An Efficient Local Optimizer-Tracking Solver for Differential-Algebraic Equations with Optimization Criteria

Alexander Fleming[1] Jens Deussen Uwe Naumann

Software and Tools for Computational Engineering[2]
RWTH Aachen, Germany

8th International Conference on Algorithmic Differentiation (AD2024)

[1] fleming@stce.rwth-aachen.de
[2] https://www.stce.rwth-aachen.de/

Contents

Motivation

Global Optimization with Interval Arithmetic
- Interval Arithmetic
- An Optimization Algorithm

Integrating a DAEO
- Event Detection
- Local Optimizer Tracking

Performance Testing

Context

Conclusion and Outlook

References

Motivation

Global Optimization with Interval Arithmetic
- Interval Arithmetic
- An Optimization Algorithm

Integrating a DAEO
- Event Detection
- Local Optimizer Tracking

Performance Testing

Context

Conclusion and Outlook

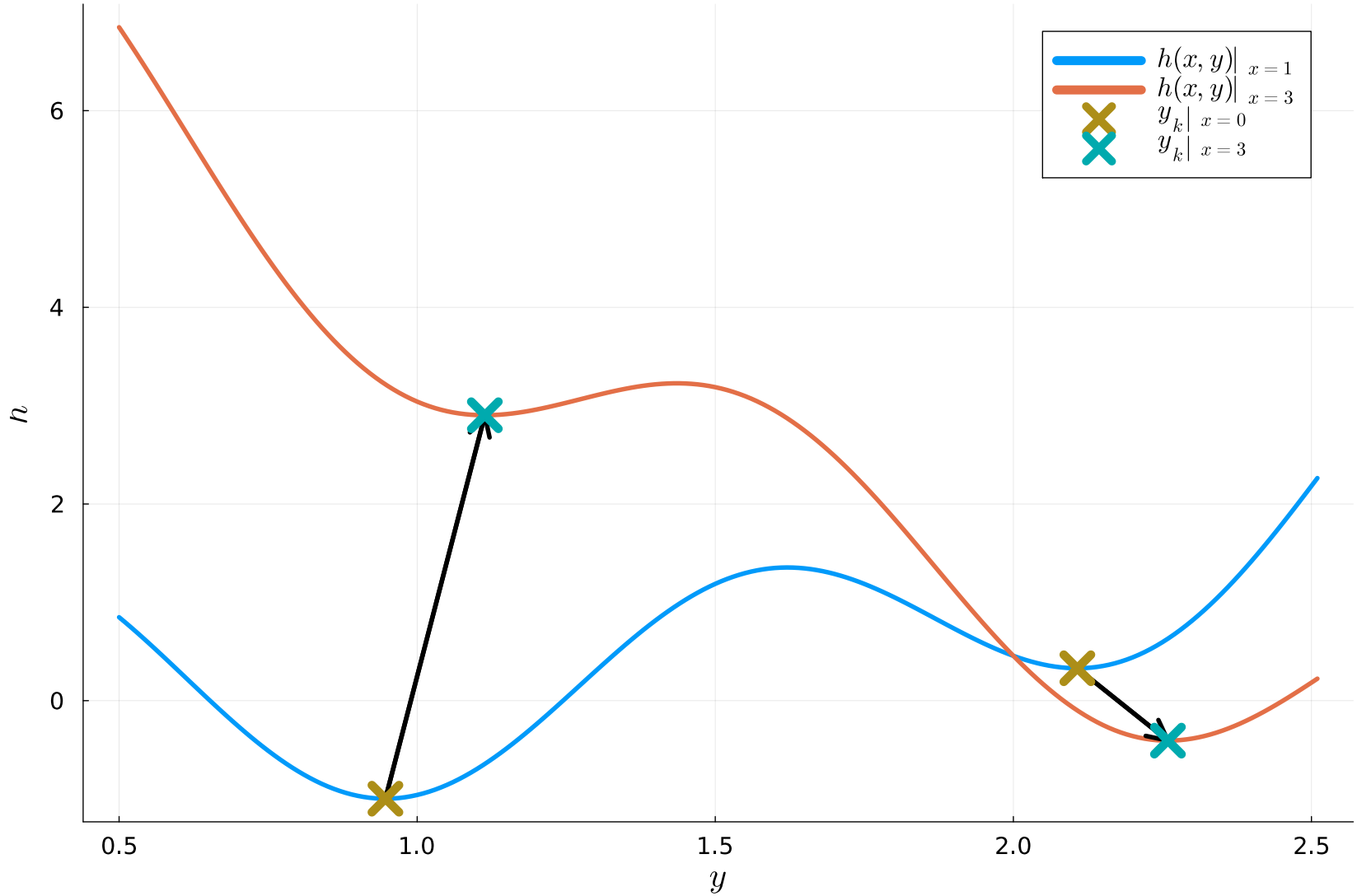
References

Consider the following differential equation with an embedded optimization problem (DAEO):

$$\begin{aligned} x(0) &= 1 \\ \dot{x}(t) &= y^\star(t) \\ \left\{y^k(t)\right\} &= \arg\min_y h(x,y) \\ h(x,y) &= (x-y)^2 + \sin 5y \end{aligned} \tag{1}$$

How might we approach solving this equation numerically?

Motivation



The circular dependence of $x(t)$ on the solution to the minimization problem $y(t)$ poses an issue. We could try separating the two problems, and solving the optimization problem between each time step:

$$\begin{aligned} y^{k\star} &= \arg\min_{y} h(x^k, y) \\ x^{k+1} &= x^k + \frac{\Delta t}{2}\left(f(x^k, y^{k\star}) + f(x^{k+1}, y^{k\star})\right) \end{aligned} \tag{2}$$

This approach has two problems:

- Global optimization is, especially for larger problems, is very expensive.
- What happens when the global optimizer $y^\star$ changes between two times t^k, t^{k+1}?

- $\{y_i\}$ is the set of minimizers of $h(x, y)$
- $y^\star$ is the global minimizer of h
- $\nabla_y f$ is the gradient of f w.r.t. y
- $\partial_y f$ and $d_y f$ are partial and total derivatives, respectively.
- Δt is the time step size.
- x^k is the value of x at time step t^k
- y_i^k is the value of y_i at time step t^k

Without some way to detect a global optimizer change during a time step, any integration method we choose for $x(t)$ will have its order of convergence *reduced to* 1[3]!

Outline

Motivation

Global Optimization with Interval Arithmetic
- Interval Arithmetic
- An Optimization Algorithm

Integrating a DAEO
- Event Detection
- Local Optimizer Tracking

Performance Testing

Context

Conclusion and Outlook

References

In order to solve this problem, we'll first need a global optimizer that can find *all* local optimizers of $h(x, y)$.

Fundamental Theorem of Interval Arithmetic[1]

An interval evaluation of a function $[y] = \left[\underline{y}, \overline{y}\right] = f([\underline{x}, \overline{x}]) = f([x])$ must yield an interval that contains all pointwise evaluations $f(x) \, \forall \, x \in [\underline{x}, \overline{x}]$.

In the world of floating-point numbers, **IEEE 1788-2015** specifies basic interval arithmetic (IA). The Boost interval library[4] follows this standard.

If we had access to *interval gradients*, this process would be much easier...

Consider some function f, and its tangent- and adjoint-mode evaluations:

$$y = f(x) : \mathbb{R}^n \mapsto \mathbb{R}^m$$
$$y^{(1)} = \nabla_x f(x) \cdot x^{(1)}$$
$$x_{(1)} = y_{(1)} \cdot \nabla_x f(x)$$

With a suitable AD tool (perhaps `dco/c++`[2]), we could substitute intervals directly into tangent- and adjoint-mode AD.

An interval evaluation of the tangent of f must yield a correct interval for $y^{(1)}$:

$$[y^{(1)}] = \nabla_x f([x]) \cdot x^{(1)}$$

An interval evaluation of the adjoint of f must yield a correct interval for $x_{(1)}$:

$$[x_{(1)}] = y_{(1)} \cdot \nabla_x f([x])$$

Testing if an interval $[\underline{x}, \bar{x}]$ contains a critical point is quite straightforward.

First-Order Optimality with Intervals

The interval $[x]$ contains a critical point if $0 \in [\nabla_x f_i]$.

Second-order optimality is a bit more difficult. We can compute the interval Hessian $[H_x f]$ via AD exactly as we would compute the regular Hessian.

Second-Order Optimality with Intervals

The interval $[x]$ contains exactly one minimum of f if, for every matrix $A \in [H_x f([x])]$, A is positive definite.

Branch-and-Act with Interval Arithmetic

Process the list of intervals to search for optimizers $\mathfrak{S} = \{[y]\}$ according to the following rules:

1. Take the first item $[y]$ from $\mathfrak{S}$.
2. **Gradient Test:** Evaluate the interval gradient $\partial_y h(x, [y])$. If the result interval does not contain 0, $[y]$ contains no optimizers and can be discarded.
3. **Hessian Test:** Test the interval Hessian $\partial_y^2 h(x, [y])$ for positive definite-ness.
   - 3.1 If the interval Hessian is negative definite, h is concave down over the interval $[y]$, and $[y]$ can be discarded.
   - 3.2 If the interval Hessian is positive definite, h is concave up over the interval $[y]$, and $[y]$ can be narrowed by any appropriate local optimization method.
4. **Branch:** If the interval Hessian is neither positive- nor negative definite, decompose the interval $[y]$ and append the results to $\mathfrak{S}$.
5. Repeat for all remaining items in $\mathfrak{S}$.

Outline

Motivation

Global Optimization with Interval Arithmetic
- Interval Arithmetic
- An Optimization Algorithm

Integrating a DAEO
- Event Detection
- Local Optimizer Tracking

Performance Testing

Context

Conclusion and Outlook

References

The reason that the initial integration strategy fails is because of "events".

An event occurs at some time $\tau \geq t_0$, when there is a $y^j(\tau) \neq y^\star(\tau) \in \{y_k(\tau)\}$ such that

$$h(x(\tau), y^\star(\tau)) = h(x(\tau), y^j(\tau)) \tag{3}$$

Two scenarios can lead to an event:

- The global optimizer shifts from $y^j(t)$ to $y^j(t)$ as the system evolves.
- The global optimizer is both $y^j(\tau)$ and $y^j(\tau)$ *only* at $t = \tau$.

We can test if a time step from t_k to t_{k+1} contained an event between any two local optimizers y^i and y^j by testing for a sign change in the *event function*

$$H(x^k, y^i, y^j) = h(x^k, y^i) - h(x^k, y^j) \tag{4}$$

In time steps where this situation occurs, a root-finding procedure can find the time τ where $H(x^k, y^i, y^j) = 0$, and the time stepping procedure for (1) can step from t_k to τ to t_{k+1}.

Local Optimizer Drift Estimate

$$
\begin{aligned}
0 &= \partial_y h(x, y_i) \\
0 &= d_x \partial_y h(x, y_i) \\
0 &= \partial^2_{yy} h(x, y_i) \cdot \partial_x y_i + \partial^2_{xy} h(x, y_i) \\
\partial_x y_i &= -\left(\partial^2_{yy} h(x, y_i)\right)^{-1} \partial^2_{xy} h(x, y_i) \\
\partial_t y_i &= -\left(\partial^2_{yy} h(x, y_i)\right)^{-1} \partial^2_{xy} h(x, y_i) f(x, y_i)
\end{aligned}
\tag{5}
$$

An Integrator for DAEOs

Time stepping:

$$
\begin{aligned}
0 &= x^k - x^{k+1} + \frac{\Delta t}{2}\left(f(x^k, y^{\star,n}) + f(x^{k+1}, y^{\star,k+1})\right) \\
0 &= \partial_{y_i^k} h\left(x^{k+1}, y_i^{k+1}\right)
\end{aligned}
\tag{6}
$$

In practice, a suitable guess for x^{k+1} and y_i^{k+1} is

$$
\begin{aligned}
x^{k+1} &= x^k + f(x^k, y^{\star,k})\Delta t \\
y_i^{k+1} &= y_i^k + \partial_t y_i^k \Delta t
\end{aligned}
$$

Outline

Motivation

Global Optimization with Interval Arithmetic
- Interval Arithmetic
- An Optimization Algorithm

Integrating a DAEO
- Event Detection
- Local Optimizer Tracking

Performance Testing

Context

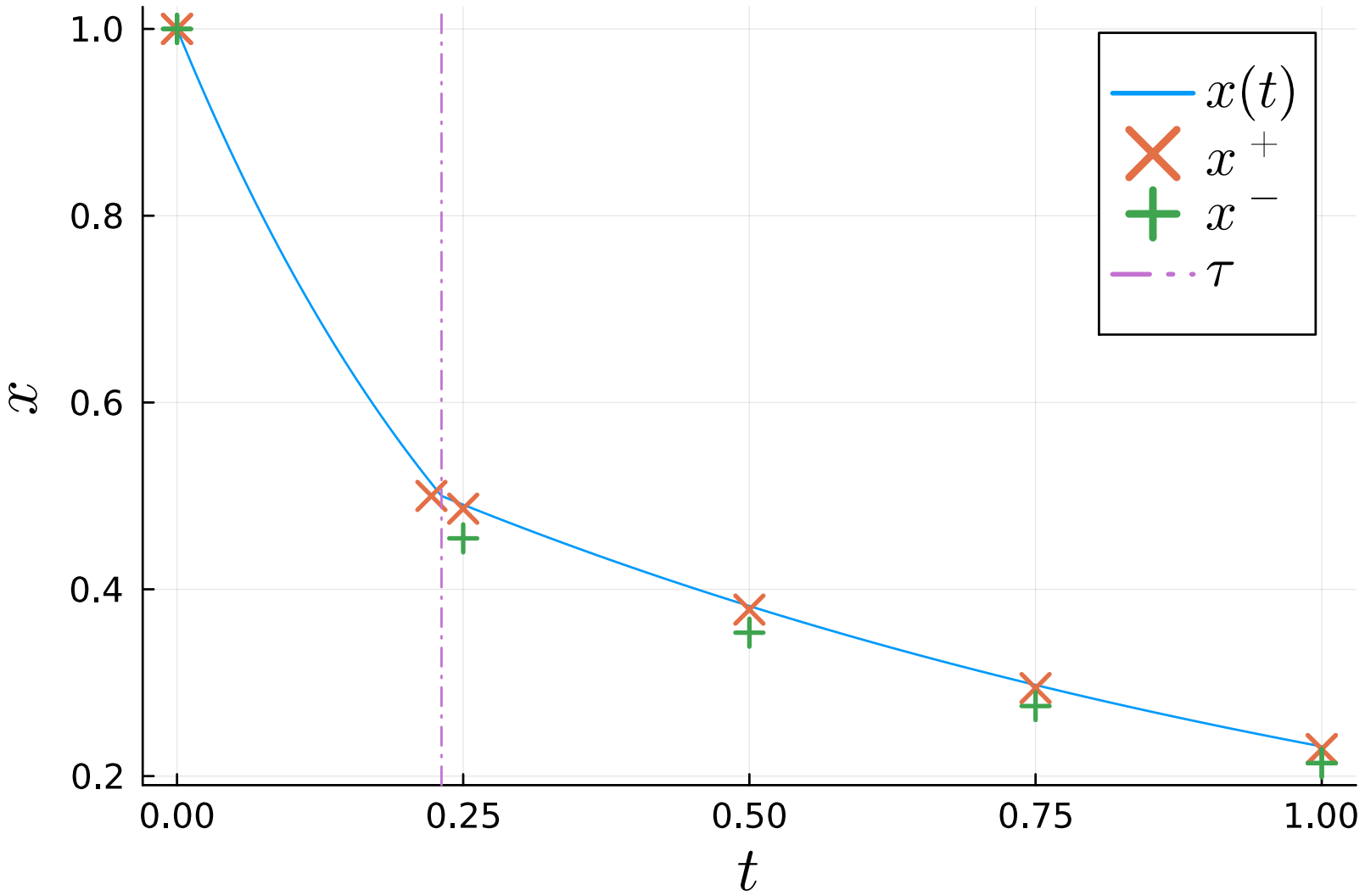
Conclusion and Outlook

References

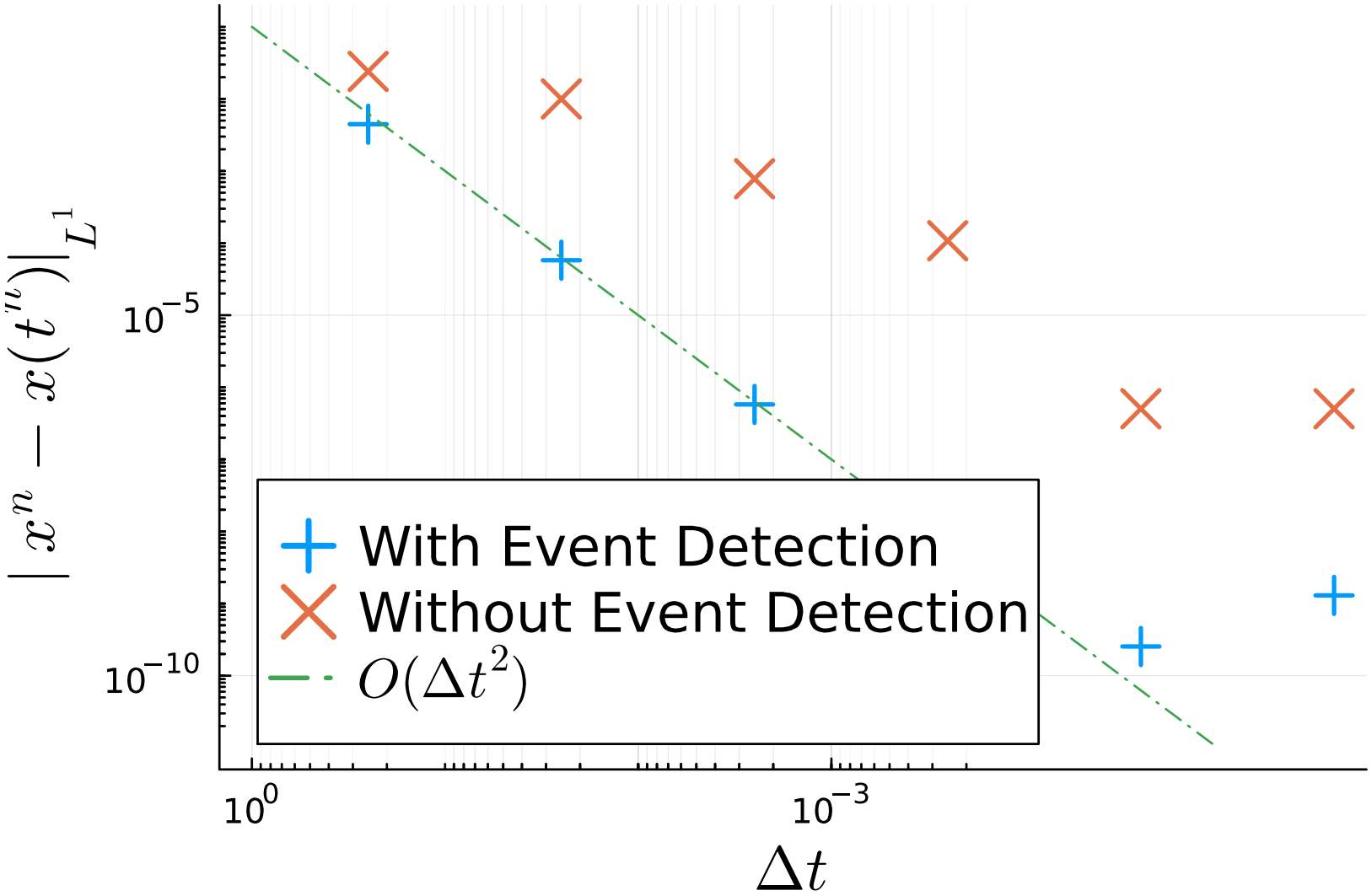
A Simpler Example

$$
\begin{aligned}
x(0) &= 1 \\
\dot{x}(t) &= -(2 + y^\star(t))x \\
\left\{y^k(t)\right\} &= \arg\min_y h(x, y) \\
h(x, y) &= (1 - y^2)^2 - (x - \frac{1}{2}) \sin\left(\frac{\pi y}{2}\right)
\end{aligned}
$$

A Simpler Example



A Simpler Example

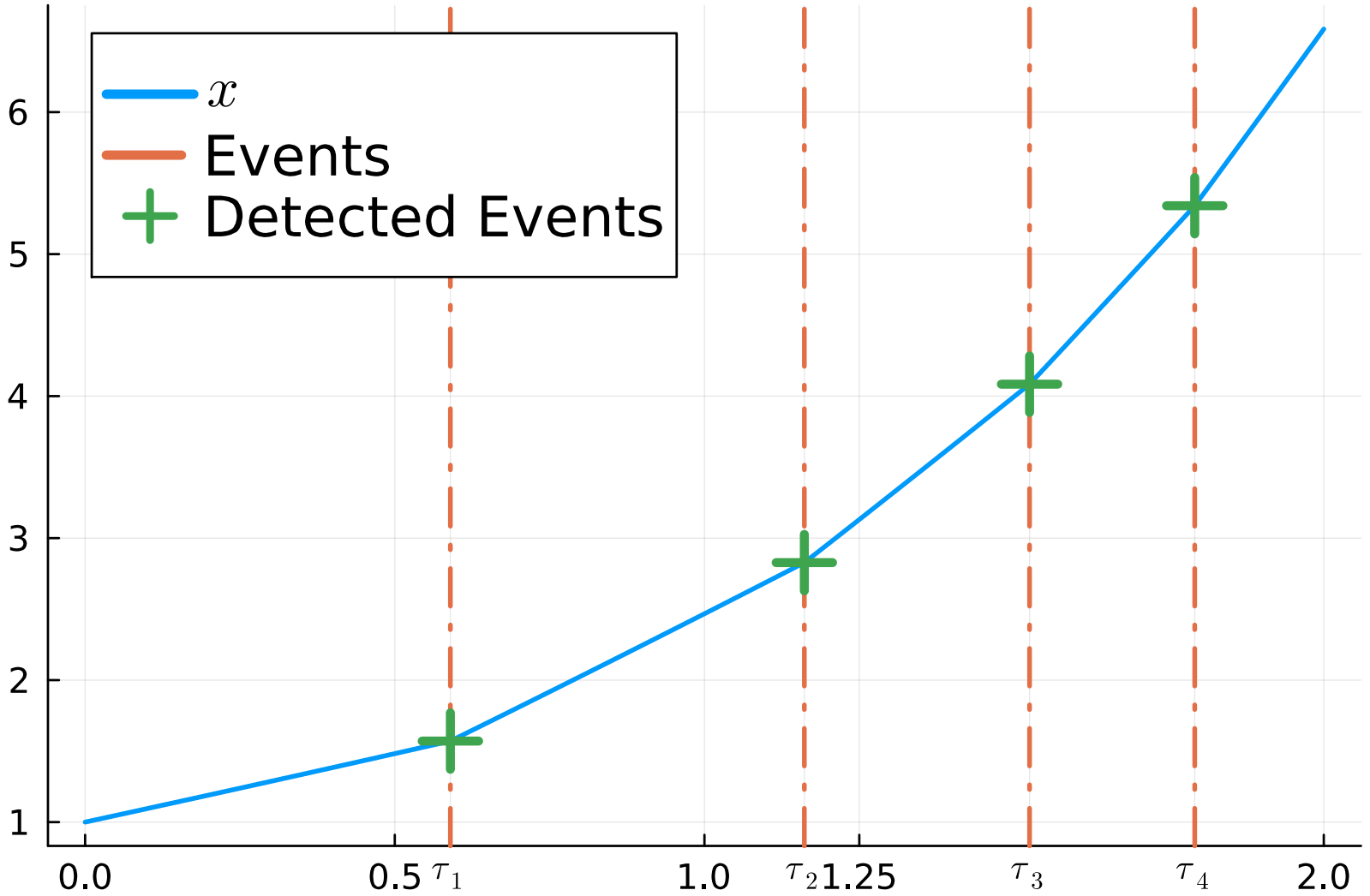


A Simpler Example

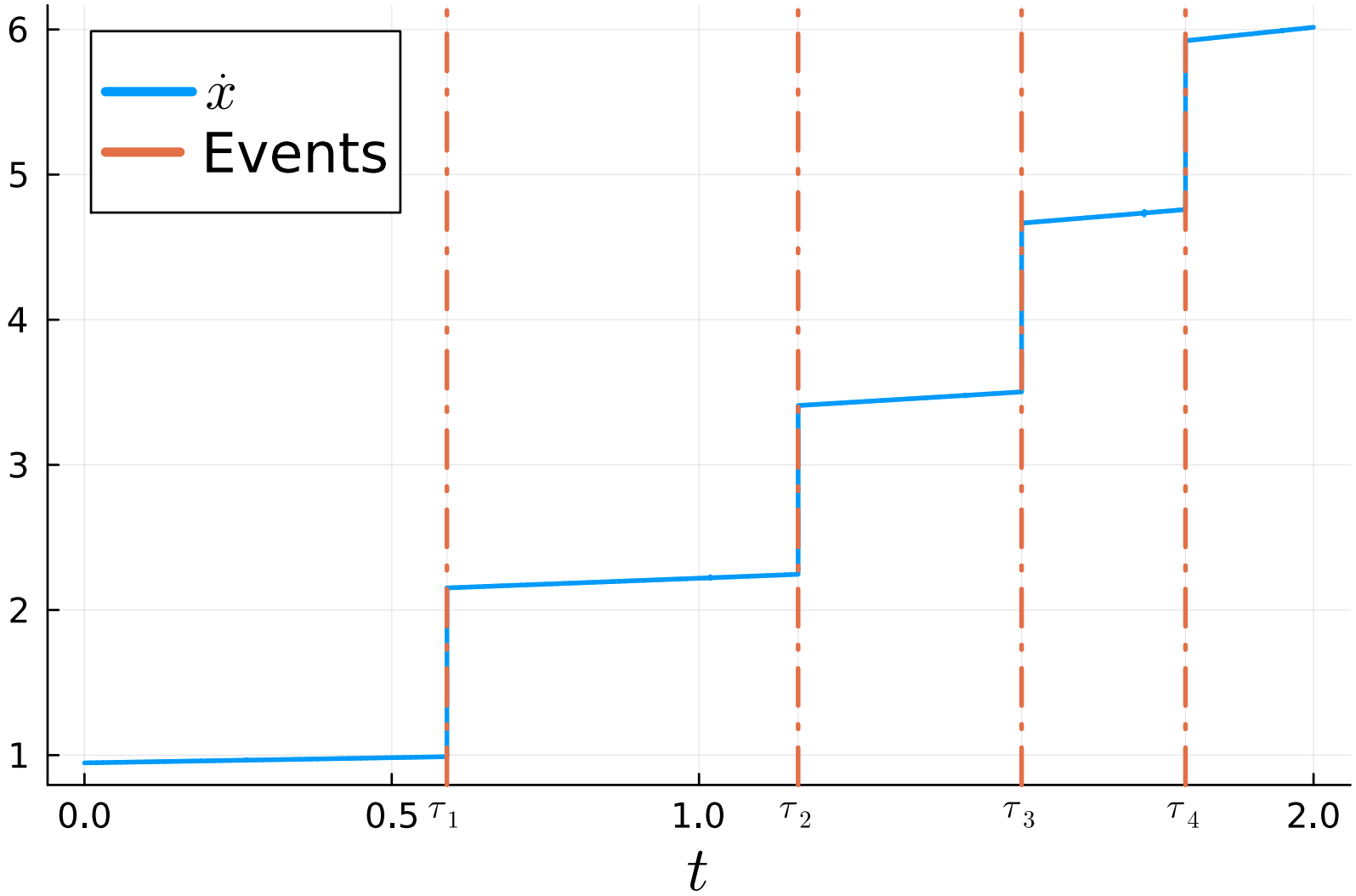
Δt	No Event Detection	Event Detection	Always Run Global Optimizer
2.50e-01	3 ms	15 ms	23 ms
2.50e-02	14 ms	20 ms	83 ms
2.50e-03	122 ms	126 ms	793 ms
2.50e-04	1125 ms	1051 ms	7593 ms
2.50e-05	11354 ms	10444 ms	74874 ms
2.50e-06	107298 ms	109945 ms	744696 ms

$$
\begin{aligned}
x(0) &= 1 \\
\dot{x}(t) &= y^\star(t) \\
\left\{y^k(t)\right\} &= \arg\min_y h(x,y) \\
h(x,y) &= (x-y)^2 + \sin 5y
\end{aligned}
$$

The Original Example



The Original Example



Outline

Motivation

Global Optimization with Interval Arithmetic
- Interval Arithmetic
- An Optimization Algorithm

Integrating a DAEO
- Event Detection
- Local Optimizer Tracking

Performance Testing

Context

Conclusion and Outlook

References

The technique developed here could be applied to process engineering problems:

- Real-time solutions to optimal-control problems [5].
- More accurate simulation of flash processes [7]

Other multi-scale simulation problems could see significant performance improvments:

- Biorefinery simulation [6]

Outline

Motivation

Global Optimization with Interval Arithmetic
- Interval Arithmetic
- An Optimization Algorithm

Integrating a DAEO
- Event Detection
- Local Optimizer Tracking

Performance Testing

Context

Conclusion and Outlook

References

- We have successfully restored the 2nd-order convergence of an implicit Euler scheme.
- We avoid expensive global optimizer calls during the integration of a DAEO.

For all the juicy details...

`STCE-at-RWTH/daeo-tracking-solver`

The current state of the project leads to some natural next steps:

- Can we quantify the quality of an estimate for the local optimizer drift?
- How should we compute derivatives w.r.t. y near events?
- Is it possible to include second-order information in the time stepping for $\{y_i\}$?
- What are the best choices for hyper-parameters (Event detection threshold, estimates for Lipshitz constants)?

The global optimizer component could also be very much improved.

- How should we perform global optimization under constraints?
- Is there a better way to test interval matrices for positive-definiteness?

Outline

Motivation

Global Optimization with Interval Arithmetic
- Interval Arithmetic
- An Optimization Algorithm

Integrating a DAEO
- Event Detection
- Local Optimizer Tracking

Performance Testing

Context

Conclusion and Outlook

References

References

[1] T. Hickey, Q. Ju, and M. H. Van Emden. "Interval Arithmetic: From Principles to Implementation". In: *Journal of the ACM* 48.5 (Sept. 2001), pp. 1038–1068. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/502102.502106.

[2] K. Leppkes, J. Lotz, and U. Naumann. *Derivative Code by Overloading in C++ (Dco/C++): Introduction and Summary of Features*. AIB-2016-08. RWTH Aachen University, Sept. 2016.

[3] R. Mannshardt. "One-Step Methods of Any Order for Ordinary Differential Equations with Discontinuous Right-Hand Sides". In: *Numerische Mathematik* 31.2 (June 1, 1978), pp. 131–152. ISSN: 0945-3245. DOI: 10.1007/BF01397472.

[4] G. Melquiond, S. Pion, and H. Brönnimann. *Boost Interval Library*. Version 1.85.0. June 24, 2022.

[5] T. Ploch et al. "Direct Single Shooting for Dynamic Optimization of Differential-Algebraic Equation Systems with Optimization Criteria Embedded". In: *Computers & Chemical Engineering* 159 (Mar. 1, 2022), p. 107643. ISSN: 0098-1354. DOI: 10.1016/j.compchemeng.2021.107643.

[6] T. Ploch et al. "Multiscale Dynamic Modeling and Simulation of a Biorefinery". In: *Biotechnology and Bioengineering* 116.10 (2019), pp. 2561–2574. ISSN: 1097-0290. DOI: 10.1002/bit.27099.

[7] T.. Ritschel et al. "An Algorithm for Gradient-Based Dynamic Optimization of UV Flash Processes". In: *Computers & Chemical Engineering* 114 (June 2018), pp. 281–295. ISSN: 00981354. DOI: 10.1016/j.compchemeng.2017.10.007.