Control Systems Design Tools: Creating and Working with System Objects

Introduction

Maple provides a series of controls systems design tools that give you the ability to work analytically with linear time-invariant dynamic systems. The *DynamicSystems* package is a collection of procedures for creating, manipulating, simulating, and plotting linear time-invariant systems models. In this Tips and Techniques, you will learn how linear systems are modeled using the *DynamicSystems* package, how to create *System Objects*, and how to transform your models between a variety of different representations.

You can get detailed information on this package by entering ?*DynamicSystems*.

Creating System Objects

The *DynamicSystems* package allows you to work with both continuous and discrete system objects representations. All analysis in the *DynamicSystems* package is done on a *System Object*, which is a data structure providing a unified representation of a linear system model. These may be created and defined in several ways, including differential equations, transfer functions, state space matrices, and zero-pole-gain representations, and may be transformed from one form to another.

The following model types are supported:

Frequency-Based Models (use transfer functions to express the relationships between inputs and outputs)	**Time-Based Models** (use *diff-equations* to express the relationships between inputs and outputs)
• Transfer Function • Zero-Pole-Gain • Coefficients (of the numerators and denominators of transfer functions)	• Diff-Equation (differential or difference equations) • State-Space • AE-Equation (algebraic equations)

Example: Creating a System Object from Differential Equations

This example deals with a system representing an electric motor, defined by two differential equations.

Before creating a system object, you need to load the *DynamicSystems* package. You can do this using the *with*( ) command:

```
with(DynamicSystems)
```

```
[AlgEquation, BodePlot, CharacteristicPolynomial, Chirp, Coefficients,
    ControllabilityMatrix, Controllable, DiffEquation, DiscretePlot, FrequencyResponse,
    GainMargin, Grammians, ImpulseResponse, ImpulseResponsePlot, IsSystem,
    MagnitudePlot, NewSystem, ObservabilityMatrix, Observable, PhaseMargin,
    PhasePlot, PrintSystem, Ramp, ResponsePlot, RootContourPlot, RootLocusPlot,
    RouthTable, SSModelReduction, SSTransformation, Simulate, Sinc, Sine, Square,
    StateSpace, Step, System, SystemOptions, ToDiscrete, TransferFunction, Triangle,
    Verify, ZeroPoleGain, ZeroPolePlot]
```

(2.1.1)

First, define the two differential equations that govern the behavior:

$$deq := \left[L\cdot\left(\frac{d}{dt}\,i(t)\right)+R\cdot i(t)=v(t)-K\cdot\left(\frac{d}{dt}\,\theta(t)\right), J\cdot\left(\frac{d^2}{dt^2}\,\theta(t)\right)+b\cdot\left(\frac{d}{dt}\,\theta(t)\right)=K\cdot i(t)\right]$$

$$\left[L\left(\frac{d}{dt}\,i(t)\right)+R\,i(t)=v(t)-K\left(\frac{d}{dt}\,\theta(t)\right), J\left(\frac{d^2}{dt^2}\,\theta(t)\right)+b\left(\frac{d}{dt}\,\theta(t)\right)=K\,i(t)\right] \quad \textbf{(2.1.2)}$$

...and the corresponding parameters:

$$params := [J=0.01, K=0.01, L=0.5, R=1, b=0.1]$$

$$[J=0.01, K=0.01, L=0.5, R=1, b=0.1] \quad \textbf{(2.1.3)}$$

You can now create a system object based on these differential equations:

$$sys := DiffEquation(deq, [v(t)], [\theta(t), i(t)])$$

```
Diff. Equation
continuous
2 output(s); 1 input(s)
inputvariable = [v(t)]
outputvariable = [θ(t), i(t)]
```

(2.1.4)

The first argument is the list of differential equations, which is followed by a list of the input and output variables. In this case the input is the voltage applied, and the two outputs are the angular position of the shaft and the current drawn.

You can review the system object properties with the *PrintSystem* command:

```
PrintSystem(sys)
```

Diff. Equation

continuous

2 output(s); 1 input(s)

inputvariable = $[v(t)]$

outputvariable = $[\theta(t), i(t)]$

$de = [L\,\dot{i}(t) + R\,i(t) = v(t) - K\,\dot{\theta}(t), J\,\ddot{\theta}(t) + b\,\dot{\theta}(t) = K\,i(t)]$ **(2.1.5)**

Creating Different System Object Representations

The following table lists commands you can use to create different types of system objects:

Command	What it does	Example
DiffEquation( )	Creates a *diff-equation* system object. The time-domain behavior of the object is modeled by differential or difference equations, depending whether the system is *continuous* or *discrete*, respectively.	*sys1* := *DiffEquation*$\left(\frac{s}{s^3 + 5\,s^2 + 7\,s + 6}\right)$: *PrintSystem*(*sys1*) **Diff. Equation** continuous 1 output(s); 1 input(s) inputvariable = $[u1(t)]$ outputvariable = $[y1(t)]$ $de = [\dot{x1}(t) = x2(t), \dot{x2}(t) = x3(t), \dot{x3}(t) = -6\,x1(t) - 7\,x2(t) - 5\,x3(t) + u1(t), y1(t) = x2(t)]$ **(2.2.1)**
TransferFunctio\ n( )	Creates a *transfer function* system object. Expresses the transfer function between a given input and output as an explicit rational polynomial of the frequency variable.	*sys2* := *TransferFunction*([1, 2], [1, 2, 3]) : *PrintSystem*(*sys2*) **Transfer Function** continuous 1 output(s); 1 input(s) inputvariable = $[u1(s)]$ outputvariable = $[y1(s)]$ $tf_{1,1} = \frac{s + 2}{s^2 + 2\,s + 3}$ **(2.2.2)**
StateSpace( )	Creates a *state-space* system object.	

		The time-domain behavior of the object is modeled by four state-space Matrices, A, B, C, and D, that describe the input, output, and state equations.	$sys3 := StateSpace\left(\frac{s-3}{s^2+2s-4}\right):$ $PrintSystem(sys3)$ **State Space** continuous 1 output(s); 1 input(s); 2 state(s) inputvariable = $[u1(t)]$ outputvariable = $[y1(t)]$ statevariable = $[x1(t), x2(t)]$ $a = \begin{bmatrix} 0 & 1 \\ 4 & -2 \end{bmatrix}$ $b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $c = \begin{bmatrix} -3 & 1 \end{bmatrix}$ $d = \begin{bmatrix} 0 \end{bmatrix}$ **(2.2.3)**
	ZeroPoleGain( )	Creates a *zero-pole-gain* system object. Expresses the transfer function between a given input and output as a list of zeros (roots of the numerator), a list of poles (roots of the denominator), and a scaling factor.	$sys4 := ZeroPoleGain([1, 2], [1, 2, 3]):$ $PrintSystem(sys4)$ **Zero Pole Gain** continuous 1 output(s); 1 input(s) inputvariable = $[u1(s)]$ outputvariable = $[y1(s)]$ $z_{1,1} = [-2]$ $p_{1,1} = [-1 - I\sqrt{2}, -1 + I\sqrt{2}]$ $k_{1,1} = 1$ **(2.2.4)**
	Coefficients( )	Creates a *coefficients* transfer system object. Expresses the transfer function between a given input and output as a list of the coefficients of the numerator and a list of the	$sys5 := Coefficients\left(\frac{s}{s^3+5s^2+7s+6}\right):$ $PrintSystem(sys5)$

	coefficients of the denominator of the rational polynomial.	**Coefficients** continuous 1 output(s); 1 input(s) inputvariable = [$u1(s)$] outputvariable = [$y1(s)$] $num_{1,1}$ = [1, 0] $den_{1,1}$ = [1, 5, 7, 6]	**(2.2.5)**
AlgEquation( )	Creates an *algebraic-equation* system object (i.e. a system whose input/output equations are given by non-differential algebraic equations). It is provided to permit code generation of algebraic-equation blocks. ***Note:*** *This object cannot be used by many of the commands in the DynamicSystems package.*	*sys6* := *AlgEquation*($y = \sin(x), x, y$) : `Warning, making the following substitutions: x = x(t), y = y(t).` *PrintSystem*(*sys6*) **Algebraic Equation** continuous 1 output(s); 1 input(s) inputvariable = [$x(t)$] outputvariable = [$y(t)$] ae = [$y(t) = \sin(x(t))$]	**(2.2.6)**

Tip: A variety of styles of arguments may be used in the above commands. You may, for example, create a TransferFunction system object using a transfer function as the argument:

sys7 := *TransferFunction*$\left(\frac{1}{s^3 + 4s^2 - s + 2}\right)$:

PrintSystem(*sys7*)

Transfer Function
continuous
1 output(s); 1 input(s)
inputvariable = [$u1(s)$]
outputvariable = [$y1(s)$]
$tf_{1,1} = \frac{1}{s^3 + 4s^2 - s + 2}$

(2.2.7)

... but this is not necessary. You may use transfer functions, differential equations, lists of coefficients, and so on as arguments for any of the above commands (except AlgEquation which

will only accept algebraic equations).

For example, you can use a transfer function as an argument when creating a DiffEquation system object:

$$sys8 := DiffEquation\left(\frac{1}{s^3 + 4\,s^2 - s + 2} \right) :$$

$PrintSystem(sys8)$

Diff. Equation

continuous

1 output(s); 1 input(s)

inputvariable = $[u1(t)]$

outputvariable = $[y1(t)]$

$$de = [\dot{x1}(t) = x2(t), \dot{x2}(t) = x3(t), \dot{x3}(t) = -2\,x1(t) + x2(t) - 4\,x3(t) + u1(t), y1(t) = x1(t)]$$

(2.2.8)

... or coefficients as the argument when creating a StateSpace system object:

$sys9 := StateSpace([1, 2], [1, 2, 3]) :$
$PrintSystem(sys9)$

State Space

continuous

1 output(s); 1 input(s); 2 state(s)

inputvariable = $[u1(t)]$

outputvariable = $[y1(t)]$

statevariable = $[x1(t), x2(t)]$

$$a = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$c = \begin{bmatrix} 2 & 1 \end{bmatrix}$$

$$d = \begin{bmatrix} 0 \end{bmatrix}$$

(2.2.9)

Incidentally, notice that the system objects above, *sys7* and *sys8*, are essentially transformations of each other, as the same transformation was used as an argument for each. The next section will explain further how to directly convert between different system object representations.

See the Help pages for more details on using each specific command.

Converting between System Object Representations

You can also easily convert system objects to different representations. For example, you can convert from the DiffEquation system object in the example:

PrintSystem(*sys*)

Diff. Equation

continuous

2 output(s); 1 input(s)

inputvariable = $[v(t)]$

outputvariable = $[\theta(t), i(t)]$

de = $[L\,\dot{i}(t) + R\,i(t) = v(t) - K\,\dot{\theta}(t),\ J\,\ddot{\theta}(t) + b\,\dot{\theta}(t) = K\,i(t)]$

(3.1)

...to a State Space system object:

sys_ss := *StateSpace*(*sys*)

State Space

continuous

2 output(s); 1 input(s); 3 state(s)

inputvariable = $[v(t)]$

outputvariable = $[\theta(t), i(t)]$

statevariable = $[x1(t), x2(t), x3(t)]$

(3.2)

Again, use the *PrintSystem* command to view the detailed representation:
PrintSystem(*sys_ss*)

State Space

continuous

2 output(s); 1 input(s); 3 state(s)

inputvariable = $[v(t)]$

outputvariable = $[\theta(t), i(t)]$

statevariable = $[x1(t), x2(t), x3(t)]$

$$a = \begin{bmatrix} -\frac{R}{L} & 0 & -\frac{K}{L} \\ 0 & 0 & 1 \\ \frac{K}{J} & 0 & -\frac{b}{J} \end{bmatrix}$$

(3.3)

$$b = \begin{bmatrix} \frac{1}{L} \\ 0 \\ 0 \end{bmatrix}$$

$$c = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Similarly, you can convert to a Transfer Function representation:

$sys\_tf := TransferFunction(sys)$

Transfer Function

continuous

2 output(s); 1 input(s)

(3.4)

inputvariable = $[v(s)]$

outputvariable = $[\theta(s), i(s)]$

... and print the details:

$PrintSystem(sys\_tf)$

Transfer Function

continuous

2 output(s); 1 input(s)

inputvariable = $[v(s)]$

outputvariable = $[\theta(s), i(s)]$ **(3.5)**

$$\mathrm{tf}_{1,1} = \frac{K}{JLs^3 + (bL + JR)s^2 + (bR + K^2)s}$$

$$\mathrm{tf}_{2,1} = \frac{Js + b}{JLs^2 + (bL + JR)s + bR + K^2}$$

Tip: You can do the conversions whether the system is symbolic or fully parametrized.

Additional Tools

The *DynamicSystems* package also contains a number of plotting, analysis, and simulation tools. Standard plotting tools, such as Bode and Root-Locus plots, are available and can be used to plot both continuous and discrete system objects. The *DynamicSystem* System Manipulation Tools provide a mechanism for the advanced analysis of systems, including stability, observability, controllability, and sensitivity. System objects may be simulated using the *DynamicSystem* Simulation Tools to obtain frequency, impulse and transient response.

For more information, please see ?*DynamicSystems*.