Numerical Optimal Control Course at IfA, ETH Zurich
on Feb-May 2011, by Prof. M. Diehl (K.U. Leuven)

Exercise 1: Nonlinear Programming and Single Shooting

Moritz Diehl, Peter Hokayem, Robin Vujanic

It is recommended that students work in pairs of two people in front of one computer. The first exercise has as its aim the formulation and numerical solution of a simple nonlinear programming problem and then of a simple optimal control problem. The solution algorithm is provided by MATLAB.

1. Log in and start MATLAB and open an editor of your choice. Use `help fmincon` to learn what is the syntax of a call of `fmincon`.

2. Now regard, first just on paper, the following nonlinear programming problem (NLP):

$$\min_{x \in \mathbb{R}^2} x_2 \quad \text{subject to} \quad \left\{ \begin{array}{rcl} x_1^2 + 4x_2^2 & \leq & 4 \\ x_1 & \geq & -2 \\ x_1 & = & 1 \end{array} \right.$$

   (a) How many degrees of freedom, how many equality, and how many inequality constraints does this problem have?

   (b) Sketch the feasible set Ω of this problem. What is the optimal solution?

   (c) Bring this problem into the NLP standard form

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad \left\{ \begin{array}{rcl} g(x) & = & 0 \\ h(x) & \geq & 0 \end{array} \right.$$

   by defining the dimension n and the functions f, g, h along with their dimensions appropriately.

3. Now formulate three MATLAB functions f, g, h for the above NLP, choose an initial guess for x, and solve the problem using `fmincon`. Check that the output corresponds to what you expected.

4. Now we want to solve the first optimal control problem in this course. Aim is to bring an harmonic oscillator to rest with minimal control effort. For this aim we regard the linear discrete time dynamic system:

$$\begin{bmatrix} p_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \Delta t \left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k \right), \quad k = 1, \ldots, N-1 \tag{1}$$

   Choose the fixed values $p_1 = 10, v_1 = 0, \Delta t = 0.2, N = 51$, and denote for simplicity from now on $x_k = (p_k, v_k)^T$. Now write a MATLAB routine `[xN]=oscisim(U)` that computes x_N as a function of the control inputs $U = (u_1, \ldots, u_{N-1})^T$. Mathematically, we will denote this function by $f_{\text{oscisim}} : \mathbb{R}^{N-1} \to \mathbb{R}^2$.

5. To verify that your routine does what you want, plot the simulated positions $p_1, \ldots, p_N$ within this routine for the input $U = 0$.

6. Now we want to solve the optimal control problem

$$\min_{U \in \mathbb{R}^{N-1}} \|U\|_2^2 \quad \text{subject to} \qquad f_{\text{oscisim}}(U) = 0$$

   Formulate and solve this problem with `fmincon`. Plot the solution vector U as well as the trajectory of the positions in the solution.

7. Now add inequalities to the problem, limiting the inputs u_k in amplitude by an upper bound $|u_k| \leq u_{\max}, k = 1, \ldots, N-1$. This adds $2(N-1)$ inequalities to your problem. Which?

8. Formulate the problem with inequalities in `fmincon` (note there is a sign change for inequalities compared to the lecture). Experiment with different values of $u_{\max}$, starting with big ones and making it smaller. If it is very big, the solution will not be changed at all. At which critical value of $u_{\max}$ does the solution start to change? If it is too small, the problem will become infeasible. At which critical value of $u_{\max}$ does this happen?

9. A remark and possible extra work: both of the above problems are convex, i.e. each local minimum is also a global minimum. Note that the equality constraint of the optimal control problems is just a linear function at the moment. If you have time left, you may make this constraint nonlinear and thus make the problem nonconvex. One way is to add a small nonlinearity into the dynamic system (1) by making the spring nonlinear, i.e. replacing the term -1 in the lower left corner of the system matrix by $-(1+\mu p_k^2)$ with a small μ, and solving the problem again. At which value of μ does the solver `fmincon` need twice as many iterations as before?