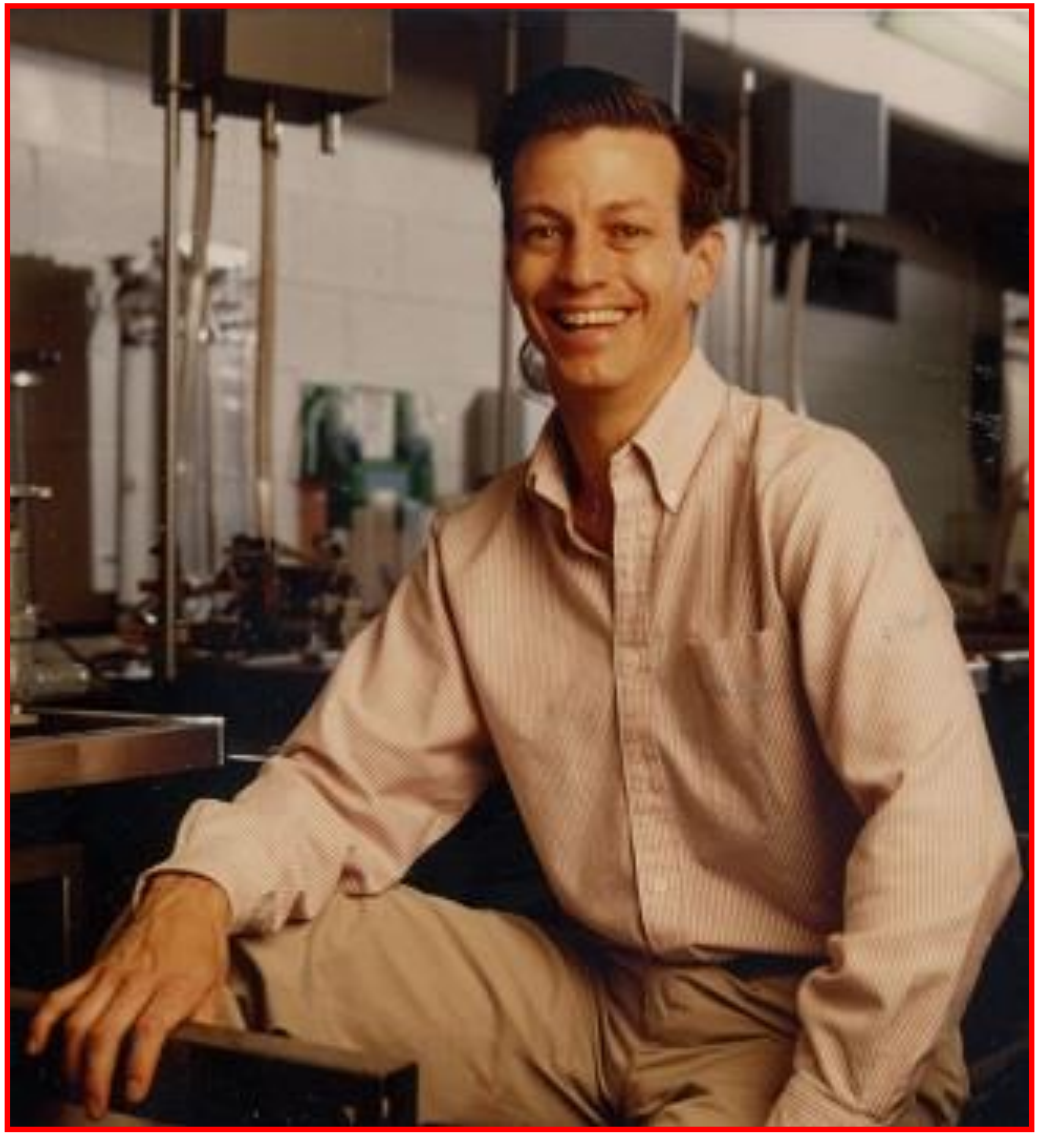
LPVTools: A Toolbox for Modeling, Analysis, and Synthesis of Parameter Varying Control

Arnar Hjartarson, Andrew Packard, and Peter Seiler

International Workshop in Robust Modeling, Design & Analysis
September 18, 2017

Gary J. Balas (Sept. 27. 1960 – Nov. 12, 2014)



Gary and Andy Packard

Spreading the Word

MUSYN Robust Control Theory Short Course (Start: 1989)

ROBUST

MULTIVARIABLE

CONTROL:

THEORY

AND

APPLICATION

USING μ-TOOLS

AUGUST 4 - 7

MUSYN

MUSYN is pleased to announce the latest short course in robust multivariable control design. A detailed, four day instructional workshop will be taught August 4-7 by three researchers in the field: Prof. John C. Doyle, Prof. Andy Packard and Prof. Gary J. Balas. The short course provides the attendees with an introduction to robust multivariable control using H_∞ and μ analysis and design techniques.

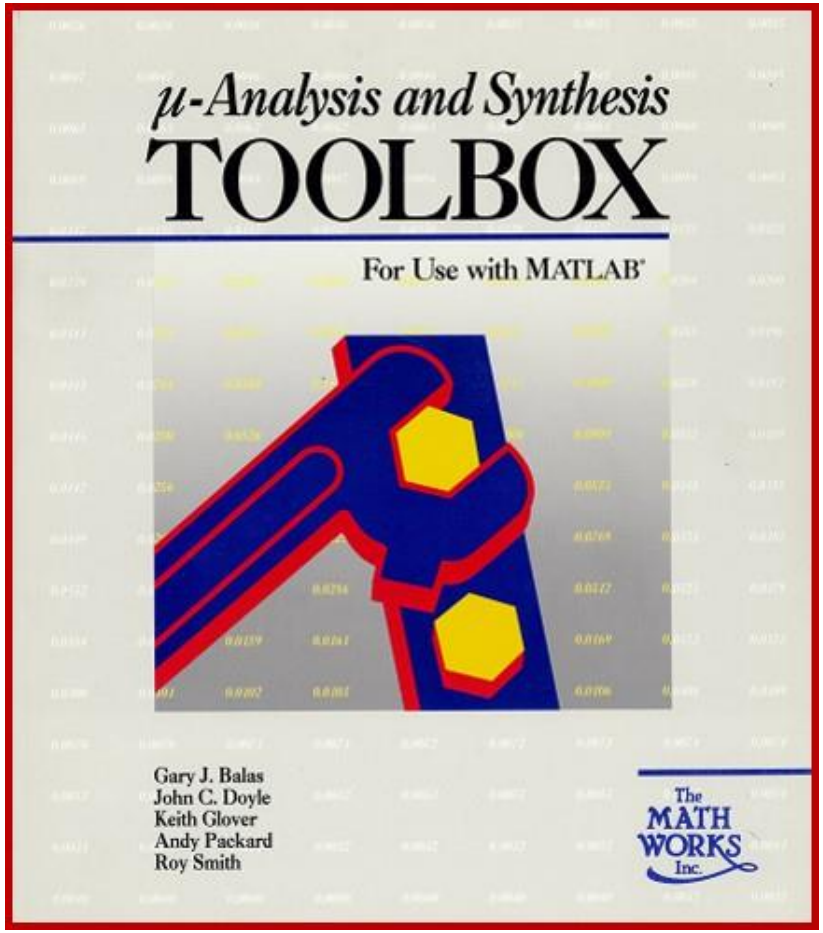
In the past three years over 200 people from industry, government laboratories and academia have attended this course. Locations have included: Los Angeles, Minneapolis, NASA Langley Research Center, Cambridge University, and Delft University, The Netherlands.

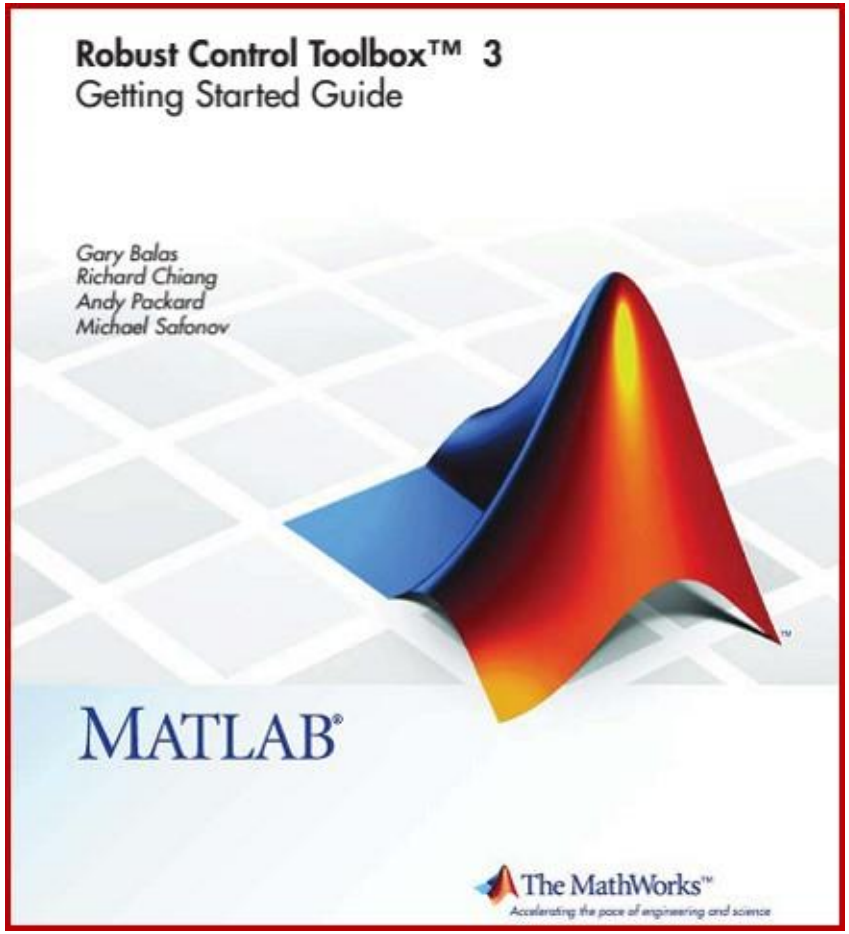
The course has been updated to reflect the latest advances in theory and software. The course covers: various models of uncertainty for components, motivation of "structured uncertainty models," analysis of effects of structured uncertainty using the structured singular value (μ), real/complex μ analysis, controller design using H_∞ and μ techniques, and example applications. Theoretical understanding of the subject material as well as its application to practical problems is emphasized.

Participants will learn and use the μ-Analysis and Synthesis Toolbox (μ-Tools) control design package in conjunction with MATLAB to apply the course material to application areas which include: flight control systems for advanced aircraft, space shuttle lateral axis control system, and vibration attenuation of flexible structures. Each application lecture will discuss modeling of the physical system, formulation of the control problem, application of μ and H_∞ techniques and corresponding results. The participants will have an opportunity to analyze and design control laws for each example with the μ-Tools software following the talks.

Software Development

μ-Analysis and Synthesis (μ-Tools) Matlab Toolbox (1990)





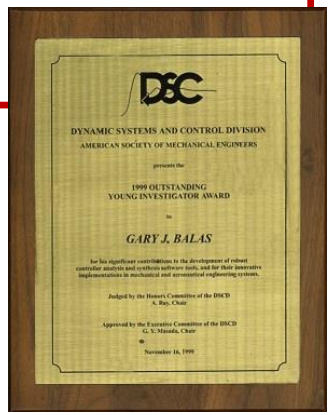
μ-Tools merged with the Matlab Robust Control Toolbox (2004)

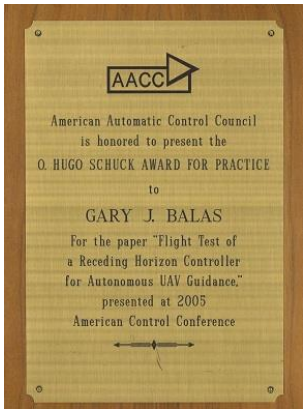
Awards

Honors and Awards

2012	Honorary Member, Hungarian Academy of Engineering
2012	Plenary Speaker, 7th IFAC Symposium on Robust Control Design, Aalborg, Denmark
2010	Prize for the Development of the Hungarian Aeronautical Science, Hungarian Scientific Association for Transport
2010	Plenary Speaker, 2nd Workshop on Clearance of Flight Control Laws, Stockholm, Sweden
2009	Plenary Speaker, 49th Israel Annual Conference on Aerospace Sciences, Tel Aviv and Haifa, Israel
2007	Distinguished McKnight University Professor, University of Minnesota
2006	O. Hugo Schuck Best Paper Award, American Automatic Control Council (with T. Keviczky)
2005	Control Systems Technology Award, IEEE Control System Society (with Prof. A.K. Packard)
2005-2006	Fellow, Committee on Institutional Cooperation Academic Leadership Program
2005	Semi-Plenary Speaker, 16th International Federation of Automatic Control (IFAC) World Congress, Prague, Czech Republic (with Prof. J. Bokor)
2004	Fellow, IEEE
2004	Plenary Speaker, Technical University of Delft Center for Systems and Control, "Challenges for the 21st Century"
2003	Institute of Technology George Taylor Distinguished Research Award, University of Minnesota
2003	Semi-Plenary Speaker, European Control Conference, Cambridge, England
2002	Associate Fellow, AIAA
2002-04	Senior Member, IEEE
2002	Session Plenary Speaker, International Council of Aeronautical Sciences Conference, Toronto
1999	Outstanding Young Investigator Award, ASME Dynamic Systems and Control
1993-1995	McKnight-Land Grant Professorship, University of Minnesota
1989-90, 2002	American Control Conference Best Paper Presentation in Session
1986-89	NASA Graduate Student Fellowship
1986	Donald Wills Douglas Fellowship in Aeronautics
1982-84	Hughes Aircraft Graduate Student Fellowship
1980-82	Hughes Aircraft Undergraduate Student Fellowship

Schuck Award w/ T. Keviczky ('05 ACC)





Head of Aerospace Eng. & Mechanics (7/06-1/14)

On Behalf of
The Minneapolis Chapter of the American Institute of Architects,
Preserve Minneapolis,
and
The Minneapolis Heritage Preservation Commission,

This Award is Presented in Recognition of

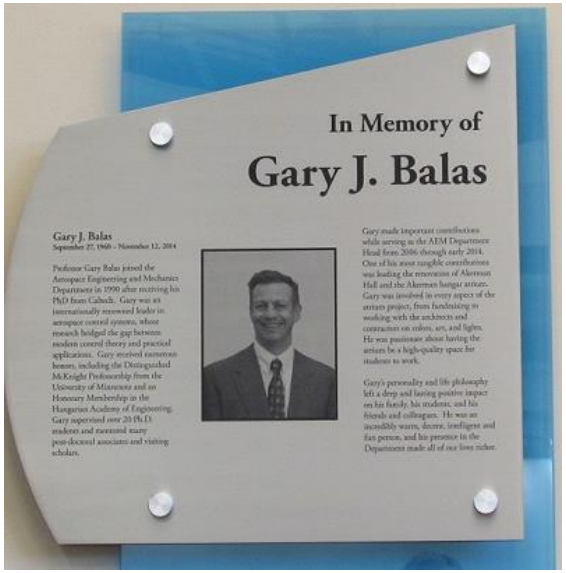
Akerman Hall Hangar Renovation

This project is a meritorious example of an adaptively reused historic building, transformed and reinvented to incorporate both new and old elements in a way that that is respectful of, but different from, the form of the past.

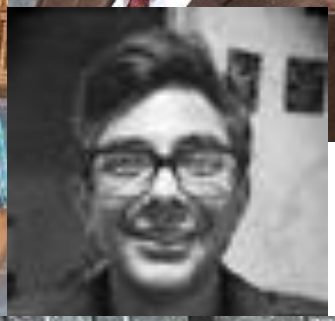
2011 Minneapolis Heritage Preservation Awards

Chair — President — Board President

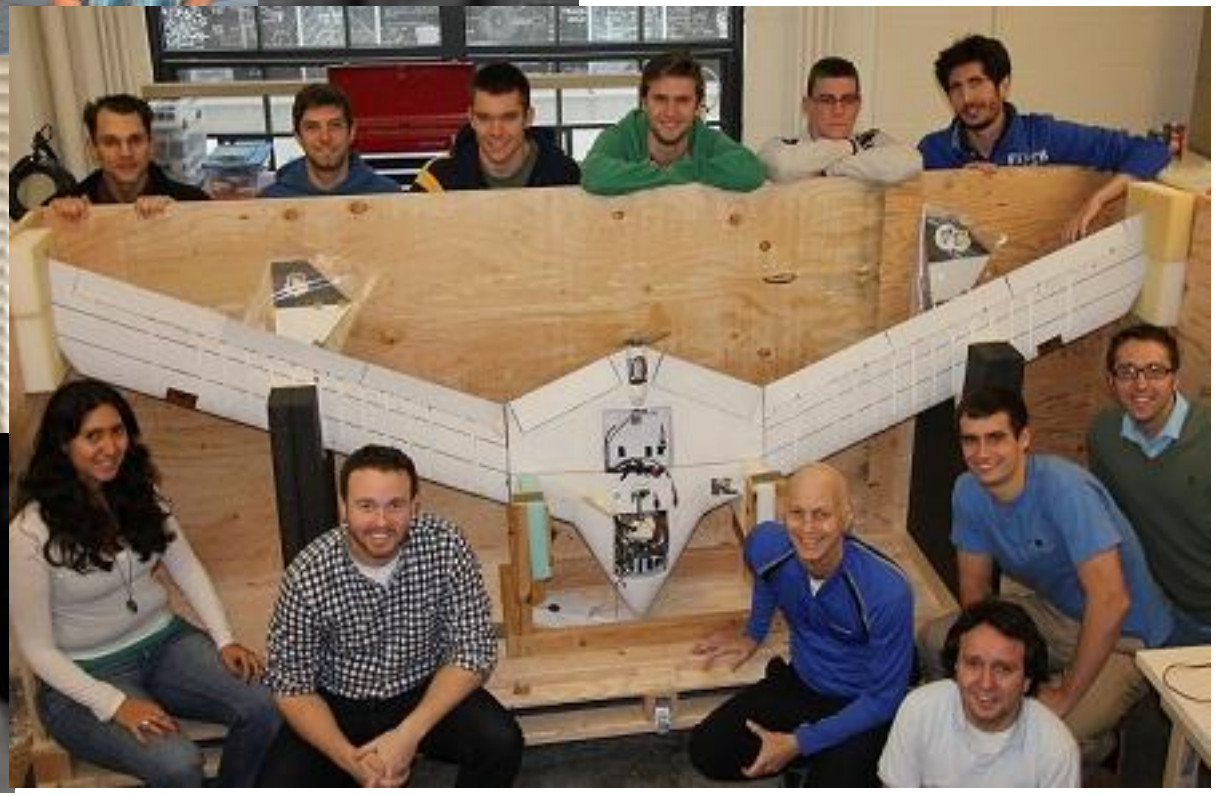
Minneapolis Heritage Preservation Commission — AIA Minneapolis — preserve minneapolis



Students & Visitors



and many others…



Collaborations

Enjoying Conferences

Biking….All year round in Minnesota!

LPVTools: Matlab Toolbox for LPV Systems

- Developed by MuSyn: Balas, Packard, Seiler, Hjartarson
  - Funded by NASA SBIR contract #NNX12CA14C
  - Contract Monitor: Dr. M.J. Brenner, NASA Armstrong.
- Goal: Unified framework for grid/LFT based LPV
  - Modeling
  - Synthesis
  - Analysis
  - Simulation
- MATLAB/Simulink integration
  - Compatible with Control Toolbox, Robust Control Toolbox, Simulink.
  - Uses MATLAB object-oriented class programming

(A Subset of) LPV Software Tools

- LFT
  - SMAC, LFR, LFRT-SLK, and Robust Feedforward Design Toolboxes (ONERA: Magni, Biannic, Roos, Ferreres, Demourant,…)
  - Enhanced LFR-toolbox (DLR: Hecker, Varga, Pfifer,…)
  - LPV Robust Control Toolbox (Milan: De Vito, Lovera; NGC Aerospace: Kron, de Lafontaine)
  - LFR-RAI (Siena: Garulli, Masi , Paoletti, Türkoğlu)
  - LPV Analysis & Synthesis (Stuttgart: Scherer, Veenman, Köse, Köroğlu,…)
- Grid-based
  - LMI Control Toolbox, HINFSTRUCT, Simulink LPV Blocks (Matlab)
- Polytopic
  - TP Toolbox (Sztaki: Baranyi, Takarics,…)

Classes of LPV Models

LPV systems depend on a time varying parameter *ρ(t)*

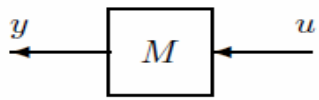
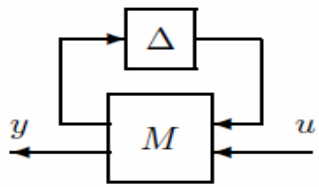
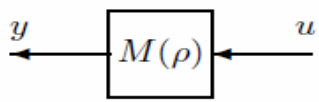
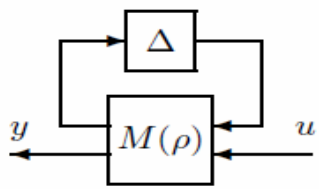
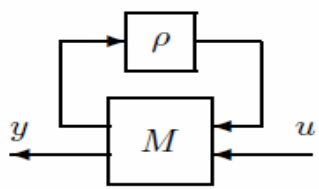
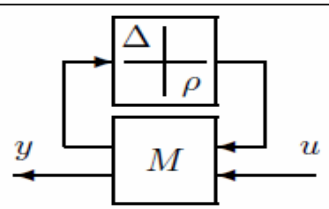
$$\begin{bmatrix} \dot{x}(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} A(\rho(t)) & B(\rho(t)) \\ C(\rho(t)) & D(\rho(t)) \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}$$

Three main classes of LPV systems

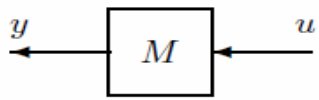
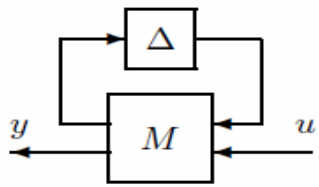
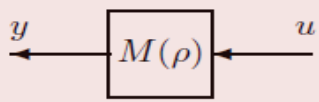
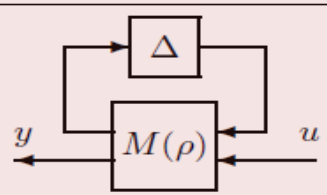
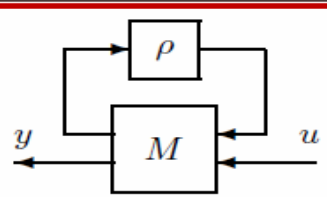
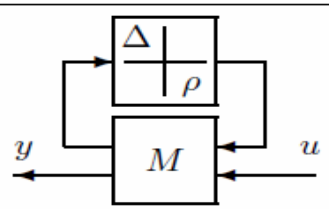
- **Grid-based** (Jacobian Linearization) Models
  - A(*ρ*), B(*ρ*), C(*ρ*), and D(*ρ*) are arbitrary functions of *ρ.*
  - State matrices defined on a grid of parameter values ρ_k
- **Linear Fractional Transformation** (LFT) Models
  - A(*ρ*), B(*ρ*), C(*ρ*), and D(*ρ*) are rational functions of *ρ.*
- **Polytopic** Models
  - A(*ρ*), B(*ρ*), C(*ρ*), and D(*ρ*) are polytopic functions of *ρ.*
  - Affine models as a special case

Modeling in LPVTools

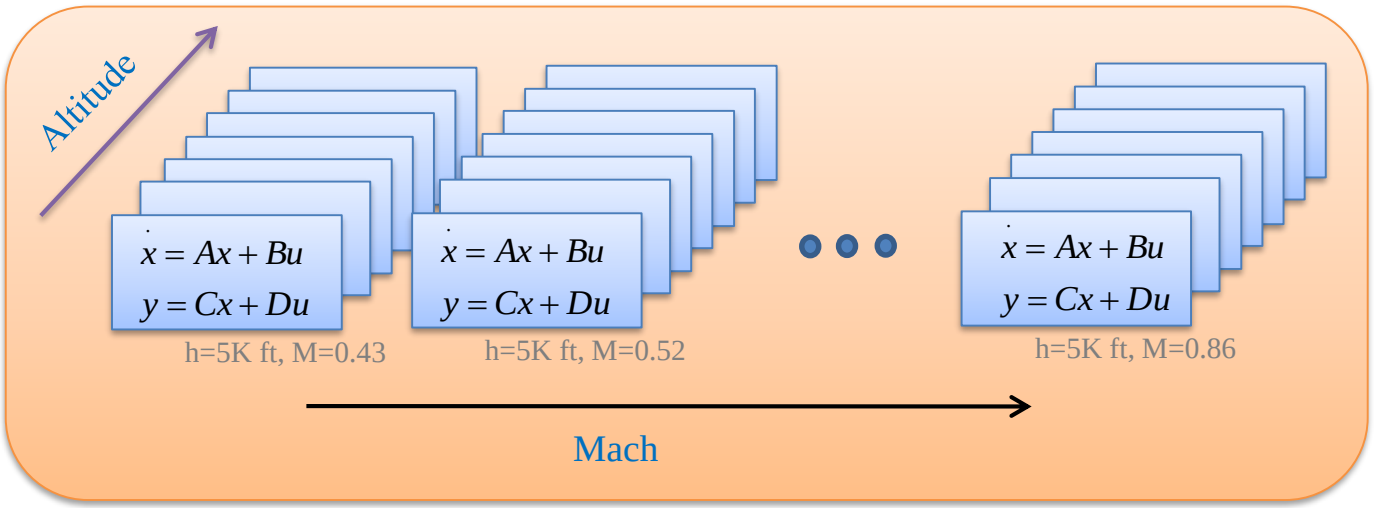
Data Structures for LPV/LFT Systems

Object Type	Block	Matrix	System	Frequency Response
Nominal		`double`	`ss`	`frd`
Uncertain		`umat`	`uss`	`ufrd`
Nominal Gridded LPV		`pmat`	`pss`	`pfrd`
Uncertain Gridded LPV		`upmat`	`upss`	`upfrd`
Nominal LFT LPV		`pmatlft`	`psslft`	
Uncertain LFT LPV		`pmatlft`	`psslft`	

Data Structures for LPV/LFT Systems

Object Type	Block	Matrix	System	Frequency Response
Nominal		`double`	`ss`	`frd`
Uncertain		`umat`	`uss`	`ufrd`
Nominal Gridded LPV		`pmat`	`pss`	`pfrd`
Uncertain Gridded LPV		`upmat`	`upss`	`upfrd`
Nominal LFT LPV		`pmatlft`	`psslft`	
Uncertain LFT LPV		`pmatlft`	`psslft`	

Grid-Based LPV Systems



Matrices or systems defined on a grid of parameter values

- Arbitrary parameter dependence
- A(ρ_k), B(ρ_k), C(ρ_k), and D(ρ_k) defined at each ρ_k.
- Analogous to gain-scheduled control framework

PGRID: Real parameter defined on a grid of points

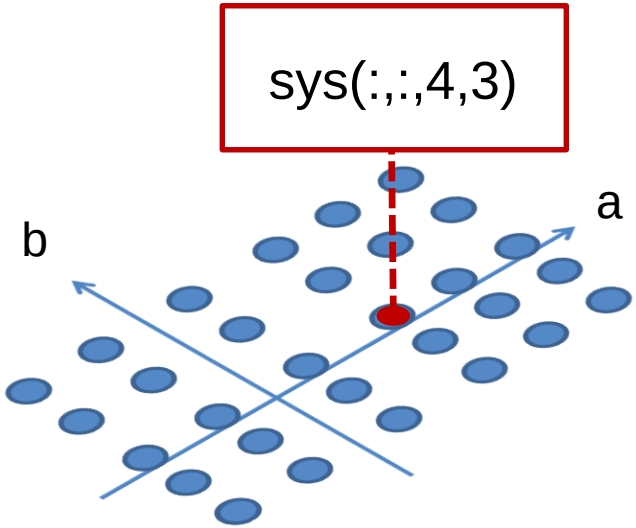
```
% Create a PGRID object
>> x = pgrid('x', -3:3 )
Gridded parameter "x", 7 points in [-3,3], rate bounds [-Inf,Inf].
>> class(x)
ans = pgrid
```

PGRID atom objects

```
% Change Name
>> x.Name = 'z'
Gridded parameter "z", 7 points in [-3,3], rate bounds [-Inf,Inf].

% PGRIDs can be used to represent time-varying parameters
% with specified rate bounds
>> y = pgrid('y', linspace(0,pi,10), [-3 3])
Gridded parameter "y", 10 points in [0,3.14], rate bounds [-3,3].

% Change RateBounds
>> y.RateBounds = [-1 1]
Gridded parameter "y", 10 points in [0,3.14], rate bounds [-1,1].
```

PSS: Parameter-varying state-space model

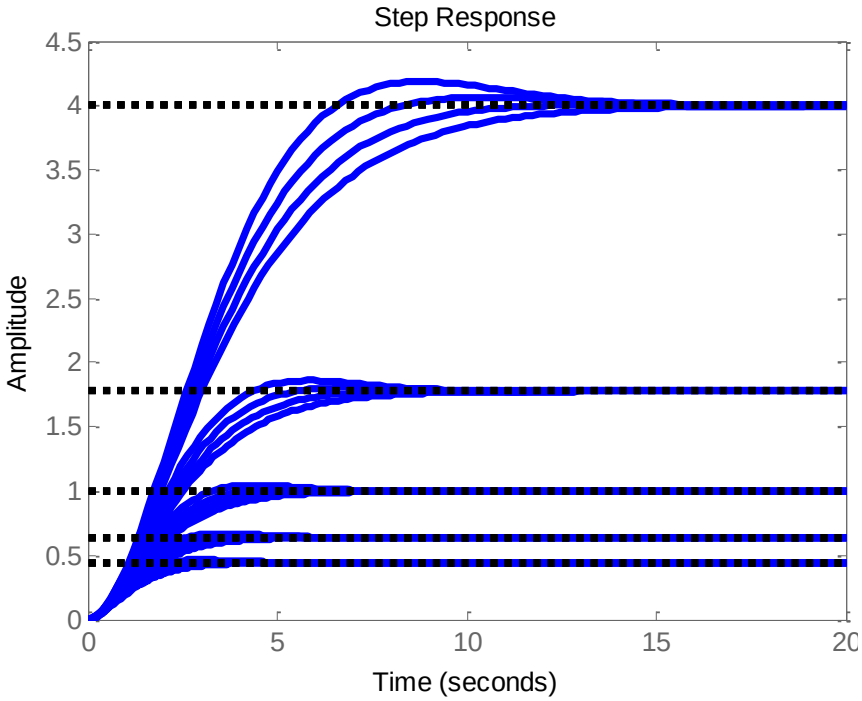
```
% Use PGRIDs to generate a PSS
>> a = pgrid('a', linspace(-3,-2,6) );
>> b = pgrid('b', linspace(2,3,5) );
>> sys = ss(a,b,1,0)
PSS with 1 States, 1 Outputs, 1 Inputs, Continuous System.
The PSS consists of the following blocks:
  a: Gridded real, 6 points in [-3,-2], rate bounds [-Inf,Inf].
  b: Gridded real, 5 points in [2,3], rate bounds [-Inf,Inf].
```



PSS

"Pointwise" control calculations

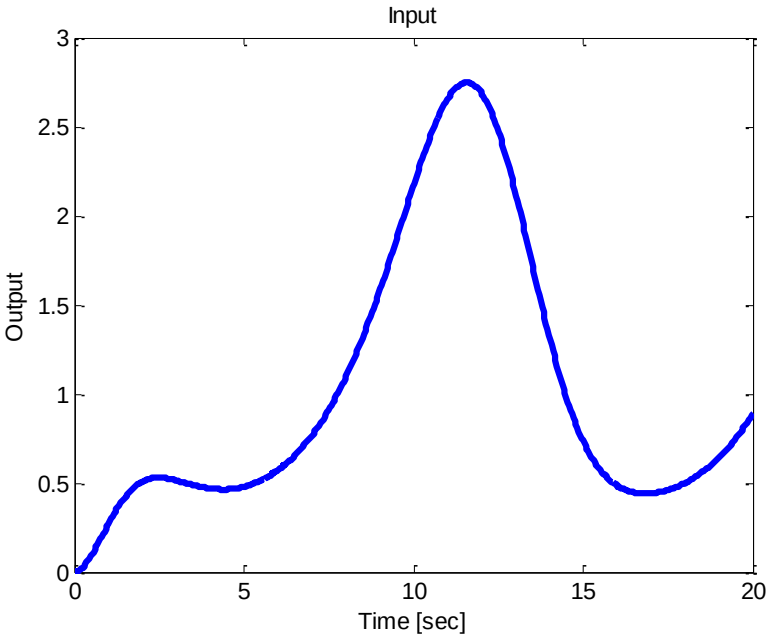
```
% Unit step response at each point
% in the parameter grid
>> wn = pgrid('wn',0.5:0.25:1.5)
>> zeta = pgrid('zeta',0.7:0.1:1);
>> G = ss([0 1; -wn^2 -2*zeta*wn],...
        [0;1],[1 0],0);
>> step(G,20)
```




```
% Compute Hinf norm at each point
>> n = norm(G,inf)
PMAT with 1 rows and 1 columns.
The PMAT consists of the following blocks:
  wn: Gridded real, 5 points in [0.5,1.5], rate bounds [-Inf,Inf].
  zeta: Gridded real, 4 points in [0.7,1], rate bounds [-Inf,Inf].

% Find peak Hinf over domain
>> lpvmax( norm(G,inf) )
PMAT with 1 rows and 1 columns.
ans =
     4
```

"LPV" calculations

```
% Unit step response along a
% specified parameter trajectory
>> wn = pgrid('wn',0.5:0.25:1.5)
>> zeta = pgrid('zeta',0.7:0.1:1);
>> G = ss([0 1; -wn^2 -2*zeta*wn],...
        [0;1],[1 0],0);

>> ptraj.time = linspace(0,20,500);
>> ptraj.zeta = repmat(0.8,[500 1]);
>> ptraj.wn = 1+0.5*sin(ptraj.time/2);
>> lpvstep(G,ptraj)
```




```
% Bound induced L2 norm over allowable (rate unbounded) trajectories
>> lpvnorm(G)
ans =
    7.9801

% Compare with point-wise analysis
>> lpvmax( norm(G,inf) )
ans =
     4
```

Sample of pointwise functions

balreal	hinfsyn
bode	lft
connect	loopmargin
damp	loopsens
dcgain	lqr
impulse	lsim
feedback	margin
gapmetric	ncfsyn
gram	pole
grid2lft	step
h2syn	sysic

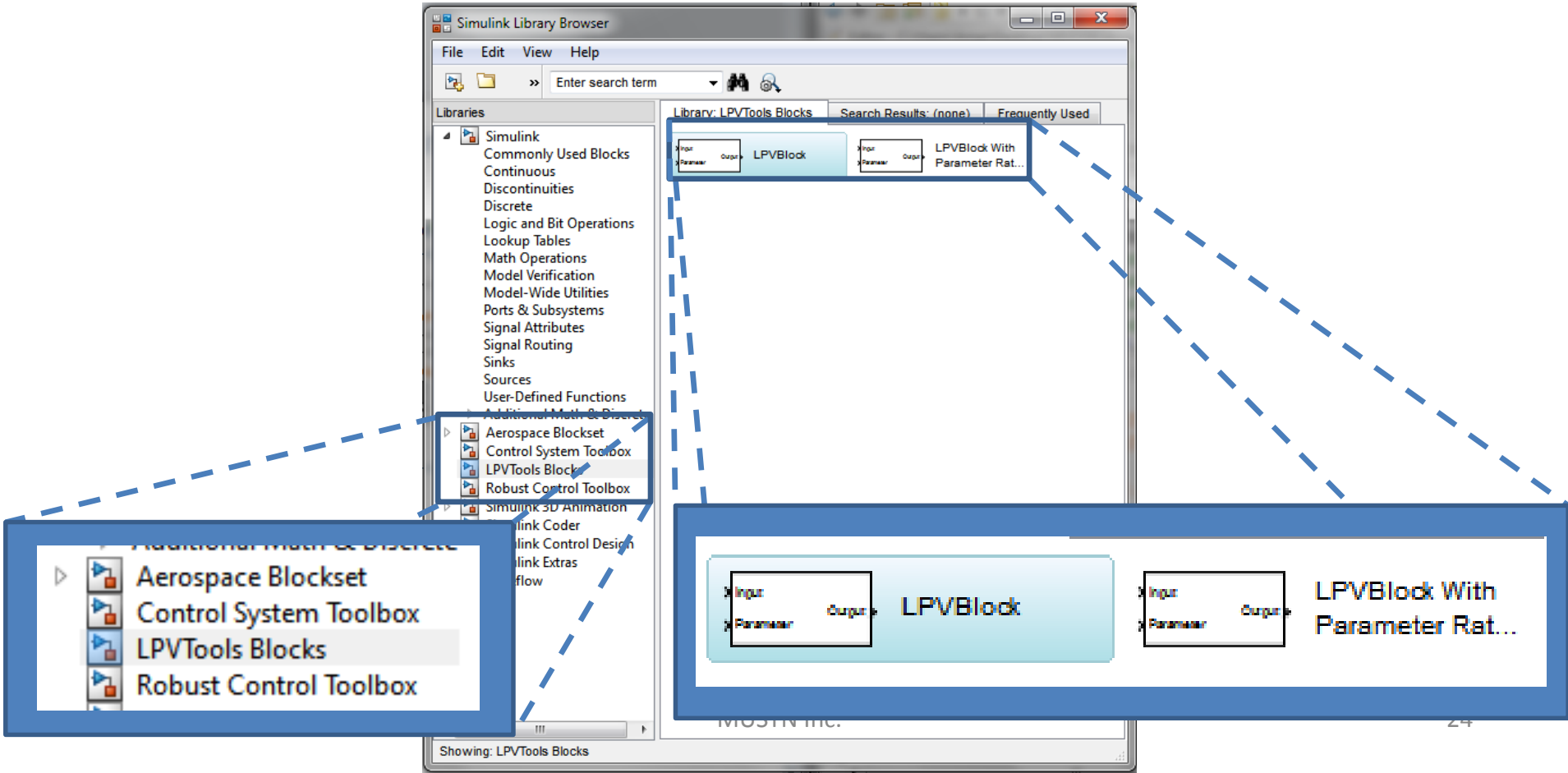
Sample of LPV functions

lpvbalance	lpvinitial
lpvbalancmr	lpvlsim
lpvbalreal	lpvnorm
lpvestsyn	lpvsfsyn
lpvncfsyn	lpvstep
lpvgram	lpvsyn
lpvimpulse	lpvwcgain

LPV Software – Simulink

Simulink Integration

- Library blocks to simulate grid- and LFT-based models.
- Fast c-code



Uncertain LPV Systems

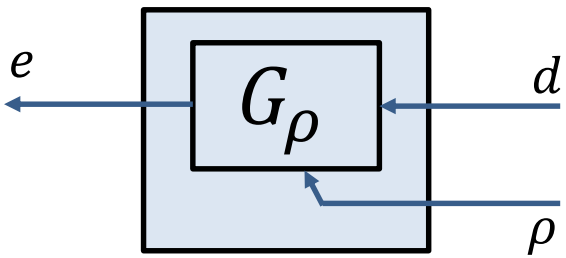
- Toolbox has objects to model uncertain LPV systems
  - Uncertain grid-based: upmat, upss
  - Uncertain LFT-based: plftmat, plftss
- Analysis and synthesis capabilities
  - Toolbox analysis functions based on IQCs: Theory described further tomorrow afternoon
  - Currently no synthesis functions in toolbox: Recent research in this area by Scherer should provide tools for uncertain LFT-based systems.

Gridded Analysis and Synthesis

LPV analysis: Allowable parameter

Parameter-dependent system G_ρ of the form

$$\begin{bmatrix} \dot{x}(t) \\ e(t) \end{bmatrix} = \begin{bmatrix} A(\rho(t)) & B(\rho(t)) \\ C(\rho(t)) & D(\rho(t)) \end{bmatrix} \begin{bmatrix} x(t) \\ d(t) \end{bmatrix}$$



<u>Allowable parameter trajectories</u>

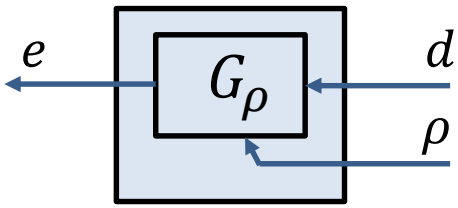
One parameter:

- Parameter trajectory $\rho(\cdot)$ is *allowable* if $-1 \leq \rho(t) \leq 1,\ -\beta \leq \dot{\rho}(t) \leq \beta$, for all t

Multiple parameters:

- Parameter trajectory $\rho(\cdot)$ is *allowable* if $\rho(t) \in P,\ -\beta_i \leq \dot{\rho}_i(t) \leq \beta_i$, for all t, and all i.

LPV analysis: gain



LPV performance metric

$$\max_{allowable\ \rho} \max_{d} \frac{\|G_\rho d\|_2}{\|d\|_2}$$

$$\begin{bmatrix} \dot{x}(t) \\ e(t) \end{bmatrix} = \begin{bmatrix} A(\rho(t)) & B(\rho(t)) \\ C(\rho(t)) & D(\rho(t)) \end{bmatrix} \begin{bmatrix} x(t) \\ d(t) \end{bmatrix}$$

Suppose there is a differentiable matrix-valued function $X(\rho)$ satisfying $X(\rho) > 0$ and

$$\begin{bmatrix} \beta \frac{dX}{d\rho} + A^T(\rho)X(\rho) + X(\rho)A(\rho) & X(\rho)B(\rho) & C^T(\rho) \\ B^T(\rho)X(\rho) & -I & D^T(\rho) \\ C(\rho) & D(\rho) & -I \end{bmatrix} < 0$$

$$\begin{bmatrix} -\beta \frac{dX}{d\rho} + A^T(\rho)X(\rho) + X(\rho)A(\rho) & X(\rho)B(\rho) & C^T(\rho) \\ B^T(\rho)X(\rho) & -I & D^T(\rho) \\ C(\rho) & D(\rho) & -I \end{bmatrix} < 0$$

for all $-1 \leq \rho \leq 1$. Along any allowable trajectory, integrate a combination of these to conclude $\|e\|_{2,T} \leq \|d\|_{2,T}$, bounding the performance metric by 1.

LPV analysis: gain (cont'd)

Analysis condition is $X(\rho) > 0$ and

$$\begin{bmatrix} \pm\beta \dfrac{dX}{d\rho} + A^T(\rho)X(\rho) + X(\rho)A(\rho) & X(\rho)B(\rho) & C^T(\rho) \\ B^T(\rho)X(\rho) & -I & D^T(\rho) \\ C(\rho) & D(\rho) & -I \end{bmatrix} < 0$$

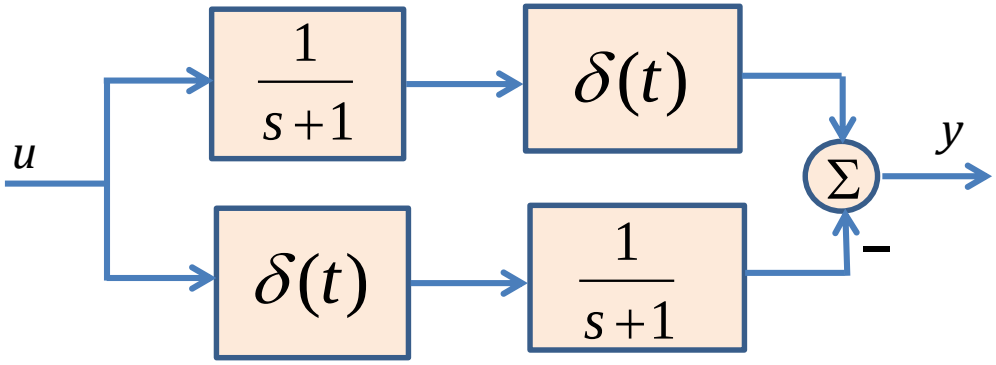
Pragmatic approach:

- Pick a basis for $X(\rho)$ so $X(\rho) = \sum_{i=1}^{n_b} f_i(\rho)X_i$
- Grid the set P
- Solve the resulting convex feasibility problem (LMI) in the matrix variables X_i subject to the constraints at the grid points.
- Assess feasibility of LMI solution on a dense parameter grid
  - Repeat as needed.

lpvnorm

```
% lpvnorm  Bound on induced L2 norm for PSS systems.
%
% Gamma = lpvnorm(P) computes an upper bound Gamma on the induced
% L2 norm of the PSS P. The L2 norm bound is valid for arbitrarily
% fast variations of the system parameters.
%
% Gamma = lpvnorm(P,Xb) computes a tighter (less conservative) bound
% on the induced L2 norm by using a parameter dependent matrix X(p)
% and bounds on the parameter rates of variation. The basis
% functions used to construct X(p) are specified with the structure
% array Xb.  Xb is an Nb-by-1 array with fields BasisFunction and
% partial derivatives.
```

Consider the following first order system with the time-varying parameter, δ(t)



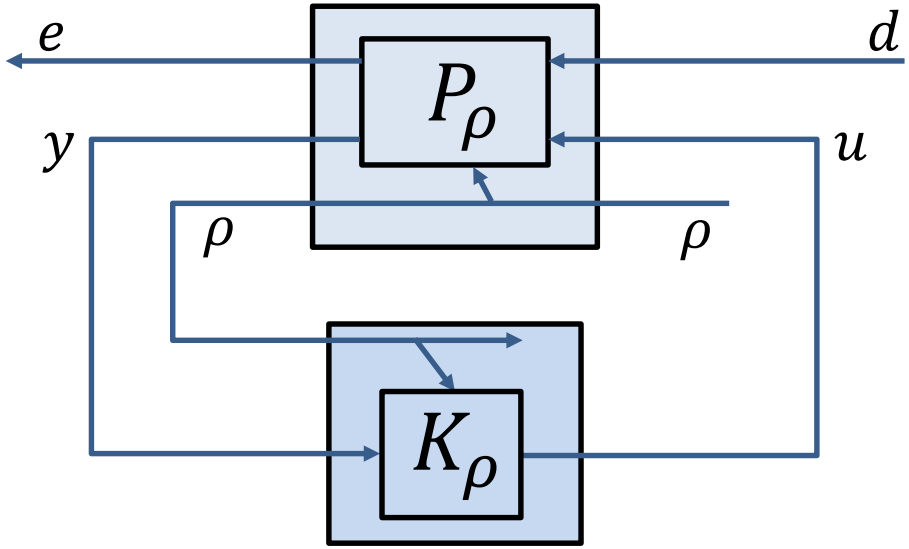
$$|\delta(t)| \le 1, -\beta \le \delta'(t) \le \beta$$

LPV Synthesis

Does there exist a controller of the form

$$\begin{bmatrix} A(\rho(t),\dot{\rho}(t)) & B(\rho(t),\dot{\rho}(t)) \\ C(\rho(t),\dot{\rho}(t)) & D(\rho(t),\dot{\rho}(t)) \end{bmatrix}$$

such that the gain-analysis holds for the closed-loop system?



Solution: search for parameter-dependent, positive-definite $X(\rho)$ and $Y(\rho)$ satisfying 3 ("full-info", "full-control", "coupled") parameter-dependent LMIs. Same issues remain…

- $2 \cdot 2^m + 1$ parameter-dependent LMIs.
- Need to hold for all $\rho \in P$
- Choice of basis functions for $X(\rho)$ and $Y(\rho)$

lpvsyn, lpvsynOptions

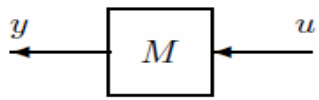
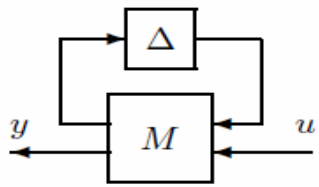
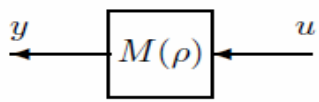
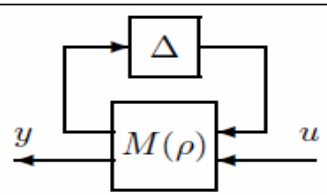
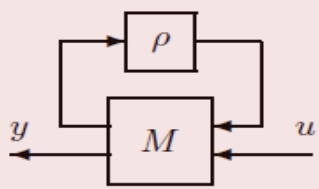
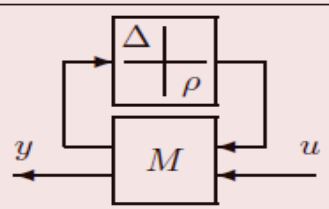
```
% lpvsyn    Synthesizes a controller which stabilizes the plant
%           model and minimize the induced L2 norm of the closed
%           loop system.
%
% [K,GAM] = lpvsyn(P,NMEAS,NCON,Xb,Yb,opt) returns a LPV controller,
% K, which stabilizes the closed-loop system and achieves an induced
% L2 norm of GAM. Xb and Yb are nbasisx-by-1 array of bases which
% correspond to bounds on the rate-of-variation of the time varying
% parameters. Xb and Yb correspond to the full information and
% control LMI solutions. opt is an lpvsynOptions options which
% allows properties of the LPV synthesis algorithms to be specified.
```

Consider a rate-dependent, output-feedback control problem involving stabilization, tracking, disturbance rejection and input penalty. The problem is take from :

- Meyer, G. and L. Cicolani, “Application of nonlinear systems inverses to automatic flight control design-system concepts and flight evaluations,” *AGARDograph: Theory and Applications 01 Optimal Control in Aerospace Systems, No. 251,1981.*
- F Wu, XH Yang, AK Packard, and G Becker, “Induced L_2-norm control for LPV systems with bounded parameter variation rates,” Int. J Robust and Nonlinear Control, 6, 9-10, 1996, pp. 983–998.

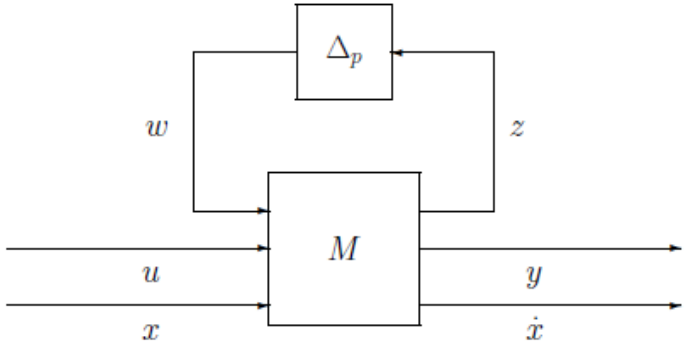
LFT Modeling, Analysis, & Synthesis

Data Structures for LPV/LFT Systems

Object Type	Block	Matrix	System	Frequency Response
Nominal		`double`	`ss`	`frd`
Uncertain		`umat`	`uss`	`ufrd`
Nominal Gridded LPV		`pmat`	`pss`	`pfrd`
Uncertain Gridded LPV		`upmat`	`upss`	`upfrd`
Nominal LFT LPV		`pmatlft`	`psslft`	
Uncertain LFT LPV		`pmatlft`	`psslft`	

LFT Based LPV Systems

$$\begin{bmatrix} z(t) \\ y(t) \\ \dot{x}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}}_{M} \begin{bmatrix} w(t) \\ u(t) \\ x(t) \end{bmatrix} \qquad w(t) = \underbrace{\begin{bmatrix} p_1(t)I_{r_1} & 0 & \cdots & 0 \\ 0 & p_2(t)I_{r_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_s(t)I_{r_s} \end{bmatrix}}_{\Delta_p} z(t)$$



LPV-LFT model

LFT matrices and systems are defined as a function of parameters:

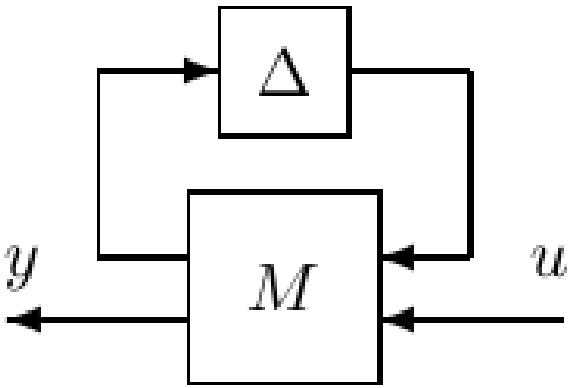
- Restricted to rational parameter dependence.
- Interconnections of LFTs are themselves LFTs.

TVREAL: Real parameter used to construct LFT models

```
% Create a TVREAL object with Name, Range, and RateBounds
>> a = tvreal('a',[1 5],[-1 1])
Time-varying real parameter "a", range [1,5], rate bounds [-1,1].
>> class(a)
ans = tvreal

% Change Name
>> a.Name = 'x'
Time-varying real parameter "x", range [1,5], rate bounds [-1,1].

% Change Range
>> a.Range = [4 7]
Time-varying real parameter "x", range [4,7], rate bounds [-1,1].

% Change RateBounds
>> a.RateBounds = [-6 6]
Time-varying real parameter "x", range [4,7], rate bounds [-6,6].
```

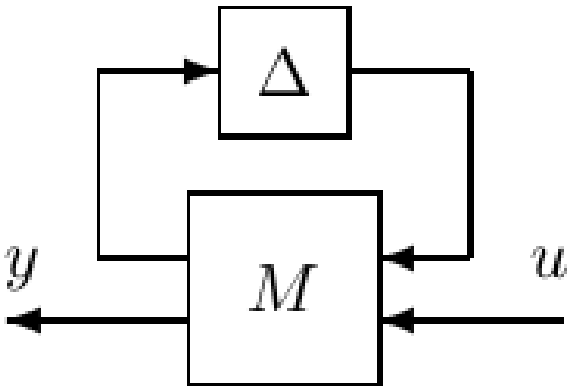
LFTDATA: Extract "M-Delta" components of a PLFTMAT

```
% Create PLFTMAT with two parameters
>> x = tvreal('x', [0 6], [-5 5] );
>> y = tvreal('y', [-2 2], [-1 1]);
>> mat = [x^2+y x*y-7]

% Extract parameters into "M-Delta" form
>> [M,Delta] = lftdata(mat,[],'Parameter');
>> class(M)
ans =
Double
>> size(M)
ans =
     6     7
>> M
M =
         0    3.0000         0         0         0    5.1962         0
         0         0         0         0         0    1.7321         0
         0         0         0         0    2.4495         0         0
         0         0         0         0         0    1.4142         0
         0         0         0         0         0         0    1.4142
    1.7321    5.1962    1.7321    1.4142    4.2426    9.0000   -7.0000
```



LFTDATA: Extract "M-Delta" components of a PLFTMAT

```
% Create PLFTMAT with two parameters
>> x = tvreal('x', [0 6], [-5 5] );
>> y = tvreal('y', [-2 2], [-1 1]);
>> mat = [x^2+y x*y-7]

% Extract parameters into "M-Delta" form
>> [M,Delta] = lftdata(mat,[],'Parameter');
>> Delta
PLFTMAT with 5 rows and 5 columns.
The PLFTMAT consists of the following blocks:
  x: Time-varying real, range=[0,6], rate bounds=[-5,5], 3 occurrences
  y: Time-varying real, range=[-2,2], rate bounds=[-1,1], 2 occurrences

% Delta is diagonal with normalized, repeated copies of parameters
>> usample(Delta)
ans =
   -0.8377         0         0         0         0
         0   -0.8377         0         0         0
         0         0   -0.8377         0         0
         0         0         0    0.8588         0
         0         0         0         0    0.8588
```

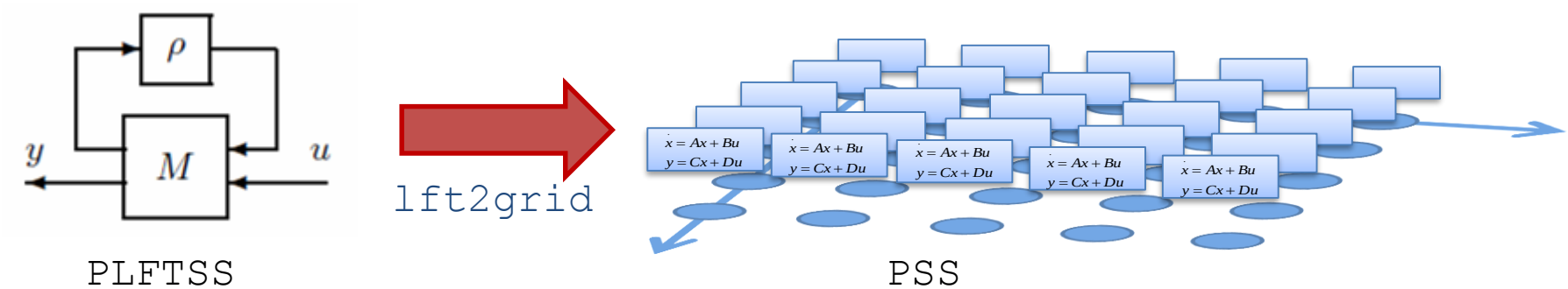


PLFTSS: LFT parameter-varying state-space model

```
% Create three TVREALs
>> a1 = tvreal('a1', [-3 -1], [-1 1] );
>> a2 = tvreal('a2', [-5 -4], [-1 1] );
>> b = tvreal('b', [1 2],[-2 2] );

% Use TVREALs to generate a PLFTSS
>> A = [0 1; a1 a2];
>> B = [0; b];
>> C = [1 0];
>> D = 0;
>> sys = ss(A,B,C,D)
Continuous-time PLFTSS with 1 outputs, 1 inputs, 2 states.
The model consists of the following blocks:
  a1: Time-varying real, range=[-3,-1], rate bounds=[-1,1], 1 occurrences
  a2: Time-varying real, range=[-5,-4], rate bounds=[-1,1], 1 occurrences
  b: Time-varying real, range=[1,2], rate bounds=[-2,2], 1 occurrences
```

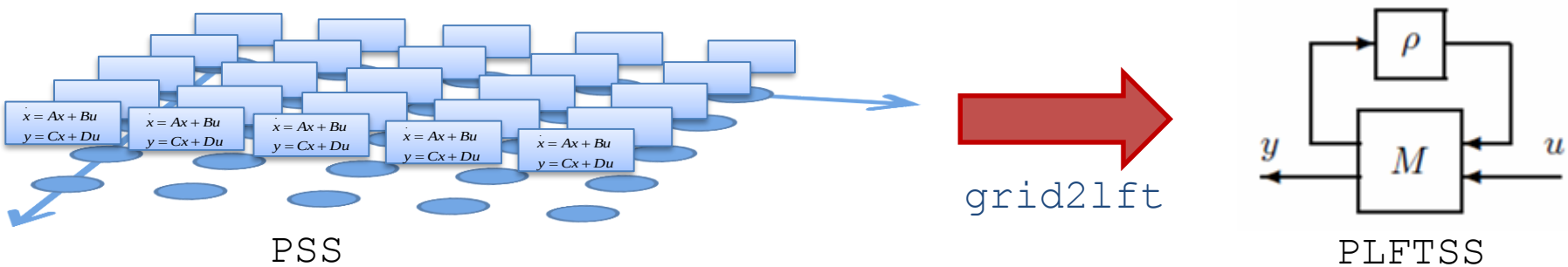
Transition from LFT to grid-based LPV

```
% lft2grid transforms a LFT-LPV into a grid-based LPV model.
>> a = tvreal('a',[1 5],[-1 1]);
>> S = ss(-a,a,1,0);
% LFT can be evaluated on a prescribed grid:
>> r = rgrid('a',[1 2 3],[-1 1]);
>> Sgrid1 = lft2grid(S,r)
PSS with 1 States, 1 Outputs, 1 Inputs, Continuous System.
The PSS consists of the following blocks:
  a: Gridded real, 3 points in range [1,3], rate bounds [-1,1].
% LFT can be evaluated on a uniform grid spanning its range
>> Sgrid2 = lft2grid(S,3)
PSS with 1 States, 1 Outputs, 1 Inputs, Continuous System.
The PSS consists of the following blocks:
  a: Gridded real, 3 points in range [1,5], rate bounds [-1,1].
```



Transition from grid-based LPV to LFT

```
% grid2lft transforms a grid based LPV model into a LFT-LPV model.
% Grid data fitted with polynomial functions of the parameters
>> Slft = grid2lft(Sgrid,2)
    Continuous-time PLFTSS with 1 outputs, 1 inputs, 1 states.
    The model consists of the following blocks:
        a: Time-varying real, range = [1,5],
           rate bounds = [-1,1], 1 occurrences

% Repeated parameter copies become an issue for higher order fits
% Obtaining a minimal realization a non-trivial problem.
```



Sample of pointwise functions

balreal
bode
connect
damp
dcgain
impulse
feedback
gapmetric
gram
grid2lft
h2syn
hinfsyn
lft
loopmargin
loopsens
lqr
lsim
margin
ncfsyn
pole
step
sysic

Sample of LPV functions

lpvimpulselp
vinitial
lpvlsim
lpvnorm
lpvstep
lpvsyn
Lpvsplit
lpvwcgain

LPVTools: Open Source Release

- Release 1.0:
  - http://www.aem.umn.edu/~SeilerControl/software.shtml
  - Google Search: SeilerControl
  - Static release under GNU Affero GPL License
  - Full documentation (manual, command line, Matlab “doc”)
- Basic objects and results implemented
  - LFT Analysis and Synthesis (Packard, Scherer, Gahinet, Apkarian, …)
  - Gridded Analysis and Synthesis (Wu, Packard, Becker, …)
  - Model Reduction with Generalized Gramians (Wood, Glover, Widowati, …)
  - Simulink interface
- Important gaps remain

Appendix

PMAT: Matrix function of real parameters

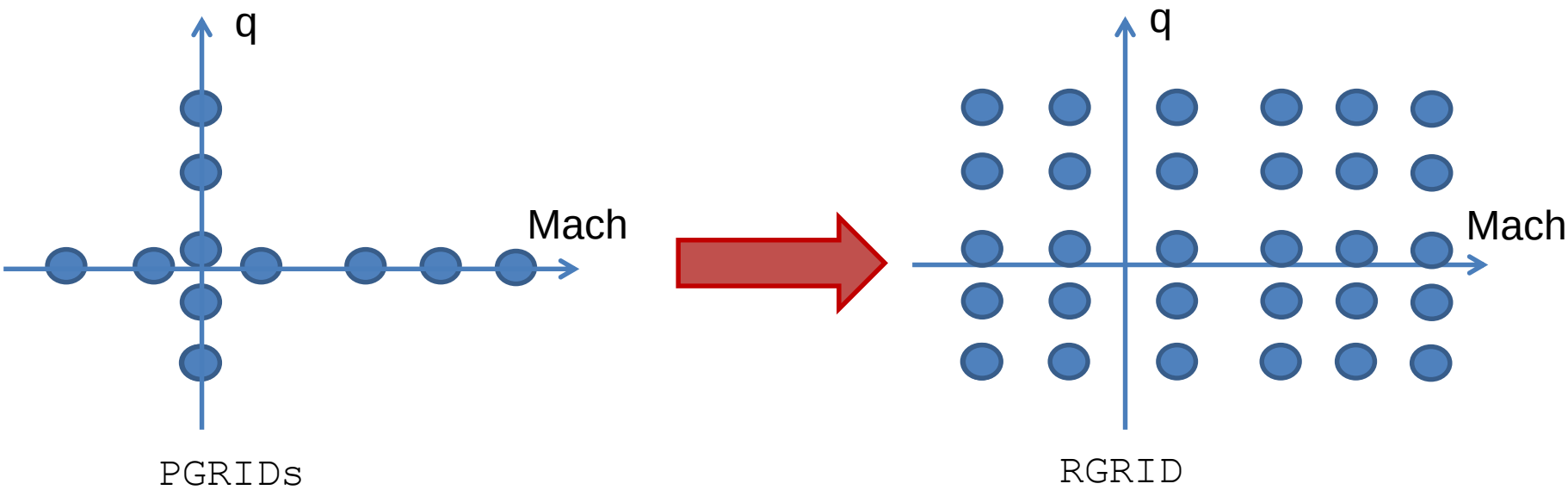
```
% Create three parameters
>> x = pgrid('x', linspace(0,6,30) );
>> y = pgrid('y', -2:2);
>> z = pgrid('z', 0:5);

% Matrix operations handle multiple parameters
>> M1 = [x y];
>> M2 = [cos(y) z^2];

>> M3 = M1+M2
PMAT with 1 rows and 2 columns.
The PMAT consists of the following blocks:
  x: Gridded real, 30 points in [0,6], rate bounds [-Inf,Inf].
  y: Gridded real, 5 points in [-2,2], rate bounds [-Inf,Inf].
  z: Gridded real, 6 points in [0,5], rate bounds [-Inf,Inf].
```

PMAT: Matrix function of real parameters

```
% Create PMAT with dependence on two parameters
>> x = pgrid('x', linspace(0,6,30) );
>> y = pgrid('y', -2:2);
>> M = [cos(x) abs(y)];

% Obtain structure with all parameters
>> M.Parameter
ans =
    x: [1x1 pgrid]
    y: [1x1 pgrid]

% "Parameter" is a universal access point to all real parameters
>> M.Parameter.y.Range
ans =
    -2     2

>> M.Parameter.x.RateBounds = [-3 3]
PMAT with 1 rows and 2 columns.
The PMAT consists of the following blocks:
  x: Gridded real, 30 points in [0,6], rate bounds [-3,3].
  y: Gridded real, 5 points in [-2,2], rate bounds [-Inf,Inf].
```

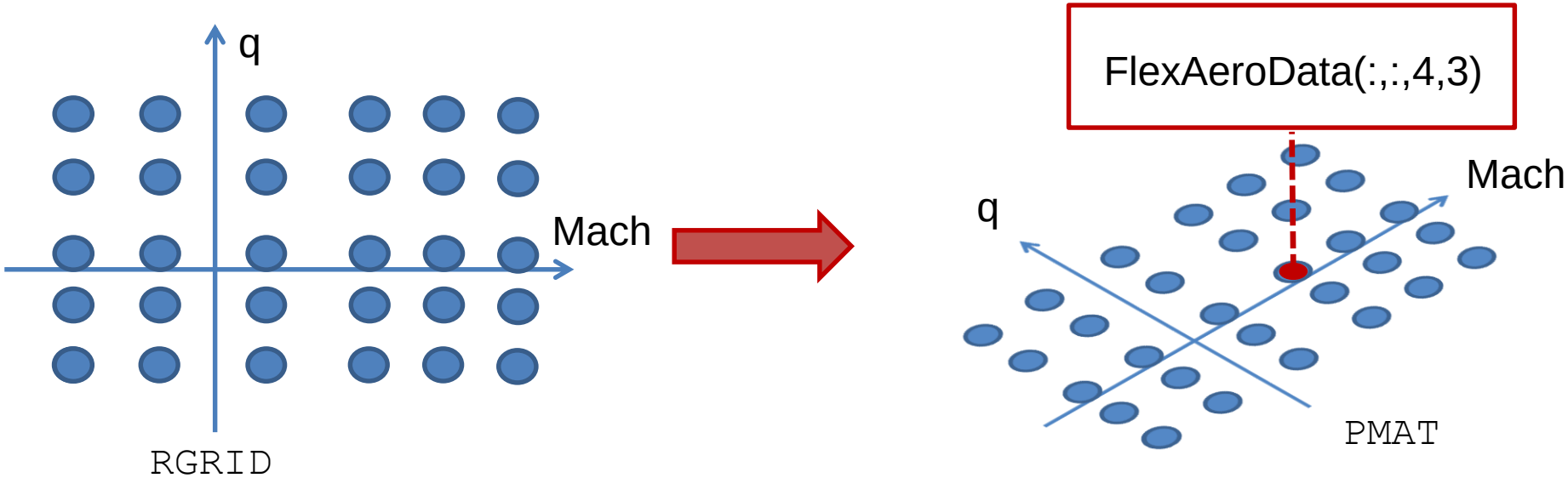
RGRID: Rectangular grid of points defined by real parameters

```
% Create RGRID domain from PGRIDs
>> Mach = pgrid('Mach', 0.7:0.02:0.8);

>> q = pgrid('q', 125:25:225,[-10 10]);

% Combine into a single RGRID
>> r = rgrid(Mach,q)
RGRID with the following parameters:
  Mach: Gridded real, 6 points in [0.7,0.8], rate bounds [-Inf,Inf].
  q: Gridded real, 5 points in [125,225], rate bounds [-10,10].
```



RGRID: Rectangular grid of points defined by real parameters

```
% Create PMAT from raw data
>> r = rgrid(Mach,q);
>> load FlexAeroData.mat
>> size(FlexAeroData)
ans =
     2     3     6     5
>> M = pmat(FlexAeroData,r)
PMAT with 2 rows and 3 columns.
The PMAT consists of the following blocks:
  Mach: Gridded real, 6 points in [0.7,0.8], rate bounds [-Inf,Inf].
  q: Gridded real, 5 points in [125,225], rate bounds [-10,10].
```

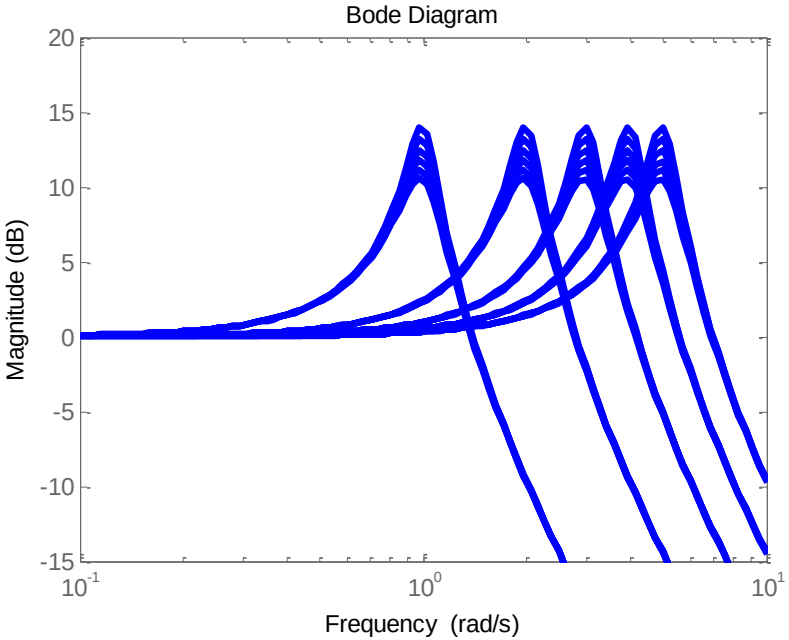


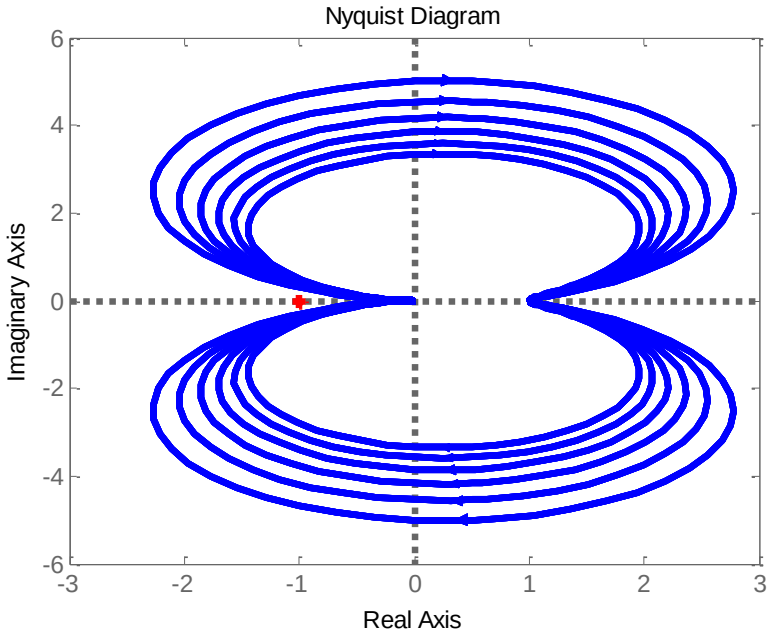
Most PMAT operations carry over to PSS

```
% Concatenation, multiplication, addition of PSS
>> [5*sys sys+4]
PSS with 2 States, 1 Outputs, 2 Inputs, Continuous System.
The PSS consists of the following blocks:
  a: Gridded real, 7 points in [-3,-2], rate bounds [-Inf,Inf].
  b: Gridded real, 4 points in [2,3], rate bounds [-Inf,Inf].

% "Parameter" is a universal access point to all real parameters
>> sys.Parameter.a.RateBounds = [-4 4]
PSS with 2 States, 1 Outputs, 2 Inputs, Continuous System.
The PSS consists of the following blocks:
  a: Gridded real, 7 points in [-3,-2], rate bounds [-4,4].
  b: Gridded real, 4 points in [2,3], rate bounds [-Inf,Inf].
```

"Pointwise" Frequency Response

```
% Construct second order LPV system
>> wn = pgrid('wn',1:5);
>> zeta = pgrid('zeta',0.1:0.01:0.15);
>> G = ss([0 1; -wn^2 -2*zeta*wn],...
                [0;wn^2],[1 0],0);

% Bode mag. plot at each grid point
>> w = logspace(-1,1,100);
>> bodemag(G,w)

% Nyquist plot at each grid point
>> nyquist(G,w)
```





LPV analysis: gain (cont'd)

Analysis condition is $X(\rho) > 0$ and

$$\begin{bmatrix} \pm\beta \dfrac{dX}{d\rho} + A^T(\rho)X(\rho) + X(\rho)A(\rho) & X(\rho)B(\rho) & C^T(\rho) \\ B^T(\rho)X(\rho) & -I & D^T(\rho) \\ C(\rho) & D(\rho) & -I \end{bmatrix} < 0$$

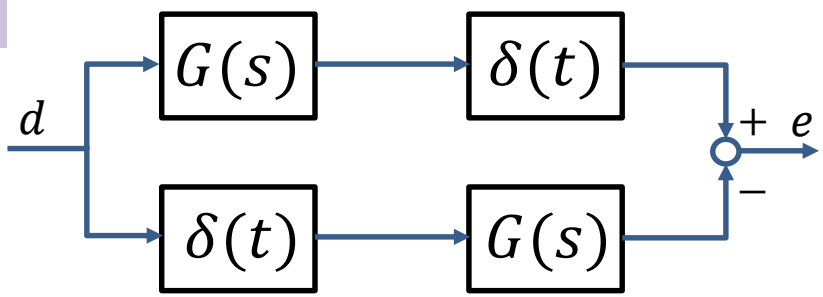
Issues:

- This is for 1-parameter, leading to 2, parameter-dependent LMIs.
- If there are m parameters, then the corners of the rate-bounds turn this into 2^m parameter-dependent LMIs.
- Needs to be solved for all $\rho \in P$
- Searching for a positive-definite, matrix function $X(\rho)$

LPV Norm: example

$-\beta \le \dot{\delta}(t) \le \beta$

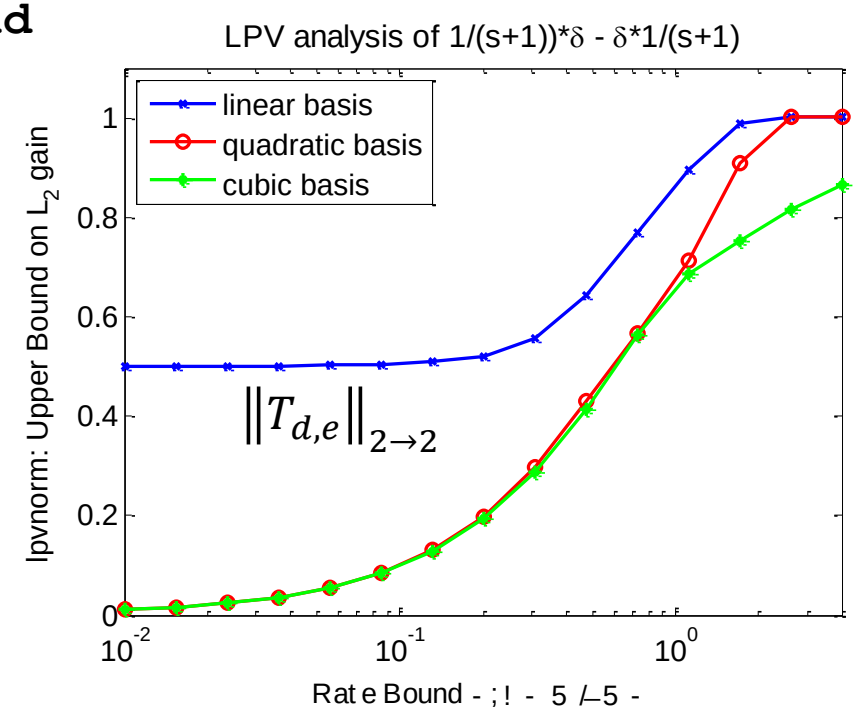
Basis functions: $1, \delta, \delta^2, \delta^3$



$-1 \le \delta(t) \le 1$ $\qquad G(s) = \dfrac{1}{s+1}$

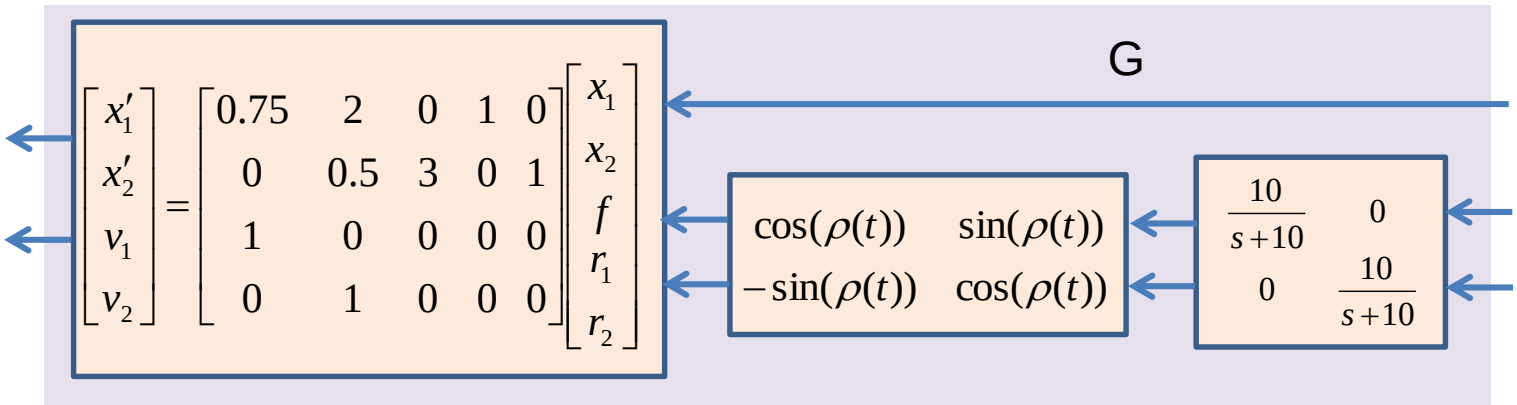
```
G = tf(1,[1 1]);
Vals = linspace(-1,1,20);
delta = pgrid('delta',Vals);
H = delta*G-G*delta;

pnorm = norm(H,inf);
OnGrid = max(abs(pnorm.Data(:)))
ans =
    0
syslpvnorm = lpvnorm(H)
syslpvnorm =
   1.0007e+00
```

```
bf = basis(delta,'delta',1);
Xb = [bf^0;bf;bf^2;bf^3];
rb = logspace(-2,log10(4),15);
for i=1:numel(rb)
  H.Parameter.delta.RateBounds...
    = [-rb(i) rb(i)];
  bnds1(i) = lpvnorm(H,Xb(1:2));
  bnds2(i) = lpvnorm(H,Xb(1:3));
  bnds3(i) = lpvnorm(H,Xb);
end
```



lpvsyn, lpvsynOptions

The generalized plant model, G, is created from 3 subsystems, an unstable continuous-time plant, P, and parameter-dependent rotation matrix, R, and two 1st order actuator models:

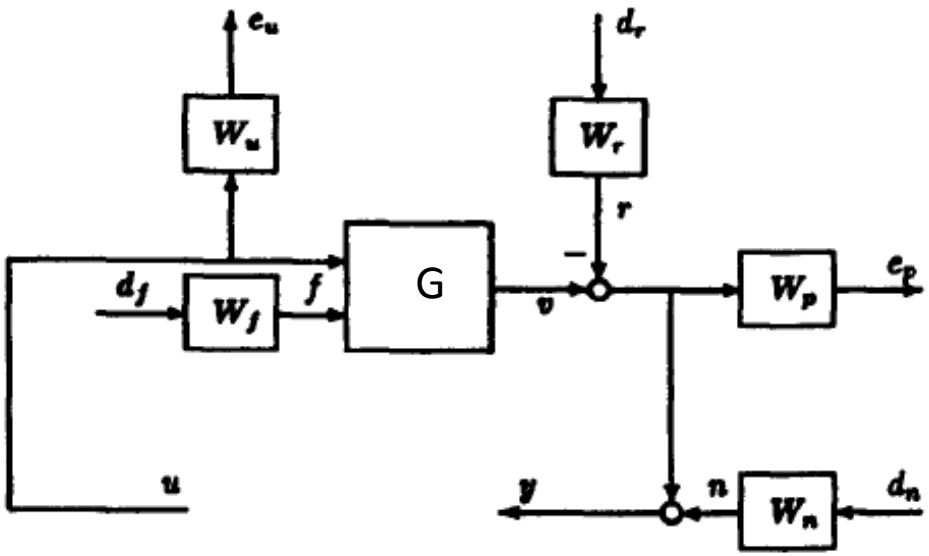


The control interconnection structure is given on the next slide. Weighting functions are as follows:

$$W_\rho = I_2 \quad W_n = \frac{10(s+10)}{s+1000} \quad W_f = 1 \quad W_u = \frac{1}{280} \quad W_r = \frac{20}{s+0.2} I_2$$

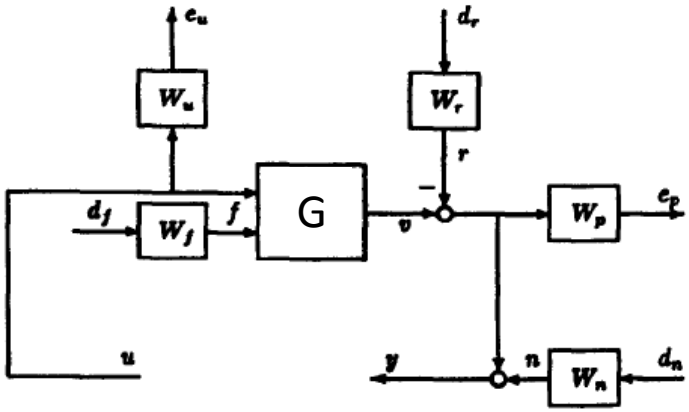
lpvsyn, lpvsynOptions

The control problem interconnection with the weighting function is denoted as *H*. The rate bounds on ρ(t) are +/- 5.




```
% LPV lpvsyn example
>> rho = pgrid('rho',linspace(-pi,pi,7));
>> rho.RateBounds = [-5 5];
>> pcos = cos(pmat(rho));
>> psin = sin(pmat(rho));
>> A = [0.75 2 pcos psin;0 0.5 -psin pcos;0 0 -10 0; 0 0 0 -10];
>> G = pss(A,[0 0 0;3 0 0;0 10 0;0 0 10],[1 0 0 0;0 1 0 0],zeros(2,3))

% Weights
>> Wp = eye(2);
>> Wn = ss(10*tf([1 10],[1 1000]))*eye(2);
>> Wf = 1;
>> Wu = (1/280)*eye(2);
>> Wr = ss(tf(20,[1 0.2]))*eye(2);
```

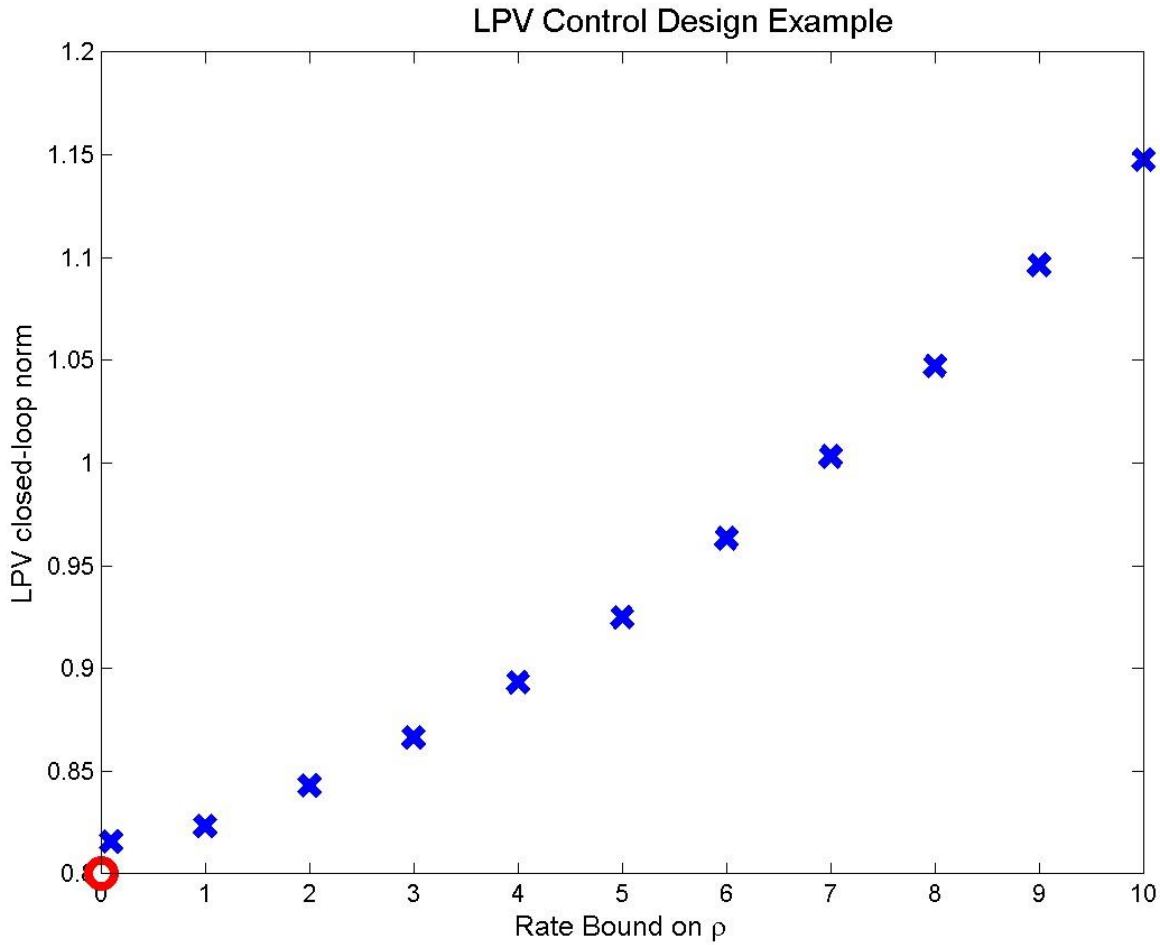
lpvsyn, lpvsynOptions




```
% Control Interconnection Structure
>> systemnames      = 'G Wp Wn Wf Wu Wr';
>> input_to_G    = '[ Wf; u ]';
>> input_to_Wp   = '[ G-Wr ]';
>> input_to_Wn   = '[ dn ]';
>> input_to_Wf   = '[ df ]';
>> input_to_Wu   = '[ u ]';
>> input_to_Wr   = '[ dr ]';
>> inputvar         = '[ df; dr(2); dn(2); u(2)]';
>> outputvar        = '[ Wu; Wp; G-Wr+Wn ]';
>> H = sysic;

% Basis function, LPV Rate-Bounded Control Design
>> b1 = basis(1,0);
>> bcos = basis(pcos,'rho',-psin);
>> bsin = basis(psin,'rho',pcos);
>> Xb = [b1;bcos;bsin]; Yb = Xb;
>> opt = lpvsynOptions('BackOffFactor' ,1.02);
>>[klpv2,normlpv] = lpvsyn(H,2,2,Xb,Yb,opt);

% Eliminate rate dependence of LPV Rate-Bounded Controller
>> klpv2r = lpvinterp(klpv2,{'rhoDot'},{0});
>> Klpv = lpvelimiv(klpv2r);
```

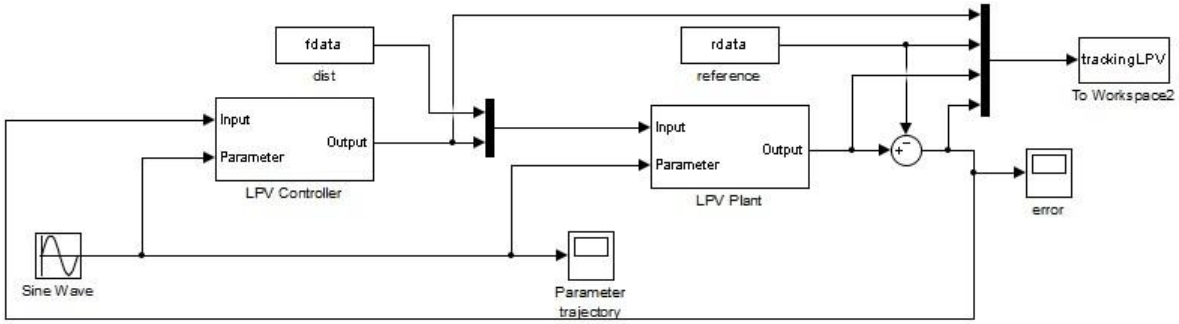
lpvsyn, lpvsynOptions

```
% Generate LPV controller for different rate bounds
>> rb = [0.1 1:10];
>> for ii = 1:numel(rb);
     Hb.Parameter.rho.RateBounds = [-rb(ii) rb(ii)];
     [~,nl1] = lpvsyn(Hb,2,2,Xb,Yb,opt);
     nlpv3(ii) = nl1;
   end
```

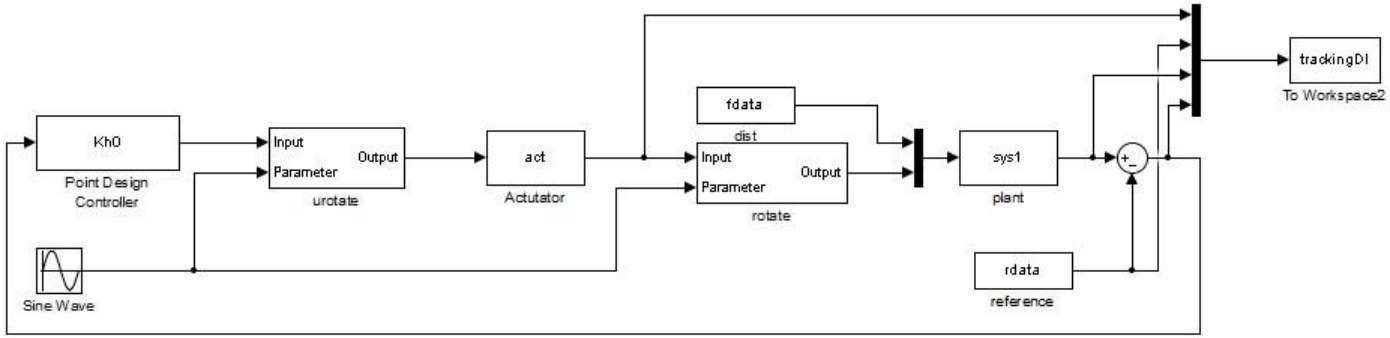


Dynamic Inversion Design

```
% Compare a dynamic inversion controller based on the H-infinity
% point design and an inverse rotation matrix.
>> purotate = pss([pcos -psin; psin pcos]);
>> Kdyninv = purotate*Kh0;
>> cllpv     = lft(H,Klpv); cldyninv  = lft(H,Kdyninv);
>> ncllpv    = lpvnorm(cllpv,Xb); ncldyninv = lpvnorm(cldyninv,Xb);
% Induced L2 norm of the LPV and dynamic inversion closed-loop
>> [ncllpv ncldyninv]
ans =
    1.3725    8.3393
```

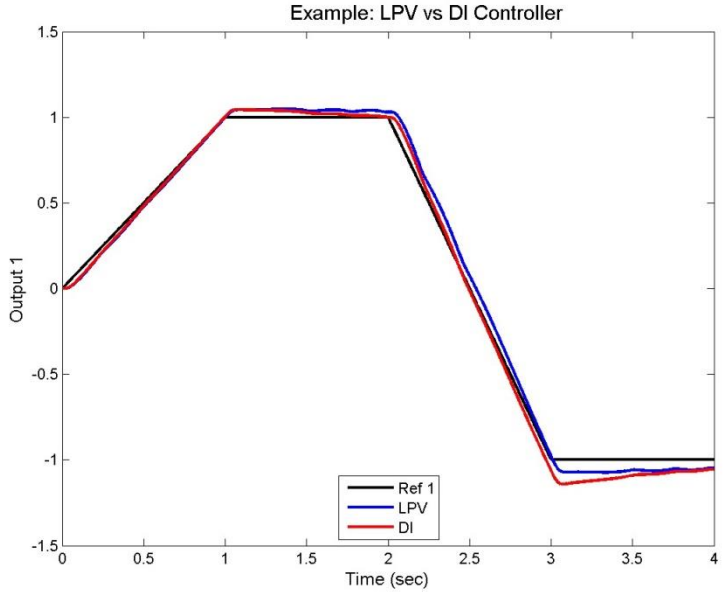


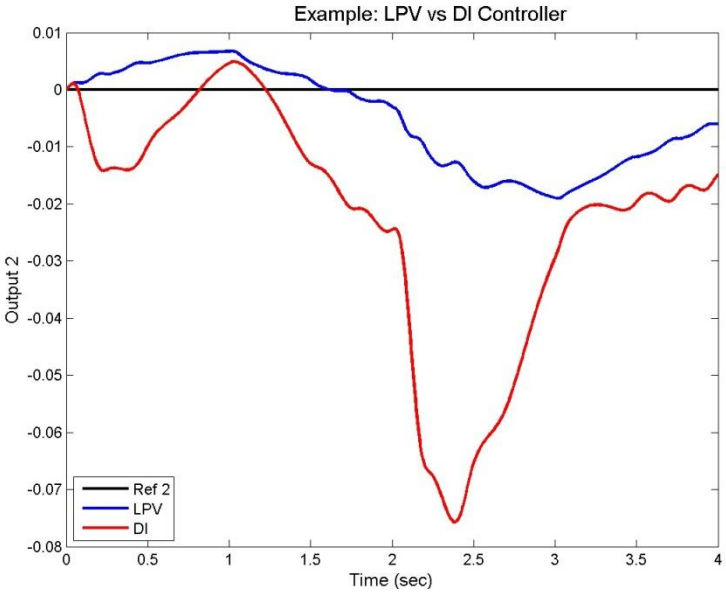
Example: LPV Controller



Example: Dynamic Inversion Controller

lpvsyn, lpvsynOptions






```
% Induced L2 norm of the LPV and dynamic inversion controller
>> [ncllpv ncldyninv]
ans =
    1.3725    8.3393
```

PLFTMAT: Matrix function of real parameters

```
% (Rational) operations on a TVREAL give back a PLFTMAT
>> x = tvreal('x', [0 10], [-2 2] );
>> y = 3*x^2-1/x
PLFTMAT with 1 rows and 1 columns.
The PLFTMAT consists of the following blocks:
  x: Time-varying real, range=[0,10], rate bounds=[-2,2], 3 occurrences

% Horizontal and vertical concatenation to build matrices
>> x = tvreal('x', [0 10], [-2 2] );
>> M = [x^2 x; -2 1/x^2]
PLFTMAT with 2 rows and 2 columns.
The PLFTMAT consists of the following blocks:
  x: Time-varying real, range=[0,10], rate bounds=[-2,2], 5 occurrences

% Matrix indexing
>> M(1,2)
PLFTMAT with 1 rows and 1 columns.
The PLFTMAT consists of the following blocks:
  x: Time-varying real, range=[0,10], rate bounds=[-2,2], 1 occurrences
```

PLFTMAT: Access to parameter properties

```
% Create PLFTMAT with dependence on two parameters
>> x = tvreal('x', [0 6], [-5 5] );
>> y = tvreal('y', [-2 2], [-1 1]);
>> M = [x^2+5 x*y];

% Obtain structure with all parameters
>> M.Parameter
ans =
    x: [1x1 tvreal]
    y: [1x1 tvreal]

% "Parameter" is a universal access point to all real parameters
>> M.Parameter.y.Range
ans =
    -2     2

>> M.Parameter.x.RateBounds = [-3 3]
PLFTMAT with 1 rows and 2 columns.
The PLFTMAT consists of the following blocks:
  x: Time-varying real, range=[0,6], rate bounds=[-3,3], 3 occurrences
  y: Time-varying real, range=[-2,2], rate bounds=[-1,1], 1 occurrences
```

Most PLFTMAT operations carry over to PLFTSS

```
% Concatenation, multiplication, addition of PLFTSS
>> [5*sys sys+4]
Continuous-time PLFTSS with 1 outputs, 2 inputs, 4 states.
The model consists of the following blocks:
  a1: Time-varying real, range=[-3,-1], rate bounds=[-1,1], 2 occurrences
  a2: Time-varying real, range=[-5,-4], rate bounds=[-1,1], 2 occurrences
  b: Time-varying real, range=[1,2], rate bounds=[-2,2], 2 occurrences

% "Parameter" is a universal access point to all real parameters
>> sys.Parameter.a1.RateBounds = [-4 4]
Continuous-time PLFTSS with 1 outputs, 2 inputs, 4 states.
The model consists of the following blocks:
  a1: Time-varying real, range=[-3,-1], rate bounds=[-4,4], 2 occurrences
  a2: Time-varying real, range=[-5,-4], rate bounds=[-1,1], 2 occurrences
  b: Time-varying real, range=[1,2], rate bounds=[-2,2], 2 occurrences
```