Formal Analysis of Robustness at Model and Code Level

Timothy Wang*,Romain Jobredeau†, Pierre-Loïc Garoche♮,Pierre Roux♮,Éric Féron†
*: UTRC United Tech Research Center
♮: ONERA - The French aerospace Lab, Toulouse
†: Georgia Tech

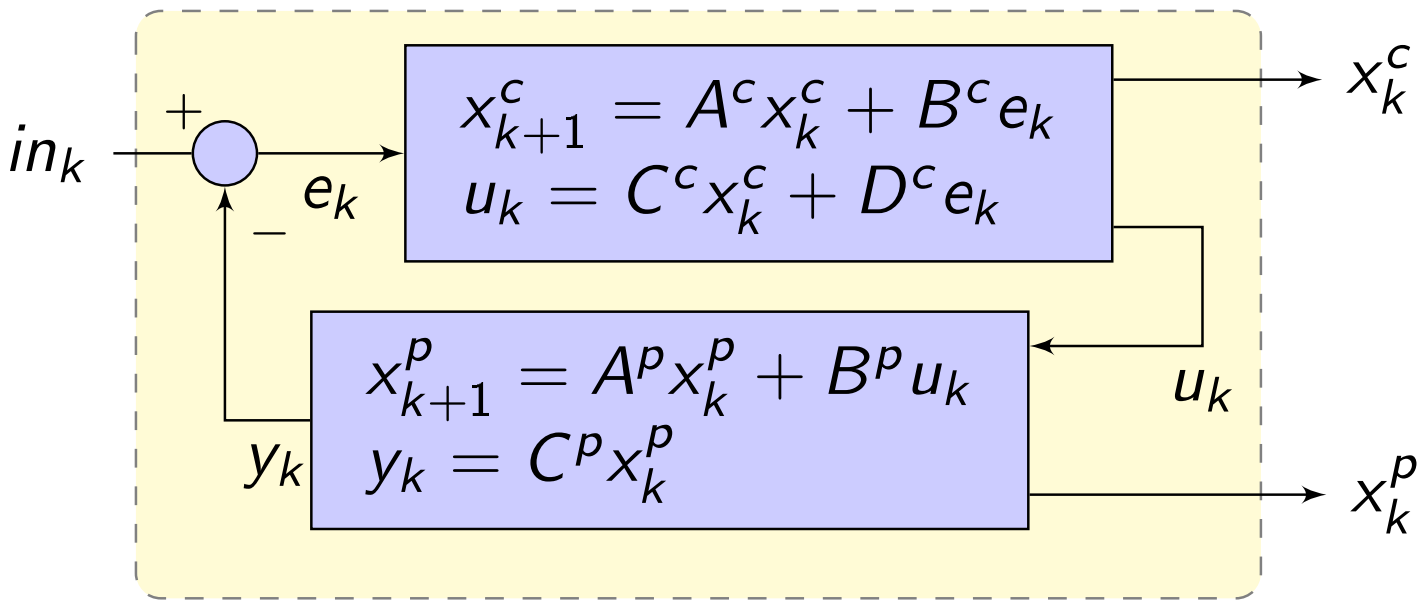
Journées CAFEIN 2016 – Fev 2016

Contents

Closed-loop linear system

$$\begin{cases} x^c_{k+1} = A^c x^c_k + B^c e_k \\ u_k = C^c x^c_k + D^c e_k \end{cases} \qquad \begin{cases} x^p_{k+1} = A^p x^p_k + B^p u_k \\ y_k = C^p x^p_k \end{cases}$$

$$e_k = in_k - y_k$$



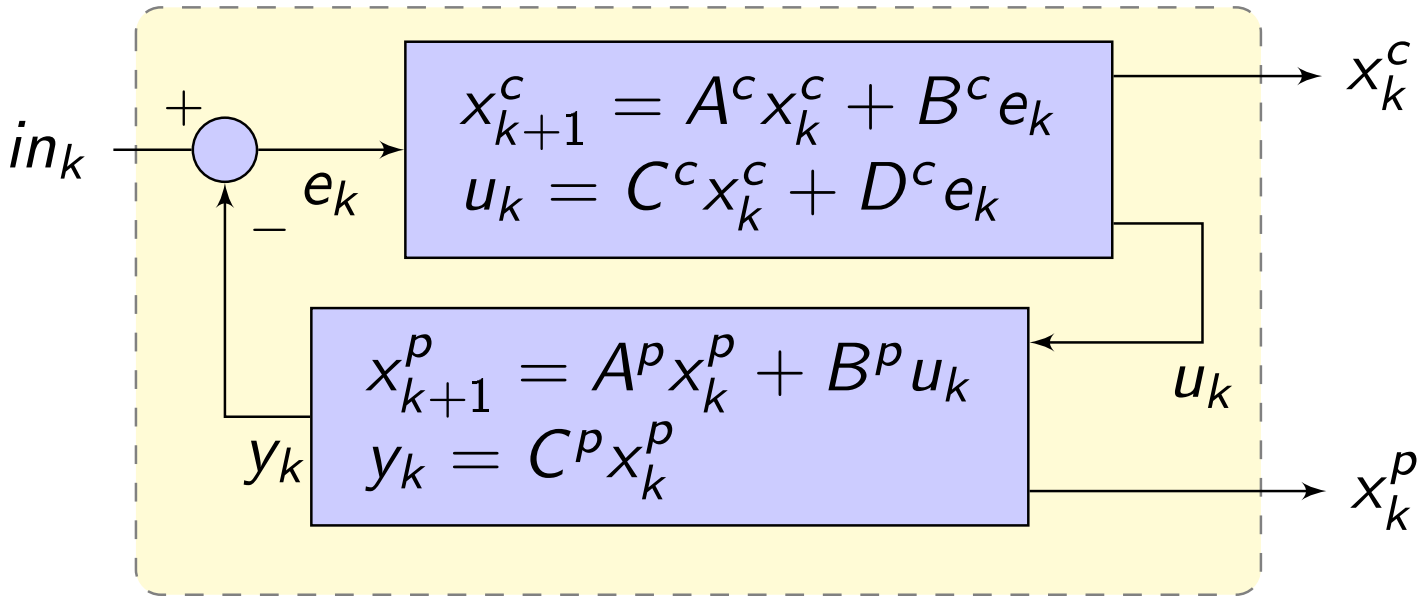
Stability

Open loop system stability:

$$\|e\|_\infty \leq 1 \implies x^{c\mathsf{T}} P^o x^c \leq 1 \text{ ie. } \|x^c\|_\infty \ll \infty$$

Closed loop system stability:

$$\|in\|_\infty \leq 1 \implies \mathbf{x}^\mathsf{T} P^c \mathbf{x} \leq 1 \text{ ie. } \|x\|_\infty \ll \infty$$



Z-transform of a discrete signal

- a discrete-time signal $(x_k)_{k\in\mathbb{N}}$
- a function $X(z)$ over the complex field or the z-domain.

The z-transform is the discrete-time analogue of the Laplace transform

$$\mathcal{Z} : x \mapsto \sum_{k=0}^{\infty} x_k z^{-k}.$$

Using Z-expression:

- Let $C(z)$ is a transfer function representation of the state-space system C.
- Let $U(z)$ a transfer function denoting an input signal
- Then $Y(z)C(z)$ denotes the output signal.
- One can also define $C(z)$ as $\frac{Y(z)}{U(z)}$

Z-transform for linear systems

$A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{k\times n}, D \in \mathbb{R}^{k\times m},$

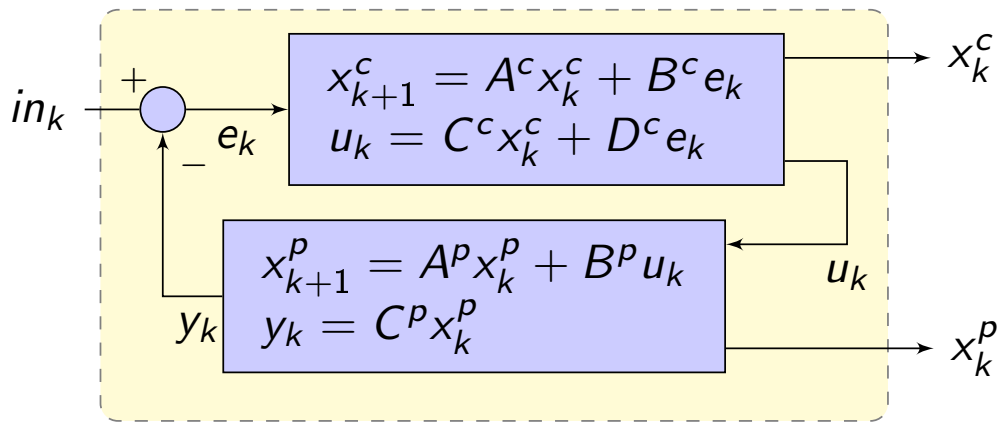
$$\begin{cases} x_{k+1} = Ax_k + Bin_k \\ out_k = Cx_k + Din_k \end{cases} \quad \mapsto \quad C\left(zI_{n\times n} - A\right)^{-1} B + D$$

In particular

$$x_{k+1} = x_k \quad \mapsto \quad \frac{1}{z}$$

In our setting the transfer function mapping $E(z)$ to $Y(z)$, aka the (open-)loop transfer function $L(z)$, is defined as

$$\frac{Y(z)}{E(z)} = \frac{Y(z)}{U(z)}\frac{U(z)}{E(z)} = P(z)C(z)$$



Z-transform for linear systems

Another example would be the closed-loop transfer function from $In(z) := \mathcal{Z}(in)$ to $Y(z)$ which is

$$\tilde{G}(z) = \frac{L(z)}{1 + L(z)}.$$

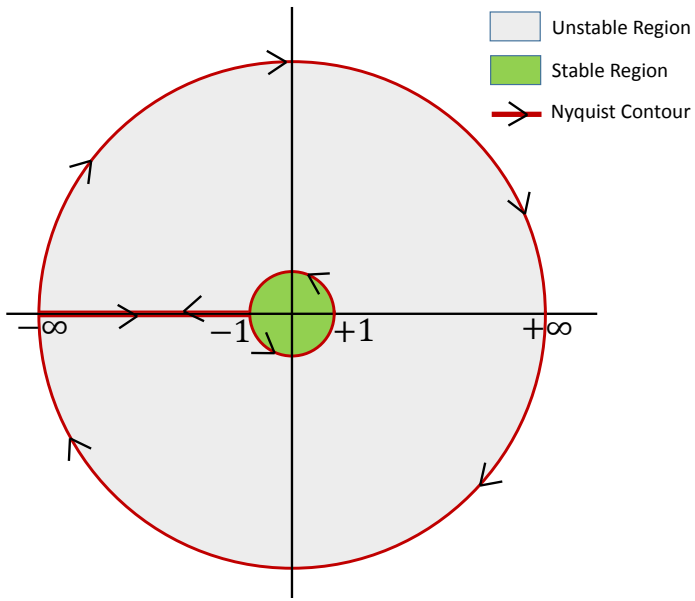
The formula can be deduced by noting that

$$\begin{aligned} & Y(z) = L(z)\left(In(z) - Y(z)\right) \\ \Longrightarrow\ & Y(z)\left(1 + L(z)\right) = L(z)In(z) \\ \Longrightarrow\ & \frac{Y(z)}{In(z)} = \frac{L(z)}{1 + L(z)}. \end{aligned}$$

Nyquist Plot

Nyquist plot: frequency response (magnitude and phase) of the loop transfer function to a sinusoidal input displayed using a polar coordinate system.

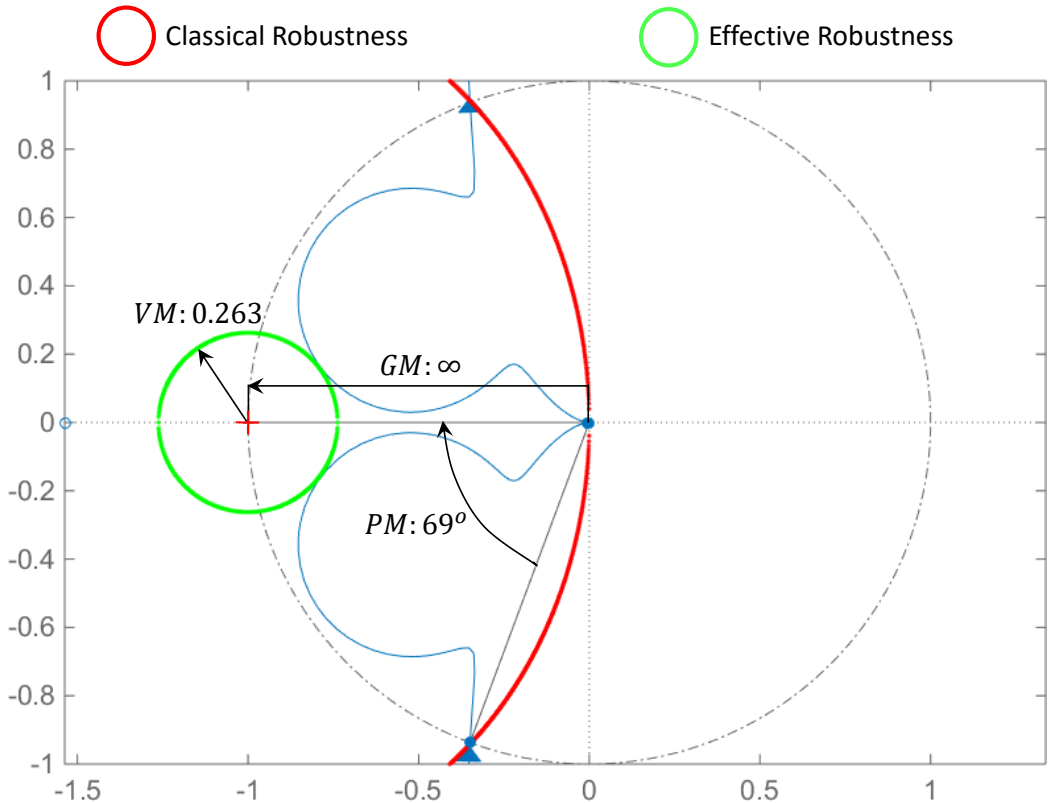
To construct the Nyquist plot, the loop transfer function $L(z)$ is evaluated along the Nyquist contour Γ. The Nyquist contour encircles the region outside of the unit disk (OUD) centered at the origin.



Nyquist Plot example

An example Nyquist plot for the loop transfer function

$$L(z) := \frac{7.552 \times 10^{-5} z^3 - 7.583 \times 10^{-5} z^2 - 7.454 \times 10^{-5} z + 7.488 \times 10^{-5}}{z^4 - 3.979 z^3 + 5.937 z^2 - 3.937 z + 0.979}$$



Nyquist Plot and Stability Criterion

Nyquist stability criterion for closed-loop stability of the system.

- Z_i number of OUD zeros of $L(z) + 1$
- P_i number of OUD poles of $L(z) + 1$

By Cauchy's principle of argument, the Nyquist plot should encircle clockwise[1] the $-1 + 0j$ point N_i number of times where

$$N_i = Z_i - P_i.$$

Then $\frac{L(z)}{1+L(z)}$ is stable: it does not diverge

[1] Counter-clockwise encirclement counts as negative.

Robustness margins: phase and gain

Phase margin

amount of clockwise rotation that can be applied to the Nyquist plot before it hits the critical point.
ϕ: clockwise angle between the point where the unit circle, centered at the origin, intersects with the Nyquist plot and $-1 + 0j$.

Gain margin

how much feedback gain the system can tolerate, *i.e.*, how much one can scale up the Nyquist plot radially before it intersects with the $-1 + 0j$ point.
Θ: $20 \log_{10} \frac{1}{x}$ where x is the magnitude of the Nyquist plot at the phase angle of π.

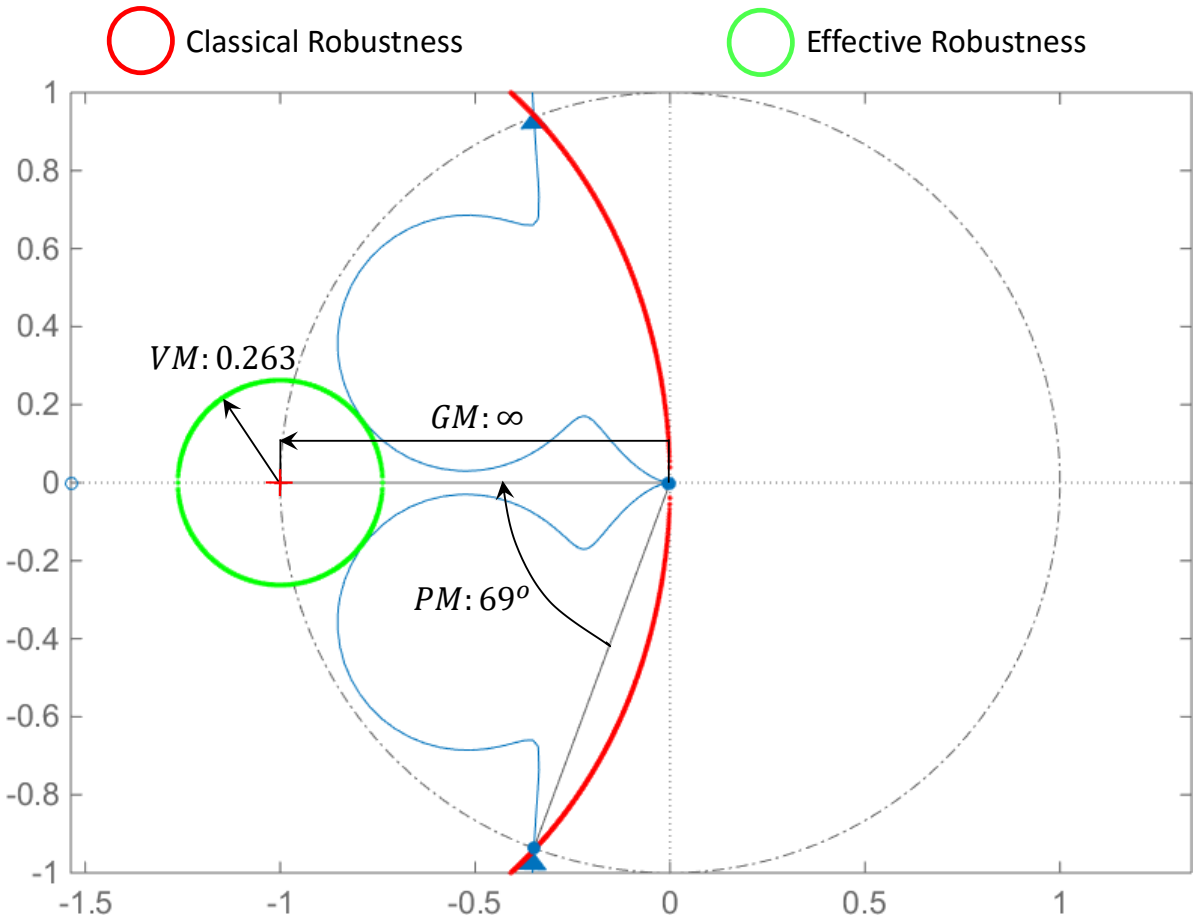
For good robustness, typical requirements:

$$\phi > 30^\circ \text{ and } \Theta > 3\text{db}.$$

Contents

A more effective robustness criteria: Vector margins

1. more faithful measure of the robustness.
2. compatible with time-domain,
   ie. expressible on the code as a quadratic invariant.
3. extends to MIMO systems



Vector margin definition

The vector margin: $\min_{z\in\Gamma} |L(z)+1|$
Remark that

$$\min_{z\in\Gamma} |L(z)+1| = \frac{1}{\max\limits_{z\in\Gamma} \frac{1}{|L(z)+1|}} = \frac{1}{\max\limits_{z\in\Gamma} \left|\frac{1}{L(z)+1}\right|} = \frac{1}{\max\limits_{z\in\Gamma} |S(z)|}.$$

where $S(z) = \frac{1}{1+L(z)}$ is sensitivity function of the closed-loop system.
Vector margin is then the inverse of the maximum modulus of the sensitivity function
The sensitivity function is a first-order approximation of the change in the output over the change in the input for the closed-loop system.

Sensitivity function

One can reformulate the transfer function as a linear system:

$$\begin{aligned} x_{k+1} &= A_s x_k + B_s in_k \\ e_k &= C_s x_k + D_s in_k \end{aligned}$$

where

$$A_s := \begin{bmatrix} A_c & -B_c C_p \\ B_p C_c & A_p - B_p D_c C_p \end{bmatrix} \quad B_s := \begin{bmatrix} B_c \\ B_p D_c \end{bmatrix}$$
$$C_s := \begin{bmatrix} 0 & -C_p \end{bmatrix} \qquad D_s := \begin{bmatrix} I \end{bmatrix}.$$

Property (Application of the bounded real lemma)

If there exists a positive-definite matrix P and $\gamma > 0$, such that

$$x_{k+1}^T P x_{k+1} - x_k^T P x_k \leq \gamma^2 \|in_k\|_2^2 - \|e_k\|_2^2$$

then $\max_{z \in \Gamma} |S(z)| \leq \gamma$.

Minimizing the maximum modulus of the sensitivity function

Previous inequality is a dissipativity condition and can be checked efficiently by solving a linear matrix inequality (LMI).

$$\begin{pmatrix} A_s^\mathsf{T} P A_s - P + C_s^\mathsf{T} C_s & A_s^\mathsf{T} P B_s + C_s^\mathsf{T} D_s \\ B_s^\mathsf{T} P A_s + D_s^\mathsf{T} C_s & D_s^\mathsf{T} D_s + B_s^\mathsf{T} P B_s - \gamma^2 I \end{pmatrix} \prec 0.$$

- A solution $P \succ 0$ and $\gamma > 0$ ensures that $\max_{z \in \Gamma} |S(z)| \leq \gamma$.
- Minimizing γ, we maximize the vector margin $\delta = \frac{1}{\gamma}$.

Robustness interpretation

Assuming we found valid $P \succ 0$ and $\gamma > 0$.
Summing the dissipativity condition from time 0 to any time T,

$$x_{T+1}^{\mathsf{T}} P x_{T+1} - x_0^{\mathsf{T}} P x_0 \leq \gamma^2 \left(\sum_{k=0}^{T} \|in_k\|_2^2 \right) - \sum_{k=0}^{T} \|e_k\|_2^2$$

and since P is positive definite, assuming $x_0 = 0$, we have

$$\sum_{k=0}^{T} \|e_k\|_2^2 \leq \gamma^2 \left(\sum_{k=0}^{T} \|in_k\|_2^2 \right).$$

Projection to phase and gain margins

Phase margins

Angle between the intersection of the Nyquist plot of the transfer function with the unit circle and the point $-1 + 0j$.
It is necessary larger than the angle between the intersection of the computed safe circle of radius δ with the unit circle and the point $-1 + 0j$

$$\phi_\delta = 2\ arcsin(\delta/2)$$

Gain margins

Acceptable scale of the Nyquist plot to avoid intersection with the point $-1 + 0j$.

$$\Theta_\delta = \frac{1}{1-\delta} \text{ or, expressed in } dB : \Theta_\delta = 20 \cdot log10 \left(\frac{1}{1-\delta}\right)$$

Contents

Floating point issues

Two main (and different) issues wrt floating-point arithmetic:

the analysis itself is performed in floating-point arithmetic, in particular the LMI is solved using approximate SDP solvers

the analyzed controller performs its computations using floating-point arithmetic rather than real numbers

Floating-Point Arithmetic in the Analysis

- we solve a convex SDP optimization problem:

  $$\text{linear objective} + \text{(LMI) constraints}$$

- the SDP solver, implemented with floating-point arithmetic, computes an approximate solution
  - the solution may not be the real optimum wrt objective
  - it may not satisfy *strictly* the constraints (ie. not a feasible solution)
  - In practice, returned values of P and γ^2, makes the LMI slightly not negative definite.
- we "pad" the initial problem $M \prec 0$ into $M + \epsilon I \prec 0$ with ϵ greater than solver precision, eg. $\epsilon := 10^{-7}$
- we check the soundness of the solution (P, γ) wrt the initial LMI.
  - LMI is instanciated into an exact matrix, computed with rational arithmetics
  - its positiveness is checked with a conservative Cholewski decomposition using floats (algorithm proved in Coq)

Floating-Point Arithmetic in the Controller

Computations of the controller being performed using floating-point arithmetic, rounding errors unavoidably occur and x_{k+1}^c is not exactly equal to $A_c x_k^c + B_c e_k$.
We can bound the floating point errors and consider a stronger condition:

Theorem

For reasonable dimensions, if

$$x_{k+1}^T P x_{k+1} - x_k^T P x_k + \epsilon \left(\|x_k\|_2^2 + \|in\|_2^2 \right) \leq \gamma^2 \|in_k\|_2^2 - \|e_k\|_2^2$$

then

$$\mathrm{fl}(x_{k+1})^T P \, \mathrm{fl}(x_{k+1}) - x_k^T P x_k \leq \gamma^2 \|in_k\|_2^2 - \|\mathrm{fl}(e_k)\|_2^2$$

where $\epsilon :=$ a function in system dimensions, matrices coefficiens and the floating point format relative error `eps` (about 2^{-53} for `Binary64`, ie. double)

Floating-Point Arithmetic in the Controller

Thus, instead of checking

$$\begin{pmatrix} A_s^\mathsf{T} P A_s - P + C_s^\mathsf{T} C_s & A_s^\mathsf{T} P B_s + C_s^\mathsf{T} D_s \\ B_s^\mathsf{T} P A_s + D_s^\mathsf{T} C_s & D_s^\mathsf{T} D_s + B_s^\mathsf{T} P B_s - \gamma^2 I \end{pmatrix} \prec 0.$$

we can check the stronger condition

$$\begin{pmatrix} A_s^\mathsf{T} P A_s - P + C_s^\mathsf{T} C_s + \epsilon I & A_s^\mathsf{T} P B_s + C_s^\mathsf{T} D_s \\ B_s^\mathsf{T} P A_s + D_s^\mathsf{T} C_s & D_s^\mathsf{T} D_s + B_s^\mathsf{T} P B_s - \gamma^2 I + \epsilon I \end{pmatrix} \prec 0.$$

In practice, $\epsilon \simeq 10^{-9}$ is small with respect to the ϵ already needed to compensate for the SDP solver precision.

Absolute underflow errors

Because of underflow errors, we can only prove

$$\mathrm{fl}(x_{k+1})^\mathsf{T} P\, \mathrm{fl}(x_{k+1}) - x_k^\mathsf{T} P x_k \leq \gamma^2 \|in_k\|_2^2 - \|\mathrm{fl}(e_k)\|_2^2 + \eta$$

where η is an absolute error, typically $\eta \simeq 10^{-300}$.
Characterization of the relative increase btw input and output becomes

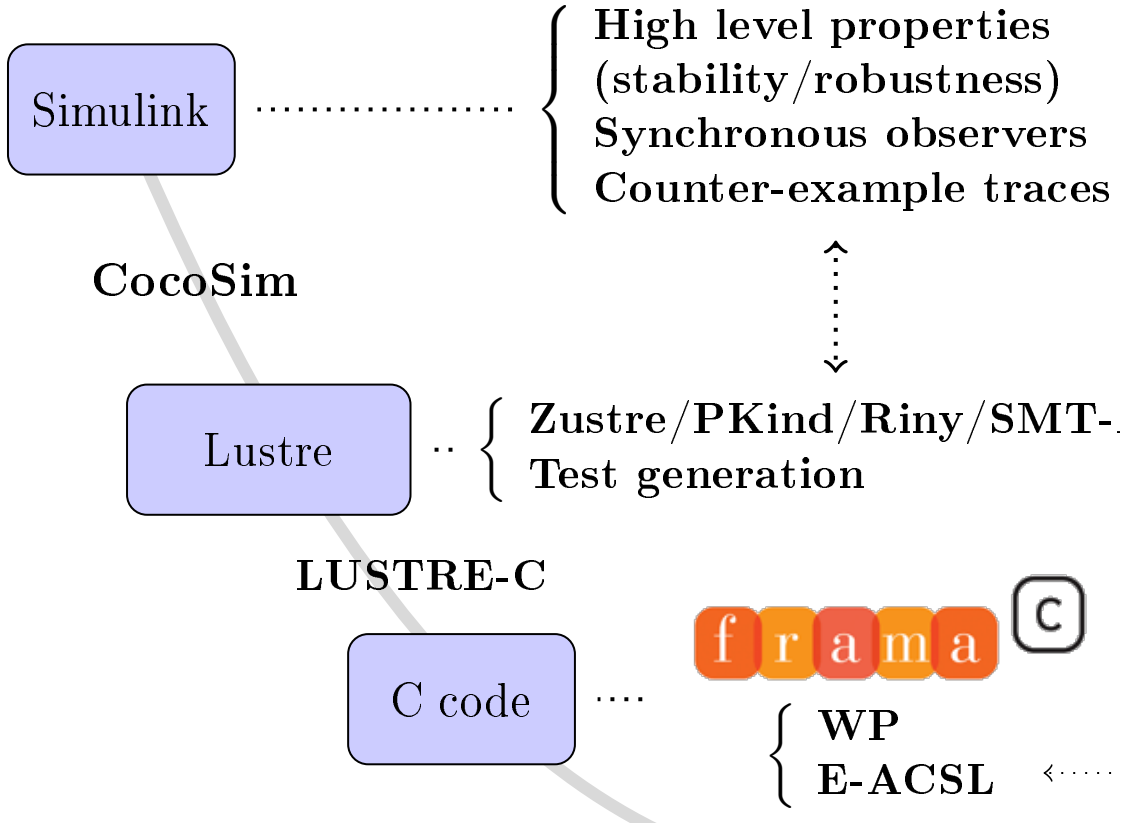
$$\sum_{k=0}^{T} \|e_k\|_2^2 \leq \gamma^2 \left(\sum_{k=0}^{T} \|in_k\|_2^2 \right) + (T+1)\eta.$$

Since η is tiny, it could impact the system only for very long execution. Eg. the flight commands of a plane typically operate at 100Hz and certainly no longer that 100 hours, meaning less than $T := 100 \times 3600 \times 100 \simeq 10^8$ iterations. Assuming it flies since the beginning of the universe, we will have $T \simeq 10^{20}$ iterations.

Contents

Experiments

- Meant to be integrated in an autocoding and analysis toolchain



- Models in Lustre are analyzed
- Ocaml SDP (OSDP) provides a Yalmip like frontend to perform convex optimization in Ocaml
- Provides a soundness checking wrt floating points computation

Current experiments are directly implemented in Ocaml

Cruise Control

First order linear approximation of a non-linear dynamical model for a car, accounting for rolling friction, aerodynamic drag and gravitational disturbance force:

$$\frac{d(v - v_e)}{dt} = -0.0101(v - v_e) + 1.32(u - u_e)$$

with v is the car velocity and u the throttle input.

Discrete-time version of the plant dynamics, with time step of $0.2s$

$$\begin{cases} x^p_{k+1} = 0.9998 * x^p_k + u_k \\ y_k = 0.0264 * x^p_k \end{cases}$$

Associated z-expression:

$$P(z) := \frac{0.0264}{z - 0.9998}.$$

Cruise Control – A first controller

A PI controller with
$$\begin{cases} \omega_0 = 1 \text{ (undamped natural frequency)} \\ \zeta = 1 \text{ (damping ratio of the dominant mode)} \end{cases}$$

$$C_1(z) := 1.51 + 0.757\frac{0.2}{z-1}.$$

Converting the z-expression to the associated linear system:

$$\begin{cases} x^c_{k+1} = x^c_k + u_k \\ u_k = 0.0150x^c_k + 1.5070u_k. \end{cases}$$

Phase and gain margins of $L(z) = C_1(z)P(z)$:

$$\Theta = 34dB \text{ and } \phi = 75^\circ$$

Cruise Control – A first controller

The sensitivity system is automatically built:

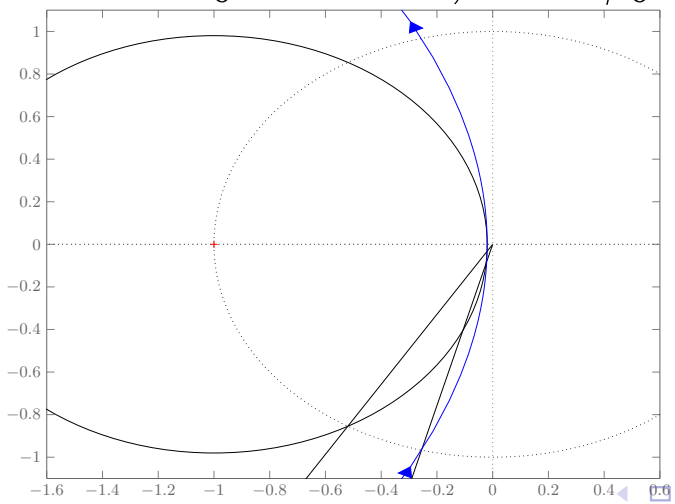
$$\begin{pmatrix} x^c_{k+1} \\ x^p_{k+1} \end{pmatrix} = \begin{bmatrix} 1.0000 & -0.0264 \\ 0.0150 & 0.9600 \end{bmatrix} \begin{pmatrix} x^p_k \\ x^c_k \end{pmatrix} + \begin{bmatrix} 1.0000 \\ 1.5070 \end{bmatrix} in_k$$

$$e_k = in_k - 0.0264 x^p.$$

And its vector margin computed using the LMI

$$P = \begin{bmatrix} 0.1865 & -0.1245 \\ -0.1245 & 0.1010 \end{bmatrix} \qquad \gamma = 1.0202\,.$$

We obtain $\delta = 0.9802$ $\Theta_\delta = 34dB,$ $\phi_\delta = 59°$



Cruise Control – A second controller

A less stable but more efficient controller

$$C_2(z) := \frac{2.72z^2 - 4.153z + 1.896}{z^2 - 1.844z + 0.8496}.$$

Tts associated linear system

$$\begin{cases} x^c_{k+1} = \begin{bmatrix} 1.8440 & -0.8496 \\ 1.0 & 0.0 \end{bmatrix} x^c_k + \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} u_k \\ u_k = \begin{bmatrix} 0.0366 & -0.0343 \end{bmatrix} x^c_k + \begin{bmatrix} 2.2720 \end{bmatrix} u_k. \end{cases}$$

Computed margins $\Theta = 31dB$ and $\phi = 84°$.

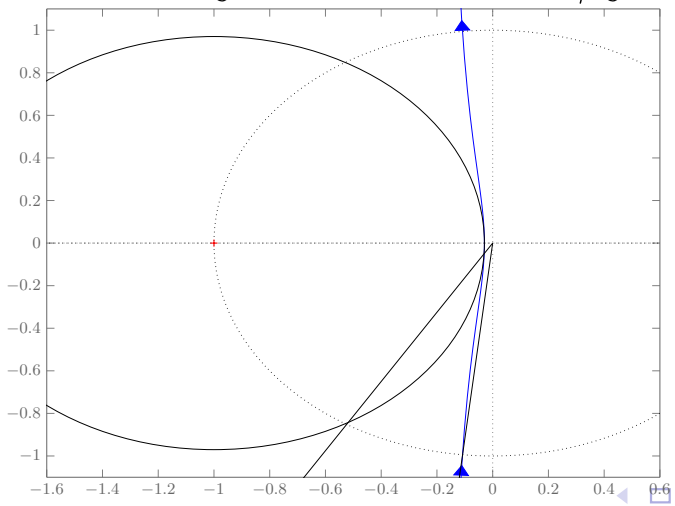
Cruise Control – A second first controller

$$\begin{pmatrix} x^c_{k+1} \\ x^p_{k+1} \end{pmatrix} = \begin{bmatrix} 1.8440 & -0.84 & -0.0264 \\ 1.0 & 0.0 & 0.0 \\ 0.0366 & -0.0343 & 0.9398 \end{bmatrix} \begin{pmatrix} x^p_k \\ x^c_k \end{pmatrix} + \begin{bmatrix} 1.0000 \\ 0.0 \\ 2.2720 \end{bmatrix} in_k$$

$$e_k = in_k - 0.0264 x^p.$$

$$P = \begin{bmatrix} 0.1189 & -0.1002 & -0.0529 \\ -0.1002 & 0.0849 & 0.0447 \\ -0.0529 & 0.0447 & 0.0355 \end{bmatrix}, \qquad \gamma = 1.0307\,.$$

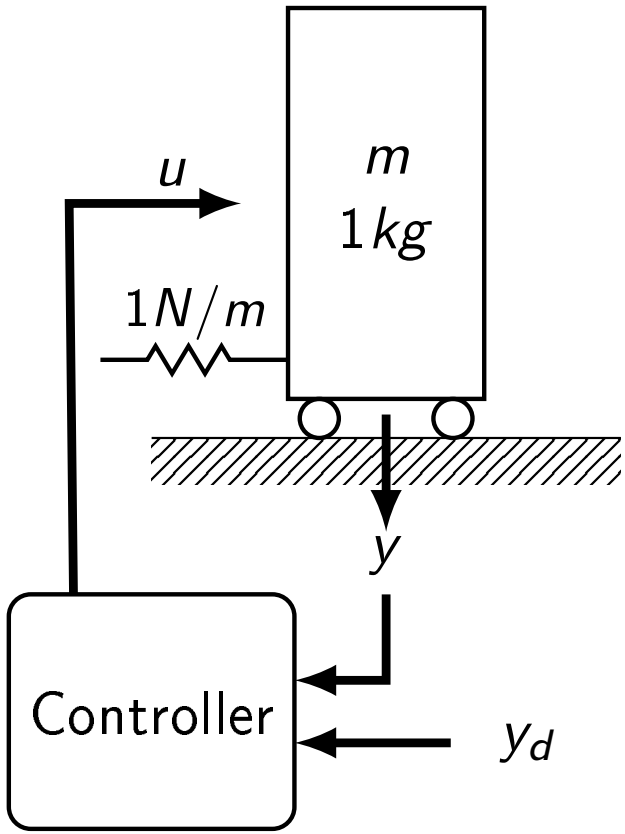
We obtain $\delta = 0.9703$ $\Theta_\delta = 31dB$ $\phi_\delta = 58°$.



Spring Mass Damper

$$A^p := \begin{bmatrix} 1 & 0.01 \\ -0.01 & 1 \end{bmatrix}, \quad B^p := \begin{bmatrix} 0.00005 \\ 0.01 \end{bmatrix},$$
$$C^p := \begin{bmatrix} 1 & 0 \end{bmatrix},$$

$$A^c := \begin{bmatrix} 0.4990 & -0.05 \\ 0.01 & 1 \end{bmatrix}, \quad B^c := \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$
$$C^c := \begin{bmatrix} 564.48 & 0 \end{bmatrix}, \quad D^c := 1280\,.$$



$\Theta = 17dB$ and $\phi = 49^\circ$
Analysis of the sensitivity system gives

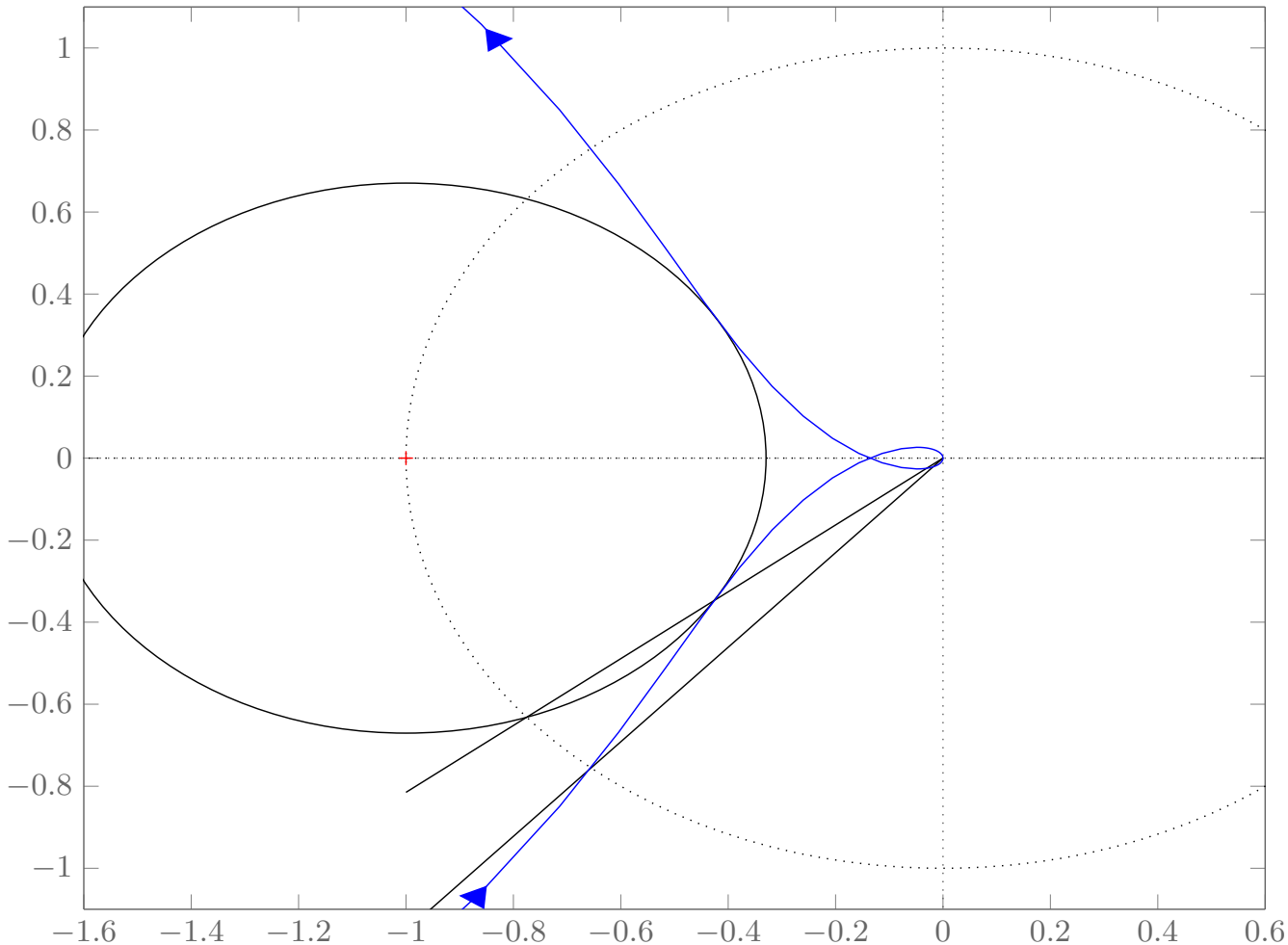
$$\gamma = 1.4914 \qquad P = \begin{bmatrix} 111.8330 & 88.4842 & -48.4990 & 8.8432 \\ 88.4842 & 278.5963 & -20.2482 & 6.9605 \\ -48.4990 & -20.2482 & 28.7964 & -3.7961 \\ 8.8432 & 6.9605 & -3.7961 & 0.7013 \end{bmatrix}$$

vector margin is $\delta = 1/\gamma = 0.6705$.

Spring Mass Damper – Results

$$\Theta = 17dB \quad \phi = 49^\circ$$

$$\delta = 1/\gamma = 0.6705: \quad \Theta_\delta = 10dB \quad \phi_\delta = 39^\circ$$



Contents

Possible future integration – Original lustre model

Lustre

```
node spring (u:real) returns (y:real);
var xp0, xp1: real ;
let
 xp0 = 0. -> pre xp0 + 0.01 * pre xp1 + 0.00005 * u;
 xp1 = 0. -> -0.01 * pre xp0 + pre xp1 + 0.01 * u;
 y = xp0;
tel

--@ plant: spring
node ctl (in,y:real) returns (u:real);
var e, xc0, xc1: real ;
let
 e = in - y;
 xc0 = 0. -> 0.4990 * pre xc0 - 0.05 * pre xc1 - e;
 xc1 = 0. -> 0.01 * pre xc0 + pre xc1;
 u = 564.48 * xc0 + 1280. * e;
tel
```

Possible future integration

Enriched model with computed properties

- Extraction of linear system of controller and plant
- Characterization of the sensitivity system
- Optimization of the LMI: computation of γ and P
- Annotation of the Lustre model

Lustre

```
--@ plant: spring
node ctl (in, y: real) returns (u: real);
--@ robustness/gamma: 1.4914;
--@ robustness/P: (computed P value);
```

Possible future integration

Header of the generated C code including node state description

C+ACSL

```
/*@logic matrix P = mat_of_4x4_scalar(
111.8330,  88.4842, -48.4990,  8.8432,
 88.4842, 278.5963, -20.2482,  6.9605,
-48.4990, -20.2482,  28.7964, -3.7961,
 8.8432,   6.9605,  -3.7961,  0.7013);
   logic real gamma = 1.4914; */

/*@ logic vector state(struct ctl_mem self) =
 vector_of_4_scalar (
   self->_reg.__ctl_3, self->_reg.__ctl_2,
   self->spec.plant._reg.__plant_3,
   self->spec.plant._reg.__plant_2);*/

/*@ predicate dissip(vector snxt, vector s, real in,
    real e, matrix P, real gamma) =
     normP(snxt, P) - normP(s, P) <= gamma**2 * in**2
         - e**2; */
```

Possible future integration

Header of the generated C code including node state description

C+ACSL

```
struct plant_mem {
  struct plant_reg {double __plant_2;
             double __plant_3; } _reg;
  struct _arrow_mem *ni_1; };
struct ctl_mem {
  struct ctl_reg {double __ctl_2;
           double __ctl_3; } _reg;
  struct _arrow_mem *ni_0;
  /*@ghost struct spec {
  struct plant_mem plant; } spec; */};
```

Possible future integration

Transfer function of the controller, including the plant description as annotation and the dissipativity property as function contract

C+ACSL

```
void ctl_step (double in, double y,
               double (*u),
               struct ctl_mem *self) {
_Bool __ctl_1;  // variables
double e; double xc0; double xc1;
// controller update
_arrow_step(1,0,&__ctl_1,self->ni_0);
if (__ctl_1) { xc1 = 0.; } else {
  xc1 = ((0.01 * self->_reg.__ctl_3) + self->_reg.
      __ctl_2); }
e = (in - y);
if (__ctl_1) { xc0 = 0.; } else {
  xc0 = (((0.499 * self->_reg.__ctl_3) - (0.05 *
      self->_reg.__ctl_2)) - e); }
*u = ((564.48 * xc0) + (1280. * e));
self->_reg.__ctl_3 = xc0;
self->_reg.__ctl_2 = xc1;
```

Possible future integration

Transfer function of the controller, including the plant description as annotation and the dissipativity property as function contract

C+ACSL

```
/*@ghost _Bool __plant_1; double xp0; double xp1;
//plant - restricted to state update
_arrow_step (1,0,&__plant_1,self->spec.plant.ni_1);
if (__plant_1) { xp0 = 0.; } else {
  xp0 = ((self->spec.plant._reg.__plant_3 + (0.01 *
      self->spec.plant._reg.__plant_2)) + (5e-05 * *u
      )); }
if (__plant_1) { xp1 = 0.; } else {
  xp1 = ((((- 0.01) * self->spec.plant._reg.__plant_3
      ) + self->spec.plant._reg.__plant_2) + (0.01 *
      *u)); }
self->spec.plant._reg.__plant_3 = xp0;
self->spec.plant._reg.__plant_2 = xp1; */
//@assert dissip(state(self), \old(state(self)), in,
    e, P, gamma);
return;
}
```

Conclusion

Contributions:

- able to express/formulate system level properties (closed-loop stability, robustness through vector margins) as invariants over system states
- automatic computation and/or revalidation
- handle floating point arithmetics in the controller code
- validation a posteriori of computed invariants
- implemented over lustre models

Future work:

- integrate in the toolchain for smoother use
- address other properties, eg. performance (bounded overshoot)
- extend the range of considered systems:
  - linear MIMO
  - piecewise linear
  - LPV