Integration Error Regularization in Direct Optimal Control using Embedded RK Methods

Jakob Harzer, Jochem De Schutter and Moritz Diehl

European Control Conference 2025
June 27, 2025

universität freiburg

Content Disclaimer

- Runge-Kutta integration methods

Content Disclaimer

- Runge-Kutta integration methods
- A weird numerical phenomenon

Content Disclaimer

- Runge-Kutta integration methods
- A weird numerical phenomenon
- . . . where it comes from

Content Disclaimer

- Runge-Kutta integration methods
- A weird numerical phenomenon
- . . . where it comes from
- . . . and what we can do about it

A weird numerical phenomenon

Motivation

- Minimal Example:

$$\begin{aligned} \min_{x(\cdot),\, u \in \mathbb{R}} \quad & 1 - x(1) \\ \text{s.t.} \quad & x(0) = 1, \\ & \dot{x}(t) = -ux, \quad \forall t \in [0,1], \\ & 0 \leq u \leq 30 \end{aligned}$$

Motivation

- Minimal Example:

$$\begin{aligned} \min_{x(\cdot),\, u \in \mathbb{R}} \quad & 1 - x(1) \\ \text{s.t.} \quad & x(0) = 1, \\ & \dot{x}(t) = -ux, \quad \forall t \in [0,1], \\ & 0 \leq u \leq 30 \end{aligned}$$

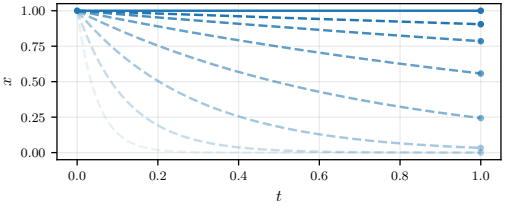
- Analytical solution to IVP:

$$x(t) = e^{-ut}$$

Motivation

- Minimal Example:

$$\begin{aligned} \min_{x(\cdot),\, u \in \mathbb{R}} \quad & 1 - x(1) \\ \text{s.t.} \quad & x(0) = 1, \\ & \dot{x}(t) = -ux, \quad \forall t \in [0, 1], \\ & 0 \leq u \leq 30 \end{aligned}$$



- Analytical solution to IVP:

$$x(t) = e^{-ut}$$

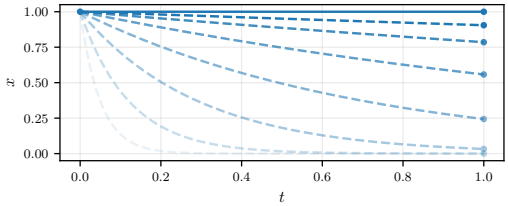
Motivation

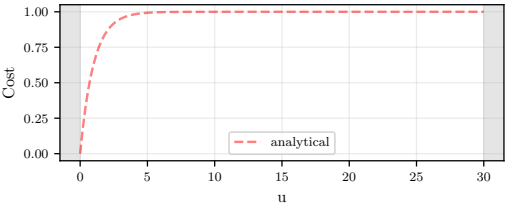
- Minimal Example:

$$\begin{aligned} \min_{x(\cdot),\, u \in \mathbb{R}} \quad & 1 - x(1) \\ \text{s.t.} \quad & x(0) = 1, \\ & \dot{x}(t) = -ux, \quad \forall t \in [0,1], \\ & 0 \leq u \leq 30 \end{aligned}$$

- Analytical solution to IVP:

$$x(t) = e^{-ut}$$





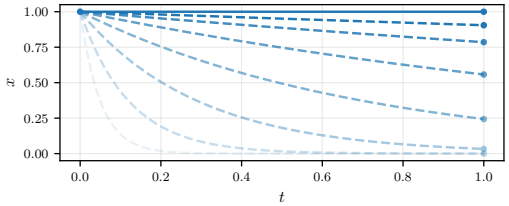
Motivation

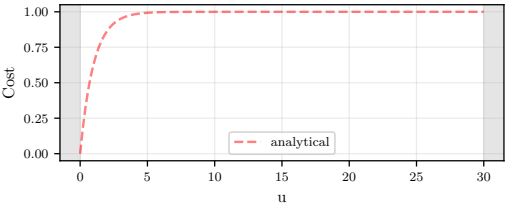
- Minimal Example:

$$\begin{aligned} \min_{x(\cdot),\, u \in \mathbb{R}} \quad & 1 - x(1) \\ \text{s.t.} \quad & x(0) = 1, \\ & \dot{x}(t) = -ux, \quad \forall t \in [0,1], \\ & 0 \leq u \leq 30 \end{aligned}$$

- Solution:

$$\begin{aligned} u^* &= 0 \\ x^*(t) &= e^{-u^* t} = 1 \end{aligned}$$





Motivation

- Discretize with single step of RK4, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, u \in \mathbb{R}} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\mathrm{RK4}}(x_0, x_1, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

Motivation

- Discretize with single step of RK4, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, u \in \mathbb{R}} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\mathrm{RK4}}(x_0, x_1, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Initialize $u_{\mathrm{init}} = 10$, solve with IPOPT.

Motivation

- Discretize with single step of RK4, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, u \in \mathbb{R}} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\mathrm{RK4}}(x_0, x_1, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

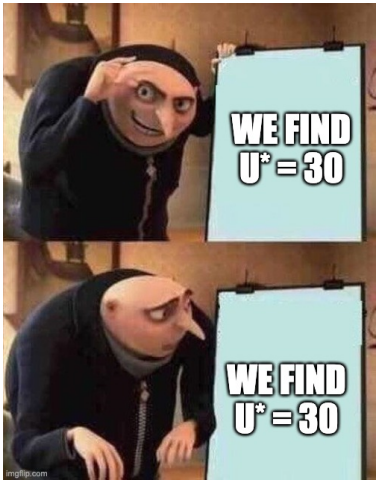
- Initialize $u_{\mathrm{init}} = 10$, solve with IPOPT.
- We find $u^*_{\mathrm{RK4}} = 30$.

Motivation

- Discretize with single step of RK4, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, u \in \mathbb{R}} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\mathrm{RK4}}(x_0, x_1, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Initialize $u_{\mathrm{init}} = 10$, solve with IPOPT.
- We find $u^*_{\mathrm{RK4}} = 30$.



Motivation

- Discretize with single step of GL8, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

Motivation

- Discretize with single step of GL8, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\mathrm{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Initialize $u_{\mathrm{init}} = 10$, solve with IPOPT.

Motivation

- Discretize with single step of GL8, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\mathrm{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

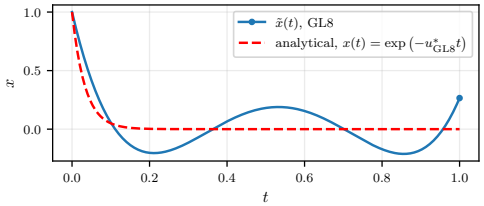
- Initialize $u_{\mathrm{init}} = 10$, solve with IPOPT.
- We find $u^*_{\mathrm{GL8}} = 30$.

Motivation

- Discretize with single step of GL8, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\mathrm{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Initialize $u_{\mathrm{init}} = 10$, solve with IPOPT.
- We find $u^*_{\mathrm{GL8}} = 30$.

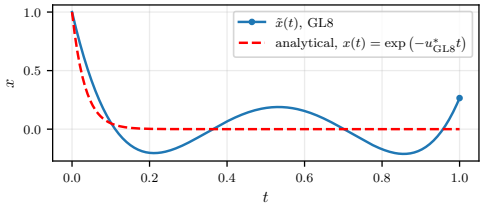


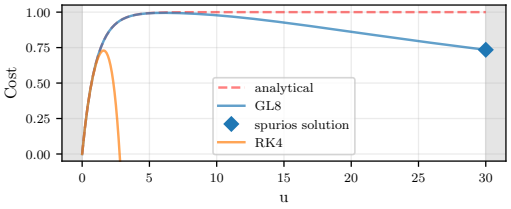
Motivation

- Discretize with single step of GL8, $h = 1$:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

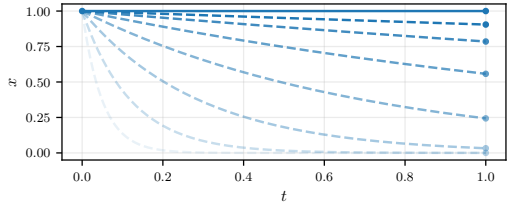
- Initialize $u_{\text{init}} = 10$, solve with IPOPT.
- We find $u^*_{\text{GL8}} = 30$.

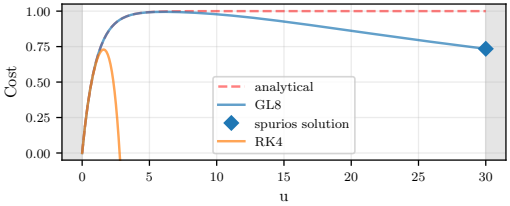




Why is this happening?

$$\dot{x}(t) = -ux(t)$$

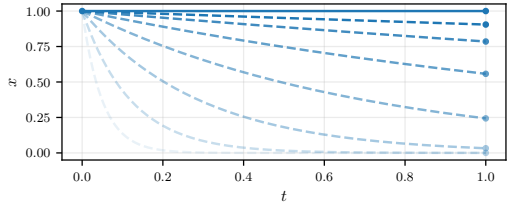


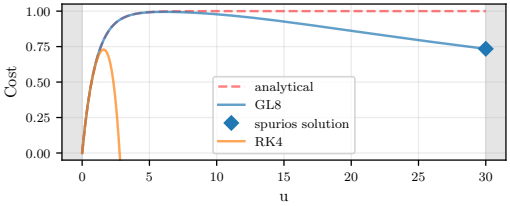


Why is this happening?

$$\dot{x}(t) = -ux(t)$$

- Stiffness of dynamics is controlled by decision variable u

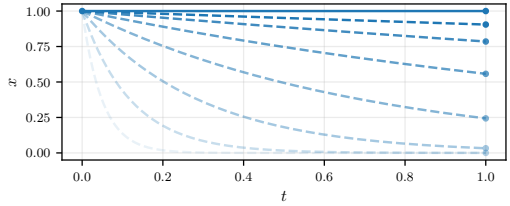


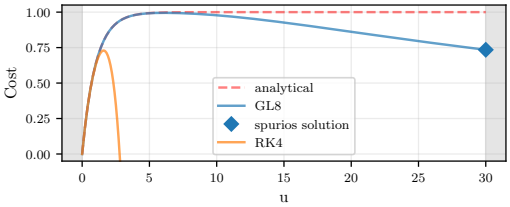


Why is this happening?

$$\dot{x}(t) = -ux(t)$$

- Stiffness of dynamics is controlled by decision variable u
- Larger stiffness $\rightarrow$ larger integration error

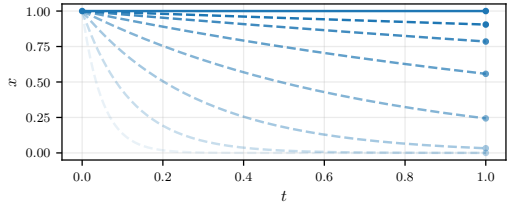


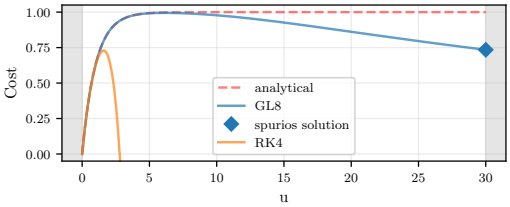


Why is this happening?

$$\dot{x}(t) = -ux(t)$$

- Stiffness of dynamics is controlled by decision variable u
- Larger stiffness $\rightarrow$ larger integration error
- A large integration error minimizes the objective

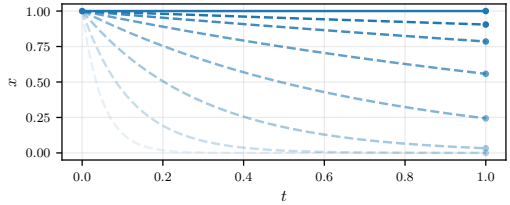


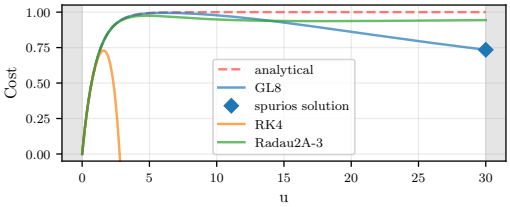


Why is this happening?

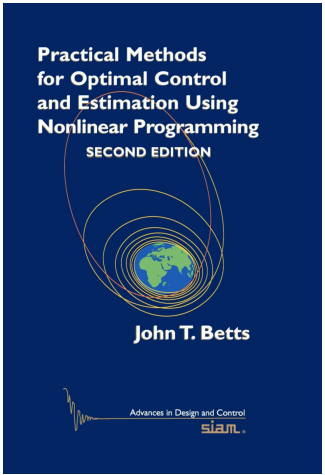
$$\dot{x}(t) = -ux(t)$$

- Stiffness of dynamics is controlled by decision variable u
- Larger stiffness $\rightarrow$ larger integration error
- A large integration error minimizes the objective

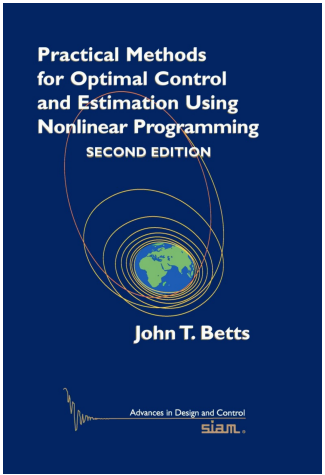




Betts' Thoughts on this [1]



Betts' Thoughts on this [1]



The methods used to solve the differential equations and optimize the functions are intimately related. Furthermore design and development of the "simulation" or experimental trial should be done with the intent to optimize. Optimization is not an afterthought. Indeed, perhaps the most important issue needed to successfully solve a problem is the proper formulation. And so I close with the following:

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

[11] The proof is left to the student.

What to do?

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

What to do?

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

- more accurate discretization

What to do?

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

- more accurate discretization
- add guiding constraints or regularization

What to do?

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

- more accurate discretization
- add guiding constraints or regularization
- provide a better initialization

What to do?

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

- more accurate discretization
- add guiding constraints or regularization
- provide a better initialization

– no compute in online setting

What to do?

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

- more accurate discretization
- add guiding constraints or regularization
- provide a better initialization

– no compute in online setting

– problem too complex, we don't know how

What to do?

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

- more accurate discretization – no compute in online setting
- add guiding constraints or regularization – problem too complex, we don't know how
- provide a better initialization – problem too complex, we don't know how

What to do?

THEERUM

If there is a flaw in the problem formulation, the optimization algorithm will find it.[11]

□

- more accurate discretization – no compute in online setting
- add guiding constraints or regularization – problem too complex, we don't know how
- provide a better initialization – problem too complex, we don't know how

We propose: efficiently **estimate and regularize** the integration error online!

Runge-Kutta Methods

Runge-Kutta Methods

- Initial Value Problem:

$$
\begin{aligned}
x(0) &= \bar{x}_0 \\
\dot{x}(t) &= f(x(t), u(t)), \quad \forall t \in [0, t_\mathrm{f}]
\end{aligned}
$$

Runge-Kutta Methods

- Initial Value Problem:

$$x(0) = \bar{x}_0$$
$$\dot{x}(t) = f(x(t), u(t)), \quad \forall t \in [0, t_\mathrm{f}]$$

- N intervals, intermediate values

$$x_0 = \bar{x}_0, x_1, \ldots, x_N$$

at times $0 = t_0, t_1, \ldots, t_{N-1}, t_N = t_\mathrm{f}$.

Runge-Kutta Methods

- Initial Value Problem:

$$x(0) = \bar{x}_0$$
$$\dot{x}(t) = f(x(t), u(t)), \quad \forall t \in [0, t_\mathrm{f}]$$

- N intervals, intermediate values

$$x_0 = \bar{x}_0, x_1, \ldots, x_N$$

  at times $0 = t_0, t_1, \ldots, t_{N-1}, t_N = t_\mathrm{f}$.

- IVP over one interval:

$$x'(t_k) = x_k$$
$$\dot{x}'(t) = f(x'(t), u(t)), \forall t \in [t_k, t_k + h]$$

Runge-Kutta Methods

- Initial Value Problem:

$$x(0) = \bar{x}_0$$
$$\dot{x}(t) = f(x(t), u(t)), \quad \forall t \in [0, t_\mathrm{f}]$$

- N intervals, intermediate values

$$x_0 = \bar{x}_0, x_1, \ldots, x_N$$

at times $0 = t_0, t_1, \ldots, t_{N-1}, t_N = t_\mathrm{f}$.

- IVP over one interval:

$$x'(t_k) = x_k$$
$$\dot{x}'(t) = f(x'(t), u(t)), \forall t \in [t_k, t_k + h]$$

- Runge-Kutta Method with d stages:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \ldots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i
and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

Runge-Kutta Methods

- Initial Value Problem:

$$x(0) = \bar{x}_0$$
$$\dot{x}(t) = f(x(t), u(t)), \quad \forall t \in [0, t_\mathrm{f}]$$

- N intervals, intermediate values

$$x_0 = \bar{x}_0, x_1, \ldots, x_N$$

at times $0 = t_0, t_1, \ldots, t_{N-1}, t_N = t_\mathrm{f}$.

- IVP over one interval:

$$x'(t_k) = x_k$$
$$\dot{x}'(t) = f(x'(t), u(t)), \forall t \in [t_k, t_k + h]$$

- Runge-Kutta Method with d stages:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \ldots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

- We write shorthand:

$$0 = G_{\mathrm{IRK}}(x_k, x_{k+1}, u_k, z_k)$$

Runge-Kutta Methods

- Runge-Kutta Method with d stages:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \ldots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i
and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

- We write shorthand:

$$0 = G_{\mathrm{IRK}}(x_k, x_{k+1}, u_k, z_k)$$

Runge-Kutta Methods

- Butcher Tableau

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,d} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_d & a_{d,1} & a_{d,2} & \cdots & a_{d,d} \\ \hline & b_1 & b_2 & \cdots & b_d \end{array}$$

- Runge-Kutta Method with d stages:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \ldots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i
and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

- We write shorthand:

$$0 = G_{\mathrm{IRK}}(x_k, x_{k+1}, u_k, z_k)$$

Runge-Kutta Methods

- Butcher Tableau

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,d} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_d & a_{d,1} & a_{d,2} & \cdots & a_{d,d} \\ \hline & b_1 & b_2 & \cdots & b_d \end{array}$$

- For some RK methods: Continuous approximation $\tilde{x}(t)$ over the interval

- Runge-Kutta Method with d stages:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \ldots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

- We write shorthand:

$$0 = G_{\mathrm{IRK}}(x_k, x_{k+1}, u_k, z_k)$$

Runge-Kutta Methods

- Butcher Tableau

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,d} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_d & a_{d,1} & a_{d,2} & \cdots & a_{d,d} \\ \hline & b_1 & b_2 & \cdots & b_d \end{array}$$

- For some RK methods: Continuous approximation $\tilde{x}(t)$ over the interval
- Global error

$$x_N - x(t_\mathrm{f}) = \mathcal{O}(h^p)$$

- Runge-Kutta Method with d stages:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \ldots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

- We write shorthand:

$$0 = G_{\mathrm{IRK}}(x_k, x_{k+1}, u_k, z_k)$$

Estimating the (Local) Integration Error

- Local error

$$e_k = x_{k+1} - x'(t_k + h) = \mathcal{O}\big(h^{p+1}\big)$$

Estimating the (Local) Integration Error

- Local error

$$e_k = x_{k+1} - x'(t_k + h) = \mathcal{O}\left(h^{p+1}\right)$$

- that we would like to estimate

Estimating the (Local) Integration Error

- Local error

$$e_k = x_{k+1} - x'(t_k + h) = \mathcal{O}\left(h^{p+1}\right)$$

- that we would like to estimate
- Why estimate?

Estimating the (Local) Integration Error

- Local error

$$e_k = x_{k+1} - x'(t_k + h) = \mathcal{O}\left(h^{p+1}\right)$$

- that we would like to estimate
- Why estimate?
  - Simulation: Step size control

Estimating the (Local) Integration Error

- Local error

$$e_k = x_{k+1} - x'(t_k + h) = \mathcal{O}\left(h^{p+1}\right)$$

- that we would like to estimate
- Why estimate?
  - Simulation: Step size control
  - Control: Adaptive Mesh Refinement

Estimating the (Local) Integration Error

- Local error

$$e_k = x_{k+1} - x'(t_k + h) = \mathcal{O}\left(h^{p+1}\right)$$

- that we would like to estimate
- Why estimate?
  - Simulation: Step size control
  - Control: Adaptive Mesh Refinement

- Some options:

Estimating the (Local) Integration Error

- Local error

$$e_k = x_{k+1} - x'(t_k + h) = \mathcal{O}\big(h^{p+1}\big)$$

- that we would like to estimate
- Why estimate?
  - Simulation: Step size control
  - Control: Adaptive Mesh Refinement

- Some options:
  - If $\tilde{x}(t)$ available: approximate dynamics violation [4, 5]

$$\int_{t_k}^{t_{k+1}} \left\| \dot{\tilde{x}}(t) - f(\tilde{x}(t), u_k) \right\| \, dt$$

Estimating the (Local) Integration Error

- Local error

$$e_k = x_{k+1} - x'(t_k + h) = \mathcal{O}\left(h^{p+1}\right)$$

- that we would like to estimate
- Why estimate?
  - Simulation: Step size control
  - Control: Adaptive Mesh Refinement

- Some options:
  - If $\tilde{x}(t)$ available: approximate dynamics violation [4, 5]

$$\int_{t_k}^{t_{k+1}} \left\| \dot{\tilde{x}}(t) - f(\tilde{x}(t), u_k) \right\| \, dt$$

  - Compare with other integrator result:

$$\hat{e}_k = x_{k+1} - \hat{x}_{k+1}$$

Embedded Runge-Kutta Methods

Embedded Runge Kutta Methods

- Two Integrators that use the same stage evaluations k_i:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \dots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

$$\hat{x}_{k+1} = x_k + h \sum_{i=1}^{d} \hat{b}_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i
and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

Embedded Runge Kutta Methods

- Two Integrators that use the same stage evaluations k_i:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \dots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

$$\hat{x}_{k+1} = x_k + h \sum_{i=1}^{d} \hat{b}_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i
and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,d} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_d & a_{d,1} & a_{d,2} & \cdots & a_{d,d} \\ \hline & b_1 & b_2 & \cdots & b_d \\ & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_d \end{array}$$

Embedded Runge Kutta Methods

- Two Integrators that use the same stage evaluations k_i:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \dots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

$$\hat{x}_{k+1} = x_k + h \sum_{i=1}^{d} \hat{b}_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i
and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

0	0	0
1	1	0
	$\frac{1}{2}$	$\frac{1}{2}$

Embedded Runge Kutta Methods

- Two Integrators that use the same stage evaluations k_i:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \dots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

$$\hat{x}_{k+1} = x_k + h \sum_{i=1}^{d} \hat{b}_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i
and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

0	0	0
1	1	0
	$\frac{1}{2}$	$\frac{1}{2}$
	1	0

Embedded Runge Kutta Methods

- Two Integrators that use the same stage evaluations k_i:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \dots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

$$\hat{x}_{k+1} = x_k + h \sum_{i=1}^{d} \hat{b}_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i
and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

0	0	0
1	1	0
	$\frac{1}{2}$	$\frac{1}{2}$
	1	0

- Order: $p(\hat{p})$

Embedded Runge Kutta Methods

- Two Integrators that use the same stage evaluations k_i:

$$z_{k,i} = x_k + h \sum_{j=1}^{d} a_{i,j} k_j, \quad i = 1, \dots, d$$

$$x_{k+1} = x_k + h \sum_{i=1}^{d} b_i k_i$$

$$\hat{x}_{k+1} = x_k + h \sum_{i=1}^{d} \hat{b}_i k_i$$

with stage values $z_{k,i} \in \mathbb{R}^{n_x}$ at times c_i and dynamics evaluations $k_i = f(z_{k,i}, u_k)$

0	0	0
1	1	0
	$\frac{1}{2}$	$\frac{1}{2}$
	1	0

- Order: $p(\hat{p})$
- Error estimate

$$\hat{e}_k = x_{k+1} - \hat{x}_{k+1}$$

of the *lower order* method.

Embedded Runge Kutta Methods: Examples

- Scipys `solve_ivp` and Matlabs `ode45`: 'RK45'

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
x_1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$\hat{x}_1$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Embedded Runge Kutta Methods: Examples

- Implicit Methods with d stages can be extended with lower order method of order d [3]
- For example: GL8(4)

$$\begin{array}{c|cc} c_0 & 0 & 0 \\ c & 0 & A \\ \hline & 0 & b^\top \\ & \gamma_0 & \hat{b}^\top \end{array}$$

Integration Error Regularization

Integration Error Regularization

- NLP from direct transcription:

$$
\begin{aligned}
\min_{w} \quad & J(w) \\
\text{s.t.} \quad & x_0 = \bar{x}_0, \\
& 0 = G_{\text{IRK}}(x_k, x_{k+1}, u_k, z_k), \quad && \forall k \in \mathcal{K}, \\
& 0 \leq h(x_k, u_k, z_k), && \forall k \in \mathcal{K}
\end{aligned}
$$

Integration Error Regularization

- NLP from direct transcription:

$$
\begin{aligned}
\min_{w} \quad & J(w) \\
\text{s.t.} \quad & x_0 = \bar{x}_0, \\
& 0 = G_{\text{IRK}}(x_k, x_{k+1}, u_k, z_k), \quad && \forall k \in \mathcal{K}, \\
& 0 \leq h(x_k, u_k, z_k), && \forall k \in \mathcal{K}
\end{aligned}
$$

- with

$$
\begin{aligned}
w = (& x_0, \ldots, x_{N-1}, x_N \\
& z_0, \ldots, z_{N-1} \\
& u_0, \ldots, u_{N-1})
\end{aligned}
$$

Integration Error Regularization

- NLP from direct transcription:

$$\begin{aligned}
\min_{w} \quad & J(w) \\
\text{s.t.} \quad & x_0 = \bar{x}_0, \\
& 0 = G_{\text{IRK}}(x_k, x_{k+1}, u_k, z_k), \quad \forall k \in \mathcal{K}, \\
& 0 \leq h(x_k, u_k, z_k), \quad \forall k \in \mathcal{K}
\end{aligned}$$

- with

$$\begin{aligned}
w = (& x_0, \ldots, x_{N-1}, x_N \\
& z_0, \ldots, z_{N-1} \\
& u_0, \ldots, u_{N-1})
\end{aligned}$$

- Error estimate

$$\hat{e}_k = \hat{x}_{k+1} - x_{k+1} = \sum_{i=1}^{d} (\hat{b}_i - b_i) k_i$$

Integration Error Regularization

- NLP from direct transcription:

$$\begin{aligned}
\min_{w} \quad & J(w) \\
\text{s.t.} \quad & x_0 = \bar{x}_0, \\
& 0 = G_{\text{IRK}}(x_k, x_{k+1}, u_k, z_k), \quad \forall k \in \mathcal{K}, \\
& 0 \leq h(x_k, u_k, z_k), \quad \forall k \in \mathcal{K}
\end{aligned}$$

- with

$$\begin{aligned}
w = (& x_0, \ldots, x_{N-1}, x_N \\
& z_0, \ldots, z_{N-1} \\
& u_0, \ldots, u_{N-1})
\end{aligned}$$

- Error estimate

$$\hat{e}_k = \hat{x}_{k+1} - x_{k+1} = \sum_{i=1}^{d} (\hat{b}_i - b_i) k_i$$

- Collect the errors in a vector

$$\hat{E}(w) = [\hat{e}_0^\top, \ldots, \hat{e}_{N-1}^\top]^\top \in \mathbb{R}^{N n_x}$$

Integration Error Regularization

- NLP from direct transcription:

$$\begin{aligned}\min_{w} \quad & J(w) \\ \text{s.t.} \quad & x_0 = \bar{x}_0, \\ & 0 = G_{\text{IRK}}(x_k, x_{k+1}, u_k, z_k), \quad \forall k \in \mathcal{K}, \\ & 0 \leq h(x_k, u_k, z_k), \quad \forall k \in \mathcal{K}\end{aligned}$$

- with

$$\begin{aligned}w = (&x_0, \ldots, x_{N-1}, x_N \\ &z_0, \ldots, z_{N-1} \\ &u_0, \ldots, u_{N-1})\end{aligned}$$

- Error estimate

$$\hat{e}_k = \hat{x}_{k+1} - x_{k+1} = \sum_{i=1}^{d} (\hat{b}_i - b_i) k_i$$

- Collect the errors in a vector

$$\hat{E}(w) = [\hat{e}_0^\top, \ldots, \hat{e}_{N-1}^\top]^\top \in \mathbb{R}^{N n_x}$$

- Maximum Error

$$E_{\max} = [e_{0,\max}^\top, \ldots, e_{k,\max}^\top]^\top \in \mathbb{R}^{N n_x}$$

Integration Error Regularization

- NLP from direct transcription:

$$\begin{aligned}\min_{w} \quad & J(w) + \phi(w) \\ \text{s.t.} \quad & x_0 = \bar{x}_0, \\ & 0 = G_{\text{IRK}}(x_k, x_{k+1}, u_k, z_k), \quad \forall k \in \mathcal{K}, \\ & 0 \leq h(x_k, u_k, z_k), \quad \forall k \in \mathcal{K}\end{aligned}$$

- Regularization term

$$\phi(w) = \|\text{diag}(E_{\max})^{-1} \hat{E}(w)\|_p^q$$

- Error estimate

$$\hat{e}_k = \hat{x}_{k+1} - x_{k+1} = \sum_{i=1}^{d} (\hat{b}_i - b_i) k_i$$

- Collect the errors in a vector

$$\hat{E}(w) = [\hat{e}_0^\top, \ldots, \hat{e}_{N-1}^\top]^\top \in \mathbb{R}^{N n_x}$$

- Maximum Error

$$E_{\max} = [e_{0,\max}^\top, \ldots, e_{k,\max}^\top]^\top \in \mathbb{R}^{N n_x}$$

Intro Example, cont.

- Minimal Example:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 + \phi(z) \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

Intro Example, cont.

- Minimal Example:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 + \phi(z) \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Estimate with GL8(4), normalize and regularize error with

$$e_{\max} = 20\,\%, \qquad \phi(z) = \left\| \frac{\hat{e}(z)}{e_{\max}} \right\|_p^q$$

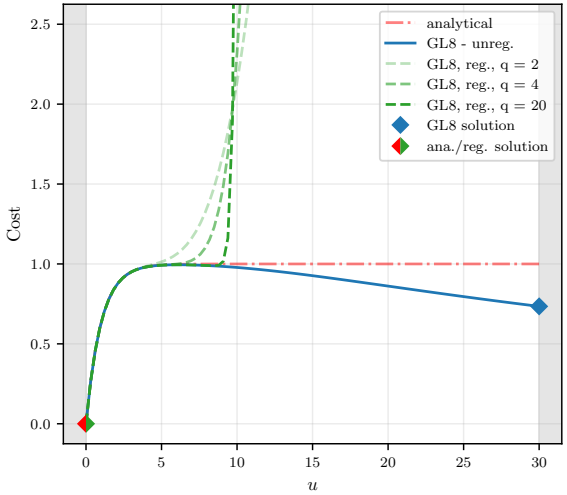
Intro Example, cont.

- Minimal Example:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 + \phi(z) \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Estimate with GL8(4), normalize and regularize error with

$$e_{\max} = 20\,\%, \qquad \phi(z) = \left\| \frac{\hat{e}(z)}{e_{\max}} \right\|_p^q$$



Example: Hang-Glider

Example: Hang-Glider

$$
\begin{aligned}
\min_{x(\cdot),\, u(\cdot),\, t_\mathrm{f}} \quad & -p_x(t_\mathrm{f}) \\
\text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)), && \forall t \in [0, t_\mathrm{f}], \\
& 0 \leq u(t) \leq 1.4, && \forall t \in [0, t_\mathrm{f}]
\end{aligned}
$$

Example: Hang-Glider

$$\begin{aligned} \min_{x(\cdot),\, u(\cdot),\, t_\mathrm{f}} \quad & -p_x(t_\mathrm{f}) \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)), \qquad \forall t \in [0, t_\mathrm{f}], \\ & 0 \le u(t) \le 1.4, \qquad \forall t \in [0, t_\mathrm{f}] \end{aligned}$$

- with initial and terminal constraint:

$$\begin{aligned} p_x(0) &= 0\,\mathrm{m} \\ p_y(0) &= 1000\,\mathrm{m} & p_y(t_\mathrm{f}) &= 900\,\mathrm{m} \\ v_x(0) &= \bar{v}_{x,0} & v_x(t_\mathrm{f}) &= \bar{v}_{x,0} \\ v_y(0) &= \bar{v}_{y,0} & v_y(t_\mathrm{f}) &= \bar{v}_{y,0} \end{aligned}$$

Example: Hang-Glider

$$\begin{aligned} \min_{x(\cdot),\, u(\cdot),\, t_\mathrm{f}} \quad & -p_x(t_\mathrm{f}) \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)), \qquad \forall t \in [0, t_\mathrm{f}], \\ & 0 \le u(t) \le 1.4, \qquad \forall t \in [0, t_\mathrm{f}] \end{aligned}$$

- with initial and terminal constraint:

$$\begin{aligned} p_x(0) &= 0\,\mathrm{m} \\ p_y(0) &= 1000\,\mathrm{m} \qquad & p_y(t_\mathrm{f}) &= 900\,\mathrm{m} \\ v_x(0) &= \bar{v}_{x,0} & v_x(t_\mathrm{f}) &= \bar{v}_{x,0} \\ v_y(0) &= \bar{v}_{y,0} & v_y(t_\mathrm{f}) &= \bar{v}_{y,0} \end{aligned}$$

- Literature optimum: $p_x^*(t_f) = 1248\,\mathrm{m}$[1]

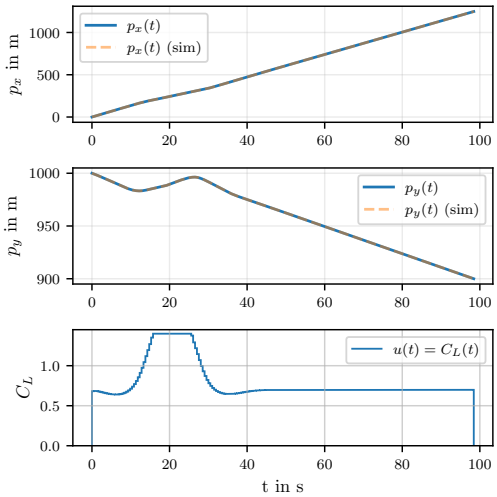
Example: Hang-Glider

$$
\begin{aligned}
\min_{x(\cdot),\,u(\cdot),\,t_\mathrm{f}} \quad & -p_x(t_\mathrm{f}) \\
\text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)), && \forall t \in [0, t_\mathrm{f}], \\
& 0 \le u(t) \le 1.4, && \forall t \in [0, t_\mathrm{f}]
\end{aligned}
$$

- with initial and terminal constraint:

$$
\begin{aligned}
p_x(0) &= 0\,\mathrm{m} \\
p_y(0) &= 1000\,\mathrm{m} & p_y(t_\mathrm{f}) &= 900\,\mathrm{m} \\
v_x(0) &= \bar{v}_{x,0} & v_x(t_\mathrm{f}) &= \bar{v}_{x,0} \\
v_y(0) &= \bar{v}_{y,0} & v_y(t_\mathrm{f}) &= \bar{v}_{y,0}
\end{aligned}
$$

- Literature optimum: $p_x^*(t_f) = 1248\,\mathrm{m}$[1]



Example: Hang-Glider

- Discretize with $N = 30$ intervals, piecewise constant controls

Example: Hang-Glider

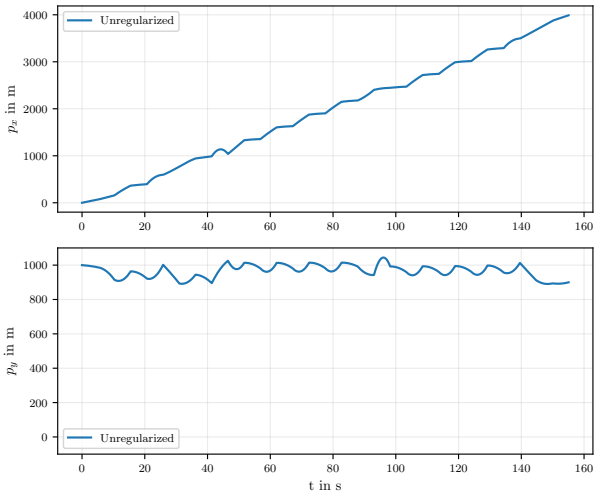
- Discretize with $N = 30$ intervals, piecewise constant controls
- and Heun-Euler of order 2(1):

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}$$

Example: Hang-Glider

- Discretize with $N = 30$ intervals, piecewise constant controls
- and Heun-Euler of order 2(1):

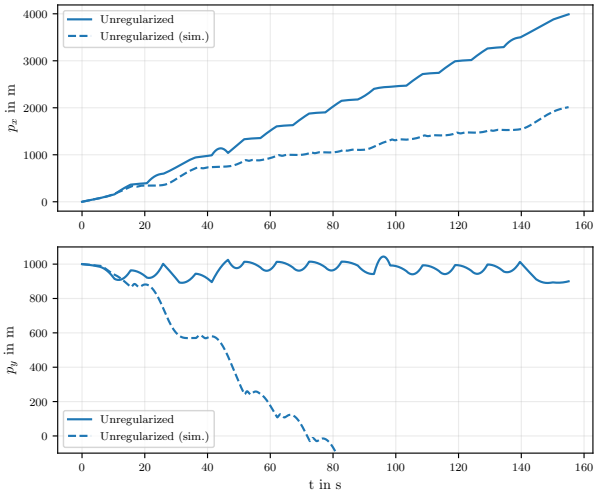
$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}$$



Example: Hang-Glider

- Discretize with $N = 30$ intervals, piecewise constant controls
- and Heun-Euler of order 2(1):

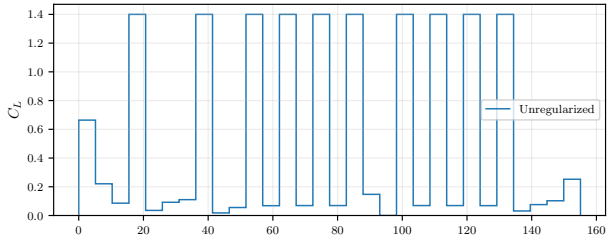
$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}$$



Example: Hang-Glider

- Discretize with $N = 30$ intervals, piecewise constant controls
- and Heun-Euler of order 2(1):

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}$$



Example: Hang-Glider

- Discretize with $N = 30$ intervals, piecewise constant controls
- and Heun-Euler of order 2(1):

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}$$

- Estimate, normalize and regularize error with

$$e_{\max} = 1\,\%$$

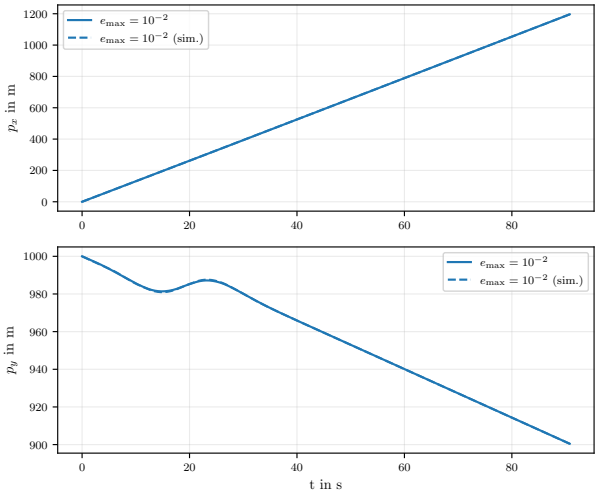
Example: Hang-Glider

- Discretize with $N = 30$ intervals, piecewise constant controls
- and Heun-Euler of order 2(1):

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}$$

- Estimate, normalize and regularize error with

$$e_{\max} = 1\,\%$$



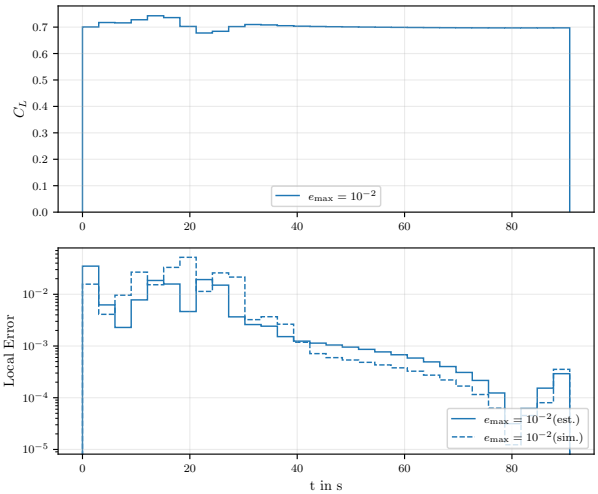
Example: Hang-Glider

- Discretize with $N = 30$ intervals, piecewise constant controls
- and Heun-Euler of order 2(1):

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}$$

- Estimate, normalize and regularize error with

$$e_{\max} = 1\,\%$$



Summary

- We presented a ‘robustification’ method to avoid solutions with large integration error

Summary

- We presented a 'robustification' method to avoid solutions with large integration error
- At the cost of optimality + (maybe) extra dynamics evaluations

Summary

- We presented a 'robustification' method to avoid solutions with large integration error
- At the cost of optimality + (maybe) extra dynamics evaluations
- We can force bad discretizations to 'work'.

Summary

- We presented a 'robustification' method to avoid solutions with large integration error
- At the cost of optimality + (maybe) extra dynamics evaluations
- We can force bad discretizations to 'work'.

Follow Up Work

Summary

- We presented a 'robustification' method to avoid solutions with large integration error
- At the cost of optimality + (maybe) extra dynamics evaluations
- We can force bad discretizations to 'work'.

Follow Up Work

- Is this really worth it? When?

Summary

- We presented a 'robustification' method to avoid solutions with large integration error
- At the cost of optimality + (maybe) extra dynamics evaluations
- We can force bad discretizations to 'work'.

Follow Up Work

- Is this really worth it? When?
- Adaptive Online Stepsize Control with Embedded RK Methods

Thank you for your attention!

(and be careful when discretizing OCPs with free final time)

Useful Sources I

[1] John T. Betts.
Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition.
Society for Industrial and Applied Mathematics, second edition, 2010.

[2] Matthias Gerdts.
Optimal Control of ODEs and DAEs.
De Gruyter, Berlin, Boston, 2012.

[3] E. Hairer, S.P. Nørsett, and G. Wanner.
Solving Ordinary Differential Equations I: Nonstiff Problems.
Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2008.

[4] Martin P. Neuenhofen and Eric C. Kerrigan.
An integral penalty-barrier direct transcription method for optimal control.
In 2020 59th IEEE Conference on Decision and Control (CDC), pages 456–463, 2020.

Useful Sources II

[5] Martin P. Neuenhofen and Eric C. Kerrigan.
Dynamic optimization with convergence guarantees, 2021.

Intro Example, cont.

- Minimal Example:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 + \phi(z) \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Estimate with GL8(4), normalize and regularize error with

$$e_{\max} = 20\,\%, \qquad \phi(z) = \left\| \frac{\hat{e}(z)}{e_{\max}} \right\|_p^q$$

Intro Example, cont.

- Minimal Example:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 + \phi(z) \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Estimate with GL8(4), normalize and regularize error with

$$e_{\max} = 20\,\%, \qquad \phi(z) = \left\| \frac{\hat{e}(z)}{e_{\max}} \right\|_p^q$$

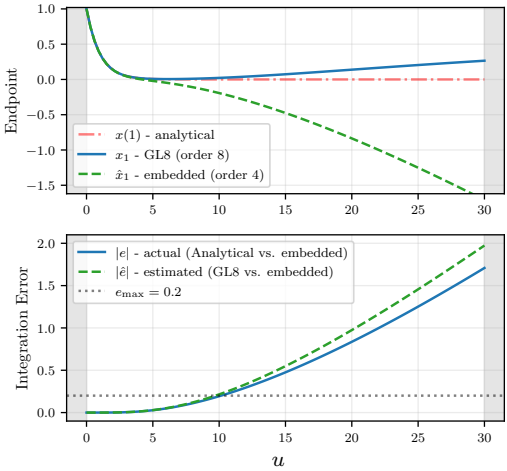
Intro Example, cont.

- Minimal Example:

$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 + \phi(z) \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Estimate with GL8(4), normalize and regularize error with

$$e_{\max} = 20\,\%, \qquad \phi(z) = \left\| \frac{\hat{e}(z)}{e_{\max}} \right\|_p^q$$



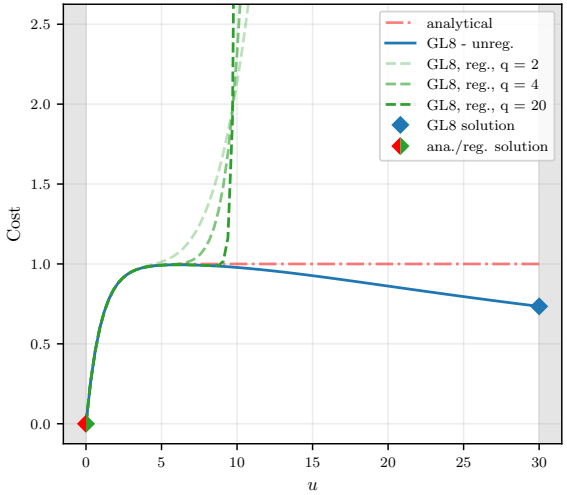
Intro Example, cont.

- Minimal Example:

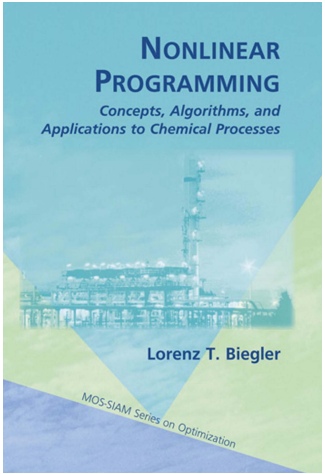
$$\begin{aligned} \min_{x_0, x_1, z, u} \quad & 1 - x_1 + \phi(z) \\ \text{s.t.} \quad & x_0 = 1, \\ & 0 = G_{\text{GL8}}(x_0, x_1, z, u), \\ & 0 \leq u \leq 30 \end{aligned}$$

- Estimate with GL8(4), normalize and regularize error with

$$e_{\max} = 20\,\%, \qquad \phi(z) = \left\| \frac{\hat{e}(z)}{e_{\max}} \right\|_p^q$$



Bonus: Biegler already did it !?



With this estimate, N sufficiently large, and a user-specified error tolerance ϵ, appropriate values of h_i can be determined by adding the constraints

$$\sum_{i=1}^{N} h_i = t_f, \quad h_i \geq 0, \tag{10.21a}$$

$$\tilde{C} \| T_i(t_{nc}) \| \leq \epsilon \tag{10.21b}$$

to (10.19). This extended formulation has been developed in [396] and demonstrated on a number of challenging reactor optimization problems. In particular, this approach allows variable finite elements to track and adapt to steep profiles encountered over the course of an optimization problem.

[396] S. Vasantharajan and L. T. Biegler. Simultaneous Strategies for Parameter Optimization and Accurate Solution of Differential-Algebraic Systems. *Comput. Chem. Eng.*, 14:1083, 1990.

Bonus: Matthias Gerdts' Observation [2]

Example 5.1.7 ([146, p. 272]). Consider the following optimal control problem:

Minimize

$$\frac{1}{2}\int_0^1 u(t)^2 + 2z(t)^2 dt$$

subject to the constraints

$$\dot{z}(t) = \frac{1}{2}z(t) + u(t), \quad z(0) = 1.$$

The optimal solution is

$$\hat{z}(t) = \frac{2\exp(3t) + \exp(3)}{\exp(3t/2)(2 + \exp(3))}, \quad \hat{u}(t) = \frac{2(\exp(3t) - \exp(3))}{\exp(3t/2)(2 + \exp(3))}.$$

We consider the modified Euler method (5.23) and Heun's method defined by

0	0	0
1/2	1/2	0
	0	1

0	0	0
1	1	0
	1/2	1/2

Table 5.2 shows the error in z in the norm $\|\cdot\|_\infty$ for modified Euler's method, if the control is discretized at t_i and $t_i + h/2$ with independent optimization variables u_i and $u_{i+1/2}$.

N	Error in z	Order
10	$0.4254224673693650E+00$	–
20	$0.4258159920666613E+00$	−0.0013339
40	$0.4260329453139864E+00$	−0.0007349
80	$0.4260267362368171E+00$	0.0000210
160	$0.4261445411996390E+00$	−0.0003989
320	$0.4260148465889140E+00$	0.0004391

Table 5.2. Error for modified Euler method with independent optimization variables at t_i and $t_{i+1/2}$.

There is no convergence at all. Figure 5.2 shows the numerical solution for $N = 40$. The control u oscillates strongly between 0 at $t_i + h/2$ and approximately $-1/(2h)$ at t_i.