Linear Model Predictive Control Toolbox

User's Guide

John Bagterp Jørgensen

February 2004

2-control ApS
www.2-control.com

ISBN XX-XXXXX-XX-X
Printed by Book Partner, Nørhaven Digital, Copenhagen, Denmark

Contents

1

Realization of Deterministic Transfer Functions

Table 1.1. Realization Algorithms in the Linear Model Predictive Control Toolbox

css2dsszoh	state space matrices of zoh discretization
sisoctf2css	cont. transfer function model to cont. state space model (siso)
sisocss2dss	cont. state space model to disc. state space model (siso)
sisodss2dimpulse	disc. state space model to disc. impulse response (siso)
sisodss2dstep	disc. state space model to disc. step response (siso)
sisoctf2dss	cont. transfer function model to disc. state space model (siso)
sisoctf2dimpulse	cont. transfer function model to impulse response (siso)
sisoctf2dstep	cont. transfer function model to disc. step response (siso)
mimoctf2dimpulse	cont. transfer function model to disc. impulse response (mimo)
mimoctf2dstep	cont. transfer function model to disc. step response (mimo)
mimodimpulse2dss	disc. impulse response to disc. state space (mimo)
mimoctf2dss	cont. transfer function to disc. state space (mimo)
mimodss2dimpulse	disc. time state space to disc. time impulse response (mimo)
mimodss2dstep	disc. time state space to disc. time step response (mimo)

Table 1.2. Examples on the Use of Realization Algorithms

sisorealization	disc. state space, impulse, and step response realization of siso transfer functions
mimorealization	state space realization of a mimo transfer function, computation of step and impulse responses
ShellStandardControlProblem	Realization of the Shell standard control problem

1.1 Introduction

The main problem problem addressed in this chapter is the realization of transfer functions for multiple-input-multiple-output (MIMO) systems as discrete-time state space systems. A realization of a transfer function is the state space matrices in a state-space representation, the impulse response matrices in an impulse response model, or the step response matrices in a step response model such that these systems have the same input-output behavior as the system specified by the transfer function.

The realizations for MIMO systems considered in this chapter are

1. Computation of the discrete-time impulse response coefficients (Markov parameters) for a continuous-time transfer function with time delay.
2. Computation of the discrete-time step response coefficients for a continuous-time transfer function with time delay.
3. Computation of the state space matrices (A, B, C, D) in a discrete-time state space model for a continuous-time transfer function with time delay.

Each of these representations may be used in the design of model predictive controllers based on linear models.

The discrete-time MIMO state space model of a continuous-time transfer function model is realized by first constructing the discrete-time impulse response and then doing a singular value decomposition of a Hankel matrix of the impulse response matrices. The advantage of this approach is that the realized system is guaranteed to be both observable and controllable. Furthermore, the impulse responses may be constructed by computing the impulse response for each individual input-output pair. This procedure is particularly efficient when the system has long time delays.

The realization of a MIMO transfer function

$$Y(s) = G(s)U(s) \tag{1.1}$$

in which

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) & \dots & g_{1m}(s) \\ g_{21}(s) & g_{22}(s) & \dots & g_{2m}(s) \\ \vdots & \vdots & & \vdots \\ g_{p1}(s) & g_{p2}(s) & \dots & g_{pm}(s) \end{bmatrix} \tag{1.2}$$

as a discrete-time state space system with sampling time T_s and zero-order-hold of the inputs may be written as

$$x_{k+1} = Ax_k + Bu_k \tag{1.3a}$$
$$y_k = Cx_k + Du_k \tag{1.3b}$$

and concerns the computation of (A, B, C, D). `mimoctf2dss` is constructed to do this computation. Each individual transfer function, $g_{ij}(s)$ has the form

$$g_{ij}(s) = h_{ij}(s)e^{-\lambda s} = \frac{b(s)}{a(s)}\, e^{-\lambda s} \tag{1.4}$$

Transfer functions are convenient for expressing dynamics of systems for which the step responses are approximately known, e.g. obtained from step response experiments. `sisoctf2dstep` may be used in the calibration of the individual transfer functions to the experimentally obtained step responses.

Application of the realization algorithms could be as this

1. For all input-output combinations, construct individual transfer functions calibrated against step response experiments using `sisoctf2dstep`. Each time a parameter is adjusted `sisoctf2dstep` is called with new SISO transfer function and the result is plotted against the experimental step response data.

2. When all SISO input-output continuous-time transfer function have been constructed they are collected as one MIMO continuous-time transfer function. This MIMO continuous-time transfer function is converted to a discrete-time state space model using `mimoctf2dss`.

1.2 Zero-Order-Hold Discretization

Consider a continuous-time linear differential equation

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1.5}$$

The solution of this equation is

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \tag{1.6}$$

Let $t_0 = kT$ and $t = (k+1)T$. Let T denote the sampling time and let the function $u(\cdot)$ be a zero-order-hold function in each sample period, i.e.

$$u(t) = u(kT) \qquad kT \leq t < (k+1)T \tag{1.7}$$

Then

$$\begin{aligned} x((k+1)T) &= e^{AT}x(kT) + \int_{kT}^{(k+1)T} e^{A(kT+T-\tau)}Bu(kT)d\tau \\ &= e^{AT}x(kT) + \int_{T}^{0} e^{A\sigma}B(-d\sigma)\,u(kT) \\ &= e^{AT}x(kT) + \int_{0}^{T} e^{A\sigma}Bd\sigma\,u(kT) \\ &= \Phi x(kT) + \Gamma u(kT) \end{aligned} \tag{1.8}$$

in which

$$\Phi = e^{AT} \tag{1.9a}$$

$$\Gamma = \int_{0}^{T} e^{A\sigma}Bd\sigma \tag{1.9b}$$

Consequently, using the zero-order-hold of $u(t)$, a discrete-time equivalent of the continuous-time linear differential equation is given by

$$x((k+1)T) = \Phi x(kT) + \Gamma u(kT) \tag{1.10}$$

Van Loan (1978) shows that Φ and Γ may be computed by computation of the following matrix exponential

$$\begin{aligned} \exp\left(\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} T\right) &= \begin{bmatrix} e^{AT} & \int_0^T e^{A(T-s)}Bds \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} e^{AT} & \int_0^T e^{A\sigma}Bd\sigma \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} \Phi & \Gamma \\ 0 & I \end{bmatrix} \end{aligned} \tag{1.11}$$

In Matlab the matrix exponential is computed using the built-in function `expm`. Moler and Van Loan (1978) describes methods for computing the matrix exponential. Sidje (1998) provides a set of FORTRAN algorithms for computation of the matrix exponential.

The Linear Model Predictive Control Toolbox contains the function `css2dsszoh` for computation of Φ and Γ given A, B, and T.

1.3 Continuous-time Realization of SISO Systems

Consider a SISO system represented by the input-output relation

$$y(s) = g(s)u(s) \tag{1.12}$$

in which the transfer function is given by

$$g(s) = h(s)e^{-\lambda s} = \frac{b(s)}{a(s)}\, e^{-\lambda s} \tag{1.13}$$

and

$$a(s) = a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1}s + a_n \tag{1.14a}$$
$$b(s) = b_0 s^n + b_1 s^{n-1} + \ldots + b_{n-1}s + b_n \tag{1.14b}$$

To be realizable as a continuous-time state space system with delay

$$\dot{x}(t) = Ax(t) + Bu(t-\lambda) \tag{1.15a}$$
$$y(t) = Cx(t) + Du(t-\lambda) \tag{1.15b}$$

the degree of the polynomial $a(s)$ must be equal to or larger than the degree of the polynomial $b(s)$. Assume that the degree of $a(s)$ is n, that is $a_0 \neq 0$. Then we may represent the transfer function as

$$y(s) = \frac{b(s)}{a(s)}\, e^{-\lambda s}u(s) = \frac{b(s)}{a(s)}\, u(s-\lambda) = \frac{\tilde{b}(s)}{\tilde{a}(s)}\, u(s-\lambda) \tag{1.16}$$

in which we have defined $\tilde{a}(s)$ and $\tilde{b}(s)$ as

$$\tilde{a}(s) = s^n + \tilde{a}_1 s^{n-1} + \ldots + \tilde{a}_{n-1}s + \tilde{a}_n \tag{1.17a}$$
$$\tilde{b}(s) = \tilde{b}_0 s^n + \tilde{b}_1 s^{n-1} + \ldots + \tilde{b}_{n-1}s + \tilde{b}_n \tag{1.17b}$$

with the coefficients given as

$$\tilde{a}_i = \frac{a_i}{a_0} \quad i = 1, 2, \ldots, n \tag{1.18a}$$
$$\tilde{b}_i = \frac{b_i}{a_0} \quad i = 0, 1, \ldots, n \tag{1.18b}$$

An observer canonical realization of this transfer function is given by the state space system

$$\dot{x}(t) = A_o x(t) + B_o u(t-\lambda) \tag{1.19a}$$
$$y(t) = C_o x(t) + D_o u(t-\lambda) \tag{1.19b}$$

in which the matrices (A_o, B_o, C_o, D_o) are constructed as

$$A_o = \begin{bmatrix} -\tilde{a}_1 & 1 & 0 & \dots & 0 \\ -\tilde{a}_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ -\tilde{a}_{n-1} & 0 & 0 & \dots & 1 \\ -\tilde{a}_n & 0 & 0 & \dots & 0 \end{bmatrix} \quad B_o = \begin{bmatrix} \tilde{b}_1 - \tilde{a}_1\tilde{b}_0 \\ \tilde{b}_2 - \tilde{a}_2\tilde{b}_0 \\ \vdots \\ \tilde{b}_{n-1} - \tilde{a}_{n-1}\tilde{b}_0 \\ \tilde{b}_n - \tilde{a}_n\tilde{b}_0 \end{bmatrix}$$

$$C_o = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \end{bmatrix} \qquad D_o = \tilde{b}_0$$

This realization of a SISO continuous-time transfer function with delay is implemented in `sisoctf2css`.

1.4 Discretization of SISO State-Space Systems

The discussion in the following section will be limited to SISO systems. However, the procedure is equally applicable to MIMO systems and will give essentially the same results, e.g. the ones in the shift part of the matrices are exchanges by identity matrices of appropriate dimension.

Consider the continuous-time linear time invariant state space system with delay, $\lambda \geq 0$:

$$\dot{x}(t) = Ax(t) + Bu(t - \lambda) \tag{1.20a}$$
$$y(t) = Cx(t) + Du(t - \lambda) \tag{1.20b}$$

For simplicity assume that $u(\cdot) : \mathbb{R} \mapsto \mathbb{R}$ and $y(\cdot) : \mathbb{R} \mapsto \mathbb{R}$, i.e. that the system is a SISO system. Assume that the inputs are sampled with zero-order-hold

$$u(t) = u(kT) \qquad kT \leq t \leq (k+1)T \tag{1.21}$$

T is the sampling period. Then the challenge is to compute an equivalent discrete-time system

$$z((k+1)T) = A_d z(kT) + B_d u((k-l)T) \tag{1.22a}$$
$$y(kT) = C_d z(kT) + D_d u((k-l)T) \tag{1.22b}$$

$l \in \mathbb{N}_0$ is a non-negative integer representing the delay in the discrete-time system.

1.4.1 Derivation of the Solution

The solution of (1.69a) is

$$x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau - \lambda)d\tau \tag{1.23}$$

Let $t_0 = kT$ and $t = (k+1)T$ then

$$x(kT + T) = e^{AT}x(kT) + \int_{kT}^{kT+T} e^{A(kT+T-\tau)}Bu(\tau - \lambda)d\tau \tag{1.24}$$

Introduce the variable $\eta = kT + T - \tau$ to simplify this integral

$$\begin{aligned} x(kT+T) &= e^{AT}x(kT) + \int_{T}^{0} e^{A\eta}Bu(kT+T-\lambda-\eta)(-d\eta) \\ &= e^{AT}x(kT) + \int_{0}^{T} e^{A\eta}Bu(kT+T-\lambda-\eta)d\eta \end{aligned} \tag{1.25}$$

Define $\lambda = lT - mT$ in which $l \in \mathbb{N}_0$ and $0 \leq m < 1$. Then using the zero-order-hold sampling of $u(\cdot)$ gives

$$\begin{aligned} x(kT+T) &= e^{AT}x(kT) + \int_{0}^{T} e^{A\eta}Bu(kT+T-\lambda-\eta)d\eta \\ &= e^{AT}x(kT) + \int_{0}^{mT} e^{A\eta}Bu(kT+T-lT+mT-\eta)d\eta \\ &\quad + \int_{mT}^{T} e^{A\eta}Bu(kT+T-lT+mT-\eta)d\eta \\ &= e^{AT}x(kT) + \int_{0}^{mT} e^{A\eta}Bd\eta\, u(kT+T-lT) \\ &\quad + \int_{mT}^{T} e^{A\eta}Bd\eta\, u(kT-lT) \\ &= \Phi x(kT) + \Gamma_1 u((k-l)T) + \Gamma_2 u((k+1-l)T) \end{aligned} \tag{1.26}$$

in which

$$\Phi = e^{AT} \quad \Gamma_1 = \int_{mT}^{T} e^{A\eta}Bd\eta \quad \Gamma_2 = \int_{0}^{mT} e^{A\eta}Bd\eta \tag{1.27}$$

In summary, the discrete time equivalent of (1.69a) is

$$x((k+1)T) = \Phi x(kT) + \Gamma_1 u((k-l)T) + \Gamma_2 u((k+1-l)T) \tag{1.28}$$

with Φ, Γ_1, and Γ_2 given by (1.27).

1.4.1.1 Case: $m = 0$

In this case, the time delay is an integer multiple of the sampling time.
With $m = 0$, Γ_1 and Γ_2 becomes

$$\Gamma_1 = \int_{0}^{T} e^{A\eta}Bd\eta \qquad \Gamma_2 = \int_{0}^{0} e^{A\eta}Bd\eta = 0 \tag{1.29}$$

Consequently, (1.28) reduces to

$$x(kT+T) = \Phi x(kT) + \Gamma_1 u((k-l)T) \tag{1.30}$$

in which

$$\Phi = e^{AT} \qquad \Gamma_1 = \int_{0}^{T} e^{A\eta}Bd\eta \tag{1.31}$$

These matrices may be computed as

$$\begin{bmatrix} \Phi & \Gamma_1 \\ 0 & I \end{bmatrix} = \exp\left(\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} T \right) \tag{1.32}$$

using `css2dsszoh`. Using the zero-order-hold sampling of $u(\cdot)$ and the fact that $m = 0$ we obtain the following system

$$x(kT+T) = \Phi x(kT) + \Gamma_1 u((k-l)T) \tag{1.33a}$$
$$y(kT) = Cx(kT) + Du((k-l)T) \tag{1.33b}$$

which is identically to the continuous-time system at the sampling instants.

Hence, for $m = 0$ we have

$$z(kT) = x(kT) \tag{1.34}$$

and

$$z((k+1)T) = A_d z(kT) + B_d u((k-l)T) \tag{1.35a}$$
$$y(kT) = C_d z(kT) + D_d u((k-l)T) \tag{1.35b}$$

in which

$$A_d = \Phi \tag{1.36a}$$
$$B_d = \Gamma_1 \tag{1.36b}$$
$$C_d = C \tag{1.36c}$$
$$D_d = D \tag{1.36d}$$

and $l = \lambda/T$.

Case: $l > 0$

In the case when $l > 0$ we will show how to express the discrete-time state space system with time delay (1.33) as a standard discrete-time state space system without time delay. For the purpose of generating step and impulse responses, this is not advantageous as it increases the state dimension. However, for some applications a standard state space model without delay is required.

Define $u_k = u(k) = u(kT)$ and introduce the states $w_1, \ldots, w_l$ defined by

$$w_1(k) = u(k-l) \qquad w_1(k+1) = u(k-l+1) = w_2(k) \tag{1.37a}$$
$$w_2(k) = u(k-l+1) \qquad w_2(k+1) = u(k-l+1+1) = w_3(k) \tag{1.37b}$$
$$\vdots \qquad\qquad \vdots \tag{1.37c}$$
$$w_{l-1}(k) = u(k-2) \qquad w_{l-1}(k+1) = u(k-1) = w_l(k) \tag{1.37d}$$
$$w_l(k) = u(k-1) \qquad w_l(k+1) = u(k) \tag{1.37e}$$

which gives

$$x(k+1) = \Phi x(k) + \Gamma_1 u(k-l) = \Phi x(k) + \Gamma_1 w_1(k) \tag{1.38}$$
$$y(k) = Cx(k) + Du(k-l) = Cx(k) + Dw_1(k) \tag{1.39}$$

Define the new state by

$$z(k) = \begin{bmatrix} x(k) \\ w_1(k) \\ w_2(k) \\ \vdots \\ w_{l-1}(k) \\ w_l(k) \end{bmatrix} = \begin{bmatrix} x(k) \\ u(k-l) \\ u(k-l+1) \\ \vdots \\ u(k-2) \\ u(k-1) \end{bmatrix} \tag{1.40}$$

Then the standard discrete time state space system corresponding to the original continuous-time state space system (1.69) is

$$z(k+1) = A_d z(k) + B_d u(k) \tag{1.41a}$$
$$y(k) = C_d z(k) + D_d u(k) \tag{1.41b}$$

in which

$$A_d = \left[\begin{array}{c|cccc} \Phi & \Gamma_1 & 0 & 0 & \dots & 0 \\ \hline 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 \end{array}\right] \qquad B_d = \left[\begin{array}{c} 0 \\ \hline 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{array}\right] \tag{1.42}$$

$$C_d = \left[\begin{array}{c|cccc} C & D & 0 & 0 & \dots & 0 \end{array}\right] \qquad D_d = 0 \tag{1.43}$$

As is evident the state dimension increases by l-states compared to the realization with an explicit time delay.

1.4.1.2 Case: $0 < m < 1$

In this situation we must compute the matrices

$$\Phi = e^{AT} \quad \Gamma_1 = \int_{mT}^{T} e^{A\eta} B d\eta \quad \Gamma_2 = \int_{0}^{mT} e^{A\eta} B d\eta \tag{1.44}$$

The idea in the following derivation is to express the integrals as quantities that may be computed by `css2dsszoh`. To do this note that Γ_1 may be expressed as

$$\Gamma_1 = \int_{mT}^{T} e^{A\eta} B d\eta = \int_{0}^{T-mT} e^{A(mT+\sigma)} B d\sigma = e^{AmT} \int_{0}^{T-mT} e^{A\sigma} B d\sigma \tag{1.45}$$

Introduce Φ_1 and $\tilde{\Gamma}_1$ as the result of the following matrix exponential evaluation

$$\begin{bmatrix} \Phi_1 & \tilde{\Gamma}_1 \\ 0 & I \end{bmatrix} = \exp\left(\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} (T - mT) \right) \tag{1.46}$$

and note that

$$\Phi_1 = e^{A(T-mT)} \tag{1.47}$$
$$\tilde{\Gamma}_1 = \int_{0}^{T-mT} e^{A\sigma} B d\sigma \tag{1.48}$$

By a similar procedure, Γ_2 may be obtained by evaluation of the matrix exponential

$$\begin{bmatrix} \Phi_2 & \Gamma_2 \\ 0 & I \end{bmatrix} = \exp\left(\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} mT \right) \tag{1.49}$$

Note that

$$\Phi_2 = e^{AmT} \tag{1.50}$$

Then Γ_1 may be computed as

$$\Gamma_1 = e^{AmT} \int_0^{T-mT} e^{A\sigma} B d\sigma = \Phi_2 \tilde{\Gamma}_1 \tag{1.51}$$

Furthermore

$$\Phi = e^{AT} = e^{AmT} e^{A(T-mT)} = \Phi_2 \Phi_1 \tag{1.52}$$

With the definition of these matrices we have the state transition equation defined

$$x(kT+T) = \Phi x(kT) + \Gamma_1 u((k-l)T) + \Gamma_2 u((k+1-l)T) \tag{1.53}$$

Now notice that $l > 0$ for $\tau \geq 0$ in the case $0 < m < 1$.
Note that due to the zero-order-hold we have

$$u(kT-\lambda) = u(kT-lT+mT) = u((k-l)T+mT)) = u((k-l)T) \tag{1.54}$$

Consequently, the output equation at time $t = kT$ may be expressed as

$$y(kT) = Cx(kT) + Du(kT-\lambda) = Cx(kT) + Du((k-l)T) \tag{1.55}$$

Define a new state vector, $z(k)$, as

$$z(k) = \begin{bmatrix} x(k) \\ u(k-l) \end{bmatrix} \tag{1.56}$$

then

$$\begin{aligned} z(k+1) = \begin{bmatrix} x(k+1) \\ u(k+1-l) \end{bmatrix} &= \begin{bmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k-l) \end{bmatrix} + \begin{bmatrix} \Gamma_2 \\ 1 \end{bmatrix} u(k+1-l) \\ &= A_d z(k) + B_d u(k+1-l) \end{aligned} \tag{1.57}$$

with

$$A_d = \begin{bmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{bmatrix} \qquad B_d = \begin{bmatrix} \Gamma_2 \\ 1 \end{bmatrix} \tag{1.58}$$

Similarly, the output equation is

$$\begin{aligned} y(k) &= Cx(k) + Du(k-l) \\ &= \begin{bmatrix} C & D \end{bmatrix} \begin{bmatrix} x(k) \\ u(k-l) \end{bmatrix} + 0u(k+1-l) \\ &= C_d z(k) + D_d u(k+1-l) \end{aligned} \tag{1.59}$$

in which

$$C_d = \begin{bmatrix} C & D \end{bmatrix} \qquad D_d = 0 \tag{1.60}$$

Consequently, we may represent the continuous-time system at the discrete times $t_k = kT_s$ by the following discrete-time state space system with delay

$$z((k+1)T) = A_d z(kT) + B_d u((k-l_1)T) \tag{1.61a}$$

$$y(kT) = C_d z(kT) + D_d u((k-l_1)T) \tag{1.61b}$$

in which $l_1 = l - 1 \geq 0$ and

$$A_d = \begin{bmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{bmatrix} \qquad B_d = \begin{bmatrix} \Gamma_2 \\ 1 \end{bmatrix} \tag{1.62a}$$

$$C_d = \begin{bmatrix} C & D \end{bmatrix} \qquad D_d = 0 \tag{1.62b}$$

For $l_1 = l - 1 = 0$ (corresponding to $l = 1$) this is a standard discrete time state space system without delay. For computation of the step and impulse responses of the considered SISO system, it is most efficient to represent the system as a discrete time system with delay. In this way the dimension of the state space matrices are not increased.

Case: $l > 1$ **(**$l_1 = l - 1 > 0$**)** For the case $l > 1$ the system obtained is a discrete-time state space with delay. In the following discussion it is demonstrated how this system may be converted to a standard discrete time state space system without delay.

Define

$$w_1(k) = u(k-l) \qquad w_1(k+1) = u(k-l+1) = w_2(k) \tag{1.63a}$$

$$w_2(k) = u(k-l+1) \qquad w_2(k+1) = u(k-l+1+1) = w_3(k) \tag{1.63b}$$

$$\vdots \qquad\qquad \vdots$$

$$w_{l-1}(k) = u(k-2) \qquad w_{l-1}(k+1) = u(k-1) = w_l(k) \tag{1.63c}$$

$$w_l(k) = u(k-1) \qquad w_l(k+1) = u(k) \tag{1.63d}$$

and define a new state vector as

$$z(k) = \begin{bmatrix} x(k) \\ w_1(k) \\ w_2(k) \\ \vdots \\ w_l(k) \end{bmatrix} = \begin{bmatrix} x(k) \\ u(k-l) \\ u(k-l+1) \\ \vdots \\ u(k-1) \end{bmatrix} \tag{1.64}$$

Then the corresponding discrete time system without delay may be represented as

$$z(k+1) = A_d z(k) + B_d u(k) \tag{1.65a}$$

$$y(k) = C_d z(k) + D_d u(k) \tag{1.65b}$$

in which

$$A_d = \left[\begin{array}{c|cccccc} \Phi & \Gamma_1 & \Gamma_2 & 0 & \dots & 0 \\ \hline 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 \end{array}\right] \qquad B_d = \left[\begin{array}{c} 0 \\ \hline 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{array}\right] \tag{1.66}$$

$$C_d = \left[\begin{array}{c|cccc} C & D & 0 & 0 & \dots & 0 \end{array}\right] \qquad D_d = 0 \tag{1.67}$$

1.4.2 Implementation

`sisocss2dss` implements computation of (A_d, B_d, C_d, D_d, l) for the SISO discrete time state space system with delay

$$z((k+1)T) = A_d z(kT) + B_d u((k-l)T) \tag{1.68a}$$
$$y(kT) = C_d z(kT) + D_d u((k-l)T) \tag{1.68b}$$

such that they are equivalent with the continuous-time SISO state space system with delay

$$\dot{x}(t) = Ax(t) + Bu(t-\lambda) \tag{1.69a}$$
$$y(t) = Cx(t) + Du(t-\lambda) \tag{1.69b}$$

and zero-order-hold sampling of $u(t)$ with sampling time, T.

1.5 SISO Discrete-Time Impulse Response

The discrete-time impulse response of a SISO transfer function

$$y(s) = g(s)u(s) = h(s)e^{-\lambda s}u(s) = h(s)u(s-\lambda) \tag{1.70}$$

is obtained by doing the following sequence of computations

1. Convert the SISO transfer function to a SISO continuous state-space model with delay.
2. Convert the SISO continuous state-space model with delay to a SISO discrete-time state-space model with delay.
3. Compute the discrete-time impulse response of a SISO discrete-time system with delay.

Therefore, assume that the SISO discrete-time state space representation with delay

$$x(k+1) = A_d x(k) + B_d u(k-l) \tag{1.71a}$$
$$y(k) = C_d x(k) + D_d u(k-l) \tag{1.71b}$$

is available. The impulse response coefficient, $\{h(k)\}_{k=0}^{N}$ at times kT for $k = 0, 1, \ldots, N$ are defined as $h(k) = y(k)$ when

$$x(0) = 0 \tag{1.72a}$$
$$u(k) = \begin{cases} 0 & k < 0 \\ 1 & k = 0 \\ 0 & k > 0 \end{cases} \tag{1.72b}$$

and may be computed by algorithm 1.

This algorithm is implemented in `sisodss2dimpulse`. The computation of the impulse response from the SISO transfer function with delay is implemented in `sisoctf2dimpulse`.

Algorithm 1 Impulse Response Coefficients for SISO system with delay.

$h(i) \leftarrow 0$ for $i = 0, 1, \ldots, l-1$
$h(l) \leftarrow D_d$
$x \leftarrow B_d$
for i=l+1 to N **do**
  $h(i) \leftarrow C_d x$
  $x \leftarrow A_d x$
end for

1.6 SISO Discrete-Time Step Response

The discrete-time step response of a SISO transfer function

$$y(s) = g(s)u(s) = h(s)e^{-\lambda s}u(s) = h(s)u(s-\lambda) \tag{1.73}$$

is obtained by doing the following sequence of computations

1. Convert the SISO transfer function to a SISO continuous state-space model with delay.
2. Convert the SISO continuous state-space model with delay to a SISO discrete-time state-space model with delay.
3. Compute the discrete-time step response of a SISO discrete-time system with delay.

Therefore, assume that the SISO discrete-time state space representation with delay

$$x(k+1) = A_d x(k) + B_d u(k-l) \tag{1.74a}$$
$$y(k) = C_d x(k) + D_d u(k-l) \tag{1.74b}$$

is available. The step response coefficient, $\{s(k)\}_{k=0}^{N}$ at times kT for $k = 0, 1, \ldots, N$ are defined as $s(k) = y(k)$ when

$$x(0) = 0 \tag{1.75a}$$
$$u(k) = \begin{cases} 0 & k < 0 \\ 1 & k \geq 0 \end{cases} \tag{1.75b}$$

and may be computed by algorithm 2.

This algorithm is implemented in `sisodss2dstep`. The computation of the step response from the SISO transfer function with delay is implemented in `sisoctf2dstep`.

1.7 Direct Computation of a MIMO Impulse Response

Let a MIMO transfer function

$$Y(s) = G(s)U(s) \tag{1.76}$$

Algorithm 2 Step Response Coefficients for SISO system with delay.

$s(i) \leftarrow 0$ for $i = 0, 1, \ldots, l-1$
$s(l) \leftarrow D_d$
$x \leftarrow B_d$
for i=l+1 to N **do**
  $s(i) \leftarrow C_d x + D_d$
  $x \leftarrow A_d x + B_d$
end for

in which

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) & \ldots & g_{1m}(s) \\ g_{21}(s) & g_{22}(s) & \ldots & g_{2m}(s) \\ \vdots & \vdots & & \vdots \\ g_{p1}(s) & g_{p2}(s) & \ldots & g_{pm}(s) \end{bmatrix} \tag{1.77}$$

and

$$g_{ij}(s) = h_{ij}(s)e^{-\lambda_{ij}s} \tag{1.78}$$

be given.

The discrete-time impulse response at time kT is denoted

$$H(k) = \begin{bmatrix} h_{11}(k) & h_{12}(k) & \ldots & h_{1m}(k) \\ h_{21}(k) & h_{22}(k) & \ldots & h_{2m}(k) \\ \vdots & \vdots & & \vdots \\ h_{p1}(k) & h_{p2}(k) & \ldots & h_{pm}(k) \end{bmatrix} \quad k = 0, 1, \ldots, N \tag{1.79}$$

in which $h_{ij}(k)$ is the impulse response coefficient at time kT of output i and input j.

Consequently, the impulse response coefficients (Markov parameters), $\{H(k)\}_{k=0}^{N}$, may be computed by computing the impulse response coefficients, $\{h_{ij}(k)\}_{k=0}^{N}$ for each input-output par, e.g. for all $i = 1, \ldots, p$ and $j = 1, \ldots, m$. The advantage of this method is that one do not need to assemble a discrete-time state space model for the entire system but can do each input-output pair individually. The state-space models that must be realized for each input-output pair are relatively small.

The computation of the discrete-time impulse response matrices $\{S(k)\}_{k=0}^{N}$ from the continuous-time MIMO transfer function is implemented in `mimoctf2dimpulse`.

1.8 Direct Computation of a MIMO Step Response

Let a MIMO transfer function

$$Y(s) = G(s)U(s) \tag{1.80}$$

in which

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) & \ldots & g_{1m}(s) \\ g_{21}(s) & g_{22}(s) & \ldots & g_{2m}(s) \\ \vdots & \vdots & & \vdots \\ g_{p1}(s) & g_{p2}(s) & \ldots & g_{pm}(s) \end{bmatrix} \tag{1.81}$$

and

$$g_{ij}(s) = h_{ij}(s)e^{-\lambda_{ij}s} \tag{1.82}$$

be given. The discrete-time step response at time kT is denoted

$$S(k) = \begin{bmatrix} s_{11}(k) & s_{12}(k) & \dots & s_{1m}(k) \\ s_{21}(k) & s_{22}(k) & \dots & s_{2m}(k) \\ \vdots & \vdots & & \vdots \\ s_{p1}(k) & s_{p2}(k) & \dots & s_{pm}(k) \end{bmatrix} \quad k = 0, 1, \dots, N \tag{1.83}$$

in which $s_{ij}(k)$ is the step response coefficient at time kT of output i and input j.

Consequently, the step response coefficients, $\{S(k)\}_{k=0}^{N}$, may be computed by computing the step response coefficients, $\{s_{ij}(k)\}_{k=0}^{N}$ for each input-output par, e.g. for all $i = 1, \dots, p$ and $j = 1, \dots, m$. The advantage of this method is that one do not need to assemble a discrete-time state space model for the entire system but can do each input-output pair individually. The state-space models that must be realized for each input-output pair are relatively small.

The computation of the discrete-time step response matrices $\{S(k)\}_{k=0}^{N}$ from the continuous-time MIMO transfer function is implemented in `mimoctf2dstep`.

1.9 Realization from an Impulse Response Model

Given a sequence, $\{H_i\}_{i=0}^{2N}$, of impulse response matrices, $H_i \in \mathbb{R}^{p\times m}$, generated by a linear time-invariant system. Determine the minimal system order n and the system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, and $D \in \mathbb{R}^{p\times m}$ of

$$\Sigma_n(A, B, C, D) \triangleq \begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k + Du_k \end{cases} \tag{1.84}$$

that produces the impulse response coefficients $\{H_i\}_{i=0}^{2N}$. This is the deterministic minimal realization of a state space model from the impulse response coefficients (Ho and Kalman, 1966Silverman, 1971de Jong, 1978Backx, 1987). In the mathematical literature, the impulse response coefficients, H_i, are often referred to as Markov parameters. The procedure described is based on the presentation by Chen (1999).

Let N be some integer larger than or equal to the minimal system order, n. As the system order n is unknown, some diagnostic measure to determine whether N is large enough is necessary. The Hankel matrix, $\mathcal{H}_{N,N}$, is defined in terms of the given impulse response matrices, $\{H_i\}_{i=1}^{2N-1}$, as

$$\mathcal{H}_{N,N} \triangleq \begin{bmatrix} H_1 & H_2 & \dots & H_N \\ H_2 & H_3 & \dots & H_{N+1} \\ \vdots & \vdots & & \vdots \\ H_N & H_{N+1} & \dots & H_{2N-1} \end{bmatrix} \tag{1.85}$$

and may therefore be constructed from the information given. A Hankel matrix, $\mathcal{H}_{i,j}$, is defined in terms of the impulse response matrices, $\{H_i\}$, as

$$\mathcal{H}_{i,j} = \begin{bmatrix} H_1 & H_2 & \dots & H_j \\ H_2 & H_3 & \dots & H_{j+1} \\ \vdots & \vdots & & \vdots \\ H_i & H_{i+1} & \dots & H_{i+j-1} \end{bmatrix} \tag{1.86}$$

According to Kalman *et al.* (1968), the impulse response $\{H_i\}_{i=1}^{\infty}$ has a minimal realization $\Sigma_n(A, B, C, D)$ of order n if and only if

$$\operatorname{rank}\{\mathcal{H}_{n+1,j}\} = \operatorname{rank}\{\mathcal{H}_{n,j}\} = n \qquad j = n, n+1, \dots \tag{1.87}$$

This implies that $\{H_i\}_{i=1}^{\infty}$ has a minimal realization, $\Sigma_n(A, B, C, D)$, of order n if and only if there exists an $N \geq n+1$ such that a SVD-decomposition of $\mathcal{H}_{N,N}$ yields

$$\mathcal{H}_{N,N} = K\Lambda L' = \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} L_1 & L_2 \end{bmatrix}' = K_1 \Lambda_1 L_1' \tag{1.88}$$

in which $\Lambda_1 \in \mathbb{R}^{n \times n}$ is a diagonal matrix

$$\Lambda_1 = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \tag{1.89}$$

with positive elements on the diagonal, $\lambda_i > 0$, and $\lambda_{i+1} \leq \lambda_i$ for $i = 1, 2, \dots, n$. The matrices K and L are orthogonal matrices of appropriate dimensions. The minimal system order $n \leq n_{max}$ may be obtained (if it exists) by increasing $N \leq n_{max} + 1$ until the first zero on the diagonal of Λ is obtained. In practice, the mathematical zero is exchanged for some tolerance (say 10^{-15}).

The extended controllability matrix, $\mathcal{C}_N$, of the linear time invariant system (1.84) is defined as

$$\mathcal{C}_N = \begin{bmatrix} B & AB & A^2B & \dots & A^{N-1}B \end{bmatrix} \tag{1.90}$$

Similarly, the extended observability matrix, $\mathcal{O}_N$, of (1.84) is defined as

$$\mathcal{O}_N = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{N-1} \end{bmatrix} \tag{1.91}$$

The impulse response coefficients, $\{H_i\}_{i=1}^{2N}$, are assumed given. They may be obtained by impulse response experiments or other system identification techniques in which an impulse response model is identified. The impulse response coefficients, H_i, are related to the matrices (A, B, C, D) in (1.84) by the expression

$$H_i = \begin{cases} D & i = 0 \\ CA^{i-1}B & i = 1, 2, \dots \end{cases} \tag{1.92}$$

The Hankel matrix, $\mathcal{H}_{N,N}$, of the linear system (1.84) is defined in terms of the impulse response coefficients and related to the extended observability matrix, $\mathcal{O}_N$, and the extended controllability matrix, $\mathcal{C}_N$, according to

$$\begin{aligned}\mathcal{H}_{N,N} &\triangleq \begin{bmatrix} H_1 & H_2 & \dots & H_N \\ H_2 & H_3 & \dots & H_{N+1} \\ \vdots & \vdots & & \vdots \\ H_N & H_{N+1} & \dots & H_{2N-1} \end{bmatrix} \\ &= \begin{bmatrix} CB & CAB & \dots & CA^{N-1}B \\ CAB & CA^2B & \dots & CA^NB \\ \vdots & \vdots & & \vdots \\ CA^{N-1}B & CA^NB & \dots & CA^{2N-2}B \end{bmatrix} = \mathcal{O}_N\mathcal{C}_N\end{aligned} \tag{1.93}$$

The Hankel matrix, $\mathcal{H}_{N,N}$ is given as the impulse response matrices, $\{H_i\}_{i=1}^{2N-1}$, needed for its construction are given. Consequently, the relation

$$\mathcal{O}_N\mathcal{C}_N = \mathcal{H}_{N,N} = K_1\Lambda_1 L_1' \tag{1.94}$$

may be used to compute the extended observability matrix, $\mathcal{O}_N$, and the extended controllability matrix, $\mathcal{C}_N$ from the SVD-decomposition of the Hankel matrix, $\mathcal{H}_{N,N}$. The expressions for the extended observability and controllability matrices are

$$\mathcal{O}_N = K_1\Lambda_1^{1/2} \tag{1.95a}$$

$$\mathcal{C}_N = \Lambda_1^{1/2}L_1' \tag{1.95b}$$

in which we have exploited that K_1 as well as L_1 are orthogonal matrices as K and L are orthogonal matrices. The matrix, B, may be extracted as the first m columns of the extended controllability matrix, $\mathcal{C}_N$, according to (1.90). Similarly, the matrix, C, may be extracted as the first p rows of the extended observability matrix, $\mathcal{O}_N$. Hence

$$B = (\mathcal{C}_N)_{:,1:m} = \Lambda_1^{1/2}[(L_1)_{1:m,:}]' \tag{1.96}$$

$$C = (\mathcal{O}_N)_{1:p,:} = (K_1)_{1:p,:}\,\Lambda_1^{1/2} \tag{1.97}$$

To obtain the matrix, A, we define a modified Hankel matrix, $\tilde{\mathcal{H}}_{N+1,N+1}$, as

$$\tilde{\mathcal{H}}_{N+1,N+1} = \begin{bmatrix} H_2 & H_3 & \dots & H_{N+1} \\ H_3 & H_4 & \dots & H_{N+2} \\ \vdots & \vdots & & \vdots \\ H_{N+1} & H_{N+2} & \dots & H_{2N} \end{bmatrix} \tag{1.98}$$

and notice that it is related to $\mathcal{O}_N$, A, and $\mathcal{C}_N$ as

$$\tilde{\mathcal{H}}_{N+1,N+1} = \begin{bmatrix} CAB & CA^2B & \dots & CA^NB \\ CA^2B & CA^3B & \dots & CA^{N+1}B \\ \vdots & \vdots & & \vdots \\ CA^NB & CA^{N+1}B & \dots & CA^{2N-1}B \end{bmatrix} = \mathcal{O}_N A\mathcal{C}_N \tag{1.99}$$

which may be expressed as

$$\tilde{\mathcal{H}}_{N+1,N+1} = \mathcal{O}_N A \mathcal{C}_N = K_1 \Lambda_1^{1/2} A \Lambda_1^{1/2} L_1' \tag{1.100}$$

using the expressions for $\mathcal{O}_N$ and $\mathcal{C}_N$. Consequently, utilizing that K_1 and L_1 are orthogonal matrices and that Λ_1 is a positive-diagonal matrix, the A-matrix may be computed as

$$A = \Lambda_1^{-1/2} K_1' \tilde{\mathcal{H}}_{N+1,N+1} L_1 \Lambda_1^{-1/2} \tag{1.101}$$

In summary, the matrices, (A, B, C, D), in the minimal realization $\Sigma_n(A, B, C, D)$ from impulse response matrices are obtained as

$$A = \Lambda_1^{-1/2} K_1' \tilde{\mathcal{H}}_{N+1,N+1} L_1 \Lambda_1^{-1/2} \tag{1.102a}$$

$$B = (\mathcal{C}_N)_{:,1:m} = \Lambda_1^{1/2} [(L_1)_{1:m,:}]' \tag{1.102b}$$

$$C = (\mathcal{O}_N)_{1:p,:} = (K_1)_{1:p,:} \Lambda_1^{1/2} \tag{1.102c}$$

$$D = H_0 \tag{1.102d}$$

This realization is balanced in the sense that

$$\mathcal{O}_N' \mathcal{O}_N = \Lambda_1^{1/2} K_1' K_1 \Lambda_1^{1/2} = \Lambda_1 \tag{1.103a}$$

$$\mathcal{C}_N \mathcal{C}_N' = \Lambda_1^{1/2} L_1' L_1 \Lambda_1^{1/2} = \Lambda_1 \tag{1.103b}$$

By construction, the obtained realization is a minimal realization, which implies that the realization is both controllable and observable. An alternative way to see this is by the relation

$$\mathcal{H}_{n,n} = \mathcal{O}_n \mathcal{C}_n \tag{1.104}$$

The Hankel matrix, $\mathcal{H}_{n,n}$, has rank n by construction of n. This implies that $\mathcal{O}_n$ and $\mathcal{C}_n$ has rank n, which implies that the realization (A, B, C, D) is both controllable and observable.

As is always the case with realization of linear time invariant systems, they are only unique to within a similar transformation.

The realization of a discrete-time MIMO state space system from discrete-time impulse response coefficients is implemented in `mimodimpulse2dss`.

1.9.1 Realization of Continuous-Time MIMO Transfer Functions

A MIMO continuous-time transfer function may be realized through a two step procedure

1. Compute the discrete-time impulse response matrices of the MIMO continuous-time transfer function using `mimoctf2dimpulse`.
2. Compute the state space matrices (A, B, C, D) in a standard (without delay) discrete-time state space model using `mimodimpulse2dss`.

This two step procedure is implemented in `mimoctf2dss`.

1.10 Matlab Implementations

```
%
% REALIZATION OF TRANSFER FUNCTIONS
% ------------------------------------------------------------------------------------------
%   css2dsszoh          State space matrices of a zoh discretization
%
%   sisoctf2css         Cont. time transfer function to cont. time state space (SISO)
%   sisocss2dss         Cont. time state space to disc. time state space (SISO)
%   sisodss2dimpulse    Discrete time state space to impulse response (SISO)
%   sisodss2dstep       Discrete time state space to step response (SISO)
%   sisoctf2dss         Cont. time transfer function to disc. time state space (SISO)
%   sisoctf2dimpulse    Cont. time transfer function to disc. time impulse response (SISO)
%   sisoctf2dstep       Cont. time transfer function to disc. time step response (SISO)
%
%   mimoctf2dimpulse    Cont. time transfer function to disc. time impulse response (MIMO)
%   mimoctf2dstep       Cont. time transfer function to disc. time step response (MIMO)
%   mimodimpulse2dss    Disc. time impulse response to disc. time state space (MIMO)
%   mimoctf2dss         Cont. time transfer function to disc. time state space (MIMO)
%   mimodsss2dimpulse   Disc. time state space to disc. time impulse response (MIMO)
%   mimodss2dstep       Disc. time state space to disc. time step response (MIMO)
%
```

1.10.1 css2dsszoh

```
%
% css2dsszoh Computes state space matrices for a zero-order hold discretization
%
%           Let a continuous-time system be given by
%
%               d/dt x(t) = A*x(t) + B*u(t)
%
%           Assume that u(t) is implemented as a zero-order-hold
%
%               u(t) = u(k*Ts)          k*Ts <= t < (k+1)*Ts
%
%           Then the discrete time equivalent system may be
%           expressed as
%
%               x[(k+1)*Ts] = Phi*x[k*Ts] + Gamma*u[k*Ts]
%
%           The matrices Phi and Gamma are given by
%
%               Phi = exp(A*Ts)
%               Gamma = int_{0}^{Ts} exp(A*tau)*B dtau
%
%           Consequently, css2dsszoh may be regarded as
%           a function for computing these matrices/integrals
%
%
% Syntax:   [Phi,Gamma]=css2dsszoh(A,B,Ts)
%
%               A       :   Continuous-time matrix
%               B       :   Continuous-time matrix
%               Ts      :   Sampling time
%
%               Phi     :   Discrete-time matrix
%               Gamma   :   Discrete-time matrix
%
```

1.10.2 sisoctf2css

```
%
% sisoctf2css   Realizes a continuous-time state space system
%               of a continuous-time transfer function (SISO)
%
%               Let a SISO system be specified by the continuous-time transfer
%               function model
%
%                   y(s) = g(s)*u(s)
%                        = h(s)*exp(-lambda*s)*u(s)
%                        = h(s)*u(s-lambda)
%
%                   h(s) = num(s)/den(s)
%
%                   den(s) = a(0)*s^na + a(1)*s^(na-1) + ... + a(na-1)*s + a(na)
%                   num(s) = b(0)*s^nb + b(1)*s^(nb-1) + ... + b(nb-1)*s + b(nb)
%
%               The corresponding continuous-time state space realization
%               of this system is
%
%                  d/dt x(t) = A*x(t) + B*u(t-tau)
%                       y(t) = C*x(t) + D*u(t-tau)
%
%               sisoctf2css computes the matrices (A,B,C,D) and tau =
%               lambda.
```

```
%
%               The time delay must be non-negative and the transfer
%               function can only be realized if the denominator degree is
%               at least the same size as the numerator degree.
%
%               The model is realized in observer canonical form.
%
% Syntax: [A,B,C,D,tau]=sisoctf2css(num,den,lambda)
%
%           num         :   coefficients for the numerator in the transfer
%                           function
%           den         :   coefficients for the denominator in the transfer
%                           function
%           lambda      :   time delay (lambda >= 0)
%
%           (A,B,C,D)   :   state space matrices
%           tau         :   time delay (tau = lambda)
%
```

1.10.3 sisocss2dss

```
%
% sisocss2dss  Computes the discrete-time state space model of a
%              a continuous-time state space model with delay.
%
%              Continuous-time system
%
%                   d/dt x(t) = A*x(t) + B*u(t-tau)
%                        y(t) = C*x(t) + D*u(t-tau)
%
%               Zero-order-hold of the input
%
%                       u(t) = u(k*Ts)  k*Ts <= t < (k+1)*Ts
%
%               Discrete-time system
%
%                       x[(k+1)*Ts] = Ad*x[k*Ts] + Bd*u[(k-l)*Ts]
%                         y[k*Ts]   = Cd*x[k*Ts] + Dd*u[(k-l)*Ts]
%
%               The discrete time system matrices (Ad,Bd,Cd,Dd) are computed
%               assuming a zero-order-hold. l is the integer delay in the
%               discrete-time state space model
%
% Syntax: [Ad,Bd,Cd,Dd,l]=sisocss2dss(A,B,C,D,Ts,tau)
%
%           A,B,C,D     :   Continuous-time state space matrices
%           Ts          :   Sampling time
%           tau         :   Delay in continuous-time model
%
%           Ad,Bd,Cd,Dd :   Discrete-time state space matrices
%           l           :   Integer delay in discrete-time model
%
```

1.10.4 sisodss2dimpulse

```
%
% sisodss2dimpulse  Computes the discrete impulse response of a discrete-time state
%                   space system with delay (SISO).
%
%                   Discrete-time state space system with delay
%
%                       x[(k+1)*Ts] = A*x[k*Ts] + B*u[(k-l)*Ts]
%                          y[k*Ts]  = C*x[k*Ts] + D*u[(k-l)*Ts]
%
%                   The impulse response coefficients are
%
%                       h[k*Ts] = y[k*Ts]     k=0,1,...,N
%
%                   when
%                       x[0] = 0 and
%                       u[0] = 1 and
%                       u[k*Ts] = 0 for k > 0
%
%
% Syntax: [h,th] = sisodss2dimpulse(A,B,C,D,l,N,Ts)
%
%           A,B,C,D     :   Discrete-time state space matrices
%           l           :   Integer delay in discrete-time state space
%                           model
%           N           :   N*Ts is the final sampling time
%           Ts          :   Sampling time intervals
%
%           h           :   Impulse response coefficients
%           th          :   Corresponding time vector
%
```

1.10.5 sisodss2dstep

```
%
% sisodss2dstep     Computes the discrete-time step response of a discrete-time state
%                   space system with delay.
%
%
%                   Discrete-time state space system with delay
%
%                       x[(k+1)*Ts] = A*x[k*Ts] + B*u[(k-l)*Ts]
%                          y[k*Ts]  = C*x[k*Ts] + D*u[(k-l)*Ts]
%
%                   The step response coefficients are
%
%                       s[k*Ts] = y[k*Ts]     k=0,1,...,N
%
%                   when
%                       x[0] = 0
%                       u[k*Ts] = 1     k = 0,1, ..., N-1
%
%
% Syntax: [s,ts] = sisodss2dstep(A,B,C,D,l,N,Ts)
%
%           A,B,C,D     :   Discrete-time state space matrices
%           l           :   Integer delay in discrete-time state space
%                           model
%           N           :   N*Ts is the final sampling time
%           Ts          :   Sampling time intervals
%
%           s           :   Step response coefficients
%           ts          :   Corresponding time vector
%
```

1.10.6 sisoctf2dss

```
%
% sisoctf2dss   Computes the discrete-time state space model with
%               integer delay for a continuous-time transfer function
%               with delay (SISO)
%
%               Let a SISO system be specified by the continuous-time transfer
%               function model
%
%                   y(s) = g(s)*u(s)
%                        = h(s)*exp(-lambda*s)*u(s)
%                        = h(s)*u(s-lambda)
%
%                   h(s) = num(s)/den(s)
%
%                   den(s) = a(0)*s^na + a(1)*s^(na-1) + ... + a(na-1)*s + a(na)
%                   num(s) = b(0)*s^nb + b(1)*s^(nb-1) + ... + b(nb-1)*s + b(nb)
%
%               The time delay, lambda, must be non-negative and the transfer
%               function can only be realized if the denominator degree is
%               at least the same size as the numerator degree.
%
%               Zero-order-hold of the input (Ts: sampling time)
%
%                       u(t) = u(k*Ts)  k*Ts <= t < (k+1)*Ts
%
%               Discrete-time system
%
%                       x[(k+1)*Ts] = Ad*x[k*Ts] + Bd*u[(k-l)*Ts]
%                         y[k*Ts]   = Cd*x[k*Ts] + Dd*u[(k-l)*Ts]
%
%               The discrete time system matrices (Ad,Bd,Cd,Dd) are computed
%               assuming a zero-order-hold for the transfer function and sampling
%               time Ts. l is the integer delay in the discrete-time state
%               space model
%
%
%
% Syntax: [Ad,Bd,Cd,Dd,l]=sisoctf2dss(num,den,lambda,Ts)
%
%           num         :   coefficients for the numerator in the transfer
%                           function
%           den         :   coefficients for the denominator in the transfer
%                           function
%           lambda      :   time delay (lambda >= 0)
%
%           Ad,Bd,Cd,Dd :   Discrete-time state space matrices
%           l           :   Integer delay in discrete-time model
%
```

1.10.7 sisoctf2dimpulse

```
%
% sisoctf2dimpulse  The discrete time impulse response of a SISO
%                   continuous-time transfer function with delay
%
%                   Let a SISO system be specified by the continuous-time transfer
%                   function model
%
%                       y(s) = g(s)*u(s)
%                            = h(s)*exp(-lambda*s)*u(s)
%                            = h(s)*u(s-lambda)
%
%                       h(s) = num(s)/den(s)
%
%                       den(s) = a(0)*s^na + a(1)*s^(na-1) + ... + a(na-1)*s + a(na)
%                       num(s) = b(0)*s^nb + b(1)*s^(nb-1) + ... + b(nb-1)*s + b(nb)
%
%                   The time delay, lambda, must be non-negative and the transfer
%                   function can only be realized if the denominator degree is
%                   at least the same size as the numerator degree.
%
%                   Zero-order-hold of the input (Ts: sampling time)
%
%                       u(t) = u(k*Ts)  k*Ts <= t < (k+1)*Ts
%
%                   The discrete-time impulse response coefficients
%
%                       h(k)    k = 0,1, ..., N
%
%                   are computed at times
%
%                       th(k) = k*Ts
%
%                   assuming that u(t) is implemented as a zero-order-hold.
%
%
% Syntax: [h,th]=sisoctf2dimpulse(num,den,lambda,Ts,N)
%
%           num         :   coefficients for the numerator in the transfer
%                           function
%           den         :   coefficients for the denominator in the transfer
%                           function
%           lambda      :   time delay (lambda >= 0)
%
%           h           :   Discrete time impulse response coefficients
%           th          :   Corresponding time vector
%
```

1.10.8 sisoctf2dstep

```
%
% sisoctf2dstep     The discrete time step response of a SISO
%                   continuous-time transfer function with delay
%
%                   Let a SISO system be specified by the continuous-time transfer
%                   function model
%
%                       y(s) = g(s)*u(s)
%                            = h(s)*exp(-lambda*s)*u(s)
%                            = h(s)*u(s-lambda)
%
%                       h(s) = num(s)/den(s)
%
%                       den(s) = a(0)*s^na + a(1)*s^(na-1) + ... + a(na-1)*s + a(na)
%                       num(s) = b(0)*s^nb + b(1)*s^(nb-1) + ... + b(nb-1)*s + b(nb)
%
%                   The time delay, lambda, must be non-negative and the transfer
%                   function can only be realized if the denominator degree is
%                   at least the same size as the numerator degree.
%
%                   Zero-order-hold of the input (Ts: sampling time)
%
%                       u(t) = u(k*Ts)  k*Ts <= t < (k+1)*Ts
%
%                   The discrete-time step response coefficients
%
%                       s(k)    k = 0,1, ..., N
%
%                   are computed at times
%
%                       ts(k) = k*Ts
%
%                   assuming that u(t) is implemented as a zero-order-hold.
%
%
% Syntax: [s,ts]=sisoctf2dstep(num,den,lambda,Ts,N)
%
%           num         :   coefficients for the numerator in the transfer
%                           function
```

```
%           den         :   coefficients for the denominator in the transfer
%                           function
%           lambda      :   time delay (lambda >= 0)
%
%           s           :   Discrete time step response coefficients
%           ts          :   Corresponding time vector
%
```

1.10.9 mimoctf2dimpulse

```
%
% mimoctf2dimpulse  Computes the discrete-time impulse response
%                   of a continuous-time MIMO transfer function.
%
%                   Let a MIMO system (p outputs, m inputs) be specified by a continuous-time
%                   transfer function model
%
%                       Y(s) = G(s)*U(s)
%
%                       Y(s) = [y1(s) y2(s) ... yp(s)]'
%                       U(s) = [u1(s) u2(s) ... um(s)]'
%
%                              | g11(s) ... g1m(s) |
%                       G(s) = |   .         .     |
%                              | gp1(s) ... gpm(s) |
%
%                     The individual transfer functions are of the form
%
%                       yi(s) = gij(s)*uj(s)
%                             = hij(s)*exp(-tau_ij*s)*uj(s)
%                             = hij(s)*uj(s-tau_ij)
%
%                       hij(s) = num_ij(s)/den_ij(s)
%
%                       den_ij(s) = a_ij(0)*s^na_ij + ... + a_ij(na_ij-1)*s + a_ij(na_ij)
%                       num_ij(s) = b_ij(0)*s^nb_ij + ... + b_ij(nb_ij-1)*s + b_ij(nb_ij)
%
%                     The time delay, tau_ij, must be non-negative and the transfer
%                     function can only be realized if the denominator degree is
%                     at least the same size as the numerator degree.
%
%                     The MIMO discrete-time impulse response coefficients are computed
%                     by computing the discrete-time impulse response
%                     coefficients for each input-output combination
%                     individually assuming a zero-order-hold of the
%                     inputs. This method is efficient as SISO systems with
%                     delays tend to have low dimension and are easily
%                     realized.
%
%
% Syntax: [H,th]=mimoctf2dimpulse(num,den,tau,Ts,N)
%
%           num         :   p-times-m cell array with the numerator polynomials
%           den         :   p-times-m cell array with the denominator polynomials
%           tau         :   p-times-m matrix with the time delays
%           Ts          :   Sampling time period
%           N           :   Final sampling time (tfinal = N*Ts)
%
%           H           :   Impulse response matrices (p-times-m-times-(N+1))
%           th          :   Corresponding time vector
%
```

1.10.10 mimoctf2dstep

```
%
% mimoctf2dstep     Computes the discrete-time step response
%                   of a continuous-time MIMO transfer function.
%
%                   Let a MIMO system (p outputs, m inputs) be specified by a continuous-time
%                   transfer function model
%
%                       Y(s) = G(s)*U(s)
%
%                       Y(s) = [y1(s) y2(s) ... yp(s)]'
%                       U(s) = [u1(s) u2(s) ... um(s)]'
%
%                              | g11(s) ... g1m(s) |
%                       G(s) = |   .         .     |
%                              | gp1(s) ... gpm(s) |
%
%                     The individual transfer functions are of the form
%
%                       yi(s) = gij(s)*uj(s)
%                             = hij(s)*exp(-tau_ij*s)*uj(s)
%                             = hij(s)*uj(s-tau_ij)
%
%                       hij(s) = num_ij(s)/den_ij(s)
```

```
%
%                       den_ij(s) = a_ij(0)*s^na_ij + ... + a_ij(na_ij-1)*s + a_ij(na_ij)
%                       num_ij(s) = b_ij(0)*s^nb_ij + ... + b_ij(nb_ij-1)*s + b_ij(nb_ij)
%
%                     The time delay, tau_ij, must be non-negative and the transfer
%                     function can only be realized if the denominator degree is
%                     at least the same size as the numerator degree.
%
%                     The MIMO discrete-time step response coefficients are computed
%                     by computing the discrete-time step response
%                     coefficients for each input-output combination
%                     individually assuming a zero-order-hold of the
%                     inputs. This method is efficient as SISO systems with
%                     delays tend to have low dimension and are easily
%                     realized.
%
%
% Syntax: [S,ts]=mimoctf2dstep(num,den,tau,Ts,N)
%
%           num         :   p-times-m cell array with the numerator polynomials
%           den         :   p-times-m cell array with the denominator polynomials
%           tau         :   p-times-m matrix with the time delays
%           Ts          :   Sampling time period
%           N           :   Final sampling time (tfinal = N*Ts)
%
%           S           :   Step response matrices (p-times-m-times-(N+1))
%           ts          :   Corresponding time vector
%
```

1.10.11 mimodimpulse2dss

```
%
% mimodimpulse2dss  Computes a discrete-time state space model from
%                   discrete-time impulse response coefficients (Markov
%                   parameters) in the MIMO case.
%
%                   The discrete-time impulse response matrices of the MIMO
%                   system must be given as a (p-times-m-times-(N+1))
%                   matrix in which
%
%                       H(:,:,k)    is the impulse response matrix at time
%                                   k*Ts
%
%                   The discrete-time state space model is
%
%                       x[k+1] = Ad*x[k] + Bd*u[k]
%                         y[k] = Cd*x[k] + Dd*u[k]
%
%
%                   This model is realized by a balanced singular value
%                   decompositio of the Hankel matrix of the impulse
%                   response matrices. The number of impulse response
%                   matrices determines the computational load and also the
%                   maximum system order of the discrete time system that
%                   can be realized. Enough impulse response matrices must
%                   be included to determine the response of the system. If
%                   possible the number of impulse responses should be
%                   about 2 times the state dimension of the state space
%                   system.
%
%                   tol specifies the cut-off value of the singular values
%                   of the Hankel matrix of the impulse responses. These
%                   values, sH, are returned as well.
%
%                   The state space system realized is guaranteed to be
%                   both observable and controllable.
%
% Syntax: [Ad,Bd,Cd,Dd,sH]=mimodimpulse2dss(H,tol)
%
%               H           :   Sequence of impulse response matrices
%               tol         :   Hankel value tolerance cut-off
%
%               Ad,Bd,Cd,Dd :   Discrete-time system matrices
%               sH          :   Hankel matrix singular values
%
```

1.10.12 mimoctf2dss

```
%
% mimoctf2dss   Computes a discrete-time state space model from a
%               continuous-time transfer function model (MIMO).
%
%               Let a MIMO system (p outputs, m inputs) be specified by a continuous-time
%               transfer function model
%
%                       Y(s) = G(s)*U(s)
%
```

```
%                       Y(s) = [y1(s) y2(s) ... yp(s)]'
%                       U(s) = [u1(s) u2(s) ... um(s)]'
%
%                              | g11(s) ... g1m(s) |
%                       G(s) = |   .          .     |
%                              | gp1(s) ... gpm(s) |
%
%                 The individual transfer functions are of the form
%
%                       yi(s) = gij(s)*uj(s)
%                             = hij(s)*exp(-tau_ij*s)*uj(s)
%                             = hij(s)*uj(s-tau_ij)
%
%                       hij(s) = num_ij(s)/den_ij(s)
%
%                       den_ij(s) = a_ij(0)*s^na_ij + ... + a_ij(na_ij-1)*s + a_ij(na_ij)
%                       num_ij(s) = b_ij(0)*s^nb_ij + ... + b_ij(nb_ij-1)*s + b_ij(nb_ij)
%
%                  The time delay, tau_ij, must be non-negative and the transfer
%                  function can only be realized if the denominator degree is
%                  at least the same size as the numerator degree.
%
%                  Assuming zero-order-hold with sampling time Ts
%
%                       u(t) = u(k*Ts)       k*Ts <= t < (k+1)*Ts
%
%                  a discrete-time state space model
%
%                       x[(k+1)*Ts] = Ad*x[k*Ts] + Bd*u[k*Ts]
%                           y[k*Ts] = Cd*x[k*Ts] + Dd*u[k*Ts]
%
%                 is computed by mimoctf2dss.
%
%                 The realization is conducted by computing the impulse
%                 repsonse of the transfer function and doing a balanced
%                 realization from the Hankel matrix of the impulse response
%                 matrices. Nmax is the final impulse response matrix
%                 computed and used in the construction of the Hankel
%                 matrix. If Nmax >= 2*n, in which n is the state order of the
%                 true system, then the realization will be exact within the tolerance, tol.
%                 tol specifies the cut-off Hankel singular value
%                 used to determine the order of the model.
%
%
% Syntax: [Ad,Bd,Cd,Dd,sH]=mimoctf2dss(num,den,tau,Ts,Nmax,tol)
%
%           num              :   p-times-m cell array with the numerator polynomials
%           den              :   p-times-m cell array with the denominator polynomials
%           tau              :   p-times-m matrix with the time delays
%           Ts               :   Sampling time period
%           Nmax             :   Final sampling time (tfinal = Nmax*Ts)
%           tol              :   Hankel singular value cut off tolerance
%                                (gives the precision of the realization)
%
%           (Ad,Bd,Cd,Dd)    :   Discrete-time state space matrices
%           sH               :   Hankel singular values
%
```

1.10.13 mimodss2dimpulse

```
%
% mimodss2dimpulse  Computes the discrete-time impulse response matrices of
%                   a discrete-time MIMO state space system.
%
%                   Discrete-time system:
%
%                       x[k+1] = Ad*x[k] + Bd*u[k]
%                       y[k]   = Cd*x[k] + Dd*u[k]
%
%                   The impulse response matrices H(:,:,k) for this system are
%                   computed for k=0,1, ..., N.
%
%
% Syntax:   H = mimodss2dimpulse(Ad,Bd,Cd,Dd,N)
%
%           Ad,Bd,Cd,Dd :   Discrete-time state space matrices
%           N           :   Last sample to be included in the response
%
%           H           :   Impulse response matrices (p-times-m-times-(N+1))
%
```

1.10.14 mimodss2dstep

```
%
% mimodss2dstep     Computes the discrete-time step response matrices of
%                   a discrete-time MIMO state space system.
%
```

```
%                 Discrete-time system:
%
%                     x[k+1] = Ad*x[k] + Bd*u[k]
%                     y[k]   = Cd*x[k] + Dd*u[k]
%
%                 The step response matrices S(:,:,k) for this system are
%                 computed for k=0,1, ..., N.
%
%
% Syntax:   S = mimodss2dstep(Ad,Bd,Cd,Dd,N)
%
%           Ad,Bd,Cd,Dd :   Discrete-time state space matrices
%           N           :   Last sample to be included in the response
%
%           S           :   Step response matrices (p-times-m-times-(N+1))
%
```

1.11 Matlab Examples

Examples demonstrating the use of the realization algorithms in the Matlab part of the linear model predictive control toolbox are provided.

1.11.1 sisorealization

The file `sisorealization.m` contains the example code used to demonstrate the realization of SISO systems.

1.11.1.1 Case 1

This case demonstrates state space realization as well as computation of the impulse and step responses for the SISO transfer function

$$g(s) = \frac{2}{s+1}\, e^{-4s} \tag{1.105}$$

The transfer function is specified by the Matlab commands

```
%             2
%  g(s) = ------- exp(-4*s)
%          s + 1
num = 2;
den = [1 1];
lambda = 4;
```

The impulse and step responses are computed from 0 to 10 with intervals of 0.01 using the statements

```
Ts = 0.01;              % Sampling time (or plot interval)
tfinal = 10;            % Final plotting time
N = floor(tfinal/Ts);   % Last sample point computed

% Computes the impulse response
[h,th] = sisoctf2dimpulse(num,den,lambda,Ts,N);

% Computes the step response
[s,ts] = sisoctf2dstep(num,den,lambda,Ts,N);

% Plot the impulse response and the step response
```

```
figure(1)
subplot(211); plot(th,h); ylabel('Impulse response')
subplot(212); plot(ts,s); ylabel('Step response')
xlabel('time (min)')
```

The result is shown in figure 1.1.

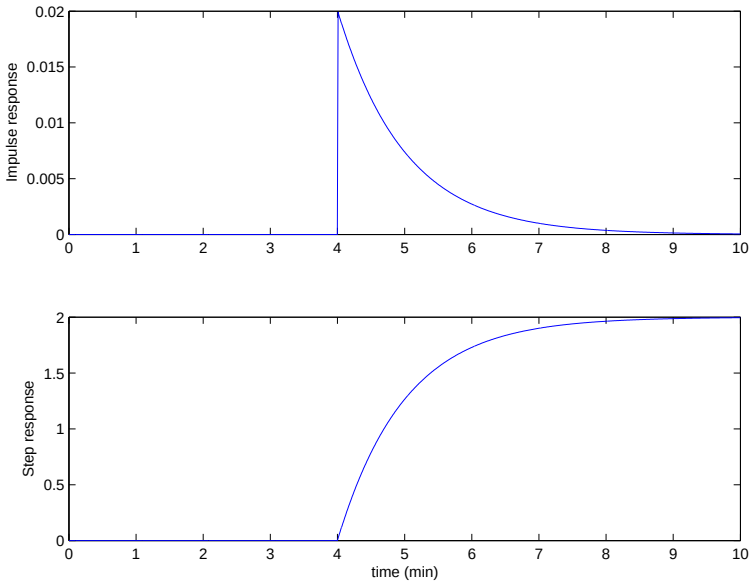

Figure 1.1. Impulse and step response for $g(s) = \frac{2}{s+1}\, e^{-4s}$

The continuous-time state space realization of the system is obtained as

```
%
% State space realization of transfer function with delay
%
[a,b,c,d,tau] = sisoctf2css(num,den,lambda)

a =
    -1

b =
     2

c =
     1

d =
     0

tau =
     4
```

and subsequently the discrete-time state space system with delay is obtained as

```
[ad,bd,cd,dd,l] = sisocss2dss(a,b,c,d,Ts,tau)

ad =
    0.9900

bd =
    0.0199

cd =
     1

dd =
     0

l =
   400
```

Note that the discrete-time delay is $l = 400$ as the continuous-time delay is $\lambda = 4$ and the sampling time is $T_s = 0.01$. A standard discrete-time state space realization of this system would give a representation with 401 states, while the representation with delay has only 1 state. The standard realization using `mimoctf2dss` yields

```
%
% Realization as a standard discrete time state space model
%
tol = 1.0e-12;
Nmax = 2*(lambda/Ts + length(den)-1);
[Ad,Bd,Cd,Dd]=mimoctf2dss({num},{den},lambda,Ts,Nmax,tol);
size(Ad,2)

ans =
   401
```

Note that {`num`} and {`den`} is used to transform the `num` and `den` matrices to cell arrays which is required by `mimoctf2dss`. The above standard discrete-time state space realization of the transfer function cannot be recommended.

1.11.1.2 Case 2

This case serves to demonstrate specification of transfer functions that are products of polynomials. The Matlab built-in command `conv` is used to multiply (convolute) polynomials.

Consider the transfer function

$$g(s) = \frac{s-1}{(s+1)(s+3)}\, e^{-4s} \tag{1.106}$$

which is specified by

```
%             (s-1)
% g1(s) = ------------ exp(-4*s)
%         (s+1)*(s+3)
```

```
num1 = [1 -1];
den1 = conv([1 1],[1 3]);
lambda1 = 4;
```

The impulse and step responses are computed by

```
Ts1 = 0.01;
tfinal1 = 10;
N1 = floor(tfinal1/Ts1);

[h1,th1] = sisoctf2dimpulse(num1,den1,lambda1,Ts1,N1);
[s1,ts1] = sisoctf2dstep(num1,den1,lambda1,Ts1,N1);

figure(2)
subplot(211); plot(th1,h1); ylabel('Impulse response')
subplot(212); plot(ts1,s1); ylabel('Step response')
xlabel('time (min)')
```

and the result is shown in figure 1.2.

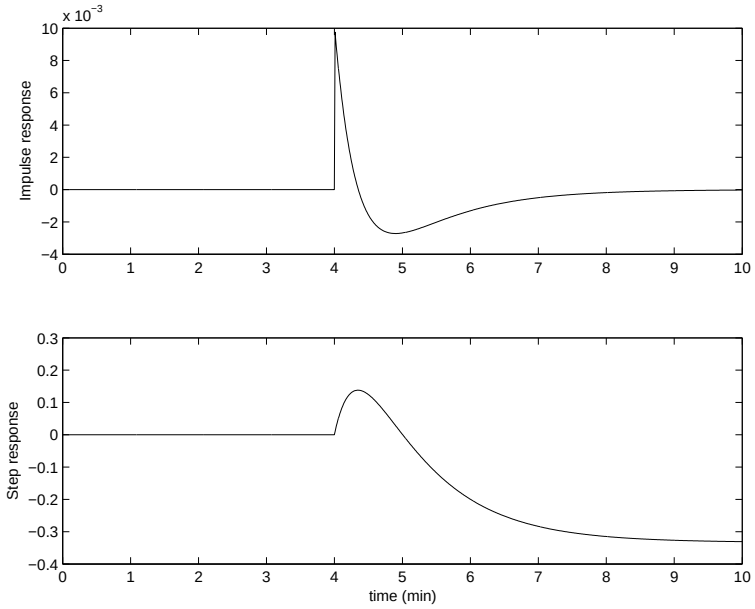

Figure 1.2. Impulse and step response for $g(s) = \frac{s-1}{(s+1)(s+3)}\, e^{-4s}$

1.11.1.3 Case 3

The transfer function

$$g(s) = e^{-4s} = \frac{1}{1}\, e^{-4s} \tag{1.107}$$

is specified by the commands

```
%
% g2(s) = exp(-4*s)
```

```
%
num2 = 1;
den2 = 1;
lambda2 = 4;
```

1.11.1.4 Case 4

The transfer function

$$g(s) = 1 = \frac{1}{1}\, e^{-0s} \tag{1.108}$$

is specified by the commands

```
%
% g3(s) = 1
%
num3 = 1;
den3 = 1;
lambda3 = 0;
```

1.11.2 mimorealization

The file `mimorealization.m` contains the code demonstrating realization of the continuous-time MIMO transfer function

$$G(s) = \begin{bmatrix} \frac{-0.2}{(15s+1)^2} & \frac{-0.2}{(15s+1)^2} \\ \frac{6}{(15s+1)^2}\, e^{-15s} & \frac{6.5}{(14s+1)^2} \\ \frac{23}{(15s+1)^2}\, e^{-15s} & \frac{33}{(12s+1)^2} \\ 0 & \frac{0.95}{(3s+1)^2}\, e^{-2s} \end{bmatrix} \tag{1.109}$$

Note that each individual SISO transfer function may be represented as

$$g_{j,i}(s) = \frac{K_{j,i}}{(T_{j,i}s + 1)^2}\, e^{-\tau_{j,i}s} \tag{1.110}$$

The specification of the transfer function, $G(s)$, is accomplished by the commands

```
%
% Specify the system. Each SISO input-output function
% has the form g(s) = K/(T1+1)^2 * exp(-tau*s)
%
p = 4;          % number of outputs
m = 2;          % number of inputs

tau = zeros(p,m);
T1 = zeros(p,m);
K = zeros(p,m);

K = [-0.2 -0.2;6 6.5; 23 33;0 0.95]
T1 = [15 15;15 14;15 12;0 3]
tau = [0 0; 15 0; 15 0;0 2]
```

```
%
% Compute cell arrays containing the transfer functions
%
num = cell(p,m);
den = cell(p,m);

for i=1:m
    for j=1:p
        num(j,i) = {K(j,i)};
        den(j,i) = {conv([T1(j,i) 1],[T1(j,i) 1])};
    end
end
```

The transfer function is realized as a balanced discrete-time state space system with sampling time $T_s = 1.0$ using `mimoctf2dss`

```
%
% Compute a state space realization
%
Ts = 1.0;       % sampling time
nmax = 100;     % number of impulse response matrices computed for the realization
tol = 1.0e-6;   % tolerance (Hankel singular value cut-off)

[Ad,Bd,Cd,Dd]=mimoctf2dss(num,den,tau,Ts,nmax,tol);
```

The realized system has 25 states.

The impulse responses of the original system and the realized state space system is computed and plotted by the following commands

```
%
% Compute the impulse response of the system
%

N = 150;                                    % Number of impulse response matrices
H1 = mimoctf2dimpulse(num,den,tau,Ts,N);  % Exact impulse response
H2 = mimodss2dimpulse(Ad,Bd,Cd,Dd,N);     % Impulse response of approximate state space mode

figure(1)                                   % Plot the results (blue = exact, red = approx.)
hh1 = zeros(N+1,1);
hh2 = zeros(N+1,1);
for i=1:m
    for j=1:p
        for k=1:N+1
            hh1(k) = H1(j,i,k);
            hh2(k) = H2(j,i,k);
        end
        subplot(p,m,(j-1)*m+i)
        hold on;
        plot(hh1,'b-')
        plot(hh2,'r-');
        hold off
    end
end
```

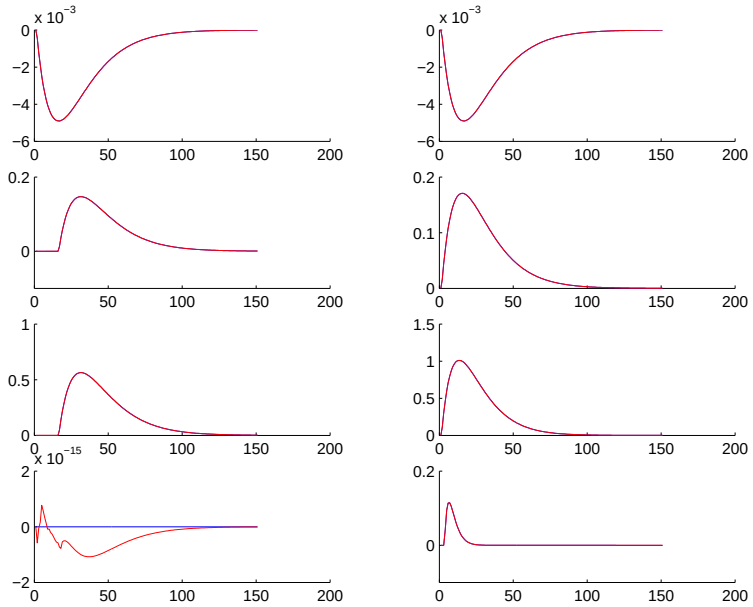

Figure 1.3. Impulse response of the exact system and the approximate system.

and the result is shown in figure 1.3.

The step responses of the original system and the realized state space system is computed and plotted by the following commands

```
%
% Compute the step response of the system
%

N = 150;                                    % Number of impulse response matrices
S1 = mimoctf2dstep(num,den,tau,Ts,N);       % Exact step response
S2 = mimodss2dstep(Ad,Bd,Cd,Dd,N);          % Step response of approximate state space model

figure(2)                                   % Plot the results (blue = exact, red = approx.)
ss1 = zeros(N+1,1);
ss2 = zeros(N+1,1);
for i=1:m
    for j=1:p
        for k=1:N+1
            ss1(k) = S1(j,i,k);
            ss2(k) = S2(j,i,k);
        end
        subplot(p,m,(j-1)*m+i)
        hold on;
        plot(ss1,'b-')
        plot(ss2,'r-');
        hold off
    end
end
```

and the result is shown in figure 1.4.

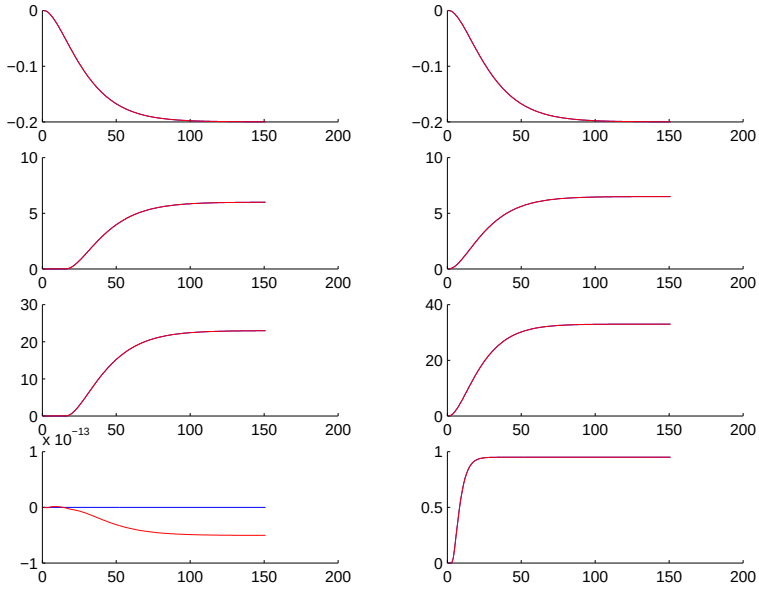

Figure 1.4. Step response of the exact system and the approximate system.

1.11.3 Realization of the Shell Standard Control Problem

The Shell standard control problem involves the MIMO transfer function

$$G(s) = \begin{bmatrix} \frac{4.05}{50s+1}e^{-27s} & \frac{1.77}{60s+1}e^{-28s} & \frac{5.88}{50s+1}e^{-27s} & \frac{1.20}{45s+1}e^{-27s} & \frac{1.44}{40s+1}e^{-27s} \\ \frac{5.39}{50s+1}e^{-18s} & \frac{5.72}{60s+1}e^{-14s} & \frac{6.90}{40s+1}e^{-15s} & \frac{1.52}{25s+1}e^{-15s} & \frac{1.83}{20s+1}e^{-15s} \\ \frac{3.66}{9s+1}e^{-18s} & \frac{1.65}{30s+1}e^{-20s} & \frac{5.53}{40s+1}e^{-2s} & \frac{1.16}{11s+1} & \frac{1.27}{6s+1} \\ \frac{5.92}{12s+1}e^{-11s} & \frac{2.54}{27s+1}e^{-12s} & \frac{8.10}{20s+1}e^{-2s} & \frac{1.73}{5s+1} & \frac{1.79}{19s+1} \\ \frac{4.13}{8s+1}e^{-5s} & \frac{2.38}{19s+1}e^{-7s} & \frac{6.23}{10s+1}e^{-2s} & \frac{1.31}{2s+1} & \frac{1.26}{22s+1} \\ \frac{4.06}{13s+1}e^{-8s} & \frac{4.18}{33s+1}e^{-4s} & \frac{6.53}{9s+1}e^{-1s} & \frac{1.19}{19s+1} & \frac{1.17}{24s+1} \\ \frac{4.38}{33s+1}e^{-20s} & \frac{4.42}{44s+1}e^{-22s} & \frac{7.20}{19s+1} & \frac{1.14}{27s+1} & \frac{1.26}{32s+1} \end{bmatrix}$$

The file `ShellStandardControlProblem.m` contains script code illustrating how this is accomplished using the linear model predictive control toolbox. Notice that each individual input-output transfer function has the form

$$g(s) = \frac{K}{Ts+1}e^{-\tau s} \tag{1.111}$$

The transfer function, $G(s)$, is described by the commands

```
%
% Definition of the system by specification of gain, time contant, and time
% delay for each transfer function
%

K = [...
```

```
    4.05 1.77 5.88 1.20 1.44;...
    5.39 5.72 6.90 1.52 1.83;...
    3.66 1.65 5.53 1.16 1.27;...
    5.92 2.54 8.10 1.73 1.79;...
    4.13 2.38 6.23 1.31 1.26;...
    4.06 4.18 6.53 1.19 1.17;...
    4.38 4.42 7.20 1.14 1.26]

T = [...
    50 60 50 45 40;...
    50 60 40 25 20;...
     9 30 40 11  6;...
    12 27 20  5 19;...
     8 19 10  2 22;...
    13 33  9 19 24;...
    33 44 19 27 32]

tau = [...
      27 28 27 27 27;...
      18 14 15 15 15;...
       2 20  2  0  0;...
      11 12  2  0  0;...
       5  7  2  0  0;...
       8  4  1  0  0;...
       20 22 0  0  0]

%
% Compute the transfer functions
%
[p,m]=size(K);
num = cell(p,m);
den = cell(p,m);

for i=1:m
    for j=1:p
        num(j,i) = {K(j,i)};
        den(j,i) = {[T(j,i) 1]};
    end
end
```

A balanced discrete-time state space system with sampling time $T_s = 1$ is constructed by the commands

```
%
% Compute a discrete-time state space realization
%
Ts = 1.0;          % Sampling time
nmax = 100;        % Maximum impulse responses used in state space realization
tol = 1.0e-6;      % tolerance
tf = 300;          % final time for step response simulation

tic
[Ad,Bd,Cd,Dd,sH]=mimoctf2dss(num,den,tau,Ts,nmax,tol);
toc
```

The Hankel singular values are plotted by the commands

```
%
% Plot the Hankel Singular Values
%
figure(1)
plot(log10(sH))
title('Hankel Singular Values')
xlabel('State dimension, n')
ylabel('log10(Hankel Singular Value)')
```

and the result is shown in figure 1.5. The Hankel singular values is a measure

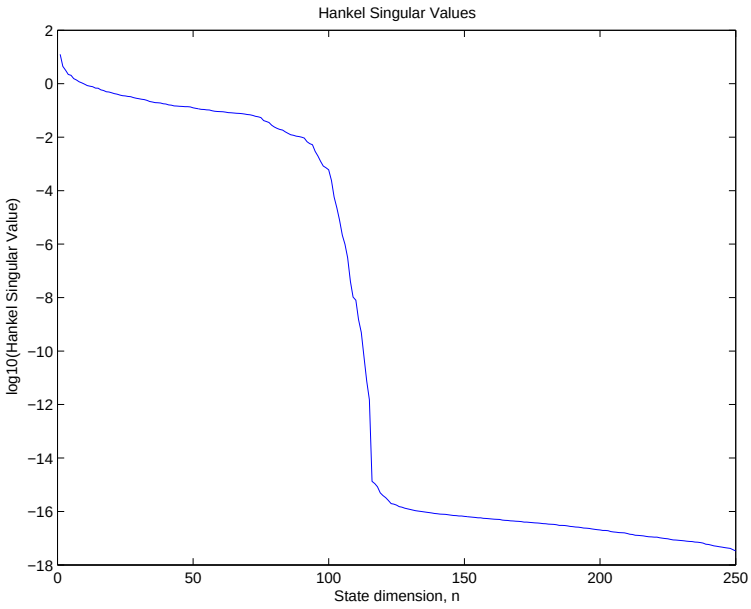

Figure 1.5. Hankel singular values as function of number of states included in the model.

of the accuracy of the realized discrete-time state space model. It is the Hankel singular values that are controlled by the tolerance specification. `tol = 1.0e-6` gives a model with $n = 105$ states.

The step response of the original transfer function and the realized approximate state space system are generated by the commands

```
%
% Simulate step response of the true system
% and the realized (approximative) system
%

N = floor(tf/Ts);
[S1,ts]=mimoctf2dstep(num,den,tau,Ts,N);    % true system (red)
n1=size(Ad,1);
S2=mimodss2dstep(Ad,Bd,Cd,Dd,N);            % approx. system (blue)
```

and the responses are plotted by

```
% Plot the computed step responses
```

```
figure(2)
ss1 = zeros(N+1,1);
ss2 = zeros(N+1,1);
for i=1:m
    j=1;
        for k=1:N+1
            ss1(k) = S1(j,i,k);
            ss2(k) = S2(j,i,k);
        end
        subplot(p,m,(j-1)*m+i)
        title(strcat('u[',int2str(i),']'))
        hold on;
        plot(ts,ss1,'r-')
        plot(ts,ss2,'b-');
        hold off

    for j=2:p
        for k=1:N+1
            ss1(k) = S1(j,i,k);
            ss2(k) = S2(j,i,k);
        end
        subplot(p,m,(j-1)*m+i)
        hold on;
        plot(ts,ss1,'r-');  % true system
        plot(ts,ss2,'b-');  % approximative system
        hold off
    end
end

i=1;
for j=1:p
    subplot(p,m,(j-1)*m+i);
    hold on;
    ylabel(strcat('y[',int2str(j),']'));
    hold off;
end
```

The result is plotted in figure 1.6.

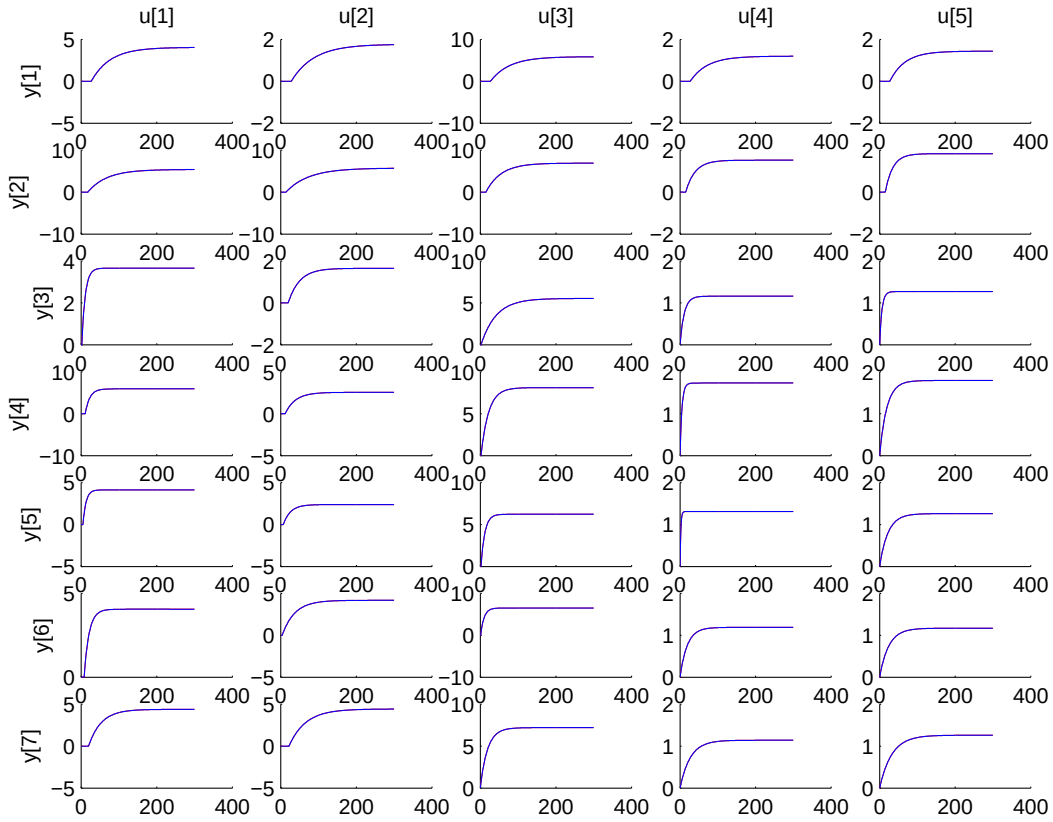

Figure 1.6. Shell step responses for the exact and the approximative model (identical).

References

Backx, T. (1987). *Identification of an Industrial Process: A Markov Parameter Approach*. Ph.D. thesis, Technical University of Eindhoven, The Netherlands.

Chen, C.-T. (1999). *Linear System Theory and Design*. Oxford University Press, Oxford.

de Jong, L. S. (1978). Numerical Aspects of Recursive Realization Algorithms. *SIAM Journal of Control and Optimization*, **16**, 646–659.

Ho, B. L. and Kalman, R. E. (1966). Efficient Construction of Linear State Variable Models from Input/Output Functions. *Regelungstechnik*, **14**, 545–548.

Kalman, R. E.; Falb, P. L. and Arbib, M. A. (1968). *Topics in Mathematical System Theory*. McGraw-Hill, New York.

Moler, C. and Van Loan, C. (1978). Nineteen Dubious Ways to Compute the Exponential of a Matrix. *SIAM Review*, **20**, 801–836.

Sidje, R. B. (1998). Expokit: A Software Package for Computing Matrix Exponentials. *ACM Transactions on Mathematical Software*, **24**, 130–156.

Silverman, L. (1971). Realization of Linear Dynamical Systems. *IEEE Transactions on Automatic Control*, **AC-16**, 554–567.

Van Loan, C. F. (1978). Computing Integrals Involving the Matrix Exponential. *IEEE Transactions on Automatic Control*, **23**, 395–404.