TEMPO Summer School on Numerical Optimal Control and Embedded Optimization
University of Freiburg, July 27 - August 7, 2015

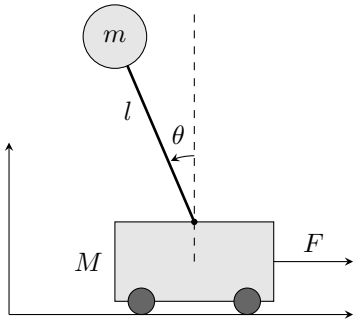
Exercise 1: Nonlinear simulation and the linear-quadratic regulator

Rien Quirynen Dimitris Kouzoupis Moritz Diehl

http://syscop.de/teaching/numerical-optimal-control/

Guiding example: inverted pendulum The guiding example that will be used multiple times throughout this course is the classical system of a pendulum, mounted on top of a cart as illustrated in the figure below. It forms an ideal tutorial example for optimal control since it is simple and intuitive but it can also exhibit rather fast dynamics and nonlinear behavior. The position of the cart will be denoted by p and the pendulum configuration described by the angle θ, using the convention that $\theta = \pi\,\text{rad}$ corresponds to the pendulum hanging down. The system dynamics are described by the following explicit ODE system

$$
\begin{aligned}
\dot{p} &= v, \\
\dot{\theta} &= \omega, \\
\dot{v} &= \frac{-ml\sin(\theta)\omega^2 + mg\cos(\theta)\sin(\theta) + F}{M + m - m(\cos(\theta))^2}, \\
\dot{\omega} &= \frac{-ml\cos(\theta)\sin(\theta)\omega^2 + F\cos(\theta) + (M+m)g\sin(\theta)}{l(M + m - m(\cos(\theta))^2)},
\end{aligned}
$$



in which the parameter values are chosen to be equal to $M = 1\,\text{kg}$, $m = 0.1\,\text{kg}$, $g = 9.81\,\text{m/s}^2$ and $l = 0.8\,\text{m}$.

Numerical simulation and sensitivities

1.1 Write a MATLAB function `ode(t,x,u)` to define the continuous time ODE model in the form $\frac{\mathrm{d}x}{\mathrm{d}t} = f(x,u)$ where the states are $x := (p, \theta, v, \omega)$ and the control input $u := F$

```
function dx = ode(t,x,u)

end
```

As a simple test, you should call your function with the inputs `ode(0,[1 2 3 4],-1)` and check whether the result corresponds to $[3, 4, -2.3415, 12.3683]$.

1.2 Convert the system to the discrete time form $x(k+1) = f_{\mathrm{d}}(x(k), u(k))$, by calling the provided Runge-Kutta integrator of order 4:

```
input.Ts = 0.05;
input.nSteps = 2;
input.u = ...;
input.x = ...;
output = RK4_integrator( @ode, input );
```

For this, the template code `run_simulation.m` is provided which performs a simulation comparison of the RK4 integrator with results from the MATLAB `ode45` routine. Plot the resulting state trajectory and check the results.

1.3 Linearize the discrete time RK4 system to make an approximate system of the form

$$x(k+1) \approx f_{\mathrm{d}}(\bar{x},\bar{u}) + \underbrace{\frac{\partial f_{\mathrm{d}}}{\partial x}(\bar{x},\bar{u})}_{A}(x(k)-\bar{x}) + \underbrace{\frac{\partial f_{\mathrm{d}}}{\partial u}(\bar{x},\bar{u})}_{B}(u(k)-\bar{u}) \tag{1}$$

using a first order Taylor expansion around the point $\bar{x} = [0,0,0,0]^\top$ and $\bar{u} = 0$. When calling the provided function `RK4_integrator`, the simulated values $f_{\mathrm{d}}(\cdot)$ as well as the sensitivity information $\frac{\partial f_{\mathrm{d}}(\cdot)}{\partial x}$, $\frac{\partial f_{\mathrm{d}}(\cdot)}{\partial u}$ are provided in the `output` struct:

```
>> output =

    value: [4x1 double]
    sensX: [4x4 double]
    sensU: [4x1 double]
```

Compute the eigenvalues of $A =$ `output.sensX` using the MATLAB routine `eig`. Is the system stable? NOTE: one can set `input.sens` to the value 0 in case no sensitivities are needed.

Linear-quadratic regulator Our aim now is to design and simulate an infinite horizon linear-quadratic state-feedback regulator for the linearized discrete time system in Eq. (1):

$$\min_{x_0,\ldots,u_0,\ldots} \sum_{k=0}^{\infty} x_k^\top Q x_k + u_k^\top R u_k \tag{2a}$$

$$\text{s.t.} \quad x_0 = \bar{x}_0, \tag{2b}$$

$$x_{k+1} = A\,x_k + B\,u_k, \quad k = 0,\ldots,\infty. \tag{2c}$$

1.4 Choose weighting matrices Q and R and compute the corresponding optimal gain matrix K, by solving the discrete time algebraic Riccati equation. HINT: type `help dlqr`

1.5 Based on the template file `closed_loop.m`, let us first apply the *offline* optimal control values directly to the nonlinear pendulum system $x(k+1) = f_{\mathrm{d}}(x(k),u(k))$. Depending on the initial states $x_0 = [0,\theta_0,0,0]^\top$, does the controller bring the system to steady state? HINT: the offline LQR optimal control trajectory can be obtained as follows:

```
state_lin = x0;
for k = 1:N
  u_LQR(k) = -K*state_lin;
  state_lin = A*state_lin + B*u_LQR(k);
end
```

1.6 Using the same template file, validate the performance of your LQR controller by closing the loop with the nonlinear simulation. This means that the optimal control input to be applied, will be determined based on the state feedback result in `state_sim`. Does the controller bring the system to steady state this time? In addition, try out different initial states for the closed-loop simulation and look for a reasonable tuning of the Q and R weighting matrices (simply based on a visual inspection of the pendulum's behavior in this case).

1.7 **Extra**: In case you have already installed the ACADO Toolkit (installation instructions on our course event page), you can additionally export efficient C-code for the `RK4_integrator` to be used in this and following exercises. Have a look at the template file `ACADO_SIMexport.m` and try to complete it with the nonlinear pendulum dynamics. Using the MATLAB routines `tic/toc`, how fast is the exported ACADO integrator compared to our original `RK4_integrator` in MATLAB?