Quantitative Local Analysis of Nonlinear Systems

Andrew Packard
Department of Mechanical Engineering
University of California, Berkeley

Ufuk Topcu
Control and Dynamical Systems
California Institute of Technology

Pete Seiler and Gary Balas
Aerospace Engineering and Mechanics
University of Minnesota

June 9, 2009

Acknowledgements

- Members of Berkeley Center for Control and Identification
  - Ryan Feeley
  - Evan Haas
  - George Hines
  - Zachary Jarvis-Wloszek
  - Erin Summers
  - Kunpeng Sun
  - Weehong Tan
  - Timothy Wheeler
- Abhijit Chakraborty (Univ of Minnesota)
- Air Force Office of Scientific Research (AFOSR) for the grant # FA9550-05-1-0266 (Development of Analysis Tools for Certification of Flight Control Laws) 05/01/05 - 04/30/08
- NASA NRA Grant/Cooperative Agreement NNX08AC80A, "Analytical Valdiation Tools for Safety Critical Systems, Dr. Christine Belcastro, Technical Monitor, 01/01/2008 - 12/31/2010

Outline

- **Motivation**
- Preliminaries
- ROA analysis using SOS optimization and solution strategies
- Robust ROA analysis with parametric uncertainty
- Local input-output analysis
- Robust ROA and performance analysis with unmodeled dynamics
- F-18

Motivation: Flight Controls

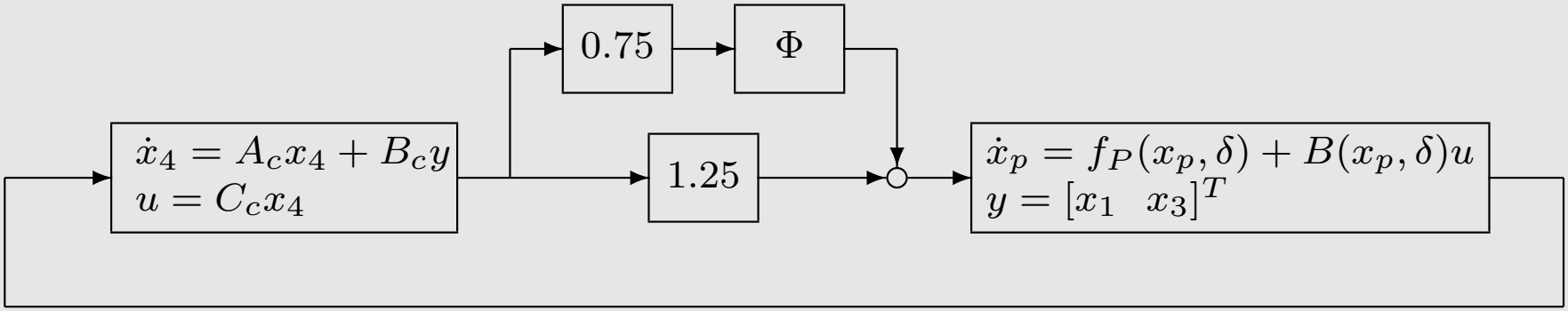
- Validation of flight controls mainly relies on linear analysis tools and nonlinear (Monte Carlo) simulations.
- This approach generally works well but there are drawbacks:
  - It is time consuming and requires many well-trained engineers.
  - Linear analyses are only valid over an infinitesimally small region of the state space.
  - Linear analyses are not sufficient to understand truly nonlinear phenomenon, e.g. the falling leaf mode in the F/18 Hornet.
  - Linear analyses are not applicable for adaptive control laws or systems with hard nonlinearities.
- There is a need for nonlinear analysis tools which provide quantitative performance/stability assessments over a provable region of the state space.

Example: F/A-18 Hornet Falling Leaf Mode

- The US Navy lost many F/A-18 A/B/C/D Hornet aircraft due to an out-of-control phenomenon known as the Falling Leaf mode.

- Revised control laws were integrated into the F/A-18 E/F Super Hornet and this appears to have resolved the issue.
- Classical linear analyses did not detect a performance issue with the baseline control laws.
- We have used nonlinear analyses to estimate the size of the region of attraction (ROA) for both controllers.
  - The ROA is the set of initial conditions which can be brought back to the trim condition by the controller.
  - The size of this set is a good metric for detecting departure phenomenon.
- These nonlinear results will be discussed later in the workshop.

Representative Example



- 3-state pitch-axis model,
  - cubic vector field, bilinear terms involving u and x_p
  - 2 uncertain parameters (δ, mass and mass-center variability)
  - unmodeled dynamics uncertainty, Φ
    - uncertainty in dynamic process how control surface deflections manifest as forces and torques on the rigid aircraft
    - Φ causal, stable operator, with $\|\Phi(z)\|_2 \leq \|z\|_2$ for all $z \in \mathcal{L}_2$
- integral control

Closed-loop system is not globally stable

- robust region-of-attraction analysis to assess effect of
  - nonzero initial conditions
  - two forms of model uncertainty

Representative Example: Results

Form of results

- Ball of initial conditions (eg., $x_0^T x_0 \leq \beta$) guaranteed to lie in the region-of-attraction
- Provably correct, certified by sum-of-squares decompositions

Nominal: Optimized quartic Lyapunov function certifies $\beta = 15.3$, and there is an initial condition with $x_0^T x_0 = 16.1$ which results in a divergent solution.

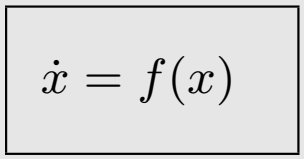
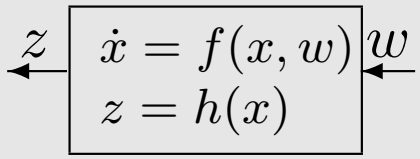
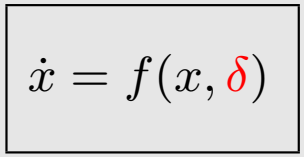
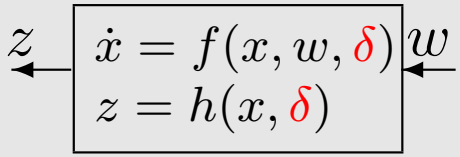
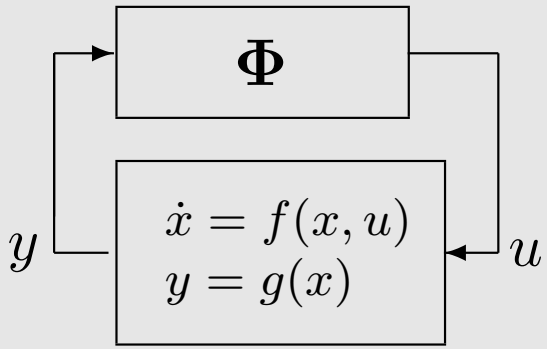
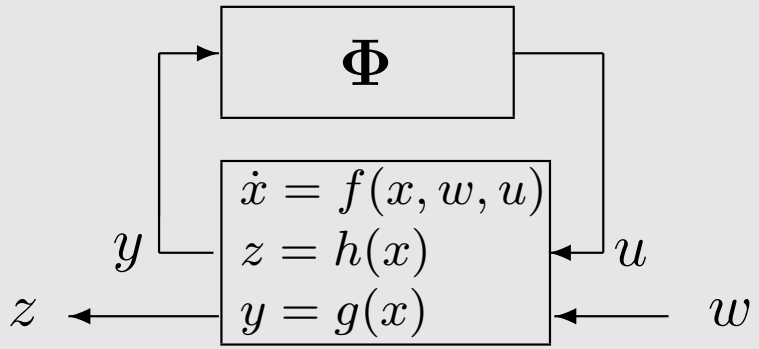
Parametric: Using a divide-and-conquer strategy, $\beta = 7.7$ is certified for all parameter values; moreover, there is an admissible parameter and initial condition with $x_0^T x_0 = 7.9$ which results in a divergent solution.

Dynamics Uncertainty: $\beta = 6.7$ is certified for all admissible operators Φ.

Parametric and Unmodeled Dynamics: $\beta = 4.1$ is certified for all parameter values and all admissible operators Φ.

Tools for quantitative nonlinear robustness/performance analysis

Quantify with **certificate**

	Internal	**Input-output**
	Regions-of-attraction	Reachable sets, Local gains
Nominal system		
Parametric uncertainty		
Unmodeled dynamics		

Outline

- Motivation
- **Preliminaries**
  - Linear Algebra Notation
  - Optimization with Matrix Inequalities (LMIs, BMIs, SDPs)
  - Polynomials and Sum of Squares
  - SOS Optimization and Connections to SDPs
  - Set Containment Conditions
- ROA analysis using SOS optimization and solution strategies
- Robust ROA analysis with parametric uncertainty
- Local input-output analysis
- Robust ROA and performance analysis with unmodeled dynamics
- F-18

Warning

- In several places, a relationship between an algebraic condition on some real variables and input/output/state properties of a dynamical system is claimed.
- In nearly all of these types of statements, we use same symbol for a particular real variable in the algebraic statement as well as the corresponding signal in the dynamical system.
- This could be a source of confusion, so care on the readers part is required.

Linear Algebra Notation

- $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}$, $\mathbb{N}$ denote the set of real numbers, complex numbers, integers, and non-negative integers, respectively.
- The set of all $n \times 1$ column vectors with real number entries is denoted $\mathbb{R}^n$.
- The set of all $n \times m$ matrices with real number entries is denoted $\mathbb{R}^{n \times m}$.
- The element in the i'th row and j'th column of $M \in \mathbb{R}^{n \times m}$ is denoted M_{ij} or m_{ij}.
- If $M \in \mathbb{R}^{n \times m}$ then M^T denotes the transpose of M.
- Set notation:
  - $a \in A$ is read "a is an element of A".
  - $X \subset Y$ is read "X is a subset of Y".
  - Given $S \subset \mathbb{R}^n$ such that $0 \in S$, S_{cc} denotes the connected component of the set containing 0.
  - $\Omega_{p,\beta}$ will denote the sublevel set $\{x \in \mathbb{R}^n \ : \ p(x) \leq \beta\}$.

Sign Definite Matrices

- $M \in \mathbb{R}^{n \times n}$ is symmetric if $M = M^T$.
- The set of $n \times n$ symmetric matrices is denoted $\mathcal{S}^{n \times n}$.
- $F \in \mathcal{S}^{n \times n}$ is:
  1. positive semidefinite (denoted $F \succeq 0$) if $x^T F x \geq 0$ for all $x \in \mathbb{R}^n$.
  2. positive definite (denoted $F \succ 0$) if $x^T F x > 0$ for all $x \in \mathbb{R}^n$.
  3. negative semidefinite (denoted $F \preceq 0$) if $x^T F x \leq 0$ for all $x \in \mathbb{R}^n$.
  4. negative definite (denoted $F \prec 0$) if $x^T F x < 0$ for all $x \in \mathbb{R}^n$.
- For $A, B \in \mathcal{S}^{n \times n}$, write $A \prec B$ if $A - B \prec 0$. Similar notation holds for $\preceq$, $\succ$, and $\succeq$.

Linear and Bilinear Matrix Inequalities

- Given matrices $\{F_i\}_{i=0}^N \subset \mathcal{S}^{n\times n}$, Linear Matrix Inequality (LMI) is a constraint on $\lambda \in \mathbb{R}^N$ of the form:

$$F_0 + \sum_{k=1}^{N} \lambda_k F_k \succeq 0$$

- Given matrices $\{F_i\}_{i=0}^N$, $\{G_j\}_{j=1}^M$, and $\{H_{k,j}\}_{k=1\ j=1}^{N\ M}$ $\subset \mathcal{S}^{n\times n}$, a Bilinear Matrix Inequality (BMI) is a constraint on $\lambda \in \mathbb{R}^N$ and $\gamma \in \mathbb{R}^M$ of the form:

$$F_0 + \sum_{k=1}^{N} \lambda_k F_k + \sum_{j=1}^{M} \gamma_k G_j + \sum_{k=1}^{N}\sum_{j=1}^{M} \lambda_k \gamma_j H_{k,j} \succeq 0$$

Semidefinite Programming (SDP)

- A semidefinite program is an optimization problem with a linear cost, LMI constraints, and matrix equality constraints.

- Given matrices $\{F_i\}_{i=0}^N \subset \mathcal{S}^{n\times n}$ and $c \in \mathbb{R}^N$, the primal and dual forms of an SDP are:
  1. Primal Form:*

$$\begin{array}{ll} \max_{Z\in\mathcal{S}^{n\times n}} & -\mathbf{Tr}\left[F_0 Z\right] \\ \text{subject to:} & \mathbf{Tr}\left[F_k Z\right] = c_k \quad k = 1,\ldots,N \\ & Z \succeq 0 \end{array}$$

  2. Dual Form:

$$\begin{array}{ll} \min_{\lambda\in\mathbb{R}^N} & c^T\lambda \\ \text{subject to:} & F_0 + \sum_{k=1}^N \lambda_k F_k \succeq 0 \end{array}$$

(∗) There exists a matrix A such that the equality constraints in the Primal form can be equivalently expressed as $Az = c$ where $z = vec(Z)$. This is the form which will appear when we consider polynomial optimizations.

Properties of SDPs

- SDPs have many interesting and useful properties:
  - The primal form is a concave optimization and the dual form is a convex optimization.
  - Local optima of both forms are global optima.
  - The primal/dual forms are Lagrange duals of each other.
  - If the primal/dual problems are strictly feasible then there is no duality gap, i.e. both problems achieve the same optimal value.

- There is quality software to solve SDPs
  - Freely available solvers: Sedumi, SDPA-M, CSDP, DSDP
  - Commercial solver: LMILab
  - Some algorithms, e.g. the method of centers, solve only the dual form.
  - Primal/dual methods, e.g. Sedumi, solve both forms simultaneously.

Optimizations with BMIs

- Given $c \in \mathbb{R}^N$, $d \in \mathbb{R}^M$, and $\{F_i\}_{i=0}^N$, $\{G_j\}_{j=1}^M$, $\{H_{k,j}\}_{k=1\ j=1}^{N\ M} \subset \mathcal{S}^{n\times n}$, a bilinear matrix optimization is of the form:

$$\min_{\lambda\in\mathbb{R}^N,\gamma\in\mathbb{R}^M} c^T\lambda + d^T\gamma$$

subject to:

$$F_0 + \sum_{k=1}^{N} \lambda_k F_k + \sum_{j=1}^{M} \gamma_k G_j + \sum_{k=1}^{N}\sum_{j=1}^{M} \lambda_k\gamma_j H_{k,j} \succeq 0$$

- Optimizations with BMIs do not have all of the nice properties of SDPs. In general,
  - They are not convex optimizations.
  - They have provably bad computational complexity.
  - There can be local minima which are not global minima.
  - The Lagrange dual is a concave optimization but there is a duality gap.
- One useful property is that the constraint is an LMI if either λ or γ is held fixed.

Solving BMI Optimizations

- Approaches to solving BMI Optimizations:
  - Commercial software designed for BMIs, e.g. PENBMI
  - Gradient-based nonlinear optimization, e.g. `fmincon`
  - Coordinate-wise Iterations:
    1. Initialize a value of λ.
    2. Hold λ fixed and solve for optimal γ This is an SDP.
    3. Hold γ fixed and solve for optimal λ This is an SDP.
    4. Go back to step 2 and repeat until values converge.
  - Branch and Bound
  - Exploit structure: If $M = 1$, the objective function is γ, and the BMI is a quasi-convex constraint on λ and γ then the BMI optimization can be solved via bisection.

- Issues:
  - Solver may converge to a local minima which is not the global minima.
  - Final solution is dependent on solver initial conditions

Polynomials

- Given $\alpha \in \mathbb{N}^n$, a monomial in n variables is a function $m_\alpha : \mathbb{R}^n \to \mathbb{R}$ defined as $m_\alpha(x) := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$.
- The degree of a monomial is defined as $\deg m_\alpha := \sum_{i=1}^n \alpha_i$.
- A polynomial in n variables is a function $p : \mathbb{R}^n \to \mathbb{R}$ defined as a finite linear combination of monomials:

$$p := \sum_{\alpha \in \mathcal{A}} c_\alpha m_\alpha = \sum_{\alpha \in \mathcal{A}} c_\alpha x^\alpha$$

  where $\mathcal{A} \subset \mathbb{N}^n$ is a finite set and $c_\alpha \in \mathbb{R}\ \forall \alpha \in \mathcal{A}$.
- The set of polynomials in n variables $\{x_1, \ldots, x_n\}$ will be denoted $\mathbb{R}[x_1, \ldots, x_n]$ or, more compactly, $\mathbb{R}[x]$.
- The degree of a polynomial f is defined as $\deg f := \max_{\alpha \in \mathcal{A}, c_\alpha \neq 0} \deg m_\alpha$.
- $\theta \in \mathbb{R}[x]$ will denote the zero polynomial, i.e. $\theta(x) = 0\ \forall x$.

Multipoly Toolbox

- Multipoly is a Matlab toolbox for the creation and manipulation of polynomials of one or more variables.
- Example:

```
pvar x1 x2
p = 2*x1^4 + 2*x1^3*x2 - x1^2*x2^2 + 5*x2^4
```

- A scalar polynomial of T terms and V variables is stored as a $T \times 1$ vector of coefficients, a $T \times V$ matrix of natural numbers, and a $V \times 1$ array of variable names.

$$\text{p.coef} = \begin{bmatrix} 2 \\ 2 \\ -1 \\ 5 \end{bmatrix}, \ \text{p.deg} = \begin{bmatrix} 4 & 0 \\ 3 & 1 \\ 2 & 2 \\ 0 & 4 \end{bmatrix}, \ \text{p.var} = \begin{bmatrix} \text{x1} \\ \text{x2} \end{bmatrix}$$

Vector Representation

- If p is a polynomial of degree $\leq d$ in n variables then there exists a coefficient vector $c \in \mathbb{R}^{l_w}$ such that $p = c^T w(x)$ where

  $$w(x) := \left[1,\ x_1,\ x_2,\ \ldots,\ x_n,\ x_1^2,\ x_1x_2,\ \ldots,\ x_n^2,\ \ldots,\ x_n^d\right]^T$$

  l_w denotes the length of w. It is easy to verify $l_w = \binom{n+d}{d}$.
- Example:

```
p = 2*x1^4 + 2*x1^3*x2 - x1^2*x2^2 + 5*x2^4;
x = [x1;x2];
w = monomials(x,0:4);
c = poly2basis(p,w);
p - c'*w
[c w]
```

Gram Matrix Representation

- If p is a polynomial of degree $\leq 2d$ in n variables then there exists a $Q \in \mathcal{S}^{l_z \times l_z}$ such that $p = z^T Q z$ where

$$z := \left[1,\ x_1,\ x_2,\ \ldots,\ x_n,\ x_1^2,\ x_1 x_2,\ \ldots,\ x_n^2,\ \ldots,\ x_n^d\right]^T$$

  The dimension of z is $l_z = \binom{n+d}{d}$.
- Equating coefficients of p and $z^T Q z$ yields linear equality constraints on the entries of Q
  - Define $q := vec(Q)$ and $l_w := \binom{n+2d}{2d}$.
  - There exists $A \in \mathbb{R}^{l_w \times l_z^2}$ and $c \in \mathbb{R}^{l_w}$ such that $p = z^T Q z$ is equivalent to $Aq = c$.
- There are $h := \frac{l_z(l_z+1)}{2} - l_w$ linearly independent homogeneous solutions $\{N_i\}_{i=1}^h$ each of which satisfies $z^T N_i z = \theta$.
- Summary: All solutions to $p = z^T Q z$ can be expressed as the sum of a particular solution and a homogeneous solution. The set of homogeneous solutions depends on n and d while the particular solution depends on p.

Gram Matrix Example

```
p = 2*x1^4 + 2*x1^3*x2 - x1^2*x2^2 + 5*x2^4;
[z,c,A,w] = gramconstraint(p);
p-c'*w
Q = full(reshape(A\c,[3 3]));
p-z'*Q*z

% Q is a particular solution in vectorized form
% Each column of N is a homogenous solution in vectorized form.
[z,Q,N] = gramsol(p);
Q = full(reshape(Q,[3 3]));
N = full(reshape(N,[3 3]));
p-z'*Q*z
z'*N*z
```

$$z = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}, \; Q = \begin{bmatrix} 2 & 1 & -0.5 \\ 1 & 0 & 0 \\ -0.5 & 0 & 5 \end{bmatrix}, \; N = \begin{bmatrix} 0 & 0 & -0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0 \end{bmatrix}$$

Positive Semidefinite Polynomials

- $p \in \mathbb{R}[x]$ is positive semi-definite (PSD) if $p(x) \geq 0 \; \forall x$. The set of PSD polynomials in n variables $\{x_1, \ldots, x_n\}$ will be denoted $\mathcal{P}[x_1, \ldots, x_n]$ or $\mathcal{P}[x]$.
- Testing if $p \in \mathcal{P}[x]$ is NP-hard when the polynomial degree is at least four.
  - For a general class of functions, verifying global non-negativity is recursively undecidable.
- Our computational procedures will be based on constructing polynomials which are PSD.
- Objective: Given $p \in \mathbb{R}[x]$, we would like a polynomial-time sufficient condition for testing if $p \in \mathcal{P}[x]$.

Reference: Parrilo, P., *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*, Ph.D. thesis, California Institute of Technology, 2000. (Chapter 4 of this thesis and the reference contained therein summarize the computational issues associated with verifying global non-negativity of functions.)

Sum of Squares Polynomials

- p is a sum of squares (SOS) if there exist polynomials $\{f_i\}_{i=1}^N$ such that $p = \sum_{i=1}^N f_i^2$.
- The set of SOS polynomials in n variables $\{x_1, \ldots, x_n\}$ will be denoted $\Sigma[x_1, \ldots, x_n]$ or $\Sigma[x]$.
- If p is a SOS then p is PSD.
  - The Motzkin polynomial, $p = x^2y^4 + x^4y^2 + 1 - 3x^2y^2$, is PSD but not SOS.
  - Hilbert (1888) showed that $\mathcal{P}[x] = \Sigma[x]$ only for a) $n = 1$, b) $d = 2$, and c) $d = 4$, $n = 2$.
- p is a SOS iff there exists $Q \succeq 0$ such that $p = z^T Q z$.

Reference: Choi, M., Lam, T., and Reznick, B., Sums of Squares of Real Polynomials, *Proceedings of Symposia in Pure Mathematics*, Vol. 58, No. 2, 1995, pp. 103 – 126.

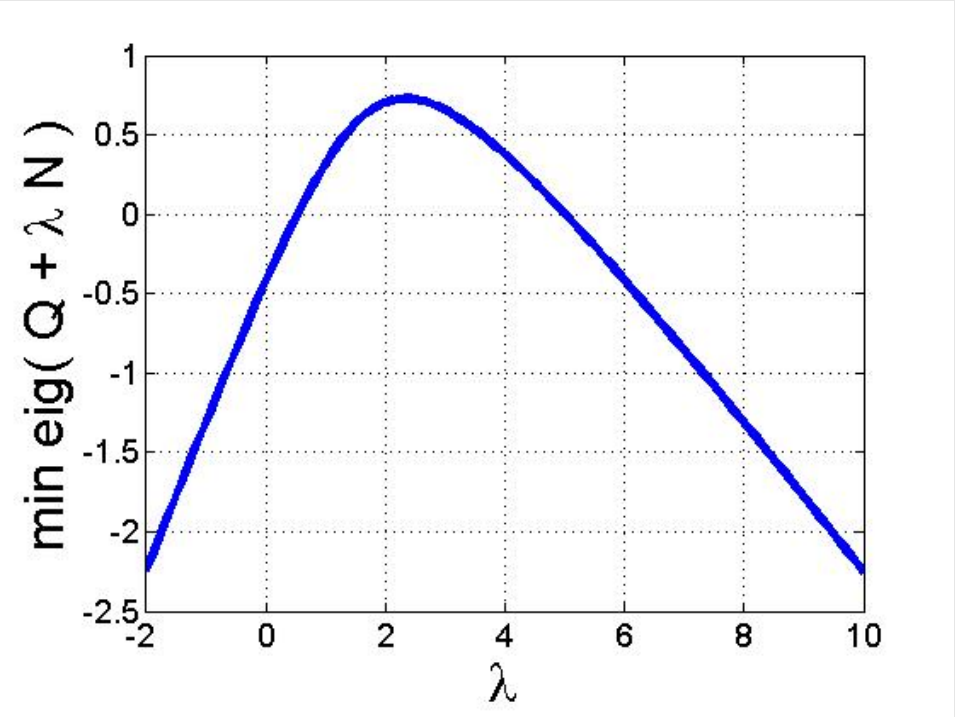
SOS Example (1)

All possible Gram matrix representations of

$$p = 2x_1^4 + 2x_1^3x_2 - x_1^2x_2^2 + 5x_2^4$$

are given by $z^T (Q + \lambda N) z$ where:

$$z = \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}, \; Q = \begin{bmatrix} 2 & 1 & -0.5 \\ 1 & 0 & 0 \\ -0.5 & 0 & 5 \end{bmatrix}, \; N = \begin{bmatrix} 0 & 0 & -0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0 \end{bmatrix}$$



p is SOS iff

$$Q + \lambda N \succeq 0$$

for some $\lambda \in \mathbb{R}$.

SOS Example (2)

p is SOS since $Q + \lambda N \succeq 0$ for $\lambda = 5$.
An SOS decomposition can be constructed from a Cholesky factorization:

$$Q + 5N = L^T L$$

where:

$$L = \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & 1 & -3 \\ 0 & 3 & 1 \end{bmatrix}$$

Thus

$$\begin{aligned} p &= 2x_1^4 + 2x_1^3 x_2 - x_1^2 x_2^2 + 5x_2^4 \\ &= (Lz)^T (Lz) \\ &= \frac{1}{2}\left(2x_1^2 - 3x_2^2 + x_1 x_2\right)^2 + \frac{1}{2}\left(x_2^2 + 3x_1 x_2\right)^2 \in \Sigma\left[x\right] \end{aligned}$$

Example from: Parrilo, P., *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*, Ph.D. thesis, California Institute of Technology, 2000.

Gram Matrix Rank

- The number of terms in the SOS decomposition is equal to the rank of the Gram matrix.
  - In the previous example $Q + 5N \succeq 0$ has rank $= 2$ and the SOS decomposition has two terms.
  - For $\lambda = 2.5$, $Q + 2.5N \succ 0$ has rank $= 3$ and the SOS decomposition has three terms.
- Low rank Gram matrix solutions are positive semidefinite but not strictly positive definite.
- For some problems, the feasible solution set is low-dimension and consists only of low-rank Gram matrix solutions. This can cause some numerical difficulties.

Connection to LMIs

Checking if a given polynomial p is a SOS can be done by solving a linear matrix inequality (LMI) feasibility problem.

1. Primal (Image) Form:
   - Find $A \in \mathbb{R}^{l_w \times l_z^2}$ and $c \in \mathbb{R}^{l_w}$ such that $p = z^T Q z$ is equivalent to $Aq = c$ where $q = vec(Q)$.
   - p is a SOS if and only if there exists $Q \succeq 0$ such that $Aq = c$.

2. Dual (Kernel) Form:
   - Let Q_0 be a particular solution of $p = z^T Q z$ and let $\{N_i\}_{i=1}^h$ be a basis for the homogeneous solutions.
   - p is a SOS if and only if there exists $\lambda \in \mathbb{R}^h$ such that $Q_0 + \sum_{i=1}^h \lambda_i N_i \succeq 0$.

Complexity of SOS LMI Feasibility Problem

If p is a degree $2d$ polynomial in n variables then the complexity of the LMI test for $p \in \Sigma[x]$ is:

- Primal (Image) Form: p is a SOS if and only if there exists $Q \succeq 0$ such that $Aq = c$ where $A \in \mathbb{R}^{l_w \times l_z^2}$ and $Q \in \mathbb{R}^{l_z \times l_z}$.
- Dual (Kernel Form): p is a SOS if and only if there exists $\lambda \in \mathbb{R}^h$ such that $Q_0 + \sum_{i=1}^{h} \lambda_i N_i \succeq 0$ where $Q, N_i \in \mathbb{R}^{l_z \times l_z}$.

$l_z = \binom{n+d}{d}$	$2d$=4	6	8	10
$n = 2$	6	10	15	21
5	21	56	126	252
9	55	220	715	2002
14	120	680	3060	11628
16	153	969	4845	20349

$l_w = \binom{n+2d}{2d}$	$2d$=4	6	8	10
$n = 2$	15	28	45	66
5	126	462	1287	3003
9	715	5005	24310	92378
14	3060	38760	319770	1961256
16	4845	74613	735471	5311735

$h = \frac{l_z(l_z+1)}{2} - l_w$	$2d$=4	6	8	10
$n = 2$	6	27	75	165
5	105	1134	6714	28875
9	825	19305	231660	1912625
14	4200	192780	4363560	65649750
16	6936	395352	11003964	201739340

SOS Test with `issos`

The `issos` function tests if $p \in \Sigma[x]$ by converting to an LMI feasibility problem:

```
[feas,z,Q,f] = issos(p)
```

`feas=1` if $p \in \Sigma[x]$ and `feas=0` otherwise. If feasible, then

- `z` and `Q` provide a Gram matrix decomposition:

  ```
  p = z'*Q*z,
  ```

  where `z` is a vector of monomials and `Q` is a positive semidefinite matrix.
- `z` may not include the complete list of $\binom{n+d}{d}$ monomials since `issos` uses some simple heuristics to prune out un-needed monomials.
- `f` is a vector of polynomials providing the SOS decomposition:

  ```
  p = f'*f,
  ```

SOS Example using issos

```
>> pvar x1 x2;
>> p = 2*x1^4 + 2*x1^3*x2 - x1^2*x2^2 + 5*x2^4;
>> [feas,z,Q,f]=issos(p);

% Verify feasibility of p \in SOS
>> feas
feas =
     1

% Verify z and Q are a Gram matrix decomposition
>> p - z'*Q*z
ans =
  -1.3185e-012*x1^4 + 6.5814e-013*x1^3*x2 - 2.3075e-012*x1^2*x2^2 +
5.6835e-016*x1*x2^3 - 3.304e-013*x2^4

% Verify Q is positive semi-definite
>> min(eig(Q))
ans =
  0.7271

% Verify SOS decomposition of p
>> p - f'*f
ans =
  -1.3221e-012*x1^4 + 6.5148e-013*x1^3*x2 - 2.3106e-012*x1^2*x2^2 +
1.3323e-015*x1*x2^3 - 3.3396e-013*x2^4
```

SOS Feasibility

SOS Feasibility: Given polynomials $\{f_k\}_{k=0}^m$, does there exist $\alpha \in \mathbb{R}^m$ such that $f_0 + \sum_{k=1}^m \alpha_k f_k$ is a SOS?

The SOS feasibility problem can also be posed as an LMI feasibility problem since α enters linearly.

1. Primal (Image) Form:
   - Find $A \in R^{l_w \times l_z^2}$ and $c_k \in \mathbb{R}^{l_w}$ such that $f_k = z^T Q z$ is equivalent to $Aq = c_k$ where $q = vec(Q)$.
   - Define $C := -[c_1, c_2, \cdots c_n] \in \mathbb{R}^{l_w \times n}$.
   - There is an $\alpha \in \mathbb{R}^n$ such that $f_0 + \sum_{k=1}^m \alpha_k f_k$ is a SOS iff there exists $\alpha \in \mathbb{R}^n$ and $Q \succeq 0$ such that $Aq + C\alpha = c_0$
2. Dual (Kernel) Form:
   - Let Q_k be particular solutions of $f_k = z^T Q z$ and let $\{N_i\}_{i=1}^h$ be a basis for the homogeneous solutions.
   - There is an $\alpha \in \mathbb{R}^n$ such that $f_0 + \sum_{k=1}^m \alpha_k f_k$ is a SOS iff there exists $\alpha \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}^h$ such that $Q_0 + \sum_{k=1}^m \alpha_k Q_k + \sum_{i=1}^h \lambda_i N_i \succeq 0$.

SOS Programming

SOS Programming: Given $c \in \mathbb{R}^m$ and polynomials $\{f_k\}_{k=0}^m$, solve:

$$\min_{\alpha \in \mathbb{R}^m} c^T \alpha$$

subject to:

$$f_0 + \sum_{k=1}^{m} \alpha_k f_k \in \Sigma\left[x\right]$$

This SOS programming problem is an SDP.

- The cost is a linear function of α.
- The SOS constraint can be replaced with either the primal or dual form LMI constraint.

A more general SOS program can have many SOS constraints.

General SOS Programming

SOS Programming: Given $c \in \mathbb{R}^m$ and polynomials $\{f_{j,k}\}_{j=1\ k=0}^{N_s\ m}$, solve:

$$\min_{\alpha \in \mathbb{R}^m} c^T \alpha$$

subject to:

$$f_{1,0}(x) + f_{1,1}(x)\alpha_1 + \cdots + f_{1,m}(x)\alpha_m \in \Sigma[x]$$

$$\vdots$$

$$f_{N_s,0}(x) + f_{N_s,1}(x)\alpha_1 + \cdots + f_{N_s,m}(x)\alpha_m \in \Sigma[x]$$

There is freely available software (e.g. SOSTOOLS, YALMIP, SOSOPT) that:

1. Converts the SOS program to an SDP
2. Solves the SDP with available SDP codes (e.g. Sedumi)
3. Converts the SDP results back into polynomial solutions

SOS Programming with sosopt

- SOS programs can be solved with `[info,dopt,sossol] = sosopt(sosconstr,x,obj)`
  - `sosconstr` is a cell array of polynomials constrained to be SOS.
  - `x` is a vector of the independent (polynomial) variables.
  - `obj` is the objective function to be minimized. `obj` must be a linear function of the decision variables.
  - Feasibility of the problem is returned in `info.feas`.
  - Decision variables are returned in `dopt`.
  - `sossol` provides a Gram decomposition for each constraint.
- Use `Z=monomials(vars,deg)` to generate a vector of all monomials in specified variables and degree.
- Use `p=polydecvar(dstr,Z,type)` to create a polynomial decision variable `p`.
  - If `type='vec'` then `p` has the form `p = D'*Z` where `D` is a column vector of decision variable coefficients.
  - If `type='mat'` then `p` has the form `p = Z'*D*Z` where `D` is a symmetric matrix of decision variable coefficients.
  - Note: For efficient implementations, only use the `'mat'` if `p` is constrained to be SOS. `p` must then be included in `sosconstr`. Do not use the `'mat'` form if `p` is not SOS constrained.

SOS Synthesis Example (1)

Problem: Minimize α subject to $f_0 + \alpha f_1 \in \Sigma[x]$ where

$$f_0(x) := -x_1^4 + 2x_1^3x_2 + 9x_1^2x_2^2 - 2x_2^4$$
$$f_1(x) := x_1^4 + x_2^4$$

For every $\alpha, \lambda \in \mathbb{R}$, the Gram Matrix Decomposition equality holds:

$$f_0 + \alpha f_1 = z^T (Q_0 + \alpha Q_1 + \lambda N_1) z$$

where

$$z := \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}, Q_0 = \begin{bmatrix} -1 & 1 & 4.5 \\ 1 & 0 & 0 \\ 4.5 & 0 & -2 \end{bmatrix}, Q_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, N_1 = \begin{bmatrix} 0 & 0 & -0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0 \end{bmatrix}$$

If $\alpha = 2$ and $\lambda = 0$ then $Q_0 + 2Q_1 + 9N_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix} \succeq 0$.

SOS Synthesis Example (2)

Use sosopt to minimize α subject to $f_0 + \alpha f_1 \in \Sigma[x]$

```
% Problem set-up with polynomial toolbox and sosopt
>> pvar x1 x2 alpha;
>> f0 = -x1^4 + 2*x1^3*x2 + 9*x1^2*x2^2 - 2*x2^4;
>> f1 = x1^4 + x2^4;
>> x = [x1;x2];
>> obj = alpha;
>> [info,dopt,sossol]=sosopt(f0+alpha*f1,x,obj);

% s is f0+alpha*f1 evaluated at the minimal alpha
>> s = sossol{1};

% z and Q are the Gram matrix decomposition of s
>> z=sossol{2}; Q=sossol{3};
```

SOS Synthesis Example (3)

```
% Feasibility of sosopt result
>> info.feas
ans =
     1

% Minimal value of alpha
>> dopt
dopt =
    'alpha'    [2.0000]

% Verify s is f0+alpha*f1 evaluated at alpha = 2.00
>> s-subs( f0+alpha*f1, dopt)
ans =
  0

% Verify z and Q are the Gram matrix decomposition of s
>> s-z'*Q*z
ans =
  -2.4095e-010*x1^4 + 4.3804e-011*x1^3*x2 - 2.1894e-011*x1^2*x2^2
+ 9.2187e-016*x1*x2^3 - 2.6285e-010*x2^4

% Verify Q is positive semi-definite
>> min(eig(Q))
ans =
  1.3718e-010
```

Set Containment Conditions

- Many nonlinear analysis problems can be formulated with set containment constraints.
- Need conditions for proving set containments:

  Given polynomials g_1 and g_2, define sets S_1 and S_2:

$$S_1 := \{x \in \mathbb{R}^n \ : \ g_1(x) \leq 0\}$$
$$S_2 := \{x \in \mathbb{R}^n \ : \ g_2(x) \leq 0\}$$

  Is $S_2 \subseteq S_1$?

- In control theory, the S-procedure is a common condition used to prove set containments involving quadratic functions. This can be generalize to higher degree polynomials.

S-Procedure

- Theorem: Suppose that g_1 and g_2 are quadratic functions, i.e. there exists matrices G_1, $G_2 \in \mathbb{R}^{n+1 \times n+1}$ such that

$$g_1(x) = \left[\begin{smallmatrix}1\\x\end{smallmatrix}\right]^T G_1 \left[\begin{smallmatrix}1\\x\end{smallmatrix}\right], \quad g_2(x) = \left[\begin{smallmatrix}1\\x\end{smallmatrix}\right]^T G_2 \left[\begin{smallmatrix}1\\x\end{smallmatrix}\right]$$

  Then $S_2 \subseteq S_1$ iff $\exists \lambda \geq 0$ such that $-G_1 + \lambda G_2 \succeq 0$.

- Proof:
  ($\Leftarrow$) If there exists $\lambda \geq 0$ such that $-G_1 + \lambda G_2 \succeq 0$ then $\lambda g_2(x) \geq g_1(x)\ \forall x$. Thus,

$$x \in S_2 \ \Rightarrow \ g_1(x) \leq \lambda g_2(x) \leq 0 \ \Rightarrow \ x \in S_1$$

  ($\Rightarrow$) See references.

- Comments:
  - For quadratic functions, an LMI feasibility problem can be solved to determine if $S_2 \subseteq S_1$.
  - λ is called a multiplier.

Reference: S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM, 1994. (See Chapter 2 and the reference contained therein for more details on the S-procedure.)

Polynomial S-Procedure

- Theorem: Let g_1 and g_2 be given polynomials. If there exists a polynomial $\lambda \in \mathcal{P}[x]$ such that $-g_1(x) + \lambda(x)g_2(x) \in \mathcal{P}[x]$ then $S_2 \subseteq S_1$.
- Proof: If $-g_1(x) + \lambda(x)g_2(x) \geq 0 \ \forall x$ and $\lambda(x) \geq 0 \ \forall x$ then:

$$x \in S_2 \ \Rightarrow \ g_1(x) \leq \lambda(x)g_2(x) \leq 0 \ \Rightarrow \ x \in S_1$$

- The PSD constraints are numerically difficult to handle. The theorem still holds if relaxed to SOS constraints:
  - If there exists a polynomial $\lambda \in \Sigma[x]$ such that $-g_1(x) + \lambda(x)g_2(x) \in \Sigma[x]$ then $S_2 \subseteq S_1$.
- Comments:
  - For polynomials, the feasibility of an SOS problem proves $S_2 \subseteq S_1$. This is only a sufficient condition.
  - This SOS feasibility problem can be converted to an LMI feasibility problem as described earlier.
  - λ is a polynomial / SOS multiplier.

Set Containment Maximization

- Given polynomials g_1 and g_2, the set containment maximization problem is:

$$\gamma^* = \max_{\gamma \in \mathbb{R}} \gamma$$
$$\text{s.t.: } \{x \in \mathbb{R}^n \ : \ g_2(x) \leq \gamma\} \subseteq \{x \in \mathbb{R}^n \ : \ g_1(x) \leq 0\}$$

- The polynomial S-procedure can be used to relax the set containment constraint:

$$\gamma_{lb} = \max_{\gamma \in \mathbb{R}, s \in \Sigma[x]} \gamma$$
$$\text{s.t.: } -g_1 + (g_2 - \gamma)s \in \Sigma[x]$$

- The solution of this optimization satisfies $\gamma_{lb} \leq \gamma^*$.

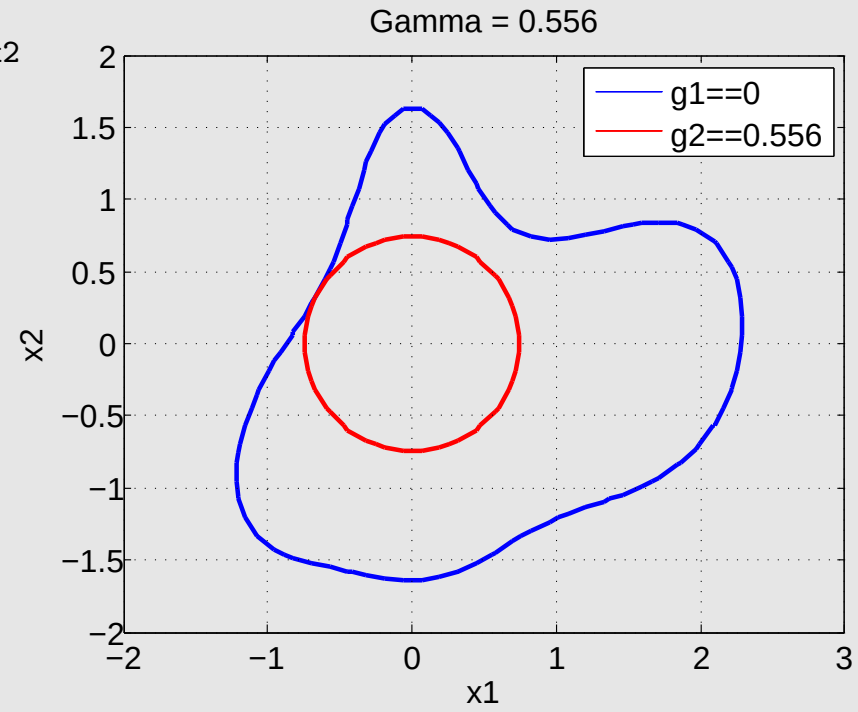
Solving the Set Containment Maximization

$$\gamma_{lb} = \max_{\gamma \in \mathbb{R}, s \in \Sigma[x]} \gamma$$
$$\text{s.t.:} \quad -g_1 + (g_2 - \gamma)s \in \Sigma\,[x]$$

- This optimization is bilinear in γ and s.
- For fixed γ, this is an SOS feasibility problem.
  - The constraint $s \in \Sigma\,[x]$ is replaced with $s = z^T Q z$ and $Q \succeq 0$.
  - The user must specify the monomials in z.
  - Let l_z denote the length of z. The $\frac{l_z(l_z+1)}{2}$ unique entries of Q are decision variables associated with s.
  - The constraint $-g_1 + (g_2 - \gamma)s \in \Sigma\,[x]$ is replaced with $-g_1 + (g_2 - \gamma)s = w^T M w$ and $M \succeq 0$.
  - $M \in \mathbb{R}^{l_w \times l_w}$ where $l_w := \binom{n+d}{d}$ and n, d are the number of variables and degree of the constraint.
- The set containment maximization can be solved via a sequence of SOS feasibility problems by bisecting on γ.
- This bisection has been efficiently implemented in `pcontain`.

pcontain Example

```
% Maximize size of a disk inside
% the contour of a 6th degree poly
pvar x1 x2;
x = [x1;x2];

% S1 := { x : g1(x)<= 0}
g1 =  0.3*x1^6 + 0.05*x2^6 - 0.5*x1^5 - 1.4*x1^3*x2
    + 2.3*x1^2*x2^2 - 0.9*x1^3 + 2.6*x1^2*x2 - 1;

% S2 := { x : g2(x)<= gamma}
g2 = x'*x;

% Define monomials for s
z = monomials(x,0:2);

% Use pcontain to maximize gamma s.t. S2 \in S1
% gbnds gives lower/upper bounds on optimal gamma
% sopt is the optimal multiplier
[gbnds,sopt]=pcontain(g1,g2,z)
gamma = gbnds(1);
gbnds =
    0.5560    0.5569

sopt =
   1.4483*x1^4 + 0.055137*x1^3*x2 + 0.44703*x1^2*x2^2 - 0.043336*x1*x2^3
   + 1.2961*x2^4 - 0.21988*x1^3 - 0.26998*x1^2*x2 - 0.050453*x1*x2^2
   + 0.13586*x2^3 + 1.6744*x1^2 - 0.41955*x1*x2 + 1.4875*x2^2
   - 0.49756*x1 + 0.50148*x2 + 1.2679

% Plot contours of unit disk and maximal ellipse
plotdomain = [-2 3 -2 2];
pcontour(g1,0,plotdomain,'b') hold on;
pcontour(g2,gamma,plotdomain,'r')
axis equal; axis(plotdomain)
```



Additional Set Containment Conditions

- There are algebraic geometry theorems (Stellensatz) which provide necessary and sufficient conditions for set containments involving polynomial constraints.
- These conditions are more complex than the polynomial S-procedure but they can be simplified to generate different sufficient conditions.
- For example, given g_0, g_1, $g_2 \in \mathbb{R}[x]$:
  1. Assume $g_0(x) > 0 \; \forall x \neq 0$ and $g_0(0) = 0$. If there exists $s_1, s_2 \in \Sigma[x]$ such that $-g_1 s_1 - g_0 + g_2 s_2 \in \Sigma[x]$ then $\{x \in \mathbb{R}^n \; : \; g_2(x) \leq 0\} \subseteq \{x \in \mathbb{R}^n \; : \; g_1(x) < 0\} \cup \{0\}$.
  2. Assume $g_0(x) > 0 \; \forall x \neq 0$ and $g_0(0) = 0$. Also assume $g_1(0) = 0$ and $g_1(x) < 0 \; \forall x \neq 0$ in a neighborhood of the origin. If there exists $r(x) \in \mathbb{R}[x]$ such that $-g_1 r + g_2 g_0 \in \Sigma[x]$ then $\{x \in \mathbb{R}^n \; : \; g_2(x) < 0\}_{cc} \subseteq \{x \in \mathbb{R}^n \; : \; g_1(x) < 0\} \cup \{0\}$.

Application of Set Containment Conditions (1)

Let $V,\ f \in \mathbb{R}[x]$. Assume that V is positive definite $\forall x$ and $\nabla V \cdot f$ is negative definite on a neighborhood of $x = 0$.

The following sets appear in ROA analysis:

$$\begin{aligned} \Omega_{V,\gamma} &:= \{x \in \mathbb{R}^n \ : \ V(x) \leq \gamma\} \\ (\Omega_{V,\gamma})_{cc} &:= \text{The connected component of } \Omega_{V,\gamma} \text{ containing } x = 0 \\ S &:= \{x \in \mathbb{R}^n \ : \ \nabla V \cdot f < 0\} \cup \{0\} \end{aligned}$$

In ROA analysis, we want to solve:

$$\max_{\gamma \in \mathbb{R}} \gamma \text{ s.t. } \Omega_{V,\gamma} \subseteq S$$

Application of Set Containment Conditions (2)

Assume $l(x) > 0 \; \forall x \neq 0$ and $l(0) = 0$.
The polynomial S-procedure and the two more general sufficient conditions can be applied to the ROA set containment problem:

1. $\Omega_{V,\gamma} \subseteq S$ if $\exists s \in \Sigma[x]$ such that $-(l + \nabla V \cdot f) + (V - \gamma)\, s \in \Sigma[x]$.
2. $\Omega_{V,\gamma} \subseteq S$ if $\exists s_1, s_2 \in \Sigma[x]$ such that $-\nabla V \cdot f s_1 - l + (V - \gamma) s_2 \in \Sigma[x]$.
3. $(\Omega_{V,\gamma})_{cc} \subseteq S$ if $\exists r \in \mathbb{R}[x]$ such that $-\nabla V \cdot f r + (V - \gamma) l \in \Sigma[x]$.

- Maximizing γ subject to constraints 1 or 2 requires a bisection on γ.
- Constraint 3 does not require a bisection on γ but the degree of the polynomial constraint is higher.
- If $s_1 = 1$, then constraint 2 reduces to constraint 1. In most cases, maximizing γ subject to constraint 1 achieves the same level set as maximizing subject to constraint 2.

Outline

- Motivation
- Preliminaries
- **ROA analysis using SOS optimization and solution strategies**
- Robust ROA analysis with parametric uncertainty
- Local input-output analysis
- Robust ROA and performance analysis with unmodeled dynamics
- F-18

Region of Attraction

Consider the autonomous nonlinear dynamical system

$$\dot{x}(t) = f(x(t))$$

where $x \in \mathbb{R}^n$ is the state vector and $f : \mathbb{R}^n \to \mathbb{R}^n$.
Assume:

- $f \in \mathbb{R}[x]$
- $f(0) = 0$, i.e. $x = 0$ is an equilibrium point.
- $x = 0$ is asymptotically stable.

Define the region of attraction (ROA) as:

$$\mathcal{R}_0 := \{\xi \in \mathbb{R}^n \ : \ \lim_{t \to \infty} \phi(\xi, t) = 0\}$$

where $\phi(\xi, t)$ denotes the solution at time t starting from the initial condition $\phi(\xi, 0) = \xi$.

Objective: Compute or estimate the ROA.

Global Stability Theorem

Theorem: Let $l_1, l_2 \in \mathbb{R}[x]$ satisfy $l_i(0) = 0$ and $l_i(x) > 0 \; \forall x$ for $i = 1, 2$. If there exists $V \in \mathbb{R}[x]$ such that:

- $V(0) = 0$
- $V - l_1 \in \Sigma[x]$
- $-\nabla V \cdot f - l_2 \in \Sigma[x]$

Then $\mathcal{R}_0 = \mathbb{R}^n$.

Proof:

- The conditions imply that V and $-\nabla V \cdot f$ are positive definite.
- V is a positive definite polynomial and hence it is both decrescent and radially unbounded.
- It follows from Theorem 56 in Vidyasagar that $x = 0$ is globally asymptotically stable (GAS) and $\mathcal{R}_0 = \mathbb{R}^n$.
- V is a Lyapunov function that proves $x = 0$ is GAS.

Reference: Vidyasagar, M., *Nonlinear Systems Analysis*, SIAM, 2002.
(Refer to Section 5.3 for theorems on Lyapunov's direct method.)

Global Stability via SOS Optimization

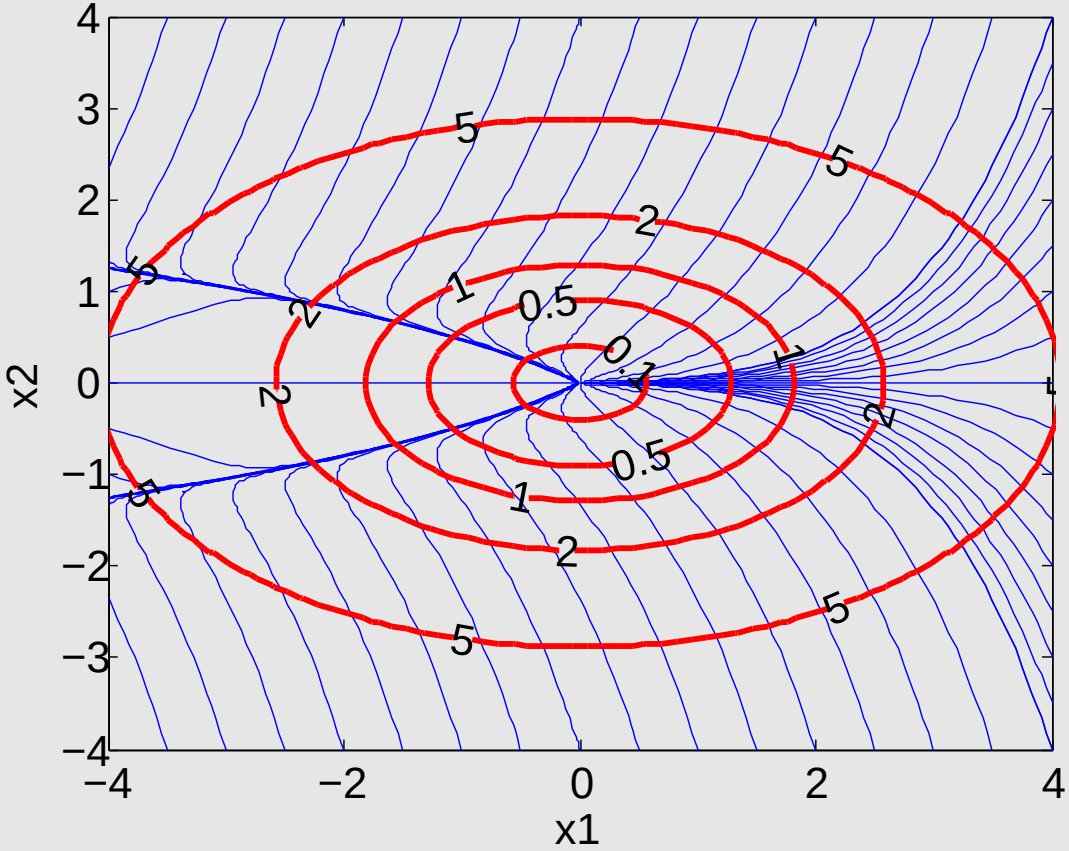
- We can search for a Lyapunov function V that proves $x = 0$ is GAS. This is an SOS feasibility problem.
- Implementation:
  - V is a polynomial decision variable in the optimization and the user must select the monomials to include.
    - V can not include constant or linear terms.
    - A good (generic) choice for V is to include all monomials from degree 2 up to d_{max}:

```
V = polydecvar('c',monomials(x,2:dmax,'vec');
```

  - l_1 and l_2 can usually be chosen as $\epsilon \sum_{i=1}^{n} x_i^{d_{min}}$ where d_{min} is the lowest degree of terms in V, e.g. $l_i = \epsilon x^T x$ for $d_{min} = 2$.
- The theorem only provides sufficient conditions for GAS.
  - If feasible, then V proves $\mathcal{R}_0 = \mathbb{R}^n$.
  - If infeasible, then additional monomials can be included in V and the the SOS feasibility problem can be re-solved.
  - If $x = 0$ is not GAS then the conditions will always be infeasible. A local stability analysis is needed to estimate $\mathcal{R}_0$.

Global Stability Example with sosopt

```
% Code from Parrilo1_GlobalStabilityWithVec.m

% Create vector field for dynamics
pvar x1 x2;
x = [x1;x2];
x1dot = -x1 - 2*x2^2;
x2dot = -x2 - x1*x2 - 2*x2^3;
xdot = [x1dot; x2dot];

% Use sosopt to find a Lyapunov function
% that proves x = 0 is GAS

% Define decision variable for quadratic
% Lyapunov function
zV = monomials(x,2);
V = polydecvar('c',zV,'vec');

% Constraint 1 : V(x) - L1 \in SOS
L1 = 1e-6 * ( x1^2 + x2^2 );
sosconstr{1} = V - L1;

% Constraint 2: -Vdot - L2 \in SOS
L2 = 1e-6 * ( x1^2 + x2^2 );
Vdot = jacobian(V,x)*xdot;
sosconstr{2} = -Vdot - L2;

% Solve with feasibility problem
[info,dopt,sossol] = sosopt(sosconstr,x);
Vsol = subs(V,dopt)
Vsol =
  0.30089*x1^2 + 1.8228e-017*x1*x2 + 0.6018*x2^2
```



polydecvar Implementation of V

- In the previous example, we enforced $V(x) > 0 \; \forall x$ by using a vector form decision variable and constraining $V - l_1 \in \Sigma[x]$:

```
zV = monomials(x,2);
V = polydecvar('c',zV,'vec');
L1 = 1e-6 * ( x1^2 + x2^2 );
sosconstr{1} = V - L1;
```

- sosopt introduces a Gram matrix variable for this constraint in addition to the coefficient decision variables in V.
- A more efficient implementation is obtained by defining the positive semidefinite part of V using the matrix form decision variable:

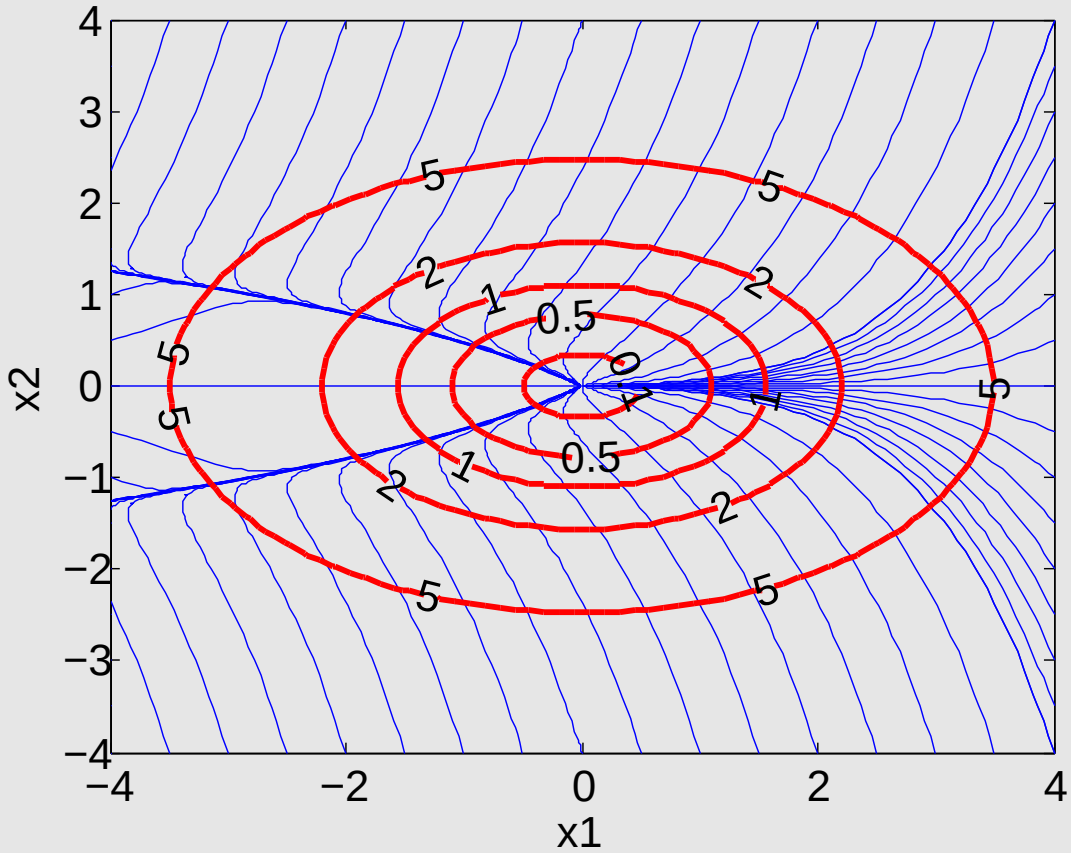
```
zV = monomials(x,1);
S = polydecvar('c',zV,'mat');
L1 = 1e-6 * ( x1^2 + x2^2 );
V = S + L1;
```

- In this implementation, the coefficient decision variables are the entries of the Gram matrix of S. These Gram matrix of S is directly constrained to be positive semidefinite by sosopt and no additional variables are introduced.

Global Stability Example with mat Implementation

```
% Code from Parrilo2_GlobalStabilityWithMat.m

% Create vector field for dynamics
pvar x1 x2;
x = [x1;x2];
x1dot = -x1 - 2*x2^2;
x2dot = -x2 - x1*x2 - 2*x2^3;
xdot = [x1dot; x2dot];

% Use sosopt to find a Lyapunov function
% that proves x = 0 is GAS

% Use 'mat' option to define psd
% part of quadratic Lyapunov function
zV = monomials(x,1);
S = polydecvar('c',zV,'mat');
L1 = 1e-6 * ( x1^2 + x2^2 );
V = S + L1;

% Constraint 1 : S \in SOS
sosconstr{1} = S;

% Constraint 2: -Vdot - L2 \in SOS
L2 = 1e-6 * ( x1^2 + x2^2 );
Vdot = jacobian(V,x)*xdot;
sosconstr{2} = -Vdot - L2;

% Solve with feasibility problem
[info,dopt,sossol] = sosopt(sosconstr,x);
Vsol = subs(V,dopt)
Vsol =
  0.40991*x1^2 + 2.4367e-015*x1*x2 + 0.81986*x2^2
```

This implementation has three fewer decision variables (the vector form coefficients of V are not needed) and sosopt finds the same V to within a scaling.



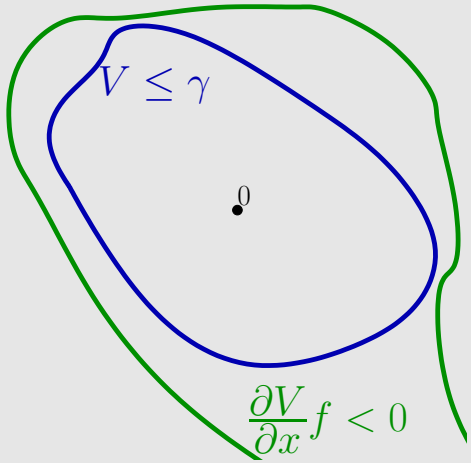
Local Stability Theorem

Theorem: Let $l_1 \in \mathbb{R}[x]$ satisfy $l_1(0) = 0$ and $l_1(x) > 0 \; \forall x$.
If there exists $V \in \mathbb{R}[x]$ such that:

- $V(0) = 0$
- $V - l_1 \in \Sigma[x]$
- $\Omega_{V,\gamma} := \{x \in \mathbb{R}^n : V(x) \leq \gamma\} \subseteq \{x \in \mathbb{R}^n : \nabla V \cdot f < 0\} \cup \{0\}$

Then $\Omega_{V,\gamma} \subseteq \mathcal{R}_0$.

Proof: The conditions imply that $\Omega_{V,\gamma}$ is bounded and hence the result follows from Lemma 40 in Vidyasagar.



Local Stability via SOS Optimization

Idea: Let $\dot{x} = Ax$ be the linearization of $\dot{x} = f(x)$. If A is Hurwitz then a quadratic Lyapunov function shows that $x = 0$ is locally asymptotically stable. Use the polynomial S-procedure to verify a quantitative estimate.

1. Select $Q \in \mathcal{S}^{n \times n}$, $Q > 0$ and compute $P > 0$ that satisfies the Lyapunov Equation: $A^T P + PA = -Q$
   - $V_{lin}(x) = x^T P x$ is a quadratic Lyapunov function proving $x = 0$ is locally asymptotically stable.
   - This step can be done with: `[Vlin,A,P]=linstab(f,x)`

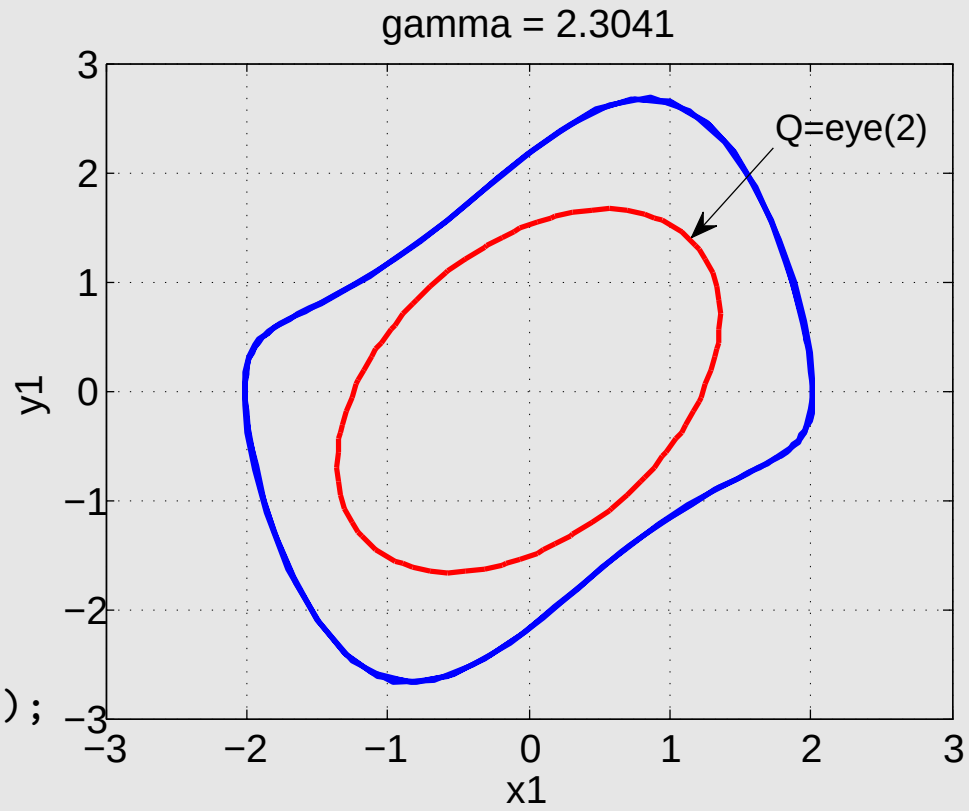
2. Define $l_2 \in \mathbb{R}[x]$ such that $l_2(0) = 0$ and $l_2(x) > 0 \; \forall x$. Solve the set containment maximization problem using `pcontain`:

$$\max_{\gamma \in \mathbb{R}} \gamma \text{ subject to } \; \Omega_{V,\gamma} \subset \{x \in \mathbb{R}^n : \nabla V_{lin} \cdot f - l_2 \leq 0\}$$

Example: ROA Estimate for the Van der Pol Oscillator (1)

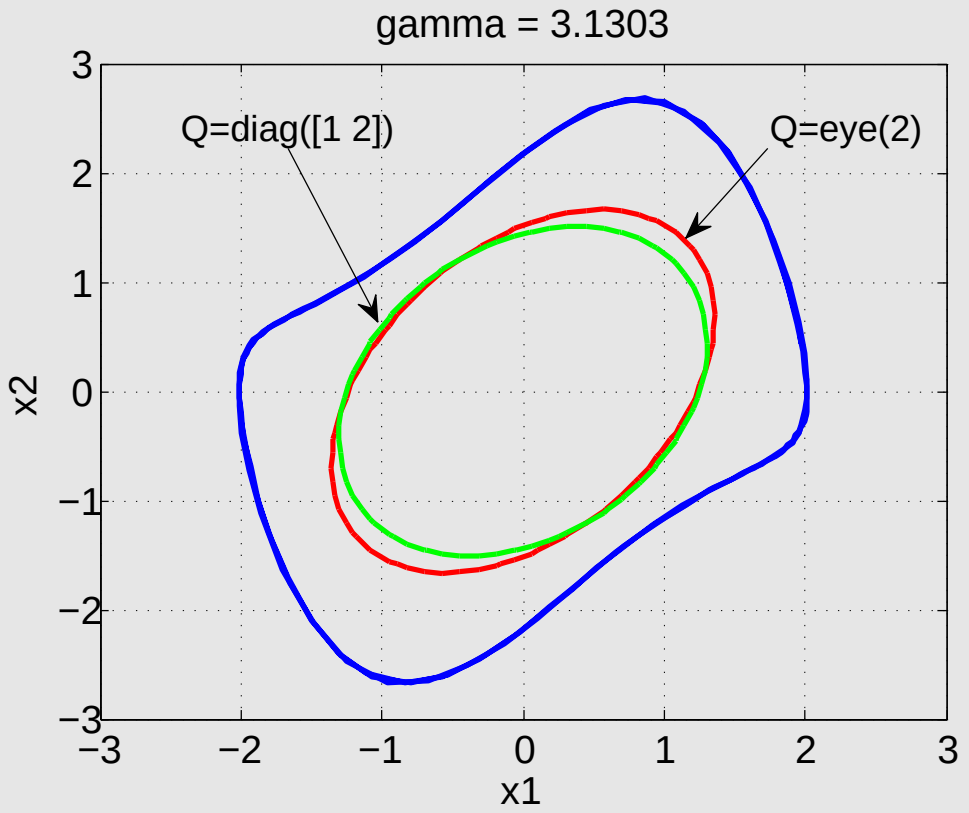
```
% Code from VDP_LinearizedLyap.m

% Vector field for VDP Oscillator
pvar x1 x2;
x = [x1;x2];
x1dot = -x2;
x2dot = x1+(x1^2-1)*x2;
f = [x1dot; x2dot];

% Lyap fnc from linearization
Q = eye(2);
Vlin = linstab(f,x,Q);

% maximize gamma
% subject to:
% {Vlin<=gamma} in {Vdot<0} U {x=0}
z = monomials(x, 1:2 );
L2 = 1e-6*(x'*x);
Vdot = jacobian(Vlin,x)*f;
[gbnds,s] = pcontain(Vdot+L2,Vlin,z);
Gamma = gbnds(1)
```

$$\dot{x}_1 = -x_2$$
$$\dot{x}_2 = x_1 + (x_1^2 - 1)x_2$$



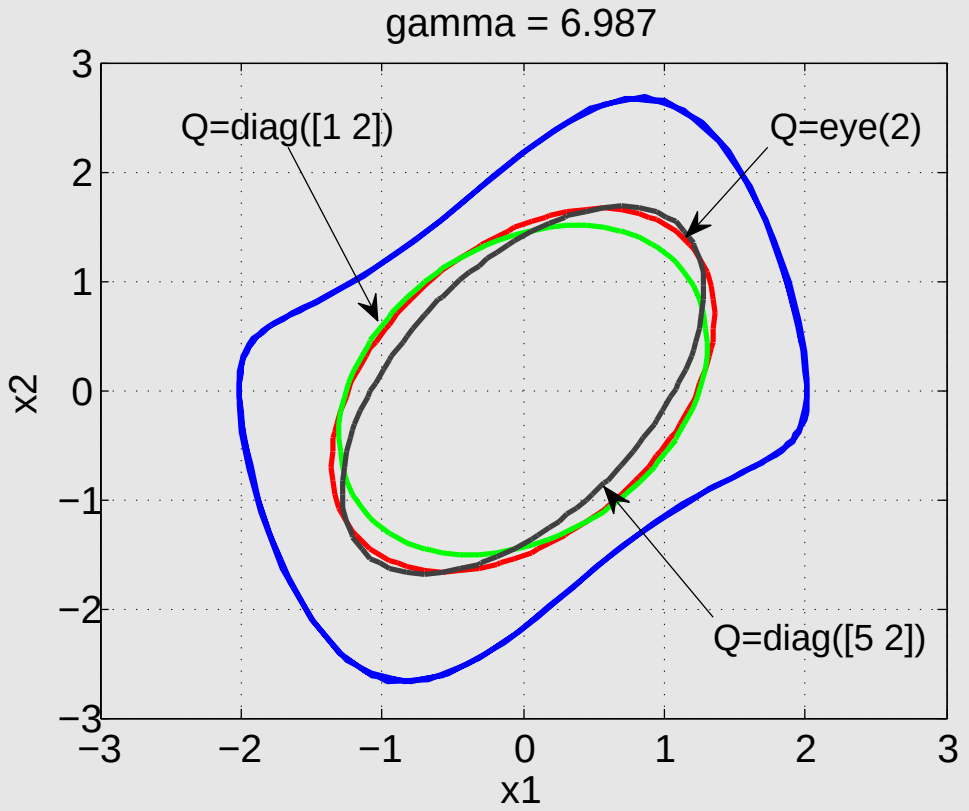
Example: ROA Estimate for the Van der Pol Oscillator (2)

Choosing $Q = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 2 \end{smallmatrix}\right]$ slightly increases $\Omega_{V,\gamma}$ along one direction but decreases it along another.



Example: ROA Estimate for the Van der Pol Oscillator (3)

Choosing $Q = \left[\begin{smallmatrix} 5 & 0 \\ 0 & 2 \end{smallmatrix}\right]$ has the opposite effect on $\Omega_{V,\gamma}$.



Increasing the ROA Estimate

For this problem, `pcontain` solves:

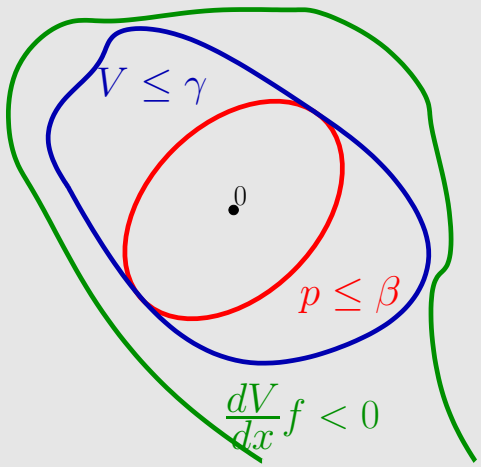
$$\max_{\gamma \in \mathbb{R}, s \in \Sigma[x]} \gamma$$
$$\text{s.t.:} \quad -\left(\nabla V_{lin} \cdot f + l_2 + s(\gamma - V_{lin})\right) \in \Sigma\left[x\right]$$

Objective: Increase the "size" of $\Omega_{V,\gamma}$ subject to the same constraints by searching over quadratic or higher degree Lyapunov functions.

Question: How should we measure the "size" of the ROA estimate?

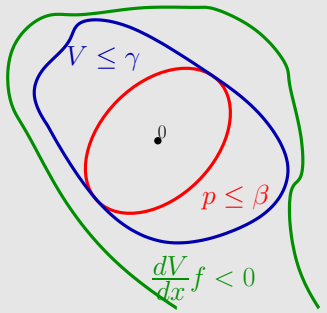
Approach:
Introduce a shape factor p which:

- is a positive definite polynomial
- captures the intent of the analyst
- (preferably) has simple sublevel sets



Interpretation of Shape Function p

$$\Omega_{p,\beta} \subseteq \Omega_{V,\gamma} \subseteq \{\nabla V \cdot f(x) < 0\} \cup \{0\}$$



- $\Omega_{p,\beta} := \{x : p(x) \leq \beta\}$ is a subset of ROA
  - p simple $\Rightarrow \Omega_{p,\beta}$ is simple
  - $\Omega_{p,\beta}$ is not an invariant set.
  - This skews the analysis in the directions implied by level sets of p.
  - This potentially misses out on other areas in the ROA
- $\Omega_{V,\gamma} := \{x : V(x) \leq \gamma\}$ is an invariant subset of ROA
  - V is chosen by optimization over a rich class of functions.
  - V is not simple $\Rightarrow \Omega_{V,\gamma}$ is unclear and additional analysis is needed to understand.
- p scalarizes the problem with β as the cost function
  - The analyst picks p to reflect a particular objective.
  - The methodology skews its goals towards this objective.
  - The methodology offers no guidelines as to the appropriateness of p.

Increasing the ROA Estimate

We increase the ROA estimate by increasing the shape function contained with a Lyapunov level set.

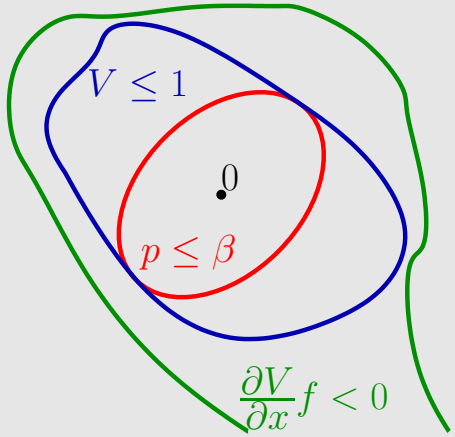
$$\beta^* = \max_{V \in \mathbb{R}[x],\ \beta \in \mathbb{R}} \beta$$

subject to:

$$\Omega_{p,\beta} \subseteq \Omega_{V,1}$$

$$\Omega_{V,1} \subseteq \{\nabla V \cdot f(x) < 0\} \cup \{0\}$$

$$V - l_1 \in \Sigma\left[x\right], V(0) = 0$$



How are the set contaiments verified?

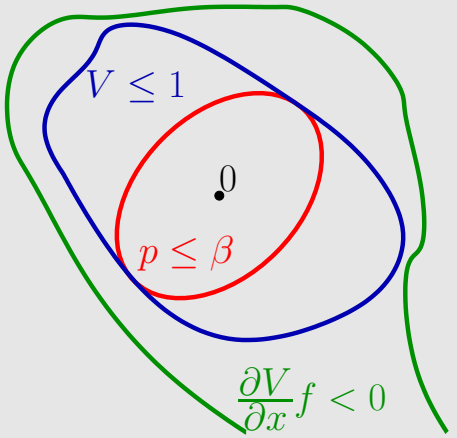
Increasing the ROA Estimate

Applying the polynomial S-procedure to both set containment conditions gives:

$$\max_{s_1, s_2 \in \Sigma[x],\ V \in \mathbb{R}[x],\ \beta \in \mathbb{R}} \beta$$

subject to:

$$-((V-1) + s_1(\beta - p)) \in \Sigma[x]$$
$$-((\nabla V \cdot f + l_2) + s_2(1-V)) \in \Sigma[x]$$
$$V - l_1 \in \Sigma[x], V(0) = 0$$



This is not an SOS programming problem since the first constraint is bilinear in variables s_1 and β and the second constraint is bilinear in variables s_2 and V.

The second constraint can be replaced by the alternative set containment condition (introducing an additional multiplier $s_3 \in \Sigma[x]$):

$$-((\nabla V \cdot f)s_3 + l_2 + s_2(1-V)) \in \Sigma[x]$$

Properties of Bilinear ROA SOS Conditions

Several properties of this formulation are presented in the following slides,

- Example with known ROA (from *Davison, Kurak, 1971*)
- Comparison with linearized analysis
- Non-convexity of local analysis conditions

Methods to solve the Bilinear ROA SOS problem will be presented after discussing these properties of the formulation.

System with known ROA (from *Davison, Kurak, 1971*)

For a positive definite matrix B,

$$\left[\dot{x} = -x + (x^T B x)x\right]_{\mathrm{ROA}_0} = \left\{x : x^T B x < 1\right\}$$

Proof: $V(x) := x^T B x$. Then $\dot{V} = 2V(V-1), \ldots$

For positive-definite, quadratic shape factor $p(x) := x^T R x$,

$$\frac{1}{\lambda_{\max}\left(R^{-1}B\right)} = \sup \beta \;\text{ s.t. } \left\{x : x^T R x \leq \beta\right\} \subset \left\{x : x^T B x < 1\right\}$$

Can the bilinear SOS formulation yield this?

- Yes (Tan thesis), any β less than supremum
  1. choose $\gamma > 1$ and any $1 < \tau < \gamma$
  2. define $V := \gamma x^T B x$
  3. for large enough α, the choices $s_2 := 2\alpha\tau x^T B x, s_3 := \alpha$ work.

Linear versus SOS-based nonlinear analysis

SOS-based nonlinear analysis

- Question: Given the shape factor p, what is the largest value of β such that $\{x \ : \ p(x) \leq \beta\}$ is in the ROA?
- Analysis method: a series of (potentially conservative) relaxations/reformulations
  - Lyapunov-type/dissipation inequality sufficient conditions
  - Finite parameterizations for the certificates
  - S-procedure
  - SOS relaxations
  - Non-convex optimization problems (BMIs)

Linearization based analysis

- Question: Is the equilibrium point asymptotically stable?
- Analysis method:
  - Linearize the dynamics
  - The system is asymptotically stable if and only if the linearization is asymptotically stable.
  - Determined through eigenanalysis - no conservatism involved.

Quantitative improvement on linearized analysis

Consider systems with **cubic** vector fields

$$\dot{x} = Ax + f_{23}(x)$$

where A is Hurwitz, and f_{23} contains only quadratic and cubic terms (so $f_{23}(0) = 0$).

Standard Analysis: "$\exists$ an open ball around origin in ROA"

SOS formulation: The SOS problem is always feasible with $\partial(V) = \partial(s_2) = 2$ and $\partial(s_1) = \partial(s_3) = 0$.

Precisely, given p, f_{23}, $l_i(x) := x^T R_i x$, if A is Hurwitz, then there exist $\gamma > 0$, $\beta > 0$, $s_1,\ s_2,\ s_3$, and V feasible for

$$\begin{gathered} V - l_1 \in \Sigma[x],\ V(0) = 0,\ s_1, s_2, s_3 \in \Sigma[x], \\ -\left[(\beta - p)s_1 + (V - \gamma)\right] \in \Sigma[x] \\ -\left[(\gamma - V)s_2 + \nabla V f s_3 + l_2\right] \in \Sigma[x]. \end{gathered}$$

The proof is constructive.

Construction of V and multipliers

- Let $\tilde{Q} \succ 0$ satisfy $A^T\tilde{Q} + \tilde{Q}A \preceq -2R_2$ and $\tilde{Q} \succeq R_1$.
- $V(x) := x^T\tilde{Q}x$.
- $\epsilon := \lambda_{min}(R_2)$.
- Let $H \succ 0$ be such that $(x^Tx)V(x) = \mathbf{z}^THz$ (where $\mathbf{z}$ is a vector of monomials of the form x_ix_j with no repetition).
- Let $M_2 \in \mathcal{R}^{n\times n_z}$ and symmetric $M_3 \in \mathcal{R}^{n_z\times n_z}$ satisfy $\nabla V f_2(x) = x^TM_2\mathbf{z}$ and $\nabla V f_3(x) = \mathbf{z}^TM_3\mathbf{z}$.
- Let M_3^+ be the positive semidefinite part of M_3, define

$$\begin{aligned}
s_1(x) &:= \frac{\lambda_{max}(\tilde{Q})}{\lambda_{min}(P)} \\
c_2 &:= \frac{\lambda_{max}\left(M_3^+ + \frac{1}{2\epsilon}M_2^TM_2\right)}{\lambda_{min}(H)} \\
s_2(x) &:= c_2x^Tx \\
\gamma &:= \frac{\epsilon}{2c_2} \\
\beta &:= \frac{\gamma}{2s_1} \\
s_3(x) &:= 1
\end{aligned}$$

Implications of the constructive proof

The construction provides suboptimal, computationally less demanding, yet less conservative (compared to straight linear analysis) solution techniques.

$$\max_{\gamma, c_2, \beta, Q=Q^T \succeq R_1} \beta \quad \text{subject to}$$

$$\begin{bmatrix} -\gamma c_2 I - R_2 - A^T Q - QA & -M_2(Q)/2 \\ -M_2(Q)^T/2 & c_2 H(Q) - M_3(Q) \end{bmatrix} \succeq 0$$

$$\begin{bmatrix} -\beta + \gamma & 0 \\ 0 & P - Q \end{bmatrix} \succeq 0.$$

Results for the VDP dynamics (with $p(x) = x^T x$):

- $\beta = 0.2$ for V from linear analysis (i.e., $V(x) = x^T Q x$ such that $Q \succ 0$ and $A^T Q + QA = -I$)
- $\beta = 0.7$ by the suboptimal technique
- $\beta = 1.54$ "optimal" value for quadratic V

Non-convexity of Local Analysis (1)

This bilinearity of the local stability analysis is an effect of the non-convexity of local stability constraints.
This contrasts with the convexity of global analysis.

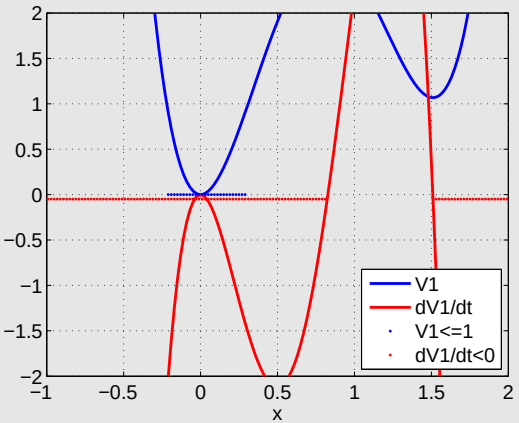
Global Analysis: The set of functions $V : \mathbb{R}^n \to \mathbb{R}$ that satisfy $V(0) = 0$, $V(x) > 0 \ \forall x \neq 0$, and $\nabla V(x) \cdot f(x) < 0 \ \forall x \neq 0$ is a convex set.

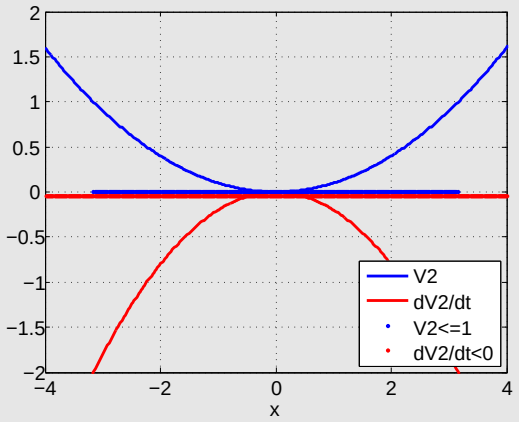
Proof: If V_1 and V_2 satisfy the global analysis constraints then $\lambda V_1 + (1 - \lambda)V_2$ is also feasible for any $\lambda \in [0, 1]$,

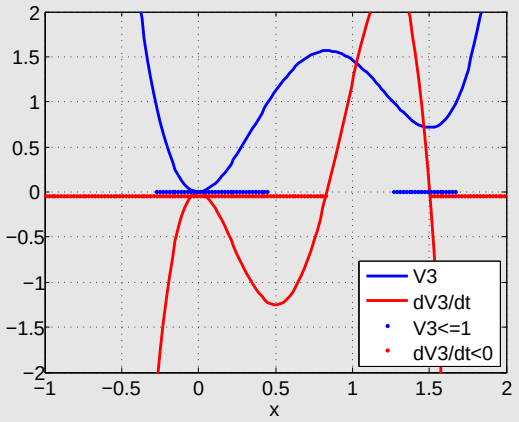
Non-convexity of Local Analysis (2)

Local Analysis: The set of functions $V : \mathbb{R}^n \to \mathbb{R}$ that satisfy $V(0) = 0$, $V(x) > 0 \; \forall x \neq 0$, and $\Omega_{V,\gamma=1} \subseteq \{\nabla V(x) \cdot f(x) < 0\} \cup \{0\}$ is a NOT convex set.

Example: Let $f(x) = -x$ and define $V_1(x) = 16x^2 - 19.95x^3 + 6.4x^4$ and $V_2(x) = 0.1x^2$
V_1 and V_2 satisfy the local analysis constraints but their convex combination $V_3 := 0.58V_1 + 0.42V_2$ does not.







Solving the Bilinear ROA Problem

A coordinate-wise V-s iteration is a simple algorithm to find a sub-optimal solution to this optimization.

- For fixed V, the constraints decouple into two subproblems

$$\begin{aligned}\gamma^* &= \max_{\gamma\in\mathbb{R}, s_2\in\Sigma[x]} \gamma \quad \text{s.t.} \quad -((\nabla V\cdot f + l_2) + s_2(1-V)) \in \Sigma[x] \\ &\leq \max_{\gamma\in\mathbb{R}} \gamma \quad \text{s.t.} \quad \Omega_{V,\gamma} \subseteq \{\nabla V\cdot f(x) < 0\}\cup\{0\} \\ \beta^* &= \max_{\beta\in\mathbb{R}, s_1\in\Sigma[x]} \beta \quad \text{s.t.} \quad -((V-\gamma^*) + s_1(\beta - p)) \in \Sigma[x] \\ &\leq \max_{\beta\in\mathbb{R}} \beta \quad \text{s.t.} \quad \Omega_{p,\beta} \subseteq \Omega_{V,\gamma^*}\end{aligned}$$

  `pcontain` can be used to compute γ^* and β^* as well as multipliers s_1 and s_2.

- For fixed s_1 and s_2, we could maximize β with V subject to the local ROA constraints. We obtain better results by re-centering V to the analytic center of the LMI associated with:

$$\begin{aligned}&-((V-1) + s_1(\beta^* - p)) \in \Sigma[x] \\ &-((\nabla V\cdot f + l_2) + s_2(\gamma^* - V)) \in \Sigma[x] \\ &V - l_1 \in \Sigma[x], V(0) = 0\end{aligned}$$

V-step as a feasibility problem

- An informal justification for the LMI re-centering in the V-step is:
  - The constraint $-\left(\nabla V \cdot f + l_2 + s_2(\gamma - V)\right) \in \Sigma\left[x\right]$ is active after the γ-step.
  - In the V-step, compute the analytic center of the LMI constraints to obtain a new feasible V. Thus the V-step feasibility problem pushes V away from the constraint.
  - Loosely, this finds V that satisfies:

$$-\left(\nabla V \cdot f + \tilde{l}_2 + s_2(\gamma - V)\right) \in \Sigma\left[x\right]$$

    where $\tilde{l}_2 \geq l_2$.
  - This means that $\Omega_{V,\gamma} \subseteq \{x \in \mathbb{R}^n \ : \ \dot{V} < -\tilde{l}_2\}$.
  - $\tilde{l}_2 \geq l_2$ would mean the next γ-step has freedom to increase γ while still satisfying the constraint with l_2.
- This feasibility step is not guaranteed to increase γ or β over each step but it typically makes an improvement.
- A more formal theory for the behavior of this feasibility step is still an open question.

Implementation Issue: Scaling of V

- If $l_2 = 0$ and $(V, \gamma^*, \beta^*, s_1, s_2)$ satisfy the local ROA constraints then $(cV, c\gamma^*, \beta^*, cs_1, s_2)$ are also feasible for any $c > 0$.
- The solution can still be scaled by some amount if l_2 is a small positive definite function.
- As a result, the scaling of V tends to drift during the V-s iteration such that larger values of γ^* are returned at each step.
- This makes it difficult to pre-determine a reasonable upper bound on γ^* for the bisection in the γ-step.
- Scaling V by γ^* after each V step roughly normalizes V. This tends to keep the γ^* computed in the next γ-step close to unity.

Implementation Issue: Constraint on s_2

The multiplier $s_2 \in \Sigma[x]$ appears in the constraint:

$$-((\nabla V \cdot f + l_2) + s_2(1 - V)) \in \Sigma[x]$$

Since $f(0) = 0$, $l_2(0) = 0$, and $V(0) = 0$, evaluating this constraint at $x = 0$ gives:

$$-s_2(0) \geq 0$$

$s_2 \in \Sigma[x]$ implies the reverse inequality:

$$s_2(0) \geq 0$$

- Hence, the constant term of the multiplier s_2 must be zero.
- The solvers can have difficulty resolving this implicit equality constraint, so this degree of freedom should be removed by directly parameterizing s_2 to have zero constant term.
- This type of analysis must be done on all SOS constraints.

Complete ROA V-s Iteration

Initialization: Find $V(x)$ which proves local stability in some neighborhood of $x = 0$.

1. γ Step: Hold $V(x)$ fixed and use `pcontain` to solve for s_2:

$$\gamma^* = \max_{s_2 \in \Sigma[x], \gamma \in \mathbb{R}} \gamma \quad \text{s.t. } -(\nabla V \cdot f + l_2 + s_2(\gamma - V)) \in \Sigma[x]$$

2. β Step: Hold $V(x)$ fixed and use `pcontain` to solve for s_1:

$$\beta^* = \max_{s_1 \in \Sigma[x], \beta \in \mathbb{R}} \beta \quad \text{s.t. } -((V - \gamma) + s_1(\beta - p)) \in \Sigma[x]$$

3. V step: Hold $s_1, s_2, \beta^*, \gamma^*$ fixed and compute V from the analytic center of:

$$-\left(\frac{\partial V}{\partial x} f + l_2 + s_2(\gamma - V)\right) \in \Sigma[x]$$
$$-((V - \gamma) + s_1(\beta - p)) \in \Sigma[x]$$
$$V - l_1 \in \Sigma[x], V(0) = 0$$

4. V Scaling: Replace V with $\frac{V}{\gamma^*}$.
5. Repeat

System with known ROA (from *Davison, Kurak, 1971*)

For a positive definite matrix B,

$$\left[\dot{x} = -x + (x^T B x)x\right]_{\mathrm{ROA}_0} = \left\{x : x^T B x < 1\right\}$$

For positive-definite, quadratic shape factor $p(x) := x^T R x$,

$$\frac{1}{\lambda_{\max}\left(R^{-1}B\right)} = \sup \beta \;\text{ s.t. } \left\{x : x^T R x \leq \beta\right\} \subset \left\{x : x^T B x < 1\right\}$$

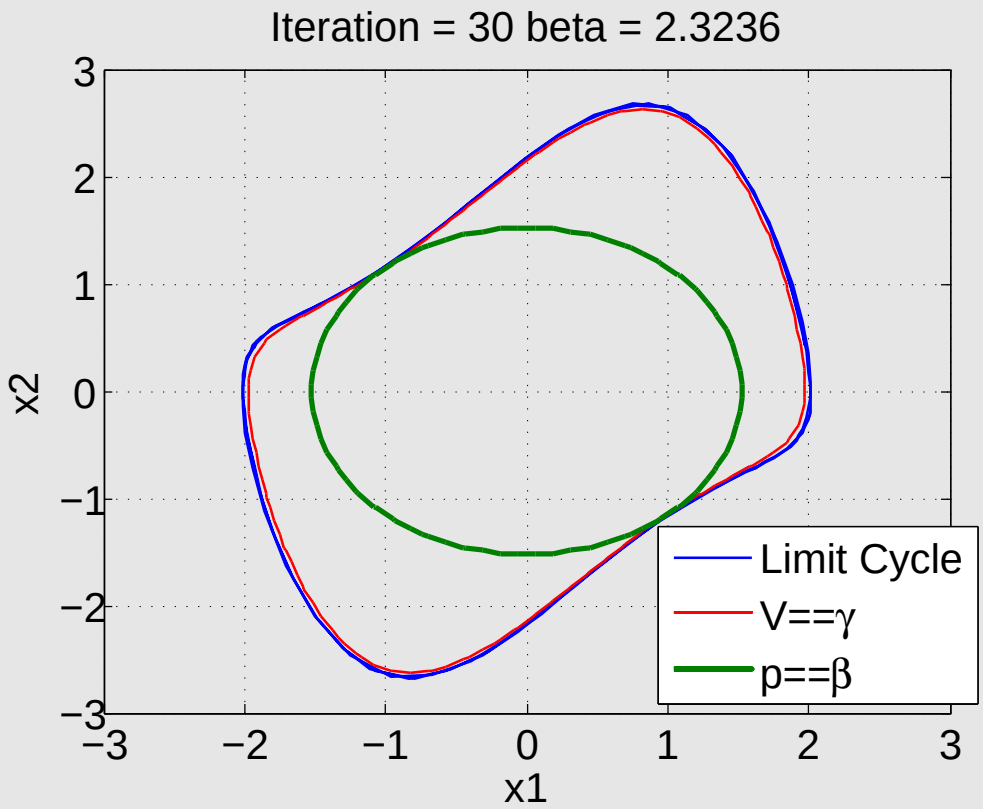
Can the iteration proposed find this solution?

- Yes, thousands of examples on $n \leq 10$
- Relatively fast, see `radialvectorfield.m`

But, not all problems work so nicely...

Example: V-s Iteration for the Van der Pol Oscillator

```
% Code from VDP_IterationWithVlin.m
pvar x1 x2;
x = [x1;x2];
x1dot = -x2;
x2dot = x1 + (x1^2-1)*x2;
f = [x1dot; x2dot];

% Create shape function and monomials vectors
p = x'*x;
zV = monomials( x, 2:6 );    % V has Deg = 6
z1 = monomials( x, 0:2 );
z2 = monomials( x, 1:2 );
L2 = 1e-6*(x'*x);

% Initialize Lyapunov Function
V = linstab(f,x);

% Run V-s iteration
opts.L2 = L2;
for i1=1:30;
    % gamma step
    Vdot = jacobian(V,x)*f;
    [gbnds,s2] = pcontain(Vdot+L2,V,z2,opts);
    gamma = gbnds(2);

    % beta step
    [bbnds,s1] = pcontain(V-gamma,p,z1,opts);
    beta = bbnds(1)

    % V step (then scale to roughly normalize)
    if i1~=30
        V = roavstep(f,p,x,zV,beta,gamma,s1,s2,opts);
        V = V/gamma;
    end
end
```

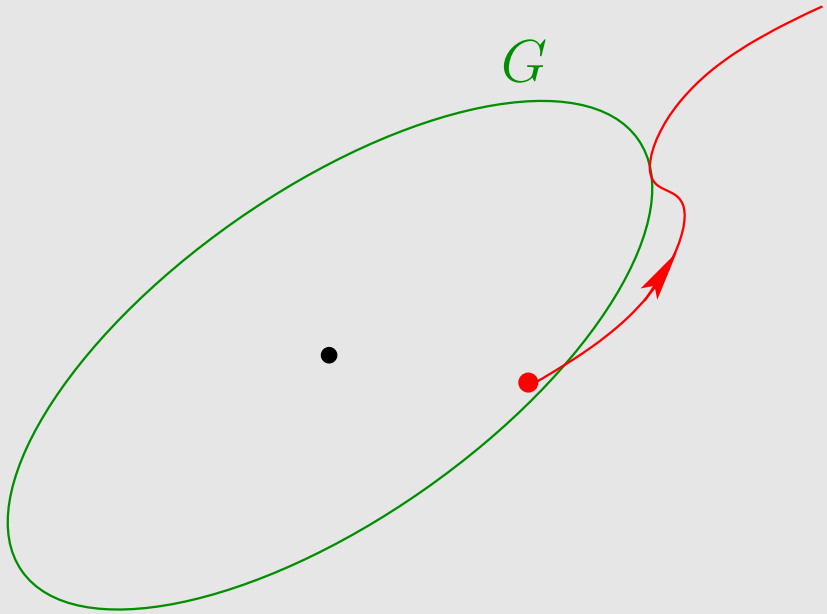


Use of Simulation Data

- The performance of the V-s iteration depends on the initial choice for V.
- Up to this point we have only started the iteration using the Lyapunov function obtained from linear analysis.
- It is also possible to used simulation data to construct initial Lyapunov function candidates for the iteration.
- The following slides explore this use of simulation data.

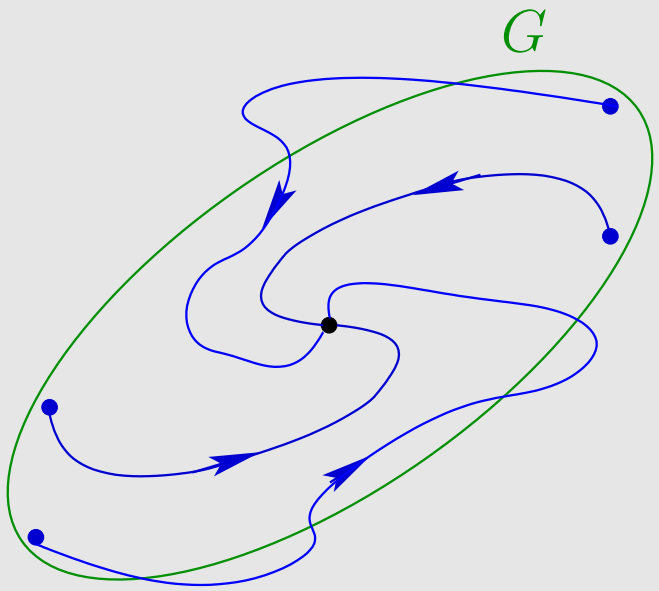
Use of Simulation Data

- Given a set G, is $G \subset \text{ ROA }$?
- Run simulations starting in G.
- If any diverge, no.



Use of Simulation Data

- Given a set G, is $G \subset$ ROA ?
- Run simulations starting in G.
- If any diverge, no.
- If all converge, "maybe yes."



Fact: A Lyapunov certificate would remove the "maybe".

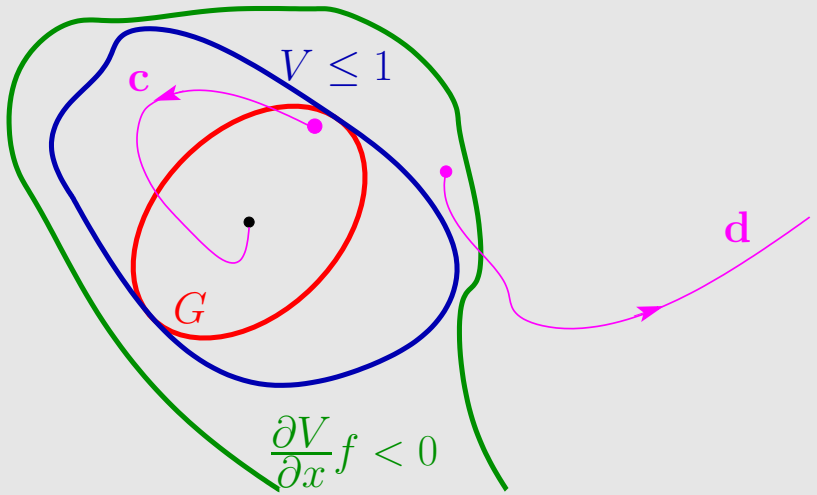
$$G \in \Omega_{V,\gamma=1} \subseteq \{x \in \mathbb{R}^n \ : \ \nabla V(x) \cdot f(x) < 0\}$$

Question: Can we use the simulation data to construct candidate Lyapunov functions for assessing the ROA?

How can the simulation data be used?

If there exists V to certify that G is in the ROA through Lyapunov arguments, it is **necessary** that

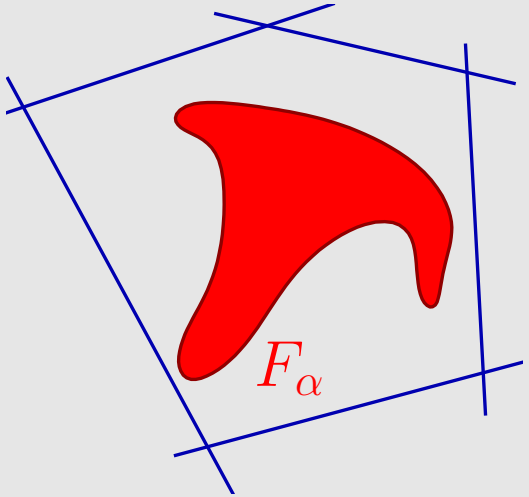
- $V > 0$
- $V \leq 1$ on converging trajectories starting in G
- $\dot{V} < 0$ on converging trajectories starting in G
- $V > 1$ on non-converging trajectories starting in the complement of G



The V we are looking for (which may not even exist) **must** satisfy these constraints.

Simulation-based constraints on V

- Assume V is linearly parameterized in some basis functions $V(x) = \alpha^T \phi(x)$, e.g. $\phi(x)$ can be a vector of monomials.
- Let F_α denote the set of coefficients α of Lyapunov functions which satisfy the constraints on some domain in the state space.
- Enforcing the constraints on the previous slide on the simulation trajectory points leads to LP constraints on α.
- The collection of the LP constraints forms a polytope outer bound on the set F_α of coefficients.



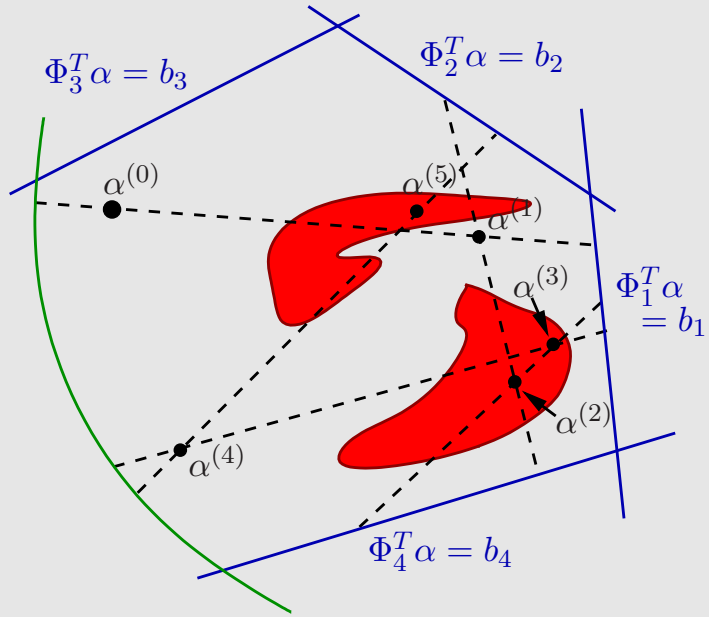
Set of Candidate V's

- We can sample the polytope outer bound of F_α by solving an LP feasibility problem.
  - If the LP is infeasible then F_α is empty.
  - If the LP is feasible then we can test if $V = \alpha^T \phi$ is a Lyapunov function using SOS optimization methods.
- We can incorporate additional convex constraints on α
  - $V - l_1 \in \Sigma[x] \Rightarrow$ LMI constraints on α
  - The linear part of f and quadratic part of V must satisfy the Lyapunov inequality $\Rightarrow$ LMI constraints on α.
- Let $\mathcal{Y}$ denote the set of α which satisfy the LP constraints from simulation data and the LMI constraints described above.

Hit-and-run (H&R) algorithm

• As the number of constraints increases, the outer convex set $\mathcal{Y}$ becomes a tighter relaxation.

$\Rightarrow$ Samples from $\mathcal{Y}$ become more likely to be in F_α.



• Strategy: generate points in $\mathcal{Y}$, i.e., Lyapunov function candidates, and evaluate β they certify.

• Generation of each point $\mathcal{Y}$ (after the initial feasible point) involves solving 4 small LMIs and trivial manipulations.

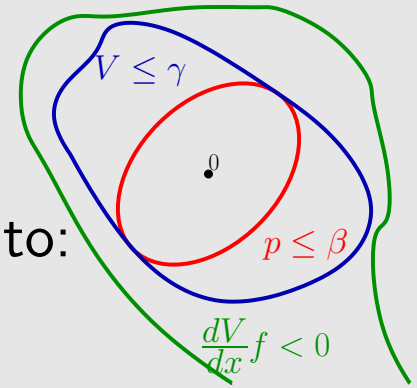
$$\bar{t}^{(k)} := \min \left\{ \max_j \left\{ 0, \frac{b_j - \Phi_j^T \alpha^{(k)}}{\Phi_j^T \zeta^{(k)}} \right\}, \bar{t}_{SOS}^{(k)}, \bar{t}_{lin}^{(k)} \right\},$$

$$\underline{t}^{(k)} := \max \left\{ \min_j \left\{ 0, \frac{b_j - \Phi_j^T \alpha^{(k)}}{\Phi_j^T \zeta^{(k)}} \right\}, \underline{t}_{SOS}^{(k)}, \underline{t}_{lin}^{(k)} \right\},$$

Assessing the candidate: checking containments

For a given V,

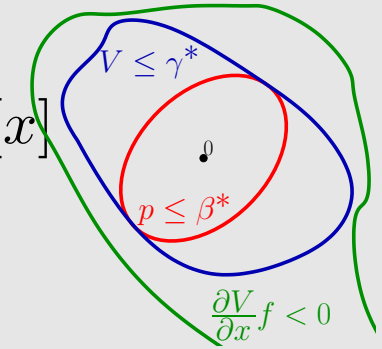
$$\beta_V := \max_{\beta,\gamma} \quad \beta \text{ subject to:}$$



This can be solved in two steps solving **smaller "affine"** SDPs sequentially:

$$\gamma^* := \quad \max \ \gamma$$
$$-\left[(\gamma - V)s_2 + s_3 \frac{dV}{dx} f + l_2\right] \in \Sigma[x]$$

$$\beta_V := \quad \max \ \beta$$
$$-\left[(\beta - p)s_1 + (V - \gamma^*)\right] \in \Sigma[x]$$



These are the same γ and β steps from the V-s iteration.

Simulation and Lyapunov function generation algorithm

Given positive definite convex $p \in \mathbb{R}[x]$, a vector of polynomials $\varphi(x)$ and constants β_{SIM}, N_{conv}, N_V, $\beta_{shrink} \in (0,1)$, and empty sets $\mathbf{C}$ and $\mathbf{D}$, set $\gamma = 1$, $N_{more} = N_{conv}$, $N_{div} = 0$.

1. Integrate $\dot{x} = f(x)$ from N_{more} initial conditions in $\Omega_{p,\beta_{SIM}}$.
2. If there is no diverging trajectory, add the trajectories to $\mathbf{C}$ and go to (3). Otherwise, add the divergent trajectories to $\mathbf{D}$ and the convergent trajectories to $\mathbf{C}$, let N_d denote the number of diverging trajectories found in the last run of (1) and set N_{div} to $N_{div} + N_d$. Set β_{SIM} to the minimum of $\beta_{shrink}\beta_{SIM}$ and the minimum value of p along the diverging trajectories. Set N_{more} to $N_{more} - N_d$, and go to (1).
3. At this point $\mathbf{C}$ has N_{conv} elements. For each $i = 1, \ldots, N_{conv}$, let $\overline{\tau}_i$ satisfy $\mathbf{c}_i(\tau) \in \Omega_{p,\beta_{SIM}}$ for all $\tau \geq \overline{\tau}_i$. Eliminate times in $\mathcal{T}_i$ that are less than $\overline{\tau}_i$.
4. Find a feasible point in $\mathcal{Y}$. If $\mathcal{Y}$ is empty, set $\beta_{SIM} = \beta_{shrink}\beta_{SIM}$, and go to (3). Otherwise, go to (5).
5. Generate N_V Lyapunov function candidates using H&R algorithm, and return β_{SIM} and Lyapunov function candidates.

Lower and upper bounds on certifiable β

Every solution of the optimization

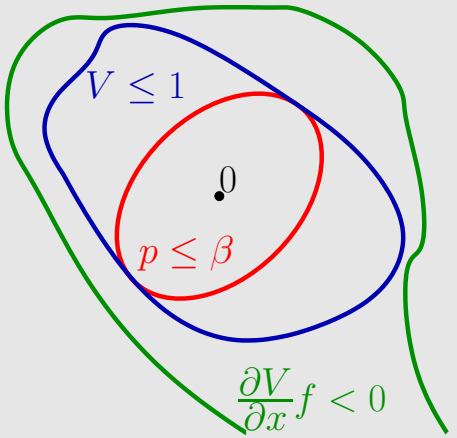
$$\max_{s_1,s_2\in\Sigma[x],\ V\in\mathbb{R}[x],\ \beta\in\mathbb{R}} \beta$$

subject to:

$$-\left((V-1)+s_1(\beta-p)\right)\in\Sigma\left[x\right]$$

$$-\left((\nabla V\cdot f+l_2)+s_2(1-V)\right)\in\Sigma\left[x\right]$$

$$V-l_1\in\Sigma\left[x\right],V(0)=0$$



provides a lower bound on the maximum certifiable value β^{cert} (through Lyapunov analysis by optimal choice of V) of β such that

$$\{x\ :\ p(x)\leq\beta^{cert}\}\subseteq\Omega_{V,1}\subset\{x\ :\ \nabla V f(x)<0\}.$$

By contrast, simulation trajectories provide upper bounds on β^{cert}. Upper bounds are used

- to assess the sub-optimality and
- to get clues for non-achievable.

Upper bound on certifiable β

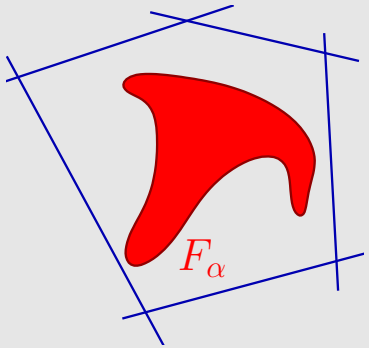
A divergent trajectory cannot enter the ROA. Consequently,

$$\beta^{cert} < \text{minimum value of } p \text{ on any divergent trajectory}$$

These minimum values are uppers bound regardless of the type of Lyapunov function and multipliers.

Upper bounds due to emptiness of $\mathcal{Y}$ (convex set in α-space)

- $\mathcal{V}_\varphi := \{\varphi(x)^T\alpha \ : \alpha \in \mathcal{R}^{N_b}\}$
- Let $\mathcal{S}$ be the set of S-procedure multipliers search over.
- Let $\mathcal{Y}_{\bar{\beta}}$ be generated using convergent simulation trajectories with initial conditions in $\Omega_{p,\bar{\beta}}$.



$$\mathcal{Y}_{\bar{\beta}} \text{ is empty} \quad \Rightarrow \quad \beta^{cert}(\mathcal{V}_\varphi, \mathcal{S}) \leq \beta^{cert}(\mathcal{V}_\varphi) \leq \bar{\beta}$$

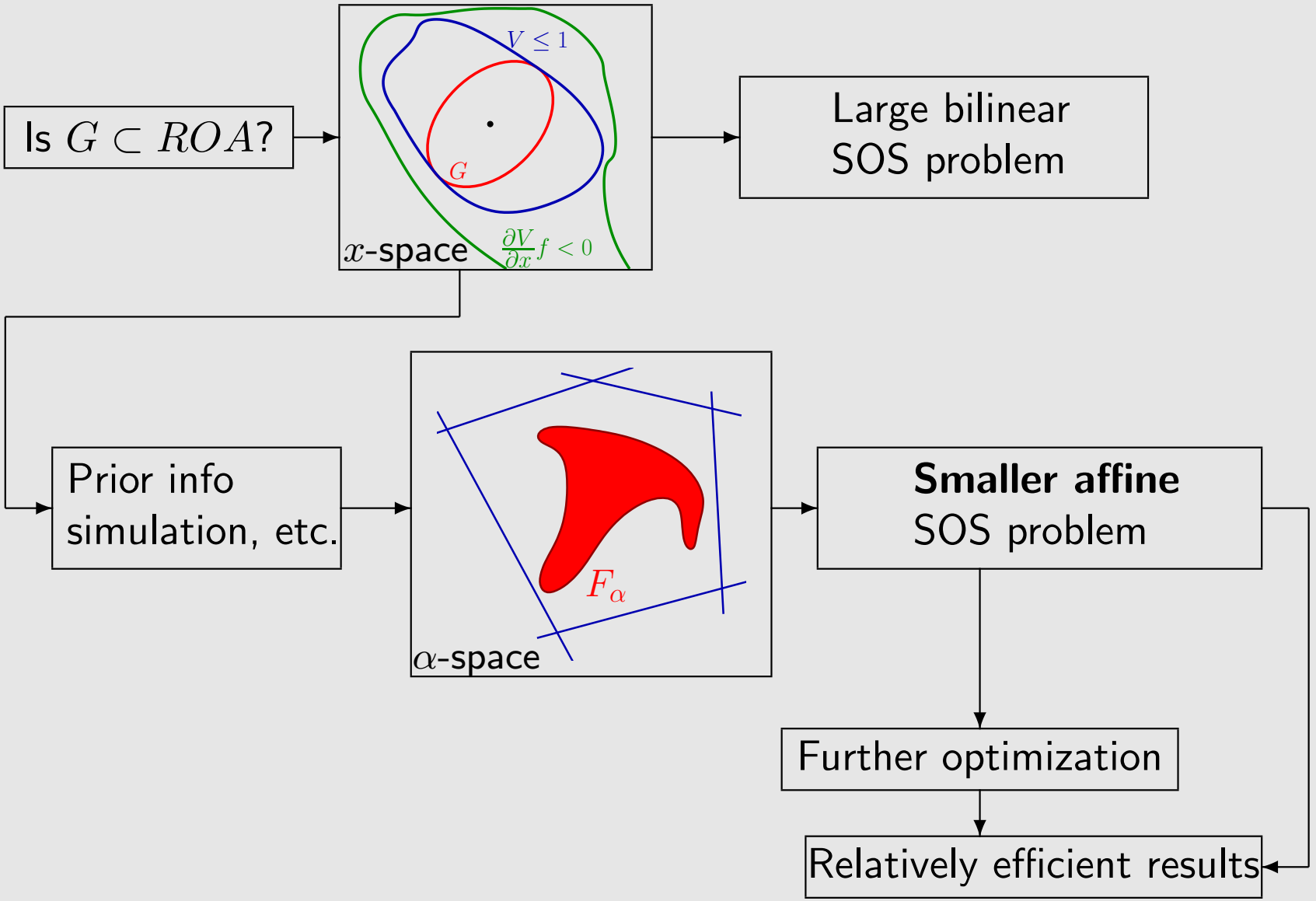
Examples: upper bounds on certifiable β

Demonstrate the upper bounds on VDP dynamics with $\deg(V) = 2$ and $\deg(V) = 6$

DemoBoundsVDP.html and DemoBoundsVDP.m

HTML codepad is rendered in the appendix.

Overview of the method



Controlled short period aircraft dynamics (1)

- States pitch rate (q), AoA (α), and pitch angle (θ).
- Cubic polynomial approximation of the dynamics (from Honeywell).

$$\dot{x}_p = \begin{bmatrix} c_1(x_p) \\ q_2(x_p) \\ x_1 \end{bmatrix} + \begin{bmatrix} \ell_b^T x_p \\ b_2 \\ 0 \end{bmatrix} u,$$

- $x_p = [x_1 \ x_2 \ x_3]^T = [q \ \alpha \ \theta]^T$.
- c_1 is a cubic polynomial, q_2 is a quadratic polynomial, ℓ_{12} and ℓ_b are vectors in $\mathcal{R}^3$, $b_2 \in \mathcal{R}$.
- The control input, elevator deflection, is determined by

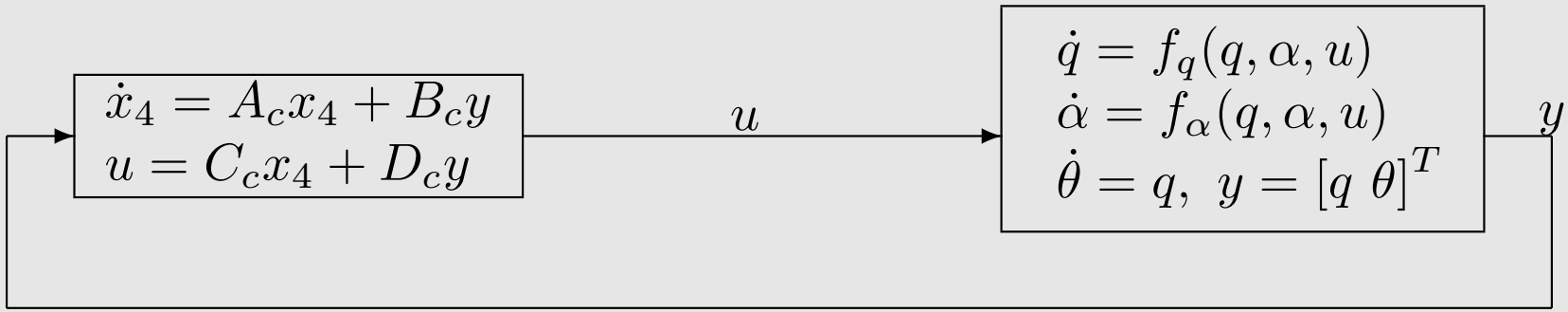
$$\begin{aligned} \dot{x}_4 &= -0.864 y_1 - 0.321 y_2 \\ u &= 2 x_4, \end{aligned}$$

  x_4 is the controller state and the plant output $y = [x_1 \ \ x_3]^T$.
- $x := \begin{bmatrix} x_p^T & x_4 \end{bmatrix}^T$.

Controlled short period aircraft dynamics (2)

$$\dot{x}_p = \begin{bmatrix} c_1(x_p) \\ q_2(x_p) \\ x_1 \end{bmatrix} + \begin{bmatrix} \ell_b^T x_p \\ b_2 \\ 0 \end{bmatrix} u,$$

$$\begin{aligned} y &= [x_1 \;\; x_3]^T \\ \dot{x}_4 &= -0.864 y_1 - 0.321 y_2 \\ u &= 2x_4, \end{aligned}$$



Outline

- Motivation
- Preliminaries
- ROA analysis using SOS optimization and solution strategies
- **Robust ROA analysis with parametric uncertainty**
- Local input-output analysis
- Robust ROA and performance analysis with unmodeled dynamics
- F-18

Systems with parametric uncertainty

System with parametric uncertainty governed by

$$\dot{x}(t) = f(x(t), \delta)$$

The parameter δ is

- constant
- unknown
- known to take values on the bounded set Δ

Assumption:

- For each $\delta \in \Delta$, the origin is an equilibrium point, i.e.,

$$f(0, \delta) = 0 \qquad \text{for all } \delta \in \Delta.$$

ROA analysis for systems with parametric uncertainty

System with **constant parametric** uncertainty governed by

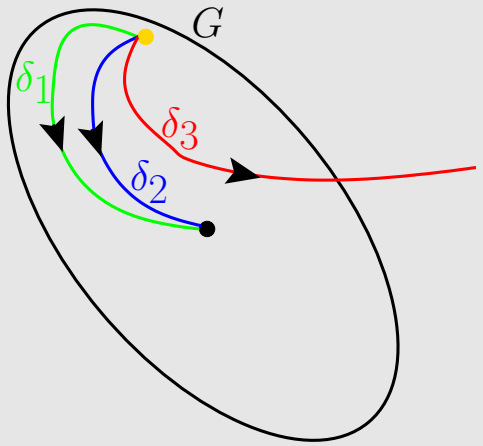
$$\dot{x}(t) = f(x(t), \delta)$$

Question: Given a set G,

- is G in the ROA for each $\delta \in \Delta$?
- is G a subset of the robust ROA, defines as

$$\bigcap_{\delta \in \Delta} \{\zeta \in \mathbb{R}^n \ : \ \lim_{t \to \infty} \varphi(\zeta, t; \delta) = 0\}?$$

[$\varphi(\zeta, t; \delta)$ is the solution at time t with initial condition ζ for δ.]



ROA analysis for $\dot{x} = f(x, \delta)$

Theorem: If there exists a continuously differentiable function V such that

- $V(0) = 0$, and $V(x) > 0$ for all $x \neq 0$
- $\Omega_{V,1} = \{x \ : \ V(x) \leq 1\}$ is bounded
- For each $\delta \in \Delta$, the set containment

$$\{x \ : \ V(x) \leq 1\} \backslash \{0\} \subset \{x \ : \ \nabla V(x) f(x, \delta) < 0\}$$

holds, then $\{x \in \mathbb{R}^n \ : \ V(x) \leq 1\}$ is an invariant subset of the robust ROA.

Proof: Apply Lyapunov theory to each system ...

A few issues:

- "For each $\delta \in \Delta$..." there are infinite number of set containment conditions.
- V does not depend on δ, though f does, will this be restrictive?

ROA analysis: $f(x, \delta)$ affine in δ

Affine uncertainty dependence & bounded, polytopic Δ (with vertices $\mathcal{E}$)

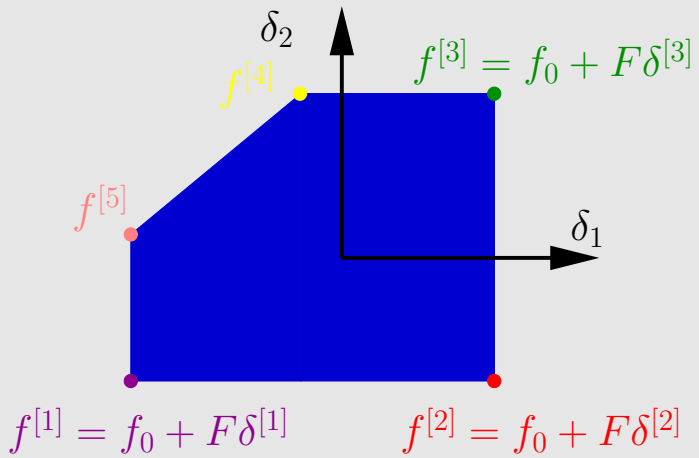
$$\dot{x}(t) = f_0(x(t)) + \sum_{i=1}^{m} f_i(x(t))\delta_i \quad = \quad f_0(x(t)) + F(x(t))\delta$$

Theorem: If Δ is a polytope, and for all $\delta \in \mathcal{E}$

$$\Omega_V \setminus \{0\} \subseteq \{x \in \mathbb{R}^n : \nabla V(x)(f_0(x) + F(x)\delta) < 0\},$$

then the set containment holds for all $\delta \in \Delta$.

Proof:
For each $\tilde{\delta} \in \Delta$, $\nabla V(x)F(x)\tilde{\delta}$ is a convex combination of $\{\nabla V(x)F(x)\delta \ : \ \delta \in \Delta\}$.

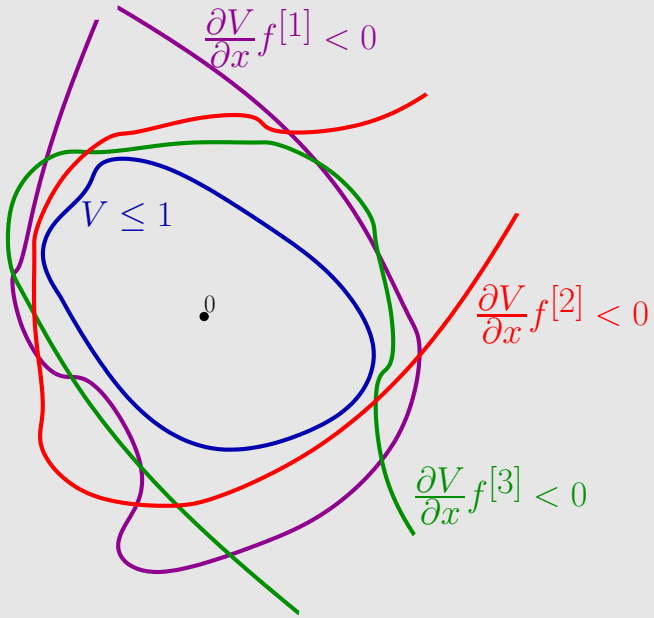


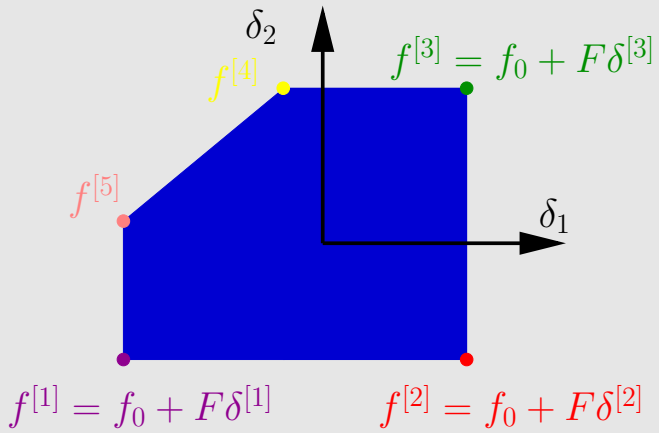
ROA analysis with parameter-independent V (2)

$$\dot{x}(t) = f_0(x(t)) + F(x(t))\delta$$

Impose at the vertices of Δ, then they hold everywhere on Δ.

$$\Omega_V \setminus \{0\} \subseteq \{x \in \mathbb{R}^n : \nabla V(x)(f_0(x) + F(x)\delta) < 0\}$$





For every $i = 1, \ldots, N_{vertex}$ (index to elements of $\mathcal{E}$),

$$-\left[(1 - V)s_2 + s_3 \nabla V \cdot (f_0 + F\delta^{[i]}) + l_2\right] \text{ is SOS in } x \text{ (only)}$$

SOS problem for robust ROA computation

$$\max_{0<\gamma,0<\beta,V\in\mathcal{V},s_1\in\mathcal{S}_1,s_{2\delta}\in\mathcal{S}_2,s_{3\delta}\in\mathcal{S}_3} \quad \beta \quad \text{subject to}$$

$$s_{2\delta} \in \Sigma[x], \text{ and } s_{3\delta} \in \Sigma[x]$$

$$-[(\gamma - V)s_{2\delta} + \nabla V(f_0 + F(x)\delta)s_{3\delta} + l_2] \in \Sigma[x] \quad \forall \delta \in \mathcal{E},$$

$$-[(\beta - p)s_1 + V - 1] \in \Sigma[x]$$

- Bilinear optimization problem
- SOS conditions:
  - only in x
  - δ does not appear, but...
  - there are a lot of SOS constraints ($\delta \in \mathcal{E}$)

Example

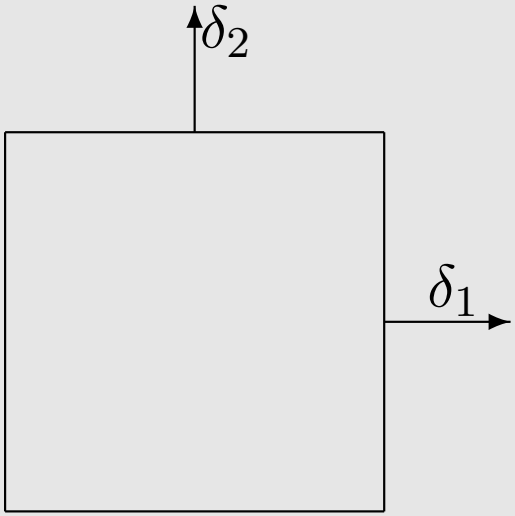
Consider the system with a single uncertain parameter δ

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_2 - (\delta + 2)(x_1 - x_1^3)$$

with $\delta \in [-1, 1]$.

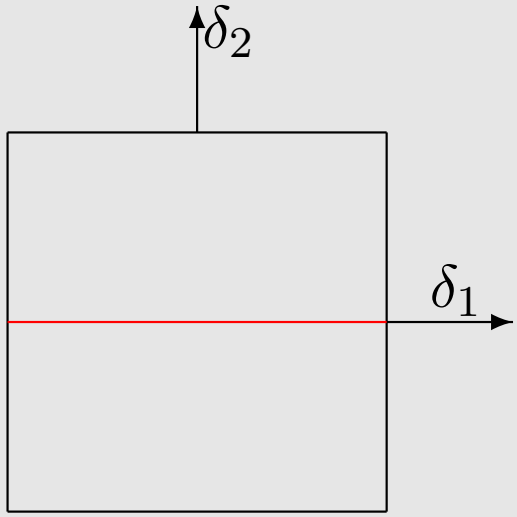
Codepad Demo: special1.m and special1.html

Dealing with conservatism: partition Δ



For all $\delta \in \Delta$:

$$\{x : V_0(x) \leq 1\} \backslash \{0\} \subset \left\{x : \frac{\partial V_0}{\partial x} f(x, \delta) < 0\right\}$$



For all $\delta \in$ upper half of Δ:

$$\{x : V_1(x) \leq 1\} \backslash \{0\} \subset \left\{x : \frac{\partial V_1}{\partial x} f(x, \delta) < 0\right\}$$

For all $\delta \in$ lower half of Δ:

$$\{x : V_2(x) \leq 1\} \backslash \{0\} \subset \left\{x : \frac{\partial V_2}{\partial x} f(x, \delta) < 0\right\}$$

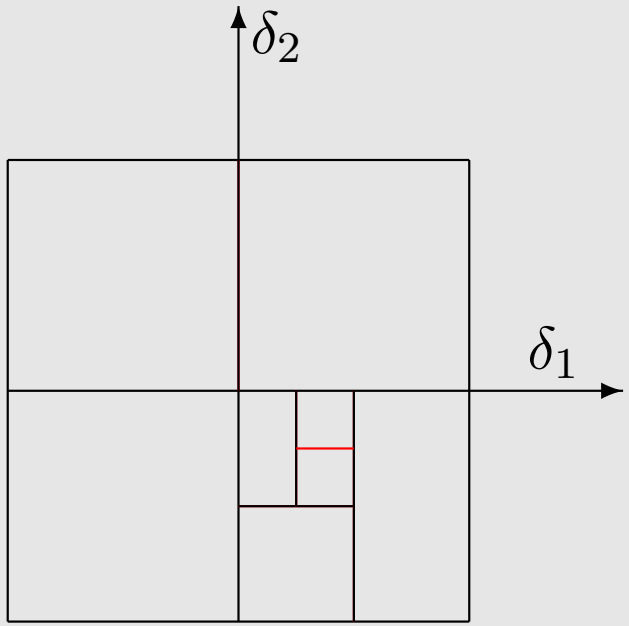
$V_1 := V_0$ and $V_2 := V_0$ are feasible for the right-hand side.
Improve the results by searching for different V_1 and V_2.

Dealing with conservatism: branch-and-bound in Δ

Systematically refine the partition of Δ:

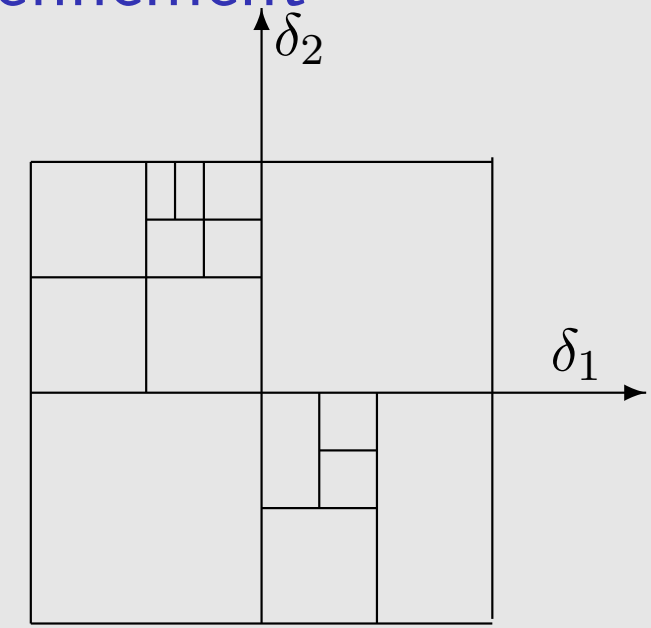
- Run an informal branch-and-bound (B&B) refinement procedure

Sub-division strategy: Divide the worst cell into 2 subcells.



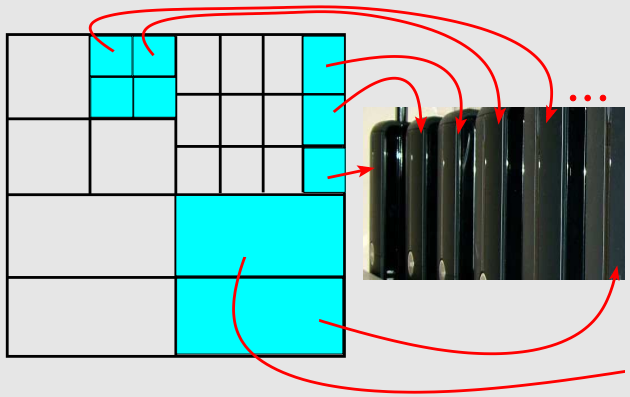
Properties of the branch-and-bound refinement

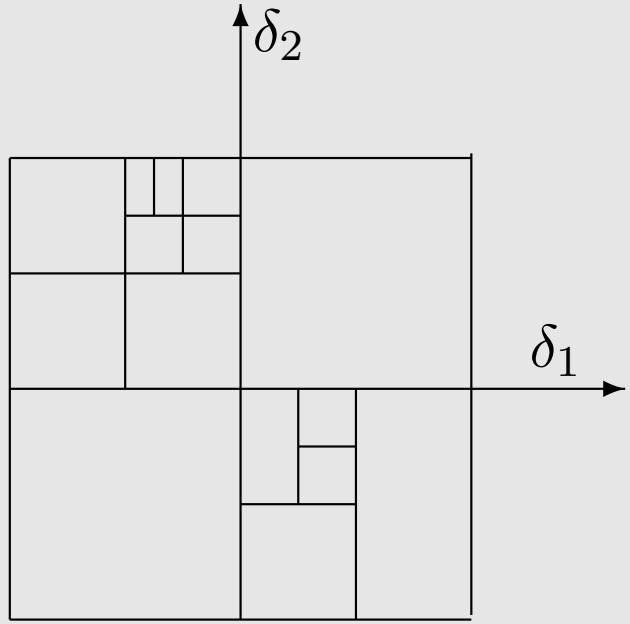
- Yields piecewise-polynomial, δ-dependent V.



Properties of the branch-and-bound refinement

- Yields piecewise-polynomial, δ-dependent V.
- Local problems are decoupled
  $\rightarrow$ parallel computing





- Organizes extra info regarding system behavior: returns a data structure with useful info about the system
  - Lyapunov functions, SOS certificates,
  - certified β,
  - worst case parameters,
  - initial conditions for divergent trajectories,
  - values of β not achievable, etc.

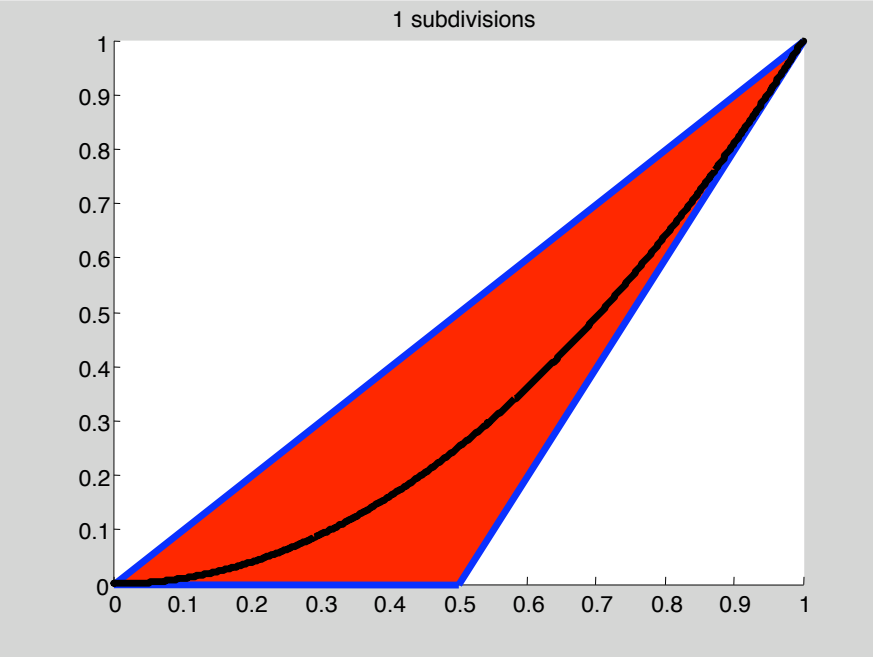
Non-affine dependence on δ

Let $g : \mathbb{R} \to \mathbb{R}$.

$$\begin{aligned}\dot{x}(t) &= f_0(x(t)) + \delta f_1(x(t)) + g(\delta) f_2(x(t)) \\ &= f_0(x(t)) + \delta f_1(x(t)) + \zeta f_2(x(t))\end{aligned}$$

Treat $(\delta, g(\delta))$ as 2 parameters, whose values lie on a 1-dimensional curve. Then

* Cover 1-d curve with 2-polytope
* Compute ROA
* Refine polytope into a union of smaller polytopes
* Solve robust ROA on each polytope
* Intersect ROAs $\to$ robust ROA



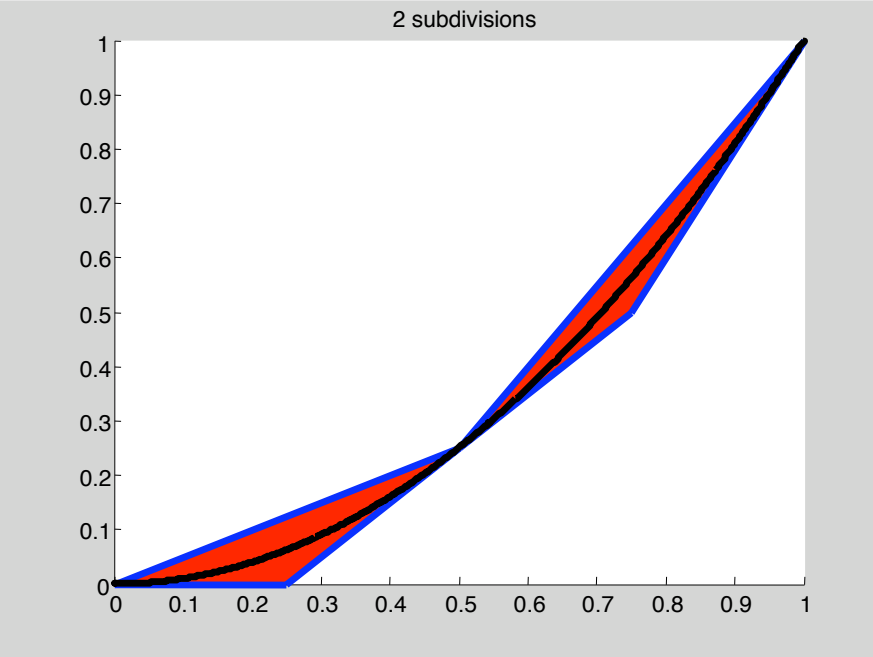
Non-affine dependence on δ

Let $g : \mathbb{R} \to \mathbb{R}$.

$$\begin{aligned} \dot{x}(t) &= f_0(x(t)) + \delta f_1(x(t)) + g(\delta) f_2(x(t)) \\ &= f_0(x(t)) + \delta f_1(x(t)) + \zeta f_2(x(t)) \end{aligned}$$

Treat $(\delta, g(\delta))$ as 2 parameters, whose values lie on a 1-dimensional curve. Then

* Cover 1-d curve with 2-polytope
* Compute ROA
* Refine polytope into a union of smaller polytopes
* Solve robust ROA on each polytope
* Intersect ROAs $\to$ robust ROA



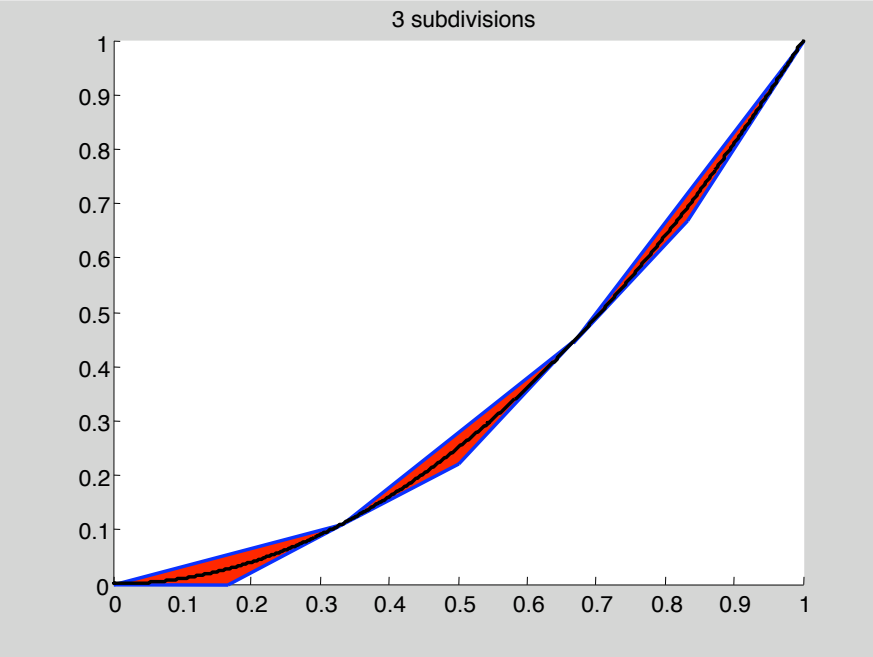
Non-affine dependence on δ

Let $g : \mathbb{R} \to \mathbb{R}$.

$$\begin{aligned} \dot{x}(t) &= f_0(x(t)) + \delta f_1(x(t)) + g(\delta) f_2(x(t)) \\ &= f_0(x(t)) + \delta f_1(x(t)) + \zeta f_2(x(t)) \end{aligned}$$

Treat $(\delta, g(\delta))$ as 2 parameters, whose values lie on a 1-dimensional curve. Then

* Cover 1-d curve with 2-polytope
* Compute ROA
* Refine polytope into a union of smaller polytopes
* Solve robust ROA on each polytope
* Intersect ROAs $\to$ robust ROA



Generalization of covering manifold

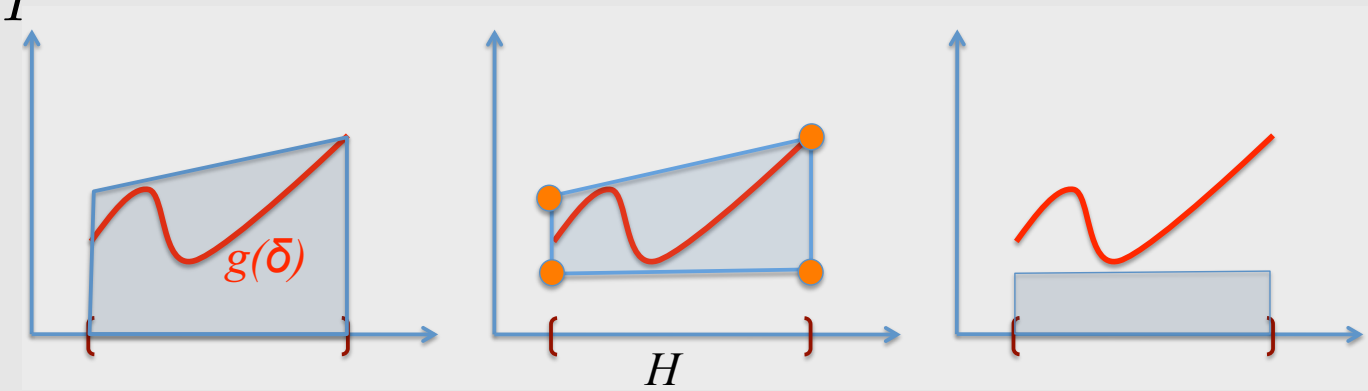
Given:

- polynomial $g(\delta)$ in many real variables, $\delta \in \mathbb{R}^q$
- domain $H \subseteq \mathbb{R}^q$, typically a polytope

Find a polytope that covers $\{(\delta, g(\delta)) : \delta \in H\} \subseteq \mathbb{R}^{q+1}$.

- Tradeoff between number of vertices, and
- excess "volume" in polytope

One approach: Find "tightest" affine upper and lower bounds to g over H

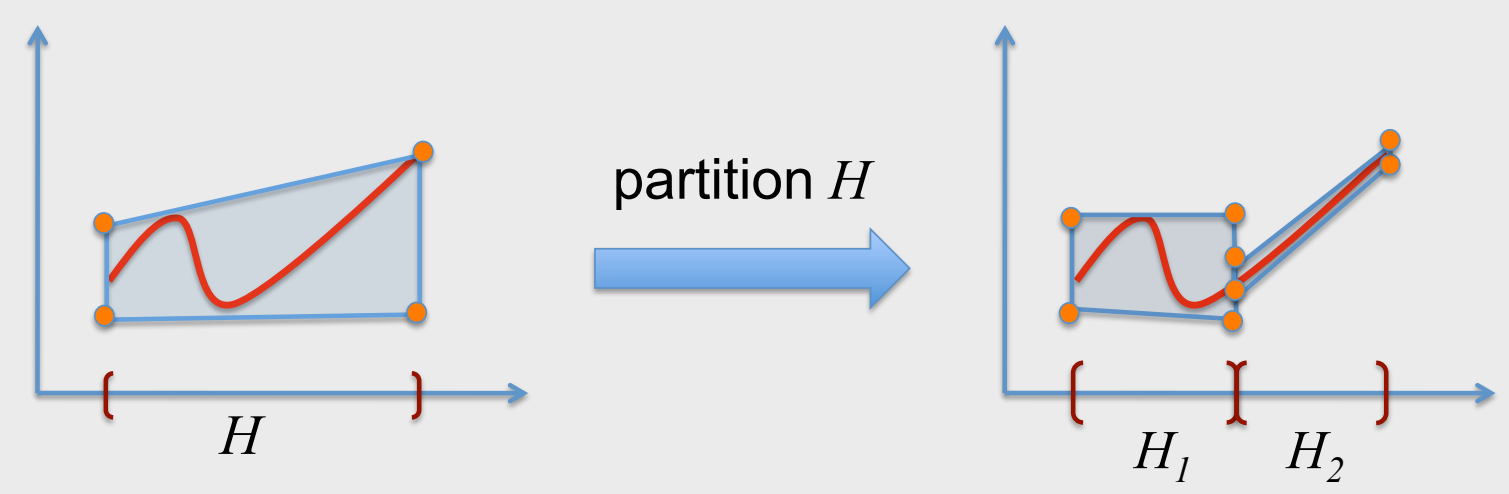


$$\min_{c_0,c} \int_H (c_0 + c^T\delta)d\delta \quad \text{subject to} \quad c_0 + c^T\delta \geq g(\delta) \;\; \forall \delta \in H$$

This optimization can be solved as a SOS program.

Non-affine dependence on δ (2)

Covering $\{(\delta, g(\delta) \ : \ \delta \in H\}$ introduces extra conservatism.



B&B refinement reduces the conservatism due to covering by reducing the extra covered space.

Multiple non-affine parametric uncertainty

For multivariable g,

$$\dot{x} = f_0(x) + \delta_1 f_1(x) + \cdots + \delta_q f_q(x) + g_1(\delta) f_{q+1}(x) + \cdots + g_m(\delta) f_{q+m}(x)$$

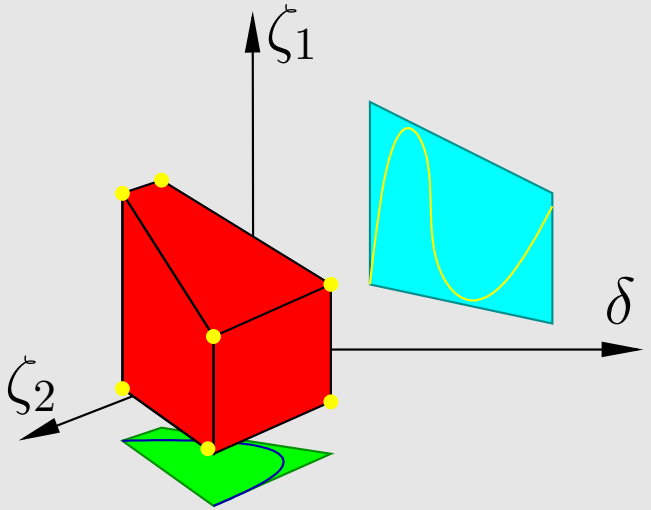
On H, bound each g_i with affine functions c_i and d_i

$$c_i(\delta) \leq g_i(\delta) \leq d_i(\delta) \quad \forall \delta \in H$$

Then (Amato, Garofalo, Gliemo) a polytope covering $\{(\delta, g(\delta)) : \delta \in H\}$ is

$$\left\{(\delta, v) \in \mathbb{R}^{q \times m} : \delta \in H, C(\delta) \leq v \leq D(\delta)\right\}$$

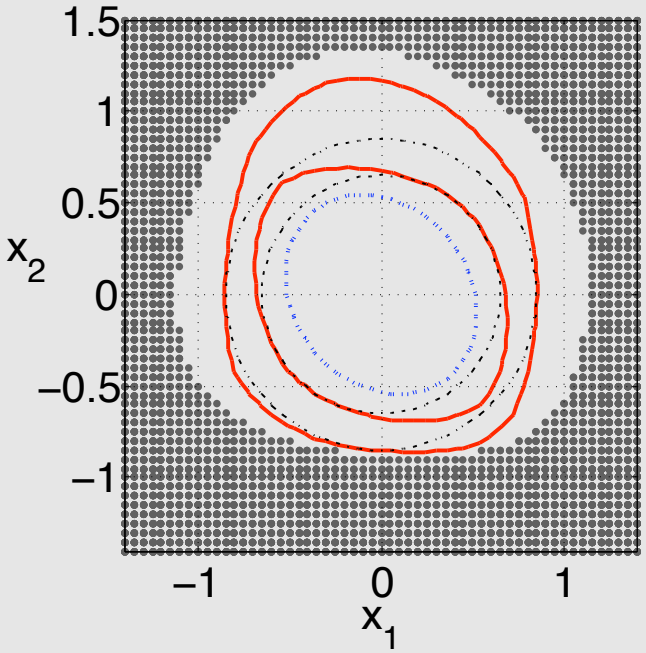
with 2^{q+m} easily computed vertices.



Example: Interesting 2-state uncertain dynamics [Chesi, 2004]

$$\dot{x} = \begin{bmatrix} -x_1 \\ 3x_1 - 2x_2 \end{bmatrix} - \begin{bmatrix} 6x_2 - x_2^2 - x_1^3 \\ 10x_1 - 6x_2 - x_1 x_2 \end{bmatrix} \delta + \begin{bmatrix} 4x_2 - x_2^2 \\ 12x_1 - 4x_2 \end{bmatrix} \delta^2,$$

- $\delta \in [0, 1]$.
- No common quadratic V for uncertain linearized dyn.
- $p(x) = x^T x$.
- 50 branch-and-bound refinements



Blue dotted curve: Result from Chesi, 2004.

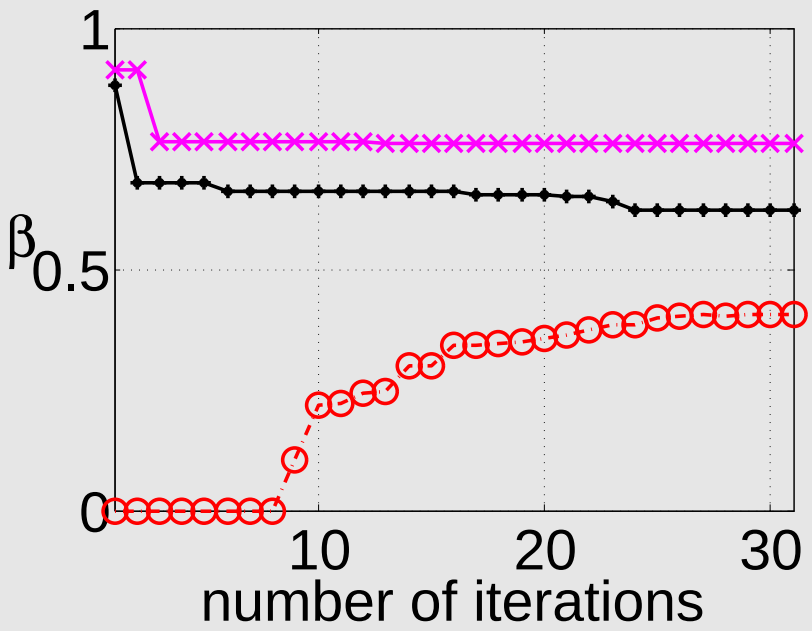
Red curves: Intersection of $\Omega_{V,1}$ for V's obtained through the B&B refinement (inner for $\deg(V) = 2$ and outer for $\deg(V) = 4$)

Black dotted curves: Certified $\Omega_{p,\beta}$ for $\deg(V) = 2$ (inner) and for $\deg(V) = 4$ (outer)

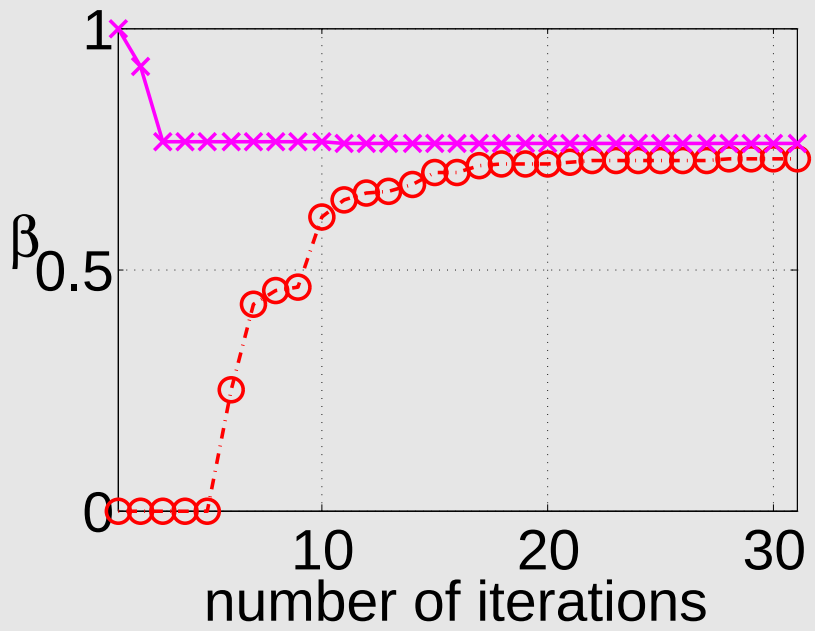
Example: Interesting 2-state uncertain dynamics

B&B iterations: Divide the cell with the smallest β into 2.

$\deg(V) = 2$



$\deg(V) = 4$



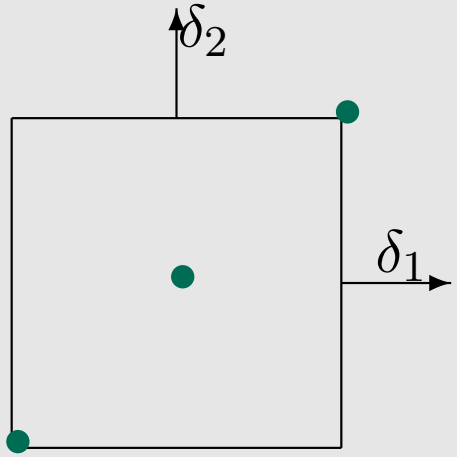
- Upper bounds from divergent trajectories
  - Upper bound does not depend on the complexity/degree of V
- Upper bounds from infeasibility of the affine relaxation
  - This bound shows how the basis choice for V impacts what is certifiable.
- Certified values (using ideas from last previous 100+ slides)

Dealing with large number of constraints

The SOS problem for the robust ROA includes the constraint:

$$-[(\gamma - V)s_{2\delta} + \nabla V(f_0 + F(x)\delta)s_{3\delta} + l_2] \in \Sigma[x] \quad \forall \delta \in \mathcal{E}$$

The number of vertices grows fast with the dimension of the uncertainty space.



Suboptimal procedure:

- Sample Δ with fewer points (fewer than in $\mathcal{E}$)
- Optimize V for this restricted sampling
- Certify a value of β, using this V, at all vertices of Δ

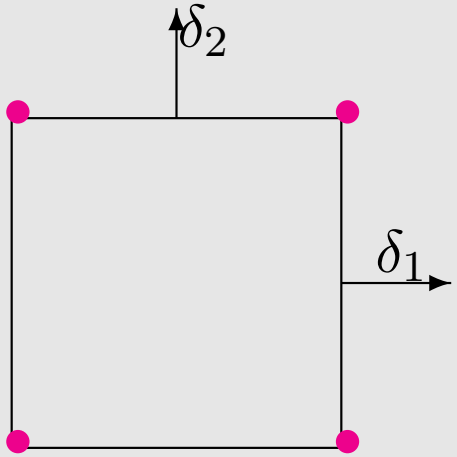
The last step involves solving decoupled smaller problems.

Dealing with large number of constraints

The SOS problem for the robust ROA includes the constraint:

$$-[(\gamma - V)s_{2\delta} + \nabla V(f_0 + F(x)\delta)s_{3\delta} + l_2] \in \Sigma[x] \quad \forall \delta \in \mathcal{E}$$

The number of vertices grows fast with the dimension of the uncertainty space.



Suboptimal procedure:

- Sample Δ with fewer points (fewer than in $\mathcal{E}$)
- Optimize V for this restricted sampling
- Certify a value of β, using this V, at all vertices of Δ

The last step involves solving decoupled smaller problems.

Dealing with large number of constraints: 2-step procedure

- Call the Lyapunov function computed for a sample of Δ as $\tilde{V}$.
- For each $\delta \in \mathcal{E}$, compute

$$\gamma_\delta := \max_{0<\gamma, s_{2\delta}\in\mathcal{S}_2, s_{3\delta}\in\mathcal{S}_3} \gamma \quad \text{subject to}$$
$$s_{2\delta} \in \Sigma[x], \text{ and } s_{3\delta} \in \Sigma[x]$$
$$-[(\gamma - \tilde{V})s_{2\delta} + \nabla\tilde{V}(f_0 + F\delta)s_{3\delta} + l_2] \in \Sigma[x],$$

  and define

$$\gamma^{subopt} := \min\{\gamma_\delta \ : \ \delta \in \mathcal{E}\}.$$

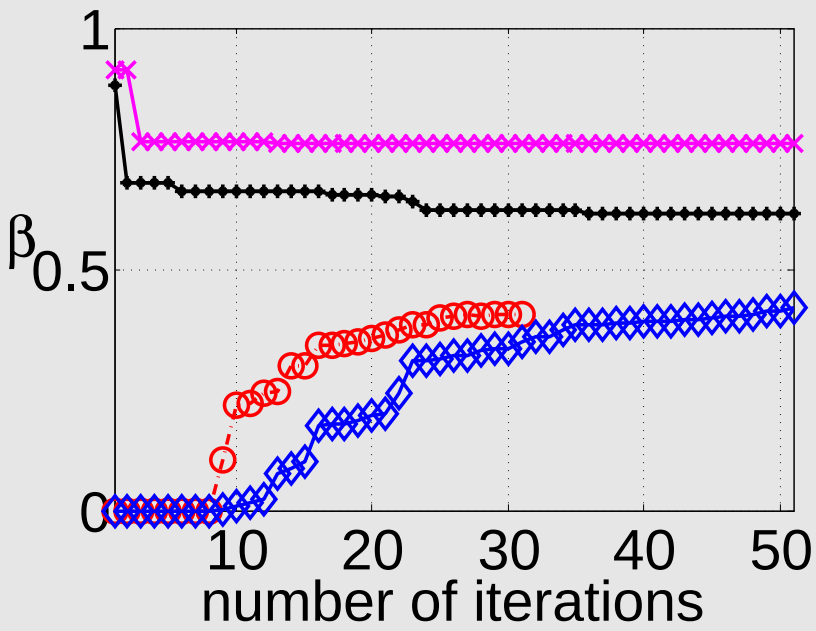
  $\Omega_{\tilde{V},\gamma^{subopt}}$ is an invariant subset of the robust ROA.
- Determine the largest sublevel set of p contained in $\Omega_{\tilde{V},\gamma^{subopt}}$

$$\max_{s_1\in\mathcal{S}_1,\beta} \beta \quad \text{subject to}$$
$$s_1 \in \Sigma[x]$$
$$-[(\beta - p)s_1 + \tilde{V} - \gamma^{subopt}] \in \Sigma[x].$$

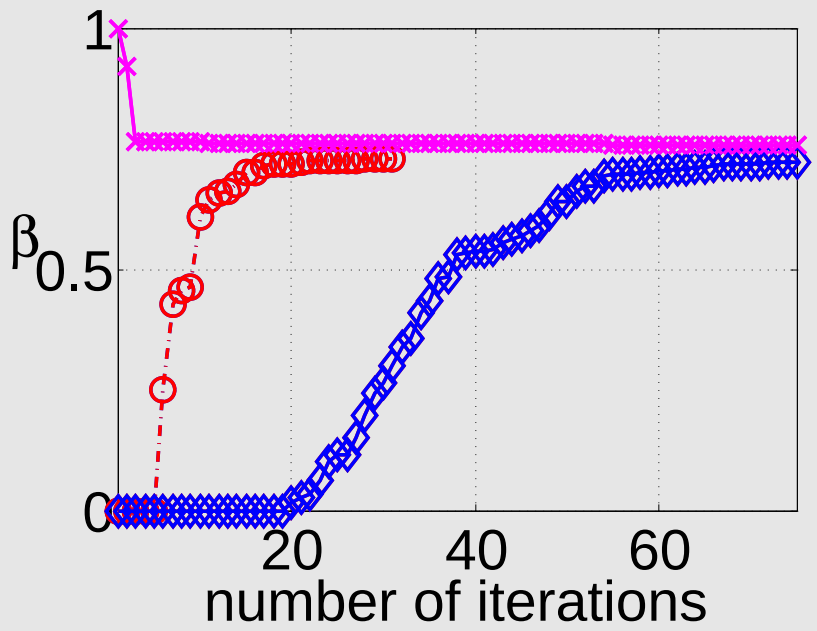
Revisit *Chesi, 2004* with suboptimal Δ sampling

B&B iterations: Divide the cell with the smallest β into 2.

$\deg(V) = 2$



$\deg(V) = 4$



- Upper bounds from divergent trajectories
- Upper bounds from infeasibility of the affine relaxation
- Lower bounds directly computing the robust ROA
- Lower bounds computing the robust ROA in two steps (sample Δ at cell center $\longrightarrow$ optimize V $\longrightarrow$ verify at the vertices)

Controlled aircraft [Short period pitch axis model]

Uncertain closed loop dynamics with

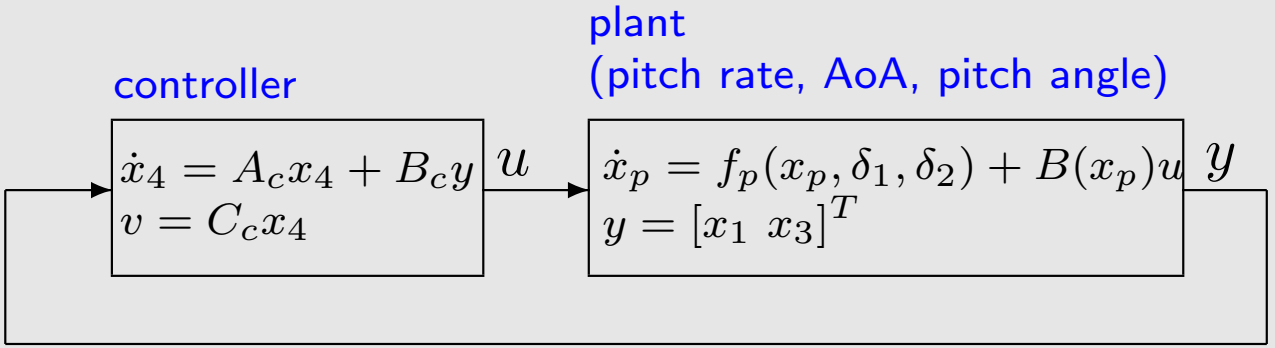
- $x = (x_p,\ x_4), \quad p(x) = x^T x$
- Cubic poly approx from Honeywell

$$\dot{x} = f_0(x) + f_1(x)\delta_1 + f_2(x)\delta_2 + f_3(x)\delta_1^2$$

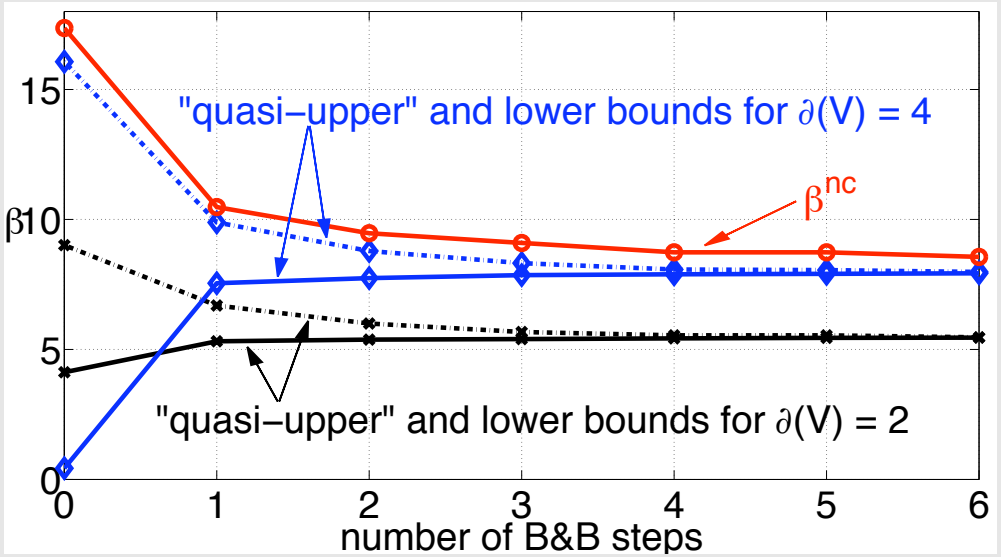
- $\delta_1 \in [0.99, 2.05]$ (uncertainty in the center of gravity)
- $\delta_2 \in [-0.1, 0.1]$ (uncertainty in mass)

Implemented on a 9-processor cluster

- Problems for 9 cells are solved at a time
- Trivial speed up as expected.



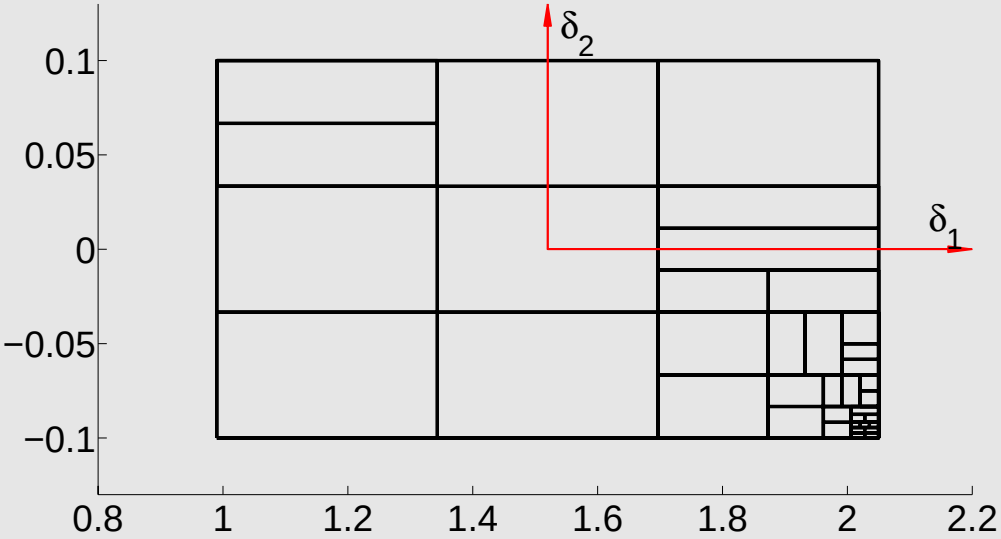
Results - controlled aircraft dynamics



Strategy:

- Optimize at the center
- Verify at the vertices

Quasi upper bound: β certified (by the SOS problem) for the "center system" in the first step.

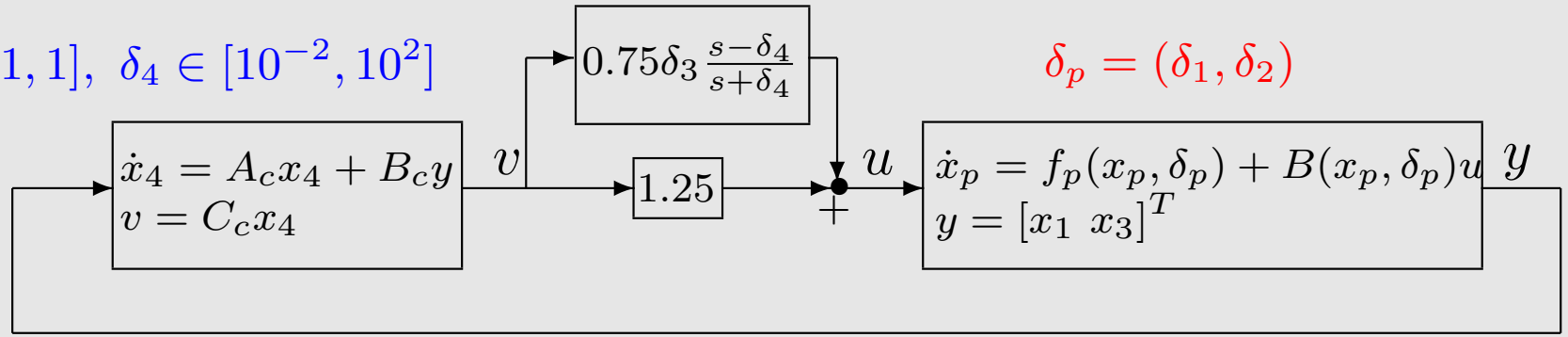


$$\dot{x} = f_0(x) + f_1(x)\delta_1 + f_2(x)\delta_2 + f_3(x)\delta_1^2$$

Controlled aircraft + 1st order unmodeled dynamics

$\delta_3 \in [-1, 1],\ \delta_4 \in [10^{-2}, 10^2]$

$\delta_p = (\delta_1, \delta_2)$



$$\dot{x} = f_0(x) + \sum_{i=1}^{4} f_i(x)\delta_i + f_5(x)\delta_1^2 + f_6(x)\delta_1\delta_3 + f_7(x)\delta_2\delta_3$$

- First order LTI unmodeled dyn (state x_5)
- $p(x) = x^T x$, $x = \begin{bmatrix} x_p^T & x_4 & x_5 \end{bmatrix}^T$.

Certified

dyn uncer \ param uncer	with	without
with	2.8	4.9
without	5.4	8.0

How about other uncertainty descriptions (e.g. unmodeled dynamics)?

Coming up later

Alternative uncertainty description

An alternate uncertainty description includes

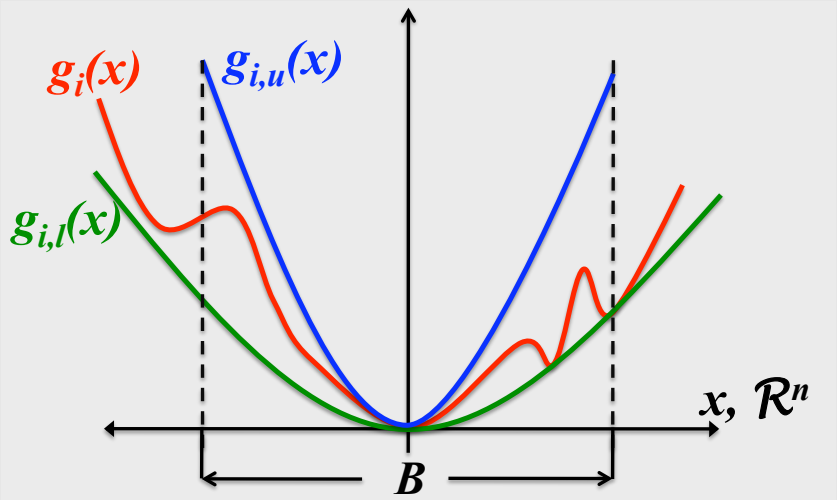
- nonpolynomial vector fields
- limited domain of validity

$$\dot{x}(t) = f_0(x(t)) + g(x(t))$$

$g : \mathbb{R}^n \to \mathbb{R}^n$ is unknown and satisfies polynomial, local bounds

$$g_l(x) \preceq g(x) \preceq g_u(x) \qquad \forall x \in B := \{x \ : \ b(x) \succeq 0\}$$

where $g_u,\ g_l \in \mathbb{R}[x]$, $g_u(0) = g_l(0) = 0$, and B contains the origin.



Recall $v \preceq w$ implied $v_i \leq w_i$ for all $i = 1, \ldots, n$.

Robust ROA with the alternative uncertainty description

A family $\mathcal{D}$ of vector fields:

$$\dot{x}(t) = f_0(x(t)) + g(x(t))$$

$g : \mathbb{R}^n \to \mathbb{R}^n$ is (only) known to satisfy

$$g_l(x) \preceq g(x) \preceq g_u(x) \quad \forall x \in B := \{x \ : \ b(x) \succeq 0\}$$

Question: Compute an estimate of the ROA for the uncertain vector field, i.e., a set that is

- invariant for each vector field of the form $f_0 + g \ \forall g \in \mathcal{D}$
- such that every trajectory with initial condition in the set converges to the origin.

Computed invariant subset of the robust ROA has to be a subset of B (region of validity).

Sufficient conditions in alternative uncertainty description

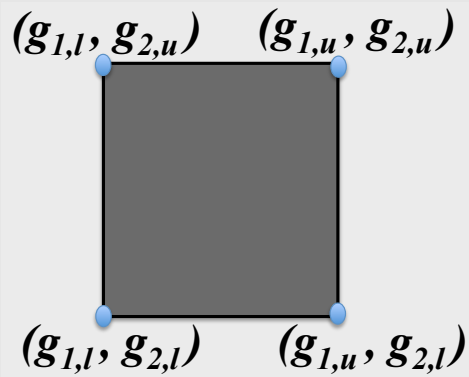
Impose the set containment constraint for each $g \in \mathcal{D}$

$$\{x \ : \ V(x) \leq 1\} \backslash \{0\} \subset \left\{ x \ : \ \frac{\partial V}{\partial x}(f_0(x) + g(x)) < 0 \right\},$$

then $\{x \in \mathbb{R}^n \ : \ V(x) \leq 1\}$ is an invariant subset of robust ROA.
$\mathcal{D}$ contains infinitely many constraints. But,

- dependence on g is affine
- $\mathcal{D}$ is a "polytope"

Vertices of "polytope of functions"
$\mathcal{E} := \{g \ : \ g_i = g_{i,\alpha} \quad \alpha = u, l\}$



Impose the set containment constraints for each $g \in \mathcal{E}$, then they will hold for each $g \in \mathcal{D}$

Computing robust ROA (alternative uncer. model)

$$\max_{V\in\mathcal{V},\beta>0} \quad \beta \ \text{ subject to}$$

$$V(0)=0 \text{ and } V(x)>0 \text{ for all } x\neq 0,\ \ \Omega_{V,1} \text{ is bounded},$$

$$\Omega_{p,\beta}\subseteq\Omega_{V,1}\subseteq B,$$

$$\Omega_{V,1}\setminus\{0\}\subseteq\bigcap_{g\in\mathcal{E}}\{x\in\mathbb{R}^n : \nabla V(f_0(x)+g(x))<0\}.$$

Let B be defined by several polynomial inequalities $B=\{x \ : \ b(x)\succeq 0\}$. Then, a SOS relaxation for the above problem is

$$\max_{V\in\mathcal{V},\beta>0,s_1\in\mathcal{S}_1,\ s_{4k}\in\mathcal{S}_{4k},\ s_{2g}\in\mathcal{S}_{2g},s_{3g}\in\mathcal{S}_{3g}} \quad \beta \ \text{ subject to}$$

$$V-l_1 \text{ is SOS},\ V(0)=0,\ \ s_1,\ \ s_{41},\ldots,s_{4,m} \text{ are SOS}$$

$$s_{2g},s_{3g} \text{ are SOS for } g\in\mathcal{E}$$

$$-\left[(\beta-p)s_1+(V-1)\right] \text{ is SOS}$$

$$b_k-(1-V)s_{4k} \text{ is SOS for } k=1,\ldots,m,$$

$$\left[(1-V)s_{2\xi}+\nabla V(f_0+g)s_{3\xi}+l_2\right] \text{ is SOS for } g\in\mathcal{E}$$

Example

Consider the system governed by

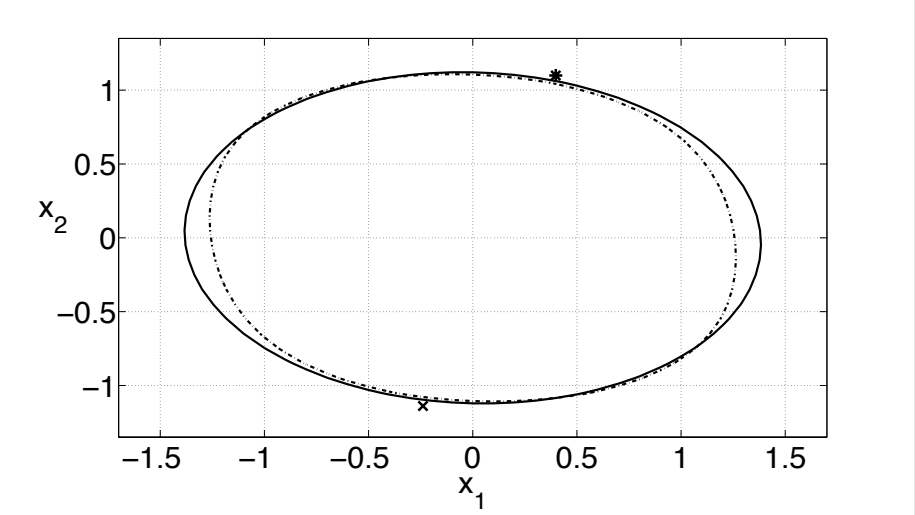
$$\dot{x} = \begin{bmatrix} -2x_1 + x_2 + x_1^3 + 1.58x_2^3 \\ -x_1 - x_2 + 0.13x_2^3 + 0.66x_1^2 x_2 \end{bmatrix} + g(x),$$

where g satisfies the bounds

$$-0.76x_2^2 \leq g_1(x) \leq 0.76x_2^2$$
$$-0.19(x_1^2 + x_2^2) \leq g_2(x) \leq 0.19(x_1^2 + x_2^2)$$

for all $x \in \{x \in \mathbb{R}^2 \ : \ x^T x \leq 2.1\}$.

- $p(x) = x^T x$
- $\deg(V) = 4$ (dashed curve)
- $\deg(V) = 2$ (solid curve)
- initial conditions for trajectories that leave the region of validity for $g(x) = \pm(0.76x_2^2, 0.19(x_1^2 + x_2^2))$ (dots)



Parameter dependent vs. independent Lyapunov functions

- Parameter-dependent Lyapunov functions: $V(x, \delta)$
  - δ explicitly appears in V
  - relatively larger SDP constraints
  - dynamic D-scales
    $\rightarrow$ parameter-dependent V rational of quadratics in δ.
- Parameter-independent Lyapunov functions: $V(x)$
  - use the same V over the whole uncertainty set
  - in certain cases, it may be possible to get constraints that do not explicitly depend on δ

computational complexity increases

more conservative estimates of the ROA

So,

- Use parameter-independent V's and handle conservatism separately

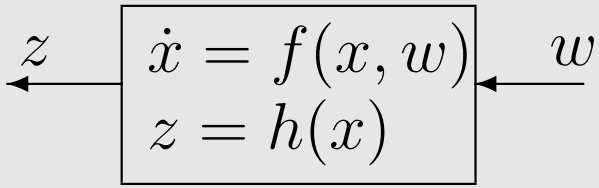
Outline

- Motivation
- Preliminaries
- ROA analysis using SOS optimization and solution strategies
- Robust ROA analysis with parametric uncertainty
- **Local input-output analysis**
- Robust ROA and performance analysis with unmodeled dynamics
- F-18

What if there is external input/disturbance?

So far, only internal properties, no external inputs!

What if there are external inputs/disturbances?



$$f(0,0) = 0,\ h(0) = 0$$

If w has **bounded energy/amplitude** and system starts from **rest**

- (reachability) how far can x be driven from the origin?
- (input-output gain) what are bounds on the output energy/amplitude in terms of input energy?

Notation

- For $u : [0, \infty) \to \mathbb{R}^n$, define the (truncated) $\mathcal{L}_2$ norm as

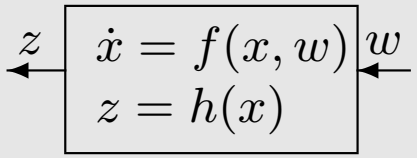
$$\|u\|_{2,T} := \sqrt{\int_0^T u(t)^T u(t) dt}.$$

- For simplicity, denote $\|u\|_{2,\infty}$ by $\|u\|_2$.
- $\mathcal{L}_2$ is the set of all functions $u : [0, \infty) \to \mathbb{R}^n$ such that $\|u\|_2 < 0$.
- For $u : [0, \infty) \to \mathbb{R}^n$ and for $T \geq 0$, define $u_T : [0, \infty) \to \mathbb{R}^n$ as

$$u_T(t) : \begin{cases} u(t), & 0 \leq t \leq T \\ 0, & T < t \end{cases}$$

- $\mathcal{L}_{2,e}$ is the set of measurable functions $u : [0, \infty) \to \mathbb{R}^n$ such that $u_T \in \mathcal{L}_2$ for all $T \geq 0$.

Upper bounds on "local" $\mathcal{L}_2 \to \mathcal{L}_2$ input-output gains



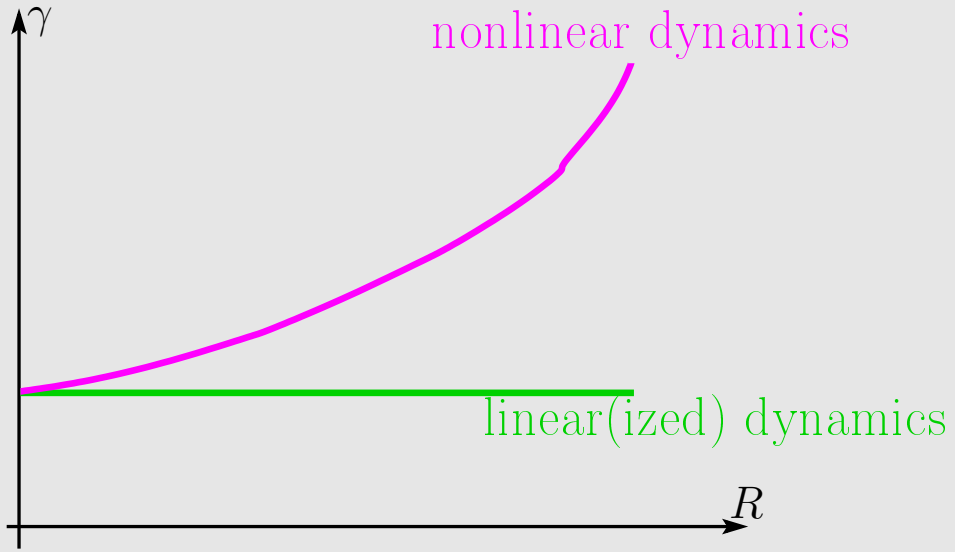
Goal: Establish relations between inputs and outputs:

$$x(0) = 0 \;\&\; \|w\|_2 \le R \qquad \Rightarrow \qquad \|z\|_2 \le \gamma \|w\|_2.$$

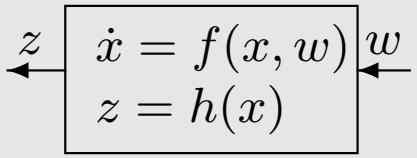
- Given R, minimize γ
- Given γ, maximize R

The $\mathcal{H}_\infty$ norm is a lower bound on the set of γ's which satisfy inequalty.

Why "local" analysis?



Upper bounds on "local" $\mathcal{L}_2 \to \mathcal{L}_2$ input-output gains



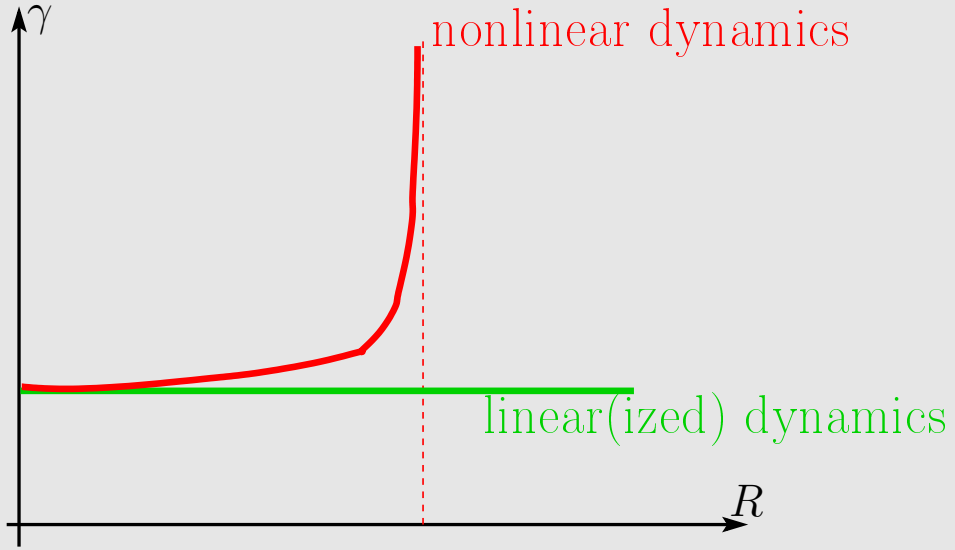
Goal: Establish relations between inputs and outputs:

$$x(0) = 0 \;\&\; \|w\|_2 \le R \qquad \Rightarrow \qquad \|z\|_2 \le \gamma \|w\|_2.$$

- Given R, minimize γ
- Given γ, maximize R

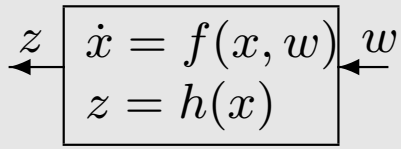
The $\mathcal{H}_\infty$ norm is a lower bound on the set of γ's which satisfy inequalty.

Why "local" analysis?



Local gain analysis

Theorem: If there exists a continuously differentiable function V such that $V(0) = 0$, $V(x) > 0$ for all $x \neq 0$,



- $\Omega_{V,R^2} := \{x \ : \ V(x) \leq R^2\}$ is bounded
- $\nabla V f(x, w) \leq w^T w - \frac{1}{\gamma^2} h(x)^T h(x)$ for all $x \in \Omega_{V,R^2}$ and $w \in \mathbb{R}^{n_w}$,

then

$$x(0) = 0, \ w \in \mathcal{L}_{2,e}, \ \& \ \|w\|_{2,T} \leq R \qquad \Rightarrow \qquad \|z\|_{2,T} \leq \gamma \|w\|_{2,T}.$$

- Note that algebraic condition on $(x, w) \in \mathbb{R}^n \times \mathbb{R}^{n_w}$ implies a relation between the signals $w \in \mathcal{L}_{2,e}$ and $z = h(x) \in \mathcal{L}_{2,e}$.
- Supply rate, $w^T w - \frac{1}{\gamma^2} h(x)^T h(x)$; Storage function, V.

Bilinear SOS problem formulation for gain analysis

For given $\gamma > 0$ and positive definite function l, define $R_{\mathcal{L}_2}$ by

$$R^2_{\mathcal{L}_2} := \max_{V \in \mathcal{V}_{poly}, R^2 > 0, s_1 \in \mathcal{S}_1} \quad R^2 \quad \text{subject to}$$

$$V(0) = 0, \quad s_1 \in \Sigma[(x,w)],$$

$$V - l \in \Sigma[x],$$

$$-\left[(R^2 - V)s_1 + \nabla V f(x,w) - w^T w + \gamma^{-2} z^T z\right] \in \Sigma[(x,w)].$$

Then,

$$x(0) = 0 \;\&\; \|w\|_2 \leq R_{\mathcal{L}_2} \qquad \Rightarrow \qquad \|z\|_2 \leq \gamma \|w\|_2.$$

- $\mathcal{V}_{poly}$ and $\mathcal{S}$'s are prescribed finite-dimensional subsets of $\mathbb{R}[x]$.
- $R^2_{\mathcal{L}_2}$ is a function of $\mathcal{V}_{poly}$, $\mathcal{S}$, and γ. This dependence will be dropped in notation.

- Similar problem for minimizing γ for given R.

Strategy to solve the bilinear SOS problem in gain analysis

Coordinate-wise affine search: Given a "feasible" V, alternate between

- maximize R^2 by choice of s_1 (requires bisection on R!)

$$R^2_{\mathcal{L}_2} := \max_{R^2>0, s_1 \in \mathcal{S}_1} \quad R^2 \quad \text{subject to}$$
$$s_1 \in \Sigma[(x,w)],$$
$$-\left[(R^2 - V)s_1 + \nabla V f(x,w) - w^T w + \gamma^{-2} z^T z\right] \in \Sigma[(x,w)].$$

- fix the multiplier and maximize R^2 by choice of V.

$$R^2_{\mathcal{L}_2} := \max_{V \in \mathcal{V}_{poly}, R^2>0} \quad R^2 \quad \text{subject to}$$
$$V(0) = 0, \quad V - l \in \Sigma[x],$$
$$-\left[(R^2 - V)s_1 + \nabla V f(x,w) - w^T w + \gamma^{-2} z^T z\right] \in \Sigma[(x,w)].$$

Strategy to solve the bilinear SOS problem in gain analysis

Finding initial "feasible" V:

- Incorporate simulation data (requires to sample the input space!)
- Let $\gamma >$ gain of the linearized dynamics

$$\begin{aligned} \dot{\delta}_x &= A\delta_x + \delta_w \\ \delta_z &= C\delta_x \end{aligned}$$

and let $P \succ 0$ satisfy

$$\begin{bmatrix} A^T P + PA + \frac{1}{\gamma^2} C^T C & PB \\ B^T P & -I \end{bmatrix} \prec 0.$$

Then, there exists a small enough R such that

$$x(0) = 0 \;\&\; \|w\|_2 \leq R \qquad \Rightarrow \qquad \|z\|_2 \leq \gamma \|w\|_2.$$

Coordinate-wise affine search with no bisection

For given l, f, γ, and h (such that $z = h(x)$), if V, $R > 0$, and s_1) are feasible for

$$V(0) = 0, \quad s_1 \in \Sigma[(x, w)],$$
$$V - l \in \Sigma[x],$$
$$-\left[(R^2 - V)s_1 + \nabla V f(x, w) - w^T w + \gamma^{-2} z^T z\right] \in \Sigma[(x, w)],$$

then

$$K := \frac{V}{R^2} \qquad \tilde{s}_1 = R^2 s_1$$

are feasible for

$$K(0) = 0, \quad \tilde{s}_1 \in \Sigma[(x, w)],$$
$$K - \tfrac{1}{R^2} l \in \Sigma[x],$$
$$-\left[(1 - K)\tilde{s}_1 + \nabla K f(x, w) - \tfrac{1}{R^2}(w^T w - \gamma^{-2} z^T z)\right] \in \Sigma[(x, w)].$$

- For given $\tilde{s}_1$, the last constraint is affine in K and $1/R^2$.

Lower bound for $\mathcal{L}_2 \to \mathcal{L}_2$ gain

Let γ and R be obtained through the SOS based gain analysis. Then, for $T \geq 0$

$$\max_{w} \{\|z\|_{2,T} \ : \ \ x(0) = 0 \ \& \ \|w\|_{2,T} \leq R\} \leq \gamma R.$$

The first-order conditions for stationarity of the above finite horizon maximum are the existence of signals (x, λ) and w which satisfy

$$\dot{x} = f(x, w)$$
$$\|w\|_{2,T}^2 = R^2$$
$$\lambda(T) = \left(\frac{\partial \|z\|_{2,T}^2}{\partial x} \right)^T$$
$$\dot{\lambda}(t) = - \left(\frac{\partial f(x(t), w(t))}{\partial x} \right)^T \lambda(t)$$
$$w(t) = \mu \left(\frac{\partial f(x(t), w(t))}{\partial w} \right)^T \lambda(t),$$

for $t \in [0, T]$, where μ is chosen such that $\|w\|_{2,T} = R$.
Tierno, et.al., propose a power-like method to solve a similar maximization.

Gain Lower-Bound Power Algorithm

Adapting for this case yields: Pick $T > 0$ and w with $\|w\|_{2,T}^2 = R^2$. Repeat the following steps until w converges.

1. Compute $\|z\|_{2,T}$ (integration $\dot{x} = f(x, w)$ with $x(0) = 0$ forward in time).
2. Set $\lambda(T) = \left(\frac{\partial \|z\|_{2,T}^2}{\partial x} \right)^T$.
3. Compute the solution of $\dot{\lambda}(t) = -\frac{\partial f(x(t),w(t))}{\partial x}^T \lambda(t)$, $t \in [0, T]$ (integration backward in time).
4. Update $w(t) = \mu \left(\frac{\partial f(x(t),w(t))}{\partial w} \right)^T \lambda(t)$.

- Step (1) of each iteration gives a valid lower bound on the maximum (over $\|w\|_2 = R$) of $\|z\|_{2,T}$, independent of whether the iteration converges;
- (main point of Tierno) if dynamics are linear and p quadratic, then the iteration is convergent power iteration for $\mathcal{H}_\infty$.

Implemented in `worstcase`.

Local $\mathcal{L}_2$ gain: Guaranteed SOS feasibility

Consider

$$\begin{array}{rcll} \dot{x} & = & Ax + Bw + f_{23}(x) + g_{12}(x)w & (=: f(x,w)) \\ z & = & Cx + h_2(x) & (=: h(x)) \end{array}$$

where f_{23} is quadratic and cubic, g_{12} is linear and quadratic, h_2 is quadratic and A Hurwitz. Note (A, B, C) is linearization at 0.

Pick any γ with

$$\left\| C(sI - A)^{-1}B \right\|_\infty < \gamma$$

Theorem: The SOS-based dissipation inequalities for local $\mathcal{L}_2$ gain

$$R > 0, s_1 \in \Sigma\left[x, w\right], V - l_1 \in \Sigma\left[x\right]$$
$$-\left(\frac{dV}{dx} f - w^T w + \frac{1}{\gamma^2} z^T z + (R^2 - V)s_1\right) \in \Sigma\left[x, w\right]$$

are <u>always</u> feasible, using $\partial(V) = 2, \partial(s_1) = 2$.

Local $\mathcal{L}_2$ gain: Proof of Guaranteed SOS feasibility

Bounded-Real lemma implies there exist $P = P^T \succ 0$ such that

$$\Lambda := \begin{bmatrix} A^T P + PA + \gamma^{-2} C^T C & PB \\ B^T P & -I \end{bmatrix} \prec 0.$$

Define $V(x) := x^T P x$, $q := \begin{bmatrix} x \\ w \end{bmatrix}$, and $v := q \otimes x$. Then, there exist constant M_1, M_2

$$\begin{aligned} & x^T M_1 v + v^T M_1^T x + v^T M_2 v \\ & \qquad = 2x^T \left[P f_{23}(x) + C^T h_2(x) + P g_{12}(x) w \right] + h_2(x)^T h_2(x) \end{aligned}$$

Substitution gives

$$\frac{dV}{dx} f - w^T w + \frac{1}{\gamma^2} z^T z = \begin{bmatrix} q \\ v \end{bmatrix}^T \begin{bmatrix} \Lambda & \begin{bmatrix} -M_1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} -M_1^T & 0 \end{bmatrix} & -M_2 \end{bmatrix} \begin{bmatrix} q \\ v \end{bmatrix}$$

Local $\mathcal{L}_2$ gain: Proof of Guaranteed SOS feasibility

Since $P \succ 0$, there exist $H \succ 0$ with $z^T H z = (x^T x + w^T w) x^T P x$.

For any $\alpha > 0$, define $s_1(x, w) := \alpha(x^T x + w^T w)$.

Collecting together, for any $R > 0$,

$$D := -\left[\nabla V f(x, w) - w^T w + h^T(x) h(x)\right] - s_1(x, w)\left(R^2 - V\right)$$

is "quadratic" in x, w and v, specifically $D =$

$$\begin{bmatrix} q \\ z \end{bmatrix}^T \left(\begin{bmatrix} -\Lambda & \begin{bmatrix} -M_1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} -M_1^T & 0 \end{bmatrix} & -M_2 \end{bmatrix} + \begin{bmatrix} -\alpha R^2 & 0 \\ 0 & \alpha H \end{bmatrix} \right) \begin{bmatrix} q \\ z \end{bmatrix}$$

By proper choice of α, R, this can be made SOS,...

Local $\mathcal{L}_2$ gain: Proof of Guaranteed SOS feasibility

Since

$$H \succ 0, \Lambda \prec 0$$

there is a $\alpha > 0$ such that

$$\begin{bmatrix} -\Lambda & \begin{bmatrix} -M_1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} -M_1^T & 0 \end{bmatrix} & -M_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \alpha H \end{bmatrix} \succ 0$$

With this α fixed, by continuity there exist $R > 0$ such that

$$\begin{bmatrix} -\Lambda & \begin{bmatrix} -M_1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} -M_1^T & 0 \end{bmatrix} & -M_2 \end{bmatrix} + \begin{bmatrix} -\alpha R^2 & 0 \\ 0 & \alpha H \end{bmatrix} \succ 0$$

as desired

Local $\mathcal{L}_2$ gain: Summary

Consider

$$\begin{aligned} \dot{x} &= Ax + Bw + f_{23}(x) + g_{12}(x)w \qquad (=: f(x,w)) \\ z &= Cx + h_2(x) \qquad (=: h(x)) \end{aligned}$$

where f_{23} is quadratic and cubic, g_{12} is linear and quadratic, h_2 is quadratic, A Hurwitz, and $\left\|C(sI-A)^{-1}B\right\|_\infty < \gamma$.

Theorem: The SOS-based dissipation inequalities for local $\mathcal{L}_2$ gain

$$R > 0, s_1 \in \Sigma[x,w], V - l_1 \in \Sigma[x]$$
$$-\left(\tfrac{dV}{dx} f - w^T w + \tfrac{1}{\gamma^2} z^T z + (R^2 - V)s_1\right) \in \Sigma[x,w]$$

are <u>always</u> feasible, using $\partial(V) = 2, \partial(s_1) = 2$. Moreover, the inequality can be strengthened to include a positive-definite term, $l(x)$,

$$-\left(\frac{dV}{dx} f - w^T w + \frac{1}{\gamma^2} z^T z + (R^2 - V)s_1 + l(x)\right) \in \Sigma[x,w]$$

Upper bounds on the reachable set

$$\dot{x} = f(x, w) \qquad \text{with} \qquad f(0, 0) = 0$$

- Find upper bounds on the reachable set from the origin for bounded $\mathcal{L}_2$ input norm
  - Denote the set of points reached from the origin with input signals w such that $\|w\|_2 \leq R$ by $\texttt{Reach}_R$.

$$\texttt{Reach}_R := \{x(t) \ : \ x(0) = 0, \ t \geq 0, \ \|w\|_2 \leq R\}$$

Goal:

- Given a shape factor p (positive definite, convex function with $p(0) = 0$), establish relations of the form

$$x(0) = 0 \ \& \ \|w\|_2 \leq R \qquad \Rightarrow \qquad p(x(t)) \leq \beta \ \ \forall t \geq 0.$$

- Two types of optimization
  - Given R, minimize β
  - Given β, maximize R

A characterization of upper bounds on the reachable set

$$\dot{x} = f(x, w) \qquad \text{with} \quad f(0,0) = 0$$

Theorem: If there exists a continuously differentiable function V such that

- $V(x) > 0$ for all $x \neq 0$ and $V(0) = 0$
- $\Omega_{V,R^2} = \left\{\xi \ : \ V(\xi) \leq R^2\right\}$ is bounded
- $\nabla V f(x, w) \leq w^T w$ for all $x \in \Omega_{V,R^2}$ and for all $w \in \mathbb{R}^{n_w}$

then $\texttt{Reach}_R \subseteq \Omega_{V,R^2}$.

Given R, solve

$$\min_{V,\beta} \quad \beta$$
$$\text{s.t.} \quad \Omega_{V,R^2} \subseteq \Omega_{p,\beta}$$
V satisfies above conditions

OR

Given β, solve

$$\max_{V,R^2} \quad R^2$$
$$\text{s.t.} \quad \Omega_{V,R^2} \subseteq \Omega_{p,\beta}$$
V satisfies above conditions

Bilinear SOS problem formulation for reachability analysis

Original Problem

$$\max_{R^2,V} \quad R^2$$

subject to:

$$V(0) = 0, \quad V(x) > 0 \; \forall x \neq 0$$

$$\left\{x \in \mathbb{R}^n \; : \; V(x) \leq R^2\right\} \text{ is bounded}$$

$$\Omega_{V,R^2} \subseteq \Omega_{p,\beta}$$

$$\nabla V f(x,w) \leq w^T w \quad \forall \; x \in \Omega_{V,R^2} \; \& \; w \in \mathbb{R}^{n_w}$$

$\Uparrow$ S-procedure - SOS

Reformulation

$$\max_{R^2,V,s_1,s_2} \quad R^2$$

subject to:

$$-\left[(\beta - p) + (V - R^2)s_1\right] \text{ is SOS}[x],$$

$$-\left[(R^2 - V)s_2 + \nabla V f(x,w) + w^T w\right] \text{ is SOS}[x,w],$$

$V - \epsilon x^T x$ is SOS$[x]$, $V(0) = 0$, and

s_1, s_2, s_3 are SOS.

Reachability Refinement (exploit slack in $\dot{V} \leq w^T w$)

Suppose $g : \mathbb{R} \to (0, 1]$ is piecewise continuous, with

$$\frac{\partial V}{\partial x} f(x, w) \leq g(V(x)) w^T w \quad \forall x \in \{x : V(x) \leq R^2\}, \forall w \in \mathbb{R}^{n_w}.$$

Define

$$K(x) := \int_0^{V(x)} \frac{1}{g(\tau)} d\tau \qquad \text{and} \qquad R_e^2 := \int_0^{R^2} \frac{1}{g(\tau)} d\tau$$

Note that $K(0) = 0, K(x) > 0$, $\frac{\partial K}{\partial x} = \frac{1}{g(V(x))} \frac{\partial V}{\partial x}$, $R_e \geq R$ and

$$\left\{x : V(x) \leq R^2\right\} = \left\{x : K(x) \leq R_e^2\right\}.$$

Divide tightened inequality by g,

$$\frac{\partial K}{\partial x} f(x, w) \leq w^T w \quad \forall x \in \{x : K(x) \leq R_e^2\}, \forall w \in \mathbb{R}^{n_w}.$$

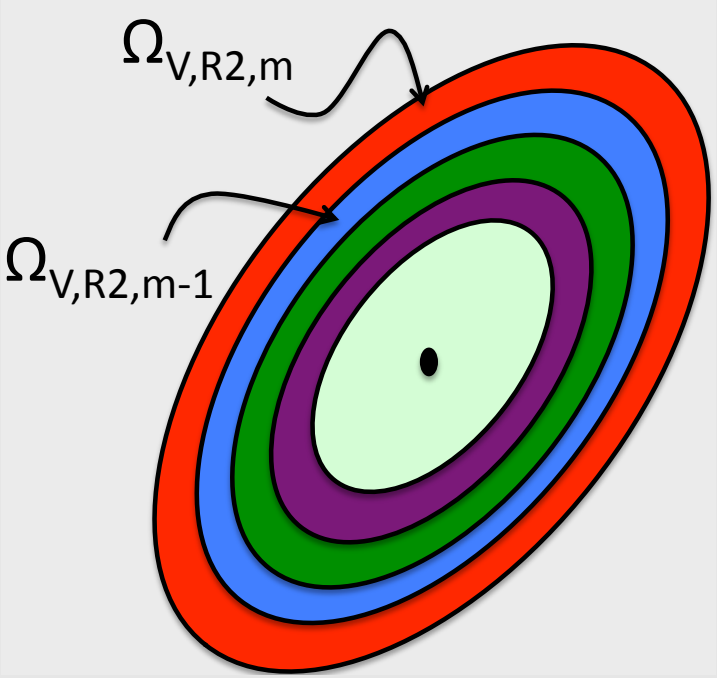
So K establishes a reachability bound, namely

$$\begin{aligned} x(0) = 0, \|w\|_{2,T} \leq R_e \quad &\Rightarrow \quad K(x(T)) \leq \|w\|_{2,T}^2 \left(\leq R_e^2\right) \\ &\Rightarrow \quad V(x(T)) \leq R^2 \quad \text{(refined bound)} \end{aligned}$$

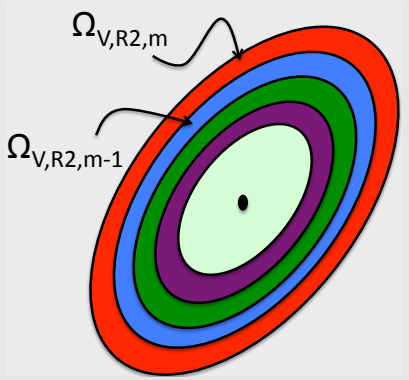
Computing a refinement function g (1)

- For given feasible V and R, search for g can be formulated as a sequence of SOS programming problems.
- Restrict g to be piecewise constant (extensions to piecewise polynomial g are straightforward).
- Let $m > 0$ be an integer, define $\epsilon := R^2/m$, and partition the set Ω_{V,R^2} into m annuli

$$\Omega_{V,R^2,k} := \{x \in \mathbb{R}^n \ : \ (k-1)\epsilon \leq V(x) \leq k\epsilon\} \ \text{ for } k = 1, \ldots, m.$$



Computing a refinement function g (2)



Given numbers $\{g_k\}_{k=1}^m$, define

$$g(\tau) = g_k \qquad \forall \epsilon(k-1) \leq \tau < \epsilon k$$

Piecewise-constant function g satisfies

$$\frac{\partial V}{\partial x} f(x,w) \leq g(V(x)) w^T w \quad \forall x \in \{x : V(x) \leq R^2\}, \forall w \in \mathbb{R}^{n_w}.$$

if and only if for all k

$$\frac{\partial V}{\partial x} f(x,w) \leq g_k w^T w \quad \forall x \in \{x : \epsilon(k-1) \leq V(x) < \epsilon k\}, \forall w \in \mathbb{R}^{n_w}.$$

This motivates m separate, uncoupled SOS optimizations, namely minimize g_k such that s_{1k} and s_{2k} are SOS

$$g_k w^T w - \nabla V f(x,w) - s_{1k}(k\epsilon - V) - s_{2k}(V - (k-1)\epsilon) \in \Sigma[(x,w)].$$

For this g : $$R_e^2 := \int_0^{R^2} \frac{1}{g(\tau)} d\tau = \epsilon \sum_{k=1}^{m} \frac{1}{g_k}$$

Reachability Lower-Bound Power Algorithm

For any $T > 0$ and β obtained from SOS analysis

$$\max_{\substack{w \in \mathcal{L}_2[0,T] \\ ||w||_2 \leq R}} p(x(T)) \leq \max_{\substack{t \geq 0 \\ w \in \mathcal{L}_2[0,\infty) \\ ||w||_2 \leq R}} p(x(t)) \leq \beta$$

The first-order conditions for stationarity of the finite horizon maximum are the existence of signals (x, λ) and w which satisfy

$$\begin{aligned}
\dot{x} &= f(x, w) \\
\|w\|_{2,T}^2 &= R^2 \\
\lambda(T) &= \left(\frac{\partial p(x(T))}{\partial x} \right)^T \\
\dot{\lambda}(t) &= -\left(\frac{\partial f(x(t), w(t))}{\partial x} \right)^T \lambda(t) \\
w(t) &= \mu \left(\frac{\partial f(x(t), w(t))}{\partial w} \right)^T \lambda(t),
\end{aligned}$$

for $t \in [0, T]$, where μ is chosen such that $\|w\|_{2,T}^2 = R^2$. Tierno, et.al., propose a power-like method to solve a similar maximization.

Reachability Lower-Bound Power Algorithm

Adapting for this case yields: Pick $T > 0$ and w with $\|w\|_{2,T}^2 = R^2$. Repeat the following steps until w converges.

1. Compute $p(x(T))$ (integration $\dot{x} = f(x, w)$ with $x(0) = 0$ forward in time).
2. Set $\lambda(T) = \left(\frac{\partial p(x(T))}{\partial x}\right)^T$.
3. Compute the solution of $\dot{\lambda}(t) = -\frac{\partial f(x(t),w(t))}{\partial x}^T \lambda(t)$, $t \in [0, T]$ (integration backward in time).
4. Update $w(t) = \mu \left(\frac{\partial f(x(t),w(t))}{\partial w}\right)^T \lambda(t)$.

- Step (1) of each iteration gives a valid lower bound on the maximum (over $\|w\|_2 = R$) of $p(x(T))$, independent of whether the iteration converges;
- (main point of Tierno) if dynamics are linear and p quadratic, then the iteration is convergent power iteration for operator norm of $w \to p(x(T))$.

Implemented in `worstcase` (used in the demos later).

Products of simple quadratic functions

For vectors x and y, define

$$x \otimes y := \begin{bmatrix} x_1 y \\ x_2 y \\ \vdots \\ x_n y \end{bmatrix}$$

For any matrix M

$$(x^T x) y^T M y = (x \otimes y)^T \begin{bmatrix} M & 0 & \dots & 0 \\ 0 & M & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M \end{bmatrix} (x \otimes y)$$

Reachability: Guaranteed SOS feasibility

Consider

$$\begin{aligned} \dot{x}_1(t) &= A_{11}x_1(t) + b(x_1, x_2) + Ew \\ \dot{x}_2(t) &= q(x_1) \end{aligned}$$

where b is bilinear, q purely quadratic, and A_{11} Hurwitz.

This has marginally stable linearization at $x = 0$, and is a common structure related to some adaptive systems.

Theorem: The SOS-based dissipation inequalities for bounded reachability,

$$\begin{gathered} R > 0, s_1 \in \Sigma\left[x, w\right], V - l_1 \in \Sigma\left[x\right] \\ -\left(\tfrac{dV}{dx} f - w^T w + (R^2 - V)s_1\right) \in \Sigma\left[x, w\right] \end{gathered}$$

are <u>always</u> feasible, using $\partial(V) = 2, \partial(s_1) = 2$.

Reachability: Proof of Guaranteed SOS feasibility

1. Choose $Q_1, Q_2 \succ 0$ with

$$\begin{bmatrix} A_{11}^T Q_1 + Q_1 A_{11} & Q_1 E \\ E^T Q_1 & -I \end{bmatrix} \prec 0$$

2. Set $V(x) := x_1^T Q_1 x_1 + x_2^T Q_2 x_2$.
3. Let $s_1(x) := \alpha x_1^T x_1$ (α to be chosen...)
4. Define $M_1 \succ 0$, $M_2 \succ 0$, and B_1 and B_2 to satisfy identities

$$\begin{aligned} x_1^T Q_1 b(x_1, x_2) &= x_1^T B_1 (x_1 \otimes x_2) \\ x_2^T Q_2 q(x_1) &= x_1^T B_2 (x_1 \otimes x_2) \\ x_1^T Q_1 x_1 x_1^T x_1 &= (x_1 \otimes x_1)^T M_1 (x_1 \otimes x_1) \\ x_2^T Q_2 x_2 x_1^T x_1 &= (x_1 \otimes x_2)^T M_2 (x_1 \otimes x_2). \end{aligned}$$

Reachability: Proof of Guaranteed SOS feasibility

Write $\left(\frac{dV}{dx}f - w^T w + (R^2 - V)s_1\right)$ as $z^T H z$, with

$$z := \begin{bmatrix} x_1 \\ w \\ x_1 \otimes x_1 \\ x_1 \otimes x_2 \end{bmatrix}$$

and

$$H = \begin{bmatrix} A_{11}^T Q_1 + Q_1 A_{11} + \alpha R^2 I & Q_1 E & 0 & B_1 + B_2 \\ E^T Q_1 & -I & 0 & 0 \\ 0 & 0 & -\alpha M_1 & 0 \\ B_1^T + B_2^T & 0 & 0 & -\alpha M_2 \end{bmatrix}$$

If $R = 0$, then for large enough α, $H \prec 0$. With such a large α, the H remains negative definite for some $R > 0$.

Generalizations: dissipation inequalities

The system

$$\dot{x} = f(x, w)$$
$$z = h(x)$$

with $f(0,0) = 0$ and $h(0) = 0$ is said to be **dissipative** w.r.t. to the **supply rate** $r : (w, z) \mapsto \mathbb{R}$ if there exists a positive definite function V such that $V(0) = 0$ and the following dissipation inequality (DIE) holds

$$\frac{\partial V}{\partial x} f(x, w) \leq r(w, z)$$

for all $x \in \mathbb{R}^n$ & $w \in \mathbb{R}^{n_w}$.

- $\mathcal{L}_2 \to \mathcal{L}_2$ **gain:** $r(w, z) = w^T w - z^T z$
- **Reachability:** $r(w, z) = w^T w$

The system is said to be **locally** dissipative if the above DIE holds only for all $x \in \{x \ : \ V(x) \leq \gamma\}$ for some $\gamma > 0$.

Incorporating $\mathcal{L}_\infty$ constraints on w

• In local gain and reachability analysis with $\|w\|_2 \leq R$, the dissipation inequalities held on

$$\{x \in \mathbb{R}^n \ : \ V(x) \leq R^2\} \times \mathbb{R}^{n_w}.$$

• If $w^T(t)w(t) \leq \alpha$ for all t, then the dissipation inequality only needs to hold on

$$\{x \in \mathbb{R}^n \ : \ V(x) \leq R^2\} \times \{w \in \mathbb{R}^{n_w} \ : \ w^T w \leq \alpha\}.$$

- Incorporate the $\mathcal{L}_\infty$ bounds on w using the S-procedure.

Incorporating $\mathcal{L}_\infty$ constraints on w

- $\mathcal{L}_2 \to \mathcal{L}_2$ **gain analysis:**

Original: $$-\left[(R^2 - V)s_1 + \nabla V f - w^T w + \gamma^{-2} z^T z\right] \in \Sigma[(x, w)]$$

New:

$$-\left[(R^2 - V)s_1 + \nabla V f - w^T w + \gamma^{-2} z^T z\right] - s_2(\rho - w^T w) \in \Sigma[(x, w)]$$

- **Reachability analysis:**

Original: $$-\left[(R^2 - V)s_1 + \nabla V f - w^T w\right] \in \Sigma[(x, w)]$$

New:

$$-\left[(R^2 - V)s_1 + \nabla V f - w^T w\right] - s_2(\rho - w^T w) \in \Sigma[(x, w)]$$

[In all constraints above: $s_1, s_2 \in \Sigma[(x, w)]$]

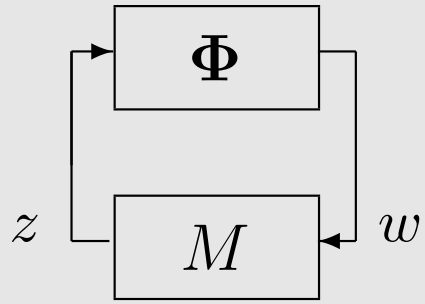
Outline

- Motivation
- Preliminaries
- ROA analysis using SOS optimization and solution strategies
- Robust ROA analysis with parametric uncertainty
- Local input-output analysis
- **Robust ROA and performance analysis with unmodeled dynamics**
- F-18

Recall: the small-gain theorem

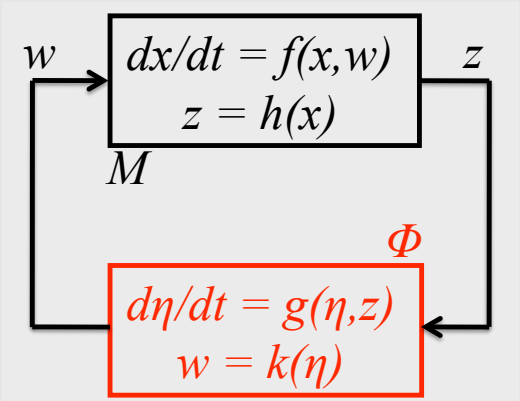
For stable M and Φ, the feedback interconnection is internally stable if

$$\gamma(M)\gamma(\Phi) < 1.$$



- γ is an upper bound on the global $\mathcal{L}_2 \to \mathcal{L}_2$ gain.
- Extensively used in linear robustness analysis where M is linear time-invariant (existence of global gains is guaranteed).
- How to generalize to nonlinear M with possibly only local gain relations?

Local small-gain theorems for stability analysis



Let l be a positive definite function with $l(0) = 0$ e.g. $l(x) = \epsilon x^T x$ and $R > 0$.
Let $\tilde{l}$ be a positive definite function with $\tilde{l}(0) = 0$.

For M: There exists a positive definite function V such that Ω_{V,R^2} is bounded and for all $x \in \Omega_{V,R^2}$ and $w \in \mathbb{R}^{n_w}$

$$\nabla V \cdot f(x,w) \leq w^T w - h(x)^T h(x) - l(x).$$

[M is "locally strictly dissipative" w.r.t. the supply rate $w^T w - z^T z$ certified by the storage function V.]

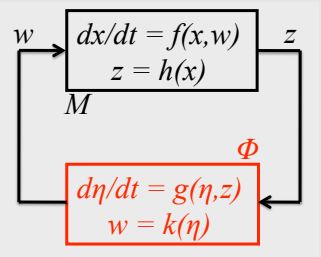
For Φ: There exists a positive definite function Q such that for all $\eta \in \mathbb{R}^{n_\eta}$ and $z \in \mathbb{R}^{n_z}$

$$\nabla Q \cdot g(\eta,z) \leq z^T z - k(\eta)^T k(\eta) - \tilde{l}(\eta).$$

[Φ is "strictly dissipative" w.r.t. $z^T z - w^T w$.]

Local small-gain theorems for stability analysis (2)

Conclusion: $S := V + Q$ is a Lyapunov function for the closed-loop for the closed-loop dynamics ($\dot{\xi} = F(\xi)$).



$$\xi = \begin{bmatrix} x \\ \eta \end{bmatrix}$$

Proof:

$$\begin{aligned} \nabla V \cdot f(x,w) &\leq w^T w - z^T z - l(x) && \forall x \in \Omega_{V,R^2} \ \& \ w \in \mathbb{R}^{n_w} \\ \nabla Q \cdot g(\eta,z) &\leq z^T z - w^T w - \tilde{l}(\eta) && \forall \eta \in \mathbb{R}^{n_\eta} \ \& \ z \in \mathbb{R}^{n_z} \end{aligned}$$

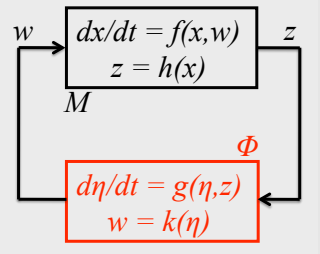
$$\nabla V \cdot f(x, g(\eta)) + \nabla Q \cdot g(\eta, h(x)) \leq l(x) + \tilde{l}(\eta)$$
$$\forall (x,\eta) \in \left\{ (x,\eta) \ : \ V(x) + Q(\eta) \leq R^2 \right\}$$

$$\nabla S \cdot F(\xi) \leq -l(x) - \tilde{l}(\eta) = -L(\xi)$$
$$\forall (x,\eta) \in \left\{ (x,\eta) \ : \ S(x,\eta) \leq R^2 \right\}$$

Corollary:

- $\{(x,\eta) \ : \ V(x) + Q(\eta) \leq R^2\}$ is an invariant subset of the ROA for the closed-loop dynamics.

Interpretation for the states x and (x, η)



$$\xi = \begin{bmatrix} x \\ \eta \end{bmatrix}$$

$\{(x, \eta) \ : \ V(x) + Q(\eta) \leq R^2\} = \{\xi \ : \ S(\xi) \leq R^2\}$ is an invariant subset of the ROA for the closed-loop dynamics ($\dot{\xi} = F(\xi)$).

Consequently,

- For $\eta(0) = 0$ and any $x(0) \in \Omega_{V,R^2}$, $x(t) \in \Omega_{V,R^2}$ for all $t \geq 0$ and $x(t) \to 0$ as $t \to \infty$.
- For any $\xi(0) = (x(0), \eta(0)) \in \Omega_{S,R^2}$, $x(t) \in \Omega_{V,R^2}$ for all $t \geq 0$ and $x(t) \to 0$ as $t \to \infty$.

For $\eta(0) = 0$, conclusions on x hold even if Φ is not known but known to be strictly dissipative w.r.t. $z^T z - w^T w$.

Estimating the ROA (for x states)

Let p be a shape factor (as before) and $(\bar{V}, \bar{\beta}, \bar{R})$ be a solution to the above optimization

$$\max_{V \in \mathcal{V}, \beta \geq 0, R \geq 0} \quad \beta \quad \text{subject to}$$

$$V(x) > 0 \text{ for all } x \neq 0, \quad V(0) = 0,$$

$$\Omega_{p,\beta} \subseteq \Omega_{V,R^2},$$

$$\Omega_{V,R^2} \text{ is bounded},$$

$$\nabla V f(x,w) \leq w^T w - z^T z - l(x) \quad \forall\, x \in \Omega_{V,R^2}, \quad \forall\, w \in \mathbb{R}^{n_w}.$$

If Φ is strictly dissipative w.r.t. $z^T z - w^T w$ and $\eta(0) = 0$, then for any $x(0) \in \Omega_{p,\bar{\beta}}$,

- $x(t)$ stays in $\Omega_{\bar{V},\bar{R}^2}$
- $x(t) \to 0$ as $t \to \infty$.

Estimating the ROA - SOS problem

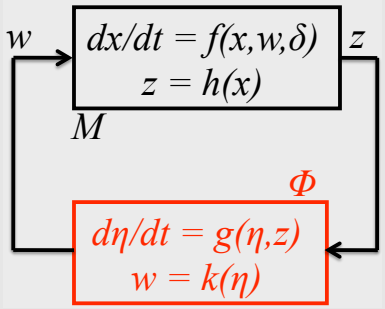
Original problem:

$$\max_{V\in\mathcal{V},\beta\geq 0,R\geq 0} \quad \beta \quad \text{subject to}$$
$$V(x) > 0 \text{ for all } x \neq 0, \quad V(0) = 0,$$
$$\Omega_{p,\beta} \subseteq \Omega_{V,R^2},$$
$$\Omega_{V,R^2} \text{ is bounded,}$$
$$\nabla V f(x,w) \leq w^T w - z^T z - l(x) \quad \forall\, x \in \Omega_{V,R^2}, \quad \forall\, w \in \mathbb{R}^{n_w}.$$

Use the S-procedure and standard relaxations to obtain a SOS reformulation:

$$\max_{V\in\mathcal{V}_{poly},\beta\geq 0,R\geq 0,s_1\in\mathcal{S}_1,s_2\in\mathcal{S}_2} \beta \text{ subject to}$$
$$V - l_1 \in \Sigma[x], \quad s_1 \in \Sigma[x], \quad s_2 \in \Sigma[(x,w)],$$
$$(R^2 - V) - s_1(\beta - p) \in \Sigma[x],$$
$$-\nabla V f + w^T w - z^T z - l(x) - s_2(R^2 - V) \in \Sigma[(x,w)].$$

Incorporating parametric uncertainties in M



Uncertain parameters δ in the vector fields f can be handled as before.

- Restrict to polytopic Δ and affine δ dependence

$$\dot{x}(t) = f_0(x(t), w(t)) + F(x(t))\delta$$

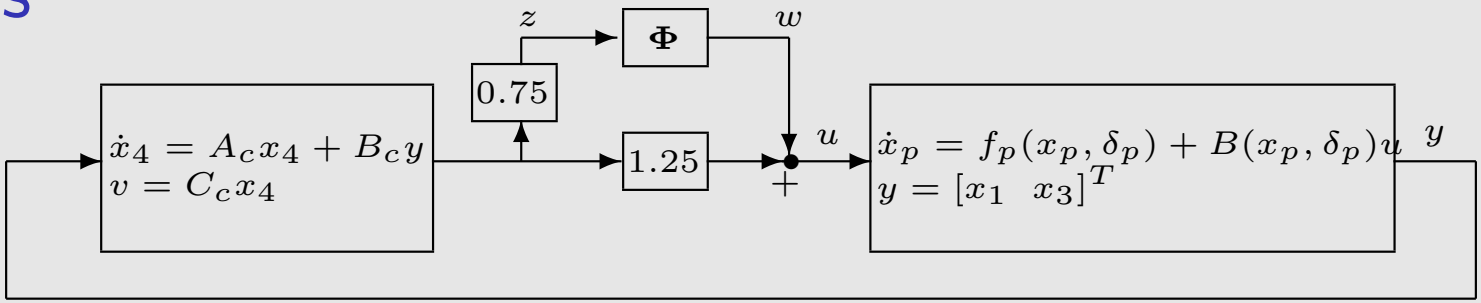
Resulting SOS condition is affine in δ

$$-\nabla V(f_0(x,w)+F(x)\delta)+w^T w-z^T z-l(x)-s_2(R^2-V) \in \Sigma[(x,w)]$$

and if it holds for the vertices of Δ then it holds for all $\delta \in \Delta$.

- Branch-and-bound in Δ
- Coverings for non-affine δ dependence and non-polytopic Δ

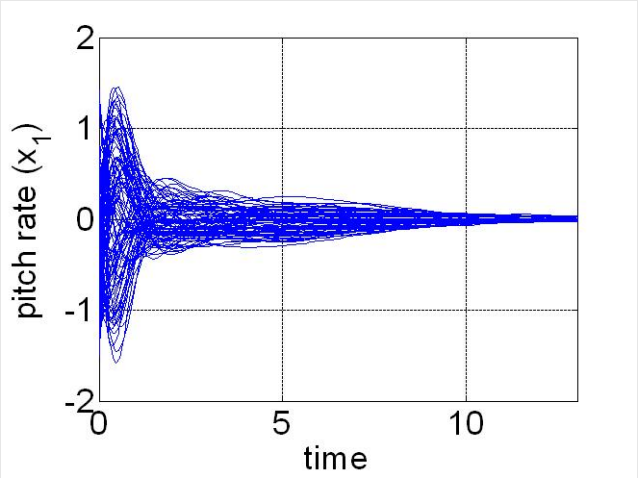
Example: Controlled aircraft dynamics with unmodeled dynamics

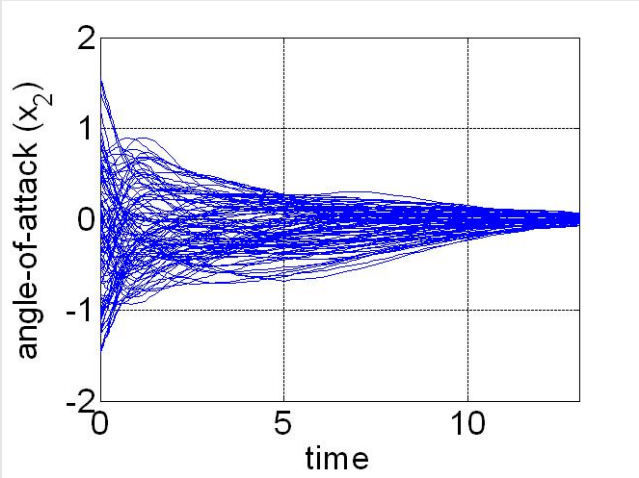


	no δ_p	with δ_p
no Δ	9.4 / 16.1	5.5 / 7.9
with Δ	4.2 / 6.7	2.4 / 4.1

In the table :
$(\partial(V) = 2/\partial(V) = 4)$

Closed-loop response with randomly generated first-order LTI Φ:





Relaxed version of the local small-gain theorem

Relax the strict dissipation for Φ by dissipation (i.e., $\tilde{l}(\eta) = 0$):

$$\nabla Q g(\eta, z) \leq z^T z - w^T w \qquad \forall \eta \in \mathbb{R}^{n_\eta} \;\; \& \;\; z \in \mathbb{R}^{n_z}$$

Weaker conclusion: For $\eta(0) = 0$ and for any $0 < \tilde{R} < R$,

- $x(0) \in \Omega_{V,\tilde{R}^2} \quad \Rightarrow \quad (x(t), \eta(t)) \in \Omega_{S,\tilde{R}^2} \quad \forall t \geq 0$
- $x(0) \in \Omega_{V,\tilde{R}^2} \quad \Rightarrow \quad x(t) \in \Omega_{V,\tilde{R}^2} \quad \forall t \geq 0 \;\; \& \lim_{t\to\infty} x(t) = 0$

Proof idea: arguments as before + Barbalat's lemma

Unit gain $\longrightarrow$ Gains γ and $1/\gamma$

Conditions (that hold appropriately for x, η, w, z as indicated before)

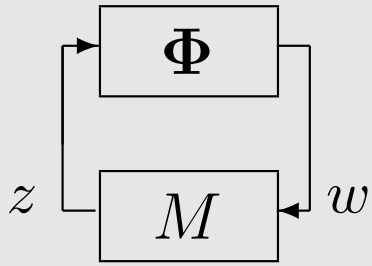
$$\begin{aligned} \nabla V f(x,w) &\leq w^T w - z^T z - l(x) \qquad \Rightarrow \|M\|_\infty < 1 \\ \nabla Q g(\eta, z) &\leq z^T z - w^T w - \tilde{l}(\eta) \qquad \Rightarrow \|\Phi\|_\infty < 1. \end{aligned}$$

These gain conditions can relaxed to gain of non-unity.

Previous results will hold when the dissipation inequalities are replaced by

$$\begin{aligned} \nabla V f(x,w) &\leq w^T w - \gamma^2 z^T z - l(x) \qquad (\Rightarrow \|M\|_\infty < 1/\gamma) \\ \nabla Q g(\eta, z) &\leq \gamma^2 z^T z - w^T w - \tilde{l}(\eta) \qquad (\Rightarrow \|\Phi\|_\infty < \gamma). \end{aligned}$$

Generalization to generic supply rates



Results hold when the "$\mathcal{L}_2$-gain supply rate" is replaced by a general supply rate.

Suppose that

- Φ is strictly dissipative w.r.t. the supply rate $r_1(z, w)$ with the corresponding storage function Q
- M satisfies

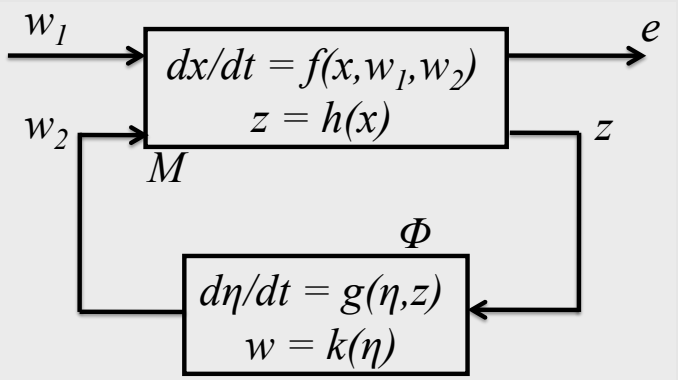
$$V f(x, w) \leq r_2(w, z) - l(x) \quad \forall x \in \Omega_{V,R^2} \ \& \ w \in \mathbb{R}^{n_w}$$

with

$$r_1(z, w) = -r_2(w, z) \quad \forall w, z.$$

Then, $\{(x, \eta) \ : \ V(x) + Q(\eta) \leq R^2\}$ is an invariant subset of the ROA for the closed-loop dynamics.

Local small-gain theorems for performance analysis



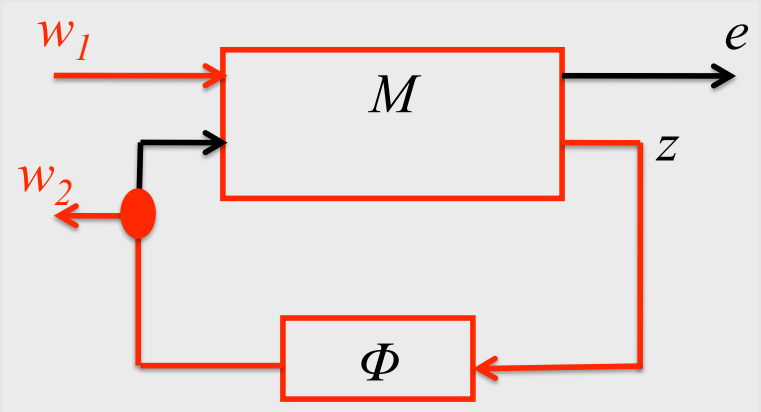
Global gain condition on Φ: starting from rest Φ satisfies

$$\|w_2\|_2 = \|\Phi(z)\|_2 \leq \|z\|_2.$$

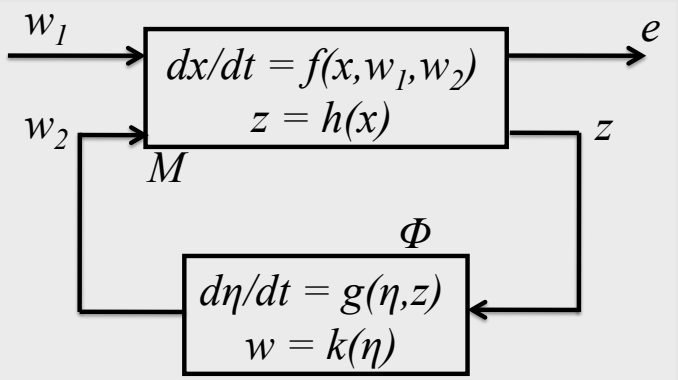
Goal: Find an upper bound on $\|e\|_2$ provided that M and Φ start from rest and $\|w_1\|_2 \leq R$.

Strategy:

- Bound $\|w_2\|_2$ in terms of $\|w_1\|_2$.



Local small-gain theorems for performance analysis



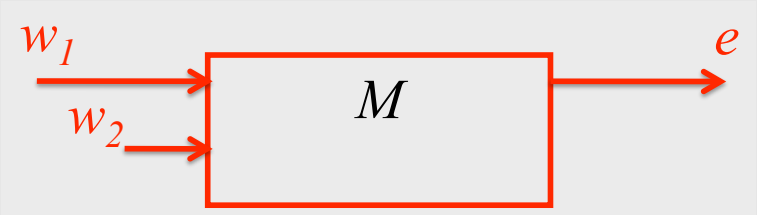
Global gain condition on Φ: starting from rest Φ satisfies

$$\|w_2\|_2 = \|\Phi(z)\|_2 \leq \|z\|_2.$$

Goal: Find an upper bound on $\|e\|_2$ provided that M and Φ start from rest and $\|w_1\|_2 \leq R$.

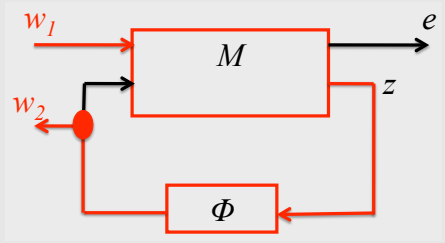
Strategy:

- Bound $\|w_2\|_2$ in terms of $\|w_1\|_2$.
- Bound $\|e\|_2$ in terms of $\|w_1\|_2$.



- Each step is a separate local gain analysis.

Step 1 (bound $\|w_2\|_{2,T}$ in terms of $\|w_1\|_{2,T}$):
For $R > 0$, $0 < \alpha < 1$ and $\beta > 0$, if $\exists\ \mathcal{C}^1$ function V s. t. $V(0) = 0$, $V(x) > 0\ \forall x \neq 0$, Ω_{V,R^2} is bounded,



$$\nabla V f(x, w_1, w_2) \leq \beta^2 w_1^T w_1 + w_2^T w_2 - \frac{1}{\alpha^2} z^T z$$

$\forall x \in \Omega_{V,R^2}$, $w_1 \in \mathbb{R}^{n_{w_1}}$, and $w_2 \in \mathbb{R}^{n_{w_2}}$, then for Φ starting from rest and for all $T \geq 0$

$$x(0) = 0\ \&\ \|w_1\|_{2,T} \leq R/\beta \qquad \Rightarrow \qquad \|w_2\|_{2,T} \leq \alpha R/\sqrt{1-\alpha^2}.$$

Step 2 (bound $\|e\|_{2,T}$ in terms of $\|w_1\|_{2,T}$): In addition to above conditions, if $\exists\ \mathcal{C}^1$ function Q s.t. $Q(0) = 0$, $Q(x) > 0\ \forall x \neq 0$, and

$$\nabla Q f(x, w_1, w_2) \leq \beta^2 w_1^T w_1 + w_2^T w_2 - \frac{1}{\gamma^2} e^T e$$

$\forall x \in \Omega_{Q,R^2/(1-\alpha^2)}$, $w_1 \in \mathbb{R}^{n_{w_1}}$, $w_2 \in \mathbb{R}^{n_{w_2}}$, then for $\mathbf{\Phi}$ starting from rest and for all $T \geq 0$

$$x(0) = 0\ \&\ \|w_1\|_{2,T} \leq R/\beta \qquad \Rightarrow \qquad \|e\|_{2,T} \leq \gamma R/\sqrt{1-\alpha^2}.$$

OTHER APPLICATIONS

Control Lyapunov functions $\dot{x} = f(x) + g(x)u$

If V is smooth, radially unbounded, positive-definite, and

$$\{x : \nabla V \cdot g(x) = 0\} \subseteq \{x : \nabla V \cdot f(x) < 0\} \bigcup \{0\}$$

then V is a *Control Lyapunov function* (CLF), and $\forall \alpha > 0$, the control defined via $u(x) := -\left(\nabla V \cdot g\right)^T \gamma(x)$, where

$$a(x) := \nabla V \cdot f(x), \qquad b(x) := \nabla V \cdot g(x),$$

and $\beta(x) := b(x)b^T(x)$, and

$$\gamma(x) := \begin{cases} \frac{a + \sqrt{a^2 + \alpha^2 \beta^2}}{\beta} & \text{if } \beta > 0 \\ 0 & \text{if } \beta = 0 \end{cases}$$

is continuous at 0, smooth everywhere else, and globally stabilizing.

Control Lyapunov functions

Containment condition

$$\{x : \nabla V \cdot g(x) = 0\} \subseteq \{x : \nabla V \cdot f(x) < 0\} \bigcup \{0\}$$

is satisfied if for positive-definite l_1, l_2, $\exists s \in \Sigma[x], p \in \mathbb{R}^{n_u}[x]$

$$V - l_1 \in \Sigma[x], \quad -((\nabla V \cdot f)\, s + (\nabla V \cdot g)\, p + l_2) \in \Sigma[x]$$

Tan (thesis) solves a global (quadratic) CLF for

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -x_1 + x_2^2 \\ -2x_2 \\ 3x_3 + x_2^2 \end{bmatrix} + \begin{bmatrix} -2 \\ 1 + x_3^2 \\ 1 + 4x_1^2 \end{bmatrix} u_1 + \begin{bmatrix} 5x_1 \\ 1 - x_2^2 \\ 3 \end{bmatrix} u_2$$

(from Banks, Salamci, Ozgoren, '99) with $\partial(s) = 0, \partial(p) = 1$.
Local version, change to include sublevel set of V...

$$\{x : V(x) \leq 1, \nabla V \cdot g(x) = 0\} \subseteq \{x : \nabla V \cdot f(x) < 0\} \bigcup \{0\}$$

Outline

- Motivation
- Dynamical system analysis
- Preliminaries
- ROA analysis using SOS optimization and solution strategies
- Robust ROA analysis with parametric uncertainty
- Local input-output analysis
- Robust ROA and performance analysis with unmodeled dynamics
- **F-18**

F/A-18 Out-of-Control Flight

- The US Navy has lost many F/A-18 A/B/C/D Hornet aircraft due to an out-of-control flight (OCF) departure phenomenon described as 'falling leaf' mode.

F/A-18 : NASA Dryden Photo

- The falling leaf mode is associated with sustained oscillatory OCF mode that can require 4.5K-6K meters altitude to recover*.

* Heller, David and Holmberg, "Falling Leaf Motion Suppression in the F/A-18 Hornet with Revised Flight Control Software," AIAA-2004-542, 42^{nd} *AIAA Aerospace Sciences Meeting*, Jan 2004, Reno, NV.

Analysis of F/A-18 Aircraft Flight Control Laws Motivation

- Administration action by the Naval Air Systems Command (NAVAIR) to prevent aircraft losses due to falling leaf entry to focused on
  - aircrew training,
  - restrictions on angle-of-attack and
  - center-of-gravity location.

F/A-18 Hornet: NASA Dryden Photo

- A solution to falling leaf mode entry was also pursued via modification of the *baseline flight control law*.
- The *revised control law* was tested and integrated into the F/A-18 E/F Super Hornet aircraft.

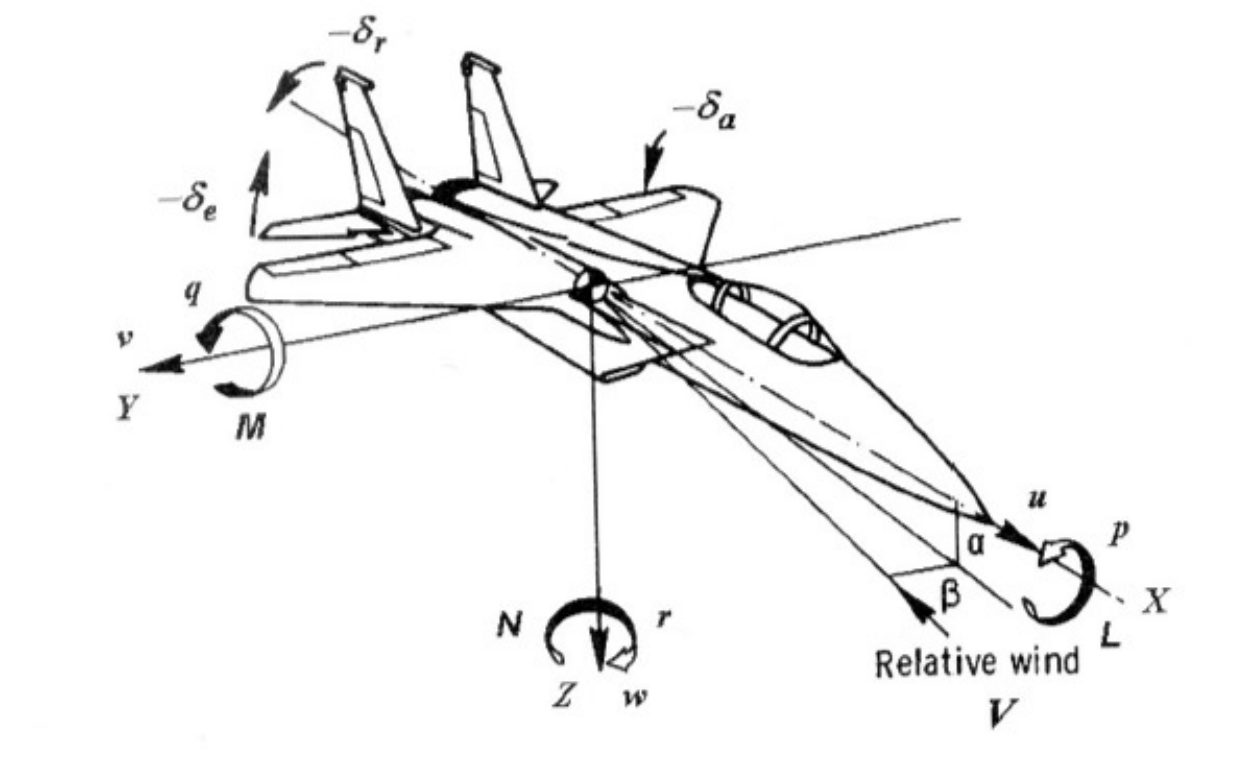
Analysis of F/A-18 Aircraft Flight Control Laws Motivation

Flight Control Law Analysis Objectives*:

- Identify the susceptibility of F/A-18 baseline and revised flight control laws to entry in falling leaf mode.
- Identify limits on the F/A-18 aircraft angle-of-attack (α) and sideslip (β) to prevent falling leaf entry for both F/A-18 flight control laws.

*Chakraborty, Seiler, and Balas, "Applications of Linear and Nonlinear Robustness Analysis Techniques to the F/A-18 Flight Control Laws," *AIAA Guidance, Navigation, and Control Conference, Chicago, IL, August 2009*

Aircraft Terminology



Terminology:

α = Angle-of-attack
β = Sideslip Angle
p = Roll Rate
L = Rolling Moment
q = Pitch Rate
M = Pitching Moment
r = Yaw Rate
N = Yawing Moment
u = Velocities in X-direction
v = Velocities in Y-direction
w = Velocities in Z-direction

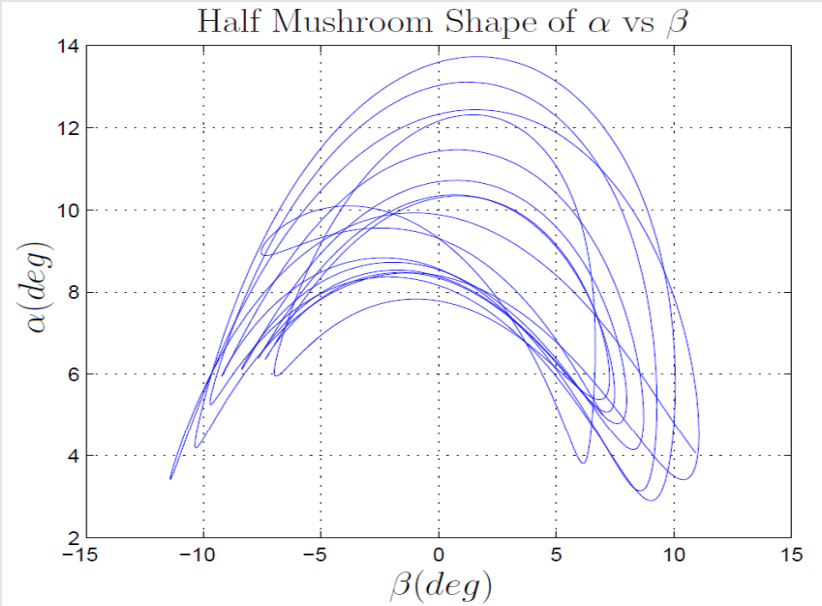
a_y = Lateral Acceleration

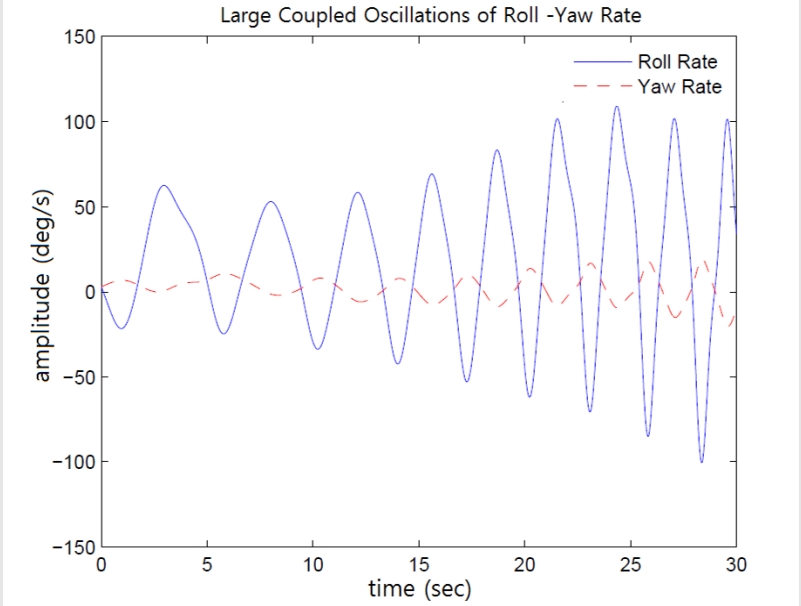
Reference: Klein Vladislav & Morelli A. Eugene, 'Aircraft System Identification: Theory and Practice' *AIAA Education Series, 2006*

Characteristics of Falling Leaf Mode

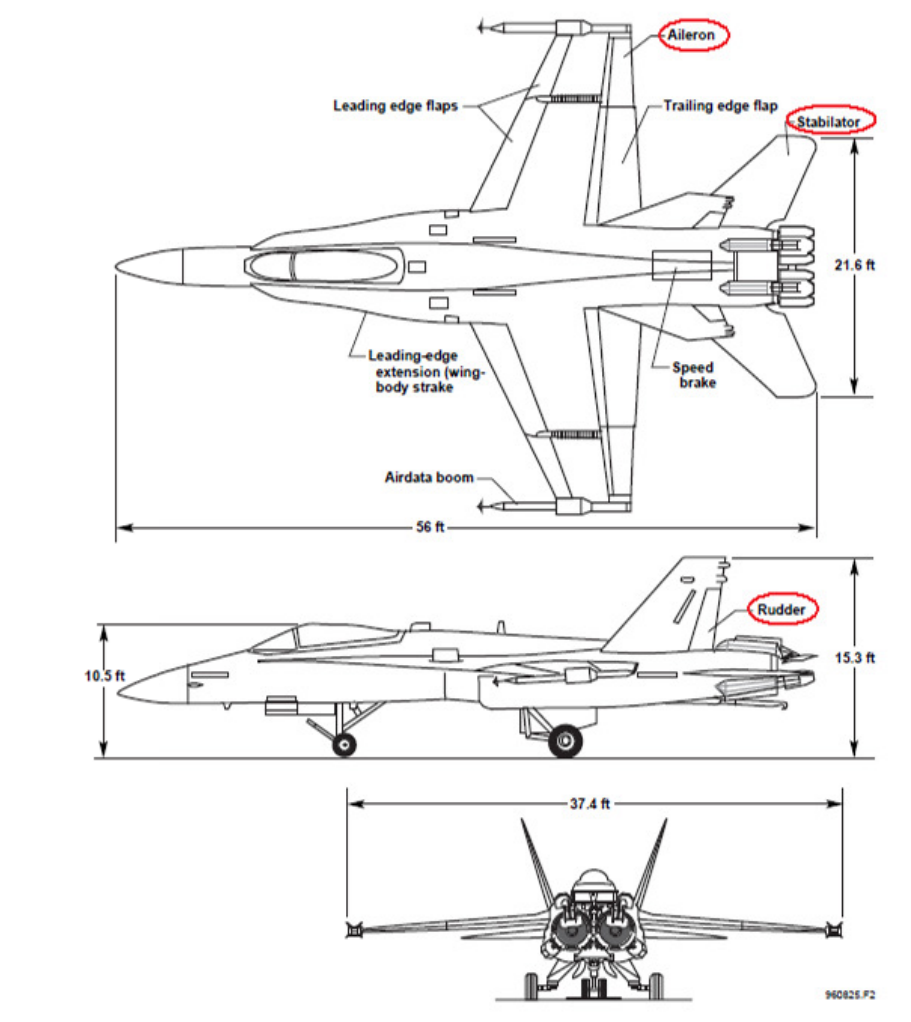
Falling leaf mode characterized by large coupled oscillations in all three axes, with large fluctuations in the angle-of-attack (α) and sideslip (β).

- Result of interaction between aerodynamic and kinematic effects, highly nonlinear.
- Vehicle often has small aerodynamic rate damping.
- Roll/yaw rates generated by aerodynamic effects of sideslip.





F/A-18 Geometric View

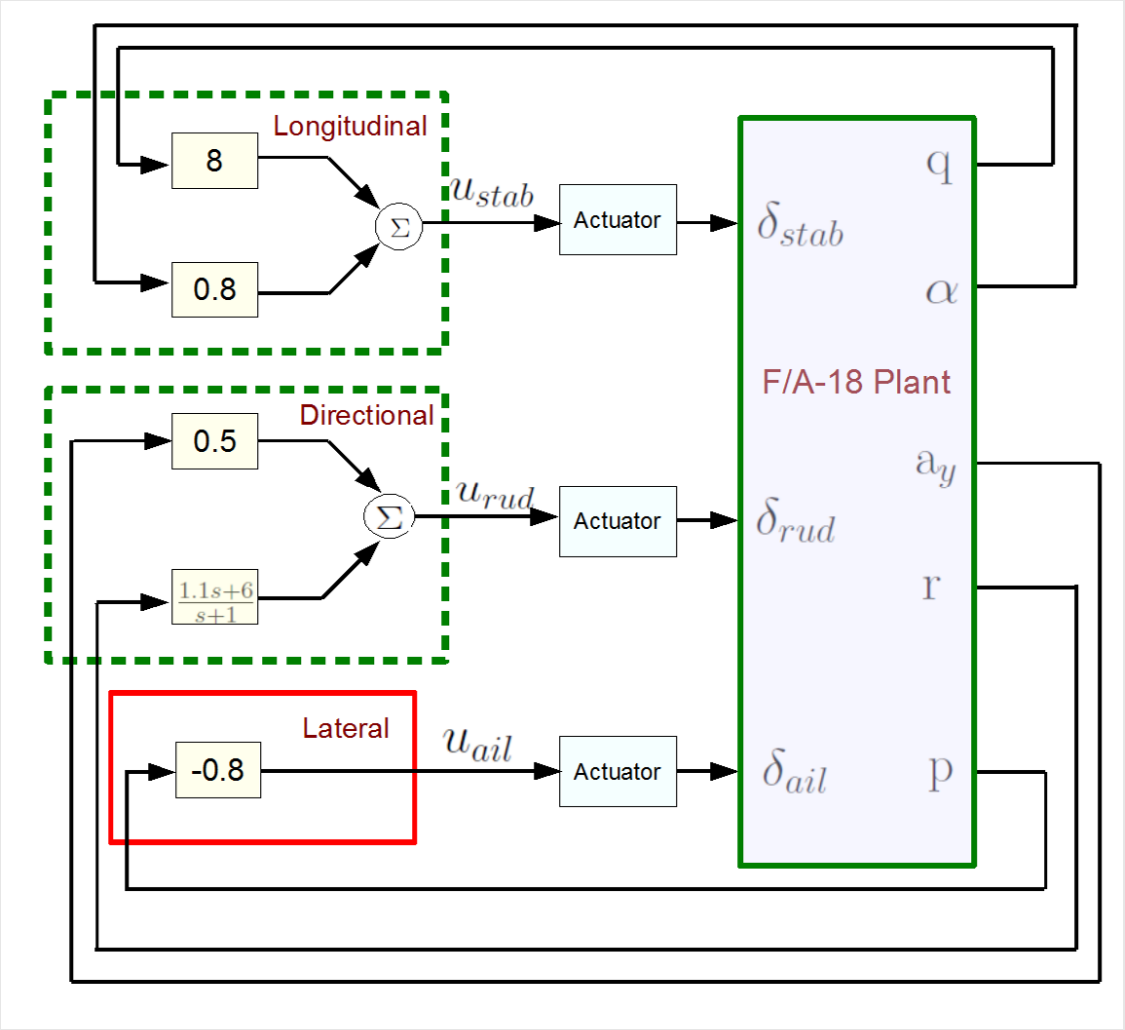


Control surfaces considered :

$$u = \begin{bmatrix} \text{aileron deflection}(\delta_{ail}) \\ \text{rudder deflection}(\delta_{rud}) \\ \text{stabilator deflection}(\delta_{stab}) \end{bmatrix}$$

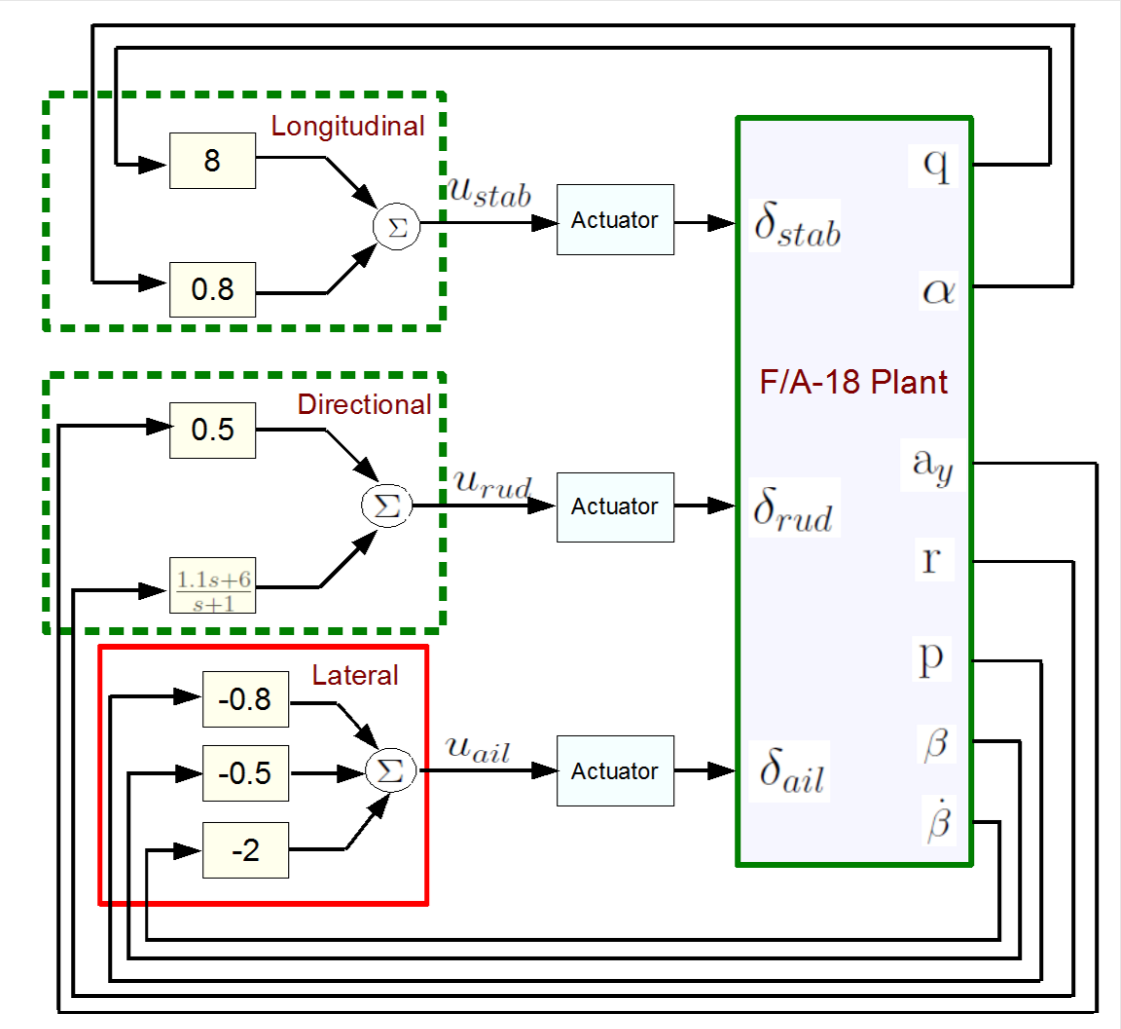
Reference: Iliff W. Kenneth, Wang C. Kon-Sheng, 'Flight Determined, Subsonic, Lateral-directional Stability and Control Derivatives of the Thrust-Vectoring F-18 high Angle of attack Research Vehicle (HARV), and Comparisons to the Basic F-18 and Predicted Derivatives', *NASA/TP-1999-206573*

Baseline Control Law Architecture



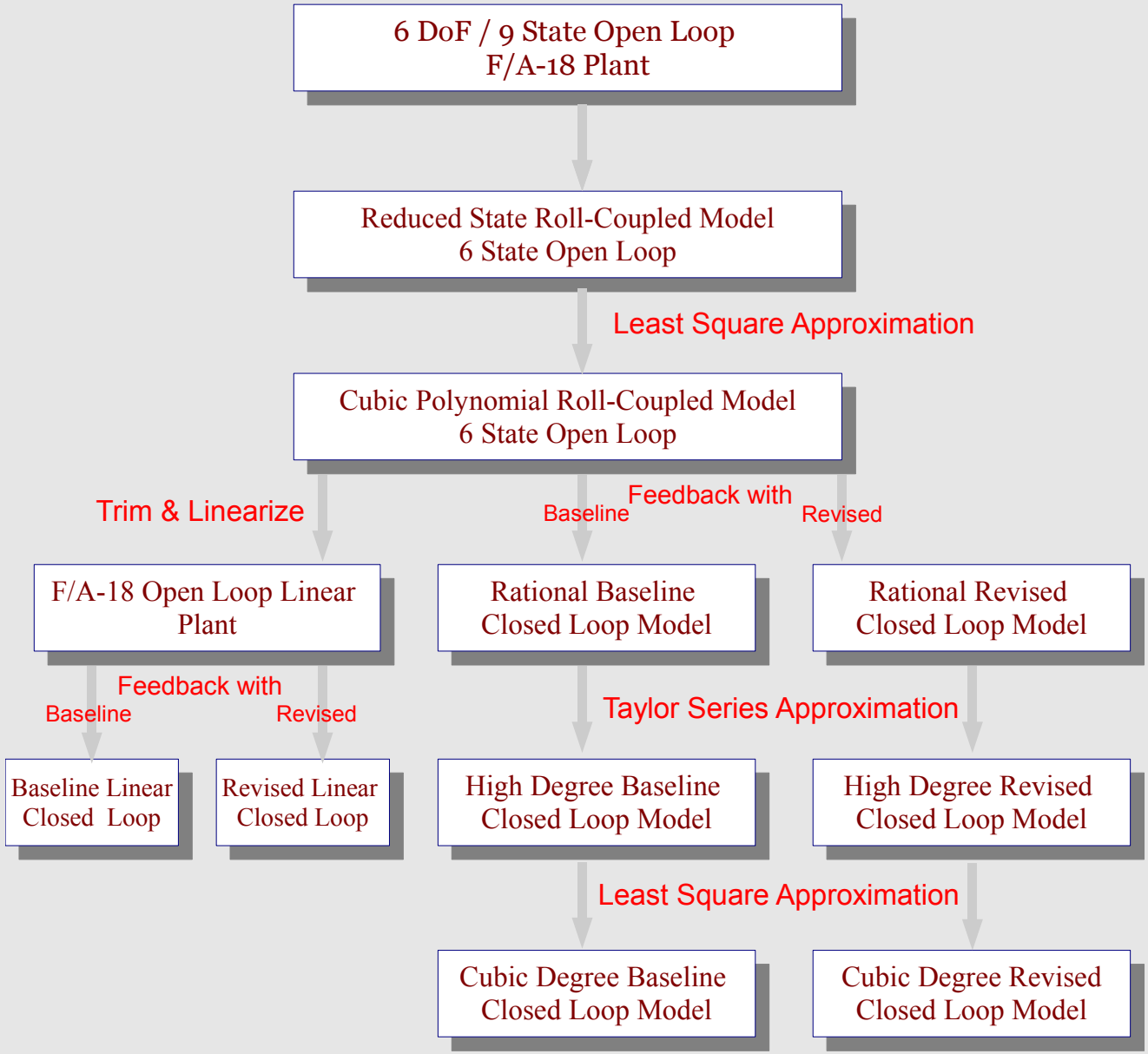
- F/A-18 is susceptible to falling leaf mode under baseline control law.
- The baseline control law is not able to damp out sideslip direction at high AoA (α).

Revised Control Law Architecture

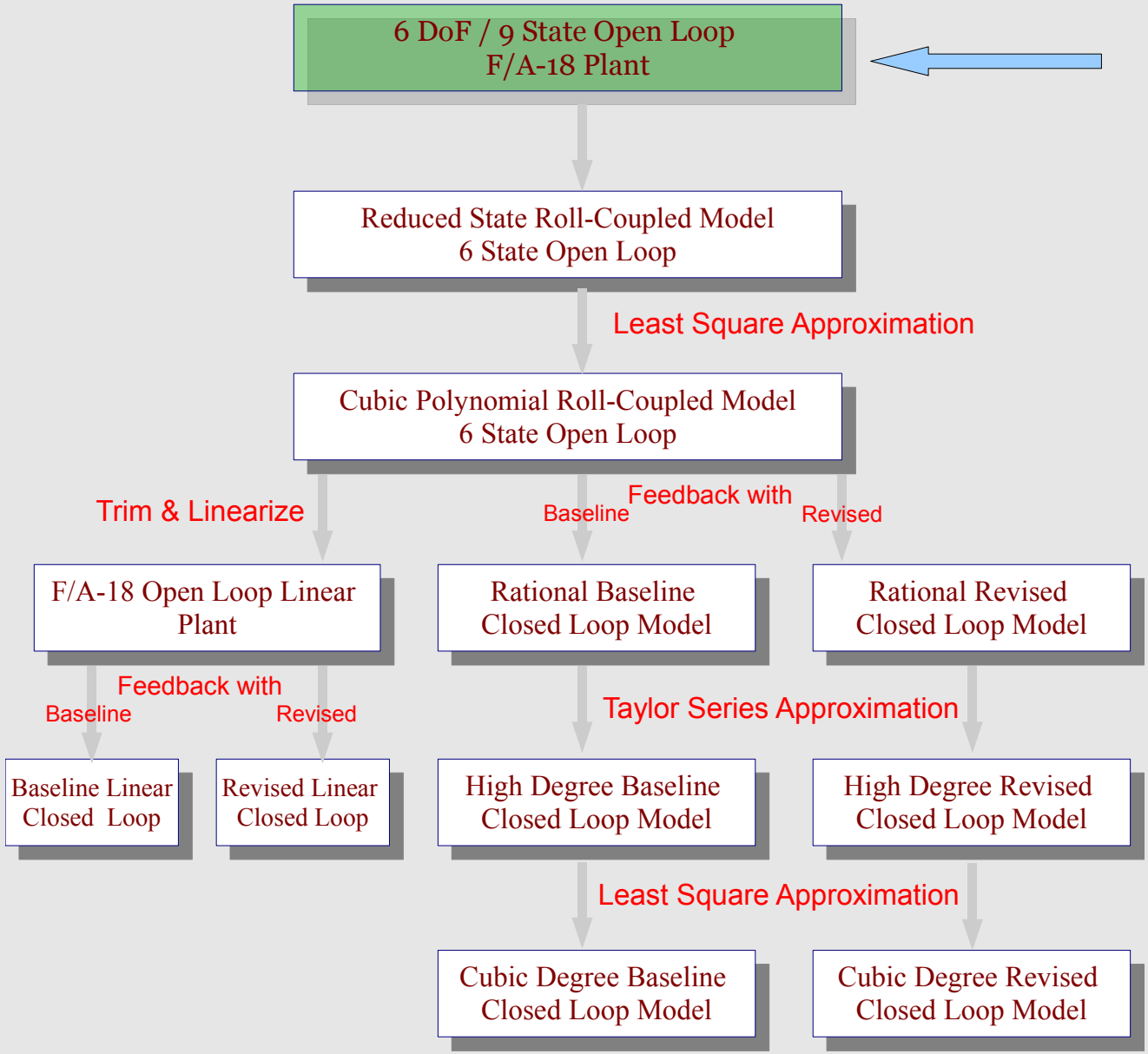


- Falling leaf mode is suppressed under revised control law.
- Sideslip (β) feedback improves sideslip damping at high AoA (α).
- Sideslip rate ($\dot{\beta}$) feedback improves lateral-directional damping.
- Sideslip rate ($\dot{\beta}$) feedback signal is computed from the kinematic equation.

Model Approximation



Model Approximation



Model Approximation: Six DoF 9-State Model

Force Equation:

$$\dot{V_{TAS}} = -\frac{\bar{q}S}{m}C_{D_{wind}} + g(\cos\phi\cos\theta\sin\alpha\cos\beta + \sin\phi\cos\theta\sin\beta) + g(-\sin\theta\cos\alpha\cos\beta) + \frac{T}{m}\cos\alpha\cos\beta$$

$$\dot{\alpha} = -\frac{\bar{q}S}{mV_{TAS}\cos\beta}C_L + q - \tan\beta(p\cos\alpha + r\sin\alpha) + \frac{g}{V_{TAS}\cos\beta}(\cos\phi\cos\theta\cos\alpha + \sin\alpha\sin\theta) - \frac{T\sin\alpha}{mV_{TAS}\cos\beta}$$

$$\dot{\beta} = -\frac{\bar{q}S}{mV_{TAS}}C_{Y_{wind}} + p\sin\alpha - r\cos\alpha + \frac{g}{V_{TAS}}\cos\beta\sin\phi\cos\theta + \frac{\sin\beta}{V_{TAS}}(g\cos\alpha\sin\theta - g\sin\alpha\cos\phi\cos\theta + \frac{T}{m}\cos\alpha)$$

The following state description have formulated the 6 DoF 9-state F/A-18 plant.

α : Angle-of-attack, rad
β : Sideslip Angle, rad
V_{TAS} : Velocity, ft/s
p : Roll rate, rad/s
q : Pitch rate, rad/s
r : Yaw rate, rad/s
ϕ : Bank angle, rad
θ : Pitch angle, rad
ψ : Yaw angle, rad

Model Approximation: Six DoF 9-State Model

Moment Equation:

$$\dot{p} = \frac{I_{zz}L + I_{xz}N - [I_{xz}(I_{yy} - I_{xx} - I_{zz})]pq - [I_{xz}^2 + I_{zz}(I_{zz} - I_{yy})]rq}{(I_{xx}I_{zz} - I_{xz}^2)}$$

$$\dot{q} = \frac{M - I_{xz}(p^2 - r^2) + (I_{zz} - I_{xx})pr}{I_{yy}}$$

$$\dot{r} = \frac{I_{xz}L + I_{xx}N + [I_{xz}(I_{yy} - I_{xx} - I_{zz})]rq + [I_{xz}^2 + I_{xx}(I_{xx} - I_{yy})]pq}{(I_{xx}I_{zz} - I_{xz}^2)}$$

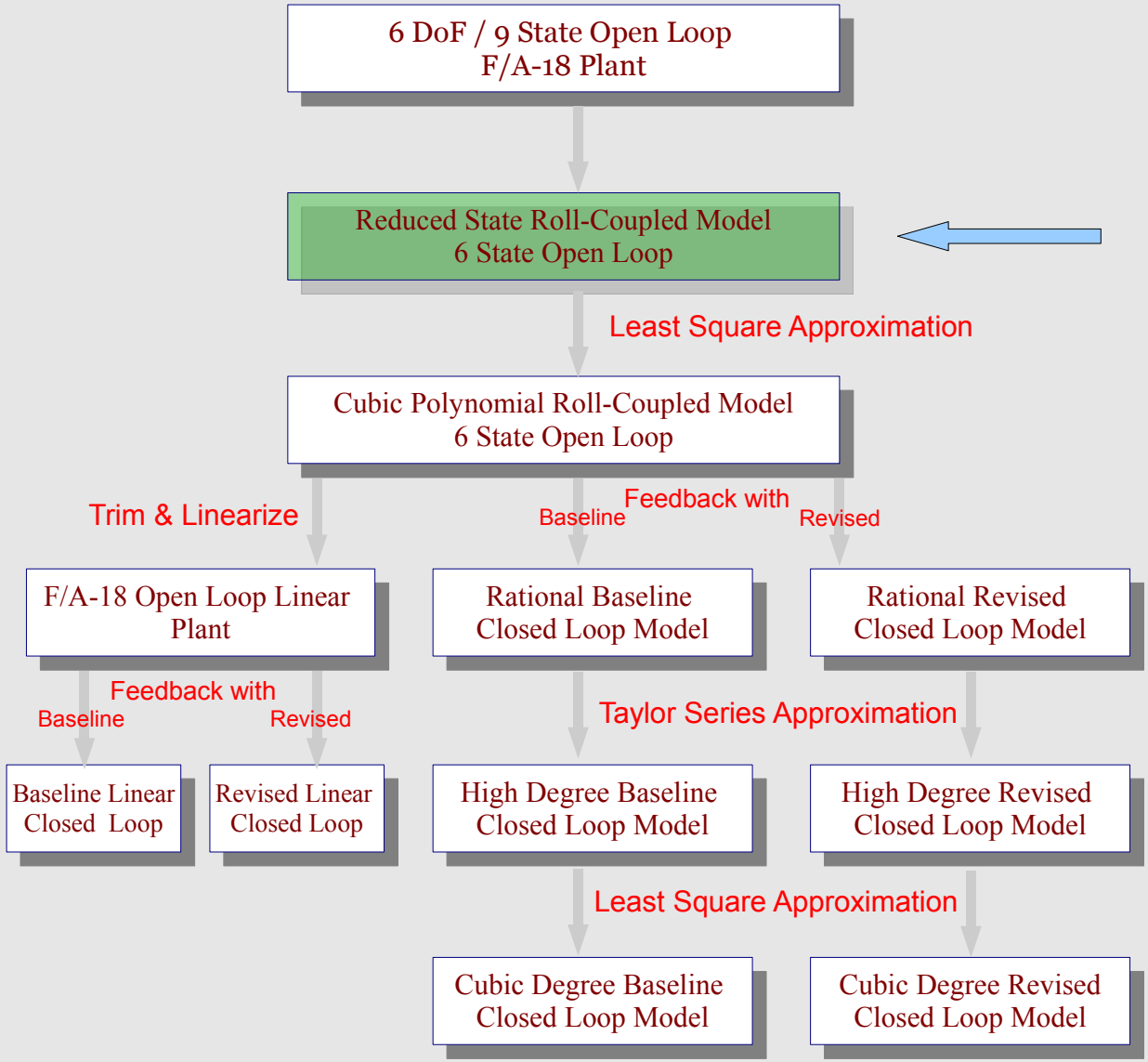
Kinematic Equation:

$$\dot{\phi} = p + (q\sin\phi + r\cos\phi)\tan\theta$$

$$\dot{\theta} = q\cos\phi - r\sin\phi$$

$$\dot{\psi} = (q\sin\phi + r\cos\phi)\sec\theta$$

Model Approximation: Reduced State Roll-Coupled Model



Model Approximation: Six DoF to Roll-Coupled Model

Approximation of Force Equation

Force Equation:

$$\cancel{\dot{V}_{TAS}}^{0} = -\frac{\bar{q}S}{m}C_{D_{wind}} + g(\cos\phi\cos\theta\sin\alpha\cos\beta + \sin\phi\cos\theta\sin\beta)$$
$$+g(-\sin\theta\cos\alpha\cos\beta) + \frac{T}{m}\cos\alpha\cos\beta$$

$$\dot{\alpha} = -\frac{\bar{q}S}{mV_{TAS}\cos\beta}C_L + q - \overbrace{\tan\beta(p\cos\alpha}^{\text{small angle approx}} \cancel{+r\sin\alpha)}^{0}$$

$$+\frac{g}{V_{TAS}\cos\beta}(\cos\phi\cos\theta\cos\alpha + \sin\alpha\sin\theta) - \frac{T\sin\alpha}{mV_{TAS}\cos\beta}$$

$$\dot{\beta} = -\frac{\bar{q}S}{mV_{TAS}}C_{Y_{wind}} + \overbrace{p\sin\alpha - r\cos\alpha + \frac{g}{V_{TAS}}\cos\beta\sin\phi\cos\theta}^{\text{small angle approx}}$$

$$+\frac{\sin\beta}{V_{TAS}}(g\cos\alpha\sin\theta - \cancel{g\sin\alpha}\cos\phi\cos\theta + \frac{T}{m}\cos\alpha)$$

Assumption:

- **Velocity has been assumed constant at 250 ft/s.**
- **Thrust effect in AoA (α) and Sideslip (β) are negligible.**
- **Small angle approximation has been made.**
- **Since roll-coupled model is being considered, we will ignore two other states: θ and ψ.**
- **Asymmetry in the aircraft does not influence inertial coupling in the moment equations.**

Model Approximation: Six DoF to Roll-Coupled Model

Approximation to Moment and Kinematic Equation

Moment Equation:

$$\dot{p} = \frac{I_{zz}L + I_{xz}N - \cancelto{0}{[I_{xz}(I_{yy} - I_{xx} - I_{zz})]pq} - [I_{xz}^2 + I_{zz}(I_{zz} - I_{yy})]rq}{(I_{xx}I_{zz} - I_{xz}^2)}$$

$$\dot{q} = \frac{M - \cancelto{0}{I_{xz}(p^2 - r^2)} + (I_{zz} - I_{xx})pr}{I_{yy}}$$

$$\dot{r} = \frac{I_{xz}L + I_{xx}N + \cancelto{0}{[I_{xz}(I_{yy} - I_{xx} - I_{zz})]rq} + [I_{xz}^2 + I_{xx}(I_{xx} - I_{yy})]pq}{(I_{xx}I_{zz} - I_{xz}^2)}$$

Kinematic Equation:

$$\dot{\phi} = p + \cancelto{0}{(q\sin\phi + r\cos\phi)\tan\theta}$$

$$\dot{\theta} = \cancelto{0}{q\cos\phi - r\sin\phi}$$

$$\dot{\psi} = \cancelto{0}{(q\sin\phi + r\cos\phi)}\sec\theta$$

Assumption:

- **Velocity has been assumed constant at 250 ft/s.**
- **Thrust effect in AoA (α) and Sideslip (β) are negligible.**
- **Small angle approximation has been made.**
- **Since roll-coupled model is being considered, we will ignore two other states: θ and ψ.**
- **Asymmetry in the aircraft does not influence inertial coupling in the Moment equations.**

Model Approximation: Reduced State Roll-Coupled Model

Force Equation:

$$\dot{\alpha} = -\frac{\bar{q}S}{mV_{TAS}} C_L + q - p\beta$$

$$\dot{\beta} = -\frac{\bar{q}S}{mV} C_{Y_{wind}} + p\alpha - r + \frac{g}{V_{TAS}}\phi$$

where

$$C_L = -C_X \sin\alpha + C_Z \cos\alpha$$
$$C_{Y_{wind}} = -C_Y \cos\beta + C_D \sin\beta$$

Moment Equation:

$$\dot{p} = \frac{I_{zz}L + I_{xz}N - [I_{xz}^2 + I_{zz}(I_{zz} - I_{yy})]rq}{(I_{xx}I_{zz} - I_{xz}^2)}$$

$$\dot{q} = \frac{M + (I_{zz} - I_{xx})pr}{I_{yy}}$$

$$\dot{r} = \frac{I_{xz}L + I_{xx}N + [I_{xz}^2 + I_{xx}(I_{xx} - I_{yy})]pq}{(I_{xx}I_{zz} - I_{xz}^2)}$$

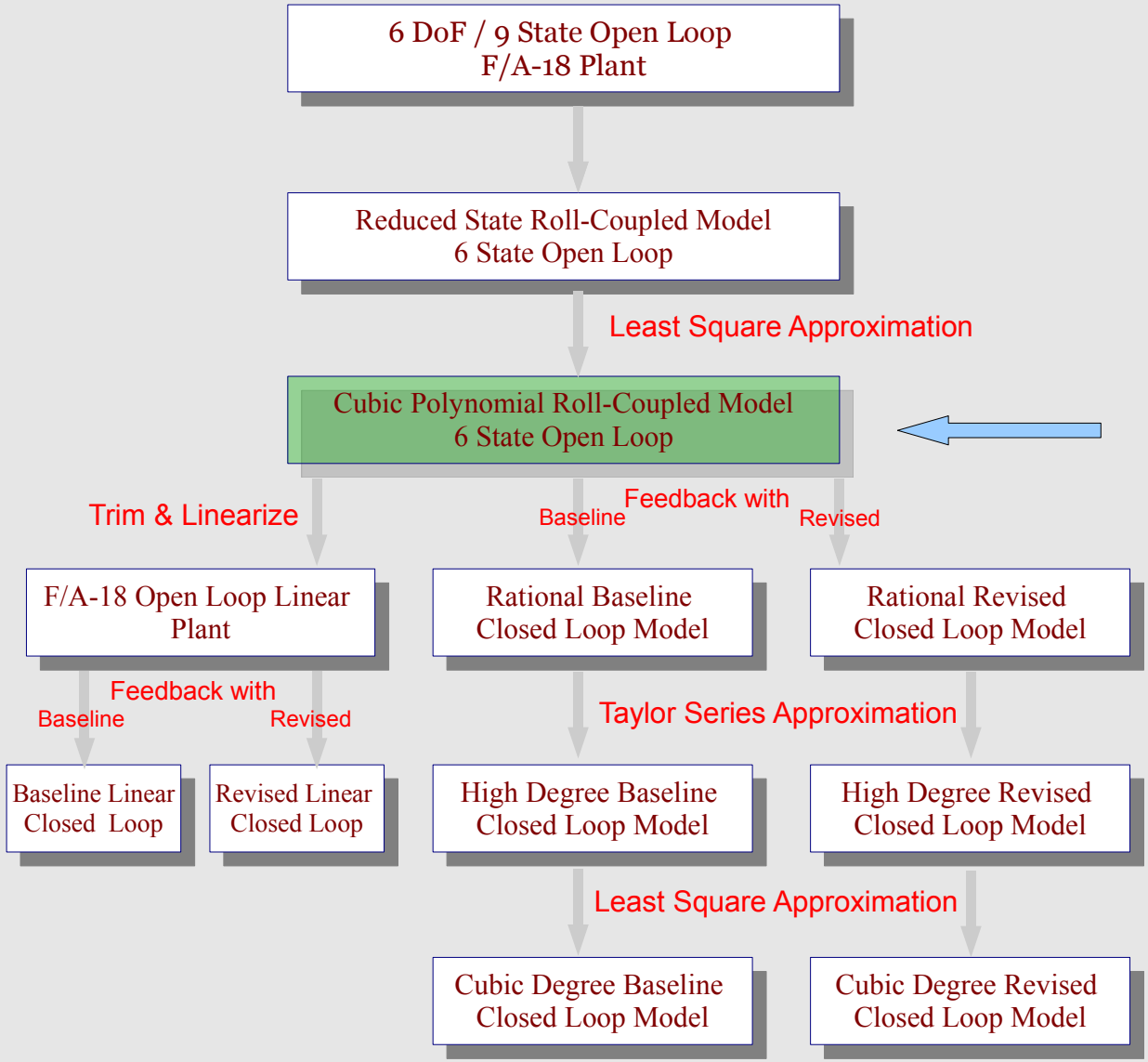
Kinematic Equation:

$$\dot{\phi} = p$$

The roll-coupled, reduced order model states are:

V_{TAS} : 250 ft/s (assumed constant)
α : Angle-of-attack, rad
β : Sideslip Angle, rad
p : Roll rate, rad/s
q : Pitch rate, rad/s
r : Yaw rate, rad/s
ϕ : Bank angle, rad

Model Approximation: Roll-Coupled to Cubic Polynomial



Model Approximation: Roll-Coupled to Cubic Polynomial

Aerodynamic terms approximation:

- $C_{Y_{wind}}$ is approximated as:

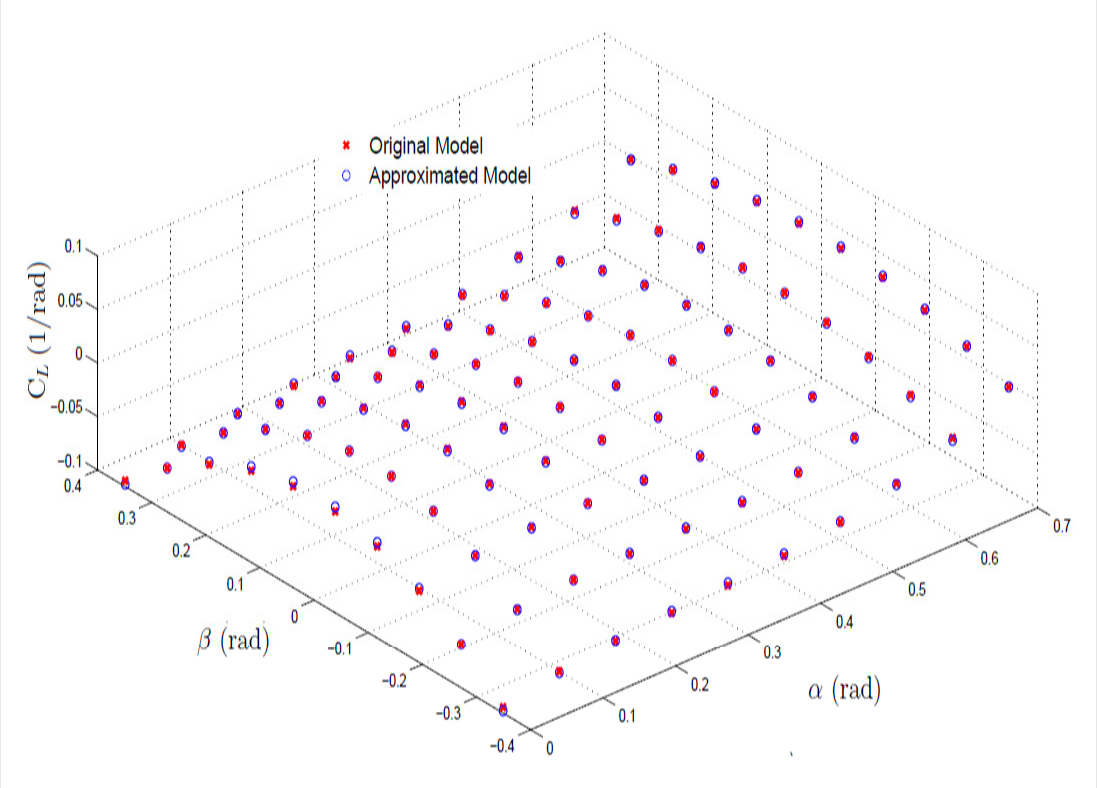
$$\begin{aligned} C_{Y_{wind}} &= -C_Y \cos\beta + C_D \sin\beta \\ &= -C_Y + C_D\beta \end{aligned}$$

- $C_Z,\ C_X,\ C_Y,\ C_D$ are aerodynamic coefficients which are polynomial function of $\alpha,\ \beta$.

$$\begin{aligned} C_X &= f(\alpha^2, \alpha, \beta) \\ C_Z &= f(\alpha^2, \alpha) \\ C_Y &= f(\alpha^3, \alpha^2, \alpha, \beta) \\ C_D &= f(\alpha^2, \alpha, \beta^2) \end{aligned}$$

- $L,\ M,\ N$ are aerodynamic moments represented as $f(\alpha^2, \alpha, \beta)$.
- Now only C_L needs to be approximated.

Model Approximation: Roll-Coupled to Cubic Polynomial

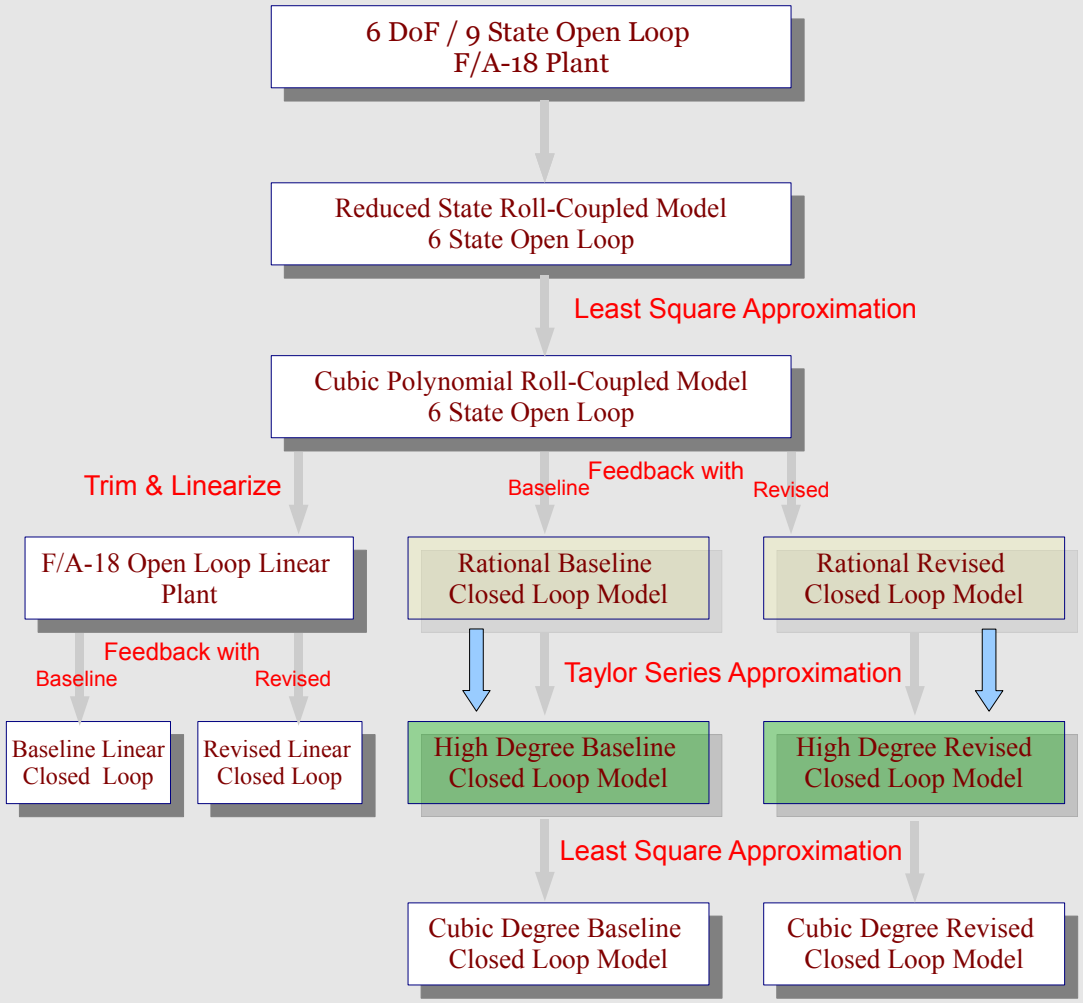


$$C_L = -C_Z \cos\alpha + C_X \sin\alpha$$

- C_L is evaluated on the grid $-20^\circ \leq \beta \leq 20^\circ$, and $0^\circ \leq \alpha \leq 40^\circ$.
- C_L is fit by a polynomial function of $\alpha - \beta$ up to cubic degree.

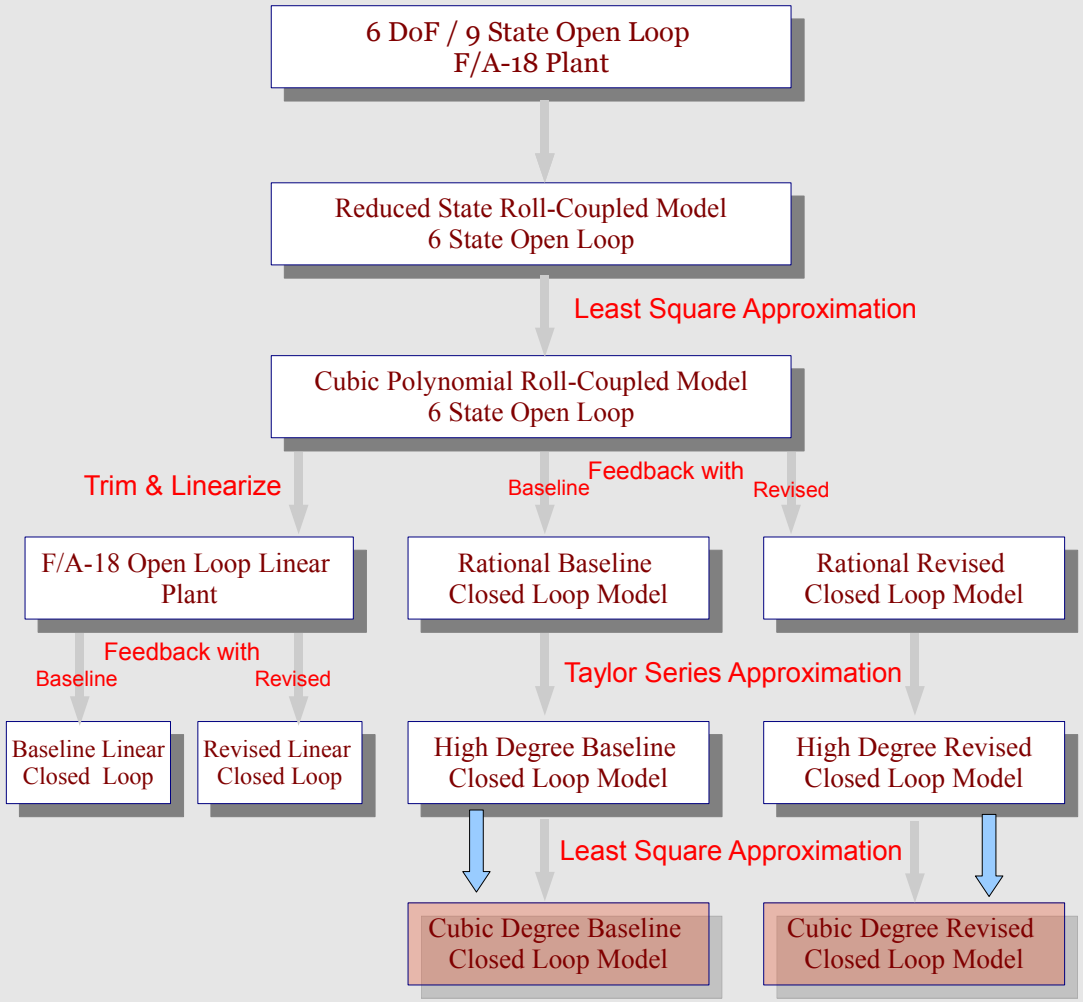
Result is a 6-state roll-coupled, cubic order *polynomial* model

Model Approximation: Cubic Degree Closed Loop Model



- Implementation of flight controller(s) with cubic polynomial model results in 4^{th} order, rational polynomial model.
- Rational terms are due to the 'D' matrix in the controller realization.
- 1^{st} order Taylor series approximation is used to handle rational terms.

Model Approximation: Cubic Degree Closed Loop Model



- Need to approximate resulting 4^{th} order F-18 closed-loop polynomial model by to 3^{rd} order polynomial model.
- Most of the nonlinearities occur as a function of $\alpha - \beta$.
- 4^{th} order polynomial model is approximated using least square technique on the $\alpha - \beta$ grid.

Modeling Summary

- The reduced order, nonlinear polynomial model captures the characteristics of the falling leaf motion.
- For analysis purpose, roll-coupled maneuvers are consider which drive the aircraft to the falling leaf motion.
- The velocity is assumed to be fixed at 250 ft/s.

$$\dot{x} = f(x, u) \ , \ \ y = h(x)$$

$$x = \begin{bmatrix} \text{angle-of-attack}(\alpha) \\ \text{sideslip angle}(\beta) \\ \text{roll rate}(p) \\ \text{yaw rate}(r) \\ \text{pitch rate}(q) \\ \text{bank angle}(\phi) \end{bmatrix} \ , \ \ y = \begin{bmatrix} \text{angle-of-attack}(\alpha) \\ \text{roll rate}(p) \\ \text{yaw rate}(r) \\ \text{pitch rate}(q) \\ \text{lateral acceleration}(a_y) \\ \text{sideslip rate}(\dot{\beta}) \\ \text{sideslip angle}(\beta) \end{bmatrix}$$

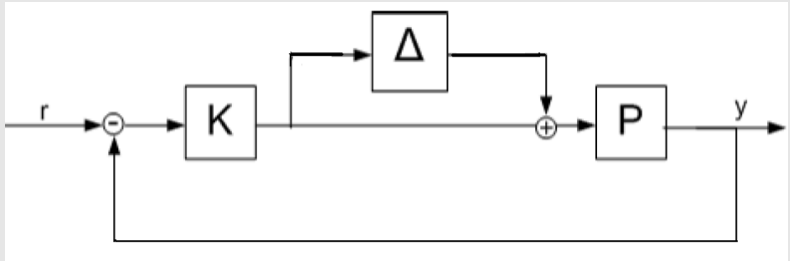
$$u = \begin{bmatrix} \text{aileron deflection}(\delta_{ail}) \\ \text{rudder deflection}(\delta_{rud}) \\ \text{stabilator deflection}(\delta_{stab}) \end{bmatrix}$$

Linear Analysis

- F/A-18 aircraft is trimmed at selected equilibrium points.
- Classical Loop-at-a-time Margin Analysis
- Disk Margin Analysis
- Multivariable Input Margin Analysis
- Diagonal Input Multiplicative Uncertainty Analysis
- Full Block Input Multiplicative Uncertainty Analysis
- Robustness Analysis with Uncertainty in Aerodynamic Coefficients & Stability Derivatives

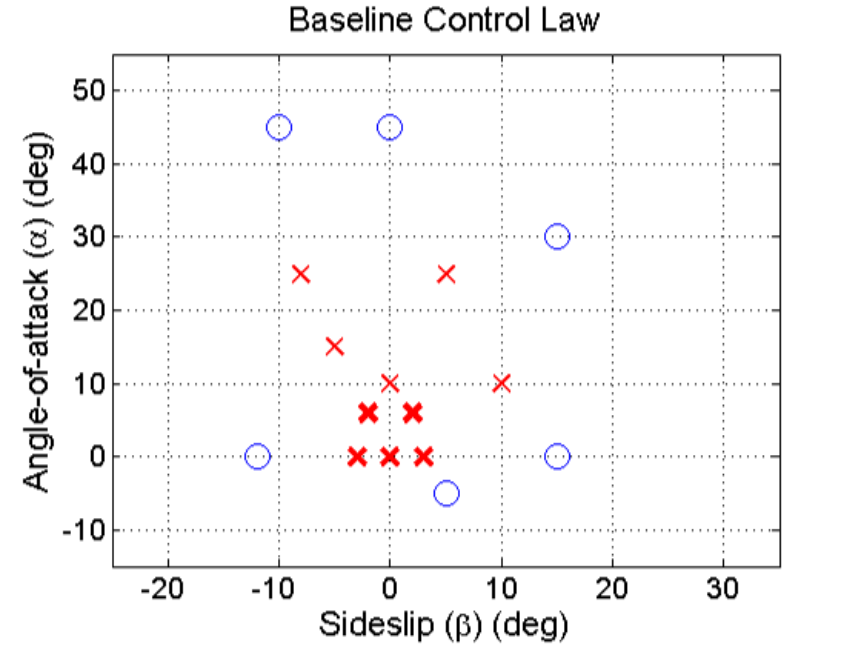
The linearized aircraft models do not exhibit the falling leaf mode characteristics.

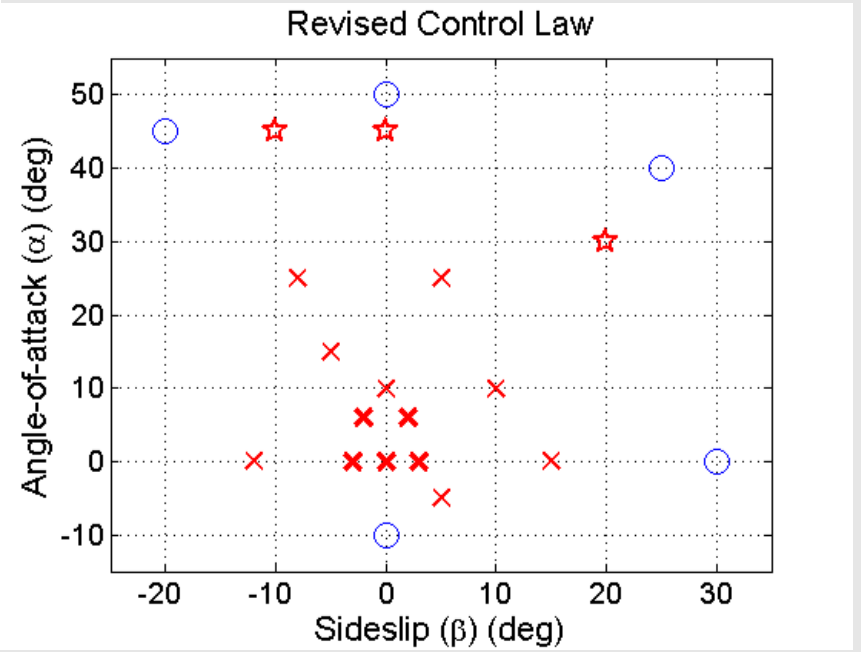
Linear Analysis (cont'd)



- Input Multiplicative uncertainty is introduced to account for unmodeled dynamics and model uncertainty at the plant input.
- Falling leaf mode is attributed to the nonlinear interaction of the aircraft dynamics and aerodynamic forces/moments. Hence, uncertainty is introduced into the aerodynamic and stability derivatives of the aircraft.

Linear Analysis: Linear Model Formulation





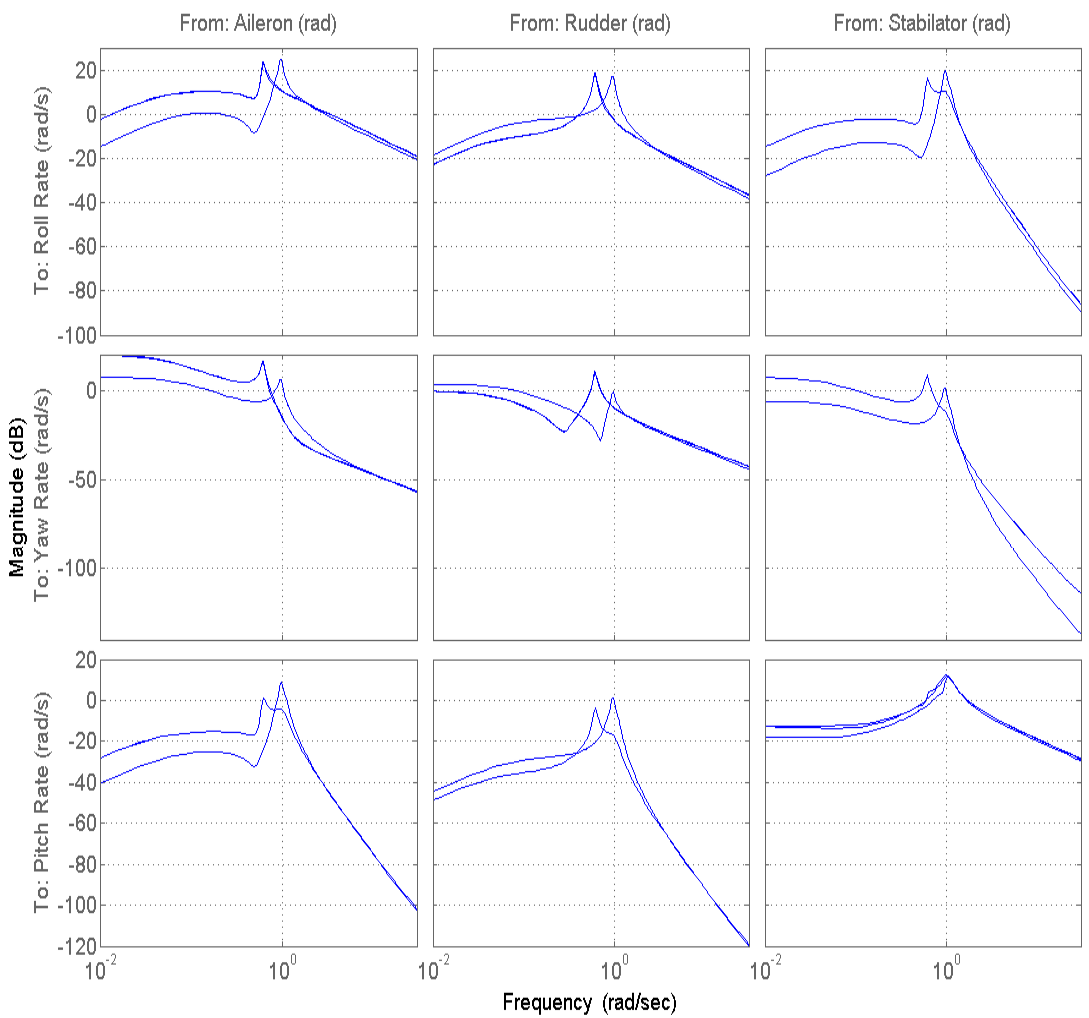
'x' indicates Stable Plant 'O' indicates Unstable Plant '⋆' Insufficient Control Authority

Equilibrium flight conditions:

- Altitude = 25,000 feet.
- V_{TAS} = 250 ft/s.
- Linear analysis is performed on the Bold Faced 'x' marked plants.

Linear Analysis (cont'd)

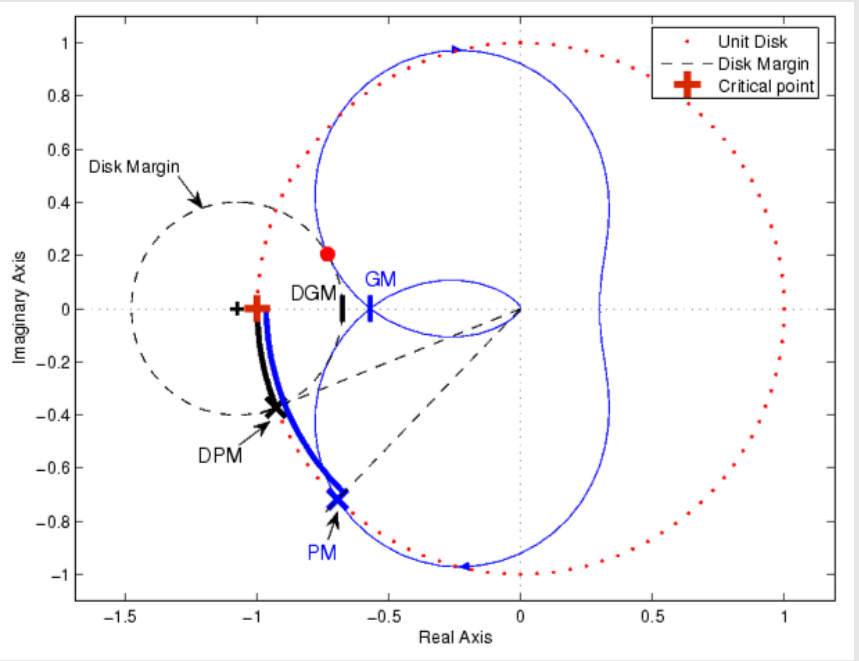
Open Loop Bode Plot: Input Channels to Rates



- Coupling in all channels play an important role in initiating Falling Leaf motion.
- There is coupling presents in all three channels.

Linear Analysis (cont'd)

- **Classical Loop-at-a-time Margin Analysis**
- **Disk Margin Analysis**
- **Multivariable Input Margin Analysis**
- Diagonal Input Multiplicative Uncertainty Analysis
- Full Block Input Multiplicative Uncertainty Analysis
- Robustness Analysis with Uncertainty in Aerodynamic Coefficients & Stability Derivatives



Linear Analysis (cont'd)

For classical margin analysis we consider the linearized plant at 25,000 feet with $\alpha = 0^o$, $\beta = -3^o$

Classical Loop-at-a-time Margin Analysis

Input Channel		**Baseline**	**Revised**
Aileron	*Gain Margin*	∞	38 dB
	Phase Margin	-85^o	94^o
	Delay Margin	2.22 sec	0.61 sec
Rudder	*Gain Margin*	24 dB	24 dB
	Phase Margin	55^o	60^o
	Delay Margin	0.82 sec	0.92 sec
Stabilator	*Gain Margin*	∞	∞
	Phase Margin	90^o	87^o
	Delay Margin	0.17 sec	0.47 sec

State	Trimmed Value
V_{TAS}	250 ft/s
Sideslip Angle, β	-3°
Roll Rate, p	0°/s
Yaw Rate, r	0.0747°/s
Bank Angle, ϕ	0°
Angle-of-Attack, α	0°
Pitch Rate, q	0°/s
Input	Trimmed Value
δ_{Stab}	0°
δ_{Ail}	-0.35°
δ_{Rud}	-3.76°

Trim Values for the Flight Condition

Linear Analysis (cont'd)

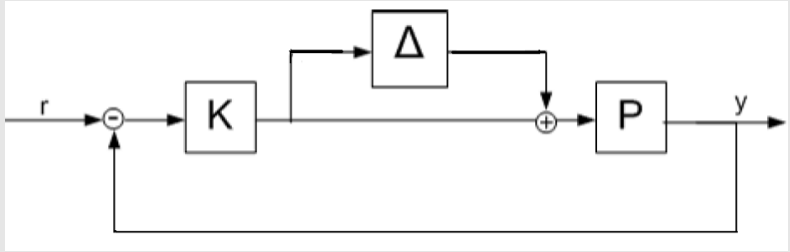
The same linearized plant at 25,000 feet, $\alpha = 0°$, $\beta = -3°$, is used for the disk margin analysis.

Disk Margin Analysis

Input Channel		**Baseline**	**Revised**
Aileron	*Gain Margin*	± 26dB	± 39dB
	Phase Margin	$\pm 83^o$	$\pm 88^o$
Rudder	*Gain Margin*	± 9dB	± 10dB
	Phase Margin	$\pm 52^o$	$\pm 55^o$
Stabilator	*Gain Margin*	∞	± 30dB
	Phase Margin	$\pm 90^o$	$\pm 86^o$

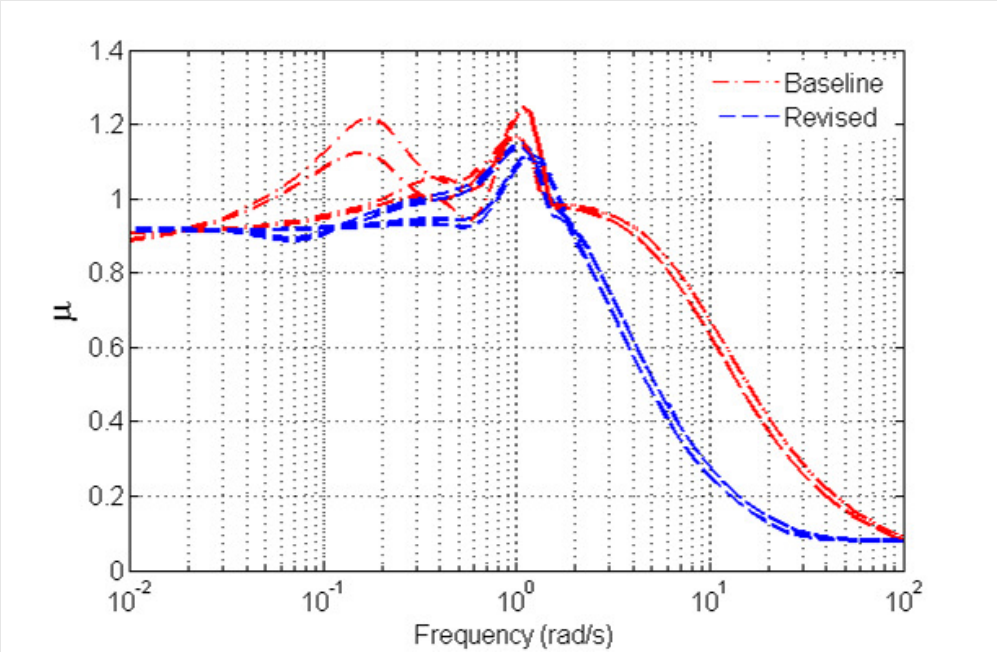
Linear Analysis (cont'd)

- Classical Loop-at-a-time Margin Analysis
- Disk Margin Analysis
- Multivariable Input Margin Analysis
- **Diagonal Input Multiplicative Uncertainty Analysis**
- **Full Block Input Multiplicative Uncertainty Analysis**
- Robustness Analysis with Uncertainty in Aerodynamic Coefficients & Stability Derivatives



Linear Analysis (cont'd)

Diagonal Input Multiplicative Uncertainty

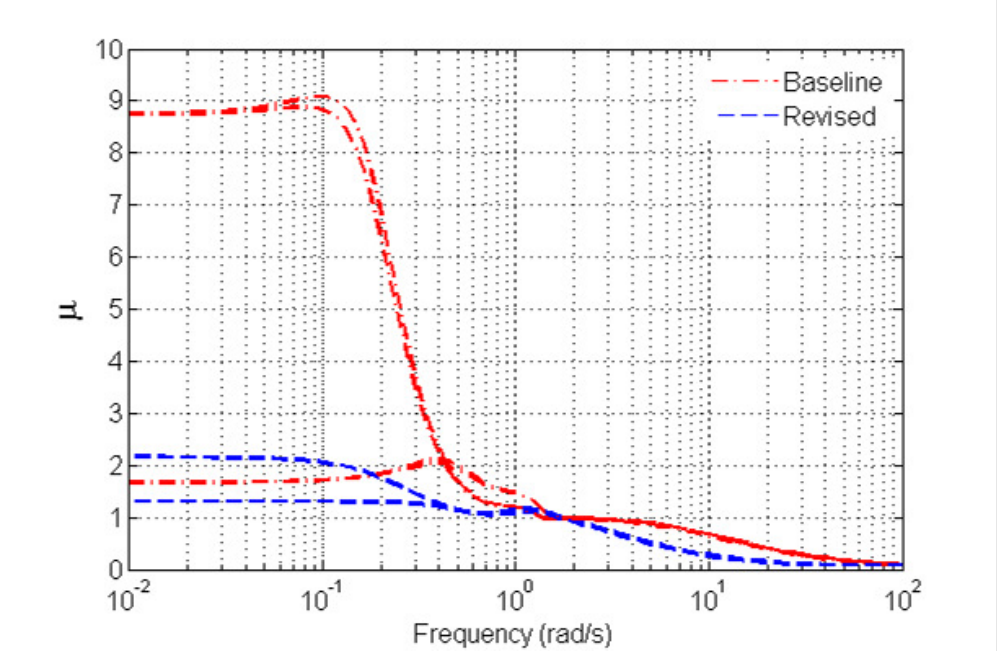


Stability Margin (k_m) can be defined as the inverse of μ : $k_m = \frac{1}{\mu}$.

- Diagonal uncertainty structure models no uncertain cross-coupling in actuation channels.
- Revised controller performs slightly better than the baseline controller.
- Results are similar to classical margin analysis.

Linear Analysis (cont'd)

Full Block Input Multiplicative Uncertainty



- Full-block uncertainty structure models potential cross-coupling between actuation channels due to uncertainty.
- Revised controller performs better than the baseline controller.
- Revised control law is better able to handle cross-coupling in actuation channels.

Linear Analysis (cont'd)

- Classical Loop-at-a-time Margin Analysis
- Disk Margin Analysis
- Multivariable Input Margin Analysis
- Diagonal Input Multiplicative Uncertainty Analysis
- Full Block Input Multiplicative Uncertainty Analysis
- **Robustness Analysis with Uncertainty in Aerodynamic Coefficients & Stability Derivatives**

Linear Analysis (cont'd)

Robustness to Uncertainty in Aerodynamic Coefficients & Stability Derivatives

- Indicated terms (below) in the linearized open-loop 'A' matrix represent important aerodynamic coefficients and stability derivatives.

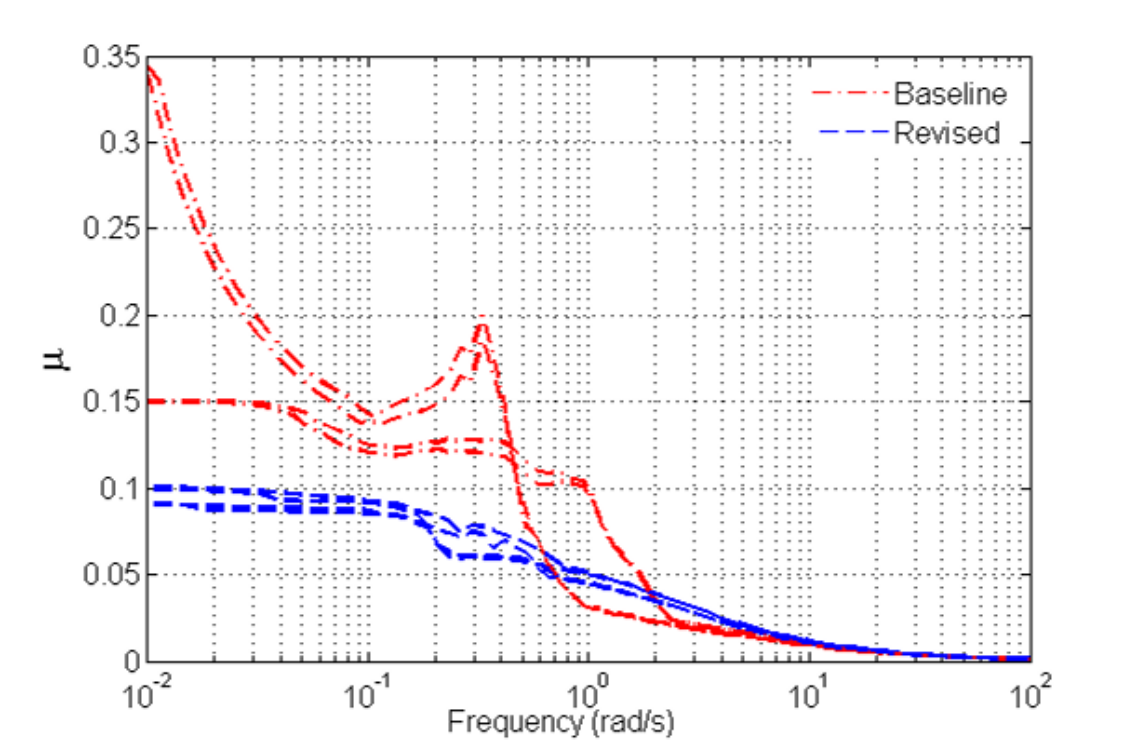
$$\begin{bmatrix} \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \\ \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} A_{11} & & & & & \\ A_{21} & A_{22} & & & & \\ A_{31} & & A_{33} & & & \\ & & & & A_{56} & \\ & & & A_{65} & A_{66} & \end{bmatrix} \begin{bmatrix} \beta \\ p \\ r \\ \phi \\ \alpha \\ q \end{bmatrix} + Bu$$

A_{11} : side force due to sideslip (Y_β)
A_{21} : rolling moment due to sideslip (L_β)
A_{22} : roll damping (L_p)
A_{31} : yawing moment due to sideslip (N_β)
A_{33} : yaw damping (N_r)
A_{56} : normal force due to pitch rate (Z_q)
A_{65} : pitch stiffness (M_α)
A_{66} : pitch damping (M_q)

- $\pm 10\%$ real parametric uncertainty is introduced in selected terms.
- The selected terms play an important role in the falling leaf motion.

Linear Analysis (cont'd)

Robustness to Uncertainty in Aerodynamic Coefficients & Stability Derivatives



- Revised controller is less sensitive to aerodynamic uncertainty than the baseline controller.

Linear Analysis: Summary

Linear Analysis	**Baseline**	**Revised**
Multivariable Input Loop Phase Margin	$\pm 45.0^{\circ}$	$\pm 53^{\circ}$
Multivariable Input Loop Gain Margin (dB)	± 7.7	± 9.6
Diagonal Input Multiplicative: $(k_m = \frac{1}{\mu})$	0.80	0.87
Full Input Multiplicative: $(k_m = \frac{1}{\mu})$	0.11	0.47
Parametric Uncertainty: $(k_m = \frac{1}{\mu})$	2.85	10.0

Results based on the linearized plant at 25,000 feet with $\alpha = 0^o$, $\beta = -3^o$.

Nonlinear Region-of-Attraction (ROA) Analysis

Motivation:

- Falling leaf motion is a nonlinear phenomenon which is not captured by linear analysis around equilibrium points.
- Linear analysis investigates robustness in a region around a stable equilibrium point.
  - Linear analysis indicates the revised controller is more robust than the baseline controller.
  - However, linear analysis does not provide any quantitative guarantee on the stable region of flight that each controller provides.
- Nonlinear analysis techniques can be used to certify regions of stability for individual controllers
  - Region-of-Attraction vs. Stability of equilibrium points.

Nonlinear Region-of-Attraction Analysis (cont'd)

Definition: Region-of-Attraction (ROA)

The Region-of-Attraction (ROA) of an asymptotically stable equilibrium point provides the set of the initial conditions whose state trajectories converge to the equilibrium point.

Consider:

$$\dot{x} = f(x),\ x(0) = x_0$$

$$R_0 = \left\{ x_0 \in \mathbb{R}^n :\ \text{If } x(0) = x_0 \text{ then } \lim_{t \to \infty} x(t) = 0 \right\}$$

Nonlinear ROA Analysis (contd.)

Why ROA Analysis?

- ROA measures the size of the set of initial conditions which will converge back to an equilibrium point.
- ROA analysis around a trim point provides a guaranteed stability region for the aircraft.
- Provides a good metric for detecting susceptible region to departure phenomenon like falling leaf motion.
- Linear analysis of equilibrium points do not provide any measure to the susceptible region to departure phenomenon.

Hence, ROA analysis is performed on both the baseline and revised control laws to compare the susceptibility to the falling leaf motion.

Nonlinear Region-of-Attraction Analysis (cont'd)

Estimating Invariant Subsets of ROA

- We will use the **Cubic Degree Baseline / Revised Closed Loop Model** as described in **Model Approximation** section.
- We will restrict in searching for the maximum ellipsoidal approximations of the ROA.

$$\beta^* = \max \beta$$
$$\text{subject to: } \{x_0^T N x_0 \leq \beta\} \subset R_0$$

- Now we will attempt to solve the upper and lower bound:

$$\underline{\beta} \leq \beta^* \leq \bar{\beta}$$

Nonlinear Region-of-Attraction Analysis (cont'd)

Choosing the Shape Factor of the Ellipsoid

- $N = N^T > 0$ is user-specified diagonal matrix which determines the shape of the ellipsoid.
- Here, we have chosen N such that the shape factor is normalized by the inverse of the maximum value each state can achieve.

```
% Create Shape Function

d2r = pi/180;
Dmax = diag([5*d2r 20*d2r 5*d2r 45*d2r 25*d2r 25*d2r 25*d2r]);
N = inv(Dmax^2);   %  Scale by inverse of max state values
N = N/max(N(:));   %  Normalize by the largest entry
p = x'*N*x;
```

- Both the closed loop models have 7-states *(ordered accordingly to Dmax)*:

β : Sideslip Angle, rad
p : Roll rate, rad/s
r : Yaw rate, rad/s
ϕ : Bank angle, rad
α : Angle-of-attack, rad
q : Pitch rate, rad/s
x_c : Controller State

Nonlinear Region-of-Attraction Analysis (cont'd)

Estimating Lower Bound on ROA

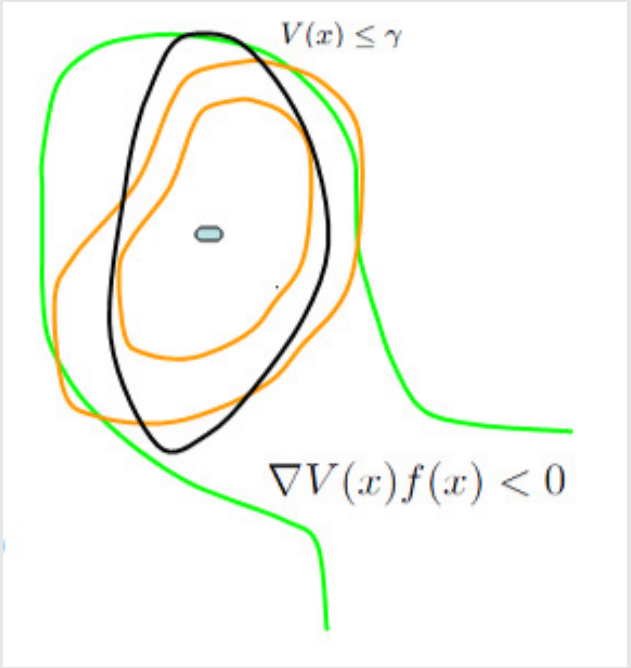
The lower bound estimation problem can be formulated as:

$$\underline{\beta} := \max \beta$$

$$\text{subject to: } \{x_0^T N x_0 \leq \beta\} \subset \Omega_\gamma$$

where: $V : \mathbb{R}^n \to \mathbb{R}$ is the Lyapunov function such that:

- $V(0) = 0$ and $V(x) > 0$ for all $x \neq 0$
- $\Omega_\gamma := \{x \in \mathbb{R}^n \ : \ V(x) \leq \gamma\}$ is bounded.
- $\Omega_\gamma \subset \{x \in \mathbb{R}^n \ : \ \nabla V(x) f(x) < 0\}$



Nonlinear Region-of-Attraction Analysis (cont'd)

Estimating Lower Bound on ROA

- Simulation data is *not* used in the analysis due to memory issue.
- **V-s iteration** procedure is used to estimate the lower Bound on ROA.
- Lyapunov function is initialized using the **linearized Lyapunov function**.
- **Quartic ($\partial V = 4$) Lyapunov** function is used to best approximate the lower bound.

Nonlinear Region-of-Attraction Analysis (cont'd)

F/A-18 V-s Iteration Step: Initialization

- Load F/A-18 model
- Initialize Lyapunov function with Linearized $V(x)$

```
% Linearized Lyapunov function
Vlin=linstab(f,x);
```

- Initialize multiplier and Lyapunov degree function:

```
% Create multiplier and function to force dV/dt<0
z2 =  monomials(x, 1);
L2 =  1e-6*(x'*x);
L1 =  1e-6*(x'*x);
z1 =  monomials(x,0:1);
Vdeg = 4;
zV = monomials(x,2:Vdeg);
```

- Create shape function for F/A-18.

```
% Create Shape Function

d2r = pi/180;
Dmax = diag([5*d2r 20*d2r 5*d2r 45*d2r 25*d2r 25*d2r 25*d2r]);
N = inv(Dmax^2);   %  Scale by inverse of max state values
N = N/max(N(:));   %  Normalize by the largest entry
p = x'*N*x;
```

Nonlinear Region-of-Attraction Analysis (cont'd)

F/A-18 V-s Iteration Step: Iteration

1. γ **Step**: Hold $V(x)$ fixed and solve for s_2

$$\max_{s_1 \in \text{SOS},\gamma} \gamma \quad \text{s.t. } -\left(\frac{\partial V}{\partial x} f + l_2 + s_2(\gamma - V)\right) \in \text{SOS}$$

2. β **Step**: Hold $V(x)$ fixed and solve for s_1

$$\max_{s_1 \in \text{SOS},\beta} \beta \quad \text{s.t. } -((V - \gamma) + s_1(\beta - p)) \in \text{SOS}$$

3. V **step**: Hold s_1, s_2, β, γ fixed and solve for V satisfying:

$$-\left(\frac{\partial V}{\partial x} f + l_2 + s_2(\gamma - V)\right) \in \text{SOS}$$
$$-((V - \gamma) + s_1(\beta - p)) \in \text{SOS}$$
$$V - l_1 \in \text{SOS}, V(0) = 0$$

4. Repeat until maximum number of iterations have been reached.

Nonlinear Region-of-Attraction Analysis (cont'd)

Estimating Upper Bound on ROA: Monte Carlo Simulation

- Randomly choose initial condition on the boundary of a large ellipsoid and simulate the system.

```
% Code from Search4UnstableTraj.m

gtry = 0.01;            % Size of the Initial Ellipsoid
gfac = 0.995;           % Decreasing Factor for the Ellipsoid

% Find random initial condition and scale so that x0'*N*x0 = gtry
x0 = randn(ns,1);
x0(end) = 0;   % Special for the A/C example--set controller state to zero
scl = sqrt((x0'*N*x0)/gtry );
x0 = x0/scl;

% Simulate and check for convergence
[xtraj,xconv]= psim(f,x,x0,Tfinal);
div = (xconv==0)
```

- If a divergent trajectory is found, decrease the ellipsoid by a factor of 0.995.

```
if div   % Decrease the ellipsoid shape since divergent trajectories found
    gtry = gfac*gtry;
end
```

- Repeat until maximum number of simulation has been reached.

Nonlinear Region-of-Attraction Analysis (cont'd)

Computational Aspects

- Computational time for estimating both lower and upper bound are as follows:

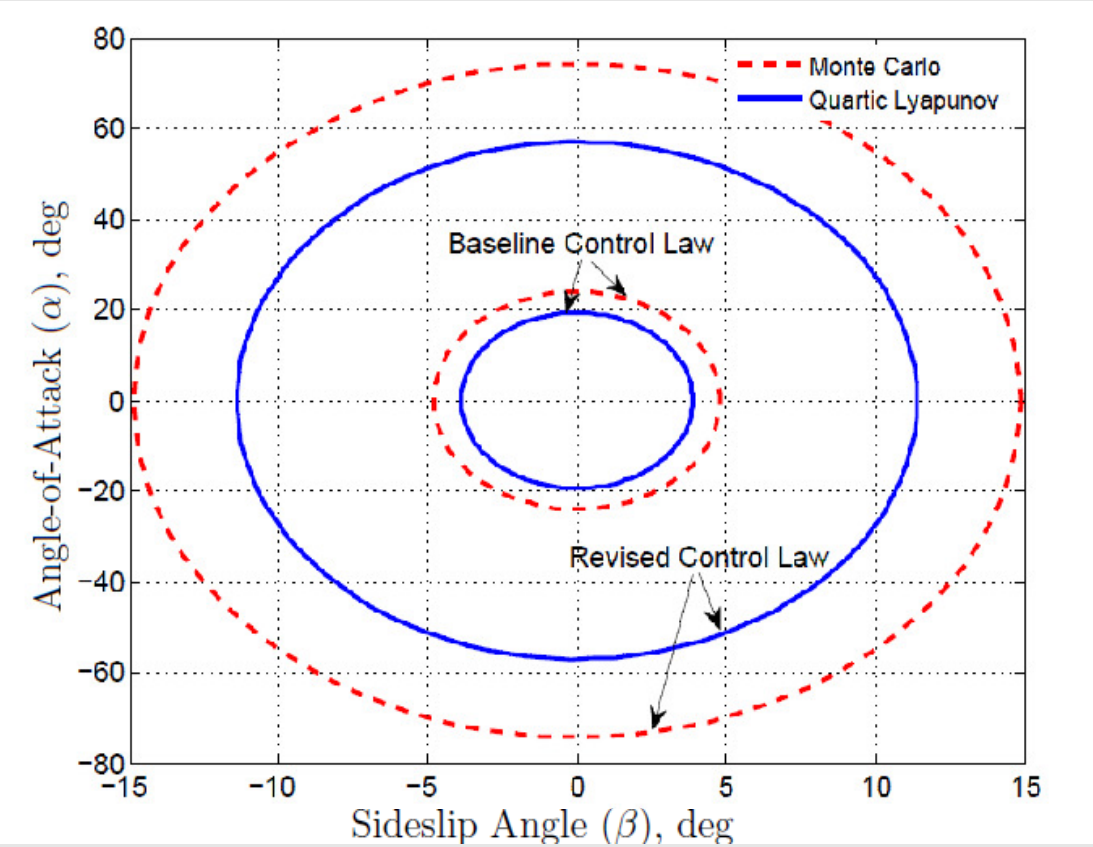
Analysis	Iteration Steps	**Baseline**	**Revised**
V-s Iteration$^{(1)}$	40	6.8 Hrs	4.7 Hrs
Monte Carlo Upper Bound$^{(2)}$	5 million	96 Hrs	96 Hrs

(1) V-s iteration analysis performed on *Intel(R) Core(TM) i7 CPU 2.67GHz 8.00GB RAM*

(2) Monte Carlo analysis performed on *Intel(R) Core(TM)2 Duo CPU E65550 2.33GHz 3.00GB RAM*

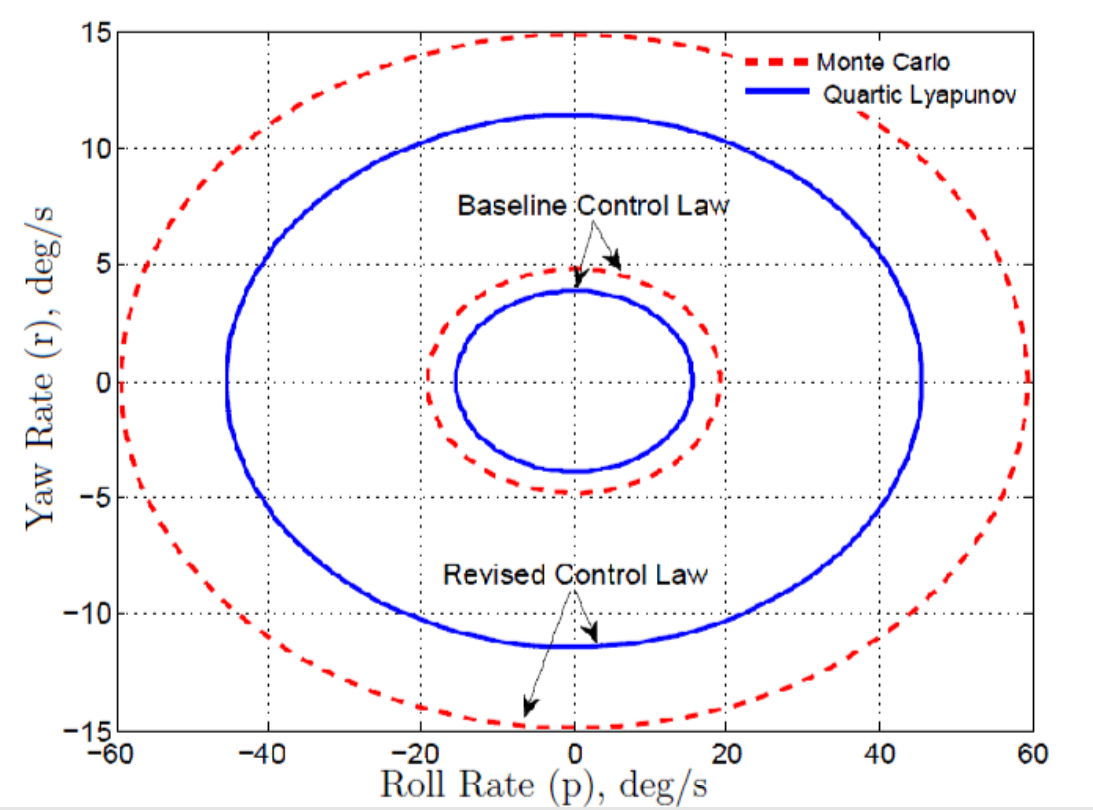
Nonlinear Region-of-Attraction Analysis (cont'd)

Results on Estimating ROA: α vs. β



Nonlinear Region-of-Attraction Analysis (cont'd)

Results on Estimating ROA: p vs. r



Summary : Analysis Results

Linear Analysis	**Baseline**	**Revised**
Multivariable Input Loop Phase Margin	$\pm 45.0^\circ$	$\pm 53^\circ$
Multivariable Input Loop Gain Margin (dB)	± 7.7	± 9.6
Diagonal Input Multiplicative: $(k_m = \frac{1}{\mu})$	0.80	0.87
Full Input Multiplicative: $(k_m = \frac{1}{\mu})$	0.11	0.47
Parametric Uncertainty: $(k_m = \frac{1}{\mu})$	2.85	10.0
Nonlinear Analysis: $\{x^{\mathrm{T}} N x \leq \beta^*\}$		
Linearized Lyapunov Function, $\underline{\beta}_{lin}$	4.62×10^{-6}	1.91×10^{-4}
Generic Quadratic Lyapunov Function, $\underline{\beta}_2$	6.96×10^{-4}	1.58×10^{-2}
Generic Quartic Lyapunov Function, $\underline{\beta}_4$	4.58×10^{-3}	3.98×10^{-2}
Monte Carlo Simulation, $\bar{\beta}_{MC}$	7.00×10^{-3}	7.00×10^{-2}

F/A-18 Aircraft Flight Control Law Recommendations

Input to the administrative action by the Naval Air Systems Command (NAVAIR) to prevent loss of F-18 aircraft to falling leaf mode entry. At $V_{\mathrm{TAS}} = 250 ft/s$

- Limit baseline controller to $\pm 15^{\circ}$ angle-of-attack.
- Limit revised controller to $\pm 50^{\circ}$ angle-of-attack

(Note that center-of-gravity location effect was not investigated.)

Summary of F/A-18 Flight Control Law Analysis

- Validation of flight control laws currently relies mainly on linear analysis tools and nonlinear (Monte Carlo) simulations.
- Linear analysis works well in general for assessing the robustness and performance around equilibrium conditions, though
  - Valid only over a small region in the state-space
  - Insufficient to address truly nonlinear phenomenon, e.g. falling leaf mode in the F/A-18 Hornet.
  - Linear analyses are not applicable for adaptive control laws or systems with hard nonlinearities.

Summary of F/A-18 Flight Control Law Analysis (cont'd)

- The nonlinear analysis tools described in this short course provide a quantitative performance/stability assement over a provable region of the state space.
- Nonlinear ROA analyses of the F/A-18 provided
  - Initial conditions which can be brought back to a stable equilibrium condition.
  - Metric for detecting departure phenomenon.
- Nonlinear ROA analyses is ideally suited to adress flight control law performance under upset conditions.
- Nonlinear analysis tools developed are also applicable to assessing stability and performance of adaptive controllers.

References

- A. J. van der Schaft, "$\mathcal{L}_2$-Gain and Passivity Techniques in Nonlinear Control," 2nd edition, Springer, 2000.
- W. M. McEneaney, "Max-Plus Methods for Nonlinear Control and Estimation," Birkhauser Systems and Control Series, 2006.
- J. W. Helton and M. R. James, "Extending $\mathcal{H}_\infty$ control to nonlinear systems: control of nonlinear systems to achieve performance objectives," SIAM, 1999.
- D. Henrion and A. Garulli, "Positive Polynomials in Control," Lecture Notes in Control and Information Sciences, Vol.312, 2005.
- M. Vidyasagar, "Nonlinear Systems Analysis," Prentice Hall, 1993.

References

- R. Tempo, G. Calafiore, and F. Dabbene, "Randomized Algorithms for Analysis and Control of Uncertain Systems," Springer, 2005.
- D. Henrion and G. Chesi, "Special Issue on Positive Polynomials in Control," IEEE Transactions on Automatic Control, 2009.
- R. Sepulchre, M. Jankovic, and P. V. Kokotovic, "Constructive Nonlinear Control," Springer, 1997.

  Exploiting other forms of Lyapunov functions has not been investigated thorougly. Composite quadratic forms seems like a fruitful area for SOS application.
- T. Hu and Z. Lin, "Composite Quadratic Lyapunov Functions for Constrained Control Systems," IEEE TAC , vol. 48, no. 3, 2003.

Van Der Pol Region of Attraction Problem

This demo will demonstrate how to use the ROA analysis tools. By increasing the degree of the Lyapunov function V(x) used to estimate the region of attraction, a larger ROA will be found.

Contents


```
format('compact')
clear all
close all
```

Problem Statement:

Given f, L1, L2, p, this code computes solutions to the problem maximize beta by choice of V, beta subject to:

$V(0) = 0$

$V \;\; L1 >= 0$

$\{x : p(x) \leq beta\} \subset \{x : V(x) \leq 1\} \subset \{x : V dot(x) < 0\}$

where Vdot = jacobian(V,x)*f(x)

This code solves SOS relaxations for the above problem following the procedure:

1. Generate an initial V feasible for the above constraints. - use simulation data + linearization - use only linearization

2. Improve the estimate of the ROA by further optimization, namely iterating between optimizing over the choice of "multipliers" for given V and optimizing over the choice of V given the multipliers.

Setup Dynamics

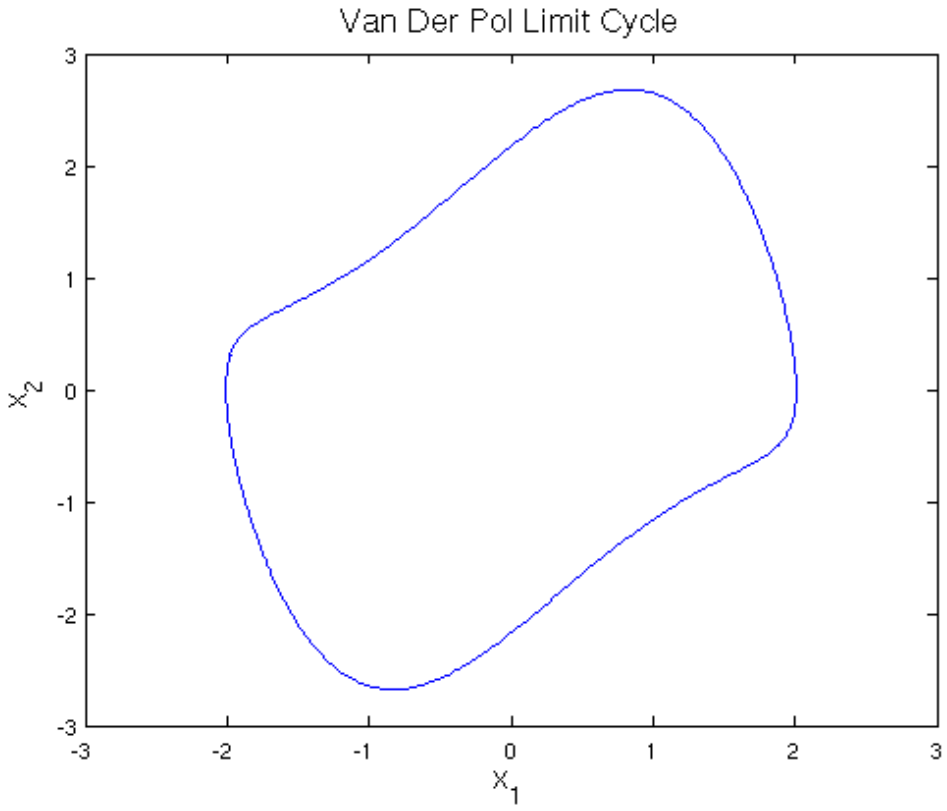
Form the vector field

```
pvar x1 x2;
x = [x1;x2];
```

```
x1dot = -x2;
x2dot = x1+(x1^2-1)*x2;
f= [x1dot; x2dot];
```

Plot Van Der Pol limit cycle

```
plotVDP
domain = [-3 3 -3 3];
axis(domain)
```



Quadratic V(x)

Extract default options. The software uses two different iterations: one requires bisection and one does not. The default does not require bisection. The SOS multipliers in the two different iterations are defined as follows:

1. Without bisection: r1, r2 (these are polynomials in x - non necessarily SOS)

2. With bisection: s1, s2, s3 (all SOS)

Most of the slides are written based on the conditions that require bisection. Therefore, to ensure that we are using bisection we set

Bis.flag=1 (default = 0).

```
Bis.flag = 1;
```

Generate the default options used for the rest of calculations. We omit the 3rd and 4th input arguments for now.

```
[roaconstr,opt,sys] = GetRoaOpts(f, x, [],[],Bis);
```

Set custom options. We adjust some of the tolerance levels in order to get better results. By decreasing the iterations and increasing the tolerances, we get worse results, but the program runs faster.

```
opt.sim.NumConvTraj = 100;
opt.coordoptim.MaxIters = 30;
opt.coordoptim.IterStopTol = 1e-4;
opt.getbeta.bistol = 1e-4;
opt.getgamma.bistol = 1e-4;
opt.display.roaest = 1;
```

If Bis.flag = 1, then the default basis vectors for s1, s2, s3 are the following. (si is constructed as si = zi*Mi*zi with Mi SOS)

```
z1 = roaconstr.z1
z2 = roaconstr.z2
z3 = roaconstr.z3
```

```
z1 =
  1
z2 =
  [ x1 ]
  [ x2 ]
z3 =
  1
```

```
V is constructed as follows: V(x)= A*zV, where A is a row vector of
decision variables. The default vector field for V is quadratic.
```

```
zV = roaconstr.zV
```

```
zV =
  [  x1^2 ]
  [ x1*x2 ]
  [  x2^2 ]
```

We will use the unit ball as the default shape for p. p(x)=x'*x

```
roaconstr.p
```

```
ans =
  x1^2 + x2^2
```

We will use 1e-6*x'*x as the default shape for L1 and L2. L1 and L2 are small positive definite sums of squares.

```
roaconstr.L1
roaconstr.L2
```

```
ans =
  1e-06*x1^2 + 1e-06*x2^2
ans =
  1e-06*x1^2 + 1e-06*x2^2
```

Estimate the ROA with Quadratic V(x). The function wrapper estimates the ROA. The second input argument is empty because our vector field, f, does not have any uncertainty.

```
outputs = wrapper(sys,[],roaconstr,opt);
```

```
------------------Beginning simulations
System 1: Num Stable = 1      Num Unstable = 1        Beta for Sims = 2.348   Beta UB = 2.348
System 1: Num Stable = 52     Num Unstable = 2        Beta for Sims = 2.231   Beta UB = 2.347
System 1: Num Stable = 100    Num Unstable = 2        Beta for Sims = 2.231   Beta UB = 2.347
------------------End of simulations
------------------Begin search for feasible V
Try = 1        Beta for Vfeas = 2.231
Try = 2        Beta for Vfeas = 2.119
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 1.495
-------------------Iteration = 1
Beta = 1.495 (Gamma = 0.739)
```

Get the V, betas and multipliers

```
[V,betaLower,gamma,p,multip,betaUpper] = extractSol(outputs);

V
s1 = multip.S1
s2 = multip.S2
betaLower
betaUpper
```

```
V =
```

```
  0.42647*x1^2 - 0.19336*x1*x2 + 0.35769*x2^2
s1 =
  0.49469
s2 =
  0.91877*x1^2 - 0.27698*x1*x2 + 2.4829*x2^2
betaLower =
    1.4947
betaUpper =
    2.2306
```

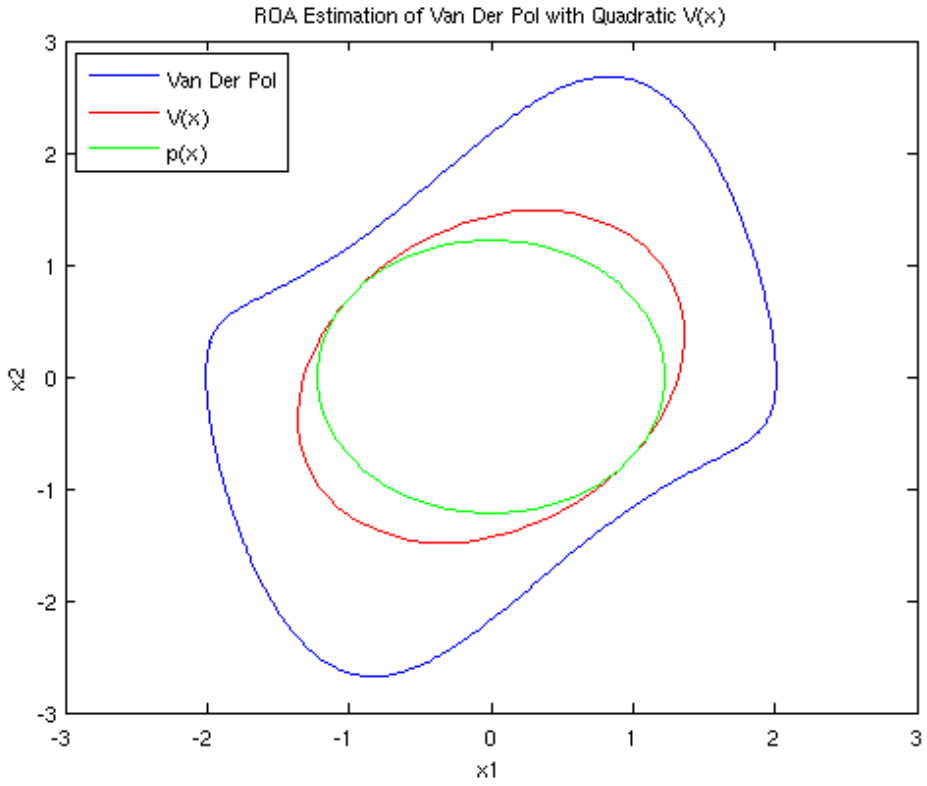
Verify polynomials are SOS

```
issos(V)
issos(s1)
issos(s2)
```

```
ans =
     1
ans =
     1
ans =
     1
```

Display Results. Plot p(x) and Quadratic V(x)

```
hold on;
betaLower=double(betaLower);
pcontour(V,gamma,domain,'r')
pcontour(p,betaLower, domain,'g')
title('ROA Estimation of Van Der Pol with Quadratic V(x)')
legend('Van Der Pol', 'V(x)', 'p(x)', 'Location', 'NorthWest')
axis(domain)
```



Quartic V(x)

Get default options again. This time zV specifies that V is quartic.

```
zV = monomials(x, 2:4);
[roaconstr,opt,sys] = GetRoaOpts(f, x, zV,[],Bis);
```

A larger ROA is obtained by searching over quartic s2 polynomials

```
roaconstr.z2 = monomials(x, 1:2);
```

Set custom options. We need to specify the settings again.

```
opt.sim.NumConvTraj = 100;
opt.coordoptim.MaxIters = 30;
opt.coordoptim.IterStopTol = 1e-4;
opt.getbeta.bistol = 1e-4;
opt.getgamma.bistol = 1e-4;
opt.display.roaest = 0;
```

Estimate the ROA with Quartic V(x)

```
outputs = wrapper(sys,[],roaconstr,opt);
```

Get the V, betas and multipliers

```
[V,betaLower,gamma,p,multip,betaUpper] = extractSol(outputs);

V
s1 = multip.S1
s2 = multip.S2
betaLower
betaUpper


V =
  0.0098431*x1^4 + 0.057867*x1^3*x2 + 0.070016*x1^2*x2^2
  - 0.045388*x1*x2^3 + 0.014722*x2^4 + 8.6629e-14*x1^3
  - 1.4808e-12*x1^2*x2 + 1.128e-13*x1*x2^2 + 1.1221e-12
  *x2^3 + 0.19913*x1^2 - 0.27147*x1*x2 + 0.14968*x2^2
s1 =
  0.13868*x1^2 + 0.099347*x1*x2 + 0.13018*x2^2 + 2.6481e
  -09*x1 + 2.2007e-09*x2 + 0.22947
s2 =
  0.90965*x1^4 - 0.54333*x1^3*x2 + 2.5467*x1^2*x2^2
  - 0.84735*x1*x2^3 + 0.14879*x2^4 + 2.3313e-11*x1^3
  - 2.5534e-11*x1^2*x2 - 5.8705e-11*x1*x2^2 + 2.8545e-11
  *x2^3 + 0.74107*x1^2 - 0.46906*x1*x2 + 0.076265*x2^2
betaLower =
    2.1413
betaUpper =
    2.3492
```

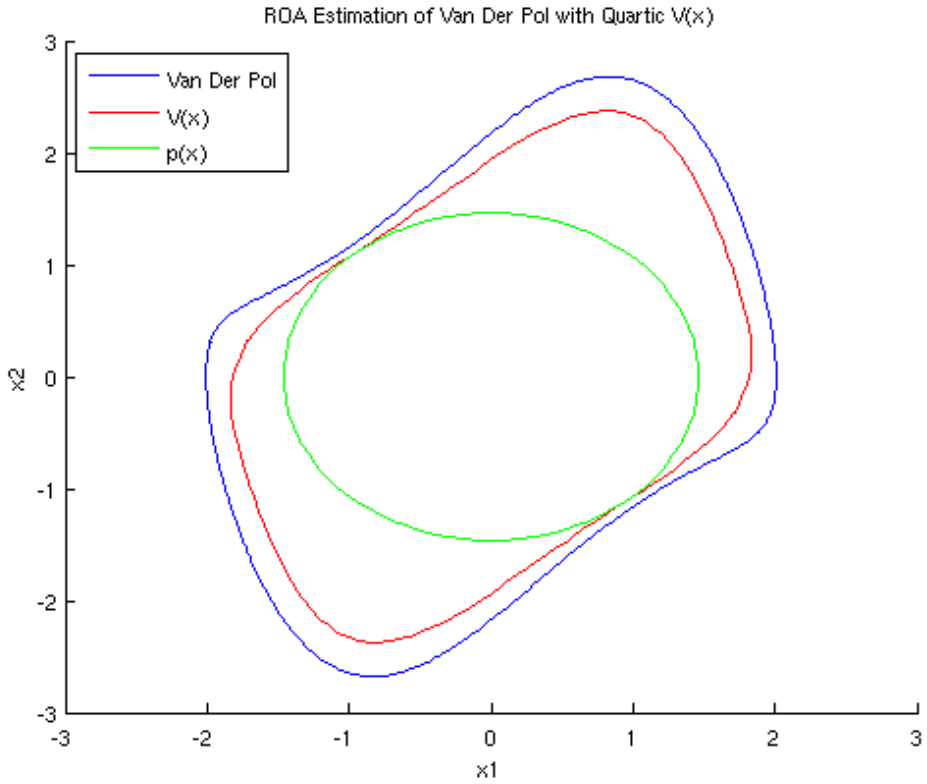
Verify polynomials are SOS

```
issos(V)
issos(s1)
issos(s2)


ans =
     1
ans =
     1
ans =
     1
```

Plot p(x) and V(x)

```
figure;
hold on;
plotVDP
```

```
pcontour(V,gamma,domain,'r')
pcontour(p,betaLower, domain,'g')
title('ROA Estimation of Van Der Pol with Quartic V(x)')
legend('Van Der Pol', 'V(x)', 'p(x)', 'Location', 'NorthWest')
axis(domain)
```



Degree 6 V(x)

Get default options again. This time zV specifies that the degree of V is 6.

```
zV = monomials(x, 2:6);
[roaconstr,opt,sys] = GetRoaOpts(f, x, zV,[],Bis);
```

A larger ROA is obtained by searching over quartic s2 polynomials

```
roaconstr.z2 = monomials(x, 1:2);
```

Set custom options. We need to specify the setting again.

```
opt.sim.NumConvTraj = 100;
opt.coordoptim.MaxIters = 30;
opt.coordoptim.IterStopTol = 1e-4;
```

```
opt.getbeta.bistol = 1e-4;
opt.getgamma.bistol = 1e-4;
opt.display.roaest = 0;
```

Estimate the ROA with degree 6 V(x) wrapper computes the ROA estimation routine. The second input argument is empty because our vector field, f, does not have any uncertainty. See "help wrapper" for instructions on specifying uncertain vector fields.

```
outputs = wrapper(sys,[],roaconstr,opt);
```

Get the V, betas and multipliers

```
[V,betaLower,gamma,p,multip,betaUpper] = extractSol(outputs);

V
s1 = multip.S1
s2 = multip.S2
betaLower
betaUpper
```

```
V =
  0.024602*x1^6 + 0.046349*x1^5*x2 - 0.0091987*x1^4*x2^2
  - 0.024579*x1^3*x2^3 + 0.041076*x1^2*x2^4 - 0.016574*x1
  *x2^5 + 0.0040183*x2^6 - 6.8829e-07*x1^5 + 2.2268e-06
  *x1^4*x2 - 4.454e-06*x1^3*x2^2 - 5.7385e-06*x1^2*x2^3
  + 3.6124e-06*x1*x2^4 - 2.5847e-06*x2^5 - 0.096009*x1^4
  - 0.11969*x1^3*x2 + 0.23584*x1^2*x2^2 - 0.12469*x1
  *x2^3 + 0.017174*x2^4 + 5.6344e-06*x1^3 + 1.6572e-06
  *x1^2*x2 + 4.3143e-06*x1*x2^2 + 1.3673e-05*x2^3
  + 0.44801*x1^2 - 0.27595*x1*x2 + 0.21466*x2^2
s1 =
  0.14772*x1^4 + 0.045747*x1^3*x2 + 0.11794*x1^2*x2^2
  + 0.039732*x1*x2^3 + 0.14143*x2^4 - 2.0472e-06*x1^3
  + 2.5874e-06*x1^2*x2 - 9.2161e-06*x1*x2^2 - 2.7463e-06
  *x2^3 + 0.031121*x1^2 + 0.12375*x1*x2 + 0.12099*x2^2
  + 7.3059e-06*x1 - 9.7849e-06*x2 + 0.60248
s2 =
  1.1735*x1^4 - 2.6837*x1^3*x2 + 2.4065*x1^2*x2^2
  - 0.022651*x1*x2^3 + 0.067581*x2^4 - 6.5573e-05*x1^3
  - 9.7436e-05*x1^2*x2 - 0.00024776*x1*x2^2 + 8.7312e-05
  *x2^3 + 0.15422*x1^2 - 0.023624*x1*x2 + 0.063628*x2^2
betaLower =
    2.3151
betaUpper =
    2.3478
```

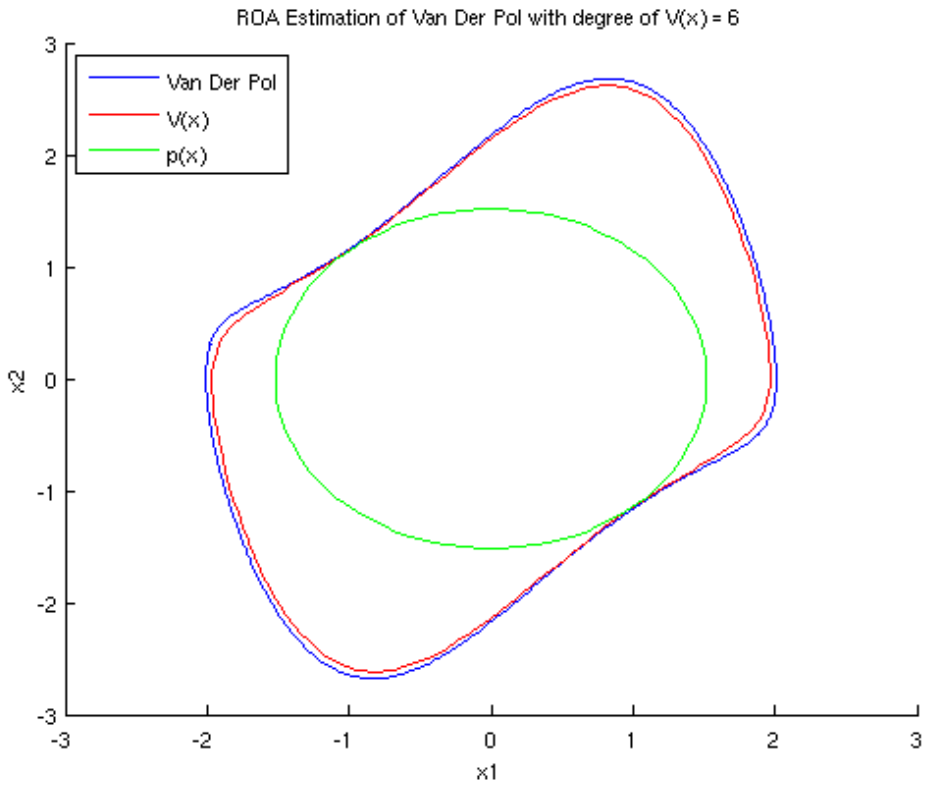
Verify polynomials are SOS

```
issos(V)
issos(s1)
issos(s2)
```

```
ans =
     1
ans =
     1
ans =
     1
```

Plot p(x) and degree 6 V(x)

```
figure;
hold on;
plotVDP
pcontour(V,gamma,domain,'r')
pcontour(p,betaLower, domain,'g')
title('ROA Estimation of Van Der Pol with degree of V(x) = 6')
legend('Van Der Pol', 'V(x)', 'p(x)', 'Location', 'NorthWest')
axis(domain)
```



For details of iterations with no bisection see:

Topcu, Seiler, and Packard, "Local Stability Analysis Using Simulations and Sum-of-Squares Programming," Automatica, 2008 or the slide titled "Application of Set Containment Conditions (2)."

Published with MATLAB® 7.8

n-th order, radial vector field with known ROA

Motivated by an example from Davison, Kurak, 1971, it is easy to create cubic vector fields with known ROA. With an arbitrarily chosen quadratic shape factor, it is also easy to compute the optimal Beta. The iteration can be tested by comparing the Beta obtained from iteration to the optimal Beta computed analytically.

Contents

Create matrices describing vector field and shape factor

2 positive-definite matrices for vector field and shape function

```
nX = 6;
[U,S,V] = svd(randn(nX,nX));  % Get "random" unitary matrices
lamB = diag(exp(2*randn(1,nX)));
lamR = diag(exp(2*randn(1,nX)));
B = U*lamB*U';  R = V*lamR*V';
```

Analytically compute BetaOpt; rescale data so BetaOpt = 1

```
betaOpt = 1/max(real(eig(B/R)));
B = B*betaOpt;
betaOpt = 1/max(real(eig(B/R)))
```

```
betaOpt =
    1.0000
```

Create Shape Function, and Vector field

```
x = mpvar('x',nX,1);
p = x'*R*x;
f = -x + x*(x'*B*x);
```

Create monomial basis for V (Vdeg = 2) and s1 and s2

```
zV = monomials(x,2:Vdeg);          % 'vec' form
```

```
z1maxd = ceil((Vdeg-p.maxdeg)/2);
z1 = monomials(x, 0:z1maxd );      % 'mat' form
z2 = monomials(x, 1:1);           % 'mat' form
L2 = 1e-6*(x'*x);
```

Run 3 steps of V-s iteration

```
Nsteps = 3;
opts = []; opts.gmin = 0; opts.gmax = 50; opts.L2 = L2;
for i1=1:Nsteps;
    if i1==1
        V = linstab(f,x);    % Initialize from Linearization
    else
        [V,c] = roavstep(f,p,x,zV,beta,gamma,s1,s2,opts);
        V = V/gamma;
    end
    [gbnds,s2] = pcontain(jacobian(V,x)*f+L2,V,z2,opts);
    gamma = gbnds(1);
    [bbnds,s1]  =pcontain(V-gamma,p,z1,opts);
    beta = bbnds(1);
    fprintf('i1 = %d  \t beta = %6.4f, betaOpt = %6.4f\n',i1,beta,betaOpt);
end
```

```
i1 = 1         beta = 0.3517, betaOpt = 1.0000
i1 = 2         beta = 0.9949, betaOpt = 1.0000
i1 = 3         beta = 0.9979, betaOpt = 1.0000
```

Published with MATLAB® 7.8

Upper bound demonstrations

Contents

VDP with deg(V) = 2

Form the vector field

```
pvar x1 x2;
x = [x1;x2];
x1dot = -x2;
x2dot = x1+(x1^2-1)*x2;
f = [x1dot; x2dot];
```

Get the default values of options to run the ROA code.

```
zV = monomials(x,2:2);
Bis.r1deg = 2;
```

Now, call GetRoaOpts to get the correponding opts, roaconstr, etc.

```
[roaconstr,opt,sys] = GetRoaOpts(f, x, zV, [],Bis);
opt.sim.NumConvTraj = 200;
opt.sim.dispEveryNth = 40;
opt.display.roaest = 1;
opt.coordoptim.IterStopTol = 1e-4;
```

Call the wrapper which in turn calls RoaEst.m

```
outputs = wrapper(sys,[],roaconstr,opt);
```

```
------------------Beginning simulations
System 1: Num Stable = 0        Num Unstable = 1        Beta for Sims = 2.348   Beta UB = 2.348
System 1: Num Stable = 40       Num Unstable = 1        Beta for Sims = 2.348   Beta UB = 2.348
System 1: Num Stable = 43       Num Unstable = 2        Beta for Sims = 2.230   Beta UB = 2.348
System 1: Num Stable = 80       Num Unstable = 2        Beta for Sims = 2.230   Beta UB = 2.348
System 1: Num Stable = 120      Num Unstable = 2        Beta for Sims = 2.230   Beta UB = 2.348
System 1: Num Stable = 160      Num Unstable = 2        Beta for Sims = 2.230   Beta UB = 2.348
System 1: Num Stable = 200      Num Unstable = 2        Beta for Sims = 2.230   Beta UB = 2.348
------------------End of simulations
------------------Begin search for feasible V
Try = 1          Beta for Vfeas = 2.230
Try = 2          Beta for Vfeas = 2.119
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 1.496
-------------------Iteration = 1
Beta = 1.513 (Gamma = 0.746)
-------------------Iteration = 2
Beta = 1.516 (Gamma = 0.747)
-------------------Iteration = 3
Beta = 1.517 (Gamma = 0.747)
-------------------Iteration = 4
Beta = 1.517 (Gamma = 0.747)
```

Extract the solution

```
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

Upper bounds from divergent trajectories

```
betaUpperDivergent = outputs.RoaEstInfo.info.SimLFG.sim.BetaUB;
betaUpperDivergent
```

```
betaUpperDivergent =

    2.3478
```

Upper bound from infeasibility of the relaxation

```
if betaUpper < outputs.RoaEstInfo.info.SimLFG.sim.BetaUB
    betaUpperInfeas = betaUpper;
    betaUpperInfeas
else
    display('No upper bound from infeasibility');
end
```

```
betaUpperInfeas =

    2.2304
```

Certified beta

```
betaCertified = beta;
betaCertified
```

```
betaCertified =

    1.5168
```

VDP with deg(V) = 6

Form the vector field

```
pvar x1 x2;
x = [x1;x2];
x1dot = -x2;
x2dot = x1+(x1^2-1)*x2;
f = [x1dot; x2dot];
```

Get the default values of options to run the ROA code.

```
zV = monomials(x,2:6);
Bis.r1deg = 4;
```

Now, call GetRoaOpts to get the correponding opts, roaconstr, etc.

```
[roaconstr,opt,sys] = GetRoaOpts(f, x, zV, [],Bis);
opt.sim.NumConvTraj = 200;
opt.sim.dispEveryNth = 40;
opt.display.roaest = 1;
opt.coordoptim.IterStopTol = 1e-4;
```

Call the wrapper which in turn calls RoaEst.m

```
outputs = wrapper(sys,[],roaconstr,opt);
```

```
------------------Beginning simulations
System 1: Num Stable = 0        Num Unstable = 1        Beta for Sims = 2.352   Beta UB = 2.352
System 1: Num Stable = 7        Num Unstable = 2        Beta for Sims = 2.235   Beta UB = 2.347
System 1: Num Stable = 40       Num Unstable = 2        Beta for Sims = 2.235   Beta UB = 2.347
System 1: Num Stable = 80       Num Unstable = 2        Beta for Sims = 2.235   Beta UB = 2.347
System 1: Num Stable = 120      Num Unstable = 2        Beta for Sims = 2.235   Beta UB = 2.347
System 1: Num Stable = 160      Num Unstable = 2        Beta for Sims = 2.235   Beta UB = 2.347
System 1: Num Stable = 200      Num Unstable = 2        Beta for Sims = 2.235   Beta UB = 2.347
------------------End of simulations
------------------Begin search for feasible V
Try = 1          Beta for Vfeas = 2.235
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 0.583
-------------------Iteration = 1
Beta = 1.436 (Gamma = 1.387)
-------------------Iteration = 2
Beta = 1.731 (Gamma = 1.618)
-------------------Iteration = 3
Beta = 1.886 (Gamma = 1.770)
-------------------Iteration = 4
Beta = 1.981 (Gamma = 1.874)
-------------------Iteration = 5
Beta = 2.049 (Gamma = 1.948)
-------------------Iteration = 6
Beta = 2.098 (Gamma = 2.001)
-------------------Iteration = 7
Beta = 2.134 (Gamma = 2.041)
-------------------Iteration = 8
Beta = 2.164 (Gamma = 2.072)
-------------------Iteration = 9
Beta = 2.188 (Gamma = 2.098)
-------------------Iteration = 10
Beta = 2.209 (Gamma = 2.119)
-------------------Iteration = 11
Beta = 2.228 (Gamma = 2.137)
-------------------Iteration = 12
Beta = 2.244 (Gamma = 2.153)
-------------------Iteration = 13
Beta = 2.259 (Gamma = 2.166)
-------------------Iteration = 14
Beta = 2.272 (Gamma = 2.178)
-------------------Iteration = 15
Beta = 2.283 (Gamma = 2.188)
-------------------Iteration = 16
Beta = 2.293 (Gamma = 2.197)
-------------------Iteration = 17
Beta = 2.301 (Gamma = 2.204)
-------------------Iteration = 18
Beta = 2.308 (Gamma = 2.209)
-------------------Iteration = 19
Beta = 2.313 (Gamma = 2.213)
-------------------Iteration = 20
Beta = 2.317 (Gamma = 2.217)
```

Extract the solution

```
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

Upper bounds from divergent trajectories

```
betaUpperDivergent = outputs.RoaEstInfo.info.SimLFG.sim.BetaUB;
betaUpperDivergent
```

```
betaUpperDivergent =

    2.3470
```

Upper bound from infeasibility of the relaxation

```
if betaUpper < outputs.RoaEstInfo.info.SimLFG.sim.BetaUB
    betaUpperInfeas = betaUpper;
    betaUpperInfeas
else
    display('No upper bound from infeasibility');
end
```

```
No upper bound from infeasibility
```

Certified beta

```
betaCertified = beta;
betaCertified
```

```
betaCertified =

    2.3173
```

Published with MATLAB® 7.6

4 state aircraft problem with nominal dynamics

Contents

deg(V) = 2 analysis

Form the vector field

```
pvar x1 x2 x3 x4
x = [x1;x2;x3;x4];

f0 = [-0.24366*x2^3+0.082272*x1*x2+0.30492*x2^2+0.015426*x2*x3-0.082272*x2
      -0.054444*x2^2+0.10889*x2*x3-0.054444*x3^2+0.91136*x1-0.64516*x2-0.0
      x1;
      -0.864*x1-0.3211*x3];

fdX = [ 0.30765*x2^3+0.099232*x2^2+0.12404*x1+0.90912*x2+0.023258*x3-0.124
        0.00045754*x2;
        0;
        0];

fdm = [0;
       -0.054444*x2^2+0.10889*x2*x3-0.054444*x3^2-0.6445*x2-0.016621*x3+0.
       0;
       0];

fQ = [-0.0068307*x2^2-0.001428*x2;
      0;
      0;
      0];

pvar dX dm

f = f0+fdX*dX+fdm*dm+fQ*dm*dX;

dXrange = [0.99 2.05];
dmrange = [-0.1 0.1];
```

Form the nominal vector field

```
fnom = subs(f,{'dX','dm'},{sum(dXrange)/2,sum(dmrange)/2});
```

Iteration options and basis vector for V

Quadratic V

```
zV = monomials(x,2:2);
```

Use iterations with no bisection

```
Bis.flag = 0;
Bis.r1deg = 1;
```

Generate all options

```
[roaconstr,opt,sys] = GetRoaOpts(fnom, x, zV, [], Bis);
```

Modify the options

Options for coordinate-wise affine iterations (stopping tolerance max number of iterations)

```
opt.coordoptim.IterStopTol = 1e-2;
opt.coordoptim.MaxIters = 15;
```

Display the results of the simulations and iterations

```
opt.display.roaest = 1;
```

Use pt.sim.NumConvTraj convergent trajectories in forming the LP and display the simulations data after every pt.sim.NumConvTraj convergent trajectories are foud

```
opt.sim.NumConvTraj = 100;
opt.sim.dispEveryNth = 20;
```

opt.sim.flag = 1 for simulations + linearization for initial V

opt.sim.flag = 0 only linearization for initial V

```
opt.sim.flag = 1;
```

Start collecting simulation data from p <= opt.sim.BetaInit

```
opt.sim.BetaMax = 8;
opt.sim.BetaInit = 8;
```

Solve the problem and extract the solution

```
outputs = wrapper(sys,[],roaconstr,opt);
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

```
------------------Beginning simulations
System 1: Num Stable = 20  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 40  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 60  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 80  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 100  Num Unstable = 0  Beta for Sims = 8.000  Beta
------------------End of simulations
------------------Begin search for feasible V
Try = 1   Beta for Vfeas = 8.000
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 6.295
-------------------Iteration = 1
Beta = 7.929 (Gamma = 0.885)
-------------------Iteration = 2
Beta = 8.553 (Gamma = 0.925)
-------------------Iteration = 3
Beta = 8.855 (Gamma = 0.943)
-------------------Iteration = 4
Beta = 8.966 (Gamma = 0.950)
-------------------Iteration = 5
Beta = 9.018 (Gamma = 0.953)
-------------------Iteration = 6
Beta = 9.047 (Gamma = 0.955)
-------------------Iteration = 7
Beta = 9.067 (Gamma = 0.956)
-------------------Iteration = 8
Beta = 9.081 (Gamma = 0.957)
-------------------Iteration = 9
Beta = 9.093 (Gamma = 0.958)
-------------------Iteration = 10
Beta = 9.103 (Gamma = 0.958)
-------------------Iteration = 11
Beta = 9.112 (Gamma = 0.959)
```

Certified beta

```
beta
```

```
beta =
```

```
    9.1122
```

Upper bound

```
if betaUpper >=  opt.getbeta.maxbeta
    display('No upper bound has been established');
else
    betaUpper
end
```

```
No upper bound has been established
```

deg(V) = 4 analysis

Iteration options and basis vector for V

Change the degree of V to 4

```
zV = monomials(x,2:4);
```

Generate all options

```
[roaconstr,opt,sys] = GetRoaOpts(fnom, x, zV, [], Bis);
```

Modify the options (see comments above)

```
opt.coordoptim.IterStopTol = 1e-2;
opt.coordoptim.MaxIter = 20;
opt.display.roaest = 1;
opt.sim.NumConvTraj = 200;
opt.sim.dispEveryNth = 20;
opt.sim.flag = 1;
opt.sim.BetaMax = 20;
opt.sim.BetaInit = 20;
```

Solve the problem and extract the solution

```
outputs = wrapper(sys,[],roaconstr,opt);
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

```
------------------Beginning simulations
```

```
System 1: Num Stable = 20  Num Unstable = 0  Beta for Sims = 20.000  Beta f
System 1: Num Stable = 40  Num Unstable = 0  Beta for Sims = 20.000  Beta f
System 1: Num Stable = 57  Num Unstable = 1  Beta for Sims = 19.000  Beta f
System 1: Num Stable = 60  Num Unstable = 1  Beta for Sims = 19.000  Beta f
System 1: Num Stable = 80  Num Unstable = 1  Beta for Sims = 19.000  Beta f
System 1: Num Stable = 100  Num Unstable = 1  Beta for Sims = 19.000  Beta
System 1: Num Stable = 120  Num Unstable = 1  Beta for Sims = 19.000  Beta
System 1: Num Stable = 130  Num Unstable = 2  Beta for Sims = 18.050  Beta
System 1: Num Stable = 140  Num Unstable = 2  Beta for Sims = 18.050  Beta
System 1: Num Stable = 160  Num Unstable = 2  Beta for Sims = 18.050  Beta
System 1: Num Stable = 180  Num Unstable = 2  Beta for Sims = 18.050  Beta
System 1: Num Stable = 181  Num Unstable = 3  Beta for Sims = 17.148  Beta
System 1: Num Stable = 200  Num Unstable = 3  Beta for Sims = 17.148  Beta
------------------End of simulations
------------------Begin search for feasible V
Try = 1   Beta for Vfeas = 17.148
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 2.771
-------------------Iteration = 1
Beta = 7.002 (Gamma = 0.730)
-------------------Iteration = 2
Beta = 9.503 (Gamma = 1.043)
-------------------Iteration = 3
Beta = 10.779 (Gamma = 1.235)
-------------------Iteration = 4
Beta = 11.707 (Gamma = 1.357)
-------------------Iteration = 5
Beta = 12.346 (Gamma = 1.434)
-------------------Iteration = 6
Beta = 12.929 (Gamma = 1.519)
-------------------Iteration = 7
Beta = 13.449 (Gamma = 1.598)
-------------------Iteration = 8
Beta = 13.892 (Gamma = 1.665)
-------------------Iteration = 9
Beta = 14.256 (Gamma = 1.719)
-------------------Iteration = 10
Beta = 14.566 (Gamma = 1.764)
-------------------Iteration = 11
Beta = 14.828 (Gamma = 1.802)
-------------------Iteration = 12
Beta = 15.060 (Gamma = 1.836)
-------------------Iteration = 13
Beta = 15.249 (Gamma = 1.862)
-------------------Iteration = 14
Beta = 15.409 (Gamma = 1.884)
-------------------Iteration = 15
```

```
Beta = 15.537 (Gamma = 1.901)
------------------Iteration = 16
Beta = 15.645 (Gamma = 1.917)
------------------Iteration = 17
Beta = 15.725 (Gamma = 1.928)
------------------Iteration = 18
Beta = 15.782 (Gamma = 1.937)
------------------Iteration = 19
Beta = 15.828 (Gamma = 1.943)
------------------Iteration = 20
Beta = 15.862 (Gamma = 1.949)
```

Certified beta

```
beta
```

```
beta =

   15.8618
```

Upper bound

```
if betaUpper >=  opt.getbeta.maxbeta
    display('No upper bound has been established');
else
    betaUpper
end
```

```
betaUpper =

   17.9737
```

Published with MATLAB® 7.6

Robust ROA calculations

dynamics:

x1dot = x2;

x2dot = -x2-2*x1+2*x1^3 + delta*(-x1+x1^3);

with delta \in [-1,1]

This example was also used in Topcu and Packard, IEEE TAC, 2009 (in the special issue on positive polynomials in controls (example 1 in the paper)

```
% Form the vector field
pvar x1 x2;
x = [x1;x2];
x1dot = x2;
x2dot = -x2-2*x1+2*x1^3;
```

Nominal system

```
f = [x1dot; x2dot];
```

Introduce an uncertain parameter

```
pvar d1
```

Specify its range

```
ini_cell = [-1 1];
```

Form the uncertain vector field

```
f = f + d1*[0; -x1+x1^3];
```

```
% Get the vertex system
[roaconstr,opt,sys] = GetRoaOpts(f, x);
[fNOM,fVER] = getf(sys,ini_cell);

% Generate the options, etc.
zV = monomials(x,2:4);
Bis.flag = 0;
Bis.r1deg = 4;
```

```
[roaconstr,opt,sys] = GetRoaOpts(fVER, x, zV, [], Bis);
sys.fWithDel = [];

opt.sim.NumConvTraj = 40;
opt.display.roaest = 1;
```

Run the computations

```
outputs = wrapper(sys,[],roaconstr,opt);
```

```
------------------Beginning simulations
System 1: Num Stable = 0        Num Unstable = 1        Beta for Sims = 3.289   Beta UB = 3.289
System 1: Num Stable = 0        Num Unstable = 2        Beta for Sims = 1.390   Beta UB = 1.390
System 1: Num Stable = 2        Num Unstable = 3        Beta for Sims = 1.306   Beta UB = 1.306
System 1: Num Stable = 4        Num Unstable = 4        Beta for Sims = 0.913   Beta UB = 0.913
System 1: Num Stable = 6        Num Unstable = 5        Beta for Sims = 0.861   Beta UB = 0.861
System 1: Num Stable = 12       Num Unstable = 6        Beta for Sims = 0.818   Beta UB = 0.842
System 1: Num Stable = 18       Num Unstable = 7        Beta for Sims = 0.777   Beta UB = 0.808
System 2: Num Stable = 1        Num Unstable = 1        Beta for Sims = 1.476   Beta UB = 0.808
System 2: Num Stable = 3        Num Unstable = 2        Beta for Sims = 1.402   Beta UB = 0.808
System 2: Num Stable = 6        Num Unstable = 3        Beta for Sims = 1.114   Beta UB = 0.808
System 2: Num Stable = 6        Num Unstable = 4        Beta for Sims = 1.058   Beta UB = 0.808
System 2: Num Stable = 8        Num Unstable = 5        Beta for Sims = 1.000   Beta UB = 0.808
System 2: Num Stable = 10       Num Unstable = 6        Beta for Sims = 0.929   Beta UB = 0.808
System 2: Num Stable = 11       Num Unstable = 7        Beta for Sims = 0.882   Beta UB = 0.808
------------------End of simulations
------------------Begin search for feasible V
Try = 1          Beta for Vfeas = 0.882
Try = 2          Beta for Vfeas = 0.838
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 0.173
-------------------Iteration = 1
Beta = 0.567 (Gamma = 0.535)
-------------------Iteration = 2
Beta = 0.665 (Gamma = 0.604)
-------------------Iteration = 3
Beta = 0.716 (Gamma = 0.640)
-------------------Iteration = 4
Beta = 0.739 (Gamma = 0.656)
```

Extract the solution

```
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

```
beta
```

```
beta =

    0.7388
```

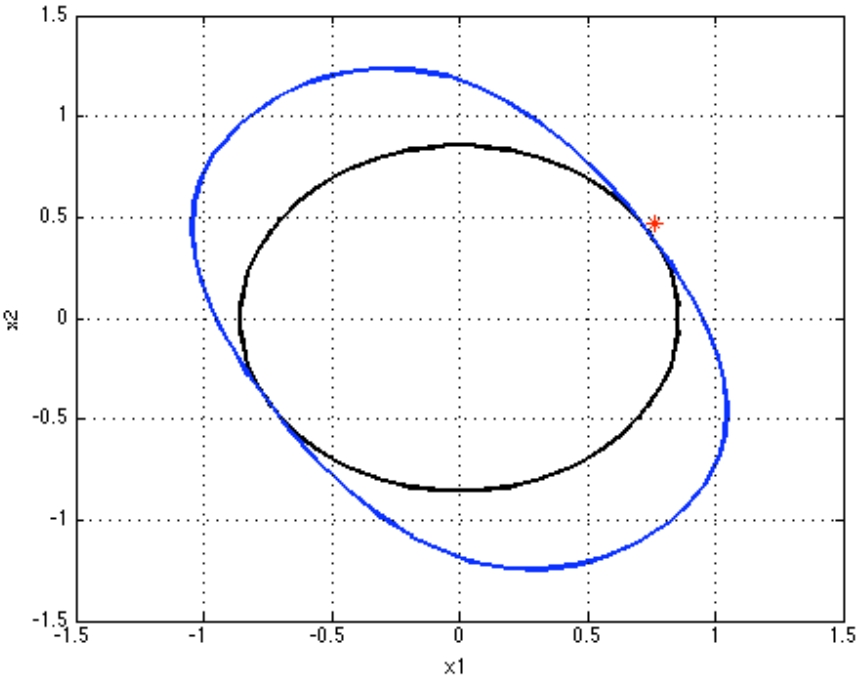
Upper bound on beta

```
betaUpper
```

```
betaUpper =

    0.8822
```

Plot the results

```
[Cp4,hp4] = pcontour(p,beta,[-2 2 -2 2],'k'); hold on;
set(hp4,'linewidth',2);
[CV4,hV4] = pcontour(V,gamma,[-2 2 -2 2],'b');
set(hV4,'linewidth',2);
set(gca,'xlim',[-1.5 1.5],'ylim',[-1.5 1.5]);

traj = outputs.RoaEstInfo.info.SimLFG.sim.Trajectories(1).unstab(end).state;
pval = peval(traj,p.coef,p.deg);
[aux,ind] = min(pval);
plot(traj(1,ind),traj(2,ind),'r*','markersize',8);
grid on;
```



Published with MATLAB® 7.6

Robust ROA calculations

dynamics:

x1dot = x2;

x2dot = -x2-2*x1+2*x1^3 + delta*(-x1+x1^3);

with delta \in [-1,1]

This example was also used in Topcu and Packard, IEEE TAC, 2009 (in the special issue on positive polynomials in controls (example 1 in the paper)

```
% Form the vector field
pvar x1 x2;
x = [x1;x2];
x1dot = x2;
x2dot = -x2-2*x1+2*x1^3;
```

Nominal system

```
f = [x1dot; x2dot];
```

Introduce an uncertain parameter

```
pvar d1
```

Specify its range

```
ini_cell = [-1 1];
```

Form the uncertain vector field

```
f = f + d1*[0; -x1+x1^3];
```

```
% Get the vertex system
[roaconstr,opt,sys] = GetRoaOpts(f, x);
[fNOM,fVER] = getf(sys,ini_cell);

% Generate the options, etc.
zV = monomials(x,2:4);
Bis.flag = 0;
Bis.r1deg = 4;
```

```
[roaconstr,opt,sys] = GetRoaOpts(fVER, x, zV, [], Bis);
sys.fWithDel = [];

opt.sim.NumConvTraj = 40;
opt.display.roaest = 1;
```

Run the computations

```
outputs = wrapper(sys,[],roaconstr,opt);
```

```
------------------Beginning simulations
System 1: Num Stable = 0        Num Unstable = 1        Beta for Sims = 3.289   Beta UB = 3.289
System 1: Num Stable = 0        Num Unstable = 2        Beta for Sims = 1.390   Beta UB = 1.390
System 1: Num Stable = 2        Num Unstable = 3        Beta for Sims = 1.306   Beta UB = 1.306
System 1: Num Stable = 4        Num Unstable = 4        Beta for Sims = 0.913   Beta UB = 0.913
System 1: Num Stable = 6        Num Unstable = 5        Beta for Sims = 0.861   Beta UB = 0.861
System 1: Num Stable = 12       Num Unstable = 6        Beta for Sims = 0.818   Beta UB = 0.842
System 1: Num Stable = 18       Num Unstable = 7        Beta for Sims = 0.777   Beta UB = 0.808
System 2: Num Stable = 1        Num Unstable = 1        Beta for Sims = 1.476   Beta UB = 0.808
System 2: Num Stable = 3        Num Unstable = 2        Beta for Sims = 1.402   Beta UB = 0.808
System 2: Num Stable = 6        Num Unstable = 3        Beta for Sims = 1.114   Beta UB = 0.808
System 2: Num Stable = 6        Num Unstable = 4        Beta for Sims = 1.058   Beta UB = 0.808
System 2: Num Stable = 8        Num Unstable = 5        Beta for Sims = 1.000   Beta UB = 0.808
System 2: Num Stable = 10       Num Unstable = 6        Beta for Sims = 0.929   Beta UB = 0.808
System 2: Num Stable = 11       Num Unstable = 7        Beta for Sims = 0.882   Beta UB = 0.808
------------------End of simulations
------------------Begin search for feasible V
Try = 1          Beta for Vfeas = 0.882
Try = 2          Beta for Vfeas = 0.838
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 0.173
-------------------Iteration = 1
Beta = 0.567 (Gamma = 0.535)
-------------------Iteration = 2
Beta = 0.665 (Gamma = 0.604)
-------------------Iteration = 3
Beta = 0.716 (Gamma = 0.640)
-------------------Iteration = 4
Beta = 0.739 (Gamma = 0.656)
```

Extract the solution

```
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

```
beta
```

```
beta =

    0.7388
```

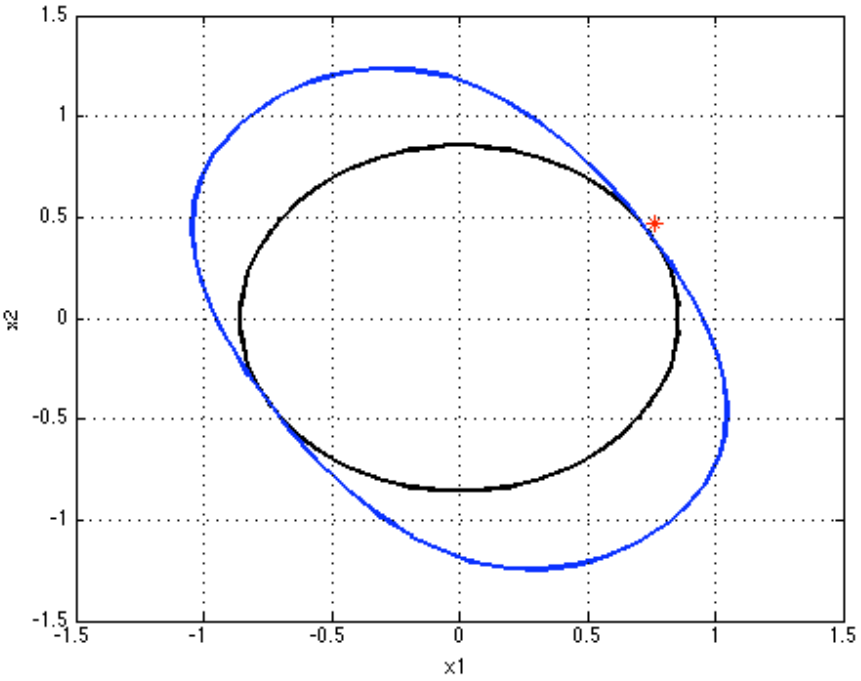
Upper bound on beta

```
betaUpper
```

```
betaUpper =

    0.8822
```

Plot the results

```
[Cp4,hp4] = pcontour(p,beta,[-2 2 -2 2],'k'); hold on;
set(hp4,'linewidth',2);
[CV4,hV4] = pcontour(V,gamma,[-2 2 -2 2],'b');
set(hV4,'linewidth',2);
set(gca,'xlim',[-1.5 1.5],'ylim',[-1.5 1.5]);

traj = outputs.RoaEstInfo.info.SimLFG.sim.Trajectories(1).unstab(end).state;
pval = peval(traj,p.coef,p.deg);
[aux,ind] = min(pval);
plot(traj(1,ind),traj(2,ind),'r*','markersize',8);
grid on;
```



Published with MATLAB® 7.6

4 state aircraft problem with parametric uncertainty

Contents

Form the vector field and generate options

```
pvar x1 x2 x3 x4
x = [x1;x2;x3;x4];

f0 = [-0.24366*x2^3+0.082272*x1*x2+0.30492*x2^2+0.015426*x2*x3-0.082272*x2
      -0.054444*x2^2+0.10889*x2*x3-0.054444*x3^2+0.91136*x1-0.64516*x2-0.0
      x1;
      -0.864*x1-0.3211*x3];

fdX = [ 0.30765*x2^3+0.099232*x2^2+0.12404*x1+0.90912*x2+0.023258*x3-0.124
        0.00045754*x2;
        0;
        0];

fdm = [0;
       -0.054444*x2^2+0.10889*x2*x3-0.054444*x3^2-0.6445*x2-0.016621*x3+0.
       0;
       0];

fQ = [-0.0068307*x2^2-0.001428*x2;
      0;
      0;
      0];

pvar dX dm

f = f0+fdX*dX+fdm*dm+fQ*dm*dX;

dXrange = [0.99 2.05];
dmrange = [-0.1 0.1];
```

Iteration options and basis vector for V

```
zV = monomials(x,2:2);
Bis.flag = 0;
Bis.r1deg = 1;
```

Generate all options

```
[roaconstr,opt,sys] = GetRoaOpts(f, x, zV, [], Bis);
```

specify the range of parameters in the order they appear in sys.delvector

```
sys.delvector
```

```
ans =
  [ dX ]
  [ dm ]
```

```
ini_cell = [dXrange dmrange];
```

Modify the options

```
opt.coordoptim.IterStopTol = 1e-2;
opt.coordoptim.MaxIters = 20;
opt.display.roaest = 1;
opt.sim.NumConvTraj = 60;
opt.sim.dispEveryNth = 20;
opt.sim.flag = 1;
opt.sim.BetaMax = 8;
opt.sim.BetaInit = 8;
```

Robust ROA with deg(V) = 2

Let's first run a quick robust ROA analysis without B&B

Solve the problem and extract the solution

```
outputs = wrapper(sys,ini_cell,roaconstr,opt);
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

```
*** Start cellBetaCenter ***
No Prior V - Run Sim-Based Analysis
------------------Beginning simulations
System 1: Num Stable = 20  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 40  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 60  Num Unstable = 0  Beta for Sims = 8.000  Beta U
```

```
------------------End of simulations
------------------Begin search for feasible V
Try = 1   Beta for Vfeas = 8.000
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 6.977
Pre-iteration V with Beta = 6.977 (Gamma = 0.919)
-------------------Iteration = 1
Beta = 8.250 (Gamma = 1.013)
-------------------Iteration = 2
Beta = 8.696 (Gamma = 1.045)
-------------------Iteration = 3
Beta = 8.895 (Gamma = 1.059)
-------------------Iteration = 4
Beta = 8.985 (Gamma = 1.065)
-------------------Iteration = 5
Beta = 9.036 (Gamma = 1.068)
-------------------Iteration = 6
Beta = 9.060 (Gamma = 1.070)
-------------------Iteration = 7
Beta = 9.076 (Gamma = 1.071)
-------------------Iteration = 8
Beta = 9.089 (Gamma = 1.072)
-------------------Iteration = 9
Beta = 9.100 (Gamma = 1.073)
-------------------Iteration = 10
Beta = 9.109 (Gamma = 1.073)
------- "Nominal" Beta =  9.109 ----
*** Verify at the vertices ***
****** Robust results:
Robust Beta = 3.949 (Nominal Beta = 9.109)
*** End of cellBetaCenter ***
```

Extract the solution

```
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

Certified beta without B&B

```
beta
```

```
beta =

    3.9487
```

B&B with deg(V) = 2

Change the options to run B&B

```
opt.display.BB = 1;
opt.BB.runBB = 1;
opt.BB.max_iter = 9;
```

Do not display the details within B&B steps

```
opt.display.roaest = 1;
```

Solve the problem and extract the solution

```
outputs = wrapper(sys,ini_cell,roaconstr,opt);
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

```
***Start B&B refinement***
-------Start cellBetaCenter for the initial partition --
*** Start cellBetaCenter ***
No Prior V - Run Sim-Based Analysis
------------------Beginning simulations
System 1: Num Stable = 20  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 40  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 60  Num Unstable = 0  Beta for Sims = 8.000  Beta U
------------------End of simulations
------------------Begin search for feasible V
Try = 1   Beta for Vfeas = 8.000
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 6.361
Pre-iteration V with Beta = 6.361 (Gamma = 0.828)
-------------------Iteration = 1
Beta = 8.185 (Gamma = 0.961)
-------------------Iteration = 2
Beta = 8.741 (Gamma = 0.997)
-------------------Iteration = 3
Beta = 8.922 (Gamma = 1.008)
-------------------Iteration = 4
Beta = 9.000 (Gamma = 1.013)
-------------------Iteration = 5
Beta = 9.039 (Gamma = 1.016)
-------------------Iteration = 6
Beta = 9.062 (Gamma = 1.017)
-------------------Iteration = 7
Beta = 9.079 (Gamma = 1.018)
```

```
------------------Iteration = 8
Beta = 9.092 (Gamma = 1.019)
------------------Iteration = 9
Beta = 9.103 (Gamma = 1.020)
------------------Iteration = 10
Beta = 9.112 (Gamma = 1.021)
------- "Nominal" Beta =  9.112 ----
*** Verify at the vertices ***
****** Robust results:
Robust Beta = 3.943 (Nominal Beta = 9.112)
*** End of cellBetaCenter ***
-------End cellBetaCenter for the initial partition --
Current Beta  = 3.943,   Number of active cells = 1
-------Start B&B iteration number = 1 --
*** Start cellBetaCenter ***
No Prior V - Run Sim-Based Analysis
------------------Beginning simulations
System 1: Num Stable = 20  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 40  Num Unstable = 0  Beta for Sims = 8.000  Beta U
System 1: Num Stable = 60  Num Unstable = 0  Beta for Sims = 8.000  Beta U
------------------End of simulations
------------------Begin search for feasible V
Try = 1   Beta for Vfeas = 8.000
------------------Found feasible V
Initial V (from the cvx outer bnd) gives Beta = 5.400
Pre-iteration V with Beta = 5.400 (Gamma = 0.699)
------------------Iteration = 1
Beta = 7.479 (Gamma = 0.879)
------------------Iteration = 2
Beta = 8.551 (Gamma = 0.949)
------------------Iteration = 3
Beta = 9.100 (Gamma = 0.979)
------------------Iteration = 4
Beta = 9.428 (Gamma = 0.996)
------------------Iteration = 5
Beta = 9.621 (Gamma = 1.007)
------------------Iteration = 6
Beta = 9.739 (Gamma = 1.014)
------------------Iteration = 7
Beta = 9.817 (Gamma = 1.018)
------------------Iteration = 8
Beta = 9.868 (Gamma = 1.021)
------------------Iteration = 9
Beta = 9.907 (Gamma = 1.024)
------------------Iteration = 10
...
```

Extract the solution

```
[V,beta,gamma,p,multip,betaUpper] = extractSol(outputs);
```

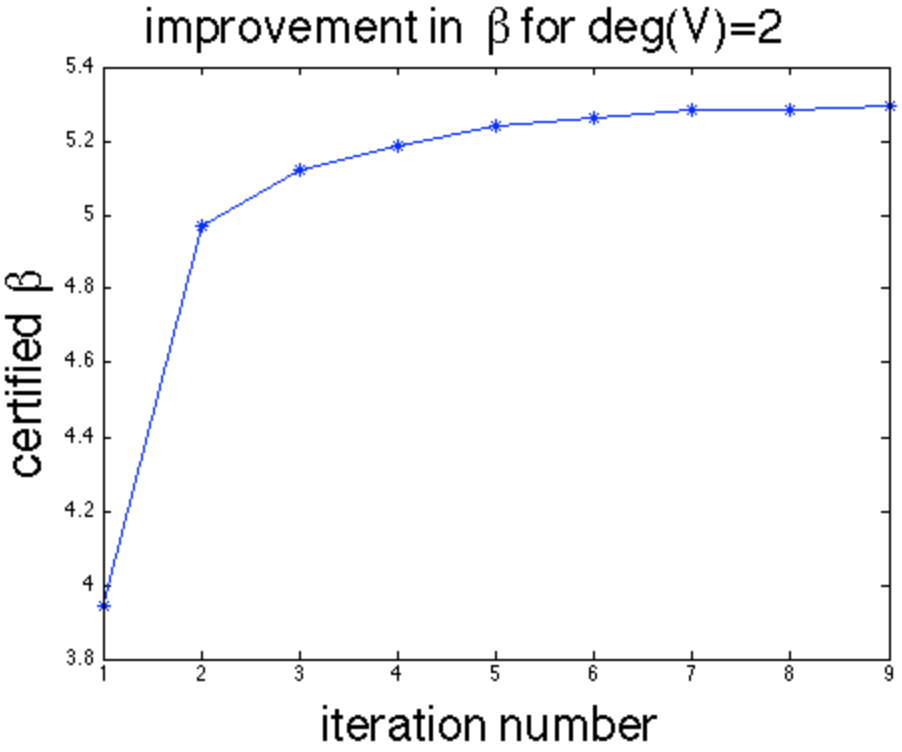
Certified beta with non B&B

```
beta
```

```
beta =

    5.2997
```

Plot the improvement in beta over B&B steps

```
dd = outputs.BBInfo.info(1);

for i1 = 1:opt.BB.max_iter
    act = dd(end).Active;
    ind = act == 1;
    B(i1) = min(dd.Beta_vec(ind));
    dd = outputs.BBInfo.info(i1*2+1);
end

plot(B,'*-');hold on;
xlabel('iteration number','fontsize',24)
ylabel('certified \beta','fontsize',24);
title('improvement in \beta for deg(V)=2','fontsize',24);



%
```



ROA computation for system with unmodeled dynamics

This code demonstrates the robust ROA calculations for systems with unmodeled dynamics on the controlled aircraft dynamics where the unmodeled dynamics connect to the nominal model as shown in slide titled "Example: Controlled aircraft dynamics with unmodeled dynamics" in section "Robust ROA and performance analysis with unmodeled dynamics"

Form the input out dynamics (w-->z) with

xdot = f(x,w)

z = h(x)

```
pvar x1 x2 x3 x4 w

x = [x1;x2;x3;x4];
y = [x1;x3];
Cc = 2;
v = Cc*x4;
Ac = [-0.864 -0.3211];
x4dot = Ac*y;
u = 1.25*v + w;
f(1,1) = -0.24366*x2^3 + 0.082272*x1*x2 + 0.30492*x2^2 - 0.082272*x2*u/2 +
f(2,1) = -0.054444*x2^2 + 0.10889*x2*x3 - 0.054444*x3^2 + 0.91136*x1 - 0.6
f(3,1) = x1;
f(4,1) = x4dot;
h = 0.75*Cc*x4;
```

The unmodeled dynamics has gain <= 1

```
gainUnmodeled = 1;
```

Generate some options used in the iterations

V - l1 is SOS

-((Beta-p)*sB - (R2-V)) is SOS

-((R2-V)*sV + Vdot - w'*w + z'*z/gamma^2) is SOS

with sB and sV SOS

```
rVgCons.p = x'*x;
rVgCons.l1 = 1e-6*x'*x;
```

basis for multipliers

```
rVgCons.sVBasis = monomials([x;w],1:1);
rVgCons.sBBasis = monomials([x],0:0);
```

basis for V

```
rVgCons.VBasis = monomials(x,2:2);
```

Options used in the optimization (there are two types of bisection)

Options for R2 bisection (given V maximize R2)

```
opts.OptBis.R2High = 100;
opts.OptBis.R2Low = 0;
opts.OptBis.R2Tol = 1e-5;
```

Options for beta bisection (given V and R2 maximize beta)

```
opts.OptBis.BetaLow = 0;
opts.OptBis.BetaHigh = 10;
opts.OptBis.BetaTol = 1e-5;
```

Options for outer iterations

```
opts.MaxIter = 30;
opts.StopTol = 1e-4;
```

Call the routine for the robust ROA calculation.

```
[bOut,ROut,VOut,sVOut,sBOut] =...
              roaViaGain(f,h,x,w,gainUnmodeled, rVgCons, opts);
```

```
--Iteration = 1; Beta = 2.237864e+00
--Iteration = 2; Beta = 2.528324e+00
--Iteration = 3; Beta = 2.674789e+00
--Iteration = 4; Beta = 2.766418e+00
--Iteration = 5; Beta = 2.841692e+00
--Iteration = 6; Beta = 2.909317e+00
--Iteration = 7; Beta = 2.969599e+00
--Iteration = 8; Beta = 3.024187e+00
--Iteration = 9; Beta = 3.073797e+00
--Iteration = 10; Beta = 3.119020e+00
--Iteration = 11; Beta = 3.160076e+00
```

```
--Iteration = 12; Beta = 3.198042e+00
--Iteration = 13; Beta = 3.232651e+00
--Iteration = 14; Beta = 3.264246e+00
--Iteration = 15; Beta = 3.293161e+00
--Iteration = 16; Beta = 3.319778e+00
--Iteration = 17; Beta = 3.344307e+00
--Iteration = 18; Beta = 3.366671e+00
--Iteration = 19; Beta = 3.387337e+00
--Iteration = 20; Beta = 3.406572e+00
--Iteration = 21; Beta = 3.424530e+00
--Iteration = 22; Beta = 3.441334e+00
--Iteration = 23; Beta = 3.457098e+00
--Iteration = 24; Beta = 3.471813e+00
--Iteration = 25; Beta = 3.485594e+00
--Iteration = 26; Beta = 3.498507e+00
--Iteration = 27; Beta = 3.510599e+00
--Iteration = 28; Beta = 3.521967e+00
--Iteration = 29; Beta = 3.532410e+00
--Iteration = 30; Beta = 3.542099e+00
```

Conclusion: For any system Phi which is known to be strictly dissipative w.r.t. z^Tz-w^Tw, if the Phi dunamics have zero initial conditions, then, for any x(0) in { x : p(x(0)) \leq bOut}, x(t) stays in { x : V(x) \leq ROut^2} and x(t) --> 0 as t--> infty where

```
bOut
```

```
bOut =

    3.5421
```

```
ROut
```

```
ROut =

    5.3042
```

```
VOut
```

```
VOut =
  0.42274*x1^2
  - 0.30165*x1
  *x2 + 0.72089
  *x1*x3
  - 0.89644*x1
  *x4 + 3.7846
  *x2^2
  - 6.3846*x2
  *x3 - 0.13911
  *x2*x4
  + 5.1233
  *x3^2
  + 2.3659*x3
  *x4 + 3.6094
  *x4^2
```

Published with MATLAB® 7.6

Using the Polysys Class

This is a quick demonstration of the capabilities of the `@polysys` class.

Timothy J. Wheeler
Dept. of Mechanical Engineering
University of California, Berkeley

Contents

Creating a `polysys` object.

Since the `polysys` class is built on the `polynomial` class, we first create some `polynomial` objects to work with:

```
pvar x1 x2 u1 u2
```

The equations of the system are of the form

$$\dot{x} = f(x,u)$$

$$y = g(x,u)$$

Define the `polynomial` objects `f` and `g`

```
mu = -1;
f = [ x2; mu*(1-x1^2)*x2 - x1 ];
g = [x1;x2];
```

The `polynomial` objects `states` and `inputs` specify the ordering of the variables. For example, specifying `states(1)=x1` indicates that `f(1)` is the time derivative of `x1`.

```
states = [x1;x2];
inputs = [];
```

Finally, the `polynomial` objects are used to create a `polysys` object:

```
vdp = polysys(f,g,states,inputs)
```

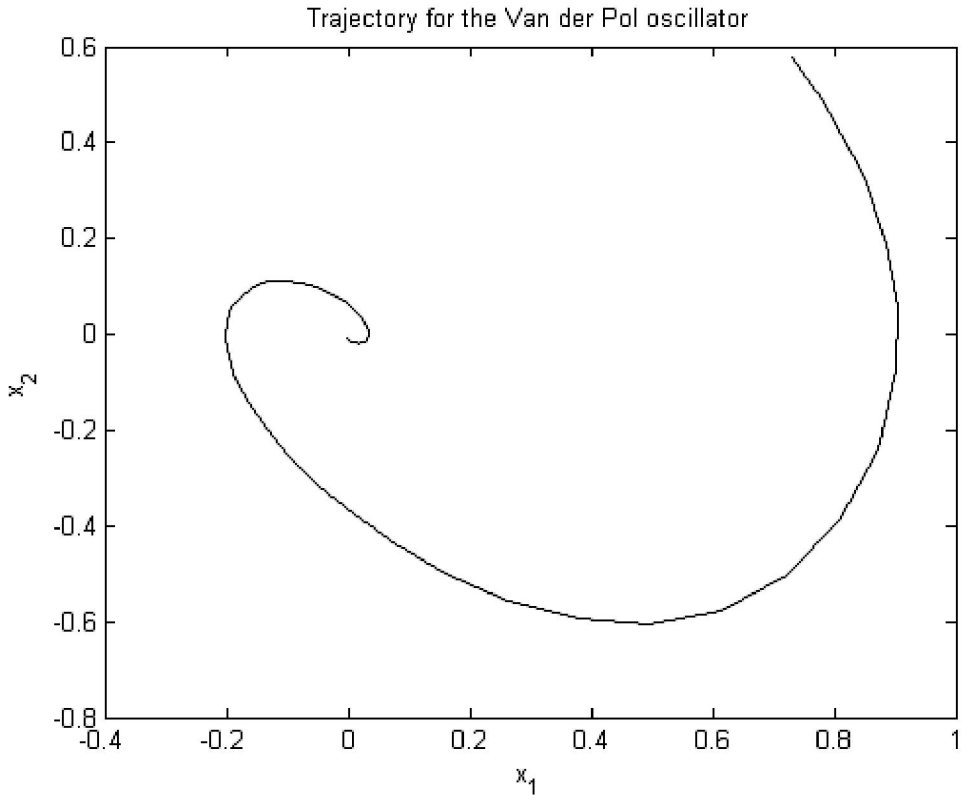
```
Continuous-time polynomial dynamic system.
```

```
States: x1,x2
State transition map is x'=f(x,u) where
  f1 = x2
  f2 = x1^2*x2 - x1 - x2
Output response map is y=g(x,u) where
  g1 = x1
  g2 = x2
```

Simulating the system.

The system is simulated over for a given time interval using the `sim` command. Note that the syntax is similar to `ode45`.

```
T = 10;
x0 = randn(2,1);
[t,x] = sim(vdp,[0,T],x0);

plot(x(:,1),x(:,2),'k-')
xlabel('x_1')
ylabel('x_2')
title('Trajectory for the Van der Pol oscillator')
```



Converting other objects to `polysys` objects.

The simplest object that can be "promoted" to a `polysys` is a `double`.

```
gainsys = polysys(rand(2,2))
```

```
Static polynomial map.
Inputs: u1,u2
Output response map is y=g(x,u) where
  g1 = 0.54722*u1 + 0.14929*u2
  g2 = 0.13862*u1 + 0.25751*u2
```

`LTI` objects can also be converted to `polysys` objects.

```
linearsys = rss(2,2,2);
linearpolysys = polysys(linearsys)
```

```
Continuous-time polynomial dynamic system.
States: x1,x2
Inputs: u1,u2
State transition map is x'=f(x,u) where
  f1 = -1.4751*u1 + 0.11844*u2 - 1.0515*x1 - 0.097639*x2
  f2 = -0.234*u1 + 0.31481*u2 - 0.097639*x1 - 1.9577*x2
Output response map is y=g(x,u) where
  g1 = 1.4435*x1 + 0.62323*x2
  g2 = -0.99209*u1 + 0.79905*x2
```

`Polynomial` objects can also be converted into a "static" `polysys` objects.

```
p = x1^2 - x1*x2;
staticsys = polysys(p)
```

```
Static polynomial map.
Inputs: u1,u2
Output response map is y=g(x,u) where
  g1 = u1^2 - u1*u2
```

Interconnections.

`Polysys` supports most of the same interconnections as the `LTI` class with the same syntax and the same semantics. Here are some examples:

```
append(linearpolysys,staticsys)
```

```
Continuous-time polynomial dynamic system.
States: x1,x2
Inputs: u1,u2,u3,u4
State transition map is x'=f(x,u) where
  f1 = -1.4751*u1 + 0.11844*u2 - 1.0515*x1 - 0.097639*x2
  f2 = -0.234*u1 + 0.31481*u2 - 0.097639*x1 - 1.9577*x2
Output response map is y=g(x,u) where
  g1 = 1.4435*x1 + 0.62323*x2
  g2 = -0.99209*u1 + 0.79905*x2
  g3 = u3^2 - u3*u4
```

```
series(linearpolysys,gainsys)
```

```
Continuous-time polynomial dynamic system.
States: x1,x2
Inputs: u1,u2
State transition map is x'=f(x,u) where
  f1 = -1.4751*u1 + 0.11844*u2 - 1.0515*x1 - 0.097639*x2
  f2 = -0.234*u1 + 0.31481*u2 - 0.097639*x1 - 1.9577*x2
Output response map is y=g(x,u) where
  g1 = -0.14811*u1 + 0.78991*x1 + 0.46034*x2
  g2 = -0.25547*u1 + 0.20011*x1 + 0.29216*x2
```

The methods `append`, `feedback`, `parallel`, and `series` are used to interconnect `polysys` objects.

Discrete-time systems.

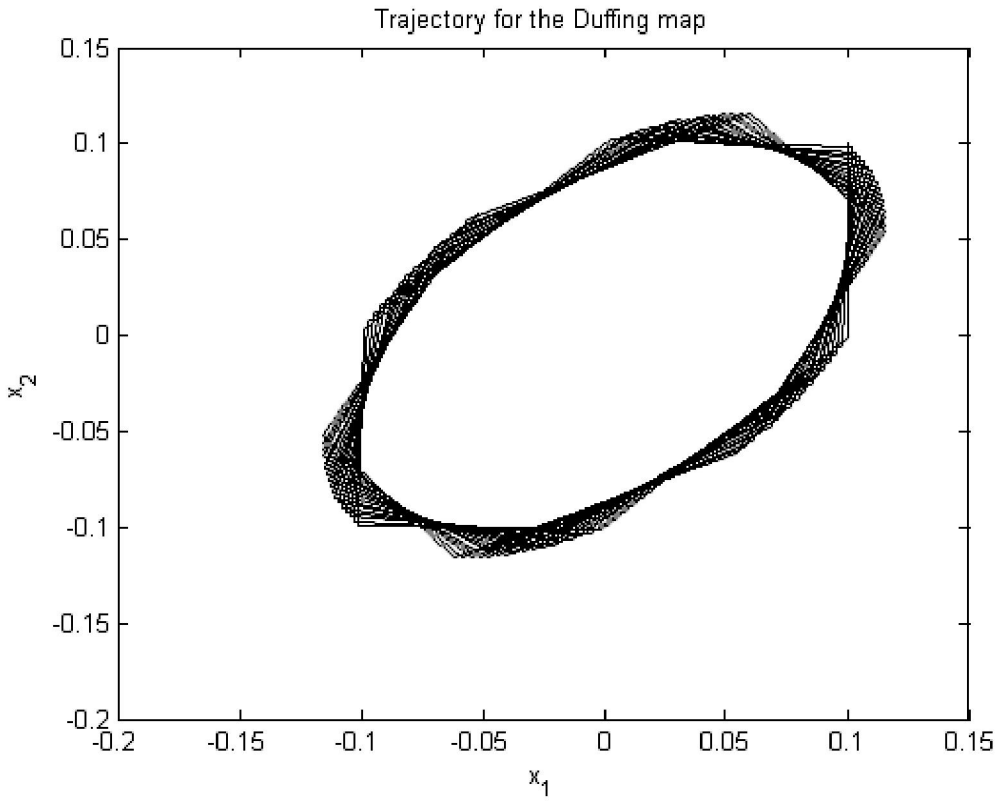
It is also possible to create discrete-time `polysys` objects, as follows:

```
a = 1;
b = 1;
fduff = [ x2; -b*x1 + a*x2 - x2^3 ];
gduff = [ x1; x2 ];

xduff = [ x1; x2];
uduff = [];
Tsample = 1;

duff = polysys(fduff,gduff,xduff,uduff,Tsample)
```

```
Discrete-time polynomial dynamic system.
Sampling time: 1
States: x1,x2
State transition map is x(k+1)=f(x(k),u(k)) where
  f1 = x2
  f2 = -x2^3 - x1 + x2
Output response map is y(k)=g(x(k),u(k)) where
  g1 = x1
  g2 = x2
```

Discrete-time systems are simulated using the command `dsim`. Note that simulation time points are specified as `(0:T)`, rather than `[0,T]`.

```
T = 100;
x0 = [.1;.1];
[t,x] = dsim(duff,(0:T),x0);

plot(x(:,1),x(:,2),'k-')
xlabel('x_1')
ylabel('x_2')
title('Trajectory for the Duffing map')
```



Other Utilities

`Polysys` object can be linearized at a given point. This syntax returns an `SS` object:

```
xe = [1;2];
vdplin = linearize(vdp,xe);
class(vdplin)


ans =
ss
```

This syntax returns the state-space data of the linearization:

```
[A,B,C,D] = linearize(vdp);
```

Check if a `polysys` object is linear.

```
islinear(linearpolysys)


ans =
     1


islinear(vdp)
```

```
ans =
     0
```

Subsystems are referenced using the same syntax as `LTI` objects:

```
linearpolysys(1,1)
```

```
Continuous-time polynomial dynamic system.
States: x1,x2
Inputs: u1
State transition map is x'=f(x,u) where
  f1 = -1.4751*u1 - 1.0515*x1 - 0.097639*x2
  f2 = -0.234*u1 - 0.097639*x1 - 1.9577*x2
Output response map is y=g(x,u) where
  g1 = 1.4435*x1 + 0.62323*x2
```

We can also get function handles to the system's state transition and output response maps. This is mostly used to build simulation routines that require handles to functions with a certain syntax (i.e., `ode45`).

```
[F,G] = function_handle(vdp);

xeval = randn(2,1);
ueval = []; % VDP is autonomous
teval = []; % The time input is just for compatibility with ode solvers
xdot = F(teval,xeval,ueval)
```

```
xdot =
   -0.7420
    0.9962
```

We can multiply `polysys` objects by scalars or matrices.

```
M = diag([2,3]);
M*vdp
```

```
Continuous-time polynomial dynamic system.
States: x1,x2
State transition map is x'=f(x,u) where
  f1 = x2
  f2 = x1^2*x2 - x1 - x2
Output response map is y=g(x,u) where
  g1 = 2*x1
  g2 = 3*x2
```

```
12*vdp
```

```
Continuous-time polynomial dynamic system.
States: x1,x2
```

```
State transition map is x'=f(x,u) where
  f1 = x2
  f2 = x1^2*x2 - x1 - x2
Output response map is y=g(x,u) where
  g1 = 12*x1
  g2 = 12*x2


linearpolysys*M


Continuous-time polynomial dynamic system.
States: x1,x2
Inputs: u1,u2
State transition map is x'=f(x,u) where
  f1 = -2.9503*u1 + 0.35533*u2 - 1.0515*x1 - 0.097639*x2
  f2 = -0.46801*u1 + 0.94443*u2 - 0.097639*x1 - 1.9577*x2
Output response map is y=g(x,u) where
  g1 = 1.4435*x1 + 0.62323*x2
  g2 = -1.9842*u1 + 0.79905*x2
```

Published with MATLAB® 7.6

Using the Worstcase Solver - Demo 1

The `worstcase` solver is used to find the induced L2-to-L2 gain of a four-state nonlinear system.

Timothy J. Wheeler
Dept. of Mechanical Engineering
University of California, Berkeley

Contents

Introduction.

Consider a dynamic system of the form

$$\dot{x} = f(x,u)$$

$$y = g(x,u),$$

where `x(0)=0`. Given positive scalars `B` and `T`, the goal is to maximize

$$\|y\|_{2,T} := \int_0^T \|y(t)\|_2 \, dt$$

subject to the constraint

$$\|u\|_{2,T} := \int_0^T \|u(t)\|_2 \, dt \leq B.$$

Note: we only consider inputs and outputs defined on the interval `[0,T]`.

System parameters.

This system is parameterized by the following constants:

```
lam = 1;
```

```
PL = 1;
gammaX = 1;
gammaR = 1;
A = 0.8;
tau = 1;
K0x = (-1/tau - A)/lam;
K0r = (1/tau)/lam;
```

Create a model of the system.

First, `polynomial` variables are created using the `pvar` command. Then, these variables are used to define the functions `f` and `g`, which are also `polynomial` variables.

```
pvar x1 xm zx zr r w
states = [x1;xm;zx;zr];
inputs = [r;w];

f(1,1) = A*x1 + lam*((zx+K0x)*x1 + (zr+K0r)*r) + w;
f(2,1) = (1/tau)*(-xm+r);
f(3,1) = -gammaX*x1*(x1-xm)*PL;
f(4,1) = -gammaR*r*(x1-xm)*PL;

g = ((zx+K0x)*x1 + (zr+K0r)*r) + w;
```

Then, a `polysys` object is created from the polynomials `f` and `g`.

```
sys = polysys(f,g,states,inputs);
```

The polynomial objects `states` and `inputs` specify the ordering of the variables. That is, by setting `states(1) = x1`, we specify that `f(1)` is the time derivative of `x1`.

Optimization parameters.

Use the following values for the optimization parameters (defined above):

```
T = 10;
B = 3;
```

The time vector t specifies the time window (T = t(end)) and the points at which the system trajectory is computed.

```
t = linspace(0,T,100)';
```

Set options for worstcase solver.

Create a `wcoptions` object that contains the default options.

```
opt = wcoptions();
```

Specify the maximum number of iterations and which ODE solver to use.

```
opt.MaxIter = 50;
opt.ODESolver = 'ode45';
```

Tell the solver to display a text summary of each iteration.

```
opt.PlotProgress = 'none';
```

Specify the optimization objective, and the bound on the input.

```
opt.Objective = 'L2';
opt.InputL2Norm = B;
```

Find worstcase input.

```
[tOut,x,y,u,eNorm] = worstcase(sys,t,opt);
```

Simulate with worstcase input.

We can only compute the worstcase input over a finite interval of time [0,T]. However, any response of the system that occurs after the input is "shut off" (i.e., u(t) = 0 for t > T) should contribute to our objective. Hence, we compute a more accurate value of the objective by continuing the simulation from the end of the previous trajectory with no input:

```
[te,xe,ye] = sim(sys,tOut,x(end,:)');
td = [tOut;tOut(2:end)+max(tOut)];
yd = [y;ye(2:end)];
```

The objective value over [0,T] is

```
eNorm
```

```
eNorm =
    4.7436
```

The objective value over [0,2T] is

```
eNormd = get2norm(yd,td)
```

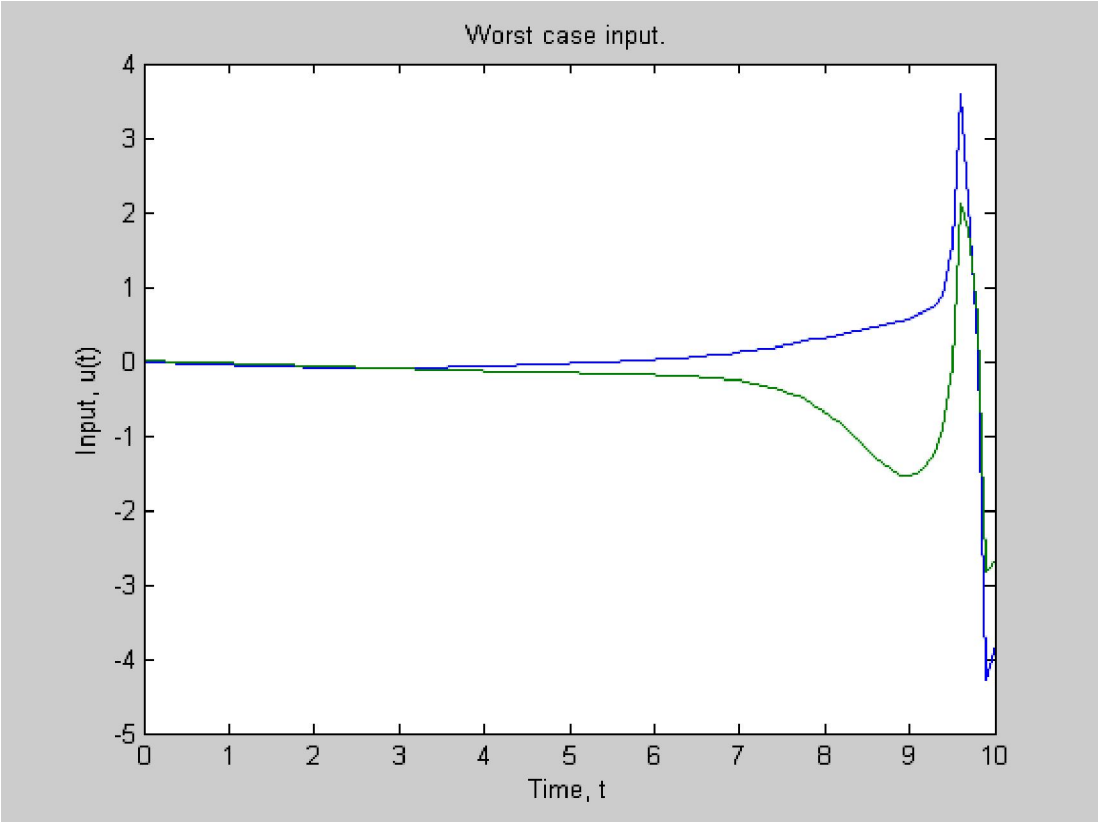
```
eNormd =
    4.9622
```

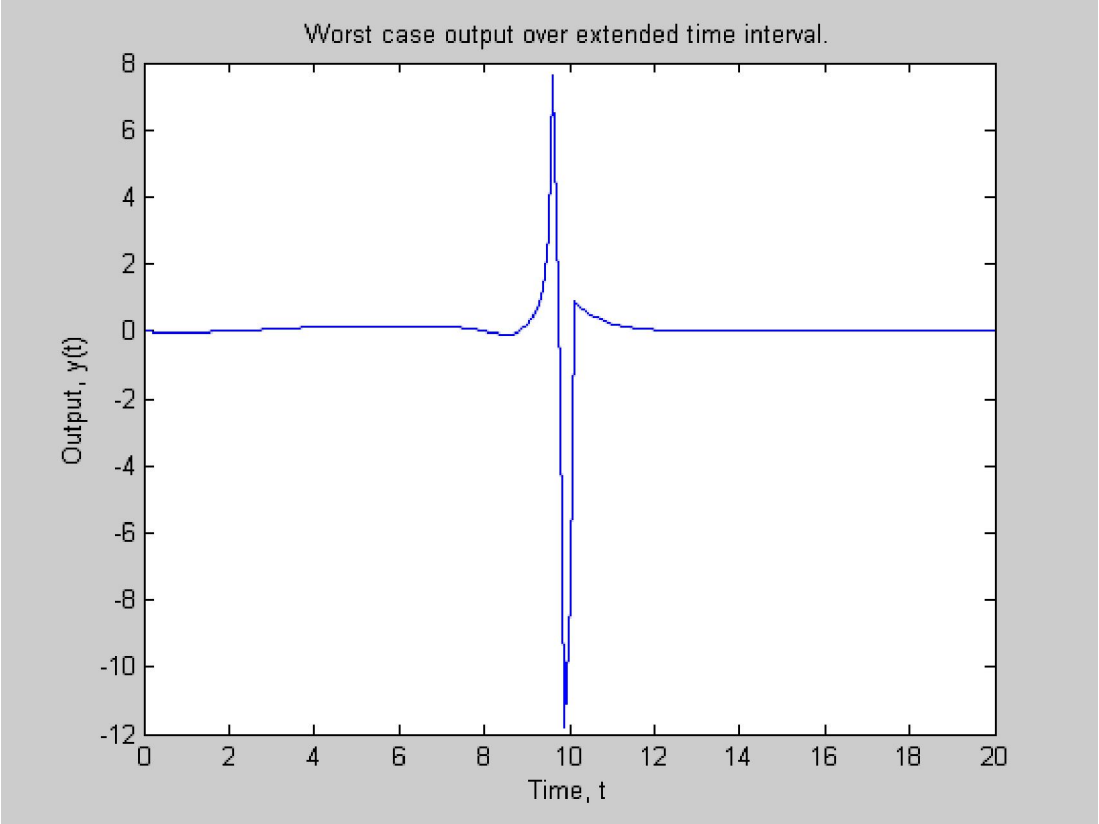
Display results.

```
fprintf( 'The L2-to-L2 gain is %f\n', eNormd/B );
```

```
figure;
plot(tOut,u)
```

```
xlabel('Time, t')
ylabel('Input, u(t)')
title('Worst case input.')

figure;
plot(td,yd)
xlabel('Time, t')
ylabel('Output, y(t)')
title('Worst case output over extended time interval.')


The L2-to-L2 gain is 1.654050
```





Specifying a starting point.

By default, the worstcase solver starts with a constant input and then searches for a better input. Since this problem is nonconvex, this search may get "stuck" at a local optimum. We can help the solver by specifying a sensible starting point that is known to exhibit a large output.

```
load demo1_badInput
u0 = B * ubad/get2norm(ubad,tbad);
opt.InitialInput = u0;
```

Run solver again.

```
[tOut,x,y,u,eNorm] = worstcase(sys,t,opt);
```

Extend this simulation.

```
[te,xe,ye] = sim(sys,tOut,x(end,:)');
td = [tOut;tOut(2:end)+max(tOut)];
yd = [y;ye(2:end)];
```

The objective value over [0,T] is

```
eNorm
```

```
eNorm =
```

```
    5.0020
```

The objective value over [0,2T] is

```
eNormd = get2norm(yd,td)
```

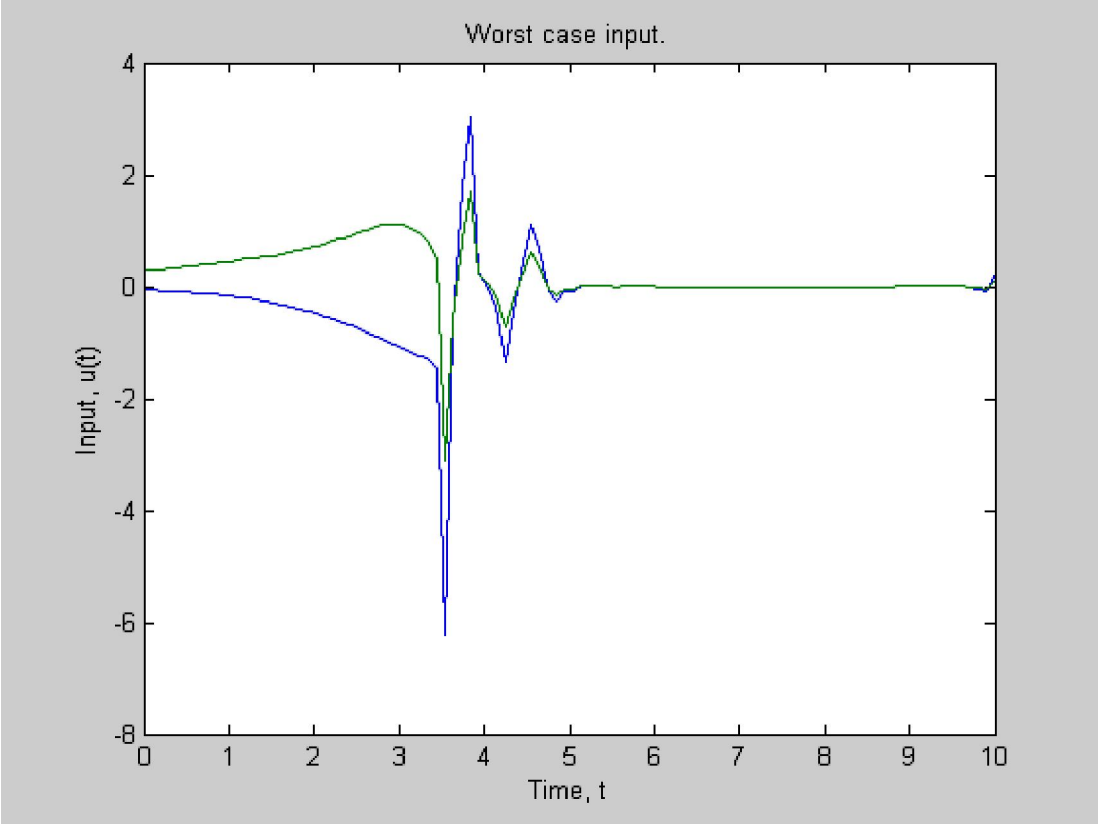
```
eNormd =
    5.0029
```

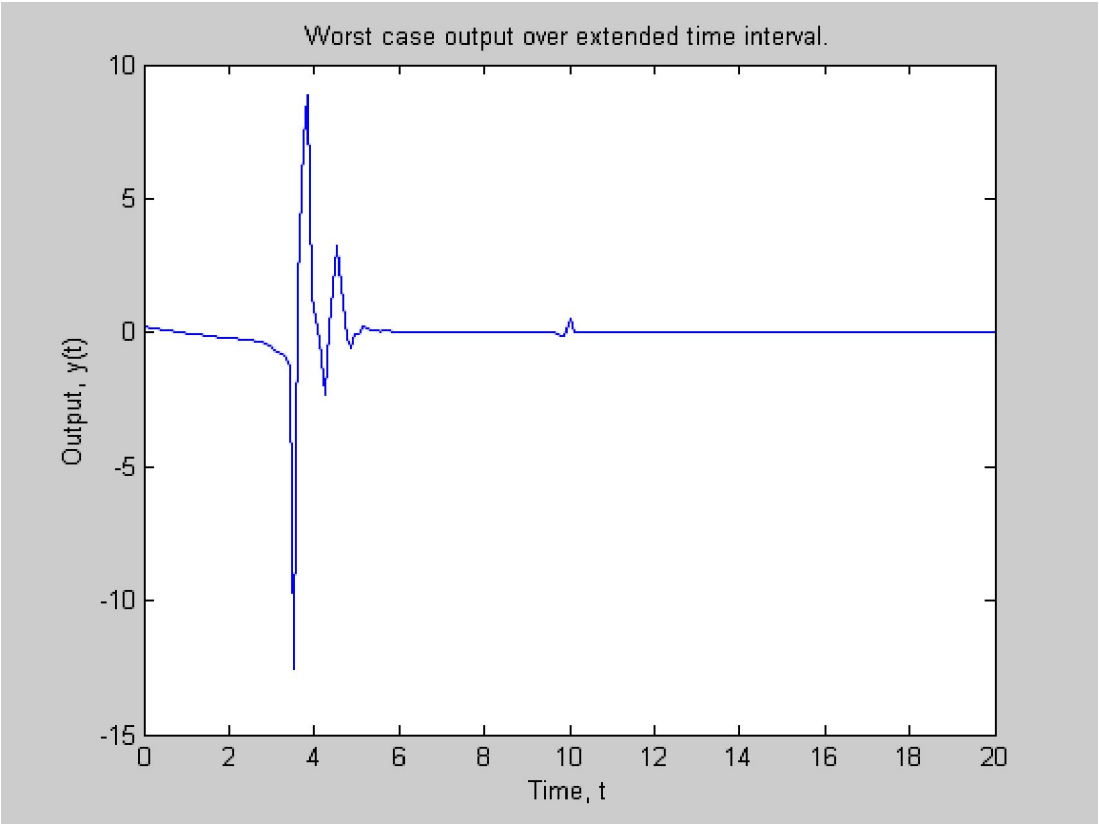
Note that we achieve a larger value of the objective when we start the solver at u0.

Display new results.

```
fprintf( 'The L2-to-L2 gain is %f\n', eNormd/B );

figure;
plot(tOut,u)
xlabel('Time, t')
ylabel('Input, u(t)')
title('Worst case input.')

figure;
plot(td,yd)
xlabel('Time, t')
ylabel('Output, y(t)')
title('Worst case output over extended time interval.')
```

```
The L2-to-L2 gain is 1.667635
```





Published with MATLAB® 7.6

Using the Worstcase Solver - Demo 2

Timothy J. Wheeler
Dept. of Mechanical Engineering
University of California, Berkeley

Contents

Introduction

Consider a dynamic system of the form

$$\dot{x} = f(x,u)$$

$$y = g(x,u),$$

where x(0)=0. Given positive scalars B and T and a positive definite matrix C, the goal is to maximize

$$x(T)'Cx(T),$$

subject to the constraints

$$\|u\|_{2,T} := \int_0^T \|u(t)\|_2 \, dt \leq B.$$

Of course, since we are only interested in the value of `x` at time `T`, we only need to consider inputs defined on the interval `[0,T]`.

Create a model of the system.

First, `polynomial` variables are created using the `pvar` command. Then, these variables are used to define the functions `f` and `g`, which are also `polynomial` variables.

```
pvar x1 x2 u
states = [x1;x2];
inputs = u;
f = [ -x1 + x2 - x1*x2^2 ; -x2*x1^2 - x2 + u ];
g = states;
```

Then, a `polysys` object is created from the polynomials `f` and `g`.

```
sys = polysys(f,g,states,inputs);
```

The `polynomial` objects `states` and `inputs` specify the ordering of the variables. That is, by setting `states(1) = x1`, we specify that `f(1)` is the time derivative of `x1`.

Optimization parameters.

Use the following values for the optimization parameters (defined above):

```
T = 10;
B = 1;
C = eye(2);
```

The time vector `t` specifies the time window (`T=t(end)`) and the points at which the system trajectory is computed.

```
t = linspace(0,T,1000)';
```

Set options for worstcase solver.

Create a `@wcoptions` object that contains the default options.

```
opt = wcoptions();
```

Specify the maximum number of iterations and tell the solver to not display any information while solving.

```
opt.MaxIter = 500;
opt.PlotProgress = 'none';
```

Specify the desired cost function.

```
opt.Objective = 'Final';
opt.FinalCostMatrix = C;
```

Specify the bound on the input.

```
opt.InputL2Norm = B;
```

Find worst input.

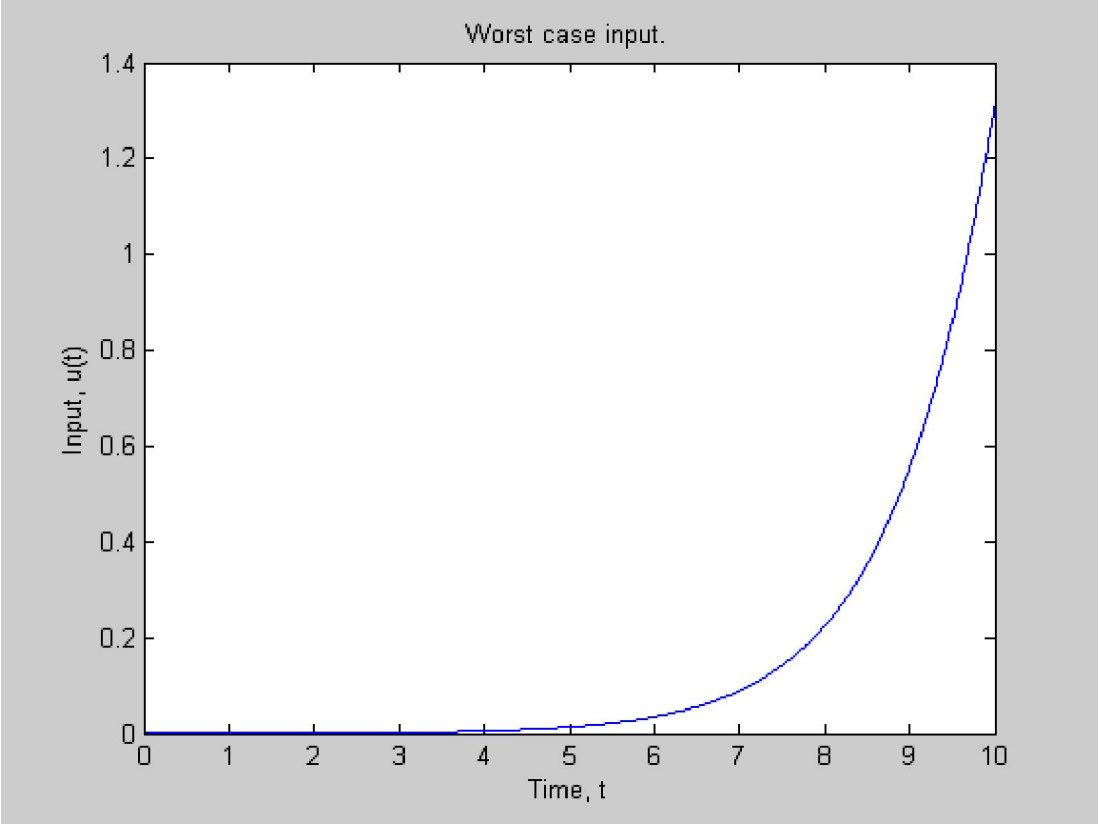
```
[tOut,x,y,u] = worstcase(sys,t,opt);
```

Display results.

```
cost = x(end,:)*C*x(end,:)';
fprintf( '||u|| = %0.4f, cost = %0.4f\n', B, cost );
```

```
figure;
plot(tOut,u)
xlabel('Time, t')
ylabel('Input, u(t)')
title('Worst case input.')
```

```
||u|| = 1.0000, cost = 0.5727
```



Published with MATLAB® 7.6

Local Gain Analysis of Nonlinear Systems

This demo illustrates how to obtain upper and lower bounds on the gain of a nonlinear system. Lower bounds are calculated using the linearized system and an iterative method. Upper bounds are calculated using SOS methods.

Contents


```
format('compact')
```

Procedure

1. Given a nonlinear system f and a cost function p, compute the reachability gain through the linearization

2. Find the worst-case input for the linearized dynamics by inputing an InputL2Norm = 1 into the WORSTCASE analysis function.

3. Simulate the nonlinear system with this worst input from the linearized analysis and plot the gain.

4. Find the worst-case input for the nonlinear dyanmics using WORSTCASE analysis to get a lower bound on the gain

5. Use REACH to estimate the upper bound on the gain

6. Use REACHREFINE to refine the upper bound obtained from Reach.m

Setup

Simulation Parameters

```
FS=14;
N_R_samples = 15;
N_beta_samples = 15;
```

```
norm_w = sqrt(linspace(1,30,N_R_samples));
beta = linspace(1,70,N_beta_samples);
```

Set up system dynamics

```
pvar x1 x2 w
x = [x1;x2];
f = [ -x1 + x2 - x1*x2^2 ; -x2*x1^2 - x2 + w ];
g = x;
```

Create POLYSYS object

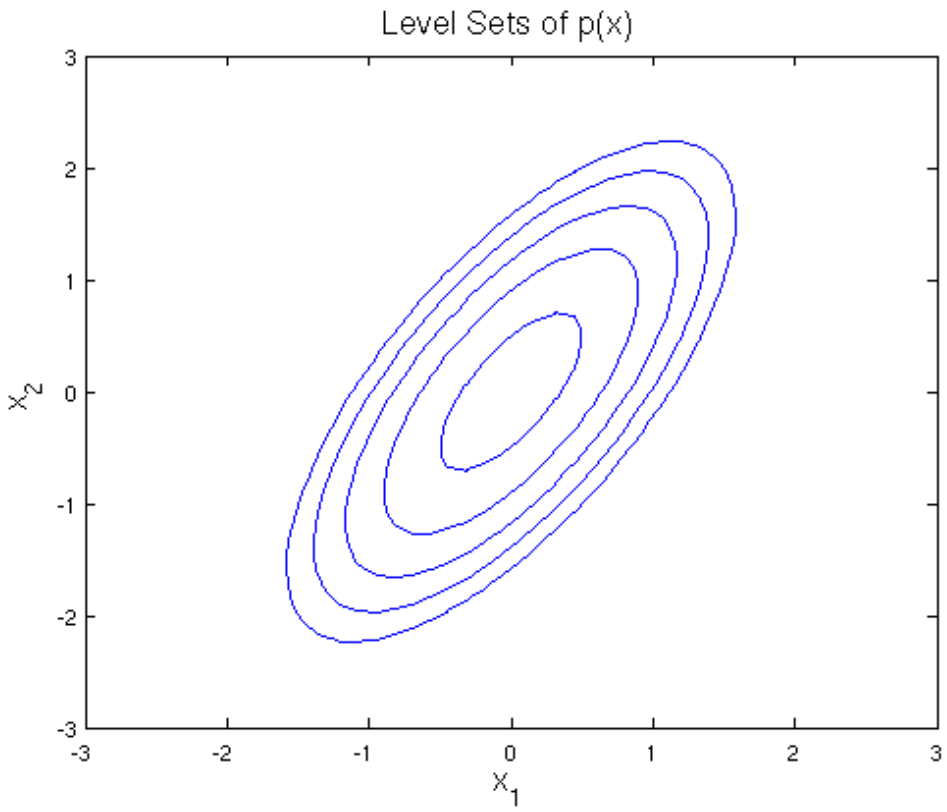
```
sys = polysys(f,g,x,w);
```

Create Quadratic final state cost p(x)

```
Q = sqrtm([8 -4; -4 4]);
p = x'*Q*Q'*x;
```

Plot Contours of p(x)

```
figure;
contour_values = linspace(1, 10, 5);
domain = [-4 4 -4 4];
[C,h] = pcontour(p ,contour_values, domain);
axis([-3 3 -3 3])
title('Level Sets of p(x)', 'FontSize', FS)
xlabel('x_1','FontSize', FS)
ylabel('x_2', 'FontSize', FS)
```



1. Extract Linearization and Exact Reachability Gain

Linearize the system

```
[A,B,C,D] = linearize(sys);
```

Convert linearization to POLYSYS

```
lin_sys = polysys(A*x+B*w, x, x, w);
```

Compute gain

```
X = lyap(A,B*B');
ExactReachabilityGain = max(eig(Q*X*Q'));
```

2. Find worst-case input for Linear Dynamics

Exact solution easily obtained with matrix exponential, but here we use WORSTCASE instead. By setting opt.InputL2Norm = 1, the cost_lin variable should be 1.

time interval

```
T = 10;
t = linspace(0,T,1000)';
```

Create wcoptions object for WORSTCASE and set options

```
opt = wcoptions();
opt.MaxIter = 10;
opt.PlotProgress = 'none';
```

Define quadratic cost function

```
opt.Objective = 'Final';
opt.FinalCostMatrix = Q'*Q;
```

Set Bound on L2 norm of input, other norms obtained by scaling

```
norm_w_lin = 1;
opt.InputL2Norm = norm_w_lin;
```

Run WORSTCASE solver to get worst-case input U_wcLinear

```
[tOut,X_wc,Y_wc,U_wcLinear] = worstcase(lin_sys,t,opt);
```

We expect the cost to be qpproximately equal to the ExactReachabilityGain

```
cost_lin = X_wc(end,:)*opt.FinalCostMatrix*X_wc(end,:)'
ExactReachabilityGain
```

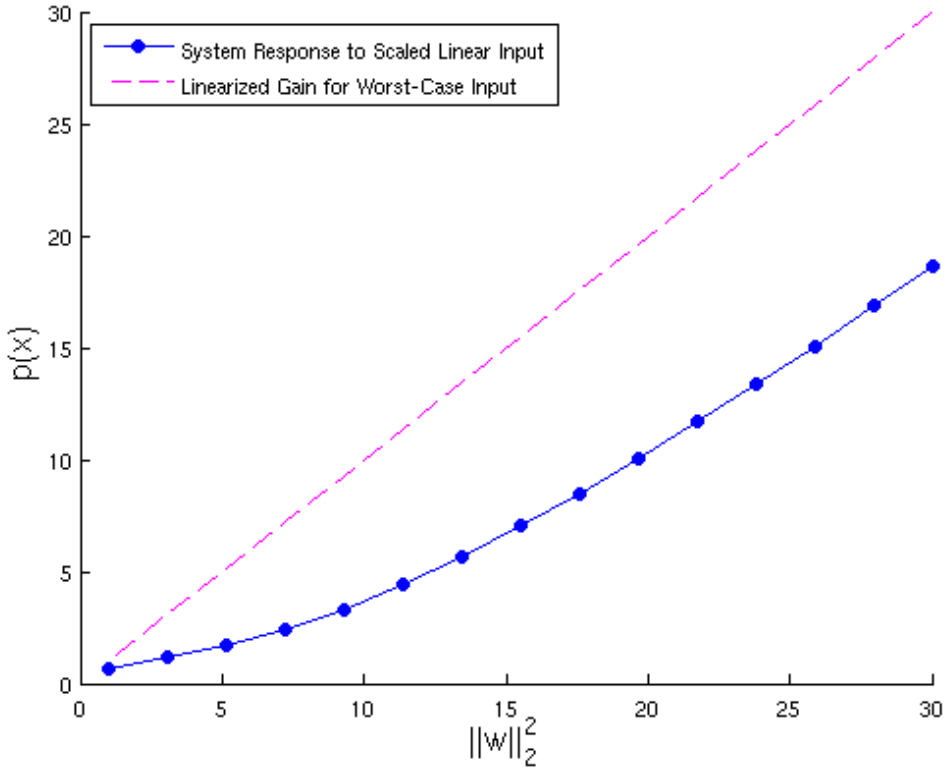
```
cost_lin =
    1.0001
ExactReachabilityGain =
    1.0000
```

3. Scaled linear worst-case inputs applied to nonlinear system

Simulate the nonlinear system with worst-case input from step 2. Note: the cost function does not scale well with the input.

```
cost_NL = zeros(N_R_samples,1);
for i=1:N_R_samples
    scaledInput = [tOut norm_w(i)*U_wcLinear];
    [T,X] = sim(sys,[0 10],zeros(2,1),scaledInput);
    cost_NL(i) = X(end,:)*opt.FinalCostMatrix*X(end,:)';
```

```
end
figure;
hold on;
plot(norm_w.^2,cost_NL,'-ob', 'MarkerFaceColor','b');
plot(norm_w.^2,(ExactReachabilityGain*norm_w).^2,'--m')
legend('System Response to Scaled Linear Input', ...
    'Linearized Gain for Worst-Case Input','Location', 'NorthWest')
xlabel('||w||_2^2', 'FontSize', FS);
ylabel('p(x)', 'FontSize', FS);
```



4. Find worst-case input for nonlinear dynamics

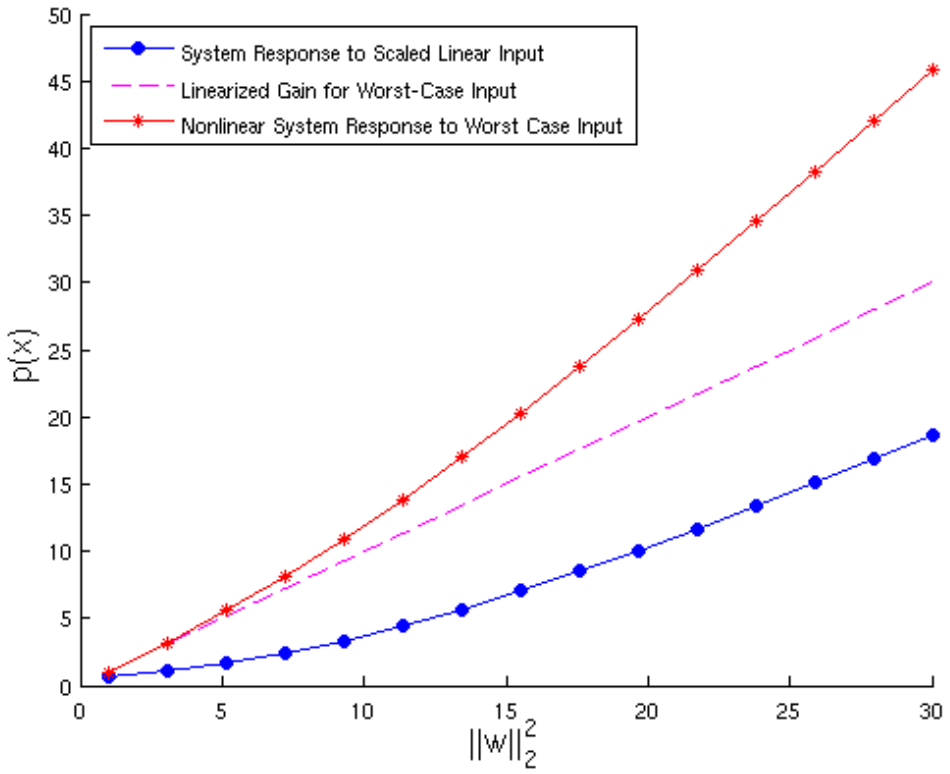
Use WORSTCASE to obtain larger values of cost function.

```
cost_nl = zeros(N_R_samples,1);

for i=1:N_R_samples
    opt.InputL2Norm = norm_w(i);
    [tOut,X_wc,Y_wc,U_wc] = worstcase(sys,t,opt);
    cost_nl(i) = X_wc(end,:)*opt.FinalCostMatrix*X_wc(end,:)';
end
```

Plot worst-case input for nonlinear dynamics

```
figure;
```

```
hold on;
plot(norm_w.^2,cost_NL,'-ob', 'MarkerFaceColor','b');
plot(norm_w.^2,(ExactReachabilityGain*norm_w).^2,'--m')
plot(norm_w.^2,cost_nl,'-r*', 'MarkerFaceColor','r')
legend('System Response to Scaled Linear Input',...
    'Linearized Gain for Worst-Case Input',...
    'Nonlinear System Response to Worst Case Input',...
    'Location', 'NorthWest')
xlabel('||w||_2^2', 'FontSize', FS);
ylabel('p(x)', 'FontSize', FS);
```



5. Find Upper Bound Using REACH

Options for REACH solver

```
Opt.R2High = 40;
Opt.R2Low = 0;
Opt.BisTol = 1e-3;
Opt.MaxIter = 50;
Opt.StopTol = 1e-3;
Opt.display = 0;

reachCons.s4Basis = monomials(x,0:0);
reachCons.s10Basis = monomials([x;w],1:1);
reachCons.VBasis = monomials(x,2:2);
reachCons.l1 = 1e-6*x'*x;
```

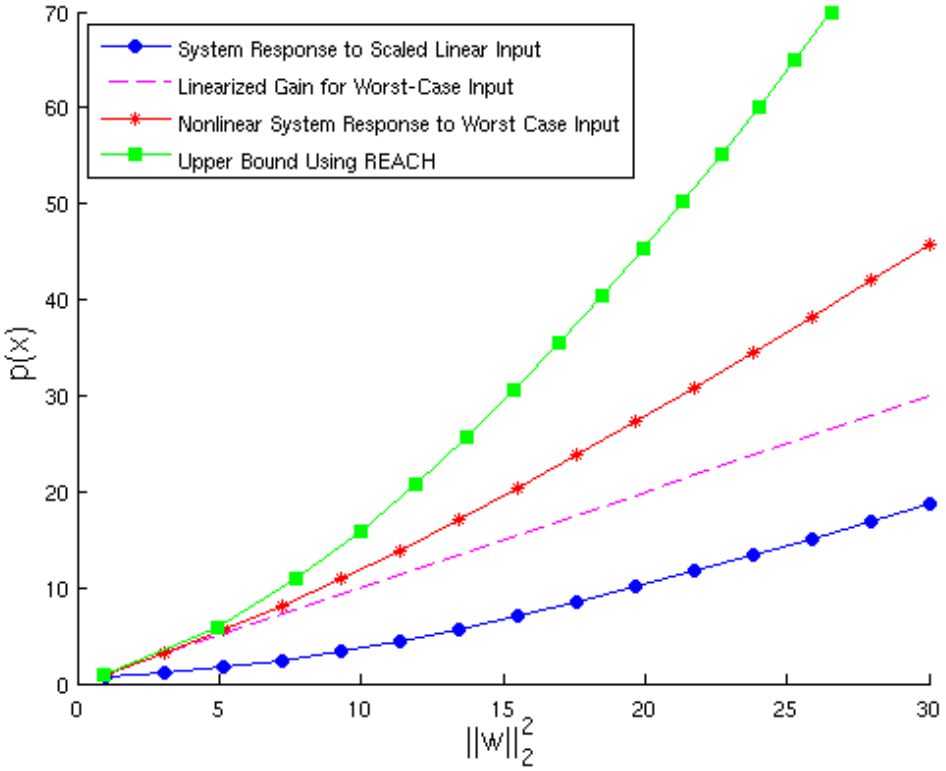
```
Rvec = zeros(1,N_beta_samples);
Vcell = cell(1, N_beta_samples);
```

Run solver for each value of beta

```
for i=1:N_beta_samples
    [Rvec(i), Vcell{i}, s4, s10] = Reach(f,x,w, p, beta(i), reachCons, Opt);
end
```

Plot Upper Bound from Reach.m

```
figure;
hold on;
plot(norm_w.^2,cost_NL,'-ob', 'MarkerFaceColor','b');
plot(norm_w.^2,(ExactReachabilityGain*norm_w).^2,'--m')
plot(norm_w.^2,cost_nl,'-r*', 'MarkerFaceColor','r')
plot(Rvec.^2, beta, '-gs', 'MarkerFaceColor','g')
legend('System Response to Scaled Linear Input',...
    'Linearized Gain for Worst-Case Input',...
    'Nonlinear System Response to Worst Case Input',...
    'Upper Bound Using REACH','Location', 'NorthWest')
xlabel('||w||_2^2', 'FontSize', FS);
ylabel('p(x)', 'FontSize', FS);
```



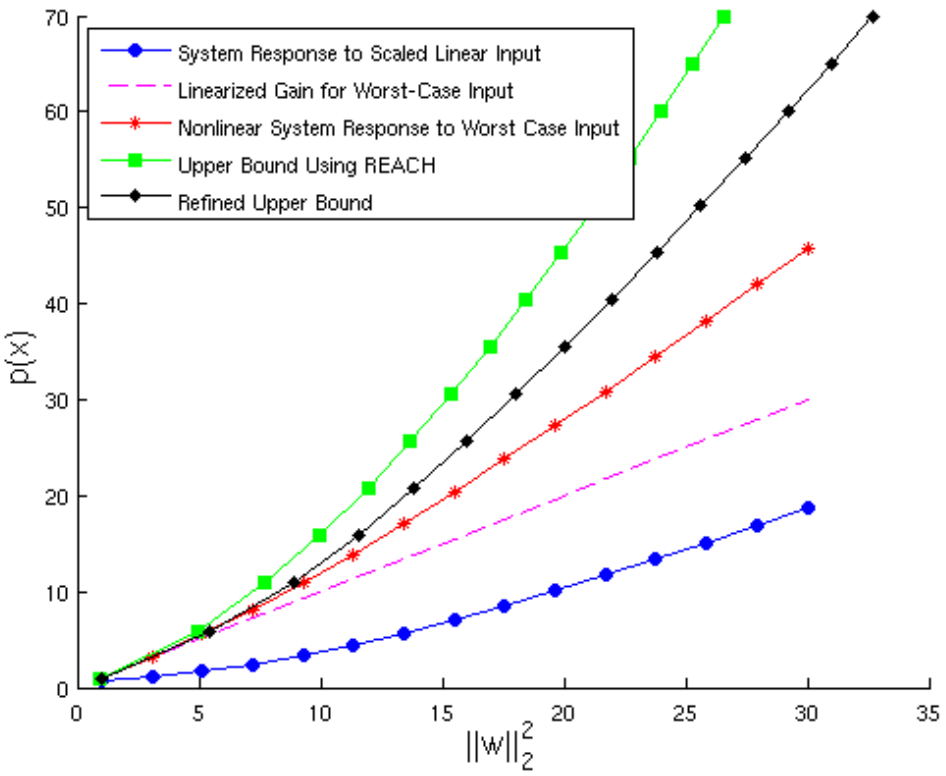
6. Refinement of Upper Bound

Use Refinement to improve upper bounds

```
NumberAnnuli = 20;
hk_sol = zeros(NumberAnnuli,N_beta_samples);
RRefine = zeros(1, N_beta_samples);

for i=1:N_beta_samples
    [hk_sol(:,i), RRefine(i)] = reachRefine(f,x,w,[Vcell{i}],Rvec(i),NumberAnnuli);
end
```

Plot Refinement

```
figure;
hold on;
plot(norm_w.^2,cost_NL,'-ob', 'MarkerFaceColor','b');
plot(norm_w.^2,(ExactReachabilityGain*norm_w).^2,'--m')
plot(norm_w.^2,cost_nl,'-r*', 'MarkerFaceColor','r')
plot(Rvec.^2, beta, '-gs', 'MarkerFaceColor','g')
plot(RRefine.^2, beta, '-dk', 'MarkerFaceColor', 'k')
legend('System Response to Scaled Linear Input',...
    'Linearized Gain for Worst-Case Input',...
    'Nonlinear System Response to Worst Case Input',...
    'Upper Bound Using REACH','Refined Upper Bound',...
    'Location', 'NorthWest')
xlabel('||w||_2^2', 'FontSize', FS);
ylabel('p(x)', 'FontSize', FS);
```



Published with MATLAB® 7.8