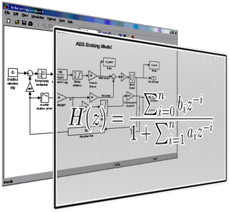
Interval-based Robustness of Linear Parametrized Filters

A. Chapoutot, T. Hilaire, P. Chevrel

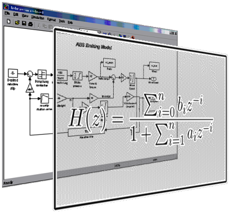
September 25th, 2012

Context – 1



Filters/Controllers
Algorithms

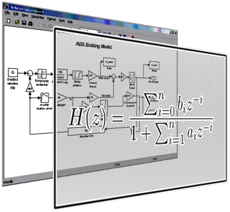
Context – 1



Filters/Controllers
Algorithms

Targets

Context – 1

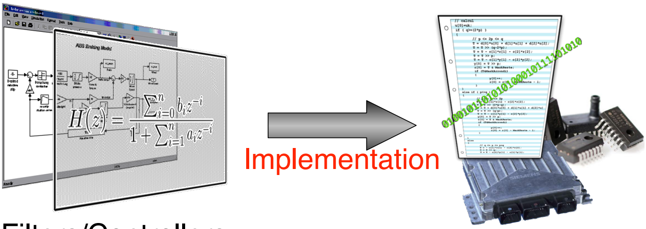


Filters/Controllers
Algorithms

Implementation

Targets

Context – 1

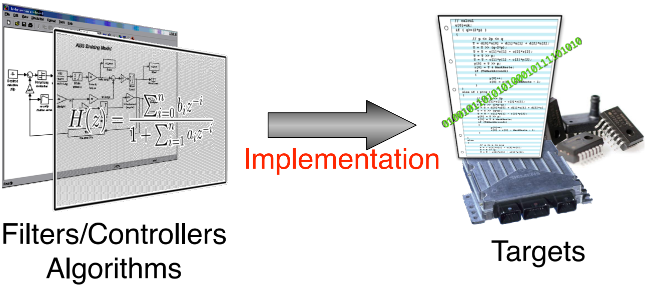


Filters/Controllers Algorithms

Targets

- Finite precision implementation (fixed-point arithmetic)
- Linear Time Invariant systems
- hardware (FPGA, ASIC) or software (DSP, μC)

Context – 1



- Finite precision implementation (fixed-point arithmetic)
- Linear Time Invariant systems
- hardware (FPGA, ASIC) or software (DSP, μC)

Evaluate the **robustness** of the implemented filter
☞Propose a methodology for the implementation of embedded controllers with finite precision considerations

Context – 2

We are considering **parametrized** linear filters/controllers, *i.e.* filters where the coefficients depend on extra parameters $\boldsymbol{\theta}$.

- These parameters are fixed, but unknown at implementation and compile-time;
- We only know intervals $[\boldsymbol{\theta}]$ they belong to;
- The coefficients are computed once *in-situ* at initialization-time;

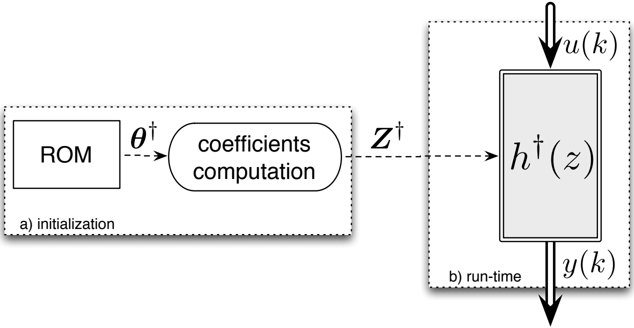
Context – 2

We are considering **parametrized** linear filters/controllers, *i.e.* filters where the coefficients depend on extra parameters $\boldsymbol{\theta}$.

- These parameters are fixed, but unknown at implementation and compile-time;
- We only know intervals $[\boldsymbol{\theta}]$ they belong to;
- The coefficients are computed once *in-situ* at initialization-time;

☞this is widely used by car manufacturers in order to calibrate controllers much more later in the development lifecycle.

Parametrized filters



Initialization and execution

Outline

Running example

Second order linear filter

We consider a continuous-time **second order Butterworth filter**.

Its transfer function is:

$$H(s) = \frac{g}{s^2 + 2\xi\omega_c s + \omega_c^2} \quad .$$

defined from 3 parameters:

- g the static gain;
- ξ the quality factor;
- ω_c the cutoff pulsation.

Object of the study: a discrete version of this filter.

Discrete-time algorithm – 1

The equivalent discrete-time filter is

$$H(z) = \frac{b_0 z^2 + b_1 z + b_2}{a_0 z^2 + a_1 z + a_2}$$

with

- $b_0 = gT^2, \quad b_1 = 2gT^2, \quad b_2 = gT^2$
- $a_0 = 4\xi\omega_c T + \omega_c^2 T^2 + 4,\ a_1 = 2\omega_c^2 T^2 - 8,$
  $a_2 = \omega_c^2 T^2 - 4\xi\omega_c T + 4$

Discrete-time algorithm – 1

The equivalent discrete-time filter is

$$H(z) = \frac{b_0 z^2 + b_1 z + b_2}{a_0 z^2 + a_1 z + a_2}$$

with

- $b_0 = gT^2, \quad b_1 = 2gT^2, \quad b_2 = gT^2$
- $a_0 = 4\xi\omega_c T + \omega_c^2 T^2 + 4,\ a_1 = 2\omega_c^2 T^2 - 8,$
  $a_2 = \omega_c^2 T^2 - 4\xi\omega_c T + 4$

To implement it, one can use Direct Form

$$\begin{cases} x(k+1) & = & \begin{pmatrix} -\frac{a_1}{a_0} & -\frac{a_2}{a_0} \\ 1 & 0 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u(k) \\ y(k) & = & \begin{pmatrix} \frac{b_1 a_0 - b_0 a_1}{a_0^2} & \frac{b_2 a_0 - b_0 a_2}{a_0^2} \end{pmatrix} x(k) + \frac{b_0}{a_0} u(k) \end{cases}$$

Discrete-time algorithm – 2

But it is also possible to use various other algorithms

- Or any other state-space form
- ρDirect Form II transposed
- cascade decomposition, parallel, lattice, LGC or LCW forms, etc.

Discrete-time algorithm – 2

But it is also possible to use various other algorithms

- Or any other state-space form
- ρDirect Form II transposed
- cascade decomposition, parallel, lattice, LGC or LCW forms, etc.

Remark

All these implementations are **only equivalent in infinite precision** arithmetic

Our goal

We seek the implementation which is the closest in finite precision than the infinite precision implementation.

Setting simulation parameters

List of parameter values and uncertainties:

- π
- f_c (cutoff frequency): $10.0 \pm 20\%$
- f_e (sampling frequency): $200.0 \pm 1\%$
- ξ (quality factor): $0.5 \pm 10\%$

Setting simulation parameters

List of parameter values and uncertainties:

- π
- f_c (cutoff frequency): $10.0 \pm 20\%$
- f_e (sampling frequency): $200.0 \pm 1\%$
- ξ (quality factor): $0.5 \pm 10\%$

Questions:

- What will be the impact of the quantization of these parameters ?
- What set of parameters will give use the worst degradation ?

Quantification Error Formalization

Formulation of the problem – 1

Notations:

- $\boldsymbol{Z}(\boldsymbol{\theta})$ the matrix containing all the coefficients used by the realization
- $h_{\boldsymbol{Z}(\boldsymbol{\theta})}$ the associated transfer function
- $\boldsymbol{\theta}^\dagger$ the quantized version of $\boldsymbol{\theta}$
- $\boldsymbol{Z}^\dagger(\boldsymbol{\theta}^\dagger)$ is then the set of the quantized coefficients, *i.e.* the quantization of coefficients $\boldsymbol{Z}(\boldsymbol{\theta}^\dagger)$ computed from the quantized parameters $\boldsymbol{\theta}^\dagger$
- The corresponding transfer function is denoted $h_{\boldsymbol{Z}^\dagger(\boldsymbol{\theta}^\dagger)}$.

Formulation of the problem – 2

For a given $\boldsymbol{\theta}$, the measure of the degradation of the finite precision implementation is given by:

$$\| \, h_{\boldsymbol{Z}(\boldsymbol{\theta})} - h_{\boldsymbol{Z}^{\dagger}(\boldsymbol{\theta}^{\dagger})} \, \|_{\diamond}, \quad \text{with} \quad \diamond \in \{2, \infty\}$$

such that, for $g : \mathbb{C} \to \mathbb{C}$, we have:

- 2-Norm: $\| \, g \, \|_2 \triangleq \sqrt{\frac{1}{2\pi} \int_0^{2\pi} | \, g(e^{j\omega}) \, |^2 \, d\omega}$
- Max Norm: $\| \, g \, \|_\infty \triangleq \max_{\omega \in [0, 2\pi]} | \, g(e^{j\omega}) \, |$

Problem

We **look for the worst-case parameters** $\boldsymbol{\theta}_0$ such that:

$$\arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \| \, h_{\boldsymbol{Z}(\boldsymbol{\theta})} - h_{\boldsymbol{Z}^{\dagger}(\boldsymbol{\theta}^{\dagger})} \, \|_{\diamond}$$

Finding Maximal Quantification Error

Interval global optimization approach

New formulation of the problem

$$\text{Maximize } \| \, [h]^{\dagger}_{\boldsymbol{z}^{\dagger}(\boldsymbol{\theta}^{\dagger})} - [h]_{\boldsymbol{z}(\boldsymbol{\theta})} \, \|_{\diamond} \quad \text{subject to} \quad \boldsymbol{\theta} \in [\boldsymbol{\theta}] \ .$$

such that $[h]$ is a transfer function with interval coefficients.

To apply for example Hansen's algorithm, we need:

- a sharp inclusion function for $\| \, [h]^{\dagger}_{\boldsymbol{z}^{\dagger}(\boldsymbol{\theta}^{\dagger})} - [h]_{\boldsymbol{z}(\boldsymbol{\theta})} \, \|_{\diamond}$

Inclusion functions – 1

Recall, for $g : \mathbb{C} \to \mathbb{C}$, we have:

- 2 Norm: $\| g \|_2 \triangleq \sqrt{\frac{1}{2\pi} \int_0^{2\pi} | g(e^{j\omega}) |^2 \, d\omega}$
- Max Norm: $\| g \|_\infty \triangleq \max_{\omega \in [0,2\pi]} | g(e^{j\omega}) |$

In both cases, the first step is to compute $| [g](e^{j\omega}) |$:

- either by using complex interval arithmetic;
- or real interval arithmetic after proper symbolic manipulations.

Inclusion functions – 2

We tried different approaches:

- Direct evaluation of $[g](e^{j\omega})$ with Cartesian complex interval form;

Inclusion functions – 2

We tried different approaches:

- Direct evaluation of $[g](e^{j\omega})$ with Cartesian complex interval form; ☹ (too loosy)

Inclusion functions – 2

We tried different approaches:

- Direct evaluation of $[g](e^{j\omega})$ with Cartesian complex interval form; ☹ (too loosy)
- Rewriting $|[g](e^{j\omega})|$ with symmetric and antisymmetric decomposition (cos and sin), and evaluation with real intervals;

Inclusion functions – 2

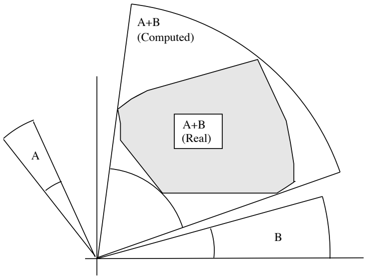
We tried different approaches:

- Direct evaluation of $[g](e^{j\omega})$ with Cartesian complex interval form; ☹ (too loosy)
- Rewriting $|[g](e^{j\omega})|$ with symmetric and antisymmetric decomposition (cos and sin), and evaluation with real intervals; ☹ (development with sin and cos doesn't help)
- Direct evaluation of $[g](e^{j\omega})$ with polar complex interval form;

Inclusion functions – 2

We tried different approaches:

- Direct evaluation of $[g](e^{j\omega})$ with Cartesian complex interval form; ☹ (too loosy)
- Rewriting $|[g](e^{j\omega})|$ with symmetric and antisymmetric decomposition (cos and sin), and evaluation with real intervals; ☹ (development with sin and cos doesn't help)
- Direct evaluation of $[g](e^{j\omega})$ with polar complex interval form; 😐 addition polar complex form very difficult – to be explored more deeply, see J. Flores *Complex Fans*



Inclusion functions – 3

But also:

- Symbolic computation of $[g](e^{j\omega}) \cdot [g](e^{j\omega})^*$ and evaluation with real interval arithmetic;

Inclusion functions – 3

But also:

- Symbolic computation of $[g](e^{j\omega}) \cdot [g](e^{j\omega})^*$ and evaluation with real interval arithmetic; ☹ Generation of very long formula which increases the problem of dependency (loosy results), more work could be done to simplify this expression (e.g. factorizing the common sub-expressions)

Inclusion functions – 3

But also:

- Symbolic computation of $[g](e^{j\omega}) \cdot [g](e^{j\omega})^*$ and evaluation with real interval arithmetic; ☹ Generation of very long formula which increases the problem of dependency (loosy results), more work could be done to simplify this expression (e.g. factorizing the common sub-expressions)
- For 2-norm, we can also use Lyapunov equation: $g(z)$ is put in form $g(z) = \boldsymbol{c}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b} + d$ and

$$\| g \|_2 = \sqrt{\operatorname{tr}(\boldsymbol{c}\boldsymbol{W}\boldsymbol{c}^\top + d^2)} \quad \text{with} \quad \boldsymbol{W} = \boldsymbol{A}\boldsymbol{W}\boldsymbol{A}^\top + \boldsymbol{b}\boldsymbol{b}^\top$$

Inclusion functions – 3

But also:

- Symbolic computation of $[g](e^{j\omega}) \cdot [g](e^{j\omega})^*$ and evaluation with real interval arithmetic; ☹ Generation of very long formula which increases the problem of dependency (loosy results), more work could be done to simplify this expression (e.g. factorizing the common sub-expressions)
- For 2-norm, we can also use Lyapunov equation: $g(z)$ is put in form $g(z) = \boldsymbol{c}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b} + d$ and

$$\| g \|_2 = \sqrt{\mathrm{tr}(\boldsymbol{c}\boldsymbol{W}\boldsymbol{c}^\top + d^2)} \quad \text{with} \quad \boldsymbol{W} = \boldsymbol{A}\boldsymbol{W}\boldsymbol{A}^\top + \boldsymbol{b}\boldsymbol{b}^\top$$

  Here $\boldsymbol{A}$ and $\boldsymbol{W}$ are interval matrices.
  Software *Versoft* (Rohn), based on *Intlab* (Rump) can deal with Lyapunov equation to solve. ☹ But results too loosy, due to dependency between coefs in $\boldsymbol{A}$ and $\boldsymbol{b}$

Global solver

Once the problem of the inclusion function solved, we will apply Hansen's algorithm[1] whose main steps are:

Input: $[f]$ inclusion function, X initial box, ϵ tolerance
Output: Y the sub-box associated to the minimal value of f.

Set $Y := X$
Compute $[f](Y)$, $\tilde{f} := \mathrm{ub}([f](\mathrm{mid}(Y)))$, $y := \mathrm{lb}([f](Y))$
Initialize list $L := \{(Y, y)\}$
$(*)$ Choose a coordinate direction k parallel to which Y has an edge of maxim length
Bisect Y following k to get V_1, V_2 such that $Y = V_1 \cup V_2$
Remove (Y, y) from L
Compute $[f](V_1)$, $[f](V_2)$ and $v_i = \mathrm{lb}\left([f](V_i)\right)$ for $i = 1, 2$
Enter pairs (V_1, v_1) and (V_2, v_2) at the end of L
Choose a pair $(\tilde{Y}, \tilde{y})$ in L such that $\tilde{y} \leq z$ for all (Z, z)
Remove all (Z, z) such that $\tilde{f} \leq z$
If $\mathrm{width}(Y) \leq \epsilon$ end algorithm
Denote the first pair of L as (Y, y), $\tilde{f} = \min(\tilde{f}, \mathrm{ub}\left([f]\left(\mathrm{mid}(Y)\right)\right))$ go to $(*)$.

[1] "New computer method for global optimization" H. Ratschek and J. Rokne

Current state of our work

We did not find a satisfactory solution for the inclusion function !
In all cases, evaluation usually produces large and useless intervals for $\| [h]^{\dagger}_{\boldsymbol{Z}^{\dagger}(\boldsymbol{\theta}^{\dagger})} - [h]_{\boldsymbol{Z}(\boldsymbol{\theta})} \|_{\diamond}$, event with small width interval parameter values.

Moreover, a Monte-Carlo-like evaluation of $\| [h]^{\dagger}_{\boldsymbol{Z}^{\dagger}(\boldsymbol{\theta}^{\dagger})} - [h]_{\boldsymbol{Z}(\boldsymbol{\theta})} \|_{\diamond}$ with desired interval parameters showed us the inclusion function should be more accurate.

Conclusion

Conclusion:

- the inclusion function is the most difficult problem to solve to apply interval global optimization method

Perspective:

- use of affine arithmetic or Taylor arithmetic to define inclusion function to avoid dependence problem.
- develop polar form interval arithmetic package
- or dedicated interval transfer-function evaluation techniques