how to get a RR from an ODE

Contents map

developed content units	taxonomy levels
Euler forward discretization	u1, e1
Euler backwards discretization	u1, e1

prerequisite content units	taxonomy levels
ODE	u1, e1
RR	u1, e1

Main ILO of sub-module "how to get a RR from an ODE"

Discretize ODEs and thus find recurrent relations
that approximate ODEs by means of Euler schemes

Discuss which factors affect the validity of Euler discretization
schemes and **exemplify** which problems may occur in practice

Roadmap

- why do we need to numerically simulate?
- Euler methods
- pros and cons
- connections with linearization

Our computers are digital machines, but the ODEs are "analogic" objects

$$\begin{cases} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \end{cases}$$

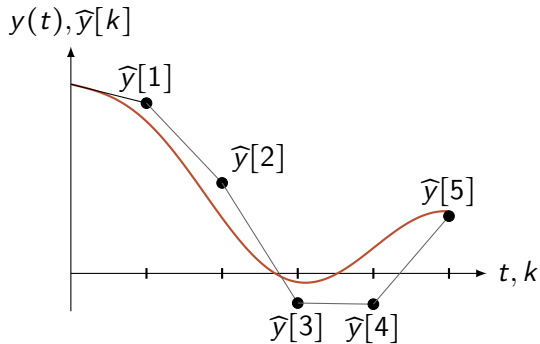
Our computers are digital machines, but the ODEs are "analogic" objects

$$\begin{cases} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \end{cases}$$

the need is for discretizing these objects, both in time and in space

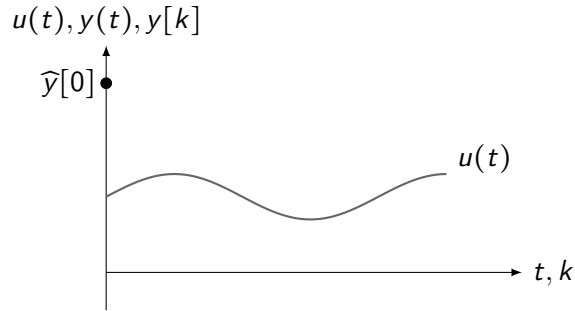
Simulating nonlinear systems = solving the ODE numerically and in a discrete way

i.e., use the fact that we know that $\dot{y} = f(y, u)$, we know the whole u, and we know the initial condition $y(0)$ to compute a series of points $\widehat{y}[0], \widehat{y}[1], \widehat{y}[2], \ldots, \widehat{y}[H]$, that approximate the whole trajectory $y(0:H)$ with H a user-defined prediction horizon:



The simplest numerical solver: Euler's (forward) method

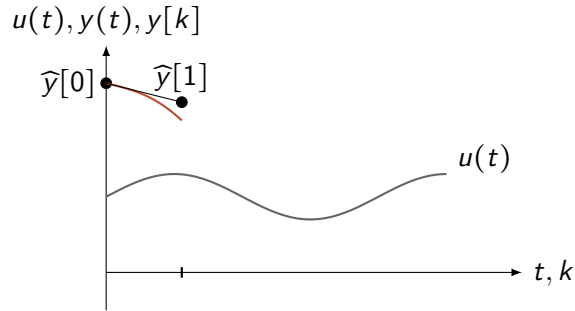
step 0: $\widehat{y}[0] = y(0)$



The simplest numerical solver: Euler's (forward) method

step 0: $\widehat{y}[0] = y(0)$

step 1: $\widehat{y}[1] = \widehat{y}[0] + f\big(\widehat{y}[0], u(0)\big)T$

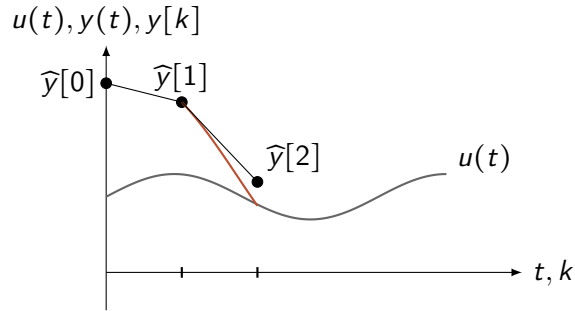


The simplest numerical solver: Euler's (forward) method

step 0: $\widehat{y}[0] = y(0)$

step 1: $\widehat{y}[1] = \widehat{y}[0] + f\big(\widehat{y}[0], u(0)\big)T$

step 2: $\widehat{y}[2] = \widehat{y}[1] + f\big(\widehat{y}[1], u(T)\big)T$



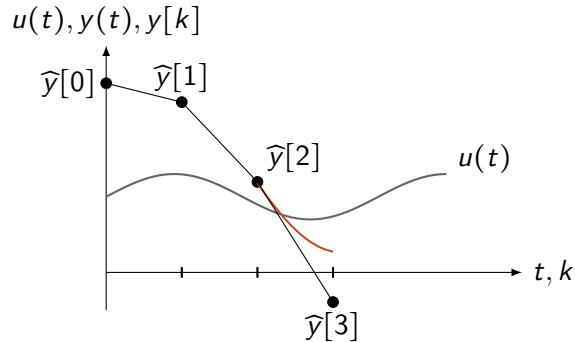
The simplest numerical solver: Euler's (forward) method

step 0: $\widehat{y}[0] = y(0)$

step 1: $\widehat{y}[1] = \widehat{y}[0] + f\big(\widehat{y}[0], u(0)\big)T$

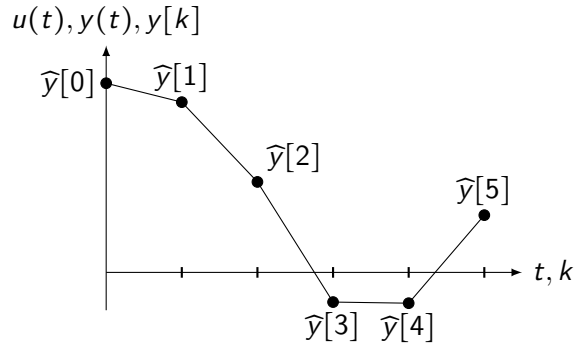
step 2: $\widehat{y}[2] = \widehat{y}[1] + f\big(\widehat{y}[1], u(T)\big)T$

step 3: $\widehat{y}[3] = \widehat{y}[2] + f\big(\widehat{y}[2], u(2T)\big)T$



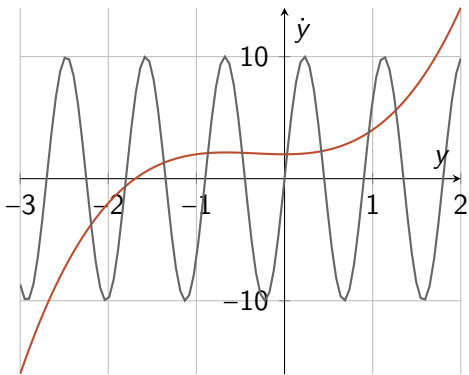
The simplest numerical solver: Euler's (forward) method

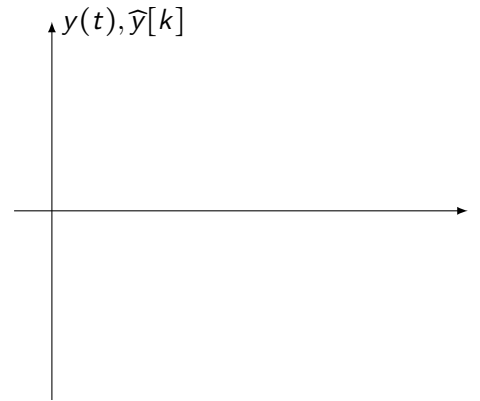
$$
\begin{array}{lll}
\text{step 0:} & \widehat{y}[0] & = y(0) \\
\text{step 1:} & \widehat{y}[1] & = \widehat{y}[0] + f\big(\widehat{y}[0], u(0)\big)T \\
\text{step 2:} & \widehat{y}[2] & = \widehat{y}[1] + f\big(\widehat{y}[1], u(T)\big)T \\
\text{step 3:} & \widehat{y}[3] & = \widehat{y}[2] + f\big(\widehat{y}[2], u(2T)\big)T \\
\vdots & \vdots & \vdots
\end{array}
$$



Tradeoffs:

- the more "gentle" $\dot{y} = f(y)$, the more accurate the approximated trajectory $\widehat{y}$
- the smaller T, the more accurate $\widehat{y}$ & the longer the CPU time
- the longer the prediction horizon in $y(0:H)$, the less accurate the final prediction





Known problem

Euler forward may be numerically unstable, especially for "stiff ODEs" (i.e., ODEs for which some terms that can lead to rapid variation in the solution). Will be seen extensively in following courses!

Note: Euler's forward is only one algorithm of many

Euler Forward:

$$\widehat{y}[k+1] = \widehat{y}[k] + f\big(\widehat{y}[k], u(kT)\big)T$$

Euler Backward:

$$\widehat{y}[k+1] = \widehat{y}[k] + f\big(\widehat{y}[k+1], u((k+1)T)\big)T$$

Note: Euler's forward is only one algorithm of many

Euler Forward:

$$\widehat{y}[k+1] = \widehat{y}[k] + f\big(\widehat{y}[k], u(kT)\big)T$$

Euler Backward:

$$\widehat{y}[k+1] = \widehat{y}[k] + f\big(\widehat{y}[k+1], u((k+1)T)\big)T$$

can be generalized to Runge-Kutta methods, with known tradeoffs and robustness properties; they will be studied in following courses

the specific case of ARMA models

From Continuous-Time to Discrete-Time ARMA Models (with backwards)

from

$$\frac{d^n y(t)}{dt^n} + a_{n-1}\frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_0 y(t) = b_m \frac{d^m u(t)}{dt^m} + \cdots + b_0 u(t)$$

substitute

$$\frac{d\star}{dt} \mapsto \frac{1-q^{-1}}{T}$$

$$\frac{d^2\star}{dt^2} \mapsto \left(\frac{1-q^{-1}}{T}\right)^2$$

$$\frac{d^3\star}{dt^3} \mapsto \left(\frac{1-q^{-1}}{T}\right)^3$$

$$\vdots$$

From Continuous-Time to Discrete-Time ARMA Models (with forward)

from

$$\frac{d^n y(t)}{dt^n} + a_{n-1}\frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_0 y(t) = b_m \frac{d^m u(t)}{dt^m} + \cdots + b_0 u(t)$$

substitute

$$\frac{d\star}{dt} \mapsto \frac{q-1}{T}$$

$$\frac{d^2\star}{dt^2} \mapsto \left(\frac{q-1}{T}\right)^2$$

$$\frac{d^3\star}{dt^3} \mapsto \left(\frac{q-1}{T}\right)^3$$

$$\vdots$$

Example with backwards

$$\dot{y}(t) + ay(t) = bu(t)$$

becomes

$$\frac{\widehat{y}[k] - \widehat{y}[k-1]}{T} + a\widehat{y}[k] = bu[k]$$

Example with backwards

$$\dot{y}(t) + ay(t) = bu(t)$$

becomes

$$\frac{\widehat{y}[k] - \widehat{y}[k-1]}{T} + a\widehat{y}[k] = bu[k]$$

or, rearranging,

$$\left(\frac{1}{T} + a\right)\widehat{y}[k+1] + \left(-\frac{1}{T}\right)\widehat{y}[k] = bu[k]$$

Example with forward

$$\dot{y}(t) + ay(t) = bu(t)$$

becomes

$$\frac{\widehat{y}[k+1] - \widehat{y}[k]}{T} + a\widehat{y}[k] = bu[k]$$

Example with forward

$$\dot{y}(t) + ay(t) = bu(t)$$

becomes

$$\frac{\widehat{y}[k+1] - \widehat{y}[k]}{T} + a\widehat{y}[k] = bu[k]$$

or, rearranging,

$$\left(\frac{1}{T}\right)\widehat{y}[k+1] + \left(-\frac{1}{T} + a\right)\widehat{y}[k] = bu[k]$$

Example with backwards

$$\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = b_1\dot{u}(t) + b_0 u(t)$$

becomes

$$\widehat{y}[k]\left(\frac{1-q^{-1}}{T}\right)^2 + a_1\widehat{y}[k]\left(\frac{1-q^{-1}}{T}\right) + a_0\widehat{y}[k] = b_1 u[k]\left(\frac{1-q^{-1}}{T}\right) + b_0 u[k]$$

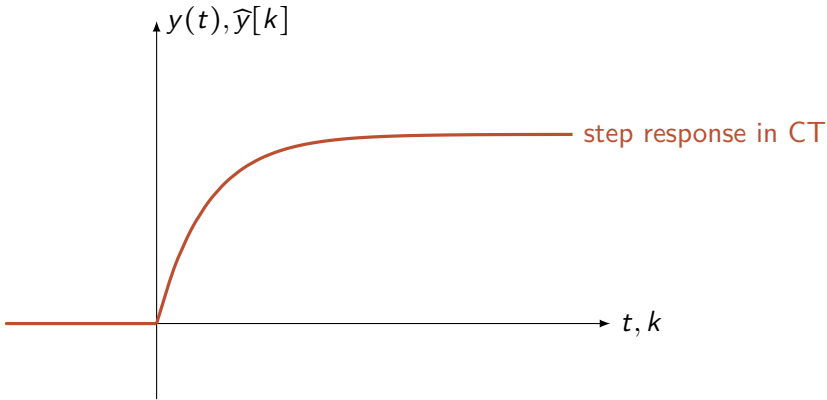
$\Longrightarrow$ rearranging $=$ "algebra"

Discussion

At work, if you have to discretize an ODE of order bigger than one, will you do by hand, or will you use `https://python-control.readthedocs.io/en/latest/generated/control.matlab.c2d.html`?

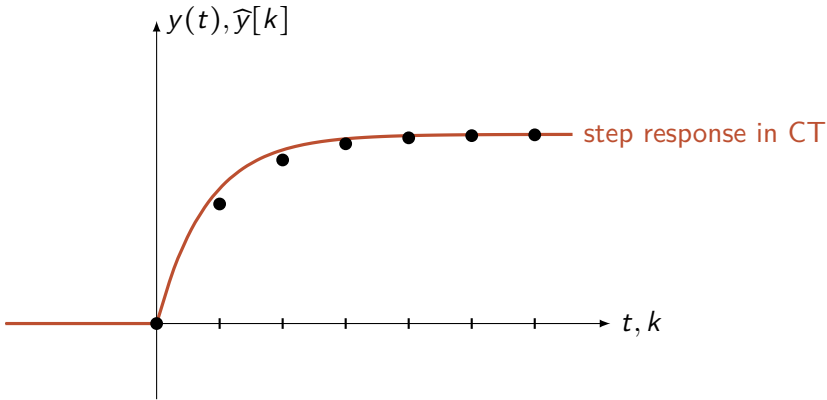
What happens as T decreases?

Example



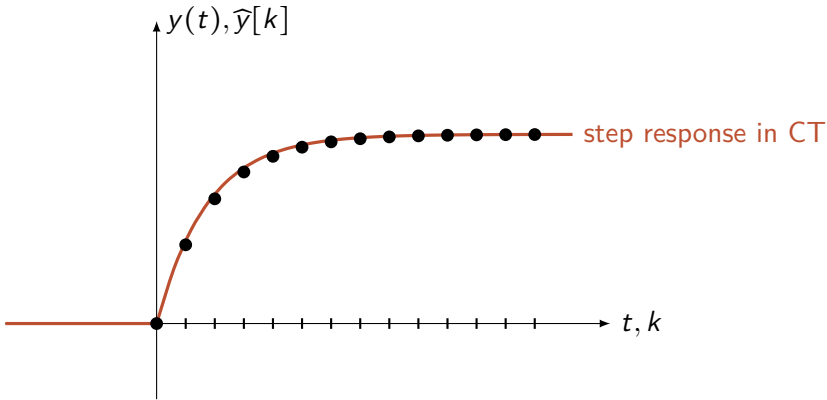
What happens as T decreases?

Example



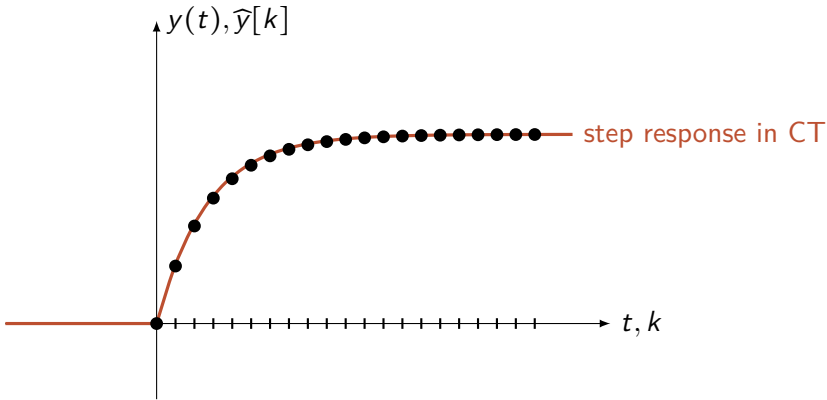
What happens as T decreases?

Example



What happens as T decreases?

Example



Summarizing

Discretize ODEs and thus find recurrent relations
that approximate ODEs by means of Euler schemes

Discuss which factors affect the validity of Euler discretization
schemes and **exemplify** which problems may occur in practice

- backwards or forward Euler = good initial choice
- stiffness of the starting ODE is an issue
- the longer the time horizon to predict, the more problems you will have

Most important python code for this sub-module

C2D

- https://python-control.readthedocs.io/en/latest/generated/control.matlab.c2d.html
- https://www.mathworks.com/help/control/ref/dynamicsystem.c2d.html

Self-assessment material

Question 1

Which of the following is the most important general key factor affecting the accuracy of Euler's forward method when discretizing ODEs?

Potential answers:

I: The step size T
II: The initial condition $y(0)$
III: The type of input signal $u(t)$
IV: The order of the ODE
V: I do not know

Question 2

What is a common issue when using Euler's forward method to solve stiff ODEs?

Potential answers:

I: Numerical instability
II: Increased computational efficiency
III: Exact solution with no error
IV: Reduced sensitivity to initial conditions
V: I do not know

Question 3

Which of the following is a tradeoff when using Euler's forward method for ODE discretization?

Potential answers:

I: Smaller step size T reduces accuracy and increases CPU time
II: Larger step size T reduces accuracy and increases CPU time
III: Smaller step size T increases accuracy but also increases CPU time
IV: Larger step size T increases accuracy and reduces CPU time
V: I do not know

Question 4

What is the main difference between Euler's forward and backward methods?

Potential answers:

I: Euler's backward method is always more accurate than Euler's forward method

II: Euler's backward method does not require an initial condition

III: Euler's backward method uses the derivative at the next time step

IV: Euler's backward method cannot be used for stiff ODEs

V: I do not know

Question 5

When discretizing a continuous-time ARMA model using Euler's backward method, what substitution is made for the first derivative $\frac{dy(t)}{dt}$?

Potential answers:

I: $\frac{1-q^{-1}}{T}$

II: $\frac{q-1}{T}$

III: $\frac{q+1}{T}$

IV: $\frac{1+q^{-1}}{T}$

V: I do not know

Recap of sub-module "how to get a RR from an ODE"

- there are several ways of solving ODEs in a computer
- Euler is the most simple one, but it does not work well with stiff ODEs
- more advanced schemes have better numerical properties

?