Descriptor Systems Toolbox : A Mathematica-Based Package for Descriptor Systems

A.I. Vardulakis[1], N.P. Karampetakis[1], E. Antoniou[2] and S. Vologiannidis[1]

***Abstract*— It is well known that descriptor systems or generalized state-space systems are the natural framework for the study of physical, electrical mechanical, interconnected, economical and social systems. Although a number of software packages has been developed for state-space systems, which is can be seen as special case of descriptor systems, there are only few for descriptor systems. In this paper we present an efficient and reliable Mathematica software package for descriptor systems that can be used both for teaching and research in the field of control theory. The proposed package is fully compatible with the Control Systems Professional Suite of Wolfram Research Inc. and is intended to be an add-on for the Control Systems Professional Suite in the near future. This package has been created in collaboration with Wolfram Research Inc. and Zenon S.A. under a grant from the General Secreteriat for Research and Technology of Greece.**

I. Introduction

We consider *descriptor* (or otherwise called *extended state space* or *singular* or *generalized state space*) systems representations, described by a set of linear and/or algebraic equations of the form :

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{1}$$

where $E, A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$. Descriptor representations have important advantages over state space systems such as

- *State-space* representations are a particular case of descriptor representations, i.e. $E = I_n$.
- Even in case where the matrix E is invertible and the above representation can be transformed to the following state-space representation

  $$\begin{aligned} \dot{x}(t) &= E^{-1}Ax(t) + E^{-1}Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

  it is often desirable to avoid the inversion of E since it is sensitive to numerical errors, especially when the matrix E has large dimension and/or the condition number of E is high.
- Descriptor representations, in contrast to state-space systems can accommodate algebraic constraints.
- The interconnection of state-space systems leads very often to descriptor representations.
- Descriptor representations can succesfully model impulsive behavior which occurs in the real world and it is usually neglected by state space models (e.g. switching phenomena).
- In contrast to state-space techniques that introduce ”artificial” variables, descriptor representations use as variables only the natural quantities describing the system.

Descriptor representations are the natural framework in engineering systems (power systems, electrical networks, aerospace engineering and chemical processes), social economic systems, network analysis, biological systems, time-series analysis, system modeling and so on (see [5], [9], [10] and the references within). Non-regular descriptor equations play also a very important role in the study of the discrete Riccatti equation, where the associated Extended Hamiltonian Pencil (EHP) is involved [1], [2].

During the recent years a number of software packages, as like as, *Scilab* [14], *Slicot* [16], *Control Systems Toolbox* 8.0.1 of *Matlab* [11], has been developed intended for systems and control theory computations. Most of them incorporate a collection of functions for state-space and transfer function objects, while only a few of them, like the *Polynomial Toolbox* (*Polyx*) for *Matlab* [13] and *Slicot* [16], can handle descriptor objects and their functionality is mainly based on numerical methods. Even in these packages the functionality is very limited compared to existing implementations for state-space systems.

Wolfram Research has recently released the *Control System Professional* (CSP) *Suite* package as an add-on package for *Mathematica* [4]. CSP Suite is a very powerful tool for the analysis, design and simulation of linear MIMO (multi-input, multi-output) systems as well as SISO (single-input, single-output) systems in both time and frequency domains. It includes both *symbolic* and and state-of-the-art *numerical* algorithms for the study of state-space systems. Additionally, it provides new tools for modeling, analysis and design of linear control systems described by polynomial matrix equations or by matrices with rational polynomial elements. Although, descriptor representations can easily be cosidered as first order polynomial matrix equations and studied using the existing functionality of the CSP Suite, there is no descriptor object (like the one for state-space representations) and of course no functionality for analysis and synthesis problems has been implemented.

In this paper we present a new Mathematica package, called Descriptor Systems Package (DSP). DSP actually extends the functionality of the CSP Suite in order to handle

[1]Aristotle University of Thessaloniki, Department of Mathematics, 54006 Greece {avardula, karampet, svol}@math.auth.gr

[2]Technological Educational Institute of Thessaloniki, General Department of Sciences, 54101 Thessaloniki, Greece, eantonio@gen.teithe.gr

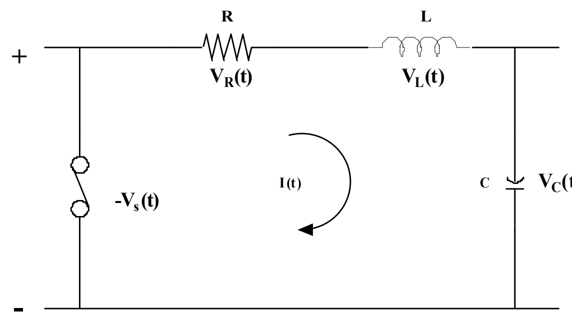
descriptor systems. It can handle descriptor representations both numerically and symbolically since it uses the existing functions of CSP Suite.

II. The Descriptor Systems Package

In this section we describe in detail the main features of DSP. A great deal of effort has been put into developing routines that integrate seamlessly with the existing programming model of the CSP Suite, while the algorithms implemented have been based on state-of-the-art numerical and symbolic methodologies found in the literature.

A. Linear Model Descriptions and Manipulations

Following the CSP methodology we have introduced a new object called `DescriptorStateSpace`. For example consider the circuit shown bellow



which can described by the following algebraic and differential equations

$$\begin{aligned} V_L(t) &= L\frac{dI(t)}{dt} \\ \frac{dV_C(t)}{dt} &= \frac{1}{C}I(t) \\ V_R(t) &= RI(t) \\ V_L(t)+V_C(t)+V_R(t) &= V_S(t) \end{aligned}$$

or in descriptor form

$$\underbrace{\begin{pmatrix} L & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{E}\begin{pmatrix} \dot{I}(t) \\ \dot{V}_L(t) \\ \dot{V}_C(t) \\ \dot{V}_R(t) \end{pmatrix} =$$

$$= \underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{C} & 0 & 0 & 0 \\ -R & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}}_{A}\begin{pmatrix} I(t) \\ V_L(t) \\ V_C(t) \\ V_R(t) \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}}_{B} V_S(t)$$

$$y(t) = V_C(t) = \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}}_{C}\begin{pmatrix} I(t) \\ V_L(t) \\ V_C(t) \\ V_R(t) \end{pmatrix}$$

This model can easily be defined by the new object as follows

First we call the package :

```
In[1]:= << DescriptorControlSystems`
```

Then we define the matrices E, A, B, C and denote the descriptor state space system by dss

```
In[2]:= e = {{L, 0, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};
        a = {{0, 1, 0, 0}, {1/C, 0, 0, 0}, {-R, 0, 0, 1}, {0, 1, 1, 1}};
        b = {{0}, {0}, {0}, {-1}};
        c = {{0, 0, 1, 0}};
        dss := DescriptorStateSpace[e, a, b, c]
```

The traditional form of dss is given by :

```
In[7]:= TraditionalForm[dss]
```

Out[7]//TraditionalForm=

$$\left(\begin{array}{cccc|cccc|c} L & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{1}{C} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -R & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 \\ \hline & & & & 0 & 0 & 1 & 0 & 0 \end{array}\right)^{\mathcal{D}}.$$

or in equation form

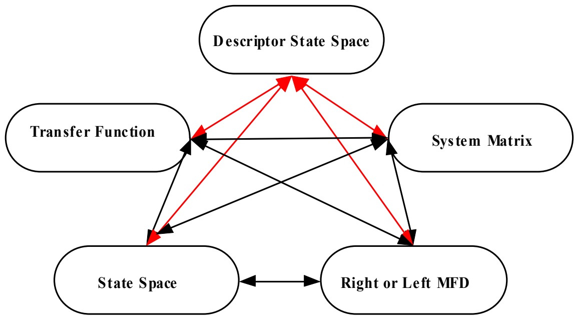
```
In[8]:= EquationForm[dss,
          StateVariables → {"I", VL, VC, VR},
          InputVariables → {VS}]
```

Out[8]//EquationForm=

$$\begin{pmatrix} L & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \dot{I} \\ \dot{V}_L \\ \dot{V}_C \\ \dot{V}_R \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{C} & 0 & 0 & 0 \\ -R & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} I \\ V_L \\ V_C \\ V_R \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}(V_S)$$

$$y = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} I \\ V_L \\ V_C \\ V_R \end{pmatrix}$$

where now the coefficient matrix on the left hand side of the first equation is singular. Descriptor objects can now be transformed to and from other CSP objects by using the existing algorithms of the CSP Suite (both numerical and symbolic) and algorithms described in [5], [17] and [7].



For example we can find the transfer function (matrix) of the above model :

```
In[9]:= tf = TransferFunction[dss]
```

$$\text{Out[9]=} \left(\frac{1}{C\,s(R+L\,s)+1}\right)^{\mathcal{T}}$$

and then find a minimal realization of this

```
In[10]:= ss = MinimalRealization[tf]

         PoleZeroCancel::nzpm : No common pole-zero pairs found.
```

$$\text{Out[10]=}\left(\begin{array}{cc|c} 0 & 1 & 0 \\ -\frac{1}{C\,L} & -\frac{R}{L} & 1 \\ \hline \frac{1}{C\,L} & 0 & 0 \end{array}\right)^{\mathcal{S}}.$$

We can convert this state-space object to a descriptor object:

```
In[11]:= dsm = DescriptorStateSpace[ss]
```

$$\text{Out[11]=}\left(\begin{array}{cc|cc|c} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & -\frac{1}{C\,L} & -\frac{R}{L} & 1 \\ \hline & & \frac{1}{C\,L} & 0 & 0 \end{array}\right)^{\mathcal{D}}.$$

Apart from the transformation between different CSP objects, there are algorithms to transform a descriptor representation to canonical forms, like the Weierstrass canonical form or the Kalman controllable/observable form (if it exists). If for example we have that $R = 1, L = C = 1/10$ in the previous example, then we have that the Weierstrass form of the descriptor system dss is given by

```
In[12]:= WeierstrassCanonicalForm[
           dss /. {R → 1, L → 1/10, C → 1/10}] // N
```

$$\text{Out[12]=}\left(\begin{array}{cccc|cccc|c} 1. & 0. & 0. & 0. & -5. & -8.66025 & 0. & 0. & -0.143438 \\ 0. & 1. & 0. & 0. & 8.66025 & -5. & 0. & 0. & 0.250104 \\ 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & -1. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\ \hline & & & & -34.7417 & -19.9248 & 0. & 0. & 0 \end{array}\right)^{\mathcal{D}}.$$

Additionaly, we can discretize a continuous time descriptor system by using known discretization methods [15] like the forward or the backward rectangular rule.

B. System analysis

Based on [10], we have developed new functions for the exact computation of the state-space and output response of descriptor systems. However, when the symbolic solution is either impossible or just too time consuming then a simulation of the state and output responses can be considered.

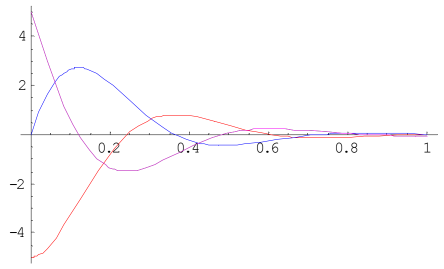
For example, we can compute the state-space impulse response of the system dss described above

```
In[15]:= xd =
          Chop[StateResponse[dss /. {R → 1, C → 0.1, L → 0.1},
            DiracDelta[t], t]]
```

$$\text{Out[15]=}\ \{e^{-5.\,t}(5.\cos(8.66025\,t) - 2.88675\sin(8.66025\,t)),$$
$$-5.\,e^{-5.\,t}\cos(8.66025\,t) + 1.\,\delta(t) - 2.88675\,e^{-5.\,t}\sin(8.66025\,t),$$
$$5.7735\,e^{-5.\,t}\sin(8.66025\,t),$$
$$5.\,e^{-5.\,t}\cos(8.66025\,t) - 2.88675\,e^{-5.\,t}\sin(8.66025\,t)\}$$

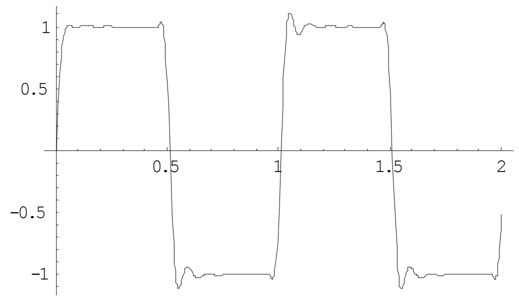
and in the sequel plot the entries of the state vector

```
In[16]:= Plot[Evaluate[xd], {t, 0, 1},
          PlotStyle → {RGBColor[0, 1, 0], RGBColor[1, 0, 0],
            RGBColor[0, 0, 1], RGBColor[1, 0, 1]},
          PlotRange -> All]
```



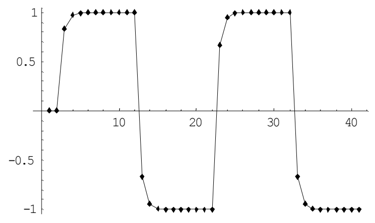
We can also simulate the output response of the system dss for a square wave input as follows

```
In[40]:= square[t_] := Sign[Sin[2 π t]]
         SimulationPlot[
          dss /. {R → 10, C → 0.001, L → 0.002}, square[t],
          {t, 0, 2}]
```



The previously simulated response can be compared to the output response of the discretized system

```
In[22]:= yt = OutputResponse[
            ddsb /. {R → 10, C → 0.001, L → 0.002,
              T → 0.05}, square[t], {t, 2}];
          MultipleListPlot[yt, PlotJoined → True]
```



Functions for testing system properties of a descriptor representation like controllability and observability have been developed following the theory in [12], which uses the standard form of the pencil avoiding more complex approaches like the ones in [3]. Thus in case where the matrix $s_0E - A$ is invertible for some $s_0 \in \mathbb{R}$, we premultiple the matrices E, A, B by $(s_0E - A)^{-1}$ and we get three new matrices

$$\tilde{E} = (s_0E - A)^{-1}\,E, \tilde{A} = (s_0E - A)^{-1}\,A$$
$$\tilde{B} = (s_0E - A)^{-1}\,B$$

where now the new matrices E, A are in standard form i.e. $s_0 \times \tilde{E} + (-1) \times \tilde{A} = I_n$ (therefore $\tilde{E}, \tilde{A}$ commute). Then according to [12], we construct the controllability matrix

$$R_S = \begin{pmatrix} \tilde{A}^{n-1}B & \tilde{E}\tilde{A}^{n-2}B & \cdots & \tilde{E}^{n-1}B \end{pmatrix}$$

and the descriptor system is strongly reachable iff the controllability matrix R_S has full row rank. Note that in case where the matrix $E = I_n$ (state-space), then we have the known controllability matrix for state-space systems. Note that the controllability matrix R_S has full rank iff the matrix $\begin{pmatrix} sE - tA & B \end{pmatrix}$ has full rank for all $(s,t) \neq (0,0)$ [12] i.e. for $s = 1, t = 0$ we have that $rank_{\mathbb{R}} \begin{pmatrix} E & B \end{pmatrix} = n$ and for $s \in \mathbb{C}, t = 1$ we have that $rank_{\mathbb{R}} \begin{pmatrix} sE - A & B \end{pmatrix} = n$.

For example the controllability matrix of the descriptor representation dss given above is

```
In[24]:= ControllabilityMatrix[
           dss /. {R → 1, C → 0.1, L → 0.1}]
```

Out[24]=
$$\begin{pmatrix} -0.00030295 & 0.00919523 & -0.0616572 & -0.30295 \\ 0.000919523 & -0.00616572 & -0.030295 & -0.0804774 \\ -0.000616572 & -0.0030295 & 0.0919523 & -0.616572 \\ -0.00030295 & 0.00919523 & -0.0616572 & -0.30295 \end{pmatrix}$$

```
In[25]:= Controllable[dss]
```

Out[25]= False

In a similar way we work for observability, since these two notions are dual to each other.

```
In[26]:= ObservabilityMatrix[dss /. {R → 1, C → 0.1, L → 0.1}]
```

Out[26]=
$$\begin{pmatrix} -0.000616572 & -0.0000919523 & -0.000616572 & 0.000616572 \\ -0.0030295 & 0.000616572 & -0.0030295 & 0.0030295 \\ 0.0919523 & 0.0030295 & 0.0919523 & -0.0919523 \\ -0.616572 & -0.0919523 & -0.616572 & 0.616572 \end{pmatrix}$$

```
In[27]:= Observable[dss]
```

Out[27]= False

DSP provides the tools needed to construct a composite system based on a given system topology and descriptions of the blocks. We have extended the functions of the CSP Suite in order to perform elementary interconnections between descriptor representations. Thus in case for example we have (before the DSP) a system with proper transfer function and its equivalent state space model

```
In[28]:= tf = TransferFunction[s, s/(s + 1)]
```

Out[28]= $\left(\frac{s}{s+1} \right)^{\mathcal{T}}$

```
In[29]:= ss = StateSpace[tf]
```

Out[29]= $\left(\begin{array}{c|c} -1 & 1 \\ \hline -1 & 1 \end{array} \right)^{\mathcal{S}}_{\bullet}$

by making the following output feedback interconnection

```
In[30]:= FeedbackConnect[ss, Positive]

Inverse::sing : Matrix (0) is singular. More…
```

Out[30]= FeedbackConnect$\left[\left(\begin{array}{c|c} -1 & 1 \\ \hline -1 & 1 \end{array} \right)^{\mathcal{S}}_{\bullet}, \text{Positive} \right]$

results into a singular system that cannot be modeled as a state space system. However doing the same interconnection using the equivalent descriptor state space system of the original transfer function

```
In[31]:= FeedbackConnect[
           DescriptorStateSpace[tf], Positive]
```

Out[31]= $\left(\begin{array}{cc|cc|c} 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & -2 & 1 \\ \hline & & 1 & -1 & 0 \end{array} \right)^{\mathcal{D}}_{\bullet}$

```
In[32]:= TransferFunction[%]
```

Out[32]= $(s)^{\mathcal{T}}$

gives a descriptor system with singular E and an (equivalent) non-proper transfer function.

C. Synthesis and design techniques

The first problem that we solved was the pole assignment problem i.e to find the state feedback $u(t) = -Kx(t) + v(t)$ that forces the poles of the descriptor system (1), that is the $k = rank_{\mathbb{R}}(E)$ finite eigenvalues of matrix $sE - A$, to assume the new positions $\lambda_1, \lambda_2, ..., \lambda_k$. We have solved this problem by using two methods : a) by using the generalized Ackermann formula [6] for single-input-single-output descriptor systems, and b) by solving an equivalent state space problem using techniques described in ([5], pp.83) and thus using the existing CSP robust Kautsky-Nichols-Van Dooren (KNVD) algorithm [8]. Thus, the functions that are used in CSP Suite for pole assignment, were generalized to descriptor state space systems as well.

For example consider the following siso-descriptor system

```
In[33]:= dsb = DescriptorStateSpace[{{0, 0, 0}, {0, 1, 1}, {1, 0, 0}},
             DiagonalMatrix[{1, 2, 1}], {{1}, {0}, {0}}, {{0, 1, 0}}, {{0}}];
         EquationForm[dsb]
```

Out[34]//EquationForm=
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \dot{x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u$$
$$y = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \end{pmatrix} u$$

where the Smith form of the polynomial matrix $sE - A$ is given by

```
In[35]:= SmithForm[dsb[[1]] * s - dsb[[2]], s]
```

Out[35]=
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s-2 \end{pmatrix}$$

and therefore the system has one finite pole at $s = 2$ and two poles at infinity. The system dsb is controllable

```
In[36]:= Controllable[dsb]
```

Out[36]= True

and therefore using a generalized version of the Ackermann's formula given in [6] we can change $rank_{\mathbb{R}}E = 2$ finite poles of the system. In what follows, we find the state feedback $u(t) = -Kx(t) + v(t)$ that change the poles of the above system $\{2, \infty, \infty\}$ to $\{p_1, p_2, \infty\}$

```
In[37]:= k = StateFeedbackGains[dsb, {p1, p2},
             Method → Ackermann] // Simplify
```

Out[37]= $\left(\frac{\text{p1 p2}}{2} + 1 \quad \frac{1}{4}(\text{p1} - 2)(\text{p2} - 2) \quad \frac{1}{4}(\text{p1}(\text{p2} - 2) - 2\,\text{p2}) \right)$

Note that the closed loop system

$$E\dot{x}(t) = (A - BK)\,x(t) + Bv(t)$$

has the desired poles

```
In[38]:= dscl = StateFeedbackConnect[dsb, k] // Simplify;
         EquationForm[dscl]
```

Out[39]//EquationForm=
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \dot{x} = \begin{pmatrix} -\frac{\text{p1 p2}}{2} & -\frac{1}{4}(\text{p1} - 2)(\text{p2} - 2) & \frac{1}{4}(2\,\text{p2} - \text{p1}(\text{p2} - 2)) \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u$$
$$y = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \end{pmatrix} u$$

since the Smith form of $sE - A$ is given by

```
In[40]:= SmithForm[dscl[[1]] * s - dscl[[2]], s]
```

$$\text{Out[40]=}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & (\text{p}1-s)(\text{p}2-s) \end{pmatrix}$$

The same functions are also working for multi-input-multi-output descriptor systems by using the KNVD method. The dual problem of the pole assignment problem is the state reconstruction problem i.e that is to find an estimator gain matrix L such that the descriptor system

$$E\dot{\hat{x}}(t) = (A - LC)\,\hat{x}(t) + Bu(t) + Ly(t)$$

give us an approximation state vector $\hat{x}(t)$ of $x(t)$ or otherwise the spectrum $\sigma(E, A - LC)$ is within the unit circle for discrete-time systems or to the left half plane for continuous-time systems. For example the above system dsb is observable

```
In[41]:= Observable[dsb]

Out[41]= True
```

and therefore there exists an estimator gain L

```
In[42]:= l = EstimatorGains[dsb, {-1, -2}]
```

$$\text{Out[42]=}\begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix}$$

such that the finite zeros of the matrix pencil $sE - (A - LC)$ be $\{-1, -2\}$

```
In[43]:= SmithForm[dsb[[1]] * s - (dsb[[2]] - l.dsb[[4]]), s]
```

$$\text{Out[43]=}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & (s+1)(s+2) \end{pmatrix}$$

Finally, we have created functions based on the algorithms presented in [1], [2] that solve the linear-quadratic optimal regulator (LQR) problem for continuous and discrete time systems. That is, we have to compute a control $u(t)$ that minimize the cost functional

$$J(x,u) := \frac{1}{2}\int_0^\infty \begin{pmatrix} x^T & u^T \end{pmatrix} \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} dt$$

where the matrix R is positive definite and the matrix $\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix}$ is positive semi-definite. The optimal solution is a state-feedback of the form $u(t) = -Kx(t)$ and K is known as the optimal stabilizing control gain. Suppose for example that we have the descriptor system dsb defined above and we are interested to minimize the above cost functional with matrices

$$Q = I_3, R = 1, S = 0_{3,1}$$

First we define the matrices

```
In[44]:= q = IdentityMatrix[3];
         r = IdentityMatrix[1];
```

and then find the optimal stabilizing control gain

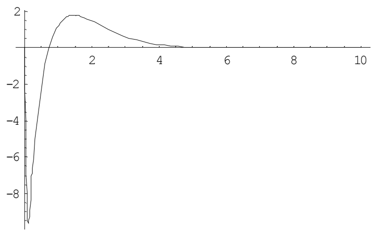
```
In[46]:= k = LQRegulatorGains[dsb // N, q // N, r // N]

Out[46]= ( 0.918643   -0.229661   -0.148304 )
```

Then, the closed loop system and the plot of the state are given below

```
In[47]:= dscl1 = StateFeedbackConnect[dsb, k]
         SimulationPlot[dscl1, UnitStep[t], {t, 0, 10}]
```

$$\text{Out[47]=}\left(\begin{array}{ccc|ccc|c} 0 & 0 & 0 & 0.081357 & 0.229661 & 0.148304 & 1. \\ 0 & 1 & 1 & 0. & 2. & 0. & 0. \\ 1 & 0 & 0 & 0. & 0. & 1. & 0. \\ \hline & & & 0. & 1. & 0. & 0. \end{array}\right)^{\mathcal{D}}.$$



Similar functions have also been developed for the solution of the LQR-problem for the discrete-time case.

III. Conclusions

A new Mathematica-based package, called Descriptor System Package, has been developed in order to extend the capabilities of the CSP Suite of Mathematica. This new toolbox provides new tools for modelling, analysis and design of descriptor systems. It follows the same programming model with the CSP Suite and uses both symbolic and numerical algorithms. DSP is a very useful tool for teaching and research in the field of control theory.

References

[1] D. Bender, A. Laub, ”The Linear-Quadratic Optimal Regulator for Descriptor Systems”, IEEE Trans. Autom., Vol AC-32, No 8, Aug 1987, pp 672-688.

[2] Bender D.J., Laub A.J., ”The Linear-quadratic Optimal regulator for Descriptor Systems: Discrete time case”, Automatica, Vol 23, 1987, No 1, pp 71-85.

[3] D. J. Cobb, “Controllability, observability and duality in singular systems,” IEEE Trans. Automat. Contr., vol. AC-29, no. 12, pp.1076-1082, 1984

[4] Control System Professional Suite, Wolfram Research Inc., http://www.wolfram.com/products/applications/csps/

[5] L. Dai, ”Singular Control Systems, Lecture Notes in Control and Information Sciences”, Springer-Verlag, 1989.

[6] Lee-Chuang Hsu, Fan-Ren Chang, The generalized Ackermann’s formula for singular systems, Systems & Control Letter, 1996, Volume 27 , Issue 2, Pages: 117 - 123.

[7] Karampetakis N.P. and Vardulakis A.I.G., 1993, Generalized state-space system matrix equivalents of a Rosenbrock system matrix., IMA Journal of Mathematical Control and its Information, NO.10, pp.323-344.

[8] Kautsky, J., N. K. Nichols, and P. Van Dooren. Robust pole assignment in linear state feedback. International Journal of Control, 41, no. 5, 1985, pp. 1129–1155.

[9] Kuijper M., ”First order representations of Linear systems”, Systems & Control: Foundations & Applications, Birkhauser, Boston, 1994.

[10] Lewis F. L., ”A Survey of Linear Singular Systems”, Circuits Systems and Signal Processing, Vol 5, No 1, 1986, pp.3 - 36.

[11] Control System Toolbox^{TM} 8.0.1, The MathWorks, http://www.mathworks.com/products/control/

[12] Nikoukhah R, Willsky A., Bernard C. Levy, ”Boundary-value descriptor systems: well-posedness, reachability and observability”, Int. J. Control, 1987, Vol 46, 1715 - 1737.

[13] The Polynomial Toolbox, http://www.polyx.com/

[14] Scilab, INRIA, http://www.scilab.org/

[15] Sincovec R., Erisman, A., Yip E., Epton M., ”Analysis of descriptor systems using numerical algorithms”, Automatic Control, IEEE Trans. Automatic Control, Vol. 26, Issue 1, Feb 1981, pp. 139 - 147

[16] Delebecque F. A Slicot based control library for Scilab, IEEE International Symposium on Computer-Aided Control System Design, 2000 (CACSD 2000), Anchorage, AK, USA, pp. 147-149.

[17] Vardulakis A.I.G., ’Linear Multivariable Control - Algebraic Analysis and Synthesis Methods’, Willey, 1991, New York.