noise are both withe noise. This means that the forming filters $F_w(s)$ and $F_m(s)$ are both equal to 1. The weights of the output signals are taken as $V_1 = 1$ and $V_2 = 2$. The other components take the following values

$$F_1(s) = \frac{1}{s+1}, \qquad G(s) = 1,$$

$$F_2(s) = \frac{2}{s+2}, \qquad H(s) = \frac{0.25}{s+0.25}.$$

The two time delays are $\tau_1 = 0.05$ sec and $\tau_2 = 0.1$ sec, whereas the sampling period is taken as $T = 0.5$ sec.

This example is run by entering the following commands

```
>>F1 = tf(1,[1 1]);
>>F2 = tf(2,[1 2]);
>>Fm = 1;
>>Fw = 1;
>>tau1 = 0.05;
>>tau2 = 0.1;
>>G = -tf(1,'iodelay',tau2);
>>H = tf(0.25,[1 0.25],'iodelay',tau1);
>>V1 = 1;
>>V2 = 2;
>>T = 0.5;
>>H0 = tf(1,[1 0]);
>>K = [V1*F2*Fw 0; 0 0];
>>L = [V1*F2*F1*H; V2*H];
>>M = [G*F2*Fw G*Fm];
>>N = G*F2*F1*H;
>>sys = [K L; M N];
>>[C, err] = sdh2(sys,1,[],H0)
```

Considering these values, the optimal controller is found as

$$C_{\text{opt}}(\zeta) = \frac{0.88946(\zeta - 1.284)(\zeta - 2.718)}{(\zeta + 3.721)(\zeta - 3.134)(\zeta - 13.81)} \quad (13)$$

with an optimal cost $J_{\text{opt}} = 0.99995$.

3 Future Version: DIRECTSD 4.0

A complete refurbishing of the toolbox is currently undergoing. We aim to recover the full capabilities of the last public release while we add some bells and whistles. Some of the improvements being added or planned to be added are

A new class for rational matrices: Up to this point, different classes have been used to represent polynomial matrices (class `poln`) and rational matrices (linear systems in `zpk, tf` or `ss` form).
This created the need to some extra code to perform transformations when two objects of different classes should operate together.
The new classes will be subclasses of the standard MATLABclasses, so that they can inherit their properties and methods, and will also have all the particular methods to operate with polynomial matrices, i.e., rational matrices with denominator equal to one.

Packages for different methods: In its version 3.0 DIRECTSD includes options to work with the lifting technique or with polynomial methods.
These functions are being reorganized in the form of packages, according to the new OOP model of MATLAB.

Support for MIMO systems: The DIRECTSDM toolbox [7] was abandoned due to numerical instabilitiesof the MIMOversion of the algorithms for polynomialdesign of sampled-data systems.
An effort is being made in to develop numerically reliablealgorithms for this case. We are attemptingto include said algorithms in the newer version.

Updated examples, demos and help files: The htmlhelp files will be revised and updated once thefinal code is in place.

Besides this user-facing improvements, the codes ofall the functions are being revised. The coding style is being standardized and the internal documentationis being improved. These steps aim to simplify the maintenance of the toolbox.

For future versions, we are already considering the possibility of developing a graphic user interface, similar to the SISO tool included in the standard Control Systems Toolbox.

4 Concluding Remarks

This paper described the current development status of version 4.0 for the DIRECTSD, a MATLAB toolbox for the analysis and design of sampled-data systems. While an interim version, which recovered the basic functionality of the obsolete code, is in place, the work to obtain a completely revised version is on going.

Acknowledgement

The authors are thankful to Dr. K. Y. Polyakov for leaving the last version of the toolbox DIRECTSD 3.0, and for helpful discussions and explanations. The authors aknowledge the financial support of the German Science Foundation DFG.

References

[1] Chen T, Francis B. *Optimal Sampled-Data Control Systems*. New York: Springer-Verlag,1995.

[2] Fujioka H, Yamamoto Y, Hara S. Sampled-data control toolbox: A software package via object oriented programming. *Proceedings of the IEEE International Symposium on ComputerAided Control Systems Design;* 1999 Aug; Hawai, HI, USA; 1999, P. 404–409.

[3] Fujioka H, Hara S, Yamamoto Y. Sampled-data control toolbox: object oriented of for sampled-data feedback control systems. *Proceedings of the IEEE Conference on ComputerAided Control Systems Design*; 2004 Sept; Taipei, Taiwan; 2004, P. 19–24.

[4] Rosenwasser E Lampe B. *Computer Controlled Systems: Analysis and Design using Proccess orientated Models, ser.* Communications and Control Engineering. London: Springer-Verlag, 2000.

[5] Rosenwasser E, Lampe B. *Multivariable Computer Controlled Systems: A Transfer Function Approach, ser.* Communications and Control Engineering. London: Springer, 2006.

[6] Polyakov K, Rosenwasser E, Lampe B. DirectSD- a toolbox for direct design of sampled-data systems. *Proceedings of the IEEE International Symposium on Computer Aided Control Systems Design*; 1999 Aug; Hawai, HI, USA; 1999, P. 357–362.

[7] Polyakov K, Lampe B, Rosenwasser E. DirectSDM- a toolbox for polynomial design of multivariable sampled-data systems. *Proceedings of the IEEE Conference on Computer Aided Control Systems Design*; 2004 Sept; Taipei, Taiwan; 2004, P. 95–100.

[8] Polyakov K, Rosenwasser E, Lampe B. DirectSD3.0 toolbox for Matlab: further progress in polynomial design of sampled-data systems. *Proceedings of the IEEE Conference on ComputerAided Control Systems Design*, 2006 Oct; Munich, Germany; 2006, P. 1946-1951.

[9] Polyakov K. DirectSD: Package for the analysis and synthesis of digital control systems [Internet]. [cited 2015 May]. Available from: http://kpolyakov.spb.ru/science/dsd.htm