International Graduate School on Control, Stuttgart, May 2024

Dissipation Inequalities and Quadratic Constraints for Control, Optimization, and Learning

Lesson 4: Numerical Methods

Murat Arcak[1] and Peter Seiler[2]

[1] University of California, Berkeley

[2] University of Michigan, Ann Arbor

Learning Objectives

In this lesson we will

- Briefly review theory of convex optimization with a focus on the special class of semidefinite programs (SDPs).
- Discuss software tools for solving SDPs and related computational issues that arise when using these tools.
- Present numerical examples that that demonstrate the use of dissipation inequalities and QCs/IQCs.

Outline

1. Brief review of convex optimization and SDPs
2. Computational issues
3. Numerical examples

Finite-Dimensional Optimization

We will consider a finite-dimensional optimization of the form:

$$p^* = \inf_{x \in \mathcal{S} \subset \mathbb{R}^v} f_0(x)$$

where:

- $x \in \mathbb{R}^v$ is the vector of optimization (decision) variables.
- $f_0 : \mathbb{R}^v \to \mathbb{R}$ is the objective function
- S is the set of feasible decision variables. This is often described by a collection of (possibly nonlinear) inequality and equality constraints.

Definition: $x^* \in \mathbb{R}^v$ is <u>(globally) optimal</u> if $x^* \in S$ and $f_0(x^*) \leq f_0(x)$ for any other $x \in S$. The optimal cost is $p^* = f_0(x^*)$.

Definition: $\bar{x} \in \mathbb{R}^v$ is <u>locally optimal</u> if for some $R > 0$,

$$f(\bar{x}) = \inf_{\substack{x \in \mathcal{S} \subset \mathbb{R}^v \\ \|x - \bar{x}\| \leq R}} f_0(x)$$

Finite-Dimensional Optimization

We will consider a finite-dimensional optimization of the form:

$$p^* = \inf_{x \in \mathcal{S} \subset \mathbb{R}^v} f_0(x)$$

where:

- $x \in \mathbb{R}^v$ is the vector of optimization (decision) variables.
- $f_0 : \mathbb{R}^v \to \mathbb{R}$ is the objective function
- S is the set of feasible decision variables. This is often described by a collection of (possibly nonlinear) inequality and equality constraints.

Comments:

- We define $p^* = +\infty$ if there are no feasible values ($S = \emptyset$).
- An optimization can have zero, one, or many optimal points.
- We will show that our DI+IQC conditions can be formulated as a special type of (convex) optimization: a semidefinite program.

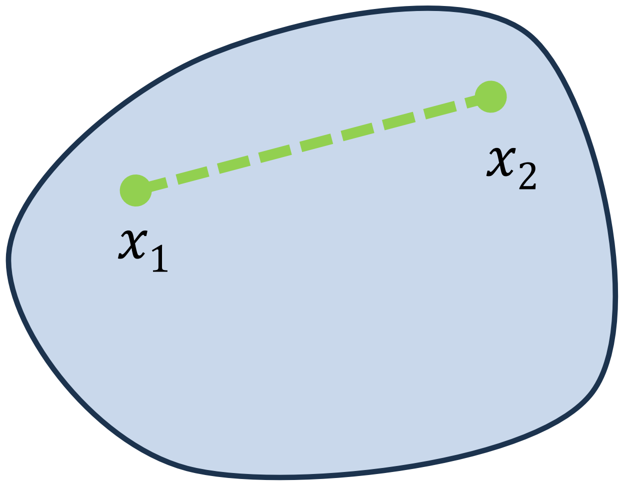
Convex Sets

Definition: A set $S \subseteq \mathbb{R}^v$ is convex if

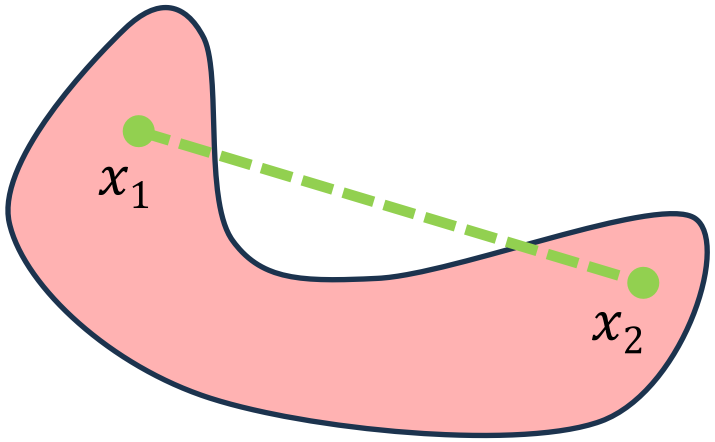
$$\lambda x_1 + (1-\lambda)x_2 \in S$$

holds for all $x_1, x_2 \in S$ and all $\lambda \in [0,1]$.

In other words, a convex set is one that contains the line segment that connects any two points in the set.



Convex



Not Convex

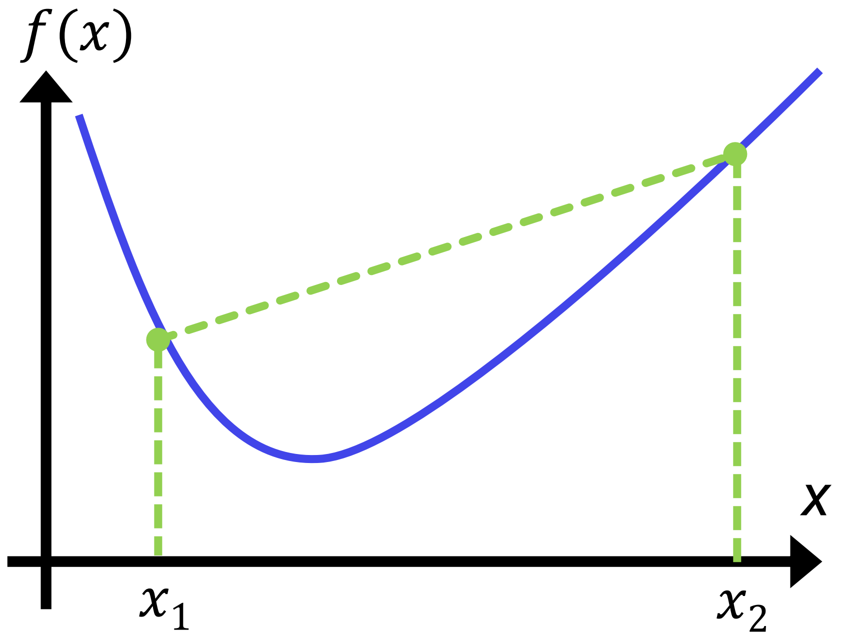
Convex Functions

Definition: A function $f\colon \mathbb{R}^v \to \mathbb{R}$ is convex if

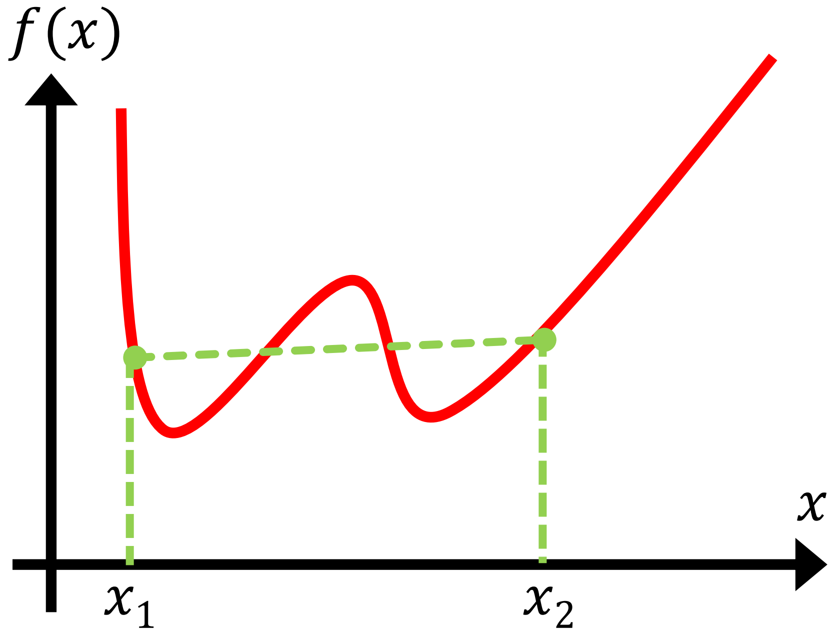
$$f\left(\lambda x_1 + (1-\lambda)x_2\right) \leq \lambda\, f(x_1) + (1-\lambda)\, f(x_2)$$

holds for all $x_1, x_2 \in \mathbb{R}^v$ and all $\lambda \in [0,1]$.

f is concave if -f is convex.



Convex



Not Convex

Convex Functions

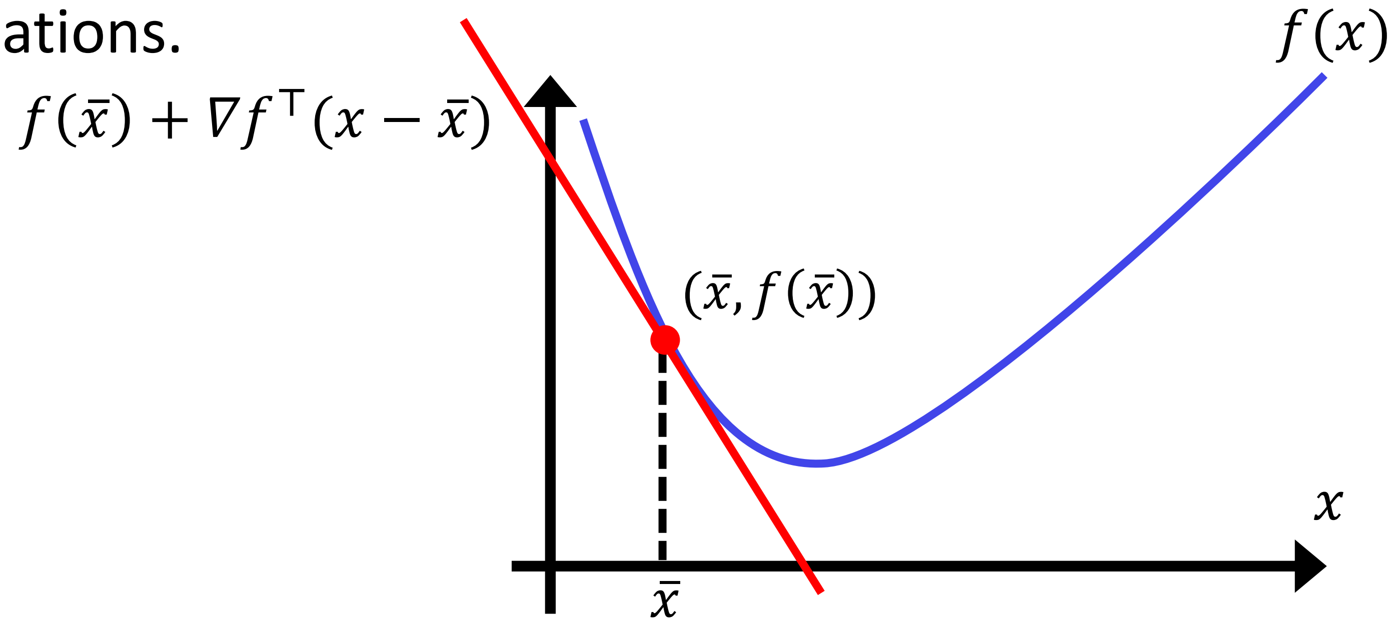
There are many conditions to verify convexity of a function $f: \mathbb{R}^v \to \mathbb{R}$ including:

1. *First-order condition:* The function f is differentiable and

$$f(x) \geq f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x}) \; \forall x, \, \bar{x} \in \mathbb{R}^v$$

where $\nabla f(\bar{x}) = \left[\frac{\partial f}{\partial x_1}(\bar{x}) \quad \dots \quad \frac{\partial f}{\partial x_v}(\bar{x})\right]^\top$.

Convex functions are lower bounded by first-order (linear) approximations.



Convex Functions

There are many conditions to verify convexity of a function $f : \mathbb{R}^v \to \mathbb{R}$ including:

1. *First-order condition:* The function f is differentiable and

$$f(x) \geq f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x}) \ \forall x,\ \bar{x} \in \mathbb{R}^v$$

2. *Second-order condition:* The function f is twice differentiable and the Hessian $\nabla^2 f(x)$ is an $v \times v$ positive semidefinite matrix for all $x \in \mathbb{R}^v$. The (i, j) entry of the Hessian is $\frac{\partial f^2}{\partial x_i \partial x_j}(x)$.

Convex Functions

There are many conditions to verify convexity of a function $f: \mathbb{R}^v \to \mathbb{R}$ including:

1. *First-order condition:* The function f is differentiable and

$$f(x) \geq f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x}) \ \forall x,\ \bar{x} \in \mathbb{R}^v$$

2. *Second-order condition:* The function f is twice differentiable and the Hessian $\nabla^2 f(x)$ is an $v \times v$ positive semidefinite matrix for all $x \in \mathbb{R}^v$. The (i, j) entry of the Hessian is $\frac{\partial f^2}{\partial x_i \partial x_j}(x)$.

3. *1D Restriction:* The function f is convex when restricted to any line: if $g: \mathbb{R} \to \mathbb{R}$ is defined by $g(t) \coloneqq f(x_1 + t x_2)$ for any $x_1, x_2 \in \mathbb{R}^v$ then g is a convex function of t.

There are many other similar conditions, e.g. functions with a restricted domain, conditions for strict convexity, etc.

Convex Sets and Functions

Wake-up Problems

1) Which of the following functions is convex?

a) ReLU, $w = f(v) := \max(v, 0)$

b) $w = f(v) = -v^2$

c) $w = \text{trace}(MV)$ where $\mathrm{M} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$ and $V = V^\top$

2) Which of the following sets is convex?

a) $S := \{z \in \mathbb{C} : |z| \leq 1\}$

b) $S := \{z \in \mathbb{C} : |z| = 1\}$

c) $S := \{x \in \mathbb{R}^2 : x_1^2 + x_2^2 \leq 1\}$

d) $S := \left\{ \begin{bmatrix} v \\ r \end{bmatrix} \in \mathbb{R}^2 : v^2 \leq r \right\}$

Convex Optimization

Again, consider a finite-dimensional optimization of the form:

$$p^* = \inf_{x \in \mathcal{S} \subset \mathbb{R}^v} f_0(x)$$

Definition: This is a convex optimization if the objective $f_0 : \mathbb{R}^v \to \mathbb{R}$ is a convex function and the feasible set S is convex.

A key property is that every locally optimal point of a convex optimization is also globally optimal.

- Any algorithm that computes a local optima, e.g. gradient descent, computes a global optima.
- There are fast and reliable software for certain classes of convex optimizations.

Linear Matrix Inequalities (LMIs)

Definition: Let symmetric matrices $\{F_i\}_{i=0}^{v} \subset \mathbb{R}^{m\times m}$ be given. An LMI is a constraint on $x \in \mathbb{R}^v$ of the form:

$$F(x) := F_0 + x_1 F_1 + \cdots + x_v F_v \le 0$$

In other words, $F(x)$ is negative semidefinite.

Fact: The set $S := \{x \in \mathbb{R}^v : F(x) \le 0\}$ is convex.

Proof: Take any $x_1, x_2 \in S$ so that $F(x_1) \le 0$ and $F(x_2) \le 0$.

Define $x = \lambda x_1 + (1-\lambda)x_2$ where $\lambda \in [0,1]$. Then,

$$F(x) = \lambda\, F(x_1) + (1-\lambda)\, F(x_2) \le 0 \implies \quad x \in S$$

Fact: Two LMI constraints $G(x) \le 0$ and $H(x) \le 0$ can be combined into a single, equivalent LMI constraint:

$$F(x) := \begin{bmatrix} G(x) & 0 \\ 0 & H(x) \end{bmatrix} \le 0$$

Linear Matrix Inequalities (LMIs)

Example: Let a 2-by-2 matrix A be given. Consider the following set of 2-by-2 symmetric matrices:

$$S := \{P = P^\top : P \geq I \text{ and } A^\top P + PA \leq 0\}$$

This set can be expressed by an LMI as follows:

1. Express $P = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix}$ where x_1, x_2, x_3 are scalar variables.

2. Combine the two LMIs as: $\begin{bmatrix} A^\top P + PA & 0 \\ 0 & I - P \end{bmatrix} \leq 0$

3. Define a basis for 2-by-2 symmetric matrices:

$$E_1 := \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, E_2 := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, E_3 := \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

4. Expand the single LMI in step 2 using the basis in step 3:

$$\begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} + x_1 \begin{bmatrix} A^\top E_1 + E_1 A & 0 \\ 0 & -E_1 \end{bmatrix} + x_2 \begin{bmatrix} A^\top E_2 + E_2 A & 0 \\ 0 & -E_2 \end{bmatrix} + x_3 \begin{bmatrix} A^\top E_3 + E_3 A & 0 \\ 0 & -E_3 \end{bmatrix} \leq 0$$

Linear Matrix Inequalities (LMIs)

Example: Let a 2-by-2 matrix A be given. Consider the following set of 2-by-2 symmetric matrices:

$$S := \{P = P^\top : P \geq I \text{ and } A^\top P + PA \leq 0\}$$

This set can be expressed by an LMI as follows:

$$S := \{x \in \mathbb{R}^3 : \ F(x) := F_0 + x_1 F_1 + x_2 F_2 + x_3 F_3 \leq 0\}$$

where:

$$F_0 := \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}$$
$$F_1 := \begin{bmatrix} A^\top E_1 + E_1 A & 0 \\ 0 & -E_1 \end{bmatrix}$$
$$F_2 := \begin{bmatrix} A^\top E_2 + E_2 A & 0 \\ 0 & -E_2 \end{bmatrix}$$
$$F_3 := \begin{bmatrix} A^\top E_3 + E_3 A & 0 \\ 0 & -E_3 \end{bmatrix}$$

Semidefinite Programs (SDPs)

Definition: A semidefinite program is an optimization with a linear objective function and an LMI constraint:

$$p^* = \inf_{x \in \mathbb{R}^v} c^\top x \ \text{ subject to: } F(x) \leq 0$$

where $c \in \mathbb{R}^v$ and symmetric matrices $\{F_i\}_{i=0}^{v} \subset \mathbb{R}^{m\times m}$ are given.

Comments:

1. The linear objective function is convex and the LMI constraint defines a convex feasible set $\Longrightarrow$ An SDP is a convex optimization.

2. Equality constraints *Ax=b* can also be added to the problem.

3. There are many parsers and solvers that can efficiently solve this class of problems (with “moderate” size). The most common algorithms use primal-dual formulations.

Semidefinite Programs (SDPs)

Example: Let a 2-by-2 matrix A be given. Define the optimization:

$$p^* = \min_{P=P^\top \in \mathbb{R}^{2\times 2}} \text{trace}(P)$$

$$\text{subject to: } P \geq I,\ A^\top P + PA \leq 0$$

This can be expressed as an SDP as follows:

1. Express $P = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix}$ where x_1, x_2, x_3 are scalar variables.

2. Combine the two LMIs as: $\begin{bmatrix} A^\top P + PA & 0 \\ 0 & I - P \end{bmatrix} \leq 0$

3. Define a basis for 2-by-2 symmetric matrices:

$$E_1 := \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, E_2 := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, E_3 := \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

4. Expand the single LMI in step 2 using the basis in step 3:

$$\begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} + x_1 \begin{bmatrix} A^\top E_1 + E_1 A & 0 \\ 0 & -E_1 \end{bmatrix} + x_2 \begin{bmatrix} A^\top E_2 + E_2 A & 0 \\ 0 & -E_2 \end{bmatrix} + x_3 \begin{bmatrix} A^\top E_3 + E_3 A & 0 \\ 0 & -E_3 \end{bmatrix} \leq 0$$

5. Rewrite the objective as $\text{trace}(P) = c^\top x$ where $c = [1, 0, 1]^\top$.

Semidefinite Programs (SDPs)

Example: Let a 2-by-2 matrix A be given. Define the optimization:

$$p^* = \min_{P=P^\top \in \mathbb{R}^{2\times 2}} \operatorname{trace}(P)$$

$$\text{subject to: } P \geq I,\ A^\top P + PA \leq 0$$

This can be expressed as an SDP as follows:

$$p^* = \inf_{x \in \mathbb{R}^3} c^\top x \ \ \text{subject to: } F(x) \leq 0$$

where:

$$c = [1, 0, 1]^\top$$

$$F_0 := \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}$$

$$F_1 := \begin{bmatrix} A^\top E_1 + E_1 A & 0 \\ 0 & -E_1 \end{bmatrix}$$

$$F_2 := \begin{bmatrix} A^\top E_2 + E_2 A & 0 \\ 0 & -E_2 \end{bmatrix}$$

$$F_3 := \begin{bmatrix} A^\top E_3 + E_3 A & 0 \\ 0 & -E_3 \end{bmatrix}$$

Convex Optimization / SDPs

Wake-up Problem

Consider the following nonlinear system:

$\dot{x}(t) = -3x(t) + \phi(x(t))$ where $\phi\colon \mathbb{R} \to \mathbb{R}$ is in the sector $[-2,2]$.

The system can be expressed as:

$$\dot{x}(t) = -3x(t) + w(t) \text{ and } w(t) = \phi(x(t))$$

A) Define a matrix $J = J^\top$ such that: $\begin{bmatrix} x \\ w \end{bmatrix}^\top J \begin{bmatrix} x \\ w \end{bmatrix} \geq 0 \quad \forall x \in \mathbb{R}.$

B) If $\exists\ \lambda \geq 0$ and $\alpha > 0$ such that $V(x) = x^2$ satisfies the following along all trajectories:

$$\frac{d}{dt}V(x(t)) + \lambda \begin{bmatrix} x(t) \\ w(t) \end{bmatrix}^\top J \begin{bmatrix} x(t) \\ w(t) \end{bmatrix} \leq -2\alpha V(x(t))$$

then $x(t) \to\ 0$ exponentially with rate α.

Write an SDP to find the maximal α subject to this condition.

C) Solve this SDP analytically for the maximal α^*.

Computational Issues

There are many available solvers for SDPs including Mosek, Sedumi, SDPT3, and LMILab.

We often express the LMI constraints using matrix variables, e.g. $P \geq I$ or $A^\top P + PA \leq 0$. However, solvers often require the SDP to have a single LMI expressed in standard form:

$$F(x) := F_0 + x_1 F_1 + \cdots + x_v F_v \leq 0$$

It can be cumbersome and time-consuming to convert this to standard form. Numerous parsers have been developed to aid in this conversion process, e.g. CVX or YALMIP.

There is a vast literature on numerical algorithms. The next few slides will highlight a few key issues with these tools.

User Interface / Parsers

To make this concrete, consider the following SDP with *A* given:

$$\min_{P=P^\top \in \mathbb{R}^{n\times n}} \quad \mathrm{trace}(P)$$

$$\text{subject to: } \; P \geq I, \; A^\top P + PA \leq 0$$

The constraint $A^\top P + P A \leq 0$ is called a Lyapunov inequality.

User Interface / Parsers

To make this concrete, consider the following SDP with *A* given:

$$\min_{P=P^\top\in\mathbb{R}^{n\times n}} \quad \text{trace}(P)$$

$$\text{subject to:} \;\; P \geq I,\; A^\top P + PA \leq 0$$

CVX code to implement this in Matlab is:

```
cvx_begin sdp
      variable P(n,n) symmetric;
      P >= eye(n);
      (A'*P+P*A ) <= 0;
      minimize( trace(P) );
cvx_end
```

The code almost exactly matches the SDP at the top of the page. This can also be implemented in Python using CVXPY.

CVX converts this to a standard form for an SDP solver (selected by the user), calls the solver, and transforms the solution back.

User Interface / Parsers

To make this concrete, consider the following SDP with *A* given:

$$\min_{P=P^\top\in\mathbb{R}^{n\times n}} \quad \mathrm{trace}(P)$$

$$\text{subject to:}\;\; P \geq I,\; A^\top P + PA \leq 0$$

Contrast this with the code to efficiently implement this SDP using a typical solver, e.g. Matlab code for Sedumi is:

```
K.s = [n n];
lidx = find( tril( ones(n,n) ) );
bs = A’+A;
bs = bs(lidx);
I = eye(n);
At = A’;
tmp = kron(A',eye(n))+kron(eye(n),A’);
As = [-reshape(tmp,[n^2,n^2]) -speye(n^2)];
As = As(lidx,:);
cs = speye(n);
cs = [cs(:); sparse(n^2,1)];
[xs,ys,infos] = sedumi(As,bs,cs,K);
```

User Interface / Parsers

To make this concrete, consider the following SDP with *A* given:

$$\min_{P=P^\top\in\mathbb{R}^{n\times n}} \quad \text{trace}(P)$$

$$\text{subject to:} \;\; P \geq I, \; A^\top P + PA \leq 0$$

Comments:

- User interfaces make it very easy to implement and solve SDPs.
- However, the conversion may not yield the best implementation in terms of computational cost, memory, and numerical conditioning.

User interfaces, e.g. CVX and Yalmip, are useful tools for prototyping or "one-off" implementations. A direct solver implementation should be used when high re-use or high efficiency is required.

Computational Complexity

Consider an SDP in standard form with v variables and an $m\times m$ LMI constraint:

$$p^* = \inf_{x\in\mathbb{R}^v} c^\top x \;\; \text{subject to: } F(x) \le 0$$

Section 11.8.3 of Boyd & Vandeberghe estimate that the number of floating point operations for a general primal/dual algorithm to solve this problem is (order of magnitude):

$$\max(vm^3, v^2m^2, v^3)$$

Specialized solvers can be faster for problems with sparsity or structure but this provides a good estimate for general problems.

Computational Complexity

Consider again the following SDP with A given:

$$\min_{P=P^\top\in\mathbb{R}^{n\times n}} \quad \text{trace}(P)$$
$$\text{subject to:} \;\; P \geq I, \; A^\top P + PA \leq 0$$

This problem has $v = \frac{n(n+1)}{2} = O(n^2)$ scalar decision variables corresponding to the entries of P. The two constraints can be stacked into a single LMI of dimension $m = 2n = O(n)$.

The number of floating point operations to solve this SDP scales roughly as follows (neglecting constants):

$$\max(vm^3, v^2m^2, v^3) \sim \max(n^5, n^6, n^6) \sim O(n^6)$$

Computational Complexity

Consider again the following SDP with *A* given:

$$\min_{P=P^\top \in \mathbb{R}^{n\times n}} \quad \text{trace}(P)$$

$$\text{subject to: } \; P \geq I, \; A^\top P + PA \leq 0$$

The number of floating point operations scales as $O(n^6)$.

For comparison, consider the Lyapunov equation with *A* given:

$$A^\top P + PA = -I$$

If *A* is Hurwitz then *P>0.*

Note: The solution of the Lyapunov equation is not necessarily the same as the min trace solution (even after accounting for a constant scaling). However, it does provide a useful comparison.

Computational Complexity

Consider again the following SDP with *A* given:

$$\min_{P=P^\top \in \mathbb{R}^{n\times n}} \quad \text{trace}(P)$$

$$\text{subject to: } \; P \geq I, \; A^\top P + PA \leq 0$$

The number of floating point operations scales as $O(n^6)$.

For comparison, consider the Lyapunov equation with *A* given:

$$A^\top P + PA = -I$$

If *A* is Hurwitz then *P>0.* This is a linear equation in *P* and the algorithm by Bartels & Stewart (‘72 ACM) scales as $O(n^3)$.

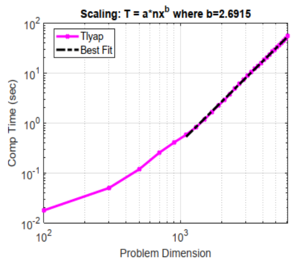
SDPs are much more expensive to solve than typical equations that arise in control problems (Lyapunov, Riccati, etc). However, they are still relatively efficient and can be used to solve problems even as the dimension grows to a few hundred.

Computational Complexity

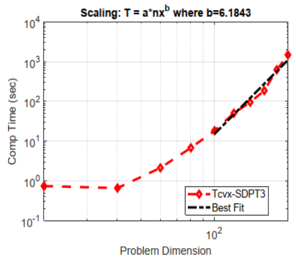
Computation time vs. problem size *n* on a standard laptop.

The Lyapunov equation can be solved in <100 seconds for $n = 6000$ while the SDP solve time is >1000 seconds for $n = 200$.

Lyapunov Equation



Lyapunov SDP with CVX+SDPT3

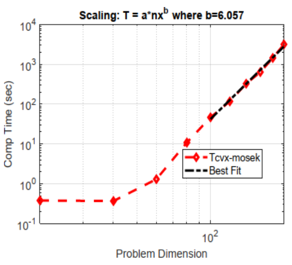


Computational Complexity

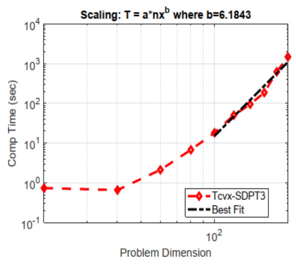
Computation time vs. problem size *n* on a standard laptop.

The plots below compare CVX with both Mosek and SDPT3. The scaling is similar for both solvers.

Lyapunov SDP with CVX+Mosek



Lyapunov SDP with CVX+SDPT3



Strict vs. Non-strict Inequalities

We posed SDPs in standard form with non-strict inequalities:

$$p^* = \inf_{x \in \mathbb{R}^v} c^\top x \;\; \text{subject to: } F(x) \leq 0$$

x is feasible if $F(x)$ is a negative semidefinite matrix.

Strict vs. Non-strict Inequalities

We posed SDPs in standard form with non-strict inequalities:

$$p^* = \inf_{x \in \mathbb{R}^v} c^\top x \quad \text{subject to: } F(x) \le 0$$

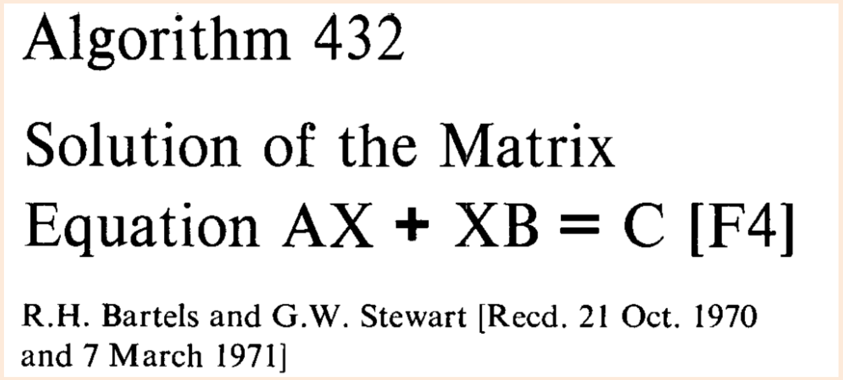
Issue:

- Most solvers enforce LMIs using non-strict inequalities. LMILab is one exception which enforces LMIs with strict constraints.
- However, Lyapunov conditions typically require strict inequalities, e.g. $P > 0$ and $A^\top P + P A < 0$.
- This may appear to be a minor technical issue. However, numerical errors can cause solvers to return slightly infeasible solutions, e.g. a Lyapunov matrix *P* with a slightly negative eigenvalue.
- These errors can lead to incorrect stability/performance conclusions.

Solutions:

- Enforce constraints as $F(x) \le -\epsilon I$ for some “small” $\epsilon > 0$.
- Use a solver that strictly enforces constraints, e.g. LMILab.

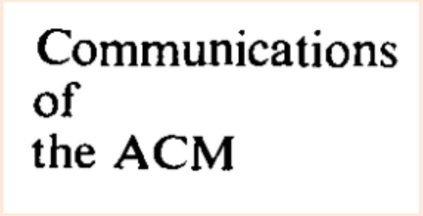
History: Numerical Algorithms



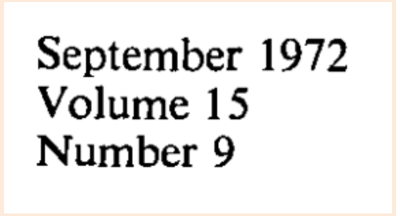
Algorithm 432

Solution of the Matrix Equation $AX + XB = C$ [F4]

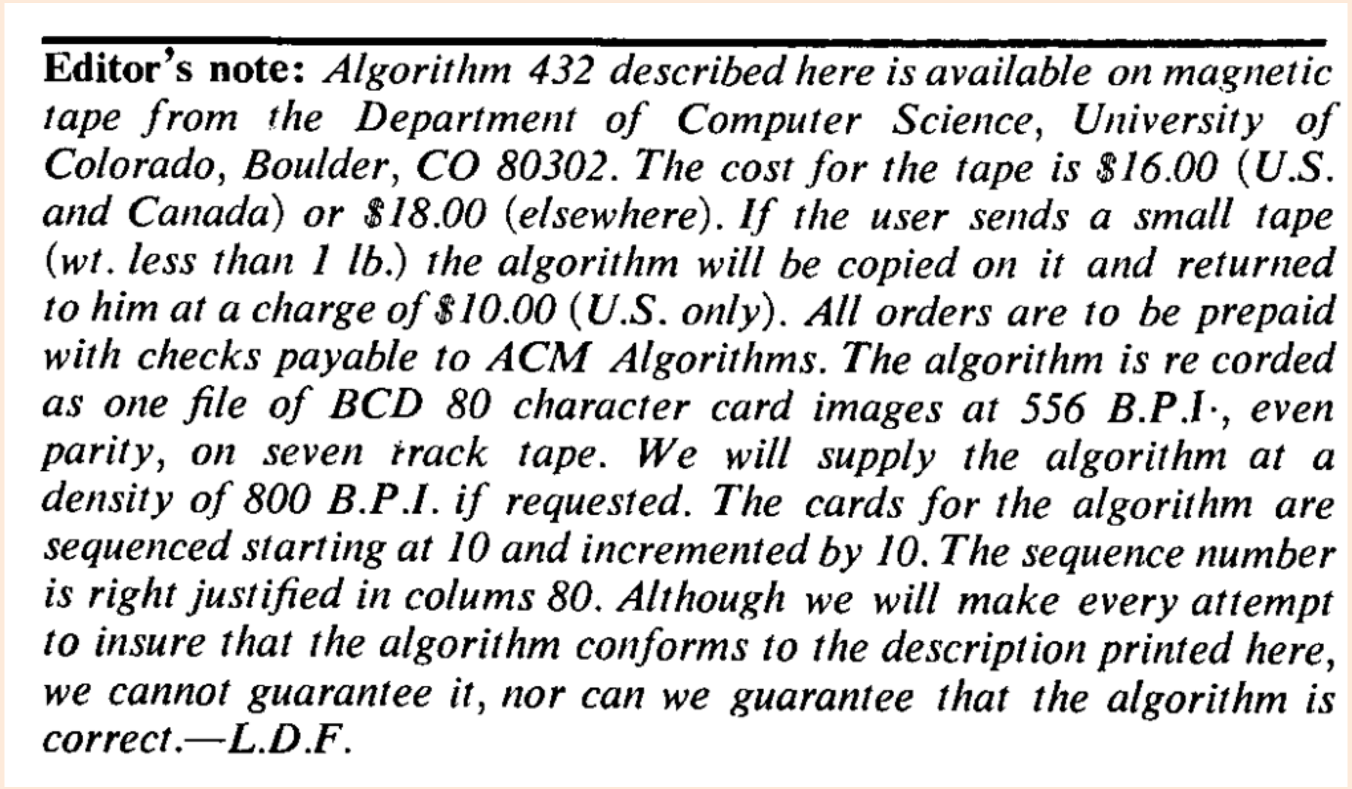
R.H. Bartels and G.W. Stewart [Recd. 21 Oct. 1970 and 7 March 1971]



Communications of the ACM



September 1972
Volume 15
Number 9



Editor's note: *Algorithm 432 described here is available on magnetic tape from the Department of Computer Science, University of Colorado, Boulder, CO 80302. The cost for the tape is \$16.00 (U.S. and Canada) or \$18.00 (elsewhere). If the user sends a small tape (wt. less than 1 lb.) the algorithm will be copied on it and returned to him at a charge of \$10.00 (U.S. only). All orders are to be prepaid with checks payable to ACM Algorithms. The algorithm is re corded as one file of BCD 80 character card images at 556 B.P.I·, even parity, on seven track tape. We will supply the algorithm at a density of 800 B.P.I. if requested. The cards for the algorithm are sequenced starting at 10 and incremented by 10. The sequence number is right justified in colums 80. Although we will make every attempt to insure that the algorithm conforms to the description printed here, we cannot guarantee it, nor can we guarantee that the algorithm is correct.—L.D.F.*

Convex Optimization / SDPs

Wake-up Problem

Download and install CVX if you don’t already have it installed.

Run the following test example:

```
A = [-2 20; 0 -2];
n = size(A,1);
cvx_begin sdp
      variable P(n,n) symmetric;
      P >= eye(n);
      (A'*P+P*A ) <= 0;
      minimize( trace(P) );
cvx_end
```

Your optimal P should be (up to numerical errors):

```
P =     1.0000     0.0001
        0.0001    25.0000
```

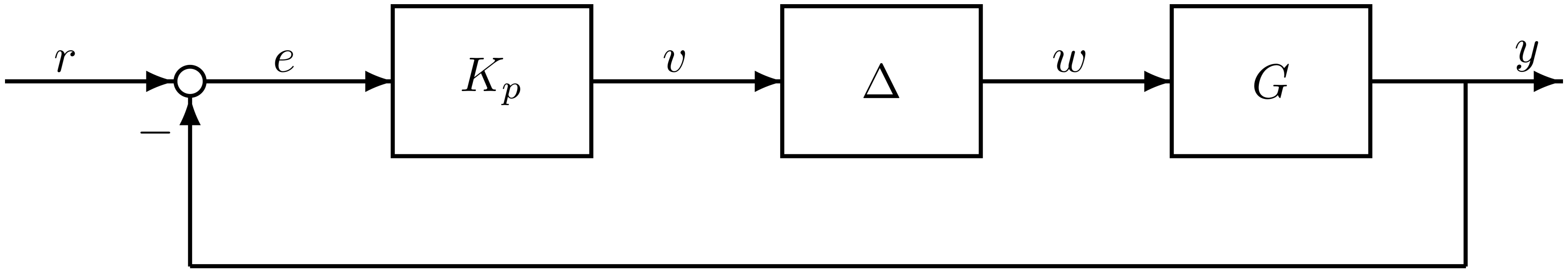
Numerical Example 1: Sector-Bounded NL

Consider the feedback system below with:

$$K_p = 20 \text{ and } G(s) = \frac{4}{s^2+8s+7}$$

Assume Δ is a static, memoryless nonlinearity in the sector $[1 - p, 1 + p]$ where p represents the level of nonlinearity.

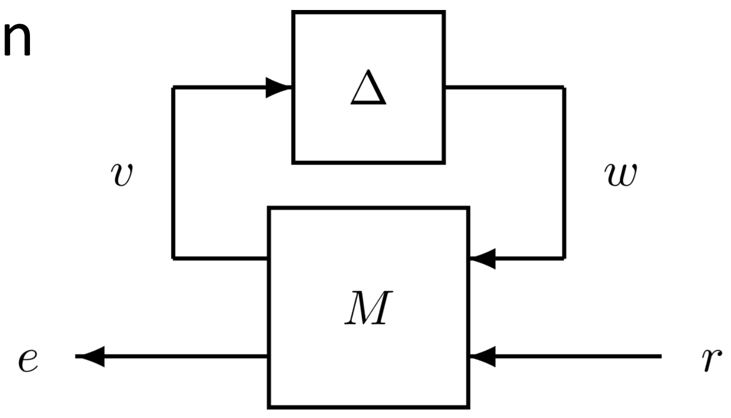
We will study the effect of the nonlinearity on the gain from reference *r* to error *e*.

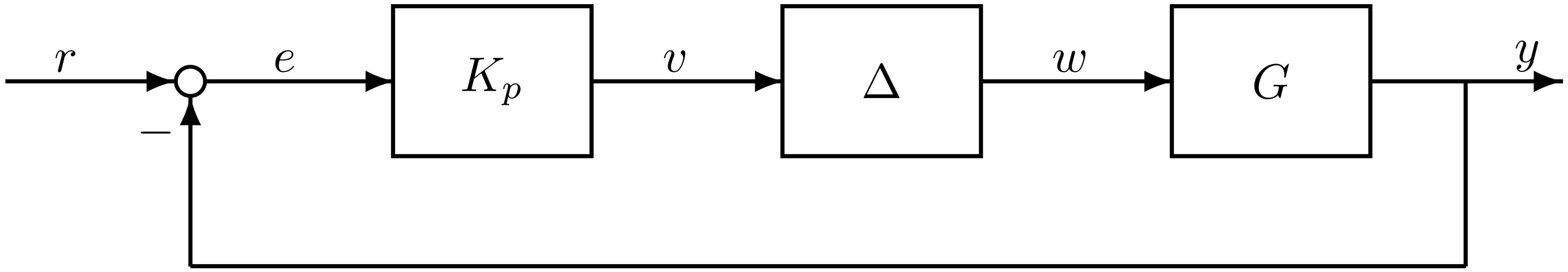


Numerical Example 1: Sector-Bounded NL

Step 1: Express the uncertain system as an LFT $F_U(M, \Delta)$ with the nonlinearity in Δ.

- $w = \Delta(v)$ and Δ is the nonlinearity.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -K_p G & K_p \\ -G & 1 \end{bmatrix}$

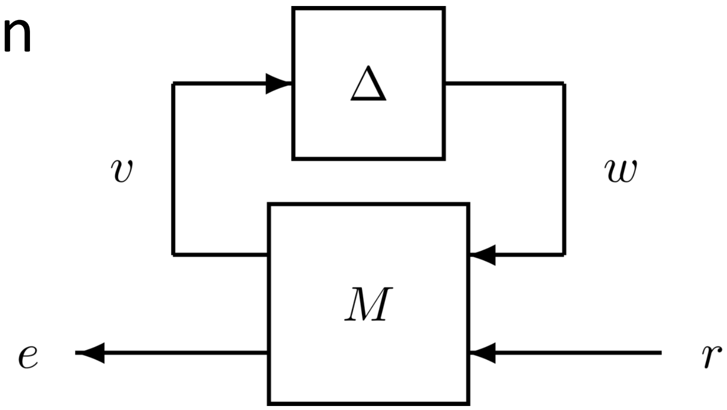




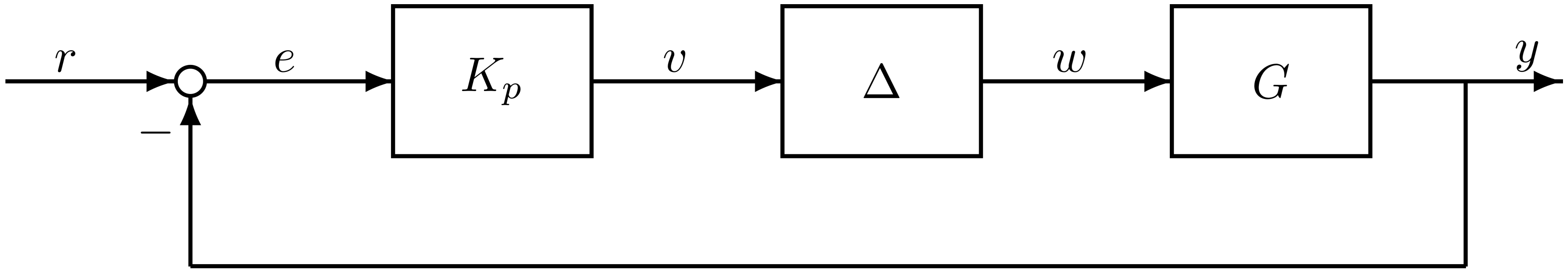
Numerical Example 1: Sector-Bounded NL

Step 1: Express the uncertain system as an LFT $F_U(M,\Delta)$ with the nonlinearity in Δ.

- $w = \Delta(v)$ and Δ is the nonlinearity.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -K_pG & K_p \\ -G & 1 \end{bmatrix}$



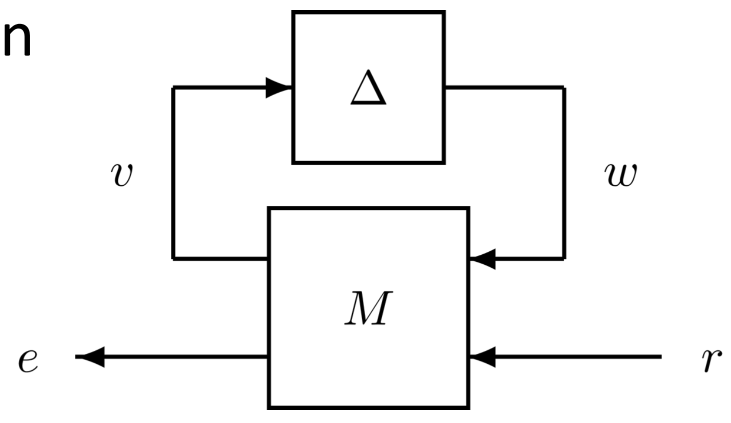
Sanity check: If $\Delta = 1$ then the closed-loop from r to e is the sensitivity $S(s) = \frac{1}{1+G(s)K_p}$.



Numerical Example 1: Sector-Bounded NL

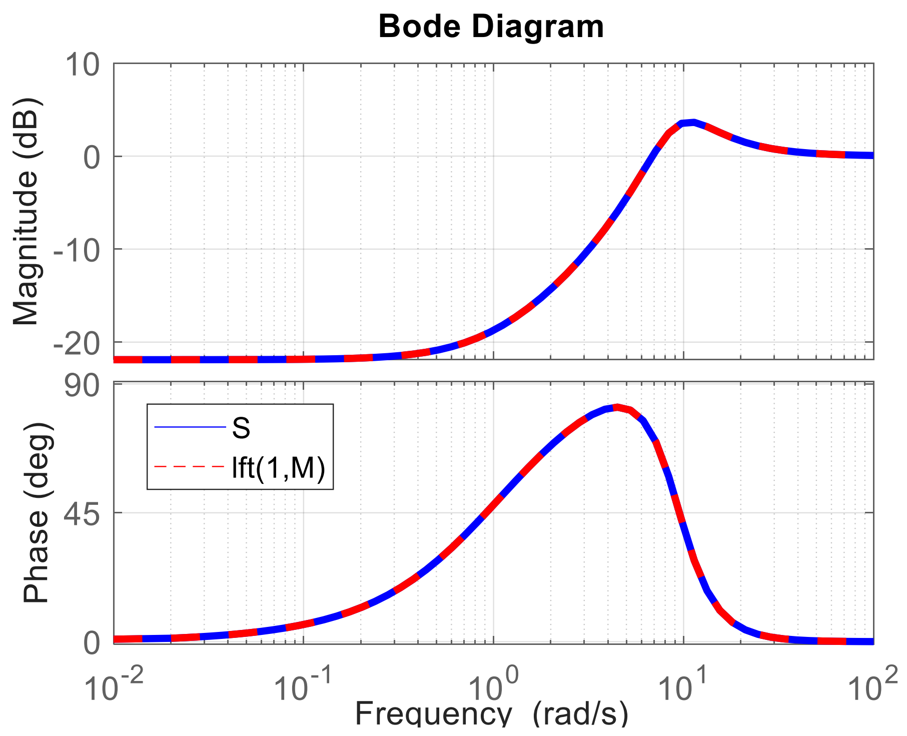
Step 1: Express the uncertain system as an LFT $F_U(M,\Delta)$ with the nonlinearity in Δ.

- $w = \Delta(v)$ and Δ is the nonlinearity.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -K_pG & K_p \\ -G & 1 \end{bmatrix}$



Sanity check: If $\Delta = 1$ then the closed-loop from r to e is the sensitivity $S(s) = \frac{1}{1+G(s)K_p}$.

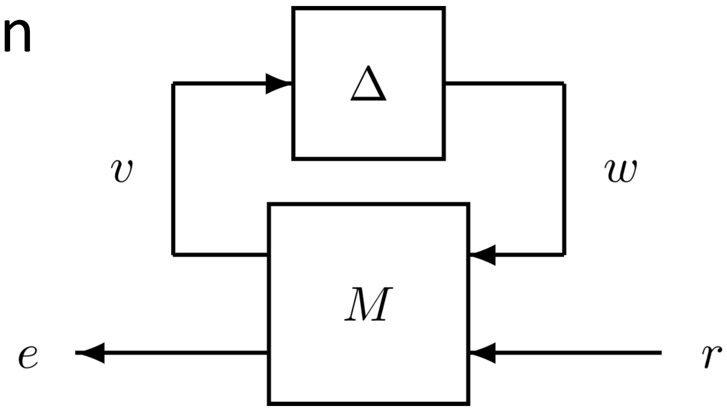
$F_U(M,1)$ matches $S(s)$ thus verifying our construction. Note that $||S||_\infty = 1.53$.



Numerical Example 1: Sector-Bounded NL

Step 1: Express the uncertain system as an LFT $F_U(M, \Delta)$ with the nonlinearity in Δ.

- $w = \Delta(v)$ and Δ is the nonlinearity.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -K_p G & K_p \\ -G & 1 \end{bmatrix}$



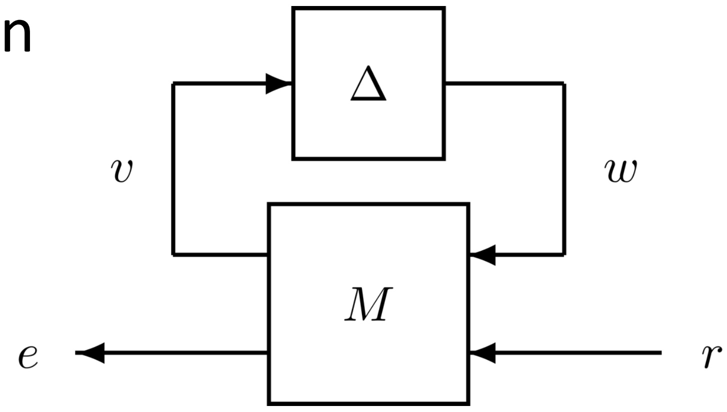
Step 2: Specify a static QC J for Δ in the sector $[1-p, 1+p]$.

$$J = \begin{bmatrix} -2\alpha\beta & \alpha+\beta \\ \alpha+\beta & -2 \end{bmatrix} \text{ where } \alpha = 1-p,\ \beta = 1+p$$

Numerical Example 1: Sector-Bounded NL

Step 1: Express the uncertain system as an LFT $F_U(M,\Delta)$ with the nonlinearity in Δ.

- $w = \Delta(v)$ and Δ is the nonlinearity.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -K_pG & K_p \\ -G & 1 \end{bmatrix}$



Step 2: Specify a static QC J for Δ in the sector $[1-p, 1+p]$.

$$J = \begin{bmatrix} -2\alpha\beta & \alpha+\beta \\ \alpha+\beta & -2 \end{bmatrix} \text{ where } \alpha = 1-p,\ \beta = 1+p$$

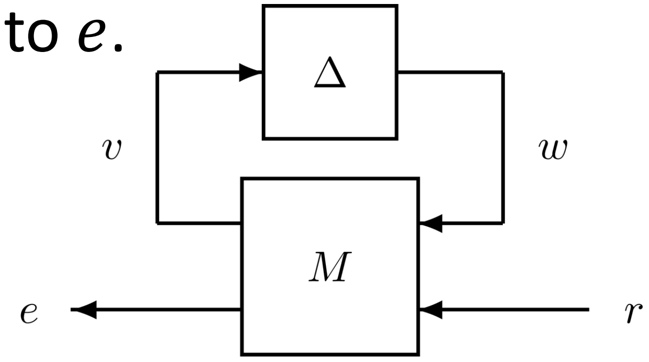
The constraint also holds when scaled by any $\lambda \geq 0$:

$$J = \lambda \begin{bmatrix} -2\alpha\beta & \alpha+\beta \\ \alpha+\beta & -2 \end{bmatrix} \text{ where } \alpha = 1-p,\ \beta = 1+p,\ \lambda \geq 0$$

This additional scaling reduces the conservatism in L_2 gain bound.

Numerical Example 1: Sector-Bounded NL

Step 3: Write SDP to bound the gain from r to e.



- A state-space model for M is:

$$\begin{bmatrix} \dot{x}(t) \\ v(t) \\ e(t) \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x(t) \\ w(t) \\ r(t) \end{bmatrix}$$

- Dissipation ineq. with $\lambda \geq 0, V(x) = x^\top P\, x, P \geq 0$ is:

$$\frac{d}{dt} V(x(t)) + \begin{bmatrix} e(t) \\ r(t) \end{bmatrix}^\top \begin{bmatrix} I & 0 \\ 0 & -\gamma^2 I \end{bmatrix} \begin{bmatrix} e(t) \\ r(t) \end{bmatrix} + \lambda \begin{bmatrix} v(t) \\ w(t) \end{bmatrix}^\top J \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} \leq 0$$

- Equivalent LMI form of the dissipation inequality is:

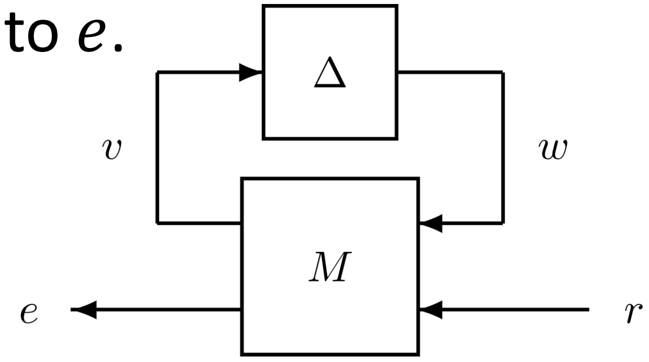
$$\begin{bmatrix} A^T P + PA & PB_1 & PB_2 \\ B_1^T P & 0 & 0 \\ B_2^T P & 0 & 0 \end{bmatrix} + (\cdot)^\top \begin{bmatrix} I & 0 \\ 0 & -\gamma^2 I \end{bmatrix} \begin{bmatrix} C_2 & D_{21} & D_{22} \\ 0 & 0 & I \end{bmatrix}$$

$$+ \lambda (\cdot)^\top J \begin{bmatrix} C_1 & D_{11} & D_{12} \\ 0 & I & 0 \end{bmatrix} \preceq 0$$

This is a version of a classical result known as the “Circle Criterion”.

Numerical Example 1: Sector-Bounded NL

Step 3: Write SDP to bound the gain from r to e.



$$\min_{\lambda\geq 0,\gamma^2>0,P\geq 0} \gamma^2$$

subject to:

$$\begin{bmatrix} A^TP+PA & PB_1 & PB_2 \\ B_1^TP & 0 & 0 \\ B_2^TP & 0 & 0 \end{bmatrix} + (\cdot)^\top \begin{bmatrix} I & 0 \\ 0 & -\gamma^2 I \end{bmatrix} \begin{bmatrix} C_2 & D_{21} & D_{22} \\ 0 & 0 & I \end{bmatrix}$$

$$+\lambda\,(\cdot)^\top J \begin{bmatrix} C_1 & D_{11} & D_{12} \\ 0 & I & 0 \end{bmatrix} \preceq 0$$

Numerical Example 1: Sector-Bounded NL

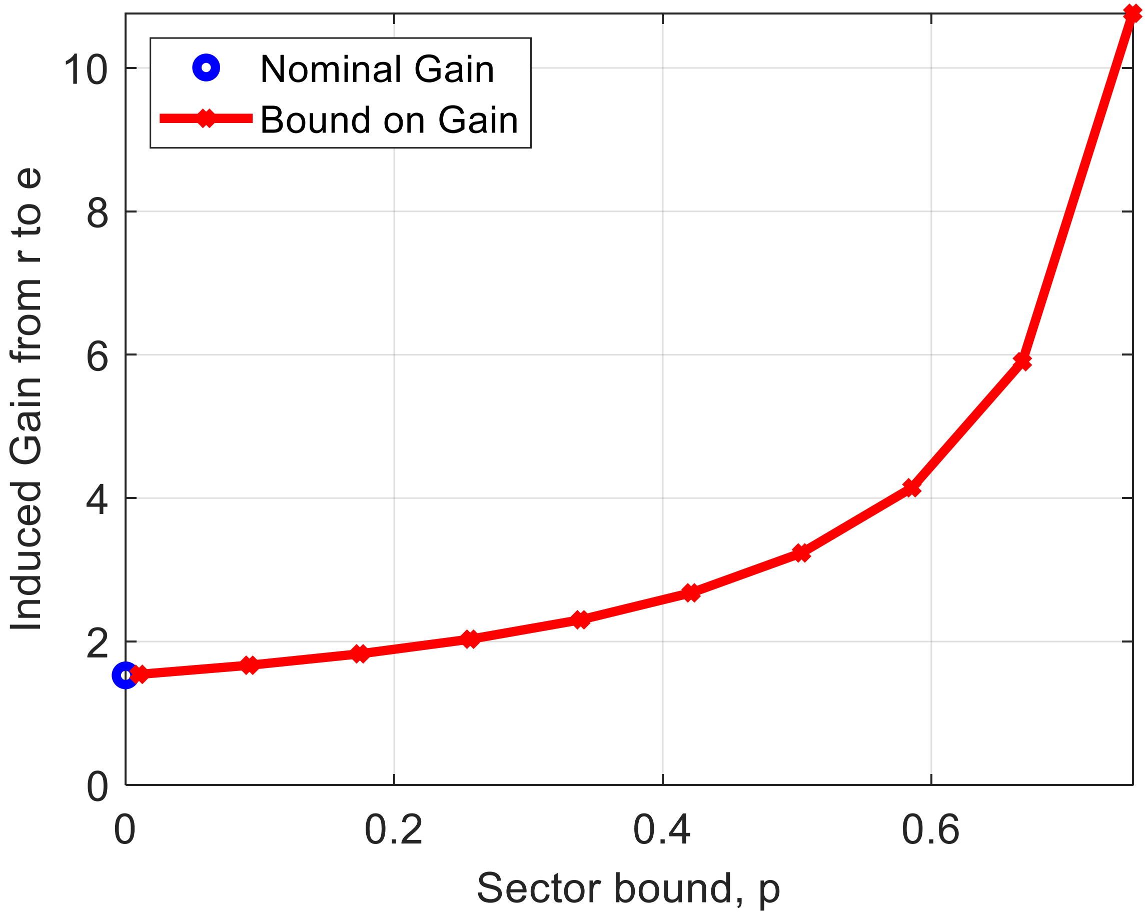
Step 3: Write SDP to bound the gain from r to e. CVX code:

```
% Form matrices used in SDP
Zwr = zeros(nw,nr); Zwx = zeros(nw,nx); Zw = zeros(nw);
Iw = eye(nw); Ir = eye(nr);
Lvw = [C1 D11 D12; Zwx Iw Zwr];
Le = [C2 D21 D22];
% Solve SDP using CVX
a = 1-p; b = 1+p;
J = [-2*a*b, (a+b); (a+b) -2];
cvx_begin sdp quiet
       variable P(nx,nx) semidefinite;
       variable gsq(1,1);
       variable lambda nonnegative;
       [Am'*P+P*Am P*B1 P*B2; B1'*P Zw Zwr; B2'*P Zwr' -gsq*Ir] ...
       +lambda*(Lvw'*J*Lvw) + Le'*Le <=0;
       minimize(gsq)
cvx_end
```

Numerical Example 1: Sector-Bounded NL

Induced L_2 gain vs. sector bound p.

Note $p = 0$ corresponds to the nominal case ($\Delta = 1$). The red curve matches the nominal gain $||S||_\infty = 1.53$ when $p = 0$.



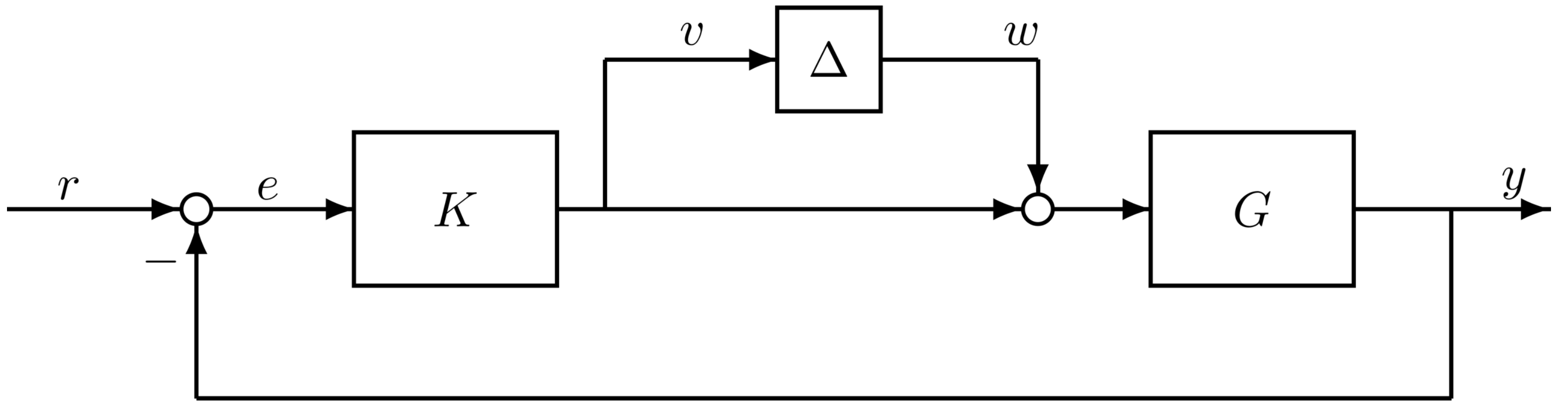
Numerical Example 2: LTI Uncertainty

Consider the feedback system below with:

$$K(s) = \frac{151.9\, s^2 + 789.3\, s + 1025}{3.32\, s^2 + 79.6\, s} \text{ and } G(s) = \frac{4}{s^2+2.8s+4}$$

$K(s)$ is a PID controller with approximate derivative. It is designed to achieve a loop bandwidth near 8 rad/sec. Assume Δ is a stable LTI uncertainty with $||\Delta||_\infty \leq \beta$ where β represents the level of uncertainty.

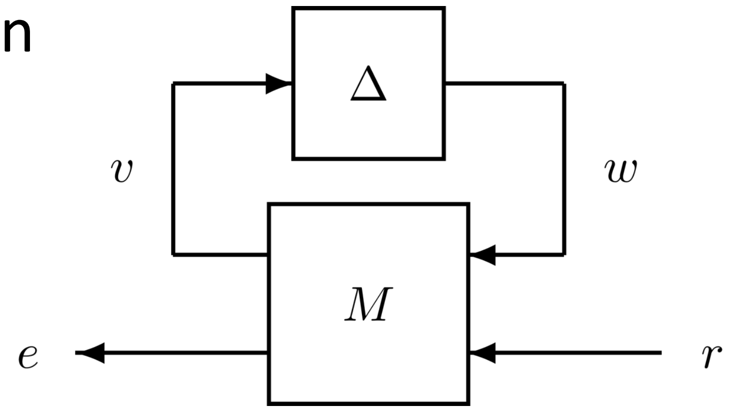
We will study the effect of the uncertainty on the gain from reference *r* to error *e*.



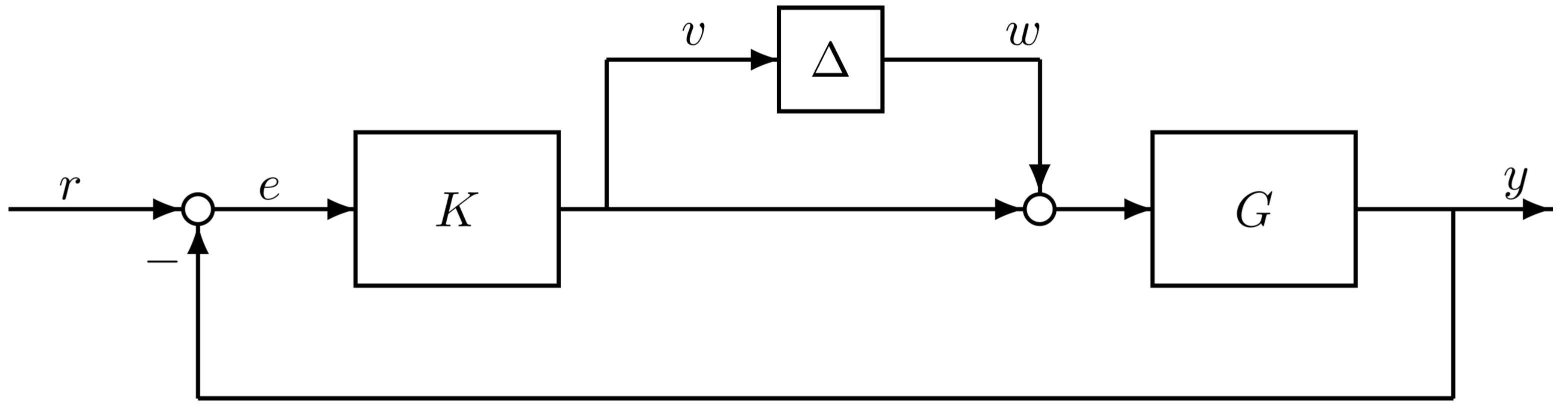
Numerical Example 2: LTI Uncertainty

Step 1: Express the uncertain system as an LFT $F_U(M, \Delta)$ with the uncertainty Δ.

- $w = \Delta(v)$ and Δ is the uncertainty.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -T & KS \\ -GS & S \end{bmatrix}$



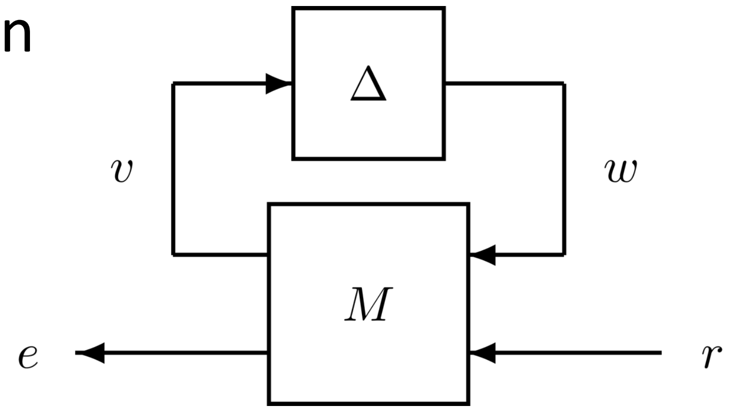
where $S(s) = \frac{1}{1+G(s)K(s)}$ and $T(s) = \frac{G(s)K(s)}{1+G(s)K(s)}$ are the nominal sensitivity and complementary sensitivity.



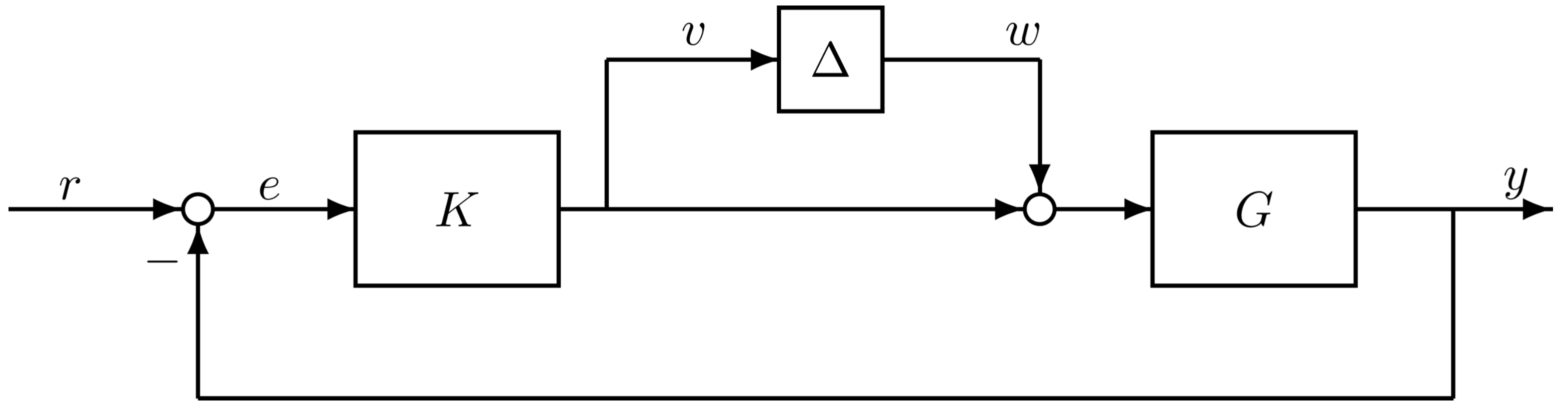
Numerical Example 2: LTI Uncertainty

Step 1: Express the uncertain system as an LFT $F_U(M, \Delta)$ with the uncertainty Δ.

- $w = \Delta(v)$ and Δ is the uncertainty.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -T & KS \\ -GS & S \end{bmatrix}$



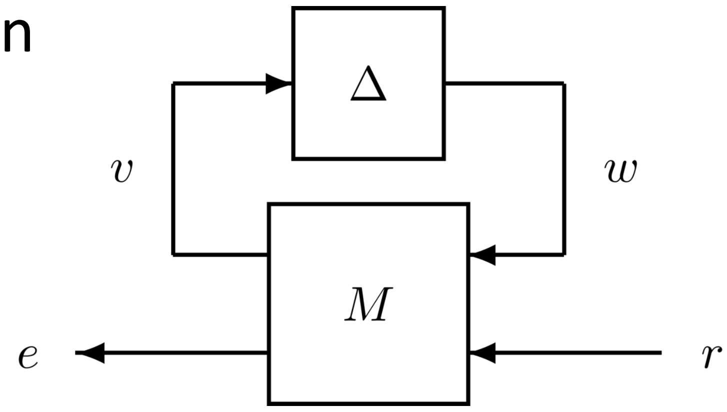
Sanity check: If $\Delta = 0$ then the closed-loop from r to e is the sensitivity $S(s) = \frac{1}{1+G(s)K(s)}$.



Numerical Example 2: LTI Uncertainty

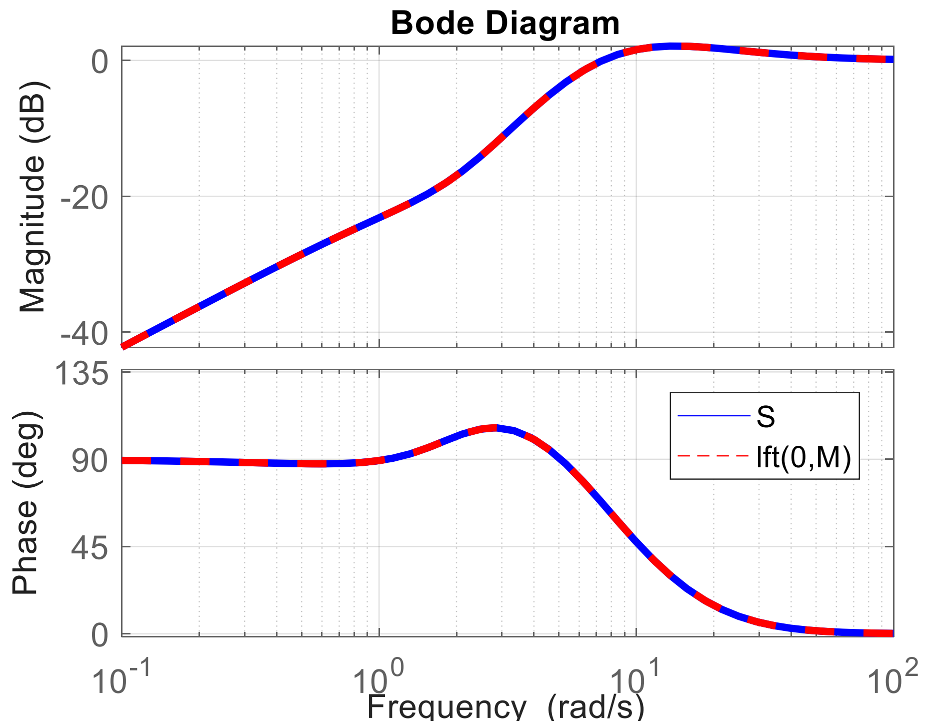
Step 1: Express the uncertain system as an LFT $F_U(M,\Delta)$ with the uncertainty Δ.

- $w = \Delta(v)$ and Δ is the uncertainty.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -T & KS \\ -GS & S \end{bmatrix}$



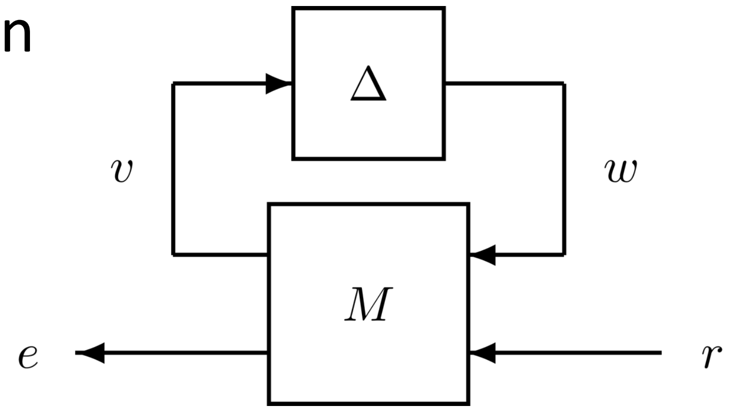
Sanity check: If $\Delta = 0$ then the closed-loop from r to e is the sensitivity $S(s) = \frac{1}{1+G(s)K(s)}$.

$F_U(M,0)$ matches $S(s)$ thus verifying our construction. Note that $||S||_\infty = 1.28$.



Numerical Example 2: LTI Uncertainty

Step 1: Express the uncertain system as an LFT $F_U(M, \Delta)$ with the uncertainty Δ.



- $w = \Delta(v)$ and Δ is the uncertainty.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -T & KS \\ -GS & S \end{bmatrix}$

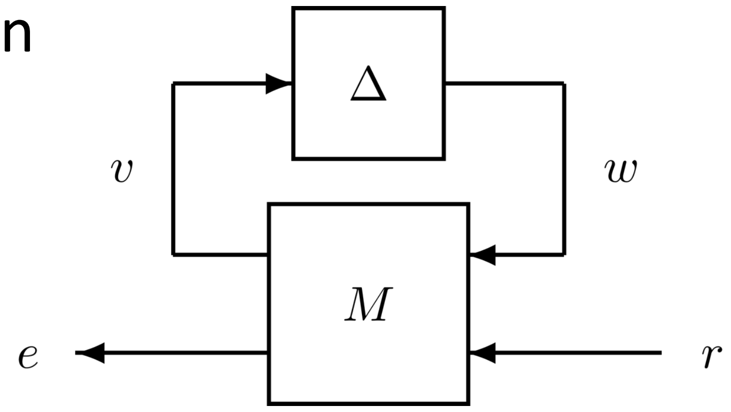
Step 2: Specify an IQC J for the gain bound $||\Delta||_\infty \leq \beta$.

$$\int_0^T \begin{bmatrix} v(t) \\ w(t) \end{bmatrix}^\top J \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} dt \geq 0 \ \ \forall T \geq 0 \ \text{ with } \ J = \begin{bmatrix} \beta^2 & 0 \\ 0 & -1 \end{bmatrix}$$

Numerical Example 2: LTI Uncertainty

Step 1: Express the uncertain system as an LFT $F_U(M, \Delta)$ with the uncertainty Δ.

- $w = \Delta(v)$ and Δ is the uncertainty.
- $\begin{bmatrix} v \\ e \end{bmatrix} = M \begin{bmatrix} w \\ r \end{bmatrix}$ and $M = \begin{bmatrix} -T & KS \\ -GS & S \end{bmatrix}$



Step 2: Specify an IQC J for the gain bound $||\Delta||_\infty \leq \beta$.

$$\int_0^T \begin{bmatrix} v(t) \\ w(t) \end{bmatrix}^\top J \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} dt \geq 0 \quad \forall T \geq 0 \;\text{ with }\; J = \begin{bmatrix} \beta^2 & 0 \\ 0 & -1 \end{bmatrix}$$

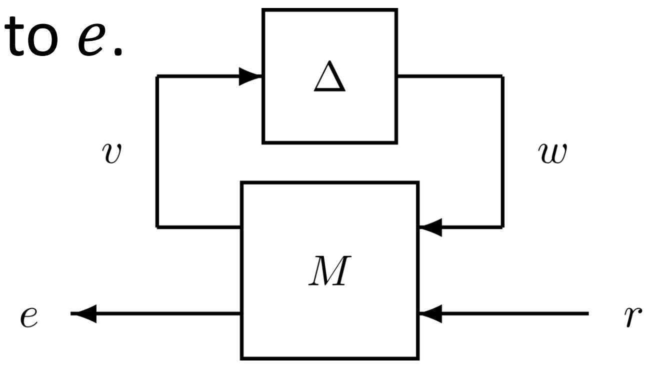
The constraint also holds when scaled by any $\lambda \geq 0$:

$$J = \lambda \begin{bmatrix} \beta^2 & 0 \\ 0 & -1 \end{bmatrix} \text{ where } \lambda \geq 0$$

This additional scaling reduces the conservatism in L_2 gain bound.

Numerical Example 2: LTI Uncertainty

Step 3: Write SDP to bound the gain from r to e.



- A state-space model for M is:

$$\begin{bmatrix} \dot{x}(t) \\ v(t) \\ e(t) \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x(t) \\ w(t) \\ r(t) \end{bmatrix}$$

- Dissipation inequality with $\lambda \geq 0,\ V(x) = x^\top P\, x$, $P \geq 0$ is:

$$\frac{d}{dt} V(x(t)) + \begin{bmatrix} e(t) \\ r(t) \end{bmatrix}^\top \begin{bmatrix} I & 0 \\ 0 & -\gamma^2 I \end{bmatrix} \begin{bmatrix} e(t) \\ r(t) \end{bmatrix} + \lambda \begin{bmatrix} v(t) \\ w(t) \end{bmatrix}^\top J \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} \leq 0$$

- Equivalent LMI form of the dissipation inequality is:

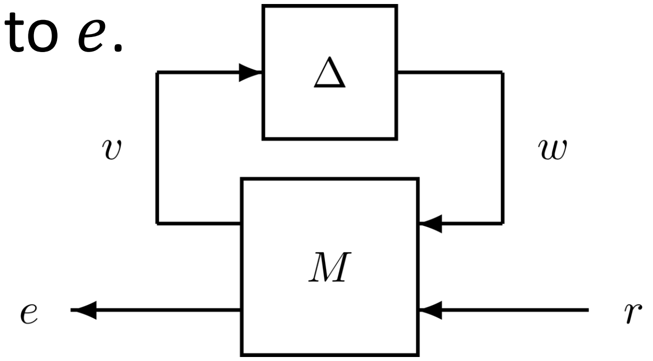
$$\begin{bmatrix} A^T P + PA & PB_1 & PB_2 \\ B_1^T P & 0 & 0 \\ B_2^T P & 0 & 0 \end{bmatrix} + (\cdot)^\top \begin{bmatrix} I & 0 \\ 0 & -\gamma^2 I \end{bmatrix} \begin{bmatrix} C_2 & D_{21} & D_{22} \\ 0 & 0 & I \end{bmatrix}$$

$$+ \lambda (\cdot)^\top J \begin{bmatrix} C_1 & D_{11} & D_{12} \\ 0 & I & 0 \end{bmatrix} \preceq 0$$

This is a version of a classical result known as the “Circle Criterion”.

Numerical Example 2: LTI Uncertainty

Step 3: Write SDP to bound the gain from r to e.



$$\min_{\lambda \geq 0, \gamma^2 > 0, P \geq 0} \gamma^2$$

subject to:

$$\begin{bmatrix} A^T P + PA & PB_1 & PB_2 \\ B_1^T P & 0 & 0 \\ B_2^T P & 0 & 0 \end{bmatrix} + (\cdot)^\top \begin{bmatrix} I & 0 \\ 0 & -\gamma^2 I \end{bmatrix} \begin{bmatrix} C_2 & D_{21} & D_{22} \\ 0 & 0 & I \end{bmatrix}$$

$$+ \lambda (\cdot)^\top J \begin{bmatrix} C_1 & D_{11} & D_{12} \\ 0 & I & 0 \end{bmatrix} \preceq 0$$

Numerical Example 2: LTI Uncertainty

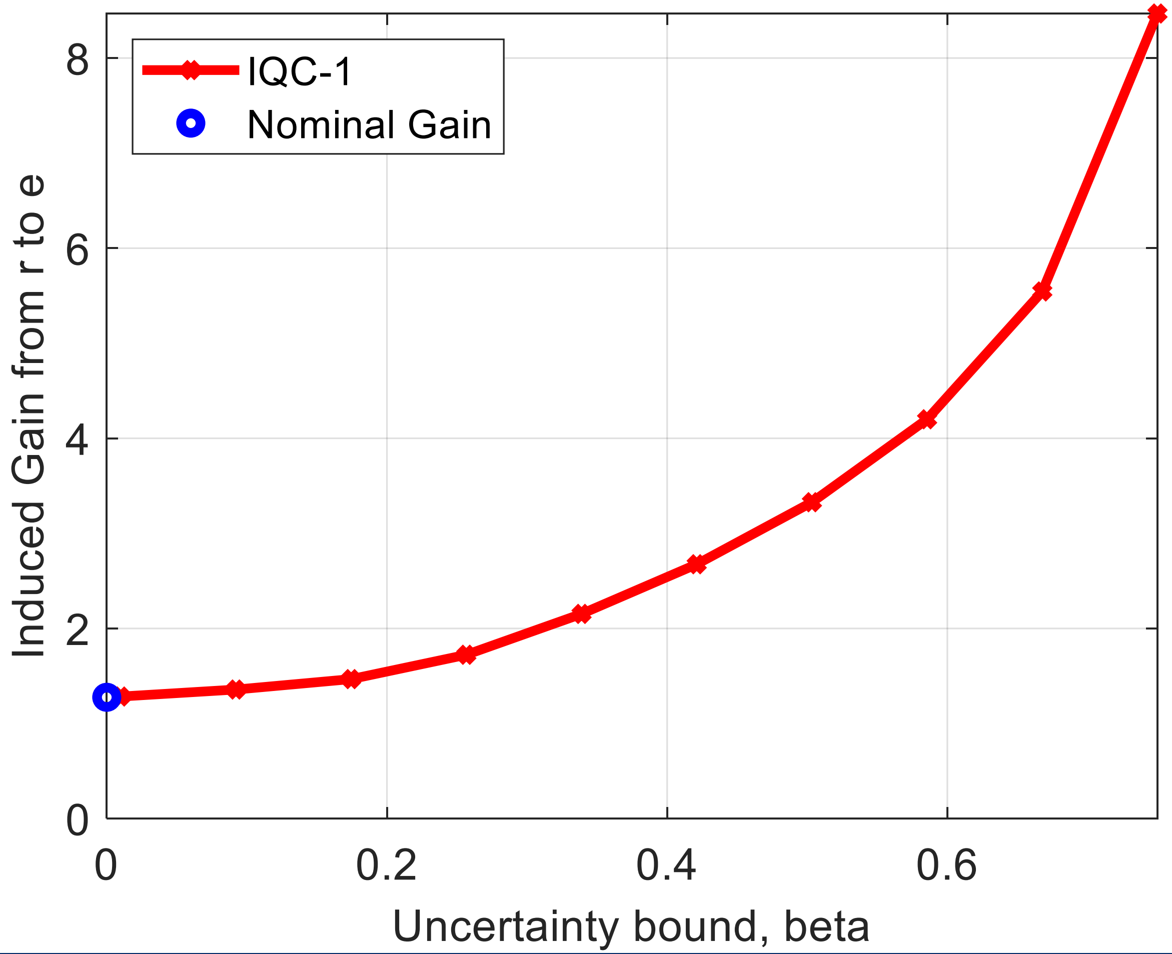
Step 3: Write SDP to bound the gain from r to e. CVX code:

```
% Form matrices used in SDP
Zwr = zeros(nw,nr); Zwx = zeros(nw,nx); Zw = zeros(nw);
Iw = eye(nw); Ir = eye(nr);
Lvw = [C1 D11 D12; Zwx Iw Zwr];
Le = [C2 D21 D22];
% Solve SDP using CVX
J = [beta^2, 0; 0 -1];  % This is the only change from Example 1
cvx_begin sdp quiet
       variable P(nx,nx) semidefinite;
       variable gsq(1,1);
       variable lambda nonnegative;
       [Am'*P+P*Am P*B1 P*B2; B1'*P Zw Zwr; B2'*P Zwr' -gsq*Ir] ...
       +lambda*(Lvw'*J*Lvw) + Le'*Le <=0;
       minimize(gsq)
cvx_end
```

Numerical Example 2: LTI Uncertainty

Induced L_2 gain vs. uncertainty bound $||\Delta||_\infty \leq \boldsymbol{\beta}$.

Note $\beta = 0$ corresponds to the nominal case ($\Delta = 0$). The red curve matches the nominal gain $||S||_\infty = 1.28$ when $\beta = 0$.



Numerical Example 2: LTI Uncertainty

The IQC with $J = \begin{bmatrix} \beta^2 & 0 \\ 0 & -1 \end{bmatrix}$ holds for any system with L_2 gain $\leq \beta$ (including NLTV systems). We can reduce the conservatism by using a dynamic IQC.

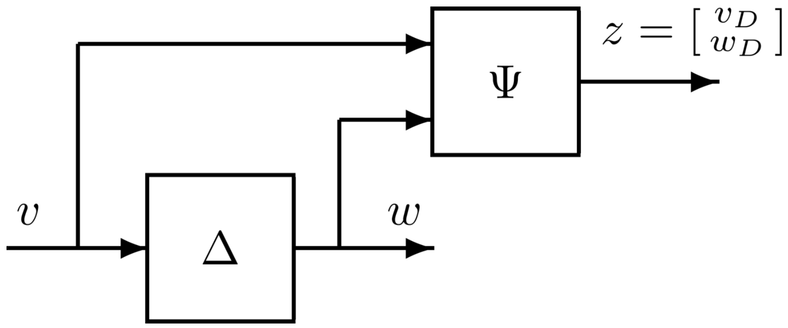
Step 2: Specify two IQCs for LTI uncertainty with the gain bound $||\Delta||_\infty \leq \beta$.

$$\int_0^T \begin{bmatrix} v(t) \\ w(t) \end{bmatrix}^\top J_1 \begin{bmatrix} v(t) \\ w(t) \end{bmatrix} dt \geq 0 \quad \forall T \geq 0 \text{ with } J_1 = \lambda_1 \begin{bmatrix} \beta^2 & 0 \\ 0 & -1 \end{bmatrix}, \; \lambda_1 \geq 0$$

$$\int_0^T z(t)^\top J_2 z(t)\, dt \geq 0 \quad \forall T \geq 0 \text{ with } J_2 = \lambda_2 \begin{bmatrix} \beta^2 & 0 \\ 0 & -1 \end{bmatrix}, \Psi = \begin{bmatrix} D & 0 \\ 0 & D \end{bmatrix}, \; \lambda_2 \geq 0$$

For illustration we choose $D(s) = \frac{10}{s+10}$.

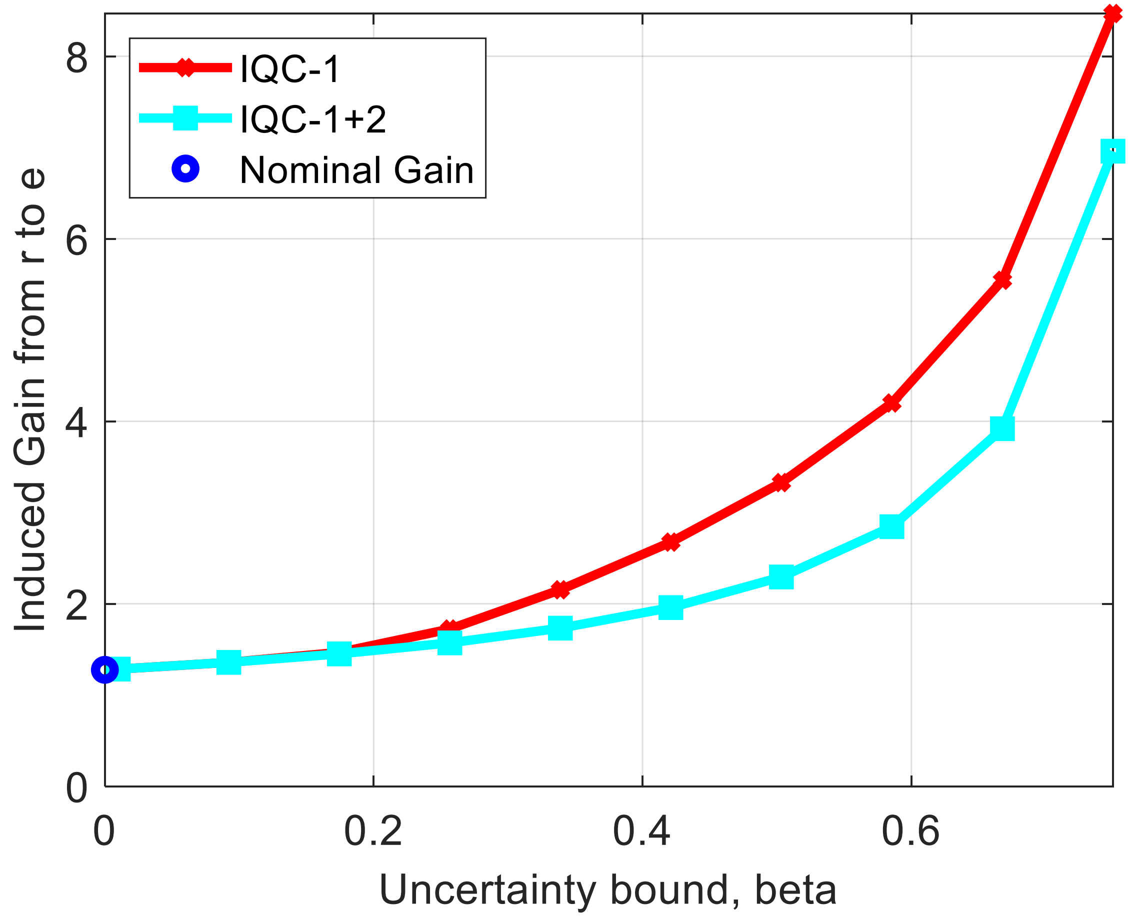
See posted code for implementation of the corresponding SDP.



Numerical Example 2: LTI Uncertainty

Induced L_2 gain vs. uncertainty bound $||\Delta||_\infty \leq \beta$.

The use of IQC 1 and 2 together reduces conservatism (reduces the bound) as compared to using only IQC 1.



Numerical Example 2: LTI Uncertainty

Induced L_2 gain vs. uncertainty bound $\|\Delta\|_\infty \leq \boldsymbol{\beta}$.

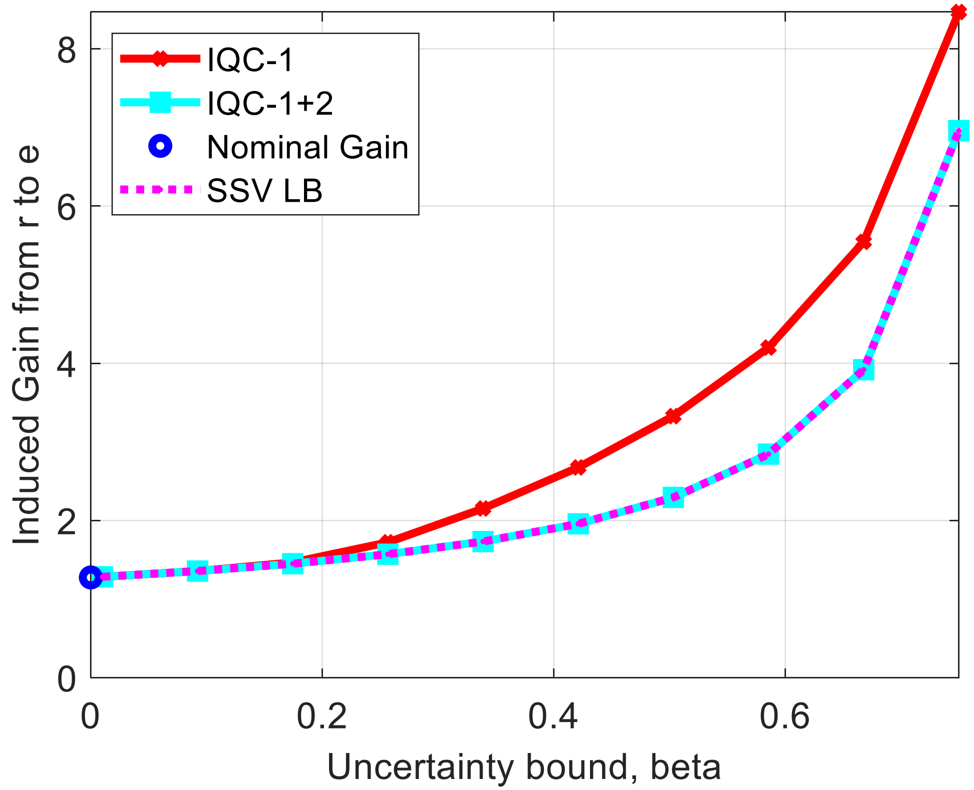
The use of IQC 1 and 2 together reduces conservatism (reduces the bound) as compared to using only IQC 1.

- The bound depends on the choice of $D(s) = \frac{10}{s+10}$. This choice was made for illustration by a heuristic search over systems of the form $D(s) = \frac{p}{s+p}$
- A more formal approach is to construct IQCs with many choices $\{D_i(s)\}_{i=1}^N$. Each IQC can be included in the dissipation inequality with its own non-negative scaling $\{\lambda_i\}_{i=1}^N$.
- The SDP will search for the "best" combination of scalings, i.e. search for the best combined IQC to minimize the bound.

Numerical Example 2: LTI Uncertainty

Induced L_2 gain vs. uncertainty bound $||\Delta||_\infty \leq \beta$.

The structured singular value (SSV), or μ, is a robustness theory specialized for LTI uncertainty. This can be used to compute a lower bound on the induced gain. The results below indicate that the IQC upper bound is not conservative.



Summary

In this lesson:

- We reviewed theory of convex optimization with a focus on the special class of semidefinite programs (SDPs).
- We introduced software tools for solving SDPs and related computational issues that arise when using these tools.
- We presented numerical examples that that demonstrate the use of dissipation inequalities and QCs/IQCs.

Next lesson: Miscellaneous topics related to dissipation inequalities and IQCs.

Further Reading

Convex Optimization / Semidefinite Programming:

- Boyd, Vandeberghe, Convex Optimization, Cambridge Univ. Press, 2004.
- Boyd, El Ghaoui, Feron, Balakrishnan, Linear Matrix Inequalities in System and Control Theory, SIAM 1994.

An (incomplete) list of solvers/parsers

- CVX (for Matlab): https://cvxr.com/cvx/
- CVXPY (for Python): https://www.cvxpy.org/
- Mosek: https://www.mosek.com/
- SDPT3: https://github.com/sqlp/sdpt3
- LMILab: https://www.mathworks.com/help/robust/lmis.html

KYP Lemma:

- Rantzer, On the Kalman—Yakubovich—Popov lemma, SCL, 1996

Self-Study Problems

A file with three numerical examples problems is posted on the Web site. This also includes solutions for you to check your work.

sites.google.com/berkeley.edu/dissipation-iqc