Interval Analysis for Guaranteed Set Estimation

MaGiX@LiX – September 2011

Eric Walter
(joint work with L. Jaulin, M. Kieffer *et al.*)

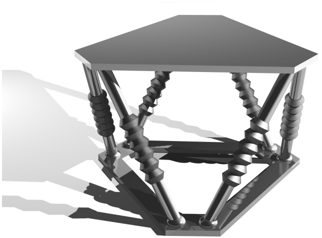
Laboratoire des Signaux et Systèmes
CNRS – Supélec – Univ Paris-Sud

September 16, 2011

Outline

Examples of Set-Estimation Problems

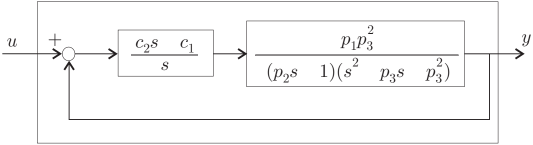
Robotics



- Stewart-Gough platform, archetypal parallel robot used, e.g., in flight simulators.
- Given the (fixed) lengths of the six limbs and geometry, find all possible configurations of mobile plate wrt base.
- Benchmark problem in computer algebra.

Examples of Set-Estimation Problems

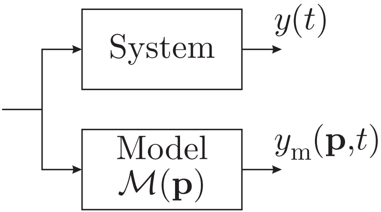
Robust control



- For given values of parameters c_1 and c_2 of PI controller, find all values of process parameter vector $\begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}$ such that behavior of controlled system is acceptable. Most basic requirement is stability.
- For given set of possible values for process parameter vector, find a tuning of controller parameters that guarantees acceptable behavior (or prove that none exists).

Examples of Set-Estimation Problems

Guaranteed parameter estimation



- Find all values of parameter vector **p** such that error between system output and model output belongs to some acceptable set.
- Find all values of **p** that are optimal in some statistical sense.

Origins of Interval Computations

Used by

- Archimedes, 3rd century BC to enclose π
- high-school physicists to assess errors...

Systematic use in numerical analysis and on computers often attributed to Ramon Moore (circa 1960), but many precursors, including M. Warmus (1956) and T. Sunaga (1958).

Limited impact outside inner circle until beginning of the 90s for various reasons, including implementation issues.

Things are improving as we shall see.

Interval Arithmetics

Basic arithmetic operations easily extend to intervals:

$$[x] \circ [y] = \{x \circ y \,|\, x \in [x] \text{ and } y \in [y]\}$$

For instance

$$\begin{aligned} [x]+[y] &= \left[\underline{x}+\underline{y}, \overline{x}+\overline{y}\right], \\ [x]-[y] &= \left[\underline{x}-\overline{y}, \overline{x}-\underline{y}\right]. \end{aligned}$$

Note that

- $[x]-[x]$ is not equal to zero! Try to avoid multi-occurences of variables...
- results computed using bounds of interval operands (intervals described by pairs of real numbers, just as complex numbers),
- division tricky when zero belongs to denominator interval.

Inclusion functions

Inclusion function $[f](\cdot)$ of $f(\cdot)$ satisfies

$$\forall [x] \subset \mathbb{R},\ f([x]) \subset [f]([x]).$$

It is

- minimal if $\subset$ can be replaced by $=$,
- convergent if $\mathrm{width}([x]) \to 0 \Rightarrow \mathrm{width}([f]([x])) \to 0$

Easy to build for monotone functions, e.g.,

$$\exp([\mathbf{x}]) = [\exp(\underline{x}), \exp(\overline{x})].$$

(Simple) algorithms available for $\sin(\cdot)$, $\cos(\cdot)$, etc.

Inclusion functions also available for solutions of nonlinear ODEs, see Berz – Makino talk.

Examples of Natural Inclusion Functions

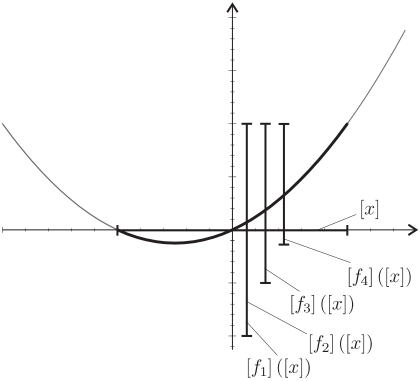
Natural inclusion functions: replace each variable and operator by its interval counterpart in formal expression.

Consider these four formal expressions of the same function

$$f_1(x) = x(x+1), \quad f_3(x) = x^2 + x,$$
$$f_2(x) = x \times x + x, \quad f_4(x) = (x + \tfrac{1}{2})^2 - \tfrac{1}{4}.$$

Evaluate their natural inclusion functions for $[x] = [-1, 1]$.

Examples of Natural Inclusion Functions



Only $[f_4]$ is minimal (x appears only once).

Another Useful Type of Inclusion Function

If f is differentiable over $[x]$, mean-value theorem states that $\forall x \in [x]$, $\exists \xi \in [x]$ such that

$$f(x) = f(m) + f'(\xi) \cdot (x - m),$$

with m the center of $[x]$. Then

$$f(x) \in f(m) + f'([x]) \cdot (x - m)$$

and

$$f([x]) \subseteq f(m) + [f']([x]) \cdot ([x] - m).$$

Hence the centred form

$$[f]_c([x]) = f(m) + [f']([x]) \cdot ([x] - m).$$

Which Inclusion Function To Use?

Compare the natural and centred inclusion functions for

$$f(x) = x^2 \exp(x) - x \exp\left(x^2\right).$$

Best inclusion function depends on width of interval argument:

$[x]$	$f([x])$	$[f]([x])$	$[f]_c([x])$
$[0.5, 1.5]$	$[-4.148, 0]$	$[-13.82, 9.44]$	$[-25.07, 25.07]$
$[0.9, 1.1]$	$[-0.05380, 0]$	$[-1.697, 1.612]$	$[-0.5050, 0.5050]$
$[0.99, 1.01]$	$[-0.0004192, 0]$	$[-0.1636, 0.1628]$	$[-0.0047, 0.0047]$

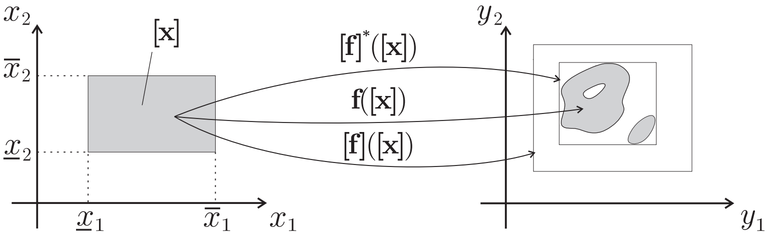
Intersecting results provided by several inclusion functions may provide a more accurate result than any of them separately.

Computing with Interval Vectors and Functions

An interval vector (or box) is a Cartesian product of scalar intervals

$$[\mathbf{x}] = [x_1] \times \cdots [x_n].$$

Interval computation extends easily to boxes, as well as notion of inclusion function.



Solving Systems of Nonlinear Equations

System writen as

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}.$$

We assume that

- there are as many equations are there are unknowns ($\dim \mathbf{f}(\mathbf{x}) = \dim \mathbf{x} = n$),
- $\mathbf{f}$ is continuously differentiable.

We want all solutions in a given box $[\mathbf{x}]_0$.

The approach is an interval variant of the Newton method.

Interval Newton Method

Mean-value theorem implies that $\forall \mathbf{x} \in [\mathbf{x}]$, $\exists \xi \in [\mathbf{x}]$ such that

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{m}) + \mathbf{J_f}(\xi)(\mathbf{x} - \mathbf{m}).$$

Now we want $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, so

$$\mathbf{f}(\mathbf{m}) + \mathbf{J_f}(\xi)(\mathbf{x} - \mathbf{m}) = \mathbf{0}.$$

with $\mathbf{J_f}$ the Jacobian matrix of $\mathbf{f}$, assumed invertible for the sake of simplicity. Thus

$$\mathbf{x} = \mathbf{m} - \mathbf{J_f^{-1}}(\xi)\mathbf{f}(\mathbf{m}).$$

Now, since the value of ξ is not known, we can only write

$$\mathbf{x} \in \mathbf{m} - \mathbf{J_f^{-1}}([\mathbf{x}])\,\mathbf{f}(\mathbf{m}),$$

or rather

$$\mathbf{x} \in \left[\mathbf{m} - \mathbf{J_f^{-1}}([\mathbf{x}])\,\mathbf{f}(\mathbf{m})\right] \cap [\mathbf{x}].$$

Interval Newton Method

This suggest an iterative method

$$[\mathbf{x}]^{k+1} = \left[\mathbf{m} - \left[\mathbf{J}_{\mathbf{f}}^{-1}\right]\left([\mathbf{x}]^{k}\right)\mathbf{f}(\mathbf{m})\right] \cap [\mathbf{x}]^{k},$$

which is an example of contractor. Actually a lot of details skipped...

- not necessary to assume that the Jacobian matrix is invertible,
- if $[\mathbf{x}]_0$ too large, it is split into subboxes $[\mathbf{x}]$ to be explored.

Note that

- no solution can be lost
- can prove that no solution exists
- solution approximate but guaranteed.

Set Inversion

Let $\mathbf{y}_{\mathrm{m}}(\cdot)$ be a vector function from parameter space to data space.

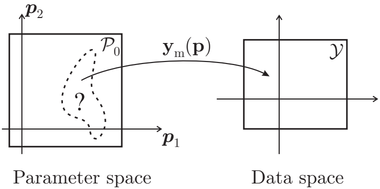
Assume $\mathbf{y}_{\mathrm{m}}(\mathbf{p})$ should belong to $\mathscr{Y}$.

We want to characterize the set of all feasible values of $\mathbf{p}$ that make the model OK:

$$\mathbb{S} = \{\mathbf{p} \in \mathscr{P}_0 \,|\, \mathbf{y}_{\mathrm{m}}(\mathbf{p}) \in \mathscr{Y}\} = \mathbf{y}_{\mathrm{m}}^{-1}(\mathscr{Y})$$

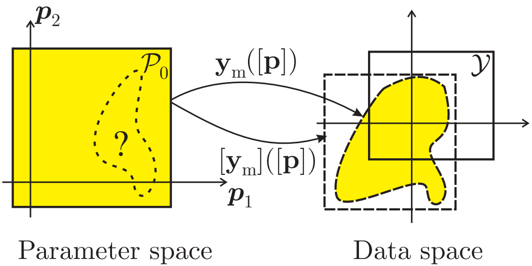
This is set inversion.

Set Inversion



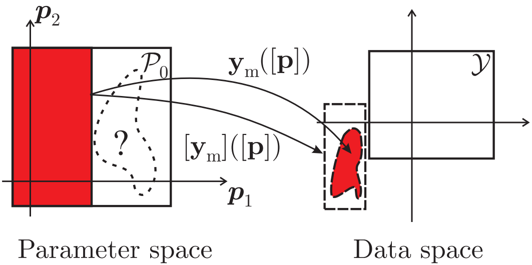
- Characterization of $\mathbb{S}$ performed with SIVIA (for set inversion via interval analysis).
- Assumes an inclusion function $[\mathbf{y}_{\mathrm{m}}](\cdot)$ is available for $\mathbf{y}_{\mathrm{m}}(\cdot)$.
- $\mathbb{S}$ bracketted between inner and outer approximations.

SIVIA



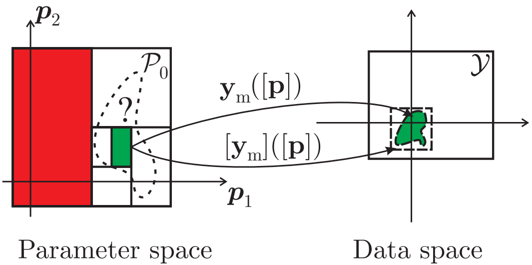
Yellow box is indeterminate.

SIVIA



Red box proven to be outside $\mathbb{S}$.

SIVIA



Green box proven to be inside $\mathbb{S}$.

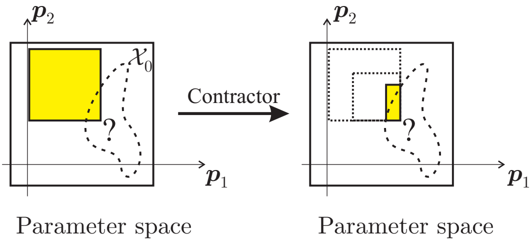
SIVIA

By bisecting indeterminate boxes that are large enough and testing the resulting subboxes, SIVIA partitions prior feasible space $\mathscr{P}_0$ into

- boxes inside $\mathbb{S}$
- boxes outside $\mathbb{S}$
- indeterminate boxes (which could be split to get a more accurate characterization of $\mathbb{S}$).

Splitting in all directions induces exponential complexity. So splitting should be avoided whenever possible. Hence the importance of contractors.

SIVIA with Contractors



Contractors make it possible to reduce the size of indeterminate boxes without bisection.

Guaranteed Optimization: Hansen's Algorithm

Three ideas, already behind the scene when solving sets of equations or performing set inversion, are implemented in Hansen's algorithm. They are

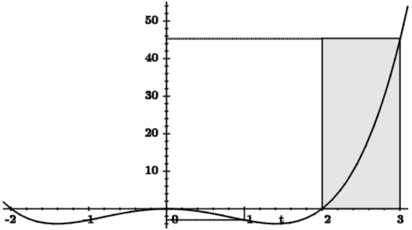
1. Eliminate
2. Contract
3. Divide & conquer

We assume $f(\cdot)$ is to be minimized, with no active constraint.

Eliminate

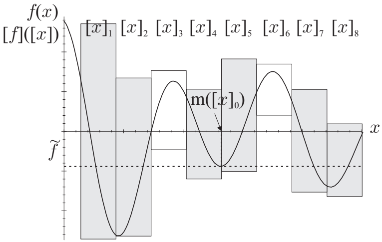
Consider $f(x) = x^4 - 4x^2$, to be minimized over $[-10, 10]$.

$f(1) = -3$ and $f([2,3]) = [0, 45]$ prove that $[2, 3]$ contains no global minimizer and $[2, 3]$ can thus be eliminated.



Eliminate Via Midpoint Test

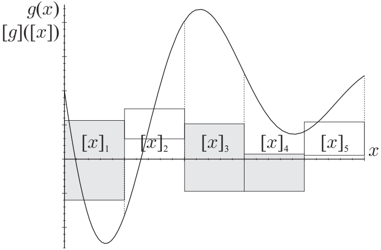
If $f(\cdot)$ is to be minimized and $\tilde{f}$ is an upper bound of f^*, then any $[\mathbf{x}]$ such that $\inf([f]([\mathbf{x}])) > \tilde{f}$ can be eliminated.



Eliminate Via Monotonicity Test

Any unconstrained optimizer $\mathbf{x}^*$ of a differentiable $f(\mathbf{x})$ should satisfy $\mathbf{g}(\mathbf{x}^*) = \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^*) = \mathbf{0}$,

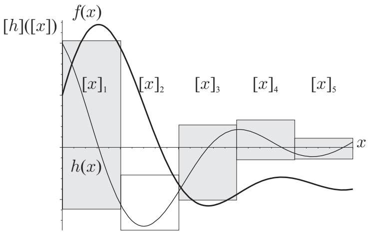
so any $[\mathbf{x}]$ such that $\mathbf{0} \notin [\mathbf{g}]([\mathbf{x}])$ can be eliminated.



Eliminate Via Convexity Test

$f(\cdot)$ to be minimized $\Longrightarrow$ should be locally convex at $\mathbf{x}^*$ $\Longrightarrow$ its Hessian $\mathbf{H}(\mathbf{x}^*)$ should have no negative eigenvalue

$\Longrightarrow$ any $[\mathbf{x}]$ such that $\exists i \,|\, [h_{ii}]([\mathbf{x}]) < 0$ can be eliminated.



Contract

Any unconstrained (local or global) optimizer $\mathbf{x}^*$ should satisfy

$$\mathbf{g}(\mathbf{x}^*) = \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^*) = \mathbf{0}.$$

So a Newton contractor can be used to solve

$$\mathbf{g}(\mathbf{x}) = \mathbf{0}$$

over $[\mathbf{x}]$ and thus reduce/eliminate/split any box of interest $[\mathbf{x}]$.

Divide and Conquer

1. Try to eliminate the (possibly very large) search box of interest $[\mathbf{x}]_0$ (you may succeed at doing so, for instance if there is no unconstrained minimizer of the cost function over $[\mathbf{x}]_0$...).
2. If box resists elimination, try to contract it.
3. Split box in two subboxes, each of which will become a search box of interest, unless too small.

Guaranteed Robust Tuning

Specifications often boil down to inequalities to be satisfied

$$\mathbf{f}(\mathbf{c},\mathbf{p}) > \mathbf{0},$$

with $\mathbf{c} \in \mathbb{S}_{\mathrm{c}}^{\mathrm{prior}}$ a vector of tuning parameters and $\mathbf{p} \in \mathbb{S}_{\mathrm{p}}$ a vector of uncertain process parameters.
We look for one $\mathbf{c} \in \mathbb{S}_{\mathrm{c}}^{\mathrm{prior}}$ such that

$$\forall \mathbf{p} \in \mathbb{S}_{\mathrm{p}},\ \mathbf{f}(\mathbf{c},\mathbf{p}) > \mathbf{0},$$

and assume for simplicity that $\mathbb{S}_{\mathrm{c}}^{\mathrm{prior}}$ and $\mathbb{S}_{\mathrm{p}}$ are boxes. The algorithm consists of two procedures: FPS (for Feasible Point Searcher) and CSC (for Computable Sufficient Conditions).

CSC Procedure

Inputs: uncertainty box $\mathbb{S}_\mathrm{p}$ for process parameters, box of interest $[\mathbf{c}]$ for tunable parameters. Output: suitability of $[\mathbf{c}]$.

1. Stack $= \mathbb{S}_\mathrm{p}$.
2. Unstack into $[\mathbf{p}]$.
3. If $\exists i \,|\, [f]_i \left([\mathbf{c}], \mathrm{center}\,(\mathbf{p})\right) \leqslant 0$, return "no robust tuning in $[\mathbf{c}]$".
4. If $[\mathbf{f}]\,(\mathrm{center}\,(\mathbf{c}), [\mathbf{p}]) > \mathbf{0}$, go to Step 7.
5. If $\mathrm{width}([\mathbf{p}])$ too small to iterate, return "$[\mathbf{c}]$ indeterminate".
6. Bisect $[\mathbf{p}]$ and stack the two resulting boxes.
7. If stack nonempty, go to Step 2.
8. Return "$\mathrm{center}\,([\mathbf{c}])$ is a robust tuning".

FPS Procedure

Organizes a systematic examination of $\mathbb{S}_c^{\text{prior}}$ by CSC.

1. Call CSC. If it returns "center ([c]) is a robust tuning", do likewise.
2. If CSC returns "no robust tuning in [c]", go to Step 4.
3. Bisect [c] and push the two resulting boxes into the queue.
4. If queue nonempty, pull its first box into [c] and go to Step 1.
5. Return "no feasible robust tuning in $\mathbb{S}_c^{\text{prior}}$".

As there might be a large set of robust tunings, one might look for the optimal one in some sense (minimax procedure, not considered here).

Returning to Examples

Let us go back to introductory examples to

- address them with IA tools,
- find more about the advantages, disadvantages and limitations of IA tools.

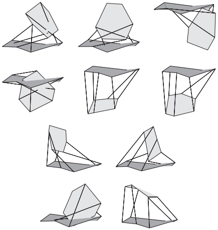
Steward-Gough Plateform

- Configuration of mobile plate wrt base specified by $\mathbf{x}$ ($\dim \mathbf{x} = 6$).
- Components of $\mathbf{x}$ include three Euler angles.
- Easy to compute the lengths of the six limbs as $\mathbf{g}(\mathbf{x})$, using trigonometric functions.
- Finding all possible configurations amounts to solving

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \text{ where } \mathbf{f}(\mathbf{x}) = \mathbf{g}(\mathbf{x}) - \mathbf{g}_{\mathrm{m}},$$

  with $\mathbf{g}_{\mathrm{m}}$ the vector of the measured lengths of the six limbs.
- Interval Newton applicable, even in the most complex case of non-planar base and mobile plate.

Interval Newton Solutions for the SG plateform
(Non-planar base and mobile plate)



- Algorithm finds ten boxes, associated with as many solutions.
- Each box proved to contain one solution only (by fixed-point theorem).
- All real solutions found, some of them irrealistic as mobile plate crashes into base (not explicitely forbidden...).

Interval Newton Versus Computer Algebra

Interval Newton has several advantages over the use of elimination theory:

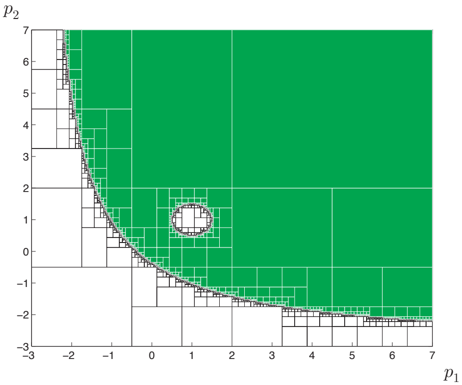
- No need to make the equations polynomial (at the cost of increasing the number of unknown and equations from 6 to 9).
- No need to solve a high-degree polynomial equation numerically.
- Only real solutions are obtained.
- Uncertainty on the measured lengths of the limbs can be taken into account.
- Solutions approximate but guaranteed, and given with their precision.

Robust Control

Characterizing stability domains

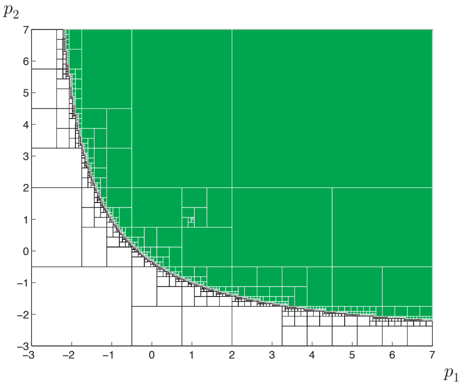
- We want to make sure stability is guaranteed for any value of the uncertain $\mathbf{p}$.
- A linear time-invariant continuous-time model is stable iff all roots of its characteristic equation have strictly negative real parts.
- Routh-Hurwitz criterion translates this into a finite number of inequalities that $\mathbf{p}$ must satisfy.
- SIVIA applicable to characterize the set of all values of $\mathbf{p}$ that correspond to a stable model.
- Ackermann benchmark: unstable disk inside a stable region. The smaller the disk, the more difficult the test becomes.

Easy Ackermann Test



Unstable disk clearly visible.

Difficult Ackermann Test



Unstable disk reduced to point (1, 1). SIVIA was unable to prove that this point was unstable, but generated an indeterminate box around this point (too small to be visible).

IA Versus Kharitonov

If the coefficients of the characteristic polynomial belong to independent intervals, then celebrated Kharitonov theorem provides necessary and sufficient conditions for stability, which involve the computation of the roots of only four deterministic polynomials.

SIVIA can deal with much more complex dependencies of the coefficients of the characteristic polynomial in $\mathbf{p}$, as examplified by Ackermann's test.

IA Versus Monte-Carlo

A commonly used approach to study uncertain systems is by randomly picking values of $\mathbf{p}$. There is, however, no guarantee that the problematic cases will be detected (and in the case of the difficult Ackermann test this is out of the question).

SIVIA gives guaranteed results for general dependencies of the coefficients of the characteristic polynomial in $\mathbf{p}$.

Robust Tuning of a PID Controller 1/3

Process described by transfer function

$$H(s,\mathbf{p}) = \frac{K\omega_0^2}{(1+Ts)(s^2+2\zeta\omega_0 s+\omega_0^2)}.$$

Uncertain process parameter vector **p** consists of:

- Static gain $K \in [0.95, 1.05]$
- Damping factor $\zeta \in [0.95, 1.05]$
- (Undamped) natural pulsation $\omega_0 \in [0.95, 1.05]$
- Negative time constant $T \in [-1.05, -0.95]$

Because of negative time constant, process is open-loop unstable. PID controller inserted in the forward path of a control loop with negative unity feedback.

Robust Tuning of a PID Controller 2/3

PID transfer function taken as

$$C(s,\mathbf{c}) = \frac{c_1 + c_2 s + c_3 s^2}{s}.$$

Characteristic equation of resulting closed-loop is

$$s^4 + \left(2\zeta\omega_0 + T^{-1}\right) s^3 + \left(2\zeta\omega_0 T^{-1} + \omega_0^2\left(1 + c_3 K T^{-1}\right)\right) s^2 \\ + \omega_0^2 \left(1 + c_2 K\right) T^{-1} s + \omega_0^2 K c_1 T^{-1} = 0.$$

Kharitonov's theorem useless here, but FPS + CSC algorithm applicable.

Robust Tuning of a PID Controller 3/3

Routh criterion provides necessary and sufficient conditions for stability, which can be written as

$$\mathbf{f}(\mathbf{c},\mathbf{p}) > \mathbf{0},$$

with $\dim \mathbf{c} = 3$, $\dim \mathbf{p} = 4$ and $\dim \mathbf{f}(\mathbf{c},\mathbf{p}) = 5$. In $\mathbb{S}_{\mathrm{c}}^{\mathrm{prior}} = [-10,10] \times [-10,10] \times [-10,10]$, FPS + CSC finds, in the blink of an eye, the robust controller

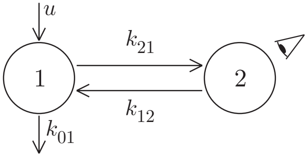
$$C(s) = -\frac{0.625 + 3.75s + 8.75s^2}{s},$$

guaranteed to stabilize all uncertain processes considered.

Guaranteed Parameter Estimation

Estimating the parameters of a compartmental model

Compartmental models widely used in biology, e.g., to describe the fate of drugs in living entities.



Model Equations

State equation

$$\begin{cases} \dot{x}_1 = -(p_1 + p_3)\, x_1 + p_2 x_2, \\ \dot{x}_2 = p_1 x_1 - p_2 x_2, \end{cases}$$

with

$$\mathbf{x}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Observation equation

$$y_{\mathrm{m}}(t_i, \mathbf{p}) = x_2(t_i, \mathbf{p}) \quad i = 1 \cdots N.$$

Problem To Be Solved

Find

$$\mathbb{S} = \{\mathbf{p} \,|\, J(\mathbf{p}) \text{ is minimal}\},$$

where cost function J is

$$J(\mathbf{p}) = \sum_{i=1}^{N} \left[y(t_i) - y_{\mathrm{m}}(t_i, \mathbf{p})\right]^2.$$

Hansen's algorithm applicable.

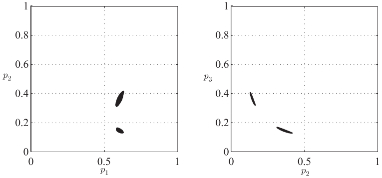
Ingredients For Global Optimization

Inclusion functions for cost and its gradient computed via

- formal derivation of gradient in terms of sentitivity of model output to $\mathbf{p}$,
- formal derivation of ODEs satisfied by this sensitivity,
- use of Müller's theorem to derive deterministic ODEs that bracket the behavior of uncertain ODEs,
- use of guaranteed ODE solver (here VNODE) for these deterministic ODEs,
- use of contractors to reduce size of boxes of interest.

Guaranteed Optimal Parameter Estimation

IA algorithm proves that all global optimizers of cost are in the sets whose projections are presented below.



Symmetry suggests an identifiability problem that can be confirmed by a theoretical study.

Conclusions

- IA provides guaranteed numerical characterization of sets of interest for nontrivial problems.
- When applicable IA has definite advantages over conventional floatting-point numerical computations (and computer algebra for that matter).
- But it is not so easy to apply, and not always applicable (curse of dimensionality, lack of efficient inclusion functions...).
- Much remains to be done to make it part of the standard numerical engineering toolbox.
- Much has already been done though, as it is...

Easier Than Ever To Compute With Intervals

Many books, code libraries, lists, including
http://www.cs.utep.edu/interval-comp/main.html

Ongoing standardization process IEEE P1788.

INTLAB library for computing with intervals in MATLAB.

For Further Reading I

E. Hansen.
Global Optimization Using Interval Analysis.
Dekker, 1992.

L. Jaulin, M. Kieffer, O. Didrit, E. Walter.
Applied Interval Analysis.
Springer, 2001.

L. Jaulin, E. Walter.
Guaranteed tuning, with application to robust control and motion planning.
Automatica. 32(8):1217–1221, 1996.

For Further Reading II

- O. Didrit, M. Petitot, E. Walter.
  Guaranteed solution of direct kinematic problems for general configurations of parallel manipulators.
  IEEE Trans. on Robotics and Automation. 14(2):259–266, 1998.
- M. Kieffer, E. Walter.
  Guaranteed estimation of the parameters of nonlinear continuous-time models: Contributions of interval analysis.
  Int. J. of Adaptive Control and Signal Processing. 25:191–207, 2011.