Exercises for Lecture Course on Optimal Control and Estimation (OCE)
Albert-Ludwigs-Universität Freiburg – Summer Term 2015

Exercise 5 - Non-linear Programs (deadline: 8.6.2015, 2:15pm)

Prof. Dr. Moritz Diehl and Jonas Koenemann

In this exercise, we use the same simple pendulum system as in the previous exercise sheets. This time, we solve the non-linear problem formulated as an Non-Linear Program (NLP) with the Matlab `fmincon` function.

Recall the system dynamics of the pendulum with state $x = [\phi, \omega]^\top$:

$$\dot{x} = \begin{bmatrix} \omega \\ 2\sin(\phi) + u \end{bmatrix}$$

and the optimal control problem that we want to solve

$$\underset{x_0, \ldots, x_N, u_0, \ldots, u_{N-1}}{\text{minimize}} \quad \sum_{k=0}^{N-1} (\phi_k^2 + u_k^2)$$

$$\begin{aligned} \text{subject to} & \\ \bar{x}_0 - x_0 &= 0 \\ f(x_k, u_k) - x_{k+1} &= 0, \quad \text{for} \quad k = 0, \ldots, N-1 \\ \omega_{\min} &\leq \omega_k \leq \omega_{\max}, \quad \text{for} \quad k = 0, \ldots, N \\ u_{\min} &\leq u_k \leq u_{\max}, \quad \text{for} \quad k = 0, \ldots, N-1 \end{aligned}$$

where the discrete time system dynamics are obtain by a Runge-Kutta integrator of order 4 with a timestep of $h = 0.2$. The horizon of the optimal control problem is $N = 60$, the given initial state is $\bar{x}_0 = [-\pi, 0]$, and the bounds are given by $\omega_{\min} = -\pi$, $\omega_{\max} = \pi$, $u_{\min} = -1.1$, $u_{\max} = 1.1$.

In order to pass the optimal control problem to the solver we first have to formulate it as an NLP. The variables of the optimal control problem are summarized in a vector $y = (x_0, u_0, \ldots, u_{N-1}, x_N)^\top$. Then the NLP has the following form:

$$\underset{y}{\text{minimize}} \quad \psi(y)$$

$$\begin{aligned} \text{subject to} & \\ G(y) &= 0, \\ y_{\min} \leq y &\leq y_{\max} \end{aligned}$$

1. Write down the objective function $\psi(y)$ and the constraints on paper. Use the same order for the constraints as in the optimal control problem.

$$G(y) = \begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix} \quad y_{\max} = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \quad y_{\min} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

   Implement the objective and the equality constraints as Matlab functions

   (3 points)

2. Check if your function $G(y)$ does what you want by writing a forward simulation function `[y]=simulate(x0,U)` that simulates, for a given initial value x_0 and control profile $U = (u_0, \ldots, u_{N-1})$, the whole trajectory $x_1, \ldots, x_N$ and constructs from this the full vector $y = (x_0, u_0, x_1, \ldots, x_N)$. If you generate for any x_0 and U a vector y and then you call your function $G(y)$ with this input, almost all of your residuals should be zero. Which ones are not zero?

   As a test, simulate e.g. with $x_0 = [0, 0.5]^\top$ and $u_k = 1$, $k = 0, \ldots, N-1$ in order to generate y, and then call $G(y)$, to test that your function G is correct. Specify the norm of the residuals $G(y)$.

   (2 points)

3. Check the Matlab documentation to find out how to use `fmincon`:

   `http://de.mathworks.com/help/optim/ug/fmincon.html`

   and how to properly pass constraints to `fmincon`:

   `http://de.mathworks.com/help/optim/ug/writing-constraints.html`

   Then in your Matlab script, use `fmincon` to solve the NLP:

   ```
   options=optimoptions(@fmincon, 'display','final/iter','MaxFunEvals',100000);
   y=fmincon(@objective,y0,[],[],[],[],lby,uby,@nonlconstraints,options);
   ```

   As an initialization for y_0 you can use $\bar{x}_0$ for all state variables and zero for all control variables.

   How many iteration does the solver need to converge (use display option `iter`)? How long does the call to the minimizer take (use `tic/toc` and the display option `final`)? Plot the evolution of the state and the applied controls in time. Make an animation to see if the pendulum swings up (no need to submit).

   (4 points)

4. Do a RK4 simulation of the pendulum and apply the optimal controls of Task 3 open-loop. Does the pendulum swing up? Does the resulting state trajectory differ from the output of the solver? Why?

   (1 points)

5. **Bonus question:** Set options to the solver to use different algorithms (Active-set, Interior-point, SQP). Compare number of iterations and computation times.

   Play with other options of the solvers like the type of finite differences for computing the Jacobian, stopping criteria, and tolerances for violating the constraints. How do they influences computation time, number of iterations, and precision of the solution.

   (2 bonus points)