Computing Quadratic Invariants

Pierre Roux[1,2,3] **Pierre-Loïc Garoche**[1]

October 4, 2014

[1]ONERA – The French Aerospace Lab, Toulouse, France

[2]ISAE, University of Toulouse, Toulouse, France

[3]currently visiting CU Boulder

Control Command Systems

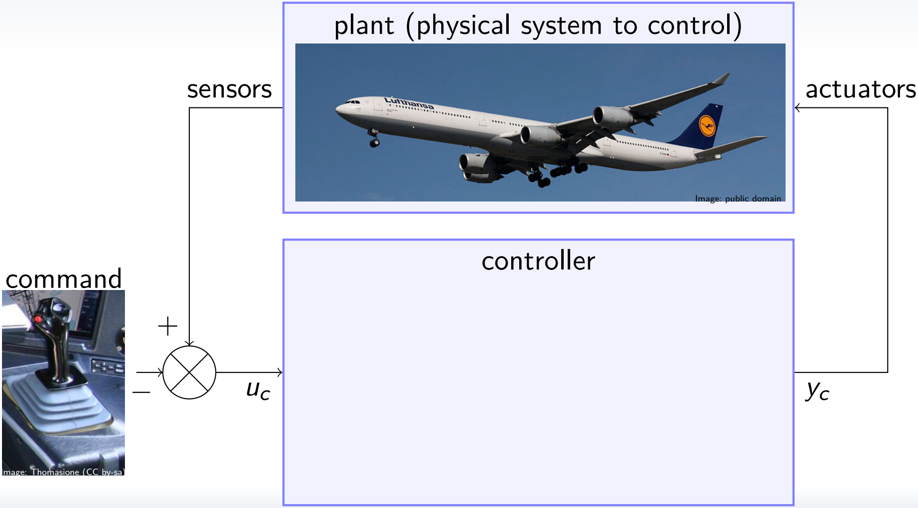
plant (physical system to control)

Image: public domain

command

Image: Thomasione (CC by-sa)

Control Command Systems



Control Command Systems

plant (physical system to control)

Image: public domain

sensors

actuators

command

Image: Thomasione (CC by-sa)

$+$

$-$

u_c

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

Control Command Systems

plant (physical system to control)

Image: public domain

sensors

actuators

command

Image: Thomasione (CC by-sa)

+

−

u_c

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

Control Command Systems

plant (physical system to control)

Image: public domain

sensors

actuators

command

Image: Thomasione (CC by-sa)

+

−

u_c

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

Control Command Systems

plant (physical system to control)

Image: public domain

sensors

actuators

command

Image: Thomasione (CC by-sa)

$+$

$-$

u_c

y_c

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

Control Command Systems

plant (physical system to control)

Image: public domain

sensors

actuators

command

Image: Thomasione (CC by-sa)

$+$

$-$

u_c

y_c

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

Control Command Systems

plant (physical system to control)

Image: public domain

sensors

actuators

command

Image: Thomasione (CC by-sa)

$+$

$-$

u_c

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

critical system (human lifes at stake)

Control Command Systems

plant (physical system to control)

Image: public domain

sensors

actuators

command

Image: Thomasione (CC by-sa)

$+$

$-$

u_c

controller

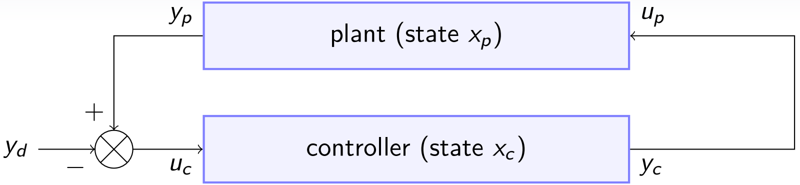
```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

critical system (human lifes at stake) $\Rightarrow$ verification

Stability Proofs

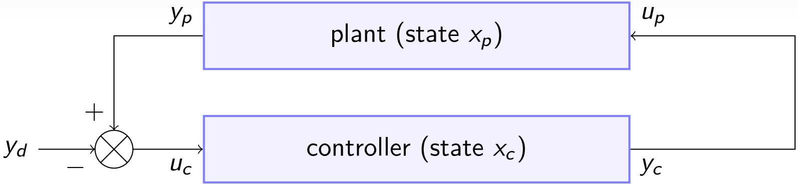
- Closed loop stability



command y_d bounded $\Rightarrow$ x_c and x_p bounded (hence y_c and y_p bounded)

Stability Proofs

- Closed loop stability

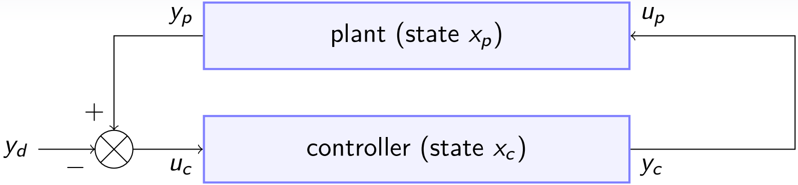


command y_d bounded $\Rightarrow$ x_c and x_p bounded (hence y_c and y_p bounded)

Intuitively: plane stays close from pilot orders.

Stability Proofs

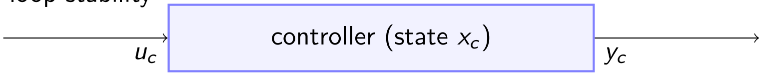
- Closed loop stability



command y_d bounded $\Rightarrow$ x_c and x_p bounded (hence y_c and y_p bounded)

Intuitively: plane stays close from pilot orders.

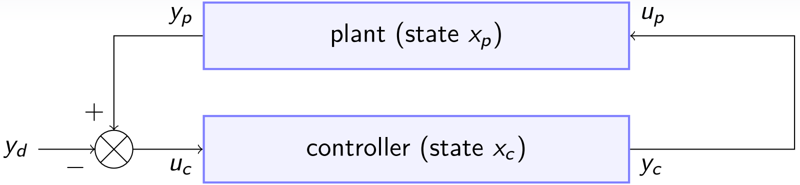
- Open loop stability



input u_c bounded $\Rightarrow$ x_c bounded (hence y_c bounded)

Stability Proofs

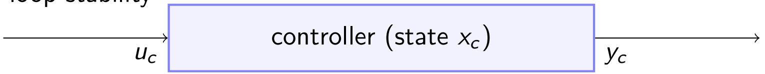
- Closed loop stability



command y_d bounded $\Rightarrow$ x_c and x_p bounded (hence y_c and y_p bounded)

Intuitively: plane stays close from pilot orders.

- Open loop stability

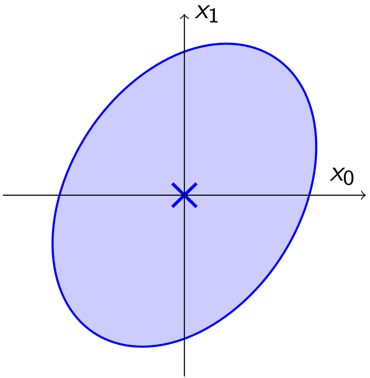


input u_c bounded $\Rightarrow$ x_c bounded (hence y_c bounded)

"Intuitively": no arithmetic overflow.

Invariants

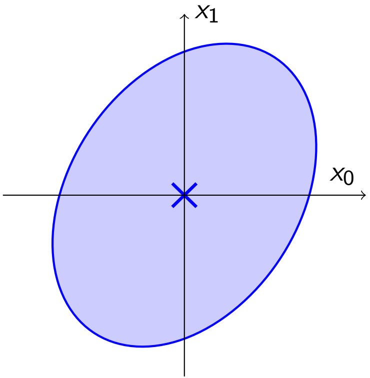
A set of points is an (inductive) invariant if it:



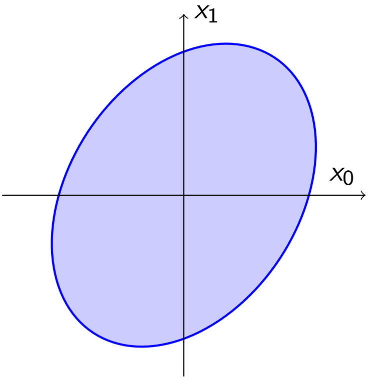
contains initial state
$((0, 0)$ for instance)

Invariants

A set of points is an (inductive) invariant if it:



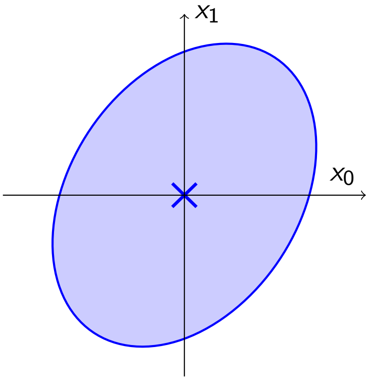
contains initial state
((0, 0) for instance)



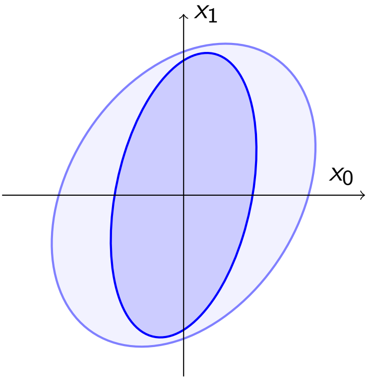
is stable in one step
(loop body)

Invariants

A set of points is an (inductive) invariant if it:



contains initial state
((0, 0) for instance)



is stable in one step
(loop body)

Example

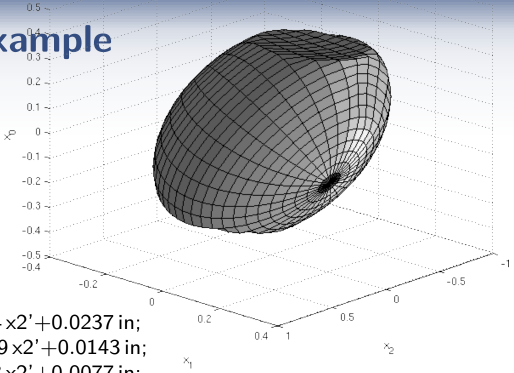
On following code:

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ≤ 0 do
   in := ?(−1, 1);
   x0' := x0; x1' := x1; x2' := x2;
   x0 := 0.9379 x0'−0.0381 x1'−0.0414 x2'+0.0237 in;
   x1 := −0.0404 x0'+0.968 x1'−0.0179 x2'+0.0143 in;
   x2 := 0.0142 x0'−0.0197 x1'+0.9823 x2'+0.0077 in;
od
```

our tool automatically proves:

$|x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251$

Example



On following code:

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ≤ 0 do
   in := ?(−1, 1);
   x0' := x0; x1' := x1; x2' := x2;
   x0 := 0.9379 x0'−0.0381 x1'−0.0414 x2'+0.0237 in;
   x1 := −0.0404 x0'+0.968 x1'−0.0179 x2'+0.0143 in;
   x2 := 0.0142 x0'−0.0197 x1'+0.9823 x2'+0.0077 in;
od
```

our tool automatically proves:

$|x_0| \leq 0.4236 \wedge |x_1| \leq 0.3371 \wedge |x_2| \leq 0.5251$

by producing the invariant:

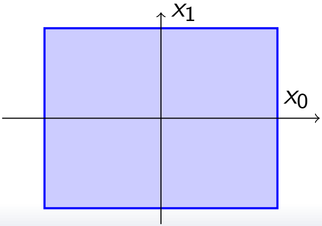
$6.2547x_0^2 + 12.1868x_1^2 + 3.8775x_2^2 - 10.61x_0x_1 - 2.4306x_0x_2 + 2.4182x_1x_2 \leq 1.0029$
$\wedge\, x_0^2 \leq 0.1795 \wedge x_1^2 \leq 0.1136 \wedge x_2^2 \leq 0.2757.$

Quadratic invariants

- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.

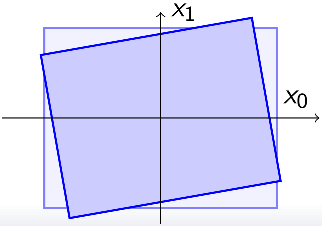
Quadratic invariants

- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



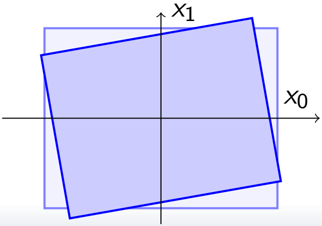
Quadratic invariants

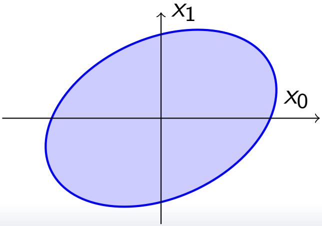
- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



Quadratic invariants

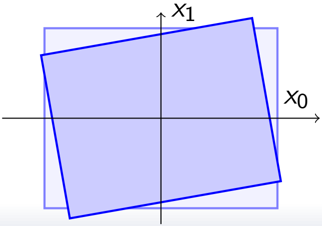
- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.
- Control theorists know for long that **quadratic invariants** are a good fit for linear systems.

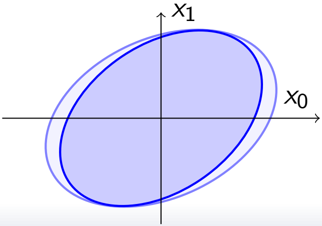




Quadratic invariants

- **Linear invariants** commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.
- Control theorists know for long that **quadratic invariants** are a good fit for linear systems.





Quadratic Invariants

Remark

Reachable state space is usually **not** an ellipsoid.

Quadratic Invariants

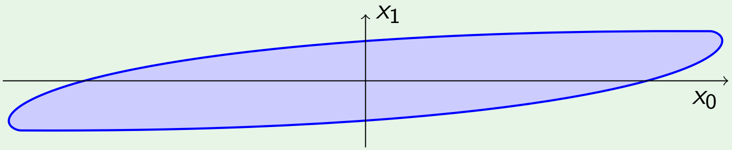
Remark

Reachable state space is usually **not** an ellipsoid.

Example

$x_0 := 0$ and $x_{k+1} := Ax_k + Bu_k$ where $\|u_k\|_\infty \leq 1$ and

$$A := \begin{bmatrix} 0.92565 & -0.0935 \\ 0.00935 & 0.935 \end{bmatrix} \qquad B := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



Quadratic Invariants

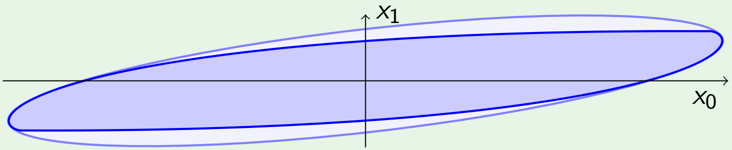
Remark

Reachable state space is usually **not** an ellipsoid.

Example

$x_0 := 0$ and $x_{k+1} := Ax_k + Bu_k$ where $\|u_k\|_\infty \leq 1$ and

$$A := \begin{bmatrix} 0.92565 & -0.0935 \\ 0.00935 & 0.935 \end{bmatrix} \qquad B := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



But remains reasonably close.

Lyapunov Stability

Theorem

For $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, the sequence

$$\begin{cases} x_0 \in \mathbb{R}^n \\ x_{k+1} = A x_k + B u_k \end{cases}$$

is bounded for all $u \in (\mathbb{R}^p)^{\mathbb{N}}$ such that for all $k \in \mathbb{N}$, $||u_k||_\infty \leq 1$ if and only if there exists $P \in \mathbb{R}^{n \times n}$ positive definite such that

$$P - A^T P A \succ 0$$

where $M \succ 0$ means that for all $x \in \mathbb{R}^n$: $x \neq 0 \Rightarrow x^T M x > 0$.

Lyapunov Stability, Invariant

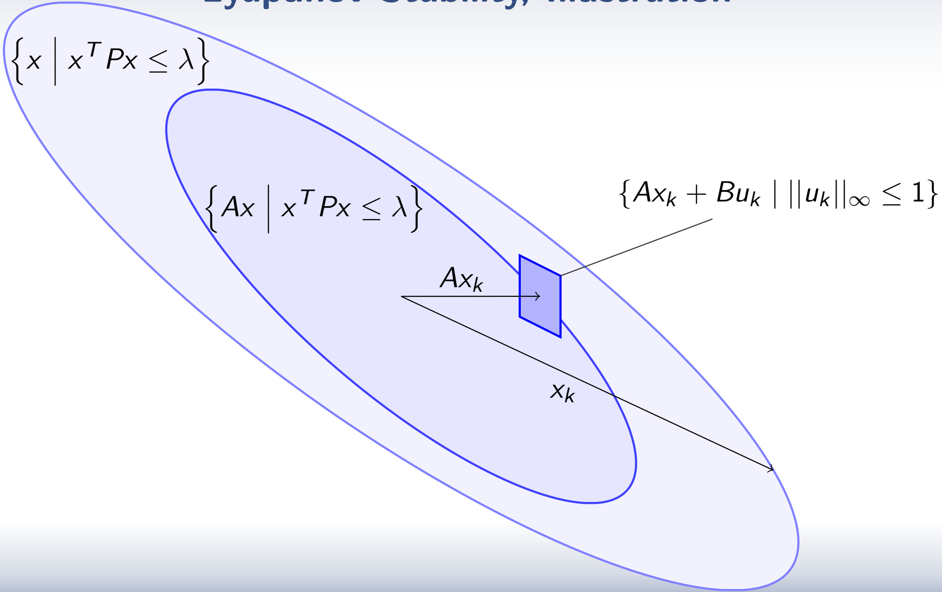
Invariant ellipsoid

Moreover, there exists a $\lambda > 0$ such that
x remains in the ellipsoid $\left\{x \in \mathbb{R}^n \,\middle|\, x^T P x \leq \lambda\right\}$.

In Computer Science Language

The property "$x^T P x \leq \lambda$" is a **loop invariant**.

Lyapunov Stability, Illustration



Tools

To solve the Lyapunov equation $P - A^T P A \succ 0$:

Semidefinite Programming

Minimize linear objective function of variables y_i
under constraint

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

the A_i are known matrices
and $M \succeq 0$ means $x^T M x \geq 0$ for all vector x.

- "Efficient" solvers exist;

Tools

To solve the Lyapunov equation $P - A^T P A \succ 0$:

Semidefinite Programming

Minimize linear objective function of variables y_i
under constraint

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

the A_i are known matrices
and $M \succeq 0$ means $x^T M x \geq 0$ for all vector x.

- "Efficient" solvers exist;
- Unknown matrices: Y can be decomposed as $\sum_{i,j} y_{i,j} E^{i,j}$;

Tools

To solve the Lyapunov equation $P - A^T P A \succ 0$:

Semidefinite Programming

Minimize linear objective function of variables y_i
under constraint

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

the A_i are known matrices
and $M \succeq 0$ means $x^T M x \geq 0$ for all vector x.

- "Efficient" solvers exist;
- Unknown matrices: Y can be decomposed as $\sum_{i,j} y_{i,j} E^{i,j}$;

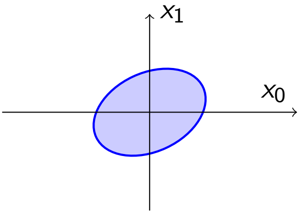
$\Rightarrow$ Lyapunov equation is numerically solvable.

Shape of the Ellipsoid

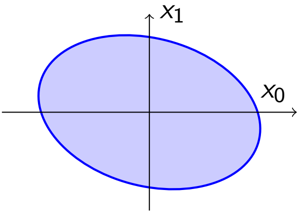
- We look for $P \succ 0$ such that $P - A^T P A \succ 0$.

Shape of the Ellipsoid

- We look for $P \succ 0$ such that $P - A^T P A \succ 0$.
- Then for λ such that $x^T P x \leq \lambda$ is invariant.
- The ellipsoid $\left\{x \mid x^T P x \leq \lambda\right\}$ should be "small".

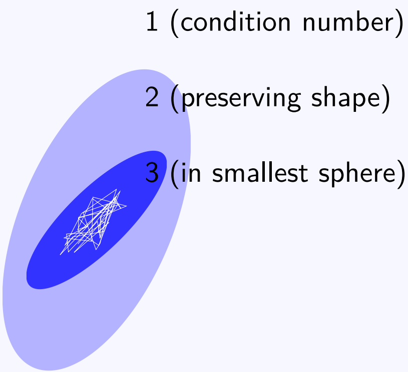


is better than



$\Rightarrow$ various heuristics tried.

Example



Experimental Results

	H.	Bounds	Reachable
Ex. 2 n=4, 1 input	1 2 3	1661, 1795, 843, 1288 5.50, 6.71, 2.20, 3.44 1.69, 1.92, 2.13, 2.42	1.42, 1.42, 1.00, 1.00
Ex. 3 Lead-lag controller n=2, 1 input	1 2 3	60.93, 60.52 36.55, 35.50 28.83, 25.85	3.97, 20.00
Ex. 4 LQG regulator n=3, 1 input	1 2 3	1.26, 1.26, 1.26 1.21, 1.23, 1.06 0.68, 0.41, 0.28	0.32, 0.24, 0.22
Ex. 5 Coupled mass system n=4, 2 inputs	1 2 3	4980, 5061, 4768, 5693 6.37, 6.20, 6.07, 9.57 4.97, 4.90, 4.77, 4.63	2.79, 2.73, 3.50, 3.30
Ex. 6 Low-pass filter n=5, 1 input	1 2 3	253, 261, 251, 280, 286 3.11, 4.30, 4.15, 8.16, 8.81 2.21, 1.10, 1.87, 1.98, 2.83	1.42, 0.91, 1.44, 1.52, 2.14

Experimental Results, Analysis Times

	Heuristic	t_P (s)	t_λ (s)	$t_{valid.}$ (s)	t_{total} (s)
Ex. 2 n=4, 1 input	1	0.10	0.63	0.02	0.75
	2	0.21	0.37	0.01	0.59
	3	0.33	0.22	0.01	0.56
Ex. 3 Discretized lead-lag controller n=2, 1 input	1	0.08	0.47	0.03	0.60
	2	0.13	0.45	0.02	0.60
	3	0.16	0.21	0.02	0.39
Ex. 4 Linear quadratic gaussian regulator n=3, 1 input	1	0.08	0.33	0.02	0.43
	2	0.14	0.29	0.02	0.45
	3	0.16	0.22	0.02	0.40
Ex. 5 Controller for a coupled mass system n=4, 2 inputs	1	0.09	0.76	0.03	0.88
	2	0.17	0.43	0.03	0.63
	3	0.27	0.23	0.03	0.53
Ex. 6 Butterworth low-pass filter n=5, 1 input	1	0.11	0.65	0.03	0.79
	2	0.22	0.37	0.02	0.61
	3	0.56	0.25	0.02	0.83

On an Intel Core2 @ 2.66GHz.

Floating Point Issues

For efficiency, use of floating point arithmetic, $\Rightarrow$ rounding errors:

Floating Point Issues

For efficiency, use of floating point arithmetic, $\Rightarrow$ rounding errors:

- In the analyzer:
  - Checking invariant $\left\{x \;\middle|\; x^T P x \leq \lambda\right\}$ amounts to **positive definiteness** of (for some $\tau \geq 0$ and $\lambda_i \geq 0$):

$$\begin{bmatrix} -A^T PA & -A^T PB & 0 \\ -B^T PA & -B^T PB & 0 \\ 0 & 0 & \lambda \end{bmatrix} - \tau \begin{bmatrix} -P & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \lambda \end{bmatrix} - \sum_{i=0}^{p-1} \lambda_i \begin{bmatrix} 0 & 0 & 0 \\ 0 & -E^{i,i} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

  - Done by bounding rounding errors in a **Cholesky decomposition**.
  - Hence an efficient soundness check (in $\mathrm{O}\left(n^3\right)$ floating point operations for an $n \times n$ matrix).
  - Proved in COQ (paper proof: 6 pages $\rightsquigarrow$ 3,8 kloc of COQ).

Floating Point Issues

- In the analyzed controller:

Theorem

If $x^T P x \leq \lambda$, $\|u\|_\infty \leq 1$ and $(Ax + Bu)^T P (Ax + Bu) \leq \lambda'$, then

$$\mathrm{fl}(Ax + Bu)^T P \, \mathrm{fl}(Ax + Bu) \leq \left(\sqrt{\lambda'} + \sqrt{\lambda} a + b\right)^2$$

with a and b (very small) constants computed from A, B and P.

Proved in COQ (paper proof: 4 pages $\rightsquigarrow$ 3,2 kloc of COQ).

Open Directions

- Invariant generation:
  - Handling disjunctions in loop (e.g., saturations, sometimes already works with policy iterations).
  - Polynomial invariants.

- Closed loop system.

- Other properties of interest for control theorists:
  - Robustness (generalization of phase and gain margins).
  - Performance (overshoot, convergence speed).

Questions

Thank you for your attention!

?