Lecture 2: Numerical simulation and direct collocation

Moritz Diehl and Armin Nurkanović

Systems Control and Optimization Laboratory (syscop)
Summer School on Direct Methods for Optimal Control of Nonsmooth Systems
September 11-15, 2023

universität freiburg

Outline of the lecture

Ordinary differential equations and controlled dynamical system

Let:

- $t \in \mathbb{R}$ be the time
- $x(t) \in \mathbb{R}^{n_x}$ the differential states
- $u(t) \in \mathbb{R}^{n_u}$ a given control function
- denote by $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$

Ordinary differential equations

Ordinary differential equations and controlled dynamical system

Let:

- $t \in \mathbb{R}$ be the time
- $x(t) \in \mathbb{R}^{n_x}$ the differential states
- $u(t) \in \mathbb{R}^{n_u}$ a given control function
- denote by $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$

Ordinary differential equations

- Let $F : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ be a function such that the Jacobian $\frac{\partial F}{\partial \dot{x}}(\cdot)$ is invertible. The system of equations:

$$F(t, \dot{x}(t), x(t), u(t)) = 0,$$

is called an Ordinary Differential Equation (ODE).

Ordinary differential equations and controlled dynamical system

Let:

- $t \in \mathbb{R}$ be the time
- $x(t) \in \mathbb{R}^{n_x}$ the differential states
- $u(t) \in \mathbb{R}^{n_u}$ a given control function
- denote by $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$

Ordinary differential equations

- Let $F : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ be a function such that the Jacobian $\frac{\partial F}{\partial \dot{x}}(\cdot)$ is invertible. The system of equations:

$$F(t, \dot{x}(t), x(t), u(t)) = 0,$$

  is called an Ordinary Differential Equation (ODE).
- Given a function $f : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ then a system of equations:

$$\dot{x}(t) = f(t, x(t), u(t)) \tag{1}$$

  is called an explicit ODE.

Sufficient conditions for existence and uniqueness

Theorem (Picard-Lindelöf / Cauchy–Lipschitz)

An initial value problem in ODE

$$\dot{x}(t) = f(t, x(t), u(t)), \quad t \in [0, T],$$
$$x(0) = x_0$$

- *with given initial state* x_0*, and controls* $u(t)$*,*
- $f(t, x(t), u(t)) = \hat{f}(t, x(t))$ *is continuous in* t *and Lipschitz continuous in* x

Sufficient conditions for existence and uniqueness

Theorem (Picard-Lindelöf / Cauchy–Lipschitz)

An initial value problem in ODE

$$\dot{x}(t) = f(t, x(t), u(t)), \quad t \in [0, T],$$
$$x(0) = x_0$$

- *with given initial state* x_0*, and controls* $u(t)$*,*
- $f(t, x(t), u(t)) = \hat{f}(t, x(t))$ *is continuous in* t *and Lipschitz continuous in* x

has a unique solution $x(t),\ t \in [0, T]$.

- f is Lipschitz if $\|f(x) - f(y)\| \leq L\|x - y\|$
- smooth ODEs modeling physics usually Lipschitz

Sufficient conditions for existence and uniqueness

Theorem (Picard-Lindelöf / Cauchy–Lipschitz)

An initial value problem in ODE

$$\dot{x}(t) = f(t, x(t), u(t)), \quad t \in [0, T],$$
$$x(0) = x_0$$

- *with given initial state* x_0*, and controls* $u(t)$*,*
- $f(t, x(t), u(t)) = \hat{f}(t, x(t))$ *is continuous in* t *and Lipschitz continuous in* x

has a unique solution $x(t),\ t \in [0, T]$*.*

- f is Lipschitz if $\|f(x) - f(y)\| \leq L\|x - y\|$
- smooth ODEs modeling physics usually Lipschitz
- if f is only continuous, existence but not uniqueness can be guaranteed, e.g. $\dot{x}(t) = \sqrt{|x(t)|}, x(0) = 0$, solutions: $x(t) = 0$ and $x(t) = \frac{t^2}{4}$

Sufficient conditions for existence and uniqueness

Theorem (Picard-Lindelöf / Cauchy–Lipschitz)

An initial value problem in ODE

$$\dot{x}(t) = f(t, x(t), u(t)), \quad t \in [0, T],$$
$$x(0) = x_0$$

- *with given initial state* x_0, *and controls* $u(t)$,
- $f(t, x(t), u(t)) = \hat{f}(t, x(t))$ *is continuous in* t *and Lipschitz continuous in* x

has a unique solution $x(t),\ t \in [0, T]$.

- f is Lipschitz if $\|f(x) - f(y)\| \leq L\|x - y\|$
- smooth ODEs modeling physics usually Lipschitz
- if f is only continuous, existence but not uniqueness can be guaranteed, e.g. $\dot{x}(t) = \sqrt{|x(t)|}, x(0) = 0$, solutions: $x(t) = 0$ and $x(t) = \frac{t^2}{4}$
- Conditions are only sufficient, ODEs with a non-Lipschiz r.h.s. can have unique solutions

A collection of results in: *Agarwal, Ratan Prakash, Ravi P. Agarwal, and V. Lakshmikantham. Uniqueness and nonuniqueness criteria for ordinary differential equations. Vol. 6. World Scientific, 1993.*

ODE Example: harmonic oscillator

Mass m with spring constant k and friction coefficient c:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -\frac{k}{m}(x_2(t) - u(t)) \quad - \frac{\beta}{m} x_1(t) \end{aligned}$$

ODE Example: harmonic oscillator

Mass m with spring constant k and friction coefficient c:

$$\begin{aligned}\dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -\frac{k}{m}(x_2(t) - u(t)) \quad - \frac{\beta}{m} x_1(t)\end{aligned}$$

- state $x(t) \in \mathbb{R}^2$
- position of mass $x_1(t)$ $\longleftarrow$ measured
- velocity of mass $x_2(t)$
- control action: spring position $u(t) \in \mathbb{R}$ $\longleftarrow$ manipulated

ODE Example: harmonic oscillator

Mass m with spring constant k and friction coefficient c:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -\frac{k}{m}(x_2(t) - u(t)) \quad - \frac{\beta}{m} x_1(t) \end{aligned}$$

- state $x(t) \in \mathbb{R}^2$
- position of mass $x_1(t)$ $\longleftarrow$ measured
- velocity of mass $x_2(t)$
- control action: spring position $u(t) \in \mathbb{R}$ $\longleftarrow$ manipulated

Can summarize as $\dot{x} = f(x, u)$ with

$$f(x,u) = \begin{bmatrix} x_2 \\ -\frac{k}{m}(x_2 - u) - \frac{c}{m} x_1 \end{bmatrix}$$

Basic definitions of numerical simulation

- IVPs have only in special cases a closed form solution
- Instead, compute numerically a solution approximation $\tilde{x}(t)$ that approximately satisfies:

$$\dot{\tilde{x}}(t) \approx f(t, \tilde{x}(t), u(t)), \quad t \in [0, T]$$
$$\tilde{x}(0) = x(0) = x_0$$

Basic definitions of numerical simulation

- IVPs have only in special cases a closed form solution
- Instead, compute numerically a solution approximation $\tilde{x}(t)$ that approximately satisfies:

$$\dot{\tilde{x}}(t) \approx f(t, \tilde{x}(t), u(t)), \quad t \in [0, T]$$
$$\tilde{x}(0) = x(0) = x_0$$

- Recursively generate solution approximation $x_n := \tilde{x}(t_n) \approx x(t_n)$ at N discrete time points $0 = t_0 < t_1 < \ldots < t_N = T$
- Integration interval $[0, T]$ split into subintervals $[t_n, t_{n+1}]$ where $h = t_{n+1} - t_n$
- h - integration step size can be constant, different for every interval, or adaptive

Single step numerical simulation as discrete time system

Single step abstract integration method

$$x_{n+1} = \phi_f(x_n, z_n, u_n),$$
$$0 = \phi_{\text{int}}(x_n, z_n, u_n), \; n = 0, \dots, N-1.$$

- ϕ_f - state transition - compute next integration step
- ϕ_{int} - internal computations, e.g., stages of a Runge-Kutta method (next section)
- z_n collects all interval variables of the integration method

Single step numerical simulation as discrete time system

Single step abstract integration method

$$x_{n+1} = \phi_f(x_n, z_n, u_n),$$
$$0 = \phi_{\text{int}}(x_n, z_n, u_n), \;\; n = 0, \dots, N-1.$$

- ϕ_f - state transition - compute next integration step
- ϕ_{int} - internal computations, e.g., stages of a Runge-Kutta method (next section)
- z_n collects all interval variables of the integration method

Example (Explicit Euler):

$$x_{n+1} = x_n + h z_n,$$
$$0 = f(x_n, u_n) - z_n.$$

Single step numerical simulation as discrete time system

Single step abstract integration method

$$x_{n+1} = \phi_f(x_n, z_n, u_n),$$
$$0 = \phi_{\text{int}}(x_n, z_n, u_n), \; n = 0, \ldots, N-1.$$

- ϕ_f - state transition - compute next integration step
- ϕ_{int} - internal computations, e.g., stages of a Runge-Kutta method (next section)
- z_n collects all interval variables of the integration method

Example (Explicit Euler):

$$x_{n+1} = x_n + h z_n,$$
$$0 = f(x_n, u_n) - z_n.$$

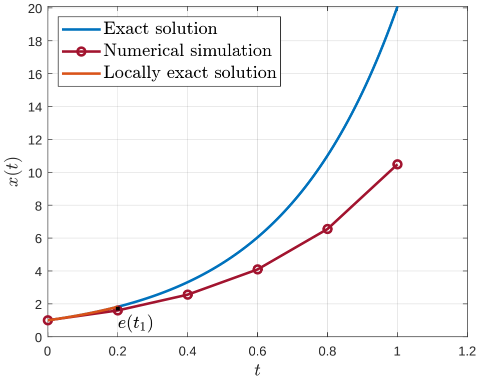
Is an overkill for simple examples but pays off for complicated methods later.

Integration error

Local and global error

- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi_f(x(t_n), z_n, u_0)\|.$$

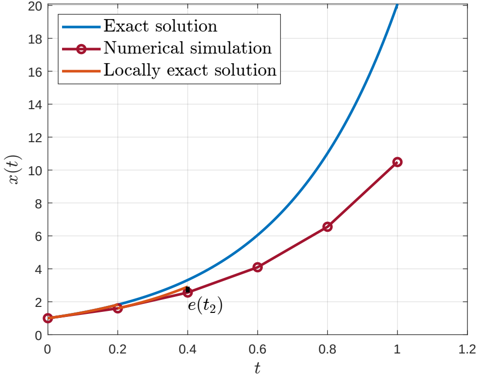


Integration error

Local and global error

- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi_f(x(t_n), z_n, u_0)\|.$$

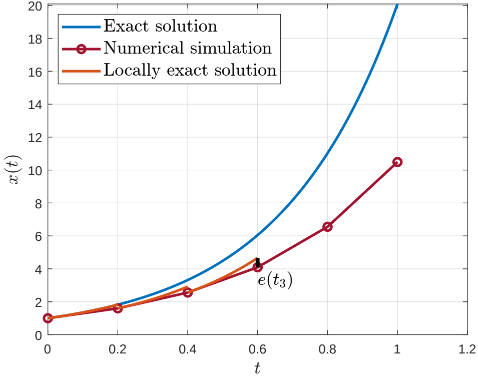


Integration error

Local and global error

- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi_f(x(t_n), z_n, u_0)\|.$$

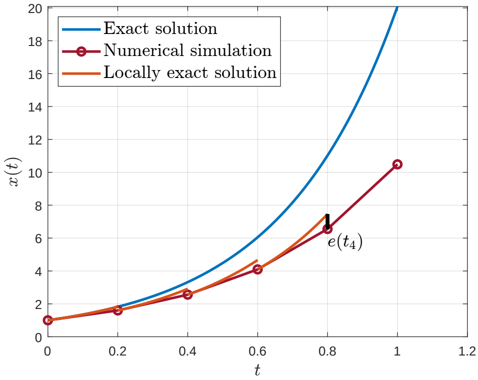


Integration error

Local and global error

- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi_f(x(t_n), z_n, u_0)\|.$$



Integration error

Local and global error

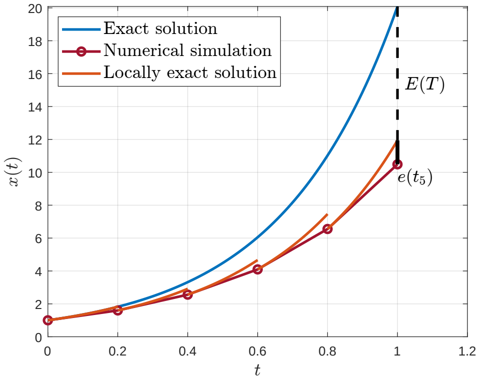
- Local integration error at t_{n+1}:

$$e(t_{n+1}) = \|x(t_{n+1}) - \phi_f(x(t_n), z_n, u_0)\|.$$

- Global integration error at $t = T$:

$$E(T) = \|x(T) - x_N\|.$$

- Global error - accumulation of local errors



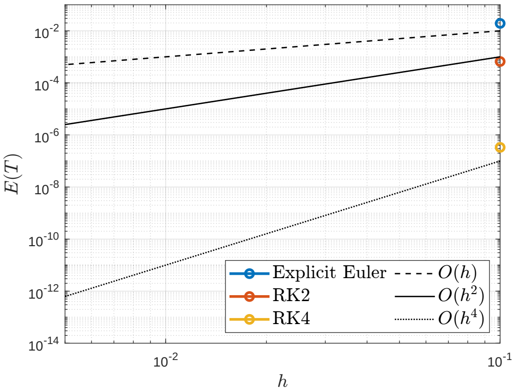
Integrator convergence and accuracy

- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$



- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy

Convergence and integrator order

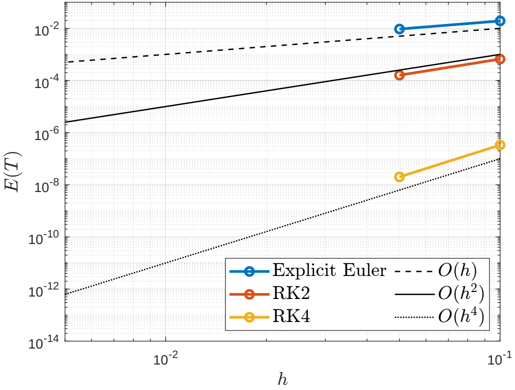
Integrator convergence and accuracy

- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$



- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy

Convergence and integrator order

Integrator convergence and accuracy

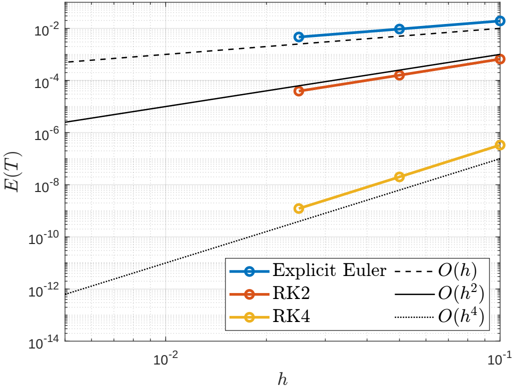
- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy



Convergence and integrator order

Integrator convergence and accuracy

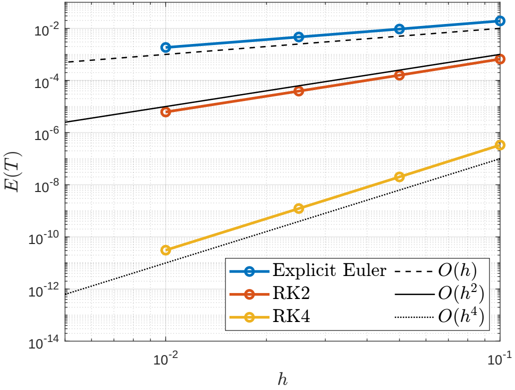
- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq Ch^{p+1} = O(h^{p+1}), C > 0$$

- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy



Convergence and integrator order

Integrator convergence and accuracy

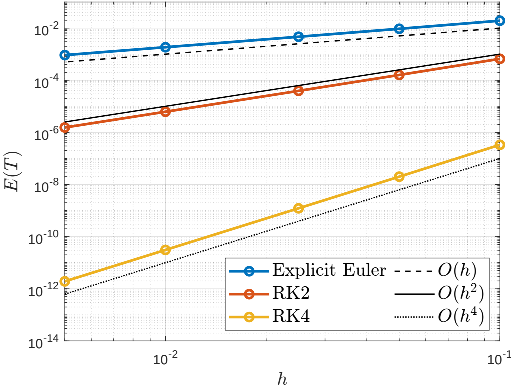
- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy



Convergence and integrator order

Integrator convergence and accuracy

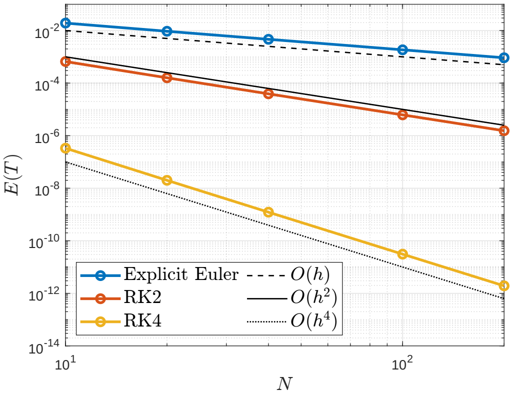
- Convergence

$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Higher order p:
  - less, but more expensive steps for same accuracy
  - in total fewer r.h.s. evaluations for same accuracy



Alternatively one can plot the error over $N \propto \frac{1}{h}$ instead of h

Stability and convergence

Integrator convergence and accuracy

- Convergence

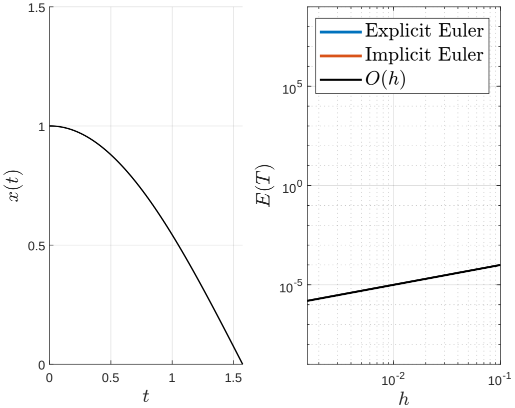
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \ t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

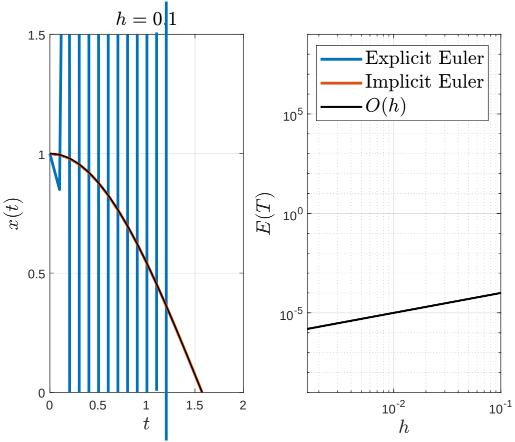
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \ \ t \in [0, 2]$$

$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

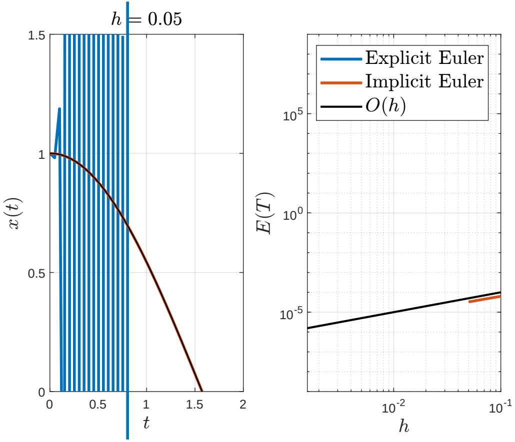
$$\lim_{h\to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h\to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)),\ \ t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

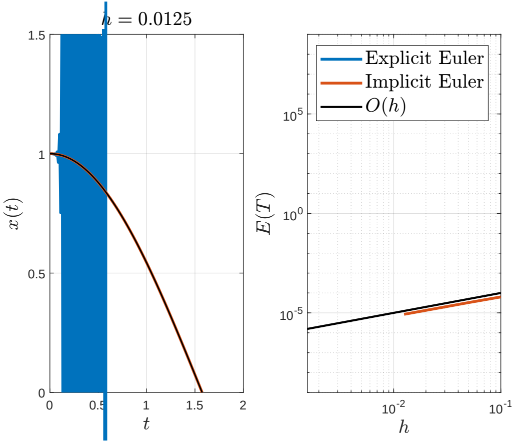
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \ t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

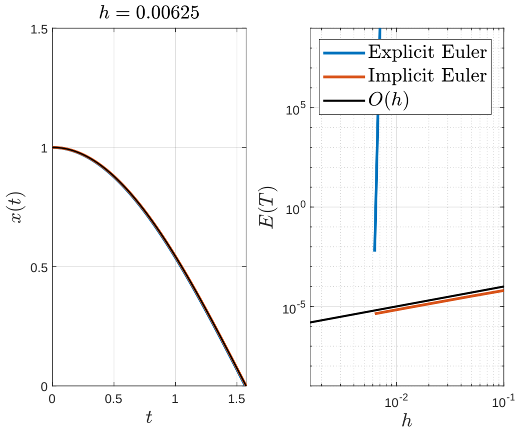
$$\lim_{h \to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h \to 0} e(t_i) \leq C h^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)), \; t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

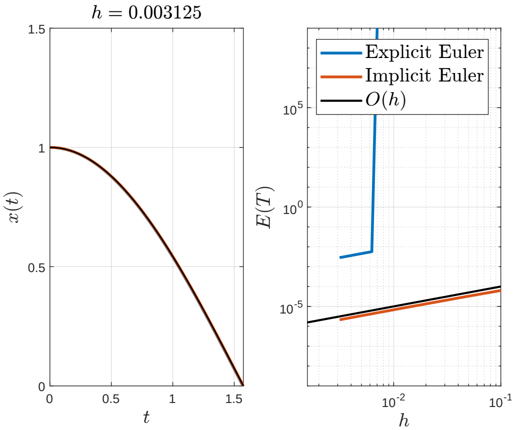
$$\lim_{h\to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h\to 0} e(t_i) \leq Ch^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)),\ t \in [0, 2]$$
$$x(0) = 1$$

Stability and convergence

Integrator convergence and accuracy

- Convergence

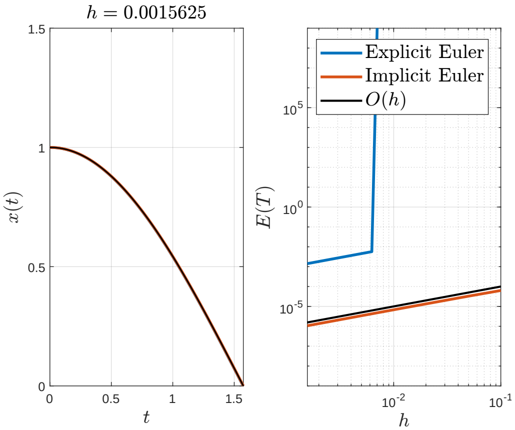
$$\lim_{h\to 0} E(T) = 0$$

- Integrator has order p if

$$\lim_{h\to 0} e(t_i) \leq Ch^{p+1} = O(h^{p+1}), C > 0$$

- Stability: damping of errors, does it work for $h \gg 0$?

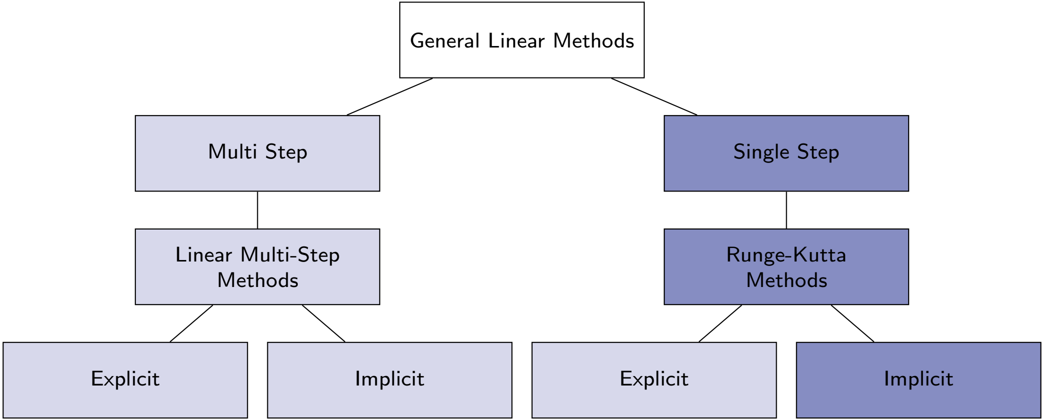
- If integrator is unstable, it does not converge and has $p = 0$, unless h very small



$$\dot{x}(t) = -300(x(t) - \cos(t)),\ t \in [0, 2]$$
$$x(0) = 1$$

Outline of the lecture

Classes of numerical simulation methods



Runge-Kutta method definition

Unknowns are derivatives at stage points

Definition (Runge-Kutta method in differential form)

Let n_s be the number of stages. Given the matrix $A \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{s}}$ with the entries $a_{i,j}$ for $i, j = 1, \ldots, n_\mathrm{s}$, and the vectors $b, c \in \mathbb{R}^{n_\mathrm{s}}$. Let $t_{n,i} = t_n + c_i h$. The system of equations:

$$k_{n,i} = f(t_{n,i}, x_n + h \sum_{j=1}^{n_\mathrm{s}} a_{i,j} k_{n,j}, u_n),\ i = 1, \ldots, n_\mathrm{s}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} b_i k_{n,i}$$

is called a n_s-stage Runge-Kutta (RK) method in the *differential form*.

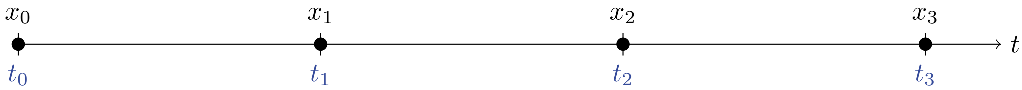
Definition (Runge-Kutta method in differential form)

Let n_s be the number of stages. Given the matrix $A \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{s}}$ with the entries $a_{i,j}$ for $i, j = 1, \ldots, n_\mathrm{s}$, and the vectors $b, c \in \mathbb{R}^{n_\mathrm{s}}$. Let $t_{n,i} = t_n + c_i h$. The system of equations:

$$k_{n,i} = f(t_{n,i}, x_n + h \sum_{j=1}^{n_\mathrm{s}} a_{i,j} k_{n,j}, u_n), \; i = 1, \ldots, n_\mathrm{s}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} b_i k_{n,i}$$

is called a n_s-stage Runge-Kutta (RK) method in the *differential form*.



Runge-Kutta method definition

Unknowns are derivatives at stage points

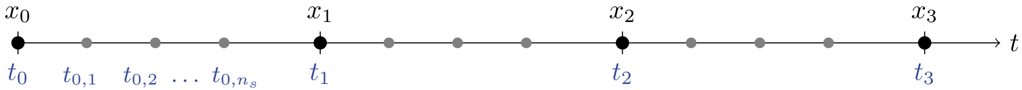
Definition (Runge-Kutta method in differential form)

Let n_s be the number of stages. Given the matrix $A \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{s}}$ with the entries $a_{i,j}$ for $i, j = 1, \ldots, n_\mathrm{s}$, and the vectors $b, c \in \mathbb{R}^{n_\mathrm{s}}$. Let $t_{n,i} = t_n + c_i h$. The system of equations:

$$k_{n,i} = f(t_{n,i}, x_n + h \sum_{j=1}^{n_\mathrm{s}} a_{i,j} k_{n,j}, u_n), \; i = 1, \ldots, n_\mathrm{s}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} b_i k_{n,i}$$

is called a n_s-stage Runge-Kutta (RK) method in the *differential form*.



Runge-Kutta method definition

Unknowns are derivatives at stage points

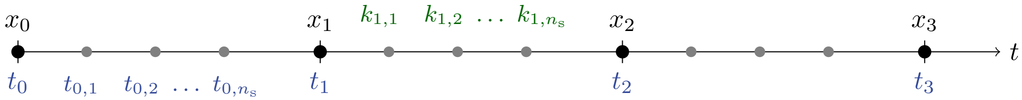
Definition (Runge-Kutta method in differential form)

Let n_s be the number of stages. Given the matrix $A \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{s}}$ with the entries $a_{i,j}$ for $i, j = 1, \ldots, n_\mathrm{s}$, and the vectors $b, c \in \mathbb{R}^{n_\mathrm{s}}$. Let $t_{n,i} = t_n + c_i h$. The system of equations:

$$k_{n,i} = f(t_{n,i}, x_n + h \sum_{j=1}^{n_\mathrm{s}} a_{i,j} k_{n,j}, u_n), \; i = 1, \ldots, n_\mathrm{s}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} b_i k_{n,i}$$

is called a n_s-stage Runge-Kutta (RK) method in the *differential form*.



Runge-Kutta method definition

Unknowns are derivatives at stage points

Definition (Runge-Kutta method in differential form)

Let n_s be the number of stages. Given the matrix $A \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{s}}$ with the entries $a_{i,j}$ for $i, j = 1, \dots, n_\mathrm{s}$, and the vectors $b, c \in \mathbb{R}^{n_\mathrm{s}}$. Let $t_{n,i} = t_n + c_i h$. The system of equations:

$$k_{n,i} = f(t_{n,i}, x_n + h \sum_{j=1}^{n_\mathrm{s}} a_{i,j} k_{n,j}, u_n), \; i = 1, \dots, n_\mathrm{s}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} b_i k_{n,i}$$

is called a n_s-stage Runge-Kutta (RK) method in the *differential form*.

Time grid	**Butcher tableau**	**Data**	**Variables**
h, t_n, $t_{n,i}$	$a_{i,j}$, b_i, c_i	x_n, u_n, $f(\cdot)$	x_{n+1}, $k_{n,i}$
$i = 1, \dots, n_\mathrm{s}$	$i, j = 1, \dots, n_\mathrm{s}$		$i = 1, \dots, n_\mathrm{s}$

Runge-Kutta method definition

Unknowns are states at stage points, cannot treat case of $c_1 = 0$

Definition (Runge-Kutta method in integral form)

Let n_s be the number of stages. Given the matrix $A \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{s}}$ with the entries $a_{i,j}$ for $i, j = 1, \ldots, n_\mathrm{s}$, and the vectors $b, c \in \mathbb{R}^{n_\mathrm{s}}$. Let $t_{n,i} = t_n + c_i h$. The system of equations:

$$x_{n,i} = x_n + h \sum_{j=1}^{n_\mathrm{s}} a_{i,j} f(t_{n,i}, x_{n,j}, u_n), \; i = 1, \ldots, n_\mathrm{s}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} b_i f(t_{n,i}, x_{n,i}, u_n),$$

is called a n_s-stage Runge-Kutta (RK) method in *integral form.*

Time grid	**Butcher tableau**	**Data**	**Variables**
h, t_n, $t_{n,i}$	$a_{i,j}$, b_i, c_i	x_n, u_n, $f(\cdot)$	x_{n+1}, $x_{n,i}$
$i = 1, \ldots, n_\mathrm{s}$	$i, j = 1, \ldots, n_\mathrm{s}$		$i = 1, \ldots, n_\mathrm{s}$

Runge-Kutta method definition

Unknowns are states at stage points, cannot treat case of $c_1 = 0$

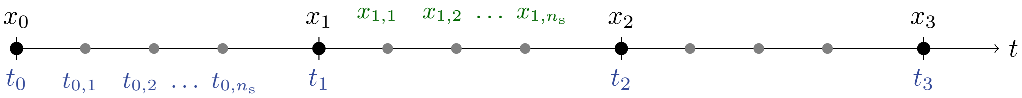
Definition (Runge-Kutta method in integral form)

Let n_s be the number of stages. Given the matrix $A \in \mathbb{R}^{n_\mathrm{s} \times n_\mathrm{s}}$ with the entries $a_{i,j}$ for $i, j = 1, \dots, n_\mathrm{s}$, and the vectors $b, c \in \mathbb{R}^{n_\mathrm{s}}$. Let $t_{n,i} = t_n + c_i h$. The system of equations:

$$x_{n,i} = x_n + h \sum_{j=1}^{n_\mathrm{s}} a_{i,j} f(t_{n,i}, x_{n,j}, u_n), \ i = 1, \dots, n_\mathrm{s}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} b_i f(t_{n,i}, x_{n,i}, u_n),$$

is called a n_s-stage Runge-Kutta (RK) method in *integral form*.



Runge-Kutta method examples

Explicit Runge-Kutta 4

$$
\begin{aligned}
k_{n,1} &= f(t_n, x_n) \\
k_{n,2} &= f(t_n + \frac{h}{2}, x_n + h\frac{k_{n,1}}{2}) \\
k_{n,3} &= f(t_n + \frac{h}{2}, x_n + h\frac{k_{n,2}}{2}) \\
k_{n,5} &= f(t_n + h, x_n + hk_{n,3}) \\
x_{n+1} &= x_n + h(\frac{1}{6}k_{n,1} + \frac{2}{6}k_{n,2} + \frac{2}{6}k_{n,3} + \frac{1}{6}k_{n,4})
\end{aligned}
$$

- All $k_{n,i}$ can be found by explicit function evaluations.

Runge-Kutta method examples

Explicit Runge-Kutta 4

$$
\begin{aligned}
k_{n,1} &= f(t_n, x_n) \\
k_{n,2} &= f(t_n + \frac{h}{2}, x_n + h\frac{k_{n,1}}{2}) \\
k_{n,3} &= f(t_n + \frac{h}{2}, x_n + h\frac{k_{n,2}}{2}) \\
k_{n,5} &= f(t_n + h, x_n + hk_{n,3}) \\
x_{n+1} &= x_n + h(\frac{1}{6}k_{n,1} + \frac{2}{6}k_{n,2} + \frac{2}{6}k_{n,3} + \frac{1}{6}k_{n,4})
\end{aligned}
$$

- All $k_{n,i}$ can be found by explicit function evaluations.

Implicit Euler Method

$$
\begin{aligned}
k_{n,1} &= f(t_n, x_n + hk_{n,1}) \\
x_{n+1} &= x_n + hk_{n,1}
\end{aligned}
$$

- All $k_{n,1}$ is found implicitly by solving $k_{n,1} - f(t_n, x_n + hk_{n,1}) = 0.$

Explicit vs implicit Runge-Kutta methods

The Butcher tableau

Explicit Runge-Kutta method

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{2,1} & & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_{n_\mathrm{s}} & a_{n_\mathrm{s},1} & a_{n_\mathrm{s},2} & \dots & a_{n_\mathrm{s},n_\mathrm{s}-1} & \\
\hline
 & b_1 & b_2 & \dots & b_{n_\mathrm{s}-1} & b_{n_\mathrm{s}}
\end{array}
$$

Explicit vs implicit Runge-Kutta methods

The Butcher tableau

Explicit Runge-Kutta method

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{2,1} & & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_{n_\mathrm{s}} & a_{n_\mathrm{s},1} & a_{n_\mathrm{s},2} & \dots & a_{n_\mathrm{s},n_\mathrm{s}-1} & \\
\hline
 & b_1 & b_2 & \dots & b_{n_\mathrm{s}-1} & b_{n_\mathrm{s}}
\end{array}
$$

- $a_{i,j} \neq 0$ only for $j < i$
- Explicit function evaluations to compute stage values and next step
- Computationally cheap
- Order: $p = n_\mathrm{s}$ if $n_\mathrm{s} \leq 4$ and $p < n_\mathrm{s}$ otherwise

Explicit vs implicit Runge-Kutta methods

The Butcher tableau

Explicit Runge-Kutta method

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{2,1} & & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_{n_\mathrm{s}} & a_{n_\mathrm{s},1} & a_{n_\mathrm{s},2} & \cdots & a_{n_\mathrm{s},n_\mathrm{s}-1} & \\ \hline & b_1 & b_2 & \cdots & b_{n_\mathrm{s}-1} & b_{n_\mathrm{s}} \end{array}$$

Implicit Runge-Kutta method

$$\begin{array}{c|ccccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,n_\mathrm{s}-1} & a_{1,n_\mathrm{s}} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,n_\mathrm{s}-1} & a_{2,n_\mathrm{s}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{n_\mathrm{s}} & a_{n_\mathrm{s},1} & a_{n_\mathrm{s},2} & \cdots & a_{n_\mathrm{s},n_\mathrm{s}-1} & a_{n_\mathrm{s},n_\mathrm{s}} \\ \hline & b_1 & b_2 & \cdots & b_{n_\mathrm{s}-1} & b_{n_\mathrm{s}} \end{array}$$

- $a_{i,j} \neq 0$ only for $j < i$
- Explicit function evaluations to compute stage values and next step
- Computationally cheap
- Order: $p = n_\mathrm{s}$ if $n_\mathrm{s} \leq 4$ and $p < n_\mathrm{s}$ otherwise

Explicit vs implicit Runge-Kutta methods

The Butcher tableau

Explicit Runge-Kutta method

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{2,1} & & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_{n_s} & a_{n_s,1} & a_{n_s,2} & \dots & a_{n_s,n_s-1} & \\
\hline
 & b_1 & b_2 & \dots & b_{n_s-1} & b_{n_s}
\end{array}
$$

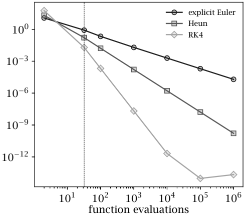
- $a_{i,j} \neq 0$ only for $j < i$
- Explicit function evaluations to compute stage values and next step
- Computationally cheap
- Order: $p = n_s$ if $n_s \leq 4$ and $p < n_s$ otherwise

Implicit Runge-Kutta method

$$
\begin{array}{c|ccccc}
c_1 & a_{1,1} & a_{1,2} & \dots & a_{1,n_s-1} & a_{1,n_s} \\
c_2 & a_{2,1} & a_{2,2} & \dots & a_{2,n_s-1} & a_{2,n_s} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
c_{n_s} & a_{n_s,1} & a_{n_s,2} & \dots & a_{n_s,n_s-1} & a_{n_s,n_s} \\
\hline
 & b_1 & b_2 & \dots & b_{n_s-1} & b_{n_s}
\end{array}
$$

- Requires solving nonlinear rootfinding problem with Newton's method
- Expensive but good for stiff systems
- Order: $p = 2n_s$, $p = 2n_s - 1$, ...
- Famous representative: collocation methods - treated next!

Butcher tableau, six examples



Euler

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

Heun

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

RK4

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}$$

Implicit Euler

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Midpoint rule (GL2)

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$$

Gauss-Legendre of order 4 (GL4)

$$\begin{array}{c|cc} 1/2-\sqrt{3}/6 & 1/4 & 1/4-\sqrt{3}/6 \\ 1/2+\sqrt{3}/6 & 1/4+\sqrt{3}/6 & 1/4 \\ \hline & 1/2 & 1/2 \end{array}$$

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ c_2 & a_{21} & \cdots & a_{2s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

Outline of the lecture

Collocation

Main ideas:

- Approximate $x(t)$ on $t \in [t_n, t_{n+1}]$ with a **polynomial** $q_n(t)$ of degree n_s

Collocation

Main ideas:

- Approximate $x(t)$ on $t \in [t_n, t_{n+1}]$ with a **polynomial** $q_n(t)$ of degree n_s
- Pick n_s distinct numbers: $0 \leq c_1 < c_2 < \ldots < c_{n_\mathrm{s}} \leq 1$

Collocation

Main ideas:

- Approximate $x(t)$ on $t \in [t_n, t_{n+1}]$ with a **polynomial** $q_n(t)$ of degree n_s
- Pick n_s distinct numbers: $0 \leq c_1 < c_2 < \ldots < c_{n_\mathrm{s}} \leq 1$
- Define **collocation points** $t_{n,i} = t_n + c_i h \ , \ i = 1, \ldots, n_\mathrm{s}$

Collocation

Main ideas:

- Approximate $x(t)$ on $t \in [t_n, t_{n+1}]$ with a **polynomial** $q_n(t)$ of degree n_s
- Pick n_s distinct numbers: $0 \leq c_1 < c_2 < \ldots < c_{n_\mathrm{s}} \leq 1$
- Define **collocation points** $t_{n,i} = t_n + c_i h \;, \; i = 1, \ldots, n_\mathrm{s}$
- The polynomial $q_n(t) \approx x(t)$ satisfies the ODE on the collocation points:

Collocation equations

$$\begin{aligned} q_n(t_n) &= x_n \\ \dot{q}_n(t_n + c_i h) &= f(t_n + c_i h, q_n(t_n + c_i h), u_n), \quad i = 1, \ldots, n_\mathrm{s} \end{aligned}$$

Collocation

Main ideas:

- Approximate $x(t)$ on $t \in [t_n, t_{n+1}]$ with a **polynomial** $q_n(t)$ of degree n_s
- Pick n_s distinct numbers: $0 \leq c_1 < c_2 < \ldots < c_{n_\mathrm{s}} \leq 1$
- Define **collocation points** $t_{n,i} = t_n + c_i h \ , \ i = 1, \ldots, n_\mathrm{s}$
- The polynomial $q_n(t) \approx x(t)$ satisfies the ODE on the collocation points:

Collocation equations

$$
\begin{aligned}
q_n(t_n) &= x_n \\
\dot{q}_n(t_n + c_i h) &= f(t_n + c_i h, q_n(t_n + c_i h), u_n), \quad i = 1, \ldots, n_\mathrm{s}
\end{aligned}
$$

- Polynomial of degree n_s: $n_\mathrm{s} + 1$ coefficient and $n_\mathrm{s} + 1$ equations

Collocation

Main ideas:

- Approximate $x(t)$ on $t \in [t_n, t_{n+1}]$ with a **polynomial** $q_n(t)$ of degree n_s
- Pick n_s distinct numbers: $0 \leq c_1 < c_2 < \ldots < c_{n_\mathrm{s}} \leq 1$
- Define **collocation points** $t_{n,i} = t_n + c_i h \ , \ i = 1, \ldots, n_\mathrm{s}$
- The polynomial $q_n(t) \approx x(t)$ satisfies the ODE on the collocation points:

Collocation equations

$$
\begin{aligned}
q_n(t_n) &= x_n \\
\dot{q}_n(t_n + c_i h) &= f(t_n + c_i h, q_n(t_n + c_i h), u_n), \quad i = 1, \ldots, n_\mathrm{s}
\end{aligned}
$$

- Polynomial of degree n_s: $n_\mathrm{s} + 1$ coefficient and $n_\mathrm{s} + 1$ equations
- Next value - simple evaluation: $x_{n+1} = q_n(t_{n+1})$

Collocation - how to implement it?

How to parameterize $q_n(t)$?

Two common (equivalent) choices

Collocation - how to implement it?

How to parameterize $q_n(t)$?

Two common (equivalent) choices

1. Find interpolating polynomial $q_n(t)$ through x_n (at t_n) and **state values** $x_{n,1}, \ldots, x_{n,n_\mathrm{s}}$ at collocation points $t_{n,i},\ i = 1, \ldots, n_\mathrm{s}$ (in Exercise 1).

Collocation - how to implement it?

How to parameterize $q_n(t)$?

Two common (equivalent) choices

1. Find interpolating polynomial $q_n(t)$ through x_n (at t_n) and **state values** $x_{n,1}, \dots, x_{n,n_\mathrm{s}}$ at collocation points $t_{n,i},\ i = 1, \dots, n_\mathrm{s}$ (in Exercise 1).
2. Find $\dot{q}_n(t)$ interpolating polynomial through **state derivatives** $k_{n,1}, \dots, k_{n,n_\mathrm{s}}$ at collocation points $t_{n,i},\ i = 1, \dots, n_\mathrm{s}$ (**this lecture**).

Collocation - how to implement it?

How to parameterize $q_n(t)$?

Two common (equivalent) choices

1. Find interpolating polynomial $q_n(t)$ through x_n (at t_n) and **state values** $x_{n,1}, \dots, x_{n,n_\mathrm{s}}$ at collocation points $t_{n,i},\ i = 1, \dots, n_\mathrm{s}$ (in Exercise 1).
2. Find $\dot{q}_n(t)$ interpolating polynomial through **state derivatives** $k_{n,1}, \dots, k_{n,n_\mathrm{s}}$ at collocation points $t_{n,i},\ i = 1, \dots, n_\mathrm{s}$ (**this lecture**).

- $q_n(t)$ is recovered via:

$$q_n(t) = x_n + \int_{t_n}^{t} \dot{q}_n(\tau; k_{n,1}, \dots, k_{n,n_\mathrm{s}}) \mathrm{d}\tau.$$

Collocation - how to implement it?

How to parameterize $q_n(t)$?

Two common (equivalent) choices

1. Find interpolating polynomial $q_n(t)$ through x_n (at t_n) and **state values** $x_{n,1}, \dots, x_{n,n_\mathrm{s}}$ at collocation points $t_{n,i},\ i = 1, \dots, n_\mathrm{s}$ (in Exercise 1).
2. Find $\dot{q}_n(t)$ interpolating polynomial through **state derivatives** $k_{n,1}, \dots, k_{n,n_\mathrm{s}}$ at collocation points $t_{n,i},\ i = 1, \dots, n_\mathrm{s}$ (**this lecture**).

- $q_n(t)$ is recovered via:

$$q_n(t) = x_n + \int_{t_n}^{t} \dot{q}_n(\tau; k_{n,1}, \dots, k_{n,n_\mathrm{s}}) \mathrm{d}\tau.$$

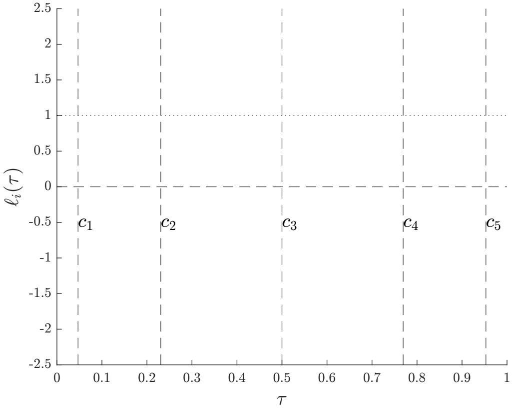
- with:

$$\begin{aligned} \dot{q}_n(t) &= \ell_1\left(\frac{t - t_n}{h}\right) k_{n,1} + \ell_2\left(\frac{t - t_n}{h}\right) k_{n,2} + \dots + \ell_{n_\mathrm{s}}\left(\frac{t - t_n}{h}\right) k_{n,n_\mathrm{s}} \\ &= \sum_{i=1}^{n_\mathrm{s}} \ell_i\left(\frac{t - t_n}{h}\right) \underbrace{f(t_n + c_i, q_n(t_n + c_i h), u_0)}_{= k_{n,i}} \end{aligned}$$

The Lagrange polynomials $\ell_i(\tau)$

Lagrange polynomial basis

$$\ell_i(\tau) = \prod_{j=1, i \neq j}^{n_s} \frac{\tau - c_j}{c_i - c_j}.$$



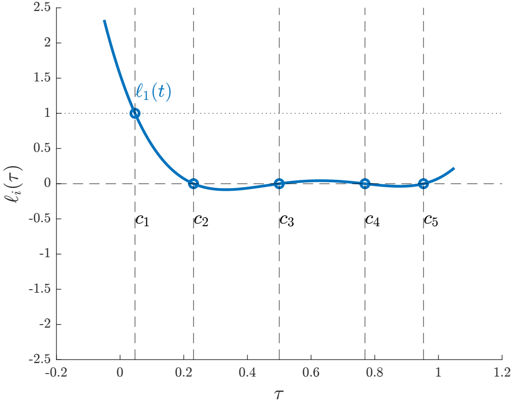
The Lagrange polynomials $\ell_i(\tau)$

Lagrange polynomial basis

$$\ell_i(\tau) = \prod_{j=1, i\neq j}^{n_\mathrm{s}} \frac{\tau - c_j}{c_i - c_j}.$$

Properties:

$$\ell_i(c_j) = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases}$$



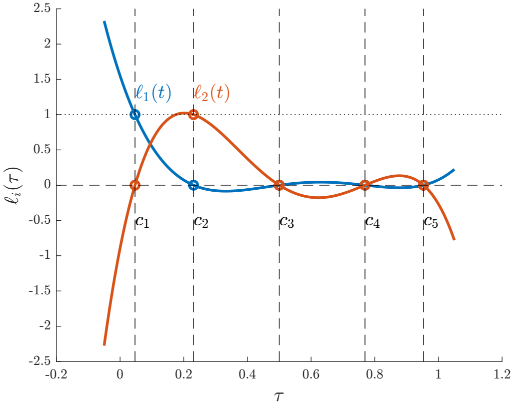
The Lagrange polynomials $\ell_i(\tau)$

Lagrange polynomial basis

$$\ell_i(\tau) = \prod_{j=1, i \neq j}^{n_s} \frac{\tau - c_j}{c_i - c_j}.$$

Properties:

$$\ell_i(c_j) = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases}$$



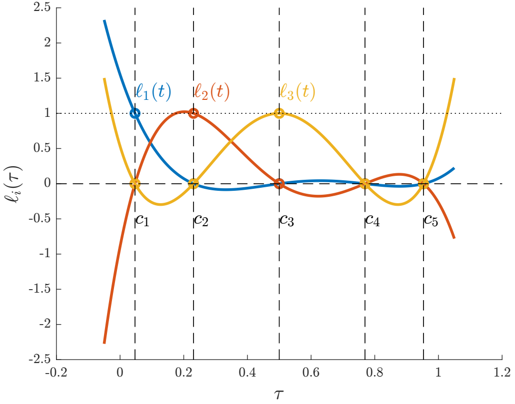
The Lagrange polynomials $\ell_i(\tau)$

Lagrange polynomial basis

$$\ell_i(\tau) = \prod_{j=1, i\neq j}^{n_\mathrm{s}} \frac{\tau - c_j}{c_i - c_j}.$$

Properties:

$$\ell_i(c_j) = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases}$$



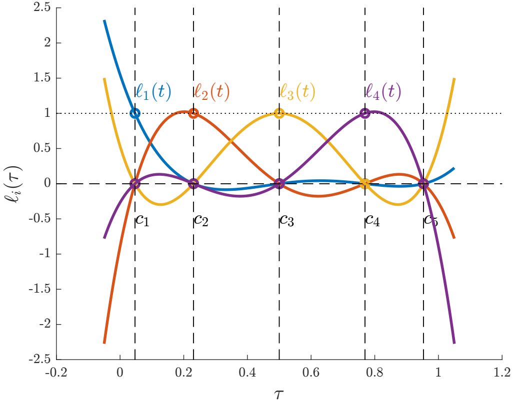
The Lagrange polynomials $\ell_i(\tau)$

Lagrange polynomial basis

$$\ell_i(\tau) = \prod_{j=1, i\neq j}^{n_s} \frac{\tau - c_j}{c_i - c_j}.$$

Properties:

$$\ell_i(c_j) = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases}$$



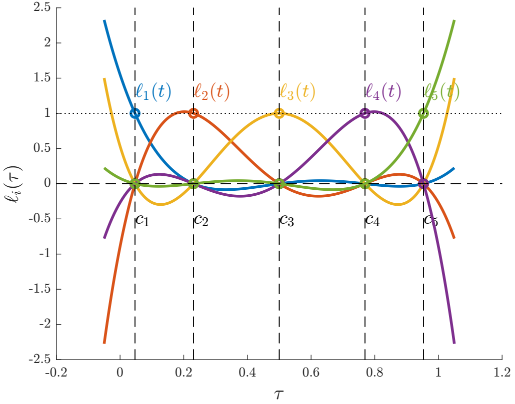
The Lagrange polynomials $\ell_i(\tau)$

Lagrange polynomial basis

$$\ell_i(\tau) = \prod_{j=1, i \neq j}^{n_s} \frac{\tau - c_j}{c_i - c_j}.$$

Properties:

$$\ell_i(c_j) = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases}$$



The Lagrange polynomials $\ell_i(\tau)$

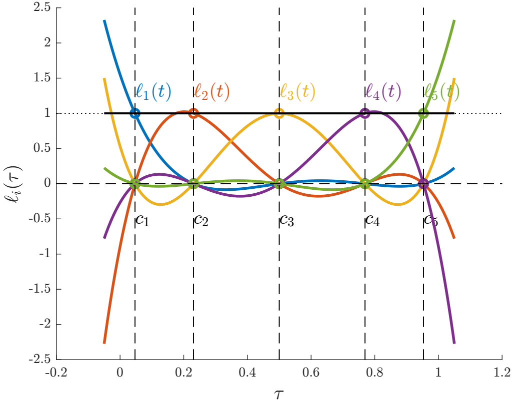
Lagrange polynomial basis

$$\ell_i(\tau) = \prod_{j=1, i\neq j}^{n_\mathrm{s}} \frac{\tau - c_j}{c_i - c_j}.$$

Properties:

$$\ell_i(c_j) = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases}$$

$$\sum_{i=1}^{n_\mathrm{s}} \ell_i(t) = 1$$



Collocation - how to implement it - continued

- Evaluate $q_n(t)$ at collocation points

$$q_n(t_n + c_i h) = x_n + \int_{t_n}^{t_n + c_i h} \dot{q}_n(\tau; k_{n,1}, \ldots, k_{n,n_\mathrm{s}}) \mathrm{d}\tau$$

Collocation - how to implement it - continued

- Evaluate $q_n(t)$ at collocation points

$$
\begin{aligned}
q_n(t_n + c_i h) &= x_n + \int_{t_n}^{t_n + c_i h} \dot{q}_n(\tau; k_{n,1}, \ldots, k_{n,n_\mathrm{s}}) \mathrm{d}\tau \\
&= x_n + \int_{t_n}^{t_n + c_i h} \sum_{j=1}^{n_\mathrm{s}} \ell_j\left(\frac{\tau - t_n}{h}\right) k_{n,j} \mathrm{d}\tau
\end{aligned}
$$

Collocation - how to implement it - continued

- Evaluate $q_n(t)$ at collocation points

$$
\begin{aligned}
q_n(t_n + c_i h) &= x_n + \int_{t_n}^{t_n + c_i h} \dot{q}_n(\tau; k_{n,1}, \dots, k_{n,n_\mathrm{s}}) \mathrm{d}\tau \\
&= x_n + \int_{t_n}^{t_n + c_i h} \sum_{j=1}^{n_\mathrm{s}} \ell_j \left(\frac{\tau - t_n}{h} \right) k_{n,j} \mathrm{d}\tau \\
&= x_n + h \sum_{j=1}^{n_\mathrm{s}} k_j \underbrace{\int_0^{c_i} \ell_j(\sigma) \mathrm{d}\sigma}_{:= a_{i,j}}
\end{aligned}
$$

Collocation - how to implement it - continued

- Evaluate $q_n(t)$ at collocation points

$$
\begin{aligned}
q_n(t_n + c_i h) &= x_n + \int_{t_n}^{t_n + c_i h} \dot{q}_n(\tau; k_{n,1}, \dots, k_{n,n_\mathrm{s}}) \mathrm{d}\tau \\
&= x_n + \int_{t_n}^{t_n + c_i h} \sum_{j=1}^{n_\mathrm{s}} \ell_j \left(\frac{\tau - t_n}{h} \right) k_{n,j} \mathrm{d}\tau \\
&= x_n + h \sum_{j=1}^{n_\mathrm{s}} k_j \underbrace{\int_0^{c_i} \ell_j(\sigma) \mathrm{d}\sigma}_{:= a_{i,j}} \\
&= x_n + h \sum_{j=1}^{n_\mathrm{s}} k_j a_{i,j}
\end{aligned}
$$

Collocation - how to implement it - continued

- Evaluate $q_n(t)$ at collocation points

$$
\begin{aligned}
q_n(t_n + c_i h) &= x_n + \int_{t_n}^{t_n + c_i h} \dot{q}_n(\tau; k_{n,1}, \dots, k_{n,n_\mathrm{s}}) \mathrm{d}\tau \\
&= x_n + \int_{t_n}^{t_n + c_i h} \sum_{j=1}^{n_\mathrm{s}} \ell_j\left(\frac{\tau - t_n}{h}\right) k_{n,j} \mathrm{d}\tau \\
&= x_n + h \sum_{j=1}^{n_\mathrm{s}} k_j \underbrace{\int_0^{c_i} \ell_j(\sigma) \mathrm{d}\sigma}_{:= a_{i,j}} \\
&= x_n + h \sum_{j=1}^{n_\mathrm{s}} k_j a_{i,j}
\end{aligned}
$$

Similarly $q_n(t)$ evaluated at $t_{n+1} = t_n + h$:

$$
q_n(t_n + h) = x_n + h \sum_{i=1}^{n_\mathrm{s}} k_i \underbrace{\int_0^1 \ell_i(\sigma) \mathrm{d}\sigma}_{:= b_i} = x_n + h \sum_{i=1}^{n_\mathrm{s}} k_i b_i
$$

All collocation methods are implicit Runge-Kuta method

Collocation equations

$$q_n(t_n) = x_n \qquad \text{(initial value)}$$

$$\dot{q}_n(t_n + c_i h) = f(t_n + c_i, q_n(t_n + c_i h), u_n), \quad i = 1, \ldots, n_\mathrm{s} \qquad \text{(stage eqs.)}$$

$$x_{n+1} = q_n(t_{n+1}) \qquad \text{(next value)}$$

All collocation methods are implicit Runge-Kuta method

Collocation equations

$$q_n(t_n) = x_n \qquad \text{(initial value)}$$

$$k_{n,i} = f(t_n + c_i h, q_n(t_n + c_i h), u_n), \quad i = 1, \dots, n_{\mathrm{s}} \qquad \text{(stage eqs.)}$$

$$x_{n+1} = q_n(t_{n+1}) \qquad \text{(next value)}$$

All collocation methods are implicit Runge-Kuta method

Collocation equations

$$q_n(t_n) = x_n \qquad \text{(initial value)}$$

$$k_{n,i} = f(t_n + c_i h, x_n + h \sum_{j=1}^{n_\mathrm{s}} k_j a_{i,j}, u_n), \quad i = 1, \dots, n_\mathrm{s} \qquad \text{(stage eqs.)}$$

$$x_{n+1} = q_n(t_{n+1}) \qquad \text{(next value)}$$

All collocation methods are implicit Runge-Kuta method

Collocation equations

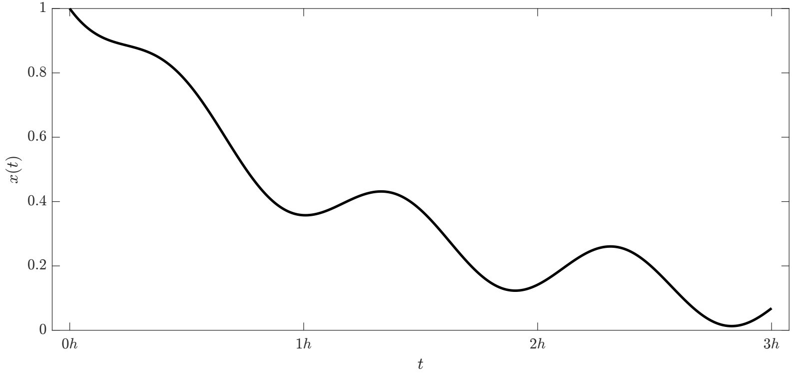
$$q_n(t_n) = x_n \qquad \text{(initial value)}$$

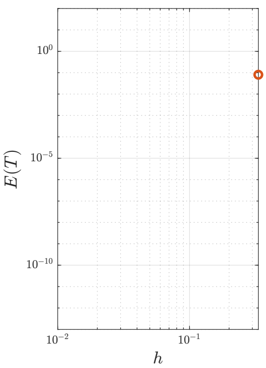
$$k_{n,i} = f(t_n + c_i h, x_n + h \sum_{j=1}^{n_\mathrm{s}} k_j a_{i,j}, u_n), \quad i = 1, \dots, n_\mathrm{s} \qquad \text{(stage eqs.)}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} k_i b_i \qquad \text{(next value)}$$

- We arrived at the implicit RK equations in differential form
- Unknowns: $x_{n+1} \in \mathbb{R}^{n_x}$ and $z_n = (k_{n,1}, \dots, k_{n,n_\mathrm{s}}) \in \mathbb{R}^{n_\mathrm{s} n_x}$
- $(n_\mathrm{s}+1)n_x$ equations and $(n_\mathrm{s}+1)n_x$ variables - solve via Newton's methods

- Choice of points $c_1, \dots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.



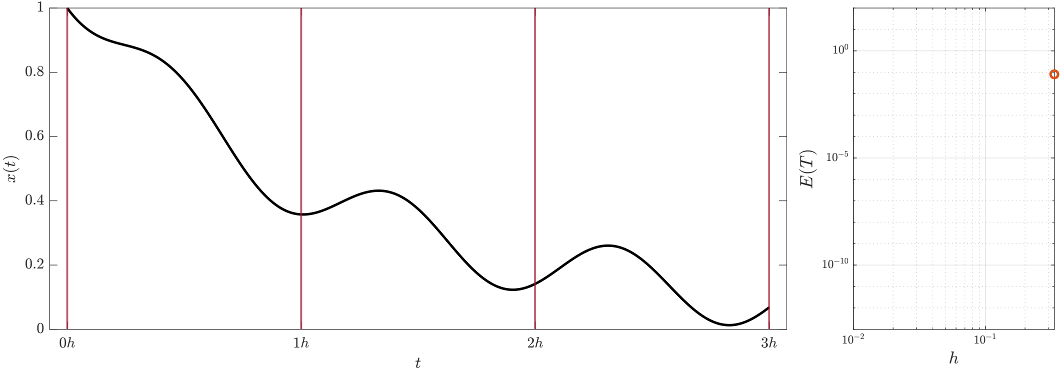


$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$

Visualization inspired by Leo Simpson's talk at the European control conference 2023

Collocation - visualization

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

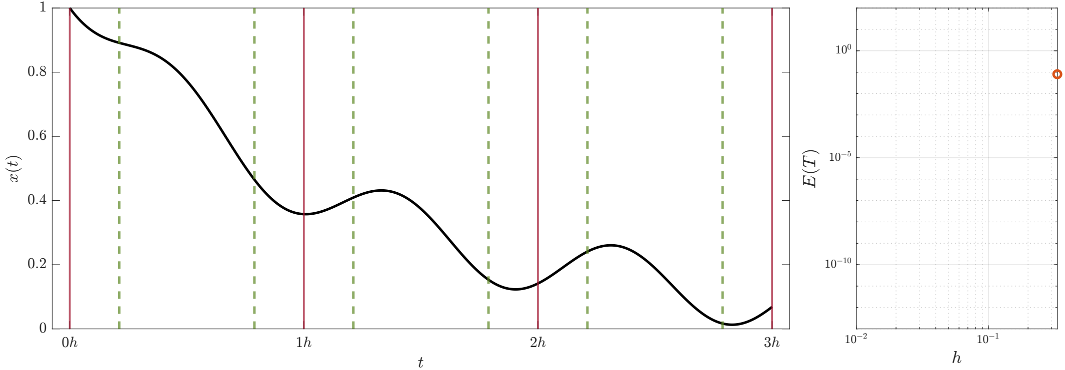


$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$

Visualization inspired by Leo Simpson's talk at the European control conference 2023

Collocation - visualization

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

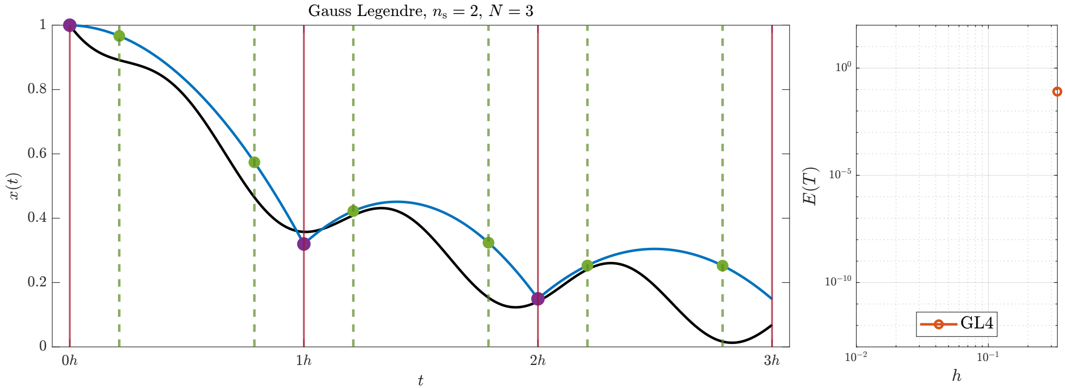


$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Visualization inspired by Leo Simpson's talk at the European control conference 2023

Collocation - visualization

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

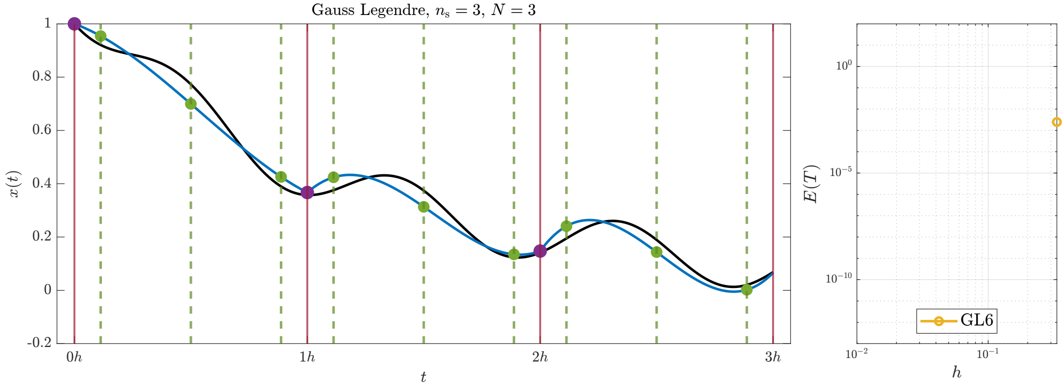


$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Visualization inspired by Leo Simpson's talk at the European control conference 2023

Collocation - visualization

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

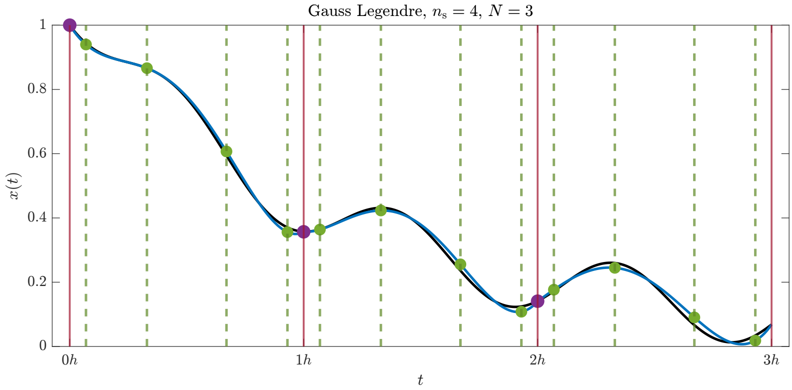


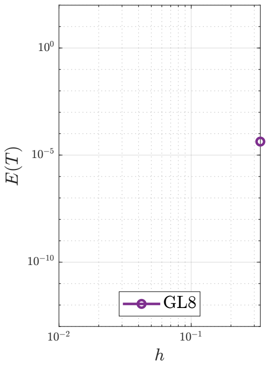
$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Visualization inspired by Leo Simpson's talk at the European control conference 2023

Collocation - visualization

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

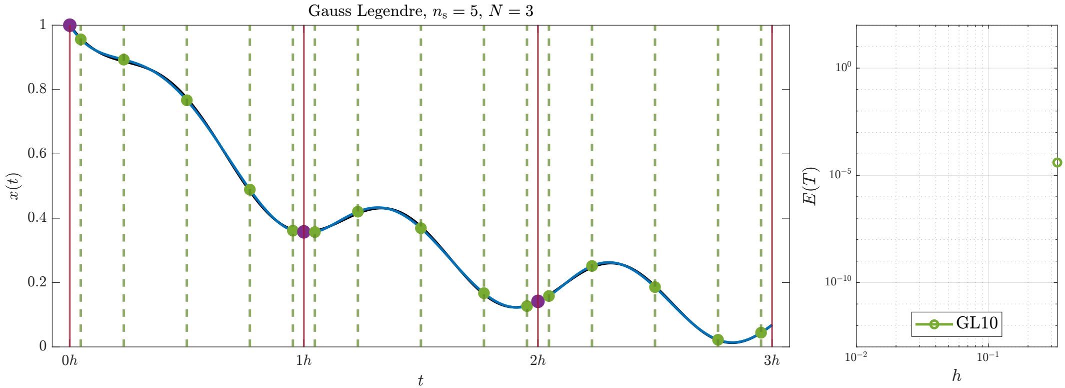




$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

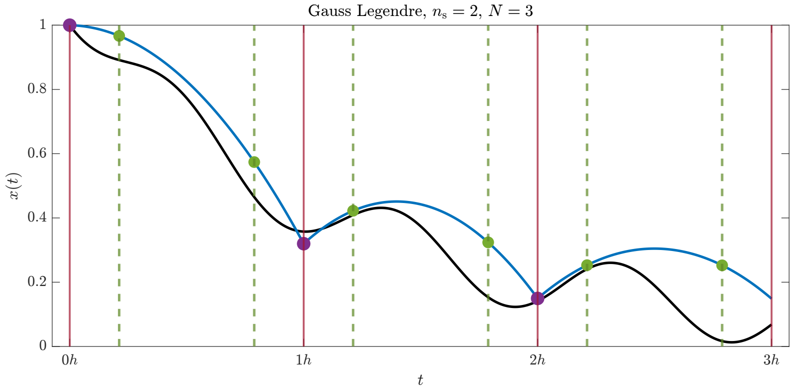
- Choice of points $c_1, \ldots, c_{n_s}$ determines properties of method.
- Gauss-Legendre $p = 2n_s$, Radau-IIA $p = 2n_s - 1$ good for stiff systems, Lobatto family $p = 2n_s - 2$.

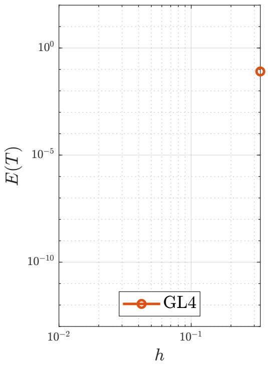


$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

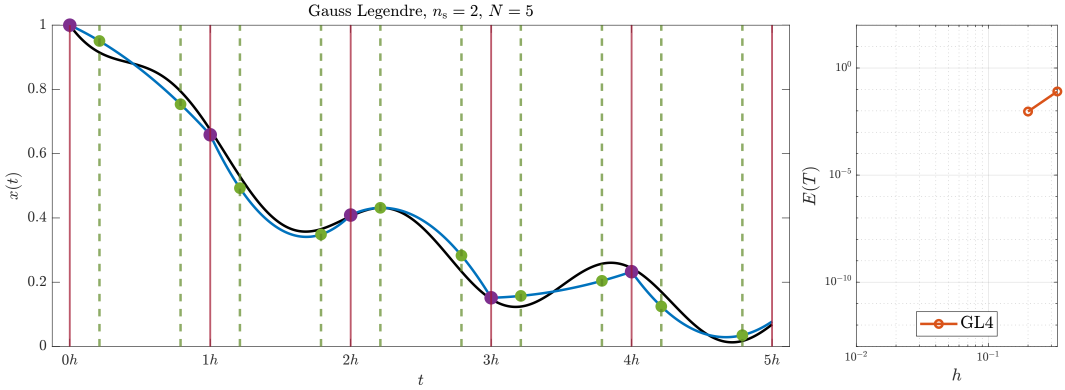




$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

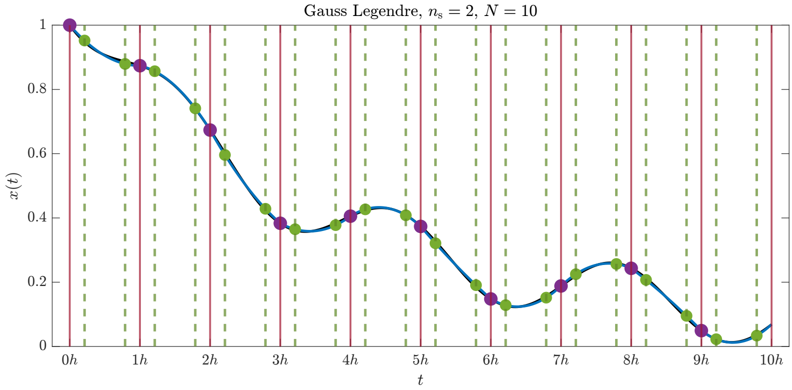
Collocation - visualization

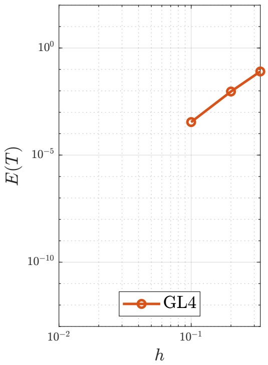
- Choice of points $c_1, \dots, c_{n_s}$ determines properties of method.
- Gauss-Legendre $p = 2n_s$, Radau-IIA $p = 2n_s - 1$ good for stiff systems, Lobatto family $p = 2n_s - 2$.



$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

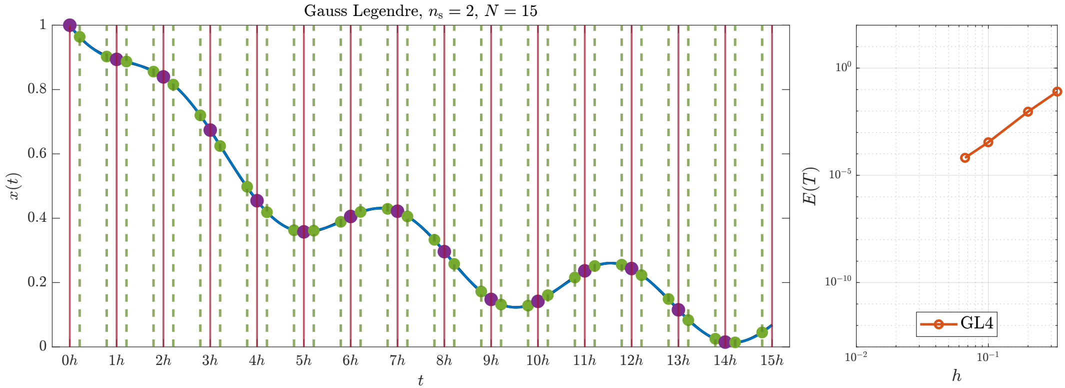
- Choice of points $c_1, \ldots, c_{n_s}$ determines properties of method.
- Gauss-Legendre $p = 2n_s$, Radau-IIA $p = 2n_s - 1$ good for stiff systems, Lobatto family $p = 2n_s - 2$.





$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

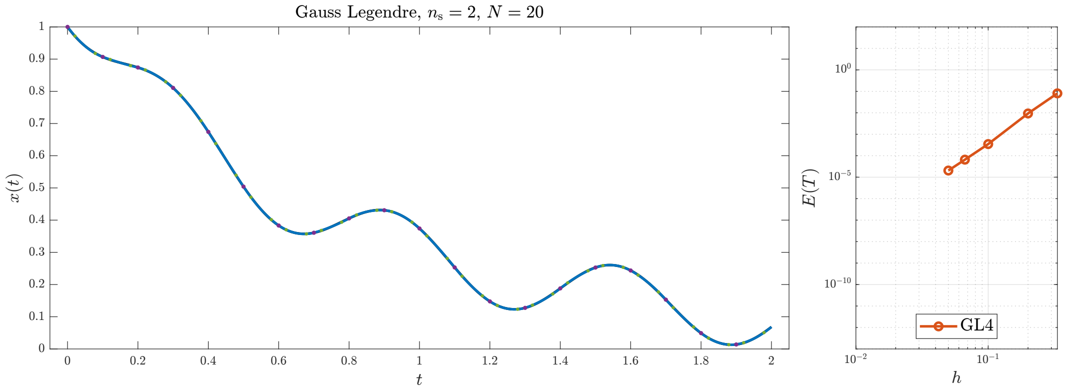
- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.



$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

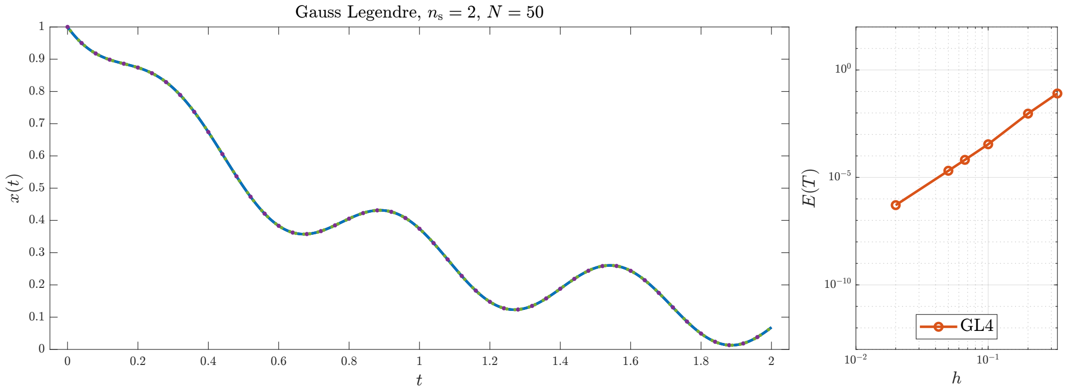
- Choice of points $c_1, \ldots, c_{n_s}$ determines properties of method.
- Gauss-Legendre $p = 2n_s$, Radau-IIA $p = 2n_s - 1$ good for stiff systems, Lobatto family $p = 2n_s - 2$.



$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

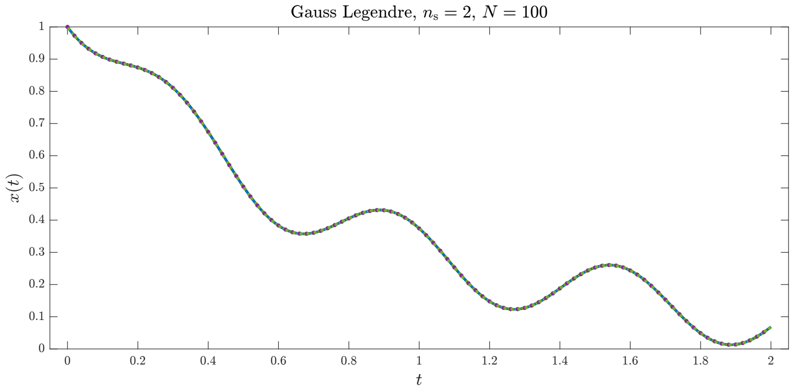
- Choice of points $c_1, \dots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

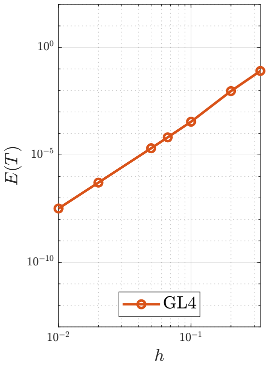


$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

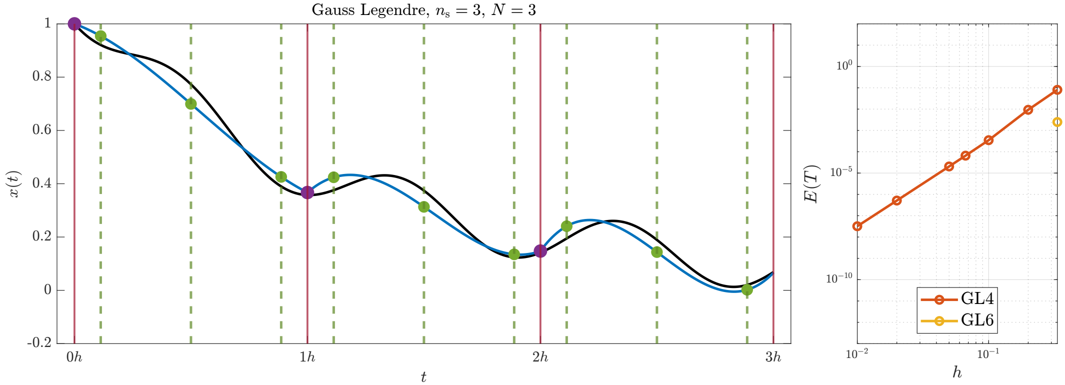




$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

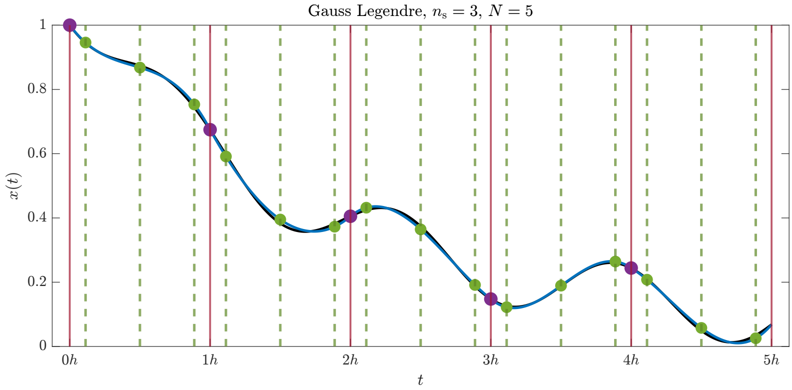
Collocation - visualization

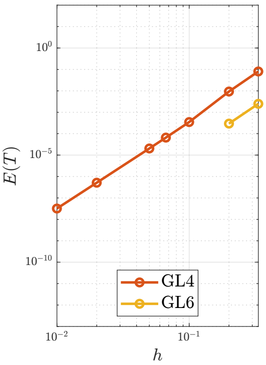
- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.



$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

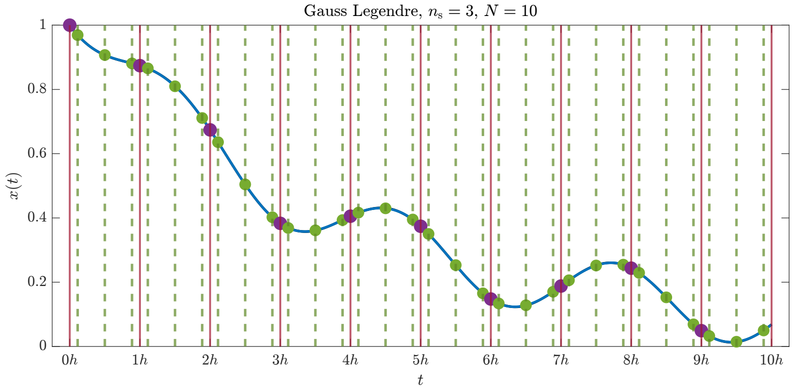


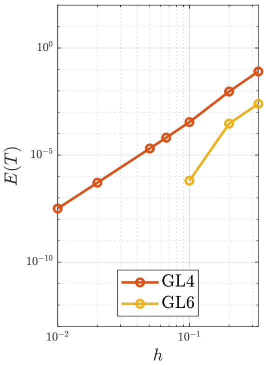


$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

- Choice of points $c_1, \ldots, c_{n_s}$ determines properties of method.
- Gauss-Legendre $p = 2n_s$, Radau-IIA $p = 2n_s - 1$ good for stiff systems, Lobatto family $p = 2n_s - 2$.

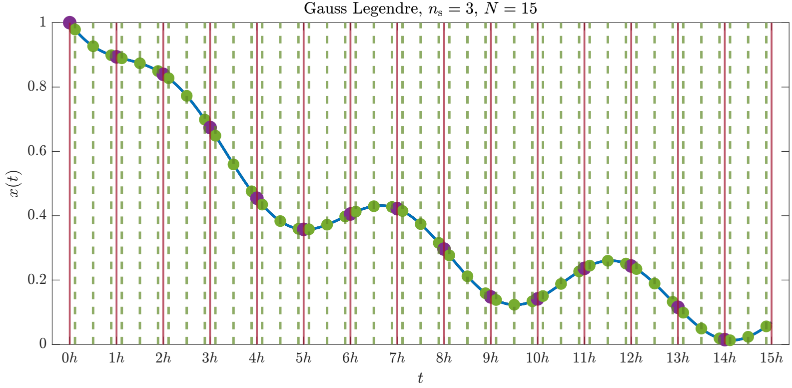


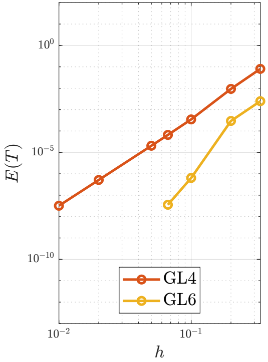


$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

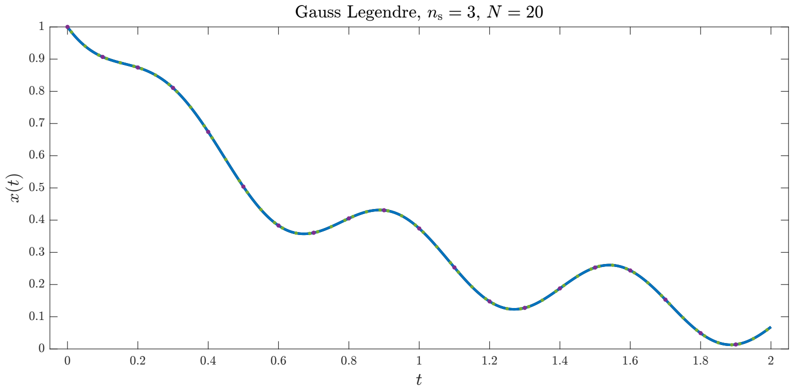
- Choice of points $c_1, \dots, c_{n_s}$ determines properties of method.
- Gauss-Legendre $p = 2n_s$, Radau-IIA $p = 2n_s - 1$ good for stiff systems, Lobatto family $p = 2n_s - 2$.

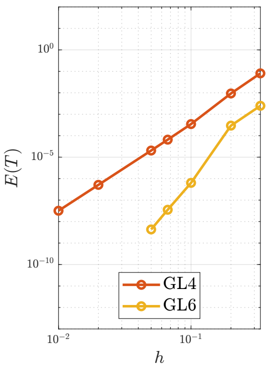




$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

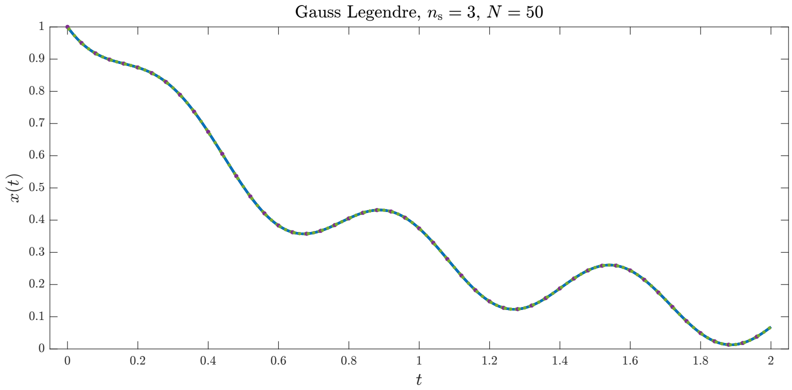


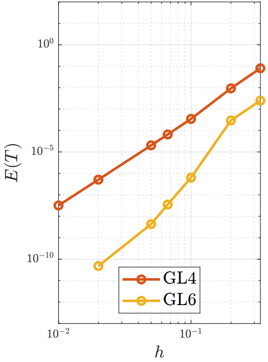


$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

- Choice of points $c_1, \dots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.

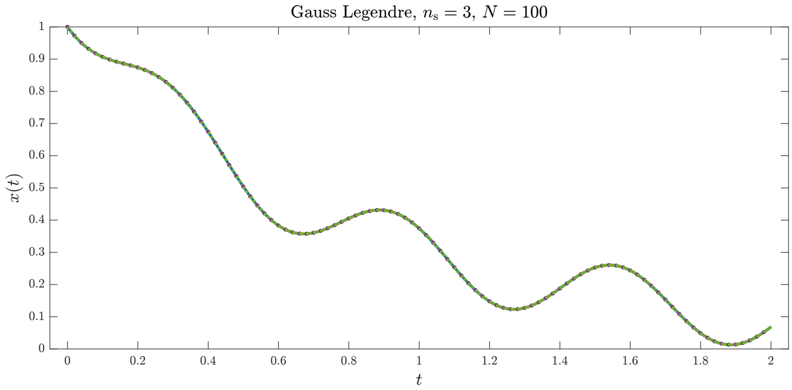


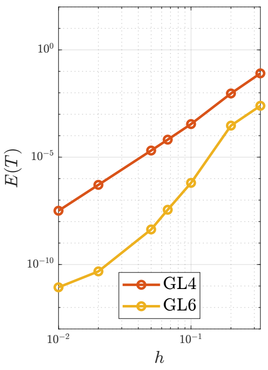


$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Collocation - visualization

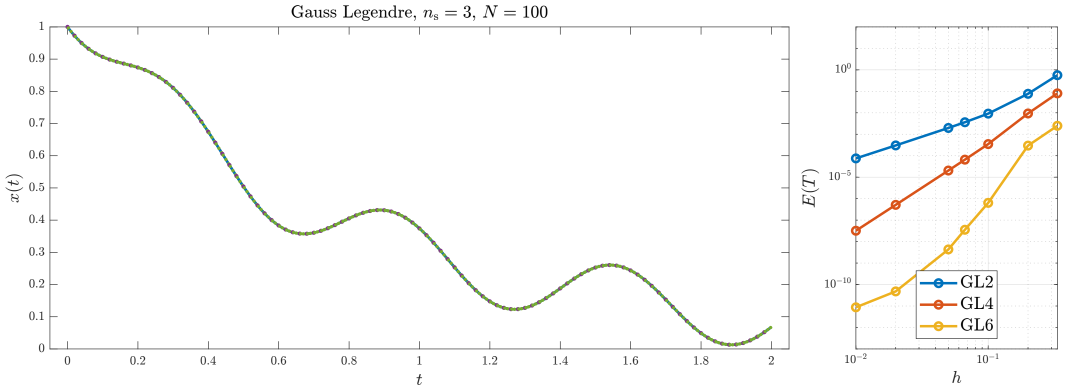
- Choice of points $c_1, \dots, c_{n_s}$ determines properties of method.
- Gauss-Legendre $p = 2n_s$, Radau-IIA $p = 2n_s - 1$ good for stiff systems, Lobatto family $p = 2n_s - 2$.





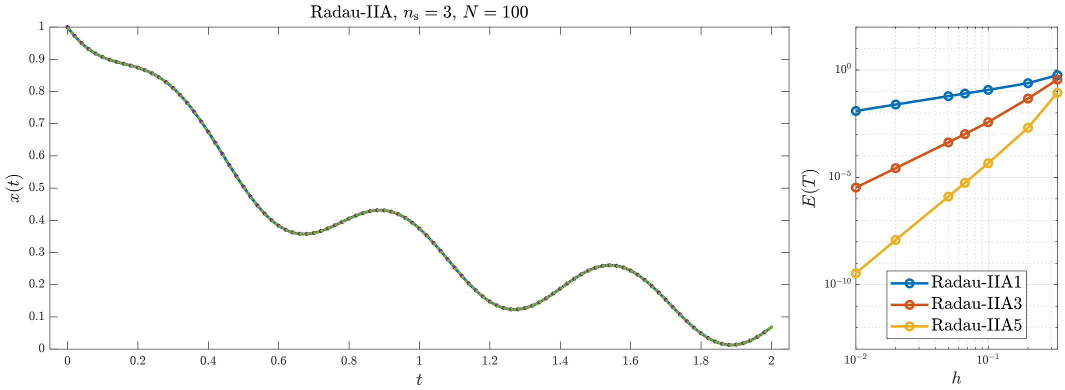
$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.



$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

- Choice of points $c_1, \ldots, c_{n_\mathrm{s}}$ determines properties of method.
- Gauss-Legendre $p = 2n_\mathrm{s}$, Radau-IIA $p = 2n_\mathrm{s} - 1$ good for stiff systems, Lobatto family $p = 2n_\mathrm{s} - 2$.



$$\dot{x}(t) = -0.5x(t)^2 - x(t) + \sin(10t), x(0) = 1$$

Outline of the lecture

Direct collocation in optimal control

Variables $x_{n+1} \in \mathbb{R}^{n_x}$ and $z_n = (k_{n,1}, \dots, k_{n,n_\mathrm{s}}) \in \mathbb{R}^{n_\mathrm{s} n_x}$

Collocation equations

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} k_i b_i \qquad \text{(next value)}$$

$$k_{n,1} = f(t_n + c_1 h, x_n + h \sum_{j=1}^{n_\mathrm{s}} k_{n,j} a_{1,j}, u_n) \qquad \text{(stage Eq. 1)}$$

$$\vdots$$

$$k_{n,n_\mathrm{s}} = f(t_n + c_{n_\mathrm{s}} h, x_n + h \sum_{j=1}^{n_\mathrm{s}} k_{n,j} a_{n_\mathrm{s},j}, u_n), \qquad \text{(stage Eq. } n_\mathrm{s}\text{)}$$

Direct collocation in optimal control

Variables $x_{n+1} \in \mathbb{R}^{n_x}$ and $z_n = (k_{n,1}, \ldots, k_{n,n_\mathrm{s}}) \in \mathbb{R}^{n_\mathrm{s} n_x}$

Collocation equations

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} k_i b_i \qquad \text{(next value)}$$

$$0 = k_{n,1} - f(t_n + c_1 h, x_n + h \sum_{j=1}^{n_\mathrm{s}} k_{n,j} a_{1,j}, u_n) \qquad \text{(stage Eq. 1)}$$

$$\vdots$$

$$0 = k_{n,n_\mathrm{s}} - f(t_n + c_{n_\mathrm{s}} h, x_n + h \sum_{j=1}^{n_\mathrm{s}} k_{n,j} a_{n_\mathrm{s},j}, u_n), \qquad \text{(stage Eq. } n_\mathrm{s}\text{)}$$

Direct collocation in optimal control

Variables $x_{n+1} \in \mathbb{R}^{n_x}$ and $z_n = (k_{n,1}, \dots, k_{n,n_\mathrm{s}}) \in \mathbb{R}^{n_\mathrm{s} n_x}$

Collocation equations

$$x_{n+1} = x_n + h \sum_{i=1}^{n_\mathrm{s}} k_i b_i =: \phi_f(x_n, z_n, u_n) \quad \text{(next value)}$$

$$0 = \begin{bmatrix} k_{n,1} - f(t_n + c_1 h, x_n + h \sum_{j=1}^{n_\mathrm{s}} k_{n,j} a_{1,j}, u_n) \\ \vdots \\ k_{n,n_\mathrm{s}} - f(t_n + c_{n_\mathrm{s}} h, x_n + h \sum_{j=1}^{n_\mathrm{s}} k_{n,j} a_{n_\mathrm{s},j}, u_n) \end{bmatrix} =: \phi_\mathrm{int}(x_n, z_n, u_n) \quad \text{(stage Eqs.)}$$

Direct collocation in optimal control

Variables $x_{n+1} \in \mathbb{R}^{n_x}$ and $z_n = (k_{n,1}, \ldots, k_{n,n_s}) \in \mathbb{R}^{n_s n_x}$

Collocation equations

$$x_{n+1} = \phi_f(x_n, z_n, u_n) \qquad \text{(next value)}$$
$$0 = \phi_{\text{int}}(x_n, z_n, u_n) \qquad \text{(stage Eqs.)}$$

- Use to discretize optimal control problem

Continious time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_c(x(t), u(t))\, \mathrm{d}t + E(x(T))$$

$$\text{s.t.} \quad x(0) = \bar{x}_0$$

$$\dot{x}(t) = f(x(t), u(t))$$

$$0 \geq h(x(t), u(t)), \ t \in [0, T]$$

$$0 \geq r(x(T))$$

- Direct methods: first discretize, then optimize

Continious time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\text{s.t.} \quad x(0) = \bar{x}_0$$

$$\dot{x}(t) = f(x(t),u(t))$$

$$0 \geq h(x(t),u(t)),\ t \in [0,T]$$

$$0 \geq r(x(T))$$

- Direct methods: first discretize, then optimize

1. Parametrize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.

Continious time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\begin{aligned}
\text{s.t.}\quad x(0) &= \bar{x}_0\\
\dot{x}(t) &= f(x(t),u(t))\\
0 &\geq h(x(t),u(t)),\ t\in[0,T]\\
0 &\geq r(x(T))
\end{aligned}$$

- Direct methods: first discretize, then optimize

1. Parametrize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.
2. Discretize cost and dynamics via collocation

$$L_{\mathrm{d}}(x_n,u_n) = \int_{t_n}^{t_{n+1}} L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t.$$

Replace $\dot{x} = f(x,u)$ by

$$\begin{aligned}
x_{n+1} &= \phi_f(x_n, z_n, u_n),\\
0 &= \phi_{\mathrm{int}}(x_n, z_n, u_n).
\end{aligned}$$

Continious time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\text{s.t.}\quad x(0) = \bar{x}_0$$

$$\dot{x}(t) = f(x(t),u(t))$$

$$0 \geq h(x(t),u(t)),\ t \in [0,T]$$

$$0 \geq r(x(T))$$

- Direct methods: first discretize, then optimize

1. Parametrize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.
2. Discretize cost and dynamics via collocation

$$L_d(x_n,u_n) = \int_{t_n}^{t_{n+1}} L_c(x(t),u(t))\,\mathrm{d}t.$$

Replace $\dot{x} = f(x,u)$ by

$$x_{n+1} = \phi_f(x_n,z_n,u_n),$$
$$0 = \phi_{\text{int}}(x_n,z_n,u_n).$$

3. Relax path constraints, e.g., evaluate only at $t = t_n$

$$0 \geq h(x_n,u_n),\ n = 0,\ldots N-1.$$

Continious time OCP into Nonlinear Programs (NLP)

Continuous time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

- Direct methods: first discretize, then optimize

Discrete time OCP (an NLP)

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \quad & \sum_{k=0}^{N-1} L_\mathrm{d}(x_k,u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi_f(x_n,z_n,u_n) \\
& 0 = \phi_\mathrm{int}(x_n,z_n u_n) \\
& 0 \geq h(x_n,u_n),\ n = 0,\dots,N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

Variables $\mathbf{x} = (x_0,\dots,x_N)$, $\mathbf{z} = (z_0,\dots,z_N)$ and $\mathbf{u} = (u_0,\dots,u_{N-1})$.

Direct optimal control methods solve Nonlinear Programs (NLP)

Discrete time OCP (an NLP)

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \ & \textstyle\sum_{k=0}^{N-1} L_{\mathrm{d}}(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad x_0 &= \bar{x}_0 \\
x_{n+1} &= \phi_f(x_n, z_n, u_n) \\
0 &= \phi_{\mathrm{int}}(x_n, z_n u_n) \\
0 &\geq h(x_n, u_n), \ n = 0, \dots, N-1 \\
0 &\geq r(x_N)
\end{aligned}
$$

Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

Direct optimal control methods solve Nonlinear Programs (NLP)

Discrete time OCP (an NLP)

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \ & \textstyle\sum_{k=0}^{N-1} L_{\mathrm{d}}(x_k, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi_f(x_n, z_n, u_n) \\
& 0 = \phi_{\mathrm{int}}(x_n, z_n u_n) \\
& 0 \geq h(x_n, u_n), \ n = 0, \ldots, N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

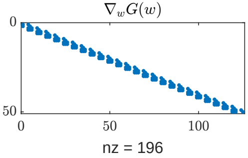
Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

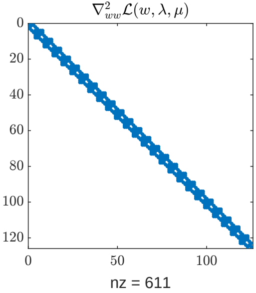
Nonlinear Program (NLP)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^{n_x}} \ & F(w) \\
\text{s.t. } & G(w) = 0 \\
& H(w) \geq 0
\end{aligned}
$$

Obtain large and sparse NLP

Direct optimal control methods solve Nonlinear Programs (NLP)





Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

Nonlinear Program (NLP)

$$\min_{w \in \mathbb{R}^{n_x}} \; F(w)$$
$$\text{s.t. } G(w) = 0$$
$$H(w) \geq 0$$

Obtain large and sparse NLP

Summary

- Numerical simulation methods used to solve ODEs approximately.
- Integration accuracy order and stability play key roles.
- Collocation methods are implicit Runge-Kutta methods with favorable properties.
- All collocation methods are IRK methods, the converse is not true.
- Collocation methods can be used to discretize an OCP into an NLP.
- Choice of discretization method has huge influence on efficacy and reliability of NLP solution.
- Best choice is problem dependent and often requires lot of care.
- Used for practical problems and straightforward to apply.
- Many good software packages exist.

References

- Moritz Diehl, Sébastien Gros. "Numerical optimal control (Draft)," Lecture notes, 2019.
- James B. Rawlings, David Q. Mayne, and Moritz Diehl. Model predictive control: theory, computation, and design. Vol. 2. Madison, WI: Nob Hill Publishing, 2017.
- Gerhard Wanner, Ernst Hairer. "Solving ordinary differential equations II." Vol. 375. New York: Springer Berlin Heidelberg, 1996.