Universität Rostock Traditio et Innovatio

Hochschule Wismar University of Applied Sciences Technology, Business and Design

Interval Methods for Reliable Modeling, Identification and Control of Dynamic Systems

Pre-Conference Workshop at ECC2015
14th European Control Conference

Linz, Austria, July 14th, 2015

Andreas Rauh*, Luise Senkel*, Ekaterina Auer**

*Chair of Mechatronics, University of Rostock, Germany

**Faculty of Engineering, University of Applied Sciences Wismar, Germany

Contents

- Part 1
  - Fundamentals of Interval Arithmetics: Concept and Software Demonstration
  - Verified Simulation of Dynamic Systems
  - Kinds of Uncertainty and their Treatment during Modeling and Simulation
  - Possibilities for Simulating Uncertain Non-Smooth Dynamic Systems
- Part 2
  - Control-Oriented Applications: Identification and Optimization
  - Interval-Based Sliding Mode Approaches: Control and Estimation
  - Application to Fuel Cell Systems for Identification and Control

Part 1

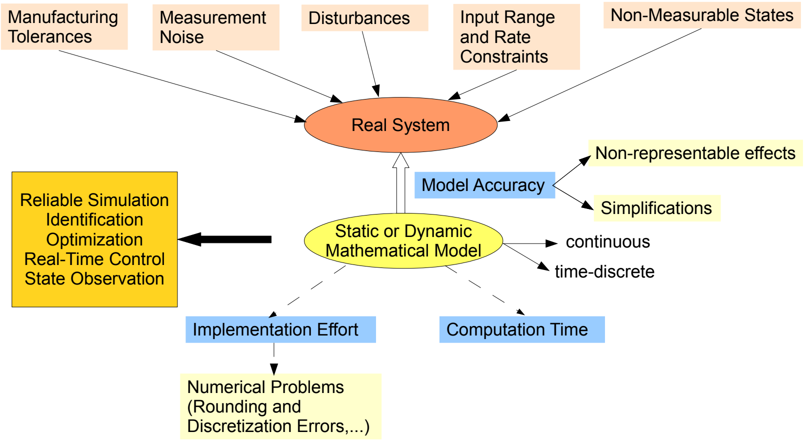
1. Fundamentals of Interval Arithmetic

Presentation of the Fundamental Mathematical Concept of Interval Arithmetic for Set-Valued Computations

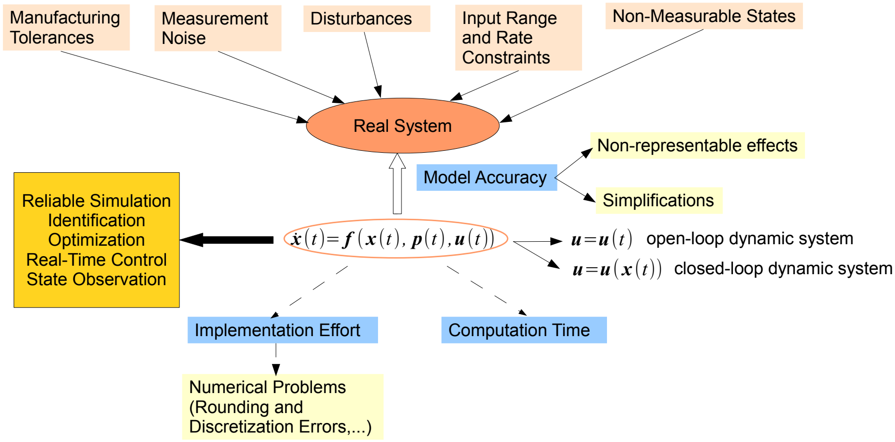
Presentation of the Fundamental Mathematical Concept of Interval Arithmetic for Set-Valued Computations

Contents

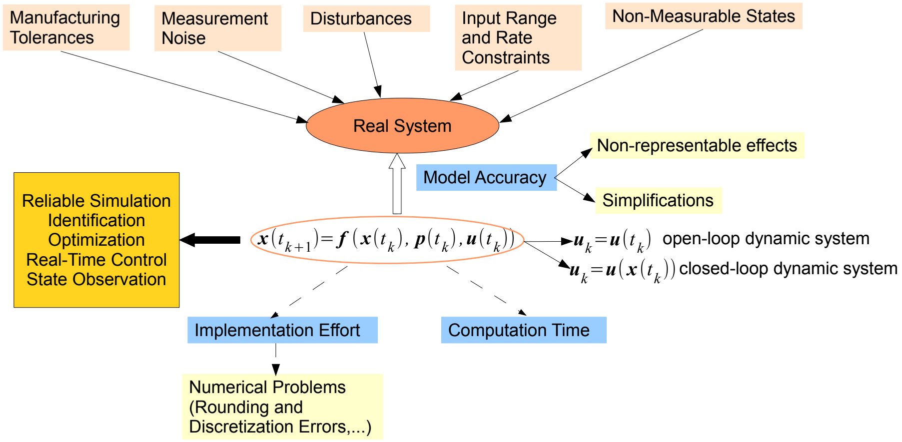
Motivation: Uncertainty



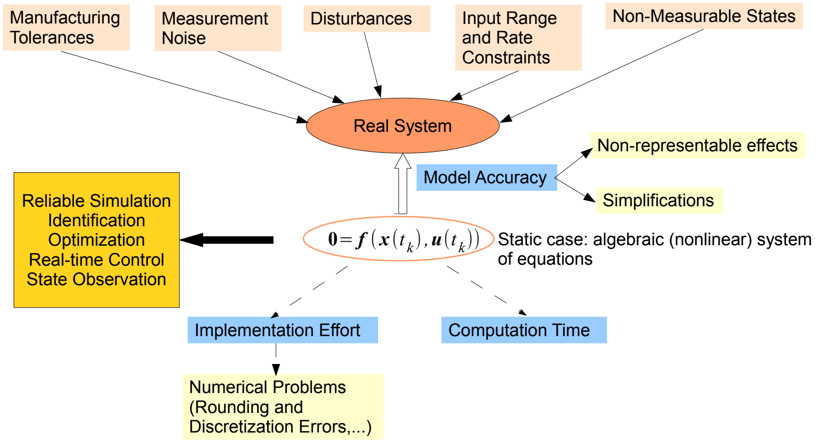
Motivation: Uncertainty



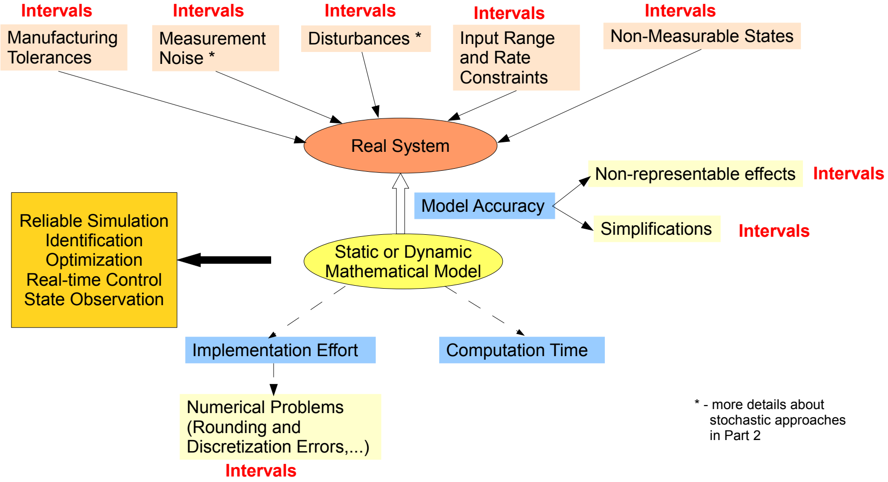
Motivation: Uncertainty



Motivation: Uncertainty



Motivation: Uncertainty



* - more details about stochastic approaches in Part 2

Definition of Real Intervals, Interval Vectors, Interval Matrices

Scalar Real Interval

$$[a] = [\underline{a}; \overline{a}] = [\mathsf{inf}([a]); \mathsf{sup}([a]))] \ , \ \underline{a} \leq \overline{a} \ , \ \{x \in \mathbb{R} | \underline{a} \leq x \leq \overline{a}\}$$

Interval Vector

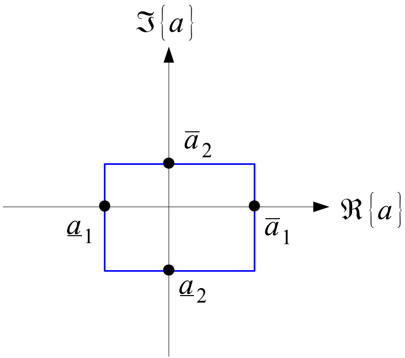
$$[\mathbf{a}] = \begin{bmatrix} [\underline{a}_1; \overline{a}_1] \\ [\underline{a}_2; \overline{a}_2] \\ \vdots \\ [\underline{a}_n; \overline{a}_n] \end{bmatrix}$$

Interval Matrix

$$[\mathbf{A}] = \begin{bmatrix} [\underline{a}_{11}; \overline{a}_{11}] & [\underline{a}_{12}; \overline{a}_{12}] & \dots & [\underline{a}_{1n}; \overline{a}_{1n}] \\ [\underline{a}_{21}; \overline{a}_{21}] & [\underline{a}_{22}; \overline{a}_{22}] & \dots & [\underline{a}_{2n}; \overline{a}_{2n}] \\ \vdots & \vdots & \ddots & \vdots \\ [\underline{a}_{n1}; \overline{a}_{n1}] & [\underline{a}_{n2}; \overline{a}_{n2}] & \dots & [\underline{a}_{nn}; \overline{a}_{nn}] \end{bmatrix}$$

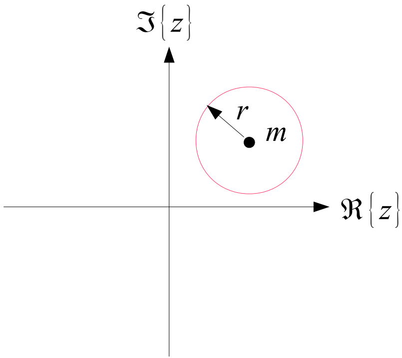
Definition of Complex Intervals

Rectangular



$$\boxed{[a]=[\underline{a}_1;\overline{a}_1]+j[\underline{a}_2;\overline{a}_2]}$$

Circular



$$\boxed{[z]=<m,r>}$$

$\Rightarrow$ Useful for dynamic systems with oscillatory behavior

Calculating with Real Intervals - Natural Interval Evaluation

Addition

$$[\underline{p};\overline{p}] + [\underline{q};\overline{q}] = [\underline{p}+\underline{q};\overline{p}+\overline{q}]$$

$$[1;2] + [-2;2] = [1+(-2);2+2] = [-1;4]$$

$$\begin{bmatrix}[-2;-1]\\ [0;4]\end{bmatrix} + \begin{bmatrix}[-10;-3]\\ [5;8]\end{bmatrix} = \begin{bmatrix}[-12;-4]\\ [5;12]\end{bmatrix}$$

$$\begin{bmatrix}[2;3] & [-4;-3]\\ [7;9] & [10;15]\end{bmatrix} + \begin{bmatrix}[12;13] & [-14;-13]\\ [17;19] & [20;25]\end{bmatrix} = \begin{bmatrix}[14;16] & [-18;-16]\\ [24;28] & [30;40]\end{bmatrix}$$

Calculating with Real Intervals - Natural Interval Evaluation

Subtraction

$$[\underline{p};\overline{p}] - [\underline{q};\overline{q}] = [\underline{p} - \overline{q};\overline{p} - \underline{q}]$$

$$[1;2] - [2;3] = [1-3;2-2] = [-2;0]$$

$$\begin{bmatrix} [2;3] & [-4;-3] \\ [7;9] & [10;15] \end{bmatrix} - \begin{bmatrix} [12;13] & [-14;-13] \\ [17;19] & [20;25] \end{bmatrix} = \begin{bmatrix} [-11;-9] & [9;11] \\ [-12;-8] & [-15;-5] \end{bmatrix}$$

Calculating with Real Intervals - Natural Interval Evaluation

Multiplication

$$[\underline{p};\overline{p}]\cdot[\underline{q};\overline{q}]=\left[\min\left\{\underline{p}\,\underline{q},\underline{p}\,\overline{q},\overline{p}\,\underline{q},\overline{p}\,\overline{q}\right\}\;;\max\left\{\underline{p}\,\underline{q},\underline{p}\,\overline{q},\overline{p}\,\underline{q},\overline{p}\,\overline{q}\right\}\right]$$

$$[1;2]\cdot[2;3]=[\min\{1\cdot 2,1\cdot 3,2\cdot 2,2\cdot 3\};\max\{1\cdot 2,1\cdot 3,2\cdot 2,2\cdot 3\}]=[2;6]$$

Calculating with Real Intervals - Natural Interval Evaluation

Division

$$\frac{[p]}{[q]} = [p] \cdot \left[\frac{1}{\overline{q}} \; ; \; \frac{1}{\underline{q}} \right] \quad \text{if} \quad 0 \notin [q]$$

$$\frac{[1;2]}{[2;3]} = [1;2] \cdot \left[\frac{1}{3} \; ; \; \frac{1}{2} \right] = \left[\frac{1}{3} ; 1 \right]$$

Calculating with Real Intervals - Natural Interval Evaluation

Radius of a Real Interval

$$r([a]) = \frac{1}{2}(\overline{a} - \underline{a})$$

Width of an Interval

$$w([a]) = \overline{a} - \underline{a} = 2 \cdot r([a])$$

Mid-point of an Interval

$$m([a]) = \frac{1}{2}(\underline{a} + \overline{a})$$

$\Rightarrow$ For real interval vectors and matrices, these characteristics hold component-wise

Continuous- and Discrete-Time Systems — Dynamic Case

Continuous-Time System $\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{p}, \mathbf{u}(t)\right)$

$\mathbf{x}(t)$	State Vector
$\mathbf{p}$	Vector of Uncertain Parameters: $p_i \in \left[\underline{p}_i \ ; \ \overline{p}_i\right], i = 1, ..., n_p$
$\mathbf{u}(t)$	Input Vector: $u_j \in \left[\underline{u}_j \ ; \ \overline{u}_j\right], j = 1, ..., n_u$

Continuous- and Discrete-Time Systems — Dynamic Case

Continuous-Time System $\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{p}, \mathbf{u}(t)\right)$

$\mathbf{x}(t)$	State Vector
$\mathbf{p}$	Vector of Uncertain Parameters: $p_i \in \left[\underline{p}_i \ ; \ \overline{p}_i\right], i = 1, ..., n_p$
$\mathbf{u}(t)$	Input Vector: $u_j \in \left[\underline{u}_j \ ; \ \overline{u}_j\right], j = 1, ..., n_u$

Discrete-Time System $\mathbf{x}(t_{k+1}) = \mathbf{f}\left(\mathbf{x}(t_k), \mathbf{p}(t_k), \mathbf{u}(t_k)\right)$

$\mathbf{x}(t_k)$	State Vector
$\mathbf{p}$	Vector of Uncertain Parameters: $p_i \in \left[\underline{p}_i \ ; \ \overline{p}_i\right]$
	$\Longrightarrow$ **Range Bounds / Tolerances**
$\mathbf{u}(t_k)$	Input Vector: $u_j(t_k) \in \left[\underline{u}_j(t_k) \ ; \ \overline{u}_j(t_k)\right]$
	$\Longrightarrow$ **Input Range Constraints**

$\Rightarrow$ Calculate all reachable states

Continuous- and Discrete-Time Systems — Static Case

Continuous-Time System $\dot{\mathbf{x}}(t) = \mathbf{0} = \mathbf{f}(\mathbf{x}(t), \mathbf{p}, \mathbf{u}(t))$

- $\mathbf{x}(t)$ State Vector
- $\mathbf{p}$ Vector of Uncertain Parameters: $p_i \in \left[\underline{p}_i \; ; \; \overline{p}_i\right], i = 1, ..., n_p$
- $\mathbf{u}(t)$ Input Vector: $u_j \in \left[\underline{u}_j \; ; \; \overline{u}_j\right], j = 1, ..., n_u$

Continuous- and Discrete-Time Systems — Static Case

Continuous-Time System $\dot{\mathbf{x}}(t) = \mathbf{0} = \mathbf{f}\left(\mathbf{x}(t), \mathbf{p}, \mathbf{u}(t)\right)$

$\mathbf{x}(t)$ State Vector

$\mathbf{p}$ Vector of Uncertain Parameters: $p_i \in \left[\underline{p}_i \; ; \; \overline{p}_i\right], i = 1, ..., n_p$

$\mathbf{u}(t)$ Input Vector: $u_j \in \left[\underline{u}_j \; ; \; \overline{u}_j\right], j = 1, ..., n_u$

Discrete-Time System $\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) = \mathbf{f}\left(\mathbf{x}(t_k), \mathbf{p}(t_k), \mathbf{u}(t_k)\right)$

$\mathbf{x}(t_k)$ State Vector

$\mathbf{p}$ Vector of Uncertain Parameters: $p_i \in \left[\underline{p}_i \; ; \; \overline{p}_i\right]$

$\Longrightarrow$ **Range Bounds / Tolerances**

$\mathbf{u}(t_k)$ Input Vector: $u_j(t_k) \in \left[\underline{u}_j(t_k) \; ; \; \overline{u}_j(t_k)\right]$

$\Longrightarrow$ **Input Range Constraints**

$\Rightarrow$ Solve for state vector $\mathbf{x}(t)$ or $\mathbf{x}_k$ resp. for 1 time step

Overestimation: Dependency Problem

Problem: Multiple occurence of an interval in one equation

Necessary: Factorizations, simplifications, reformulations as far as possible
$\Rightarrow$ Reduction of overestimation and computation time

Example

$[f]([x]) = 2 \cdot [x] - [x] \cdot [x]$ and $[x] = [-1\ ;\ 2]$, Results provided by Intlab

1. $[f]([x]) = 2 \cdot [x] - [x] \cdot [x] = [-6\ ;\ 6]$
2. $[f]([x]) = 2 \cdot [x] - [x]^2 = [-6\ ;\ 4]$
3. $[f]([x]) = -([x] - 1) \cdot ([x] - 1) + 1 = [-3\ ;\ 3]$
4. $[f]([x]) = -([x] - 1)^2 + 1 = [-3\ ;\ 1]$ (Exact evaluation)

Overestimation: Dependency Problem

Problem: Multiple occurence of an interval in one equation

Necessary: Factorizations, simplifications, reformulations as far as possible
$\Rightarrow$ Reduction of overestimation and computation time

Example

$[f]([x]) = 2 \cdot [x] - [x] \cdot [x]$ and $[x] = [-1\ ;\ 2]$, Results provided by Intlab

5. Higher-order Interval Evaluation for Polynomials: Taylor Expansion ($x_m = \text{mid}([x])$) according to

$$T(f) = f(x_m) + \left(\sum_{i=1}^{n-1} \frac{\partial^i f(x)}{\partial x^i}\bigg|_{x_m} \cdot \frac{([x]-x_m)^i}{i!} \right) + \frac{\partial^n f(x)}{\partial x^n}\bigg|_{[x]} \cdot \frac{([x]-x_m)^n}{n!}$$

Overestimation: Dependency Problem

Problem: Multiple occurence of an interval in one equation

Necessary: Factorizations, simplifications, reformulations as far as possible
$\Rightarrow$ Reduction of overestimation and computation time

Example

$[f]([x]) = 2 \cdot [x] - [x] \cdot [x]$ and $[x] = [-1\ ;\ 2]$, Results provided by Intlab

5. Taylor Expansion with $x_m = \text{mid}([x]) = 0.5$

$$[f]([x]) = f(x_m) + \left.\frac{\partial f}{\partial x}\right|_{x_m} \cdot ([x] - x_m) + \left.\frac{\partial^2 f}{\partial x^2}\right|_{[x]} \cdot \frac{([x]-x_m)^2}{2!}$$

$$f(x_m) = 2 \cdot 0.5 - 0.5^2 = 0.75$$

$$\left.\frac{\partial f}{\partial x}\right|_{x_m} \cdot ([x] - x_m) = (2 - 2 \cdot x)|_{x_m} \cdot ([x] - x_m) = [-1.5\ ;\ 1.5]$$

Overestimation: Dependency Problem

Example

$[f]([x]) = 2 \cdot [x] - [x] \cdot [x]$ and $[x] = [-1\ ;\ 2]$, Results provided by Intlab

5. Taylor Expansion with $x_m = \text{mid}([x]) = 0.5$

$$[f]([x]) = 0.75 + [-1.5\ ;\ 1.5] + \left.\frac{\partial^2 f}{\partial x^2}\right|_{[x]} \cdot \frac{([x]-x_m)^2}{2!} =$$

$$\left.\frac{\partial^2 f}{\partial x^2}\right|_{[x]} \cdot \frac{([x]-x_m)\cdot([x]-x_m)}{2!} = -2 \cdot \frac{([x]-x_m)\cdot([x]-x_m)}{2!} =$$

$$-1 \cdot [-1.5\ ;\ 1.5] \cdot [-1.5\ ;\ 1.5] = [-2.25\ ;\ 2.25]$$

$\Rightarrow$ Taylor expansion will be demonstrated later with two software libraries

Overestimation: Dependency Problem

Example

$[f]([x]) = 2 \cdot [x] - [x] \cdot [x]$ and $[x] = [-1\ ;\ 2]$, Results provided by Intlab

5. Taylor Expansion with $x_m = \text{mid}([x]) = 0.5$

$$[f]([x]) = 0.75 + [-1.5\ ;\ 1.5] + \left.\frac{\partial^2 f}{\partial x^2}\right|_{[x]} \cdot \frac{([x]-x_m)^2}{2!} =$$

$$\left.\frac{\partial^2 f}{\partial x^2}\right|_{[x]} \cdot \frac{([x]-x_m)\cdot([x]-x_m)}{2!} = -2 \cdot \frac{([x]-x_m)\cdot([x]-x_m)}{2!} =$$

$$-1 \cdot [-1.5\ ;\ 1.5] \cdot [-1.5\ ;\ 1.5] = [-2.25\ ;\ 2.25]$$

$$[x]^2 = \begin{cases} [\text{min}(\underline{a}\underline{a}, \overline{a}\overline{a})\ ;\ \text{max}(\underline{a}\underline{a}, \overline{a}\overline{a})] & \text{if } 0 \notin [x] \\ [0\ ;\ \text{max}(\underline{a}\underline{a}, \overline{a}\overline{a})] & \text{if } 0 \in [x] \end{cases}$$

$$\left.\frac{\partial^2 f}{\partial x^2}\right|_{[x]} \cdot \frac{([x]-x_m)^2}{2!} = -2 \cdot \frac{([x]-x_m)^2}{2!} = -1 \cdot [-1.5\ ;\ 1.5]^2 = [-2.25\ ;\ 0]$$

$\Rightarrow$ Taylor expansion will be demonstrated later with two software libraries

Overestimation: Dependency Problem

Example

$[f]([x]) = 2 \cdot [x] - [x] \cdot [x]$ and $[x] = [-1\ ;\ 2]$, Results provided by Intlab

5. Taylor Expansion with $x_m = \mathrm{mid}([x]) = 0.5$

$$[f]([x]) = f(x_m) + \left.\frac{\partial f}{\partial x}\right|_{x_m} \cdot ([x] - x_m) + \left.\frac{\partial^2 f}{\partial x^2}\right|_{[x]} \cdot \frac{([x]-x_m)^2}{2!} =$$

$$f(x_m) = 2 \cdot 0.5 - 0.5^2 = 0.75$$

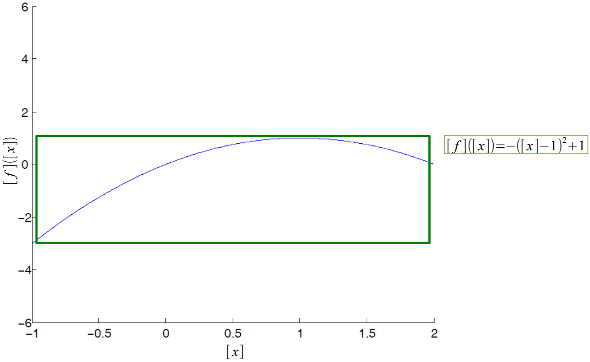
$$\left.\frac{\partial f}{\partial x}\right|_{x_m} \cdot ([x] - x_m) = (2 - 2 \cdot x)|_{x_m} \cdot ([x] - x_m) = [-1.5\ ;\ 1.5]$$

$$\left.\frac{\partial^2 f}{\partial x^2}\right|_{[x]} \cdot \frac{([x]-x_m)^2}{2!} = -2 \cdot \frac{([x]-x_m)^2}{2!} = -1 \cdot [-1.5\ ;\ 1.5]^2 = [-2.25\ ;\ 0]$$

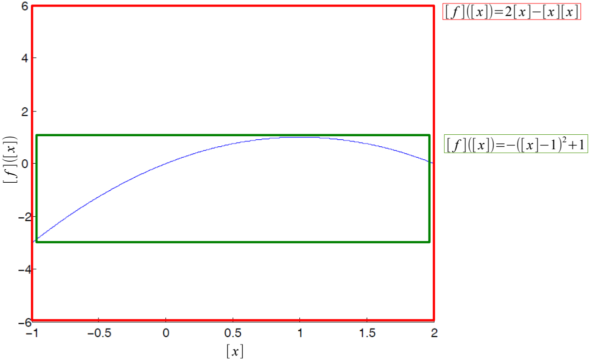
$$\Rightarrow [f]([x]) = 0.75 + [-1.5\ ;\ 1.5] + [-2.25\ ;\ 0] = [-3\ ;\ 2.25]$$

$\Rightarrow$ Taylor expansion will be demonstrated later with two software libraries

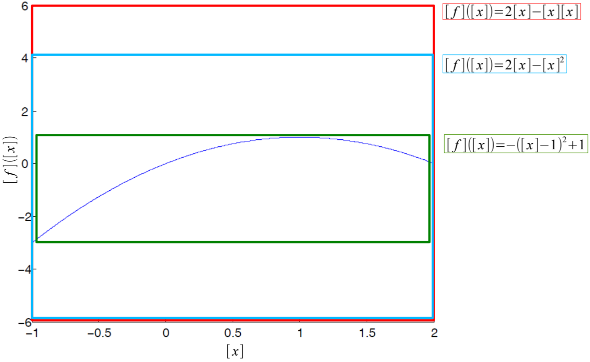
Overestimation: Dependency Problem



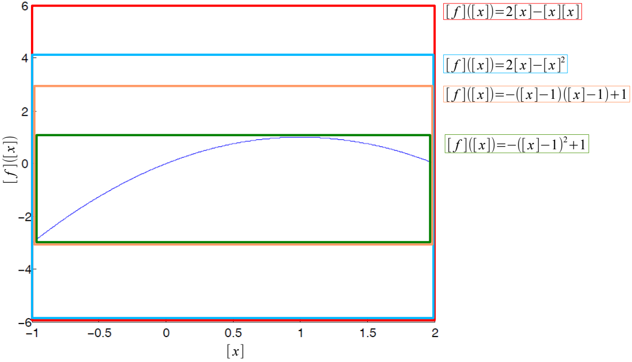
Overestimation: Dependency Problem



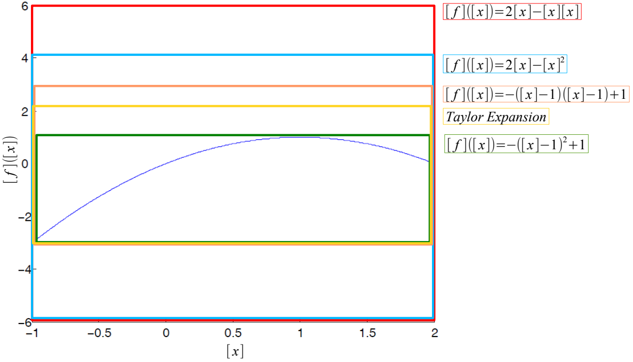
Overestimation: Dependency Problem



Overestimation: Dependency Problem

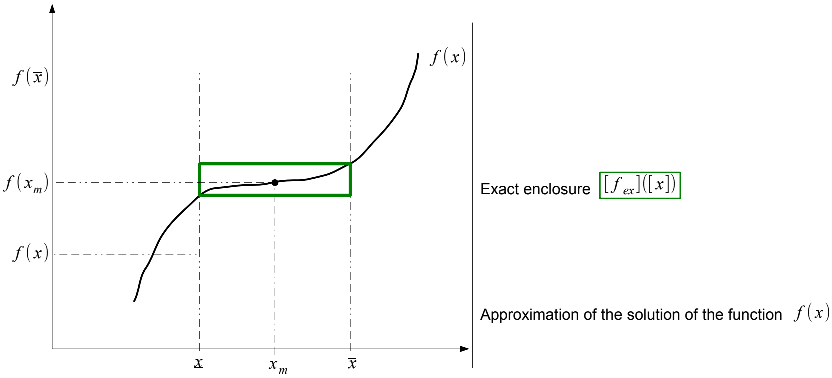


Overestimation: Dependency Problem



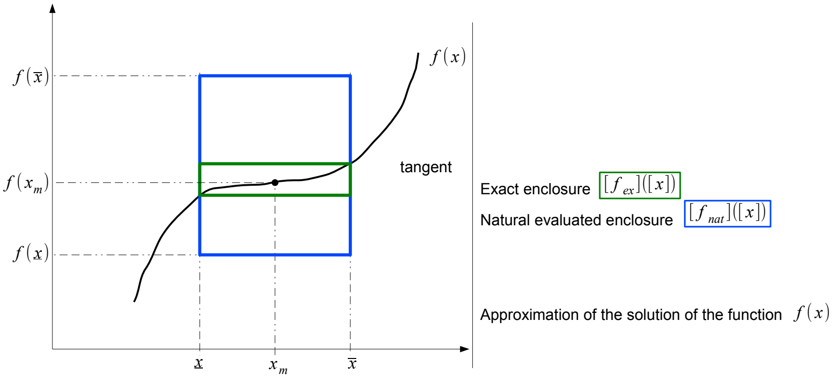
Special Case of Taylor Expansion: Mid-point Rule

$$f(x) \subseteq f_m([x]) = f(x_m) + \left.\frac{\partial f}{\partial x}\right|_{[x]} ([x] - x_m)$$



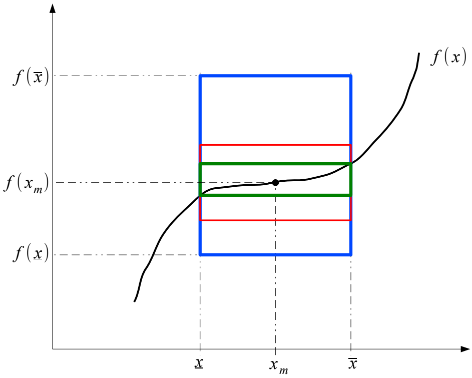
Special Case of Taylor Expansion: Mid-point Rule

$$f(x) \subseteq f_m([x]) = f(x_m) + \left.\frac{\partial f}{\partial x}\right|_{[x]} ([x] - x_m)$$



Special Case of Taylor Expansion: Mid-point Rule

$$f(x) \subseteq f_m([x]) = f(x_m) + \left.\frac{\partial f}{\partial x}\right|_{[x]} ([x] - x_m)$$



Exact enclosure $[f_{ex}]([x])$

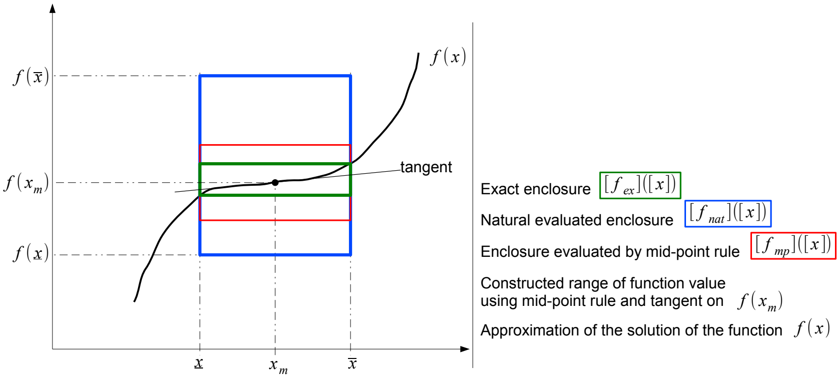
Natural evaluated enclosure $[f_{nat}]([x])$

Enclosure evaluated by mid-point rule $[f_{mp}]([x])$

Approximation of the solution of the function $f(x)$

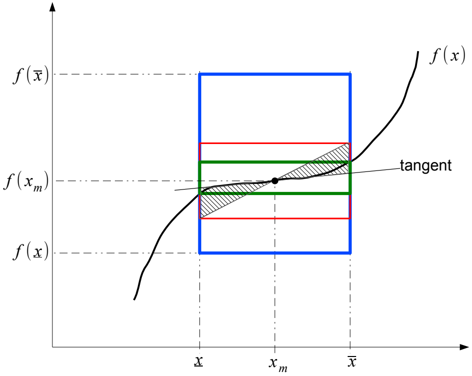
Special Case of Taylor Expansion: Mid-point Rule

$$f(x) \subseteq f_m([x]) = f(x_m) + \left.\frac{\partial f}{\partial x}\right|_{[x]} ([x] - x_m)$$



Special Case of Taylor Expansion: Mid-point Rule

$$f\left(x\right) \subseteq f_m\left(\left[x\right]\right) = f\left(x_m\right) + \left.\frac{\partial f}{\partial x}\right|_{\left[x\right]}\left(\left[x\right] - x_m\right)$$



Exact enclosure $\boxed{[f_{ex}]([x])}$

Natural evaluated enclosure $\boxed{[f_{nat}]([x])}$

Enclosure evaluated by mid-point rule $\boxed{[f_{mp}]([x])}$

Constructed range of function value using mid-point rule and tangent on $f(x_m)$

Approximation of the solution of the function $f(x)$ by using the smallest and largest slope depicted by the triangles

Monotonicity

Consider: Interval-Valued Function given by $F = x + x \cdot x$

- Two intervals $[x_1] = [-2; 4]$ and $[x_2] = [-1; 4]$ with $[x_1] \subset [x_2]$
- $F([x_1]) = [-2; 4] + [-2; 4] \cdot [-2; 4] = [-2; 4] + [-8; 16] = [-10; 20]$
- $F([x_2]) = [-1; 4] + [-1; 4] \cdot [-1; 4] = [-1; 4] + [-4; 16] = [-5; 20]$
- Consequence $F([x_2]) \subset F([x_1]) \Rightarrow F$ is an inclusion monotonic function
- 4 basic arithmetic operators are also `inclusion monotonic`

Consequence for Calculating with Intervals

- Splitting of large intervals
- Hull of all evaluations with the subintervals
- Tighter range bounds than with original interval

Monotonicity

Consider: Interval-Valued Function given by $F = x + x \cdot x$

- Two intervals $[x_1] = [-2; 4]$ and $[x_2] = [-1; 4]$ with $[x_1] \subset [x_2]$
- $F([x_1]) = [-2; 4] + [-2; 4] \cdot [-2; 4] = [-2; 4] + [-8; 16] = [-10; 20]$
- $F([x_2]) = [-1; 4] + [-1; 4] \cdot [-1; 4] = [-1; 4] + [-4; 16] = [-5; 20]$
- Consequence $F([x_2]) \subset F([x_1]) \Rightarrow F$ is an inclusion monotonic function
- 4 basic arithmetic operators are also `inclusion monotonic`

Monitonicity of a Function Using Derivatives

$$\left.\frac{\partial F}{\partial \mathbf{x}}\right|_{x \in [x]} < 0 \quad \Rightarrow F \in [F(\overline{x}); F(\underline{x})]$$

$$\left.\frac{\partial F}{\partial \mathbf{x}}\right|_{x \in [x]} > 0 \quad \Rightarrow F \in [F(\underline{x}); F(\overline{x})]$$

Overestimation: Wrapping Effect —Example

Discrete System Model

$[\mathbf{x}](t_{k+1}) = \mathbf{A} \cdot [\mathbf{x}](t_k)$ with $[\mathbf{x}](t_0) = \begin{bmatrix} [-1\ ;\ 1] \\ [-1\ ;\ 1] \end{bmatrix}$ and $\mathbf{A} = \frac{1}{2}\sqrt{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$

Aim

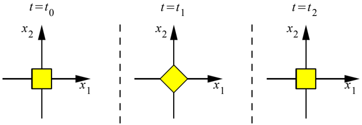
Evaluation of interval enclosure $[\mathbf{x}](t_{k+1})$

Problem in Engineering Tasks

Uncertainty in parameters, significantly larger than representation errors of floating-point values (rounding errors)

Overestimation: Wrapping Effect — Example

$$[\mathbf{x}](t_{k+1}) = \mathbf{A} \cdot [\mathbf{x}](t_k) \text{ with } [\mathbf{x}](t_0) = \begin{bmatrix} [-1\ ;\ 1] \\ [-1\ ;\ 1] \end{bmatrix} \text{ and } \mathbf{A} = \frac{1}{2}\sqrt{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

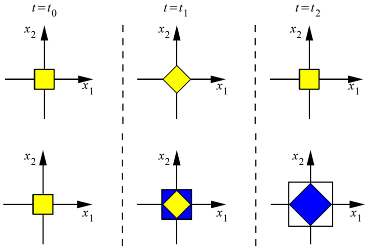


- Exact recursive evaluation

$\Rightarrow$ Rotation of 45° due to structure of system matrix $\mathbf{A}$

Overestimation: Wrapping Effect — Example

$$[\mathbf{x}](t_{k+1}) = \mathbf{A} \cdot [\mathbf{x}](t_k),\ [\mathbf{x}](t_0) = \begin{bmatrix} [-1\ ;\ 1] \\ [-1\ ;\ 1] \end{bmatrix},\ \mathbf{A} = \mathbf{A}_k = \frac{1}{2}\sqrt{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$



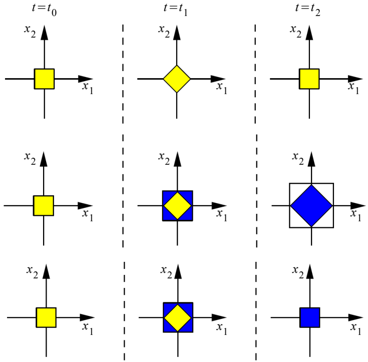
- Traditional recursive interval evaluation (using e.g. Intlab)

$$\begin{aligned} [\mathbf{x}](t_1) &= \mathbf{A}\,[\mathbf{x}](t_0) \\ [\mathbf{x}](t_2) &= \mathbf{A}\,[\mathbf{x}](t_1) \\ &\vdots \\ [\mathbf{x}](t_{k+1}) &= \mathbf{A}\,[\mathbf{x}](t_k) \end{aligned}$$

$\Rightarrow$ Exponential growth of the enclosing interval boxes

Overestimation: Wrapping Effect — Example

$$[\mathbf{x}](t_{k+1}) = \mathbf{A} \cdot [\mathbf{x}](t_k),\ [\mathbf{x}](t_0) = \begin{bmatrix} [-1\ ;\ 1] \\ [-1\ ;\ 1] \end{bmatrix},\ \mathbf{A} = \mathbf{A}_k = \frac{1}{2}\sqrt{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$



- Intelligent recursive evaluation (affine): Modified system matrix $\tilde{\mathbf{A}}_k = \mathbf{A}\tilde{\mathbf{A}}_{k-1}$

$$[\mathbf{x}](t_1) = \mathbf{A}\,[\mathbf{x}](t_0) = \tilde{\mathbf{A}}_0\,[\mathbf{x}](t_0)$$

$$[\mathbf{x}](t_2) = \mathbf{A}\tilde{\mathbf{A}}_0\,[\mathbf{x}](t_0) = \tilde{\mathbf{A}}_1\,[\mathbf{x}](t_0)$$

$$\vdots$$

$$[\mathbf{x}](t_{k+1}) = \mathbf{A}\tilde{\mathbf{A}}_{k-1}\,[\mathbf{x}](t_0) = \tilde{\mathbf{A}}_k\,[\mathbf{x}](t_0)$$

$\Rightarrow$ Significant reduction of the wrapping effect for linear systems

Affine System Representation for Discrete Systems

Advantages

- Directly mapping of interval variables to their initial intervals in each time step
- No dependencies between intervals $\Rightarrow$ no interval box rotations

Discretization depends on

- Additive interval or multiplicatively coupled parameter interval
- Input variable constant or changing
- Explicit or implicit Euler method

Affine System Representation for a SISO System

Case 1: Additive interval uncertainty $[a]$, $u(t_k) \neq$ const, step size $T = 1$

$$f(y(t_k), [u(t_k)]) = 2 \cdot [y](t_k) + 1 \cdot u(t_k) + 3 \cdot [a]$$

$$[\mathbf{x}](t_{k+1}) = \underbrace{\begin{bmatrix} [y](t_{k+1}) \\ [a](t_{k+1}) \end{bmatrix}}_{\text{extended state vector } \mathbf{x}(t_{k+1})} = \underbrace{\begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix}}_{\mathbf{M}} \cdot \underbrace{\begin{bmatrix} [y](t_k) \\ [a](t_k) \end{bmatrix}}_{\mathbf{x}(t_k)} + \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot u(t_k)}_{\boldsymbol{\rho}(t_k)}$$

E.g. Explicit Euler Discetization (time discretization error neglected)

$$[\mathbf{x}](t_{k+1}) = \mathbf{M}(t_{k+1}) \cdot [\mathbf{x}](t_0) + \boldsymbol{\gamma}(t_{k+1}) \quad \text{with}$$

$$\mathbf{M}(t_k) = \mathbf{M} \quad \Rightarrow \quad \mathbf{M}(t_{k+1}) = \mathbf{M} \cdot \mathbf{M}(t_k)$$

$$\boldsymbol{\gamma}(t_{k+1}) = \mathbf{M}(t_k) \cdot \boldsymbol{\gamma}(t_k) + T \cdot \boldsymbol{\rho}(t_k)$$

initial conditions $\mathbf{x}(t_0) = \begin{bmatrix} [y](t_0) \\ [a](t_0) \end{bmatrix}$, $\mathbf{M}(t_0) = \mathbf{I}^{2\times 2}$, $\boldsymbol{\gamma}(t_0) = \mathbf{0}$

Affine System Representation for a SISO System

Case 2: Additive interval uncertainty $[a]$, u=const, step size $T = 1$

$$f(y(t), [u(t)]) = 2 \cdot [y](t) + 1 \cdot u(t) + 3 \cdot [a]$$

$$[\mathbf{x}](t_{k+1}) = \underbrace{\begin{bmatrix} [y](t_{k+1}) \\ [a](t_{k+1}) \\ u(t_{k+1}) \end{bmatrix}}_{\text{extended state vector } \mathbf{x}(t_{k+1})} = \underbrace{\begin{bmatrix} 2 & 3 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{M}} \cdot \underbrace{\begin{bmatrix} [y](t_k) \\ [a](t_k) \\ u(t_k) \end{bmatrix}}_{\mathbf{x}(t_k)}$$

Explicit Euler Discetization (time discretization error neglected)

$$[\mathbf{x}](t_{k+1}) = \mathbf{M}(t_{k+1}) \cdot [\mathbf{x}](t_0) \quad \text{with}$$

$$\mathbf{M}(t_k) = \mathbf{M} \quad \Rightarrow \quad \mathbf{M}(t_{k+1}) = \mathbf{M} \cdot \mathbf{M}(t_k)$$

$$[\mathbf{x}](t_0) = \begin{bmatrix} [y](t_0) \\ [a](t_0) \\ u(t_0) \end{bmatrix}$$

Affine System Representation — Comparison

Implicit Euler Method ($u(t_{k+1}) = u(t_k) = \text{const}$)

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t)) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t)$$

$$\Rightarrow \mathbf{f}(\mathbf{x}(t_{k+1}), u(t_{k+1})) \approx \frac{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)}{T}$$

$$\mathbf{x}(t_{k+1}) = (\mathbf{I} - T \cdot \mathbf{A})^{-1} \cdot (\mathbf{x}(t_k) + T \cdot \mathbf{b} \cdot u(t_{k+1}))$$

$$\underbrace{\begin{bmatrix} \mathbf{x}(t_{k+1}) \\ u(t_{k+1}) \end{bmatrix}}_{\tilde{\mathbf{x}}(t_{k+1})} = \underbrace{\begin{bmatrix} (\mathbf{I} - T \cdot \mathbf{A})^{-1} & (\mathbf{I} - T \cdot \mathbf{A})^{-1} \cdot T \cdot \mathbf{b} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\tilde{\mathbf{A}}} \cdot \underbrace{\begin{bmatrix} \mathbf{x}(t_k) \\ u(t_k) \end{bmatrix}}_{\tilde{\mathbf{x}}(t_k)}$$

Affine System Representation — Comparison

Explicit Euler Method ($u(t_{k+1}) = u(t_k) = \text{const}$)

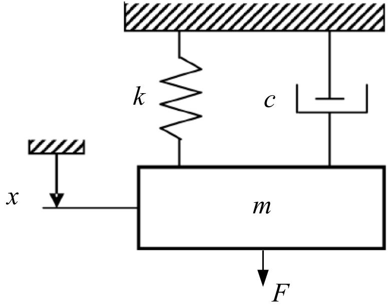
$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t)) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}u(t)$$

$$\Rightarrow \mathbf{f}(\mathbf{x}(t_k), u(t_k)) \approx \frac{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)}{T}$$

$$\mathbf{x}(t_{k+1}) = (\mathbf{I} + T \cdot \mathbf{A}) \cdot \mathbf{x}(t_k) + T \cdot \mathbf{b} \cdot u(t_k)$$

$$\underbrace{\begin{bmatrix} \mathbf{x}(t_{k+1}) \\ u(t_{k+1}) \end{bmatrix}}_{\tilde{\mathbf{x}}(t_{k+1})} = \underbrace{\begin{bmatrix} (\mathbf{I} + T \cdot \mathbf{A}) & T \cdot \mathbf{b} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\tilde{\mathbf{A}}} \cdot \underbrace{\begin{bmatrix} \mathbf{x}(t_k) \\ u(t_k) \end{bmatrix}}_{\tilde{\mathbf{x}}(t_k)}$$

Affine System Representation for a One-Mass Oscillator



$$m \cdot \ddot{x}(t) + c \cdot x(t) + k \cdot x(t) = F(t)$$

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t)$$

Affine System Representation for a One-Mass Oscillator

$$m \cdot \ddot{x}(t) + c \cdot x(t) + k \cdot x(t) = F(t)$$

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t)$$

Euler Discretization with constant input variable

$$\tilde{\mathbf{x}}(t_{k+1}) = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}(t_k)$$

$$\begin{aligned} &\tilde{\mathbf{x}}(t_1) = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}(t_0) \\ &\tilde{\mathbf{x}}(t_2) = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}(t_1) = \tilde{\mathbf{A}} \cdot (\tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}(t_0)) = \tilde{\mathbf{A}}^2 \cdot \tilde{\mathbf{x}}(t_0) \\ &\vdots \\ &\tilde{\mathbf{x}}(t_{k+1}) = \tilde{\mathbf{A}}^k \cdot \tilde{\mathbf{x}}(t_0) \end{aligned}$$

Affine System Representation for a One-Mass Oscillator

$$m \cdot \ddot{x}(t) + c \cdot x(t) + k \cdot x(t) = F(t)$$

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t)$$

Euler Discretization with constant input variable

$$\tilde{\mathbf{x}}(t_{k+1}) = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}(t_k)$$

$$\tilde{\mathbf{x}}(t_1) = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}(t_0)$$
$$\tilde{\mathbf{x}}(t_2) = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}(t_1) = \tilde{\mathbf{A}} \cdot (\tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}}(t_0)) = \tilde{\mathbf{A}}^2 \cdot \tilde{\mathbf{x}}(t_0)$$
$$\vdots$$
$$\tilde{\mathbf{x}}(t_{k+1}) = \tilde{\mathbf{A}}^k \cdot \tilde{\mathbf{x}}(t_0)$$

Example in MATLAB

Part 1

1. Fundamentals of Interval Arithmetic

Software Demonstration

Software Demonstration of Interval Arithmetics

- INTLAB: INTerval LABoratory — Matlab toolbox for Reliable Computing
- C-XSC — C++ Class Library

Importance of Verified Computing

- Floating-point arithmetics on today's computer is always affected by a maximum accuracy
  $\Rightarrow$ rounded results differ at most by 1 unit in the last place from the exact result
- After further calculations, the result may be wrong because of rounding $\Rightarrow$ Results have to be verified

INTerval LABoratory

Development by Prof. Dr. Siegfried M. Rump, Hamburg University of Technology

http://www.ti3.tu-harburg.de/rump/intlab/

Standard interval arithmetic

- Arithmetic operators $+, -, \cdot, /$
- Real and complex intervals

Automatic Differentiation

- Forward mode: forward substitution to find the derivatives
- Compute derivatives using the chain rule for composite functions
- Calculate an enclosure of the true derivative of an interval function

INTerval LABoratory

http://www.ti3.tu-harburg.de/rump/intlab/

Verified Functions for Linear Systems of Equations

- Solution of linear systems of equations in a verified way
- Computation of an enclosure of the solution hull
- Aim: produce a tight bound on the true solution

Rounding Mode

- Function `setround(y)`: changes the rounding mode of the processor to the nearest (0), round down (-1), round up (1)
- Function `getround` outputs the current rounding mode

INTerval LABoratory

http://www.ti3.tu-harburg.de/rump/intlab/

Verified Functions for Linear Systems of Equations

- Solution of linear systems of equations in a verified way
- Computation of an enclosure of the solution hull
- Aim: produce a tight bound on the true solution

Rounding Mode

- Function `setround(y)`: changes the rounding mode of the processor to the nearest (0), round down (-1), round up (1)
- Function `getround` outputs the current rounding mode

Matlab Example: intlab_fundamentals.m and taylor_expansion.m

C-XSC

http://www2.math.uni-wuppertal.de/~xsc/xsc/cxsc.html

Information

- C++ Class Library for Extended Scientific Computing
- Compatible to Windows, Linux, Mac Os

Data Types

- real, interval, complex, cinterval (complex interval)
- rvector, ivector, cvector, civector (complex interval vector)
- rmatrix, imatrix, cmatrix, cimatrix (complex interval matrix)

C-XSC

http://www2.math.uni-wuppertal.de/~xsc/xsc/cxsc.html

Data Types

- real, interval, complex, cinterval (complex interval)
- rvector, ivector, cvector, civector (complex interval vector)
- rmatrix, imatrix, cmatrix, cimatrix (complex interval matrix)

Rounding Mode

- by-default: all operations are only one rounding away from the exact result
- Modes: long fix-point accumulator for dot product computations (default), pure floating point operations, DotK algorithm (based on so-called error free transformations)

C-XSC

http://www2.math.uni-wuppertal.de/~xsc/xsc/cxsc.html

Data Types fundamentals.cpp

- real, interval, complex, cinterval (complex interval)
- rvector, ivector, cvector, civector (complex interval vector)
- rmatrix, imatrix, cmatrix, cimatrix (complex interval matrix)

Rounding Mode

- by-default: all operations are only one rounding away from the exact result
- Modes: long fix-point accumulator for dot product computations (default), pure floating point operations, DotK algorithm (based on so-called error free transformations)

FADBAD++

http://www.fadbad.com/fadbad.html#General_introduction

General

- Flexible Automatic Differentiation using templates and operator overloading in C++
- Implementing the forward, backward and Taylor methods utilizing C++ templates and operator overloading
- Differentiate a C++ function by replacing all occurrences of the original arithmetic type with the AD-template version
- Possible to generate high-order derivatives

FADBAD++

http://www.fadbad.com/fadbad.html#General_introduction

General taylor_expansion_FF.cpp and taylor_expansion_T.cpp

- Flexible Automatic Differentiation using templates and operator overloading in C++
- Implementing the forward, backward and Taylor methods utilizing C++ templates and operator overloading
- Differentiate a C++ function by replacing all occurrences of the original arithmetic type with the AD-template version
- Possible to generate high-order derivatives

Advantage of Using C++ instead of Intlab

Interface to rapid control prototyping environments is possible

Thank you for your attention!

All presentations, examples and selected publications will be available at
http://www.com.uni-rostock.de/ecc15/
in the 1st week of August

User: ECC15
Password: intervals-are-fun