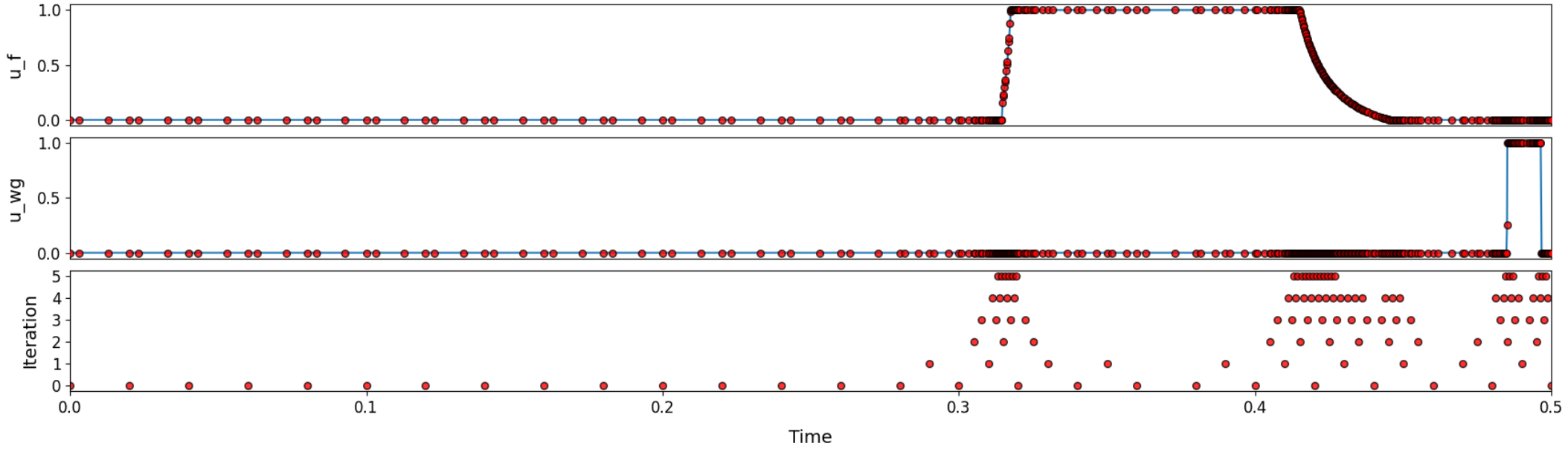
Hochschule Bielefeld

University of Applied Sciences and Arts



Extending Direct Collocation-Based Dynamic Optimization with Adaptive Mesh Refinement

Linus Langenkamp, Bernhard Bachmann

03.02.2025

HS'BI Hochschule Bielefeld University of Applied Sciences and Arts

TABLE OF CONTENTS

INTRODUCTION TO DYNAMIC OPTIMIZATION

- optimization of dynamic systems $\rightarrow$ optimal control
- applications
  - aerospace engineering and trajectory optimization
  - chemical and biological processes
  - robotics
  - economics
- collocation
  - states are approximated by piecewise polynomials
  - reduce continuous problem to NLP
  - often perform mesh refinement

DRAWBACKS OF THE CURRENT IMPLEMENTATION

- dynamic optimization in OpenModelica has drawbacks

→ Python / C++ re-implementation with more advanced features

	RadauIIA	Jacobian	Hessian	parameter support	mesh refinement	solver
Current	1 or 3 steps	symbolic	numeric	no	no	Ipopt
New	1 - 70 steps	symbolic	symbolic	yes	yes	Ipopt

Definition (General Dynamic Optimization Problem (GDOP))

$$
\begin{aligned}
\min_{\boldsymbol{u}(t),\boldsymbol{p}} \; & M(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{p},t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t)\,\mathrm{d}t \\
\text{s.t.} \\
\dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t)\ \forall t \in [t_0,t_f] \\
\boldsymbol{x}(t_0) &= \boldsymbol{x}_0 \\
\boldsymbol{g}^L &\leq \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t) \leq \boldsymbol{g}^U\ \forall t \in [t_0,t_f] \\
\boldsymbol{r}^L &\leq \boldsymbol{r}(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{p},t_f) \leq \boldsymbol{r}^U \\
\boldsymbol{a}^L &\leq \boldsymbol{a}(\boldsymbol{p}) \leq \boldsymbol{a}^U
\end{aligned}
$$

Definition (Collocation)

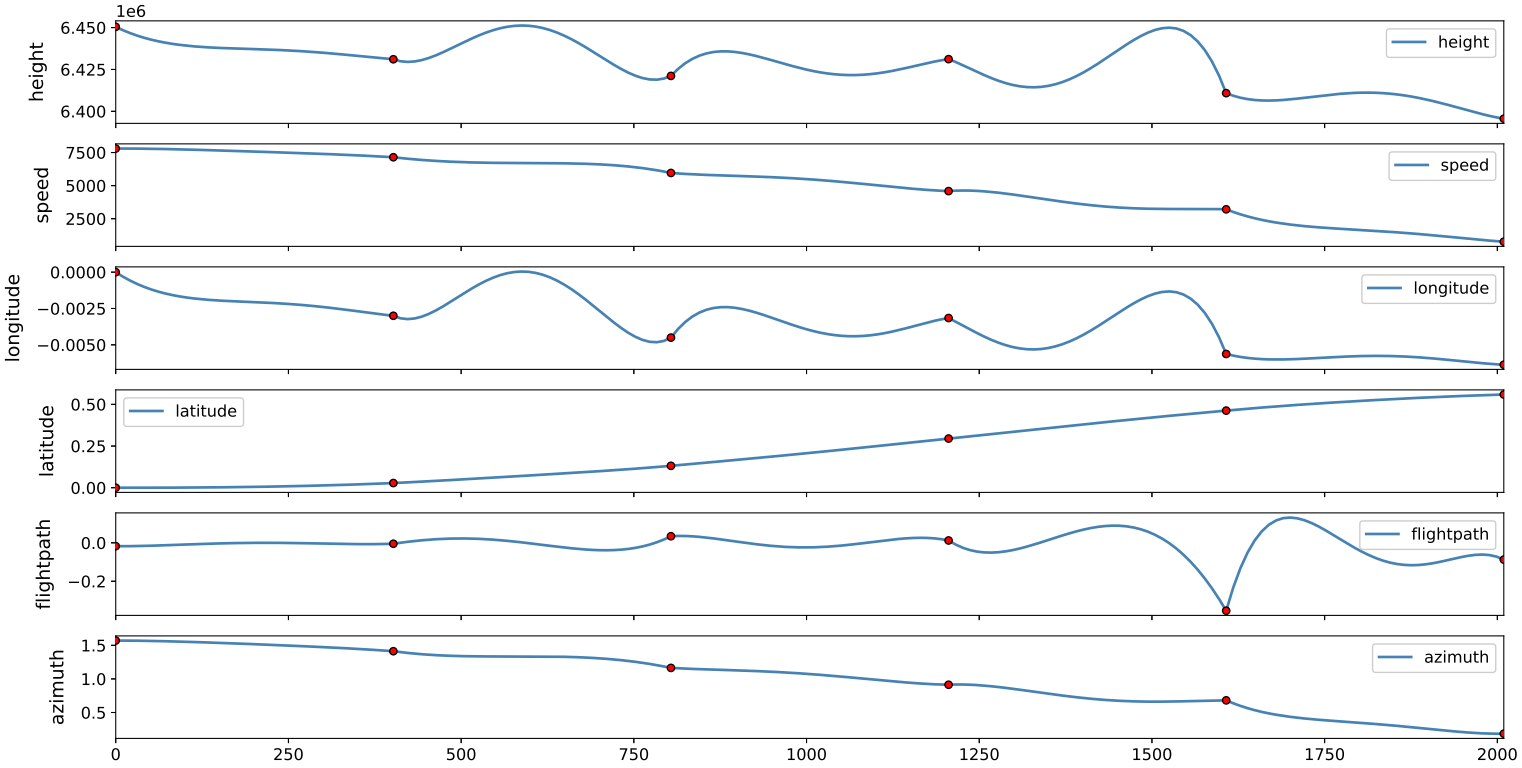
Let $c_1, \ldots, c_m$ be distinct real numbers (usually $0 \leq c_j \leq 1$). The *collocation polynomial* $\boldsymbol{q}(t)$ is a polynomial of degree m satisfying

$$\begin{aligned} \boldsymbol{q}(t_0) &= \boldsymbol{x}_0 \\ \dot{\boldsymbol{q}}(t_0 + c_j h) &= \boldsymbol{f}(t_0 + c_j h, \boldsymbol{q}(t_0 + c_j h)), \ j = 1, \ldots, m, \end{aligned}$$

and the numerical solution of the *collocation method* is defined by $\boldsymbol{x}_1 = \boldsymbol{q}(t_0 + h)$.

- equivalent to a m-stage Runge-Kutta method
- collocation approaches are widely used in dynamic optimization frameworks

COLLOCATION



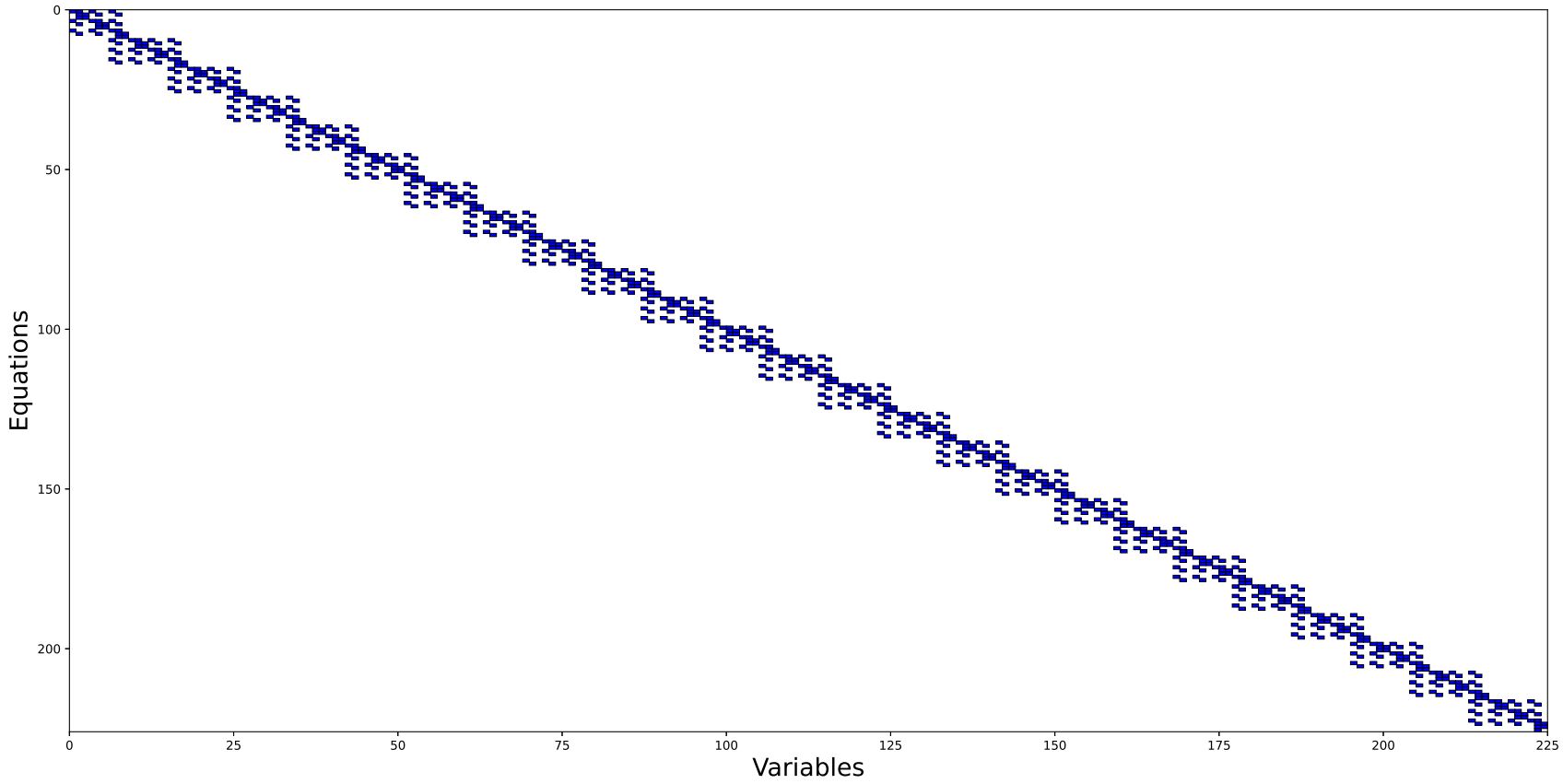
Definition (discretized General Dynamic Optimization Problem (dGDOP))

$$\min_{\boldsymbol{x}_{i,j},\boldsymbol{u}_{i,j},\boldsymbol{p}} \; M(\boldsymbol{x}_{n,m},\boldsymbol{u}_{n,m},\boldsymbol{p},t_{n,m}) + \sum_{i=0}^{n} \Delta t_i \sum_{j=1}^{m} b_j L(\boldsymbol{x}_{i,j},\boldsymbol{u}_{i,j},\boldsymbol{p},t_{i,j})$$

s.t.

$$\begin{aligned}
\mathbf{0} &= \sum_{k=1}^{m} \tilde{a}_{jk}(\boldsymbol{x}_{0,k} - \boldsymbol{x}_0) - \Delta t_0 \boldsymbol{f}(\boldsymbol{x}_{0,j},\boldsymbol{u}_{0,j},\boldsymbol{p},t_{0,j}), \;\; j = 1,\ldots,m \\
\mathbf{0} &= \sum_{k=1}^{m} \tilde{a}_{jk}(\boldsymbol{x}_{i,k} - \boldsymbol{x}_{i-1,m}) - \Delta t_i \boldsymbol{f}(\boldsymbol{x}_{i,j},\boldsymbol{u}_{i,j},\boldsymbol{p},t_{i,j}), \;\; i = 1,\ldots,n, j = 1,\ldots,m \\
\boldsymbol{g}^L &\leq \boldsymbol{g}(\boldsymbol{x}_{i,j},\boldsymbol{u}_{i,j},\boldsymbol{p},t_{i,j}) \leq \boldsymbol{g}^U, \;\; \forall i = 0,\ldots,n \;\; \forall j = 1,\ldots,m \\
\boldsymbol{r}^L &\leq \boldsymbol{r}(\boldsymbol{x}_{n,m},\boldsymbol{u}_{n,m},\boldsymbol{p},t_{n,m}) \leq \boldsymbol{r}^U \\
\boldsymbol{a}^L &\leq \boldsymbol{a}(\boldsymbol{p}) \leq \boldsymbol{a}^U
\end{aligned}$$

JACOBIAN OF THE NLP



WHY USE ADAPTIVE MESH REFINEMENT?

- optimization is very expensive for poor guesses on large meshes

→ only solve small NLPs with the initial guess

- optimal solution may contain discontinuities or switches, kinks, bends and steep sections

→ increase resolution to reduce errors

PROTOTYPE ITERATIVE MESH REFINEMENT ALGORITHM

Input: GDOP, guesses $\boldsymbol{x}_{ij}$, $\boldsymbol{u}_{ij}$, $\boldsymbol{p}$, mesh $\mathcal{M}_0$, number of collocation nodes m
Output: $\boldsymbol{x}_{ij}^*, \boldsymbol{u}_{ij}^*, \boldsymbol{p}^*$

$k \leftarrow 0$;
$\boldsymbol{x}_{ij}^*, \boldsymbol{u}_{ij}^*, \boldsymbol{p}^* \leftarrow$ Solve the dGDOP on $\mathcal{M}_0$ with m collocation nodes and guess $\boldsymbol{x}_{ij}$, $\boldsymbol{u}_{ij}$, $\boldsymbol{p}$;
while *stopping condition is not met* **do**
    $\mathcal{M}_{k+1} \leftarrow$ Update the mesh $\mathcal{M}_k$ with the optimal solution $\boldsymbol{x}_{ij}^*, \boldsymbol{u}_{ij}^*, \boldsymbol{p}^*$;
    $\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, \boldsymbol{p} \leftarrow$ Interpolate $\boldsymbol{x}_{ij}^*, \boldsymbol{u}_{ij}^*, \boldsymbol{p}^*$ on $\mathcal{M}_{k+1}$;
    $\boldsymbol{x}_{ij}^*, \boldsymbol{u}_{ij}^*, \boldsymbol{p}^* \leftarrow$ Solve the dGDOP on $\mathcal{M}_{k+1}$ with m collocation nodes and guess $\boldsymbol{x}_{ij}$, $\boldsymbol{u}_{ij}$, $\boldsymbol{p}$;
    $k \leftarrow k + 1$;
end
return $\boldsymbol{x}_{ij}^*, \boldsymbol{u}_{ij}^*, \boldsymbol{p}^*$;

NEW MESH REFINEMENT ALGORITHM

- aims to uniformize the control trajectories by successive bisection
- *on-interval* and *boundary* conditions
- advantages
    - very effective for a variety of benchmark problems
    - captures non-smooth / steep behavior
    - extremely fast
    - guaranteed termination for smooth problems (under suitable assumptions)
- limitations
    - does not incorporate error estimates
    - does not utilize spectral convergence

ON-INTERVAL CONDITION

- estimate the change for each control variable $u^{(d)}$ on each interval $[t_i, t_{i+1}]$
- $u^{(d)}$ is given by $m+1$ sample points at $t_i + \Delta t_i c_j, j = 0, \ldots, m$ with $c_0 = 0$
- construct an interpolating polynomial of $u^{(d)}$ on the nominal interval $[0, 1]$:

$$p(c_j) = u^{(d)}\left(t_i + \Delta t_i c_j\right) = u_{i,j}^{(d)}, j = 0, \ldots, m \text{ and } p \in P^m$$

- estimate the change by

$$\|\dot{p}\|_{L^q} = \left(\int_0^1 |\dot{p}(t)|^q \,\mathrm{d}t\right)^{\frac{1}{q}} \text{ and } \|\ddot{p}\|_{L^q} = \left(\int_0^1 |\ddot{p}(t)|^q \,\mathrm{d}t\right)^{\frac{1}{q}}$$

- choosing $q = 2 \rightarrow$ best middle ground, advantageous properties

ON-INTERVAL CONDITION

If for a given interval $[t_i, t_{i+1}]$ exists at least one control variable $u^{(d)}$, such that the interpolating polynomial $p(t)$ of $u^{(d)}$ on the nominal interval $[0, 1]$ satisfies

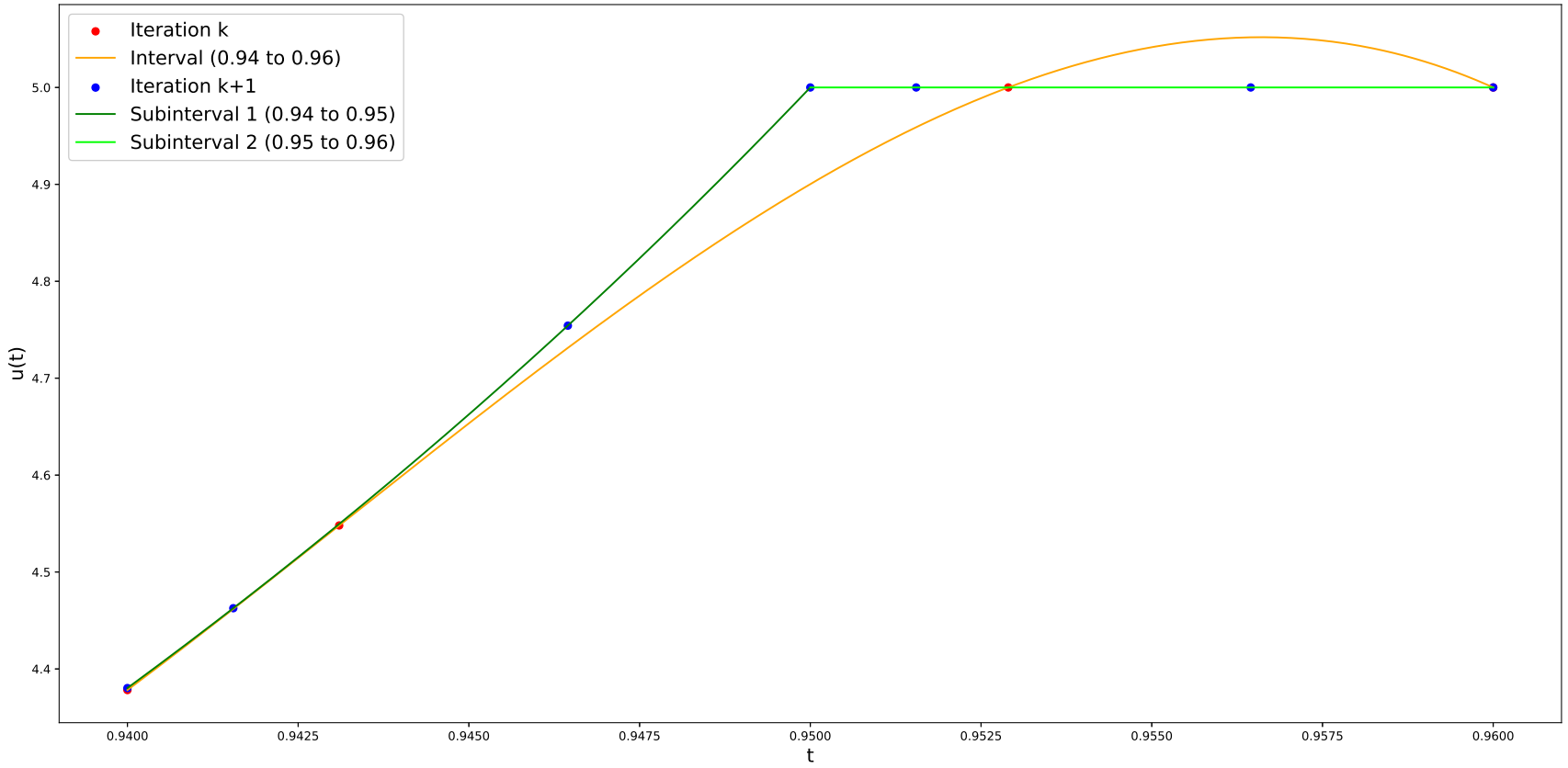
$$\|\dot{p}(t)\|_{L^2} \geq TOL_1 \text{ or } \|\ddot{p}(t)\|_{L^2} \geq TOL_2$$

for specified tolerances TOL_1, TOL_2, then the interval is bisected.

- allows a fast computation in $\mathcal{O}(m^2)$ via

$$\begin{aligned}\|\dot{p}(t)\|_{L^2} &= \left(\int_0^1 \dot{p}(t)^2 \,\mathrm{d}t\right)^{\frac{1}{2}} = \left(\sum_{k=1}^{m} b_k \dot{p}^2(c_k)\right)^{\frac{1}{2}} \\ &= \left(\sum_{k=1}^{m} b_k \left(\sum_{j=0}^{m} u_{i,j}^{(d)} \dot{l}_j(c_k)\right)^2\right)^{\frac{1}{2}} = \left(\sum_{k=1}^{m} b_k \left(\sum_{j=0}^{m} u_{i,j}^{(d)} D_{kj}^{(1)}\right)^2\right)^{\frac{1}{2}}\end{aligned}$$

BOUNDARY CONDITION

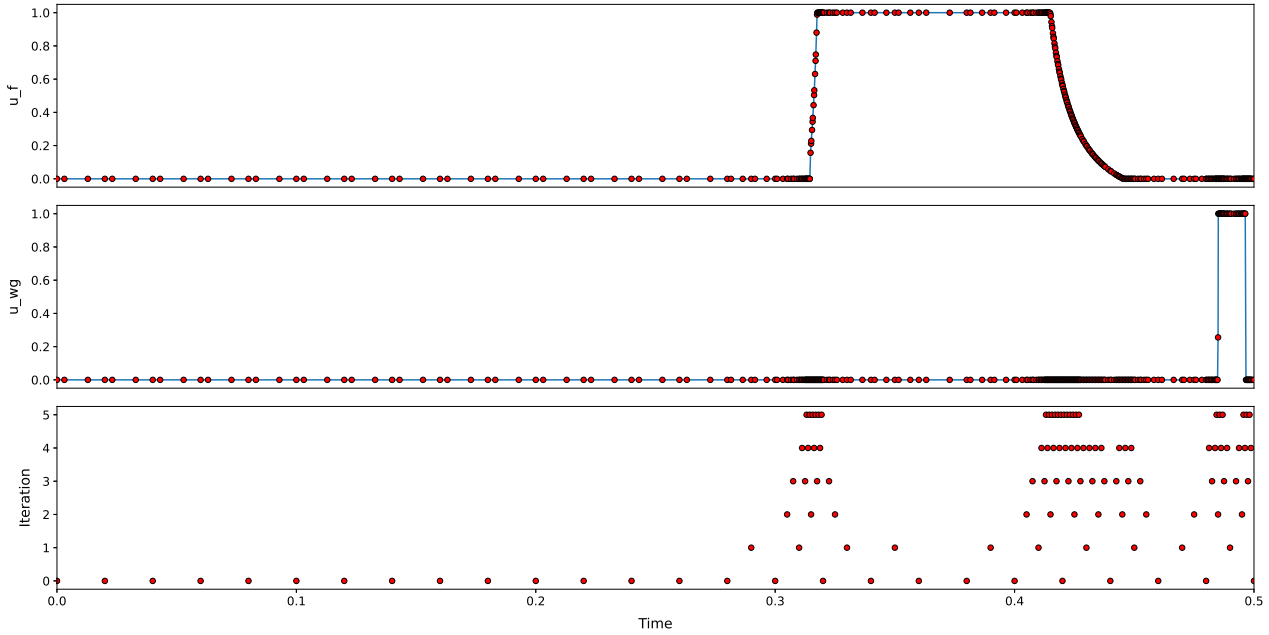


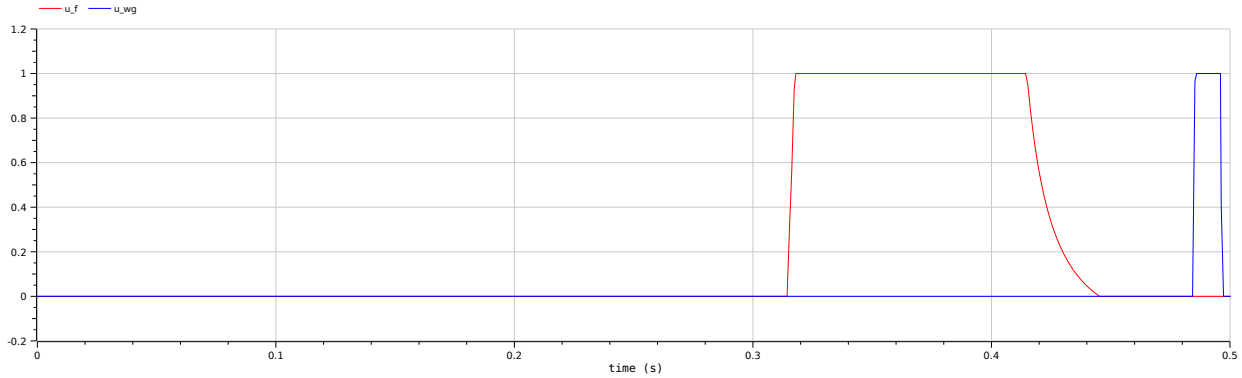
BOUNDARY CONDITION

Given two adjacent intervals with their respective interpolating polynomials p, q, then both intervals are bisected if

$$\frac{|\dot{p}(t_i) - \dot{q}(t_i)|}{1 + \min\{|\dot{p}(t_i)|, |\dot{q}(t_i)|\}} \geq CTOL_1 \text{ or } \frac{|\ddot{p}(t_i) - \ddot{q}(t_i)|}{1 + \min\{|\ddot{p}(t_i)|, |\ddot{q}(t_i)|\}} \geq CTOL_2$$

for the common boundary point t_i and specified corner tolerances $CTOL_1, CTOL_2$.


u_f
u_wg
Iteration
Time
0.0
0.1
0.2
0.3
0.4
0.5


u_f
u_wg
time (s)

Algorithm	k_{max}	n	m	$\lvert\mathcal{M}_{final}\rvert$	$\phi^{(opt)}$	t_{lpopt} in s	t_{eval} in s	Termination
Current	-	25	3	25	$1.11171326863 \cdot 10^{-3}$	0.191	0.096	optimal
New	0	25	3	25	$1.11171326863 \cdot 10^{-3}$	0.069	0.011	optimal
Current	-	100	3	100	$1.11155875712 \cdot 10^{-3}$	0.337	0.235	optimal
New	0	100	3	100	$1.11155875712 \cdot 10^{-3}$	0.146	0.036	optimal
Current	-	250	3	250	$1.11155972241 \cdot 10^{-3}$	0.873	0.686	optimal
New	0	250	3	250	$1.11155972241 \cdot 10^{-3}$	0.336	0.079	optimal
Current	-	1000	3	1000	$1.11155920322 \cdot 10^{-3}$	4.157	3.195	acceptable
New	0	1000	3	1000	$1.11155856386 \cdot 10^{-3}$	1.757	0.391	optimal
New	5	25	3	114	$1.11155856029 \cdot 10^{-3}$	0.189	0.041	optimal
New	5	25	5	116	$1.11155852132 \cdot 10^{-3}$	0.558	0.097	optimal
New	5	25	7	117	$1.11155856606 \cdot 10^{-3}$	0.859	0.110	optimal
New	5	100	3	292	$1.11155853676 \cdot 10^{-3}$	0.559	0.143	optimal
New	5	100	5	293	$1.11155853568 \cdot 10^{-3}$	1.326	0.242	optimal
New	5	100	7	299	$1.11155853848 \cdot 10^{-3}$	3.382	0.384	optimal

$\rightarrow$ test it yourself: https://github.com/linuslangenkamp/GDOPT

GOALS

- extension of the framework → integration in OpenModelica
- more general problem structure
  - multi-phase / multi-mode
  - free endpoints
- handling of algebraic variables
- variable polynomial degree on each interval / *hp*-method
- error estimates and validation, e.g. by simulating or costate estimations
- strong symbolic handling, well established modeling language
- applications
  - optimal control
  - parameter optimization
  - training neural components