

Intervals and OpenMP:

Towards an Efficient Parallel Result-Verifying Nonlinear Solver

EWOMP 2003
Aachen University, Germany
September 23, 2003

Thomas Beelitz



Scientific Computing
University of Wuppertal
beelitz@math.uni-wuppertal.de



Joint work with

- Christian Bischof, Scientific Computing, Aachen University
- Bruno Lang, Scientific Computing, University of Wuppertal
- Wolfgang Marquardt, Process Engineering Systems, Aachen University
- Martin Mönnigmann, Process Engineering Systems, Aachen University

This work is partially funded by **VolkswagenStiftung**.

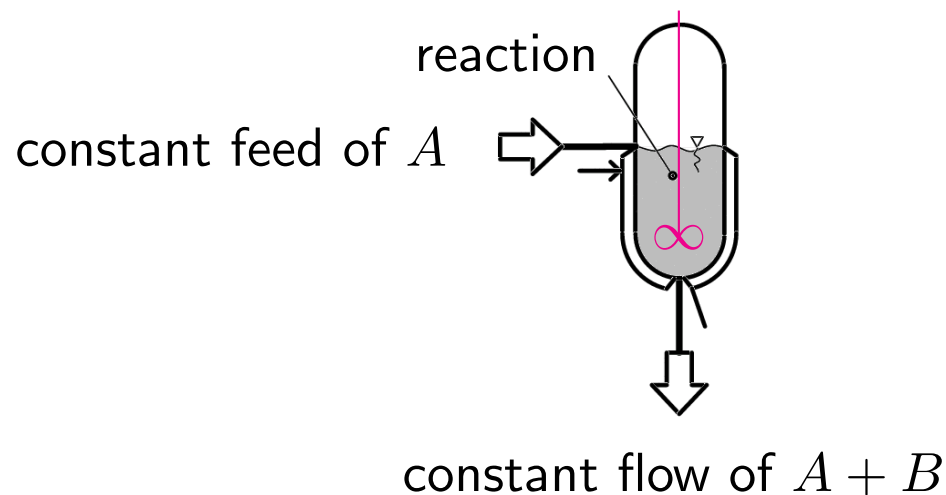
Overview

- Singularities in chemical processes
- Approaches for determining the singularities
- Interval arithmetic
- Basic branch-and-bound-algorithm + OpenMP implementation
- Numerical results
- Summary

Singularities in chemical processes

Example: **CSTR** (continuous-flow stirred-tank reactor)

general model for exothermic reactions $A \rightarrow B$ (e.g., hydrations, oxidations)



Process description (ODE)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{p}, \mathbf{x})$$

with

(parameters) $\mathbf{p} = (\text{inflow concentration, normalized time})^T$ and
(state) $\mathbf{x} = (\text{concentration, temperature})^T$.

Process description (ODE)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{p}, \mathbf{x})$$

with

$$\begin{array}{ll} \text{(parameters)} & \mathbf{p} = (\text{inflow concentration, normalized time})^T \\ \text{(state)} & \mathbf{x} = (\text{concentration, temperature})^T. \end{array} \quad \text{and}$$

Steady-state analysis $\dot{\mathbf{x}} \equiv 0$ leads to a nonlinear system

$$\mathbf{f}(\mathbf{p}, \mathbf{x}) = 0, \quad \mathbf{p} \in \mathbb{R}^k, \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{f} : \mathbb{R}^{n+k} \rightarrow \mathbb{R}^n$$

Process description (ODE)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{p}, \mathbf{x})$$

with

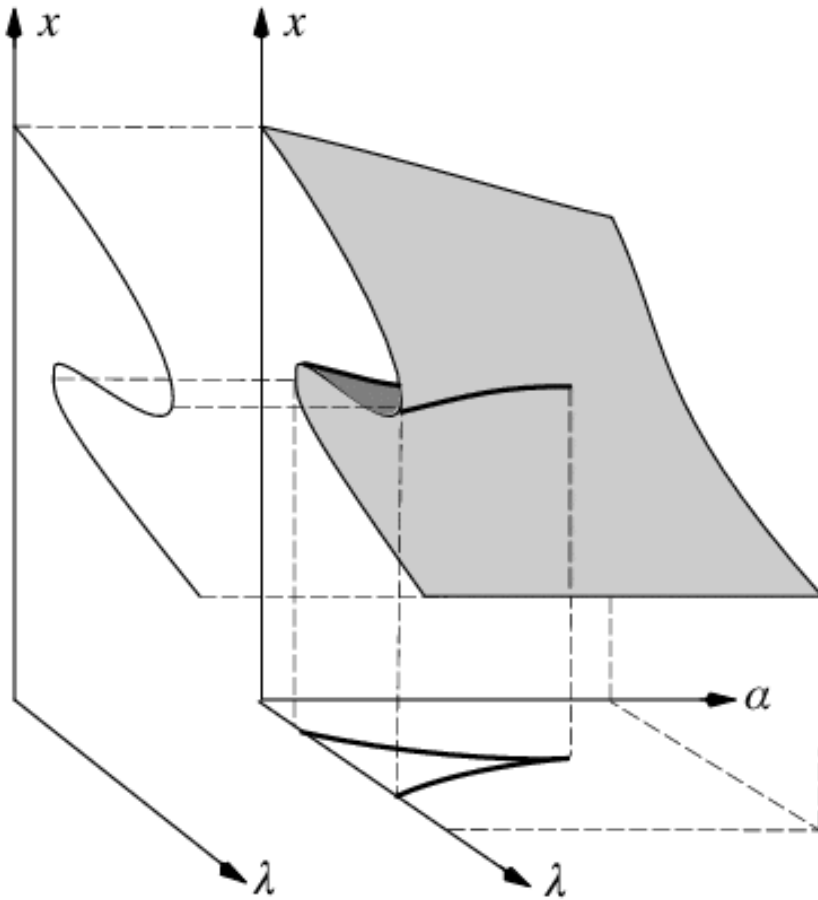
$$\begin{array}{ll} \text{(parameters)} & \mathbf{p} = (\text{inflow concentration, normalized time})^T \\ \text{(state)} & \mathbf{x} = (\text{concentration, temperature})^T. \end{array} \quad \text{and}$$

Steady-state analysis $\dot{\mathbf{x}} \equiv 0$ leads to a nonlinear system

$$\mathbf{f}(\mathbf{p}, \mathbf{x}) = 0, \quad \mathbf{p} \in \mathbb{R}^k, \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{f} : \mathbb{R}^{n+k} \rightarrow \mathbb{R}^n$$

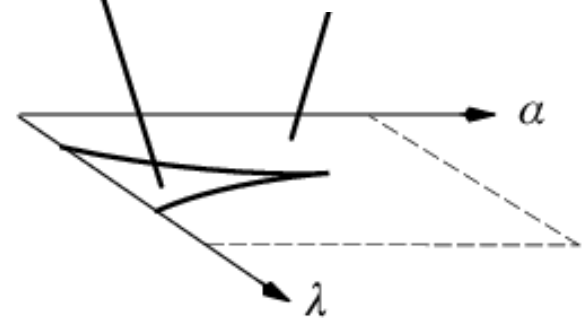
Investigate the set of solutions with respect to the **parameters** \mathbf{p} .

Singularities

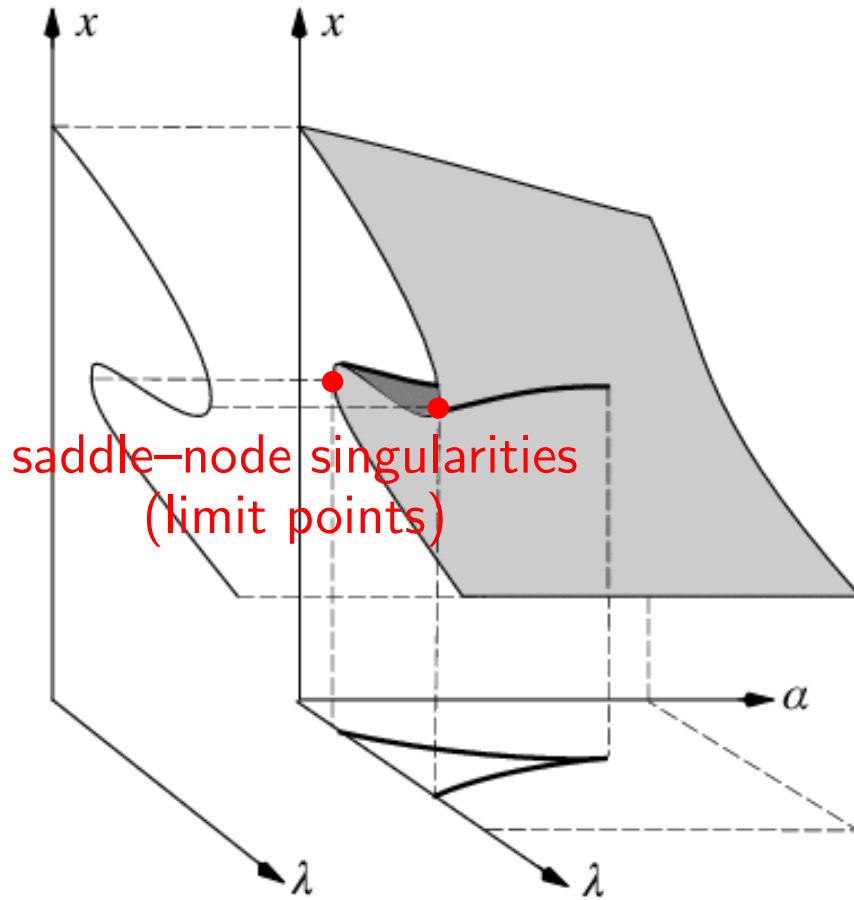


multiple states

unique states

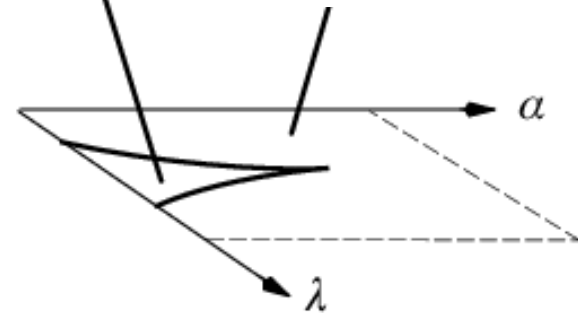


Singularities

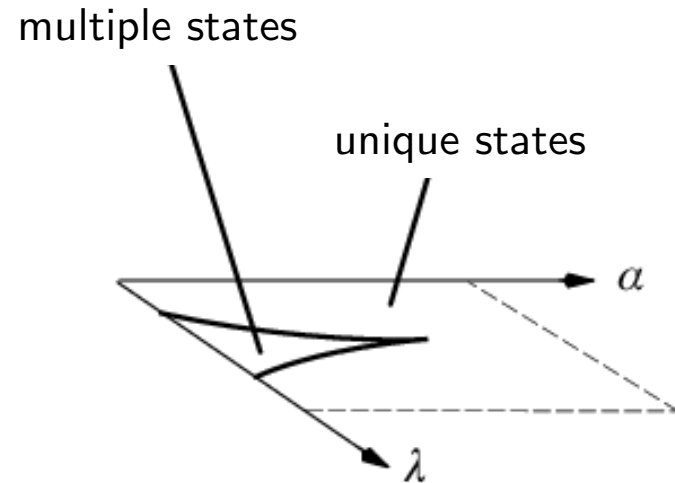
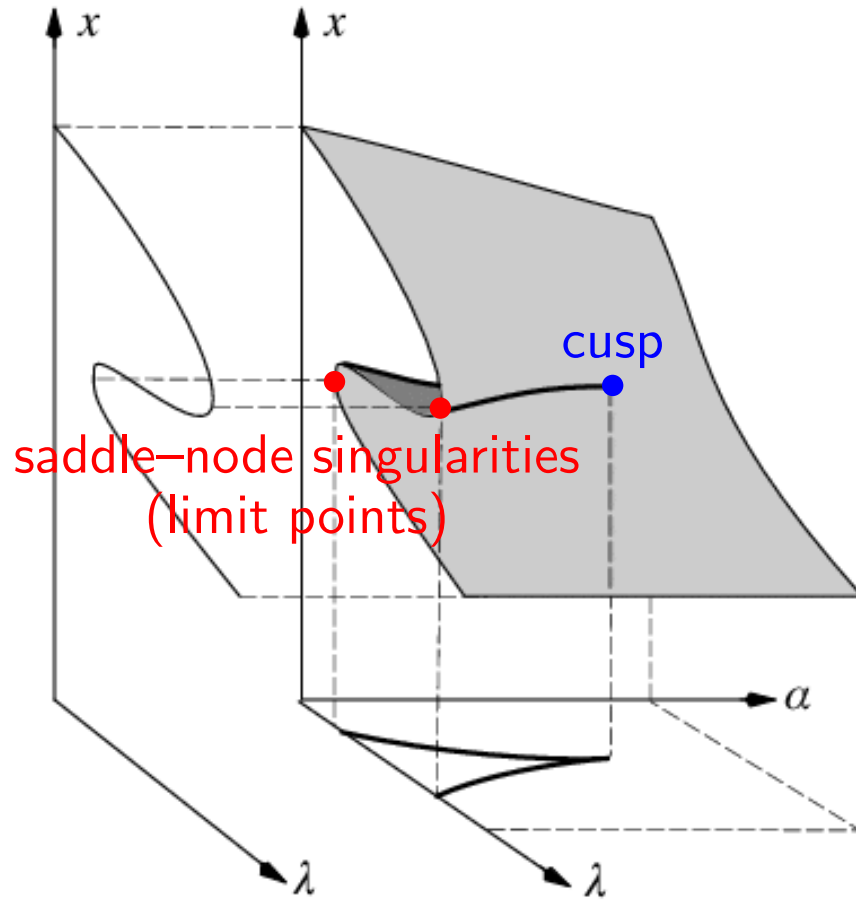


multiple states

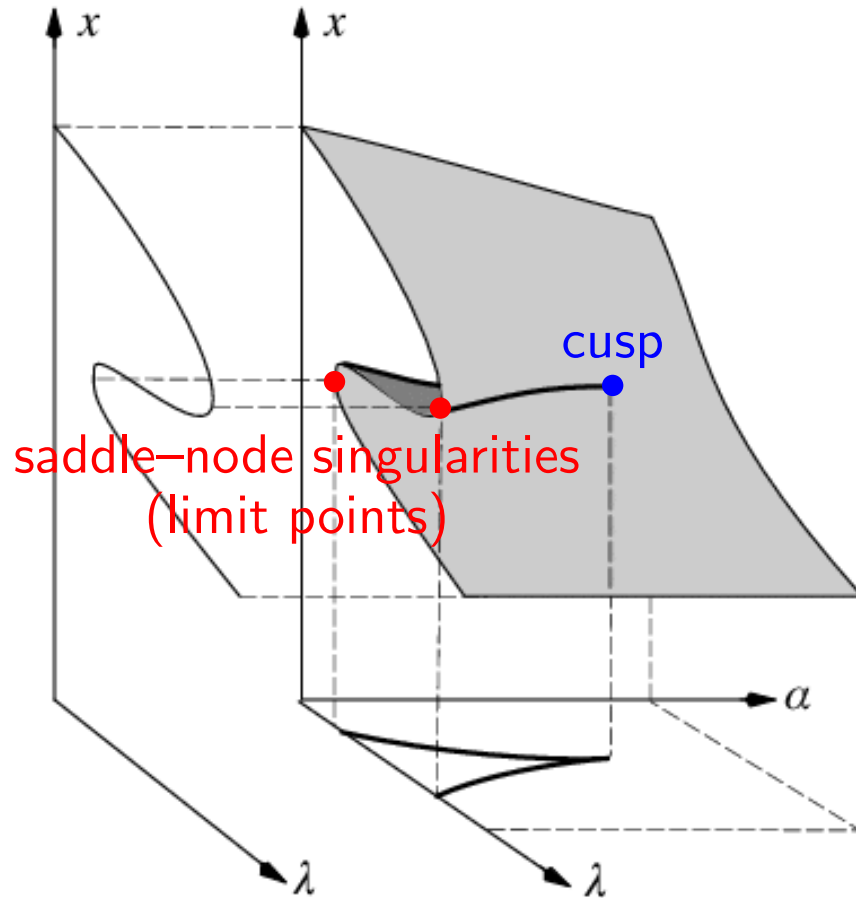
unique states



Singularities



Singularities

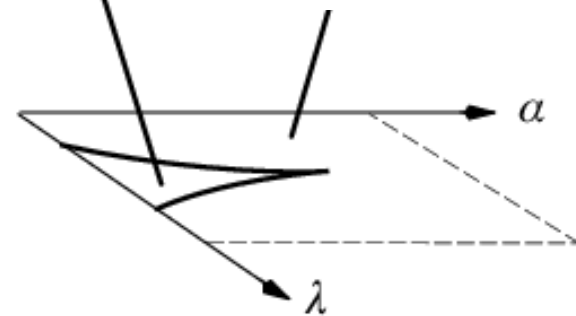


Jumping between states:

- drastically reduced gain or
- "ignition" of the process

multiple states

unique states



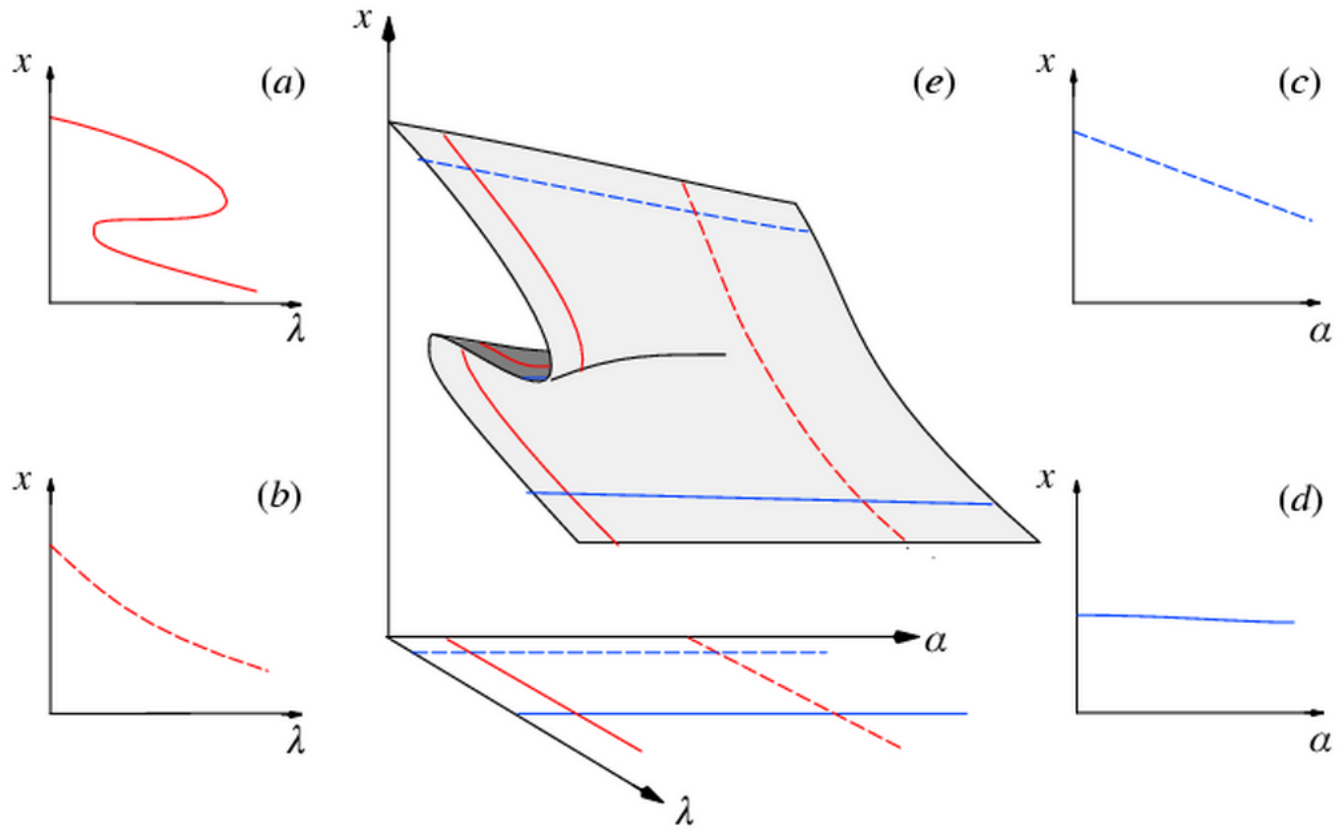
Approaches for determining the singularities

Goal:

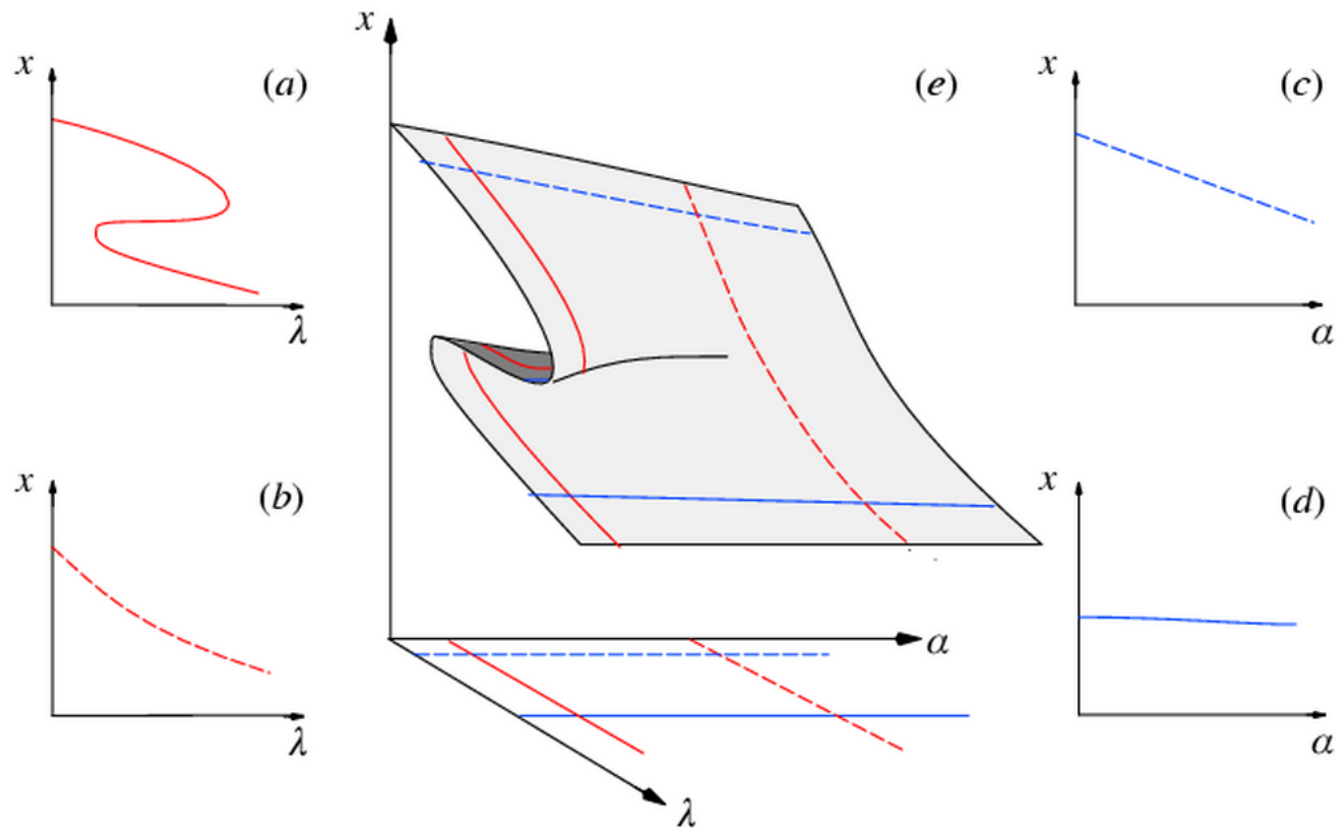
- **prove absence** of certain singularities or
- determine (guaranteed) **bounds** for **all** singularities of a certain type

within given ranges for the variables and parameters.

Continuation approach



Continuation approach



may easily miss the singularities or even components of the manifold

Augmented system

for cusps (cf. Golubitzky/Schaeffer, 1985):

$$\begin{aligned} \mathbf{f}(\mathbf{p}, \mathbf{x}) &= \mathbf{0} \\ A\mathbf{v} &= \mathbf{0} \\ A^T\mathbf{w} + \rho\mathbf{v} &= \mathbf{0} \\ \|\mathbf{v}\|_2^2 = \|\mathbf{w}\|_2^2 &= 1 \\ \mathbf{w}^T\mathcal{B}\mathbf{v}\mathbf{v} &= 0 \end{aligned}$$

where

$$A = \frac{\partial \mathbf{f}(\mathbf{p}, \mathbf{x})}{\partial \mathbf{x}},$$

$$\mathcal{B} = \frac{\partial^2 \mathbf{f}(\mathbf{p}, \mathbf{x})}{\partial \mathbf{x}^2},$$

$$\mathbf{v}, \mathbf{w} \in \mathbb{R}^n,$$

$$\rho \in \mathbb{R} \quad (\text{regularization variable})$$

For the CSTR: 9 equations in 9 unknowns: $F(\mathbf{z}) = 0$

Augmented system

for cusps (cf. Golubitzky/Schaeffer, 1985):

$$\begin{aligned} \mathbf{f}(\mathbf{p}, \mathbf{x}) &= \mathbf{0} \\ A\mathbf{v} &= \mathbf{0} \\ A^T\mathbf{w} + \rho\mathbf{v} &= \mathbf{0} \\ \|\mathbf{v}\|_2^2 = \|\mathbf{w}\|_2^2 &= 1 \\ \mathbf{w}^T\mathcal{B}\mathbf{v}\mathbf{v} &= 0 \end{aligned}$$

where

$$A = \frac{\partial \mathbf{f}(\mathbf{p}, \mathbf{x})}{\partial \mathbf{x}},$$

$$\mathcal{B} = \frac{\partial^2 \mathbf{f}(\mathbf{p}, \mathbf{x})}{\partial \mathbf{x}^2},$$

$$\mathbf{v}, \mathbf{w} \in \mathbb{R}^n,$$

$$\rho \in \mathbb{R} \quad (\text{regularization variable})$$

For the CSTR: 9 equations in 9 unknowns: $F(\mathbf{z}) = 0$

\Rightarrow **verified nonlinear solver**

Interval arithmetic

Real intervals

$$\mathbb{IR} := \{[a] = [\underline{a}, \bar{a}] : \underline{a}, \bar{a} \in \mathbb{R}, \underline{a} \leq \bar{a}\}$$

Interval operations

For $\circ \in \{+, -, \times, /\}$ and $[a], [b] \in \mathbb{IR}$ ($0 \notin [b]$ in case of $\circ = /$) let

$$[a] \circ [b] := \{a \circ b : a \in [a], b \in [b]\} \in \mathbb{IR}$$

For $\varphi \in \{\sqrt{}, \sin, \cos, \dots\}$ and $[a] \in \mathbb{IR}$ ($\underline{a} \geq 0$ for $\varphi = \sqrt{} \dots$) let

$$\varphi[a] := \{\varphi(a) : a \in [a]\} \in \mathbb{IR}$$

Interval arithmetic on the computer

- Calculate with interval boundaries

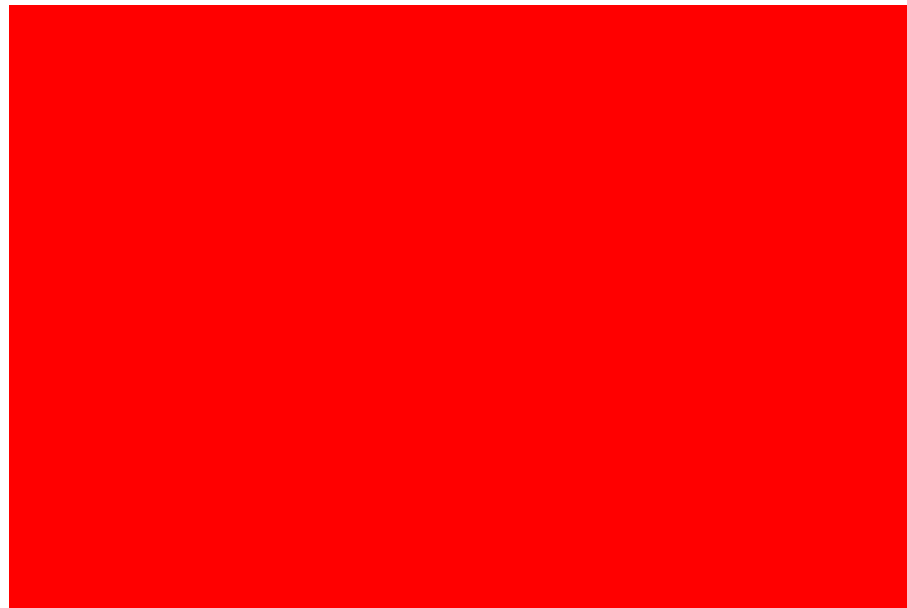
$$\left\{ \begin{array}{l} [a] + [b] = [\underline{a} + \underline{b}, \bar{a} + \bar{b}] \\ [a] \cdot [b] = [\min S, \max S] \quad \text{with} \\ \qquad \qquad \qquad S = \{\underline{a}\underline{b}, \underline{a}\bar{b}, \bar{a}\underline{b}, \bar{a}\bar{b}\} \\ \sqrt{[a]} = [\sqrt{\underline{a}}, \sqrt{\bar{a}}] \\ \qquad \qquad \qquad \vdots \end{array} \right.$$

- Rounding errors can be accounted for by **directed rounding** of the interval boundaries (\Rightarrow results are guaranteed)

Basic branch-and-bound algorithm

```
function Check( $[z]$ ) : /* Checks if the box  $[z]$  may contain a solution */  
if  $0 \in F_i[z]$  for each  $i = 1, \dots, n$  /*  $F_i[z]$ : enclosure of the range of  $F_i$  over  $[z]$  */  
    and none of the more sophisticated tests allows discarding  $[z]$   
then  
    if  $[z]$  is small enough  
        then  
            add  $[z]$  to the list of possible solutions  
    else  
        split  $[z]$  into two subboxes  $[z^{(1)}], [z^{(2)}]$   
        Check( $[z^{(1)}]$ )  
        Check( $[z^{(2)}]$ )
```

Basic branch-and-bound algorithm



- may contain a solution
- cannot contain a solution

Basic branch-and-bound algorithm

$$0 \in F_i[\mathbf{z}] ?$$

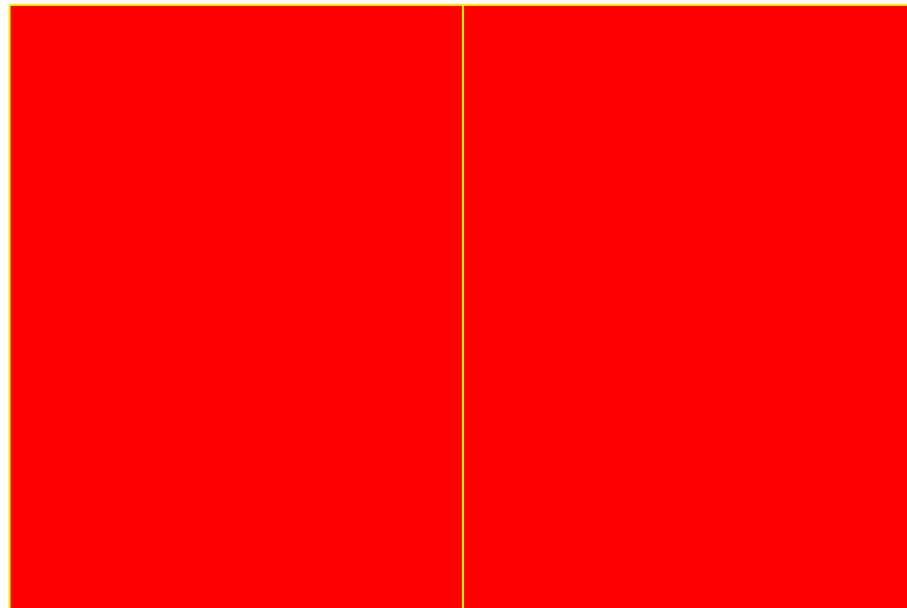
- may contain a solution
- cannot contain a solution

Basic branch-and-bound algorithm



- may contain a solution
- cannot contain a solution

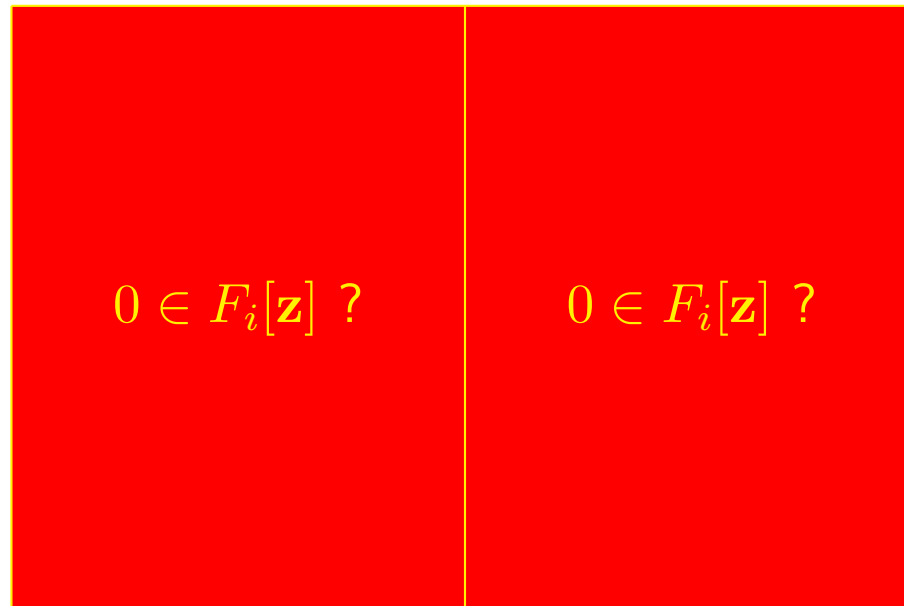
Basic branch-and-bound algorithm



■ may contain a solution

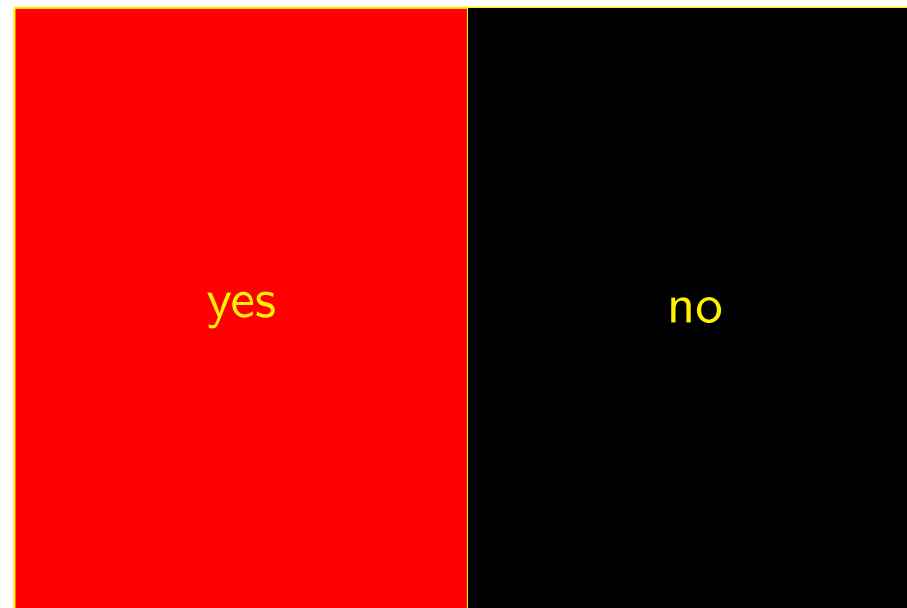
■ cannot contain a solution

Basic branch-and-bound algorithm



- may contain a solution
- cannot contain a solution

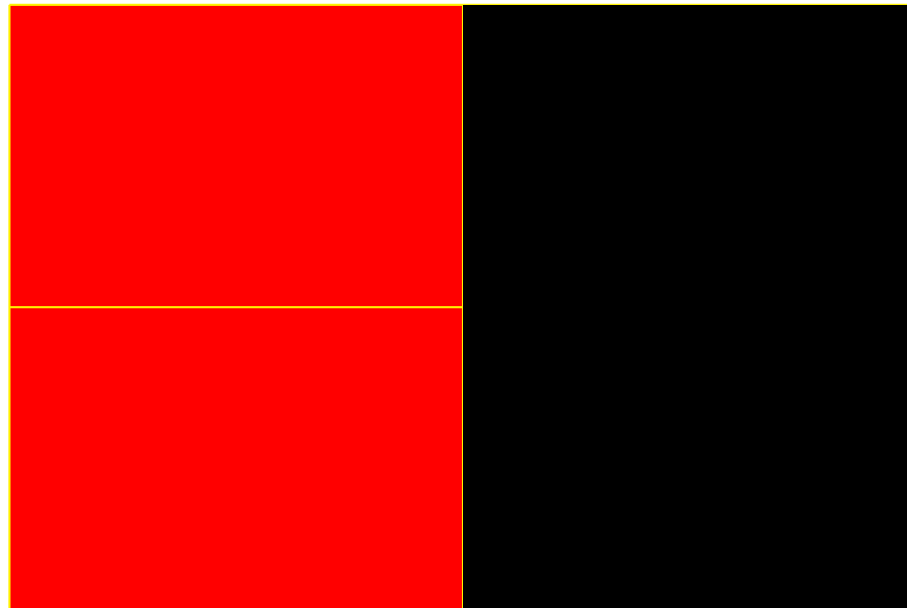
Basic branch-and-bound algorithm



■ may contain a solution

■ cannot contain a solution

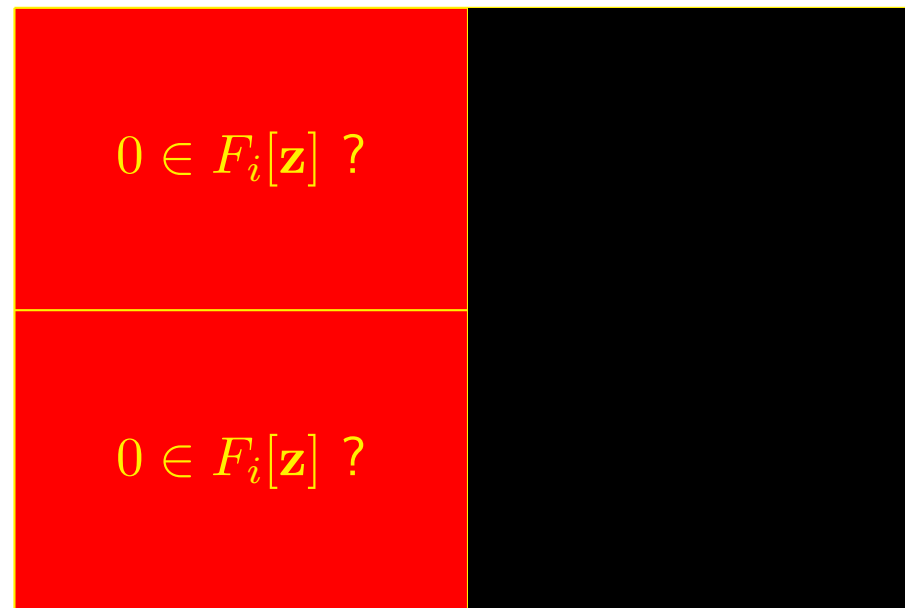
Basic branch-and-bound algorithm



■ may contain a solution

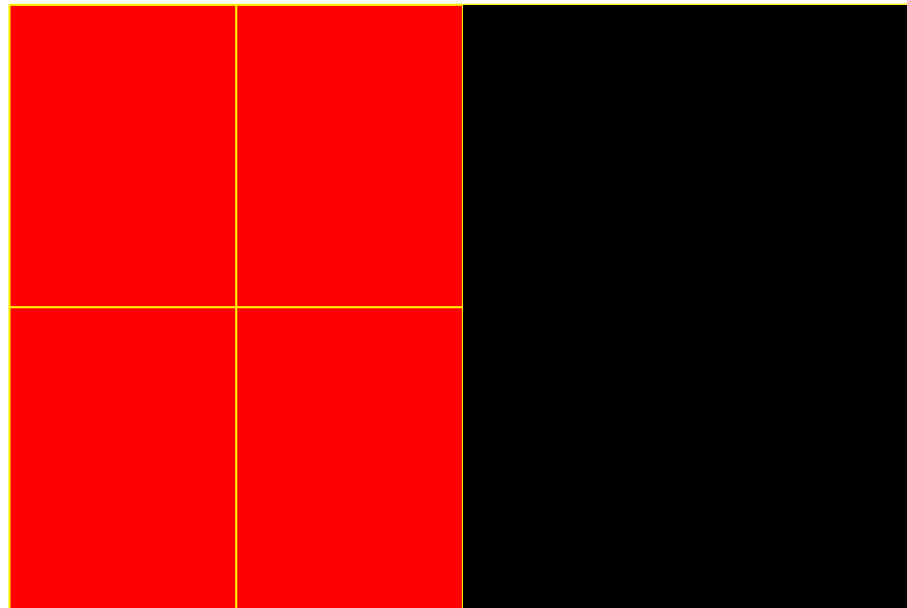
■ cannot contain a solution

Basic branch-and-bound algorithm



- may contain a solution
- cannot contain a solution

Basic branch-and-bound algorithm



■ may contain a solution

■ cannot contain a solution

Inter-box parallelism

$L_0 = \{ \text{the box } [\mathbf{z}] \text{ to be searched for singularities } \}$

for $k = 0, 1, \dots$, until $L_k = \emptyset$

$L_{k+1} = \emptyset$

for all boxes $[\mathbf{z}]$ in L_k

if $0 \in F_i[\mathbf{z}]$ for each $i = 1, \dots, n$

and none of the more sophisticated tests allows discarding $[\mathbf{z}]$

then

if $[\mathbf{z}]$ is small enough

then

add $[\mathbf{z}]$ to the list of the possible solutions

else

split $[\mathbf{z}]$ into two subboxes $[\mathbf{z}^{(1)}]$, $[\mathbf{z}^{(2)}]$

add $[\mathbf{z}^{(1)}]$ and $[\mathbf{z}^{(2)}]$ to L_{k+1}

Inter-box parallelism

$L_0 = \{ \text{the box } [\mathbf{z}] \text{ to be searched for singularities } \}$

for $k = 0, 1, \dots$, until $L_k = \emptyset$

$L_{k+1} = \emptyset$

for all boxes $[\mathbf{z}]$ in L_k \leftarrow parallelize this loop with **OpenMP**

if $0 \in F_i[\mathbf{z}]$ for each $i = 1, \dots, n$

and none of the more sophisticated tests allows discarding $[\mathbf{z}]$

then

if $[\mathbf{z}]$ is small enough

then

add $[\mathbf{z}]$ to the list of the possible solutions

else

split $[\mathbf{z}]$ into two subboxes $[\mathbf{z}^{(1)}]$, $[\mathbf{z}^{(2)}]$

add $[\mathbf{z}^{(1)}]$ and $[\mathbf{z}^{(2)}]$ to L_{k+1}

Inter-box parallelism

$L_0 = \{ \text{the box } [\mathbf{z}] \text{ to be searched for singularities} \}$

for $k = 0, 1, \dots$, until $L_k = \emptyset$

$L_{k+1} = \emptyset$

for all boxes $[\mathbf{z}]$ in L_k \leftarrow parallelize this loop with **OpenMP**

if $0 \in F_i[\mathbf{z}]$ for each $i = 1, \dots, n$

and none of the more sophisticated tests allows discarding $[\mathbf{z}]$

then

if $[\mathbf{z}]$ is small enough

then

add $[\mathbf{z}]$ to the list of the possible solutions

else

split $[\mathbf{z}]$ into two subboxes $[\mathbf{z}^{(1)}], [\mathbf{z}^{(2)}]$

add $[\mathbf{z}^{(1)}]$ and $[\mathbf{z}^{(2)}]$ to L_{k+1} \leftarrow synchronization

Inter-box parallelism

$L_0 = \{ \text{the box } [\mathbf{z}] \text{ to be searched for singularities } \}$

for $k = 0, 1, \dots$, until $L_k = \emptyset$

$L_{k+1} = \emptyset$

for all boxes $[\mathbf{z}]$ in L_k \leftarrow parallelize this loop with **OpenMP**

if $0 \in F_i[\mathbf{z}]$ **for each** $i = 1, \dots, n$

and none of the more sophisticated tests allows discarding $[\mathbf{z}]$

then

if $[\mathbf{z}]$ is small enough

then

add $[\mathbf{z}]$ to the list of the possible solutions

else

split $[\mathbf{z}]$ into two subboxes $[\mathbf{z}^{(1)}], [\mathbf{z}^{(2)}]$

add $[\mathbf{z}^{(1)}]$ and $[\mathbf{z}^{(2)}]$ to L_{k+1} \leftarrow synchronization

allows additional parallelization

Inter-box parallelism

$L_0 = \{ \text{the box } [\mathbf{z}] \text{ to be searched for singularities} \}$

for $k = 0, 1, \dots$, until $L_k = \emptyset$

$L_{k+1} = \emptyset$

for all boxes $[\mathbf{z}]$ in L_k \leftarrow parallelize this loop with **OpenMP**

if $0 \in F_i[\mathbf{z}]$ for each $i = 1, \dots, n$

and none of the **more sophisticated tests** allows discarding $[\mathbf{z}]$

then

if $[\mathbf{z}]$ is small enough

then

add $[\mathbf{z}]$ to the list of the possible solutions

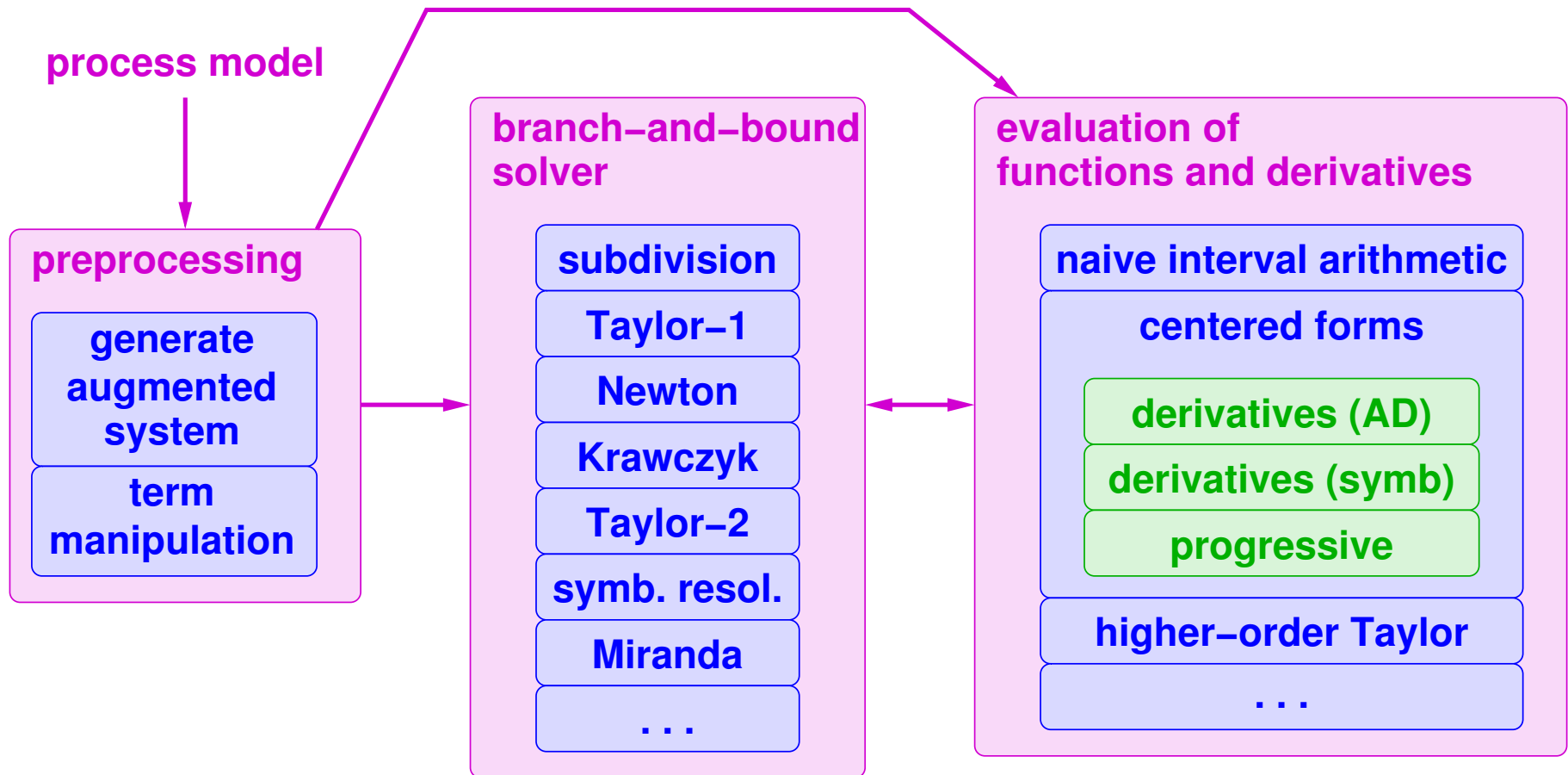
else

split $[\mathbf{z}]$ into two subboxes $[\mathbf{z}^{(1)}], [\mathbf{z}^{(2)}]$

add $[\mathbf{z}^{(1)}]$ and $[\mathbf{z}^{(2)}]$ to L_{k+1} \leftarrow synchronization

allows additional parallelization

Structure of the solver



Numerical results

- program was written in C++
- compiled with version 5.5 of Sun C++ compiler
- compiler provides support for interval arithmetic together with OpenMP constructs
- numerical experiments were performed on a Sun Fire 6800 server with 24 processors (900 MHz) and 24 GB of main memory running Solaris 9

Timings for solving a system with $n = 29$ unknowns

Program version	#Procs	Time	Speedup
Serial	1	17:32	
Serial, compiled with <code>-xopenmp</code>	1	21:18	
Parallel	1	22:55	1.00
Parallel, <code>schedule(static, 1)</code>	2	13:52	1.65
Parallel, <code>schedule(static, 1)</code>	3	11:20	2.02
Parallel, <code>schedule(static, 1)</code>	4	10:46	2.13
Parallel, standard scheduling	4	11:41	1.96
Parallel, <code>schedule(dynamic, 1)</code>	4	12:00	1.91

Timings for solving a system with $n = 29$ unknowns

Program version	#Procs	Time	Speedup
Serial	1	17:32	
Serial, compiled with <code>-xopenmp</code>	1	21:18	
Parallel	1	22:55	1.00
Parallel, <code>schedule(static, 1)</code>	2	13:52	1.65
Parallel, <code>schedule(static, 1)</code>	3	11:20	2.02
Parallel, <code>schedule(static, 1)</code>	4	10:46	2.13
Parallel, standard scheduling	4	11:41	1.96
Parallel, <code>schedule(dynamic, 1)</code>	4	12:00	1.91

speedup seems to be limited to the factor of 2 !!!

Summary

- Branch-and-bound nonlinear solver
- Intervals (guaranteed answers) + OpenMP (performance)
- Simple parallelization scheme
- Should work according to theory
(minor improvements expected from better scheduling mechanisms: taskq, ...)
- Checking out satisfactory explanations for limited speedup in practice

⇒ **lab session of EWOMP 2003**

Intervals and OpenMP:

Towards an Efficient Parallel Result-Verifying Nonlinear Solver

EWOMP 2003
Aachen University, Germany
September 23, 2003

Thomas Beelitz



Scientific Computing
University of Wuppertal
beelitz@math.uni-wuppertal.de

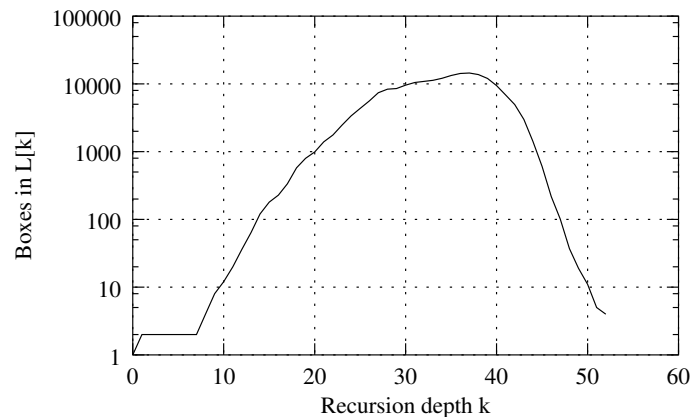


Possible limiting factors

- **Granularity too small?**

total number of boxes: 195 381 \Rightarrow average of 5 milliseconds per box

- **Load imbalance?**



90.1% of boxes are handled at recursion levels 25, ..., 42 (list length $> 4\,000$)

- experiments with simpler programs indicate that constructor calls for creating thread-local objects might be responsible