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

Exercise 4: Direct single and multiple shooting

Rien Quirynen Dimitris Kouzoupis Moritz Diehl

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

The inverted pendulum Let us look again at the example of an inverted pendulum, mounted on top of a cart, of which the system dynamics are described by the same nonlinear ODE system as in Exercise 1. We decide on the following continuous time OCP formulation for the *swing-up* of the inverted pendulum, where the terminal constraint is defined by the upward configuration:

$$\begin{aligned}
\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_0^T \frac{1}{2}\,\|u(t)\|_R^2\,\mathrm{d}t && \text{(1a)}\\
\text{subject to} \quad & x(0) \;=\; \bar{x}_0, && \text{(1b)}\\
& \dot{x}(t) \;=\; f(x(t),u(t)), \qquad \forall t\in[0,T], && \text{(1c)}\\
& x(T) \;=\; x^{\text{ref}}, && \text{(1d)}\\
& -20 \;\le\; u(t) \;\le\; 20, \qquad \forall t\in[0,T], && \text{(1e)}
\end{aligned}$$

where the states are defined $x=(p,\theta,v,\omega)$ and the control input $u=F$. To keep the implementation of this exercise simple, we have no stage cost on the differential states in the OCP formulation.

Single shooting method For the sake of simplicity, we also consider an equidistant grid over the control horizon consisting in the collection of time points t_i, where $t_{i+1}-t_i=\frac{T}{N}=: T_s$ for $i=0,\dots,N-1$. Additionally, we consider a piecewise constant control parametrization $u(\tau)=u_i$ for $\tau\in[t_i,t_{i+1})$. In the context of single shooting, let us define the function $\phi_{\text{sim}}(\bar{x}_0,U)$ which represents a numerical approximation for the solution $x(T)$ of the following initial value problem:

$$\dot{x}(\tau)=f(x(\tau),u(\tau)), \quad \text{where } u(\tau)=u_i \text{ for } \tau\in[t_i,t_{i+1}) \quad \forall i=0,\dots,N-1, \tag{2}$$

where $x(0)=\bar{x}_0$. This function typically needs to be evaluated numerically using an integration method, such as the RK4 integrator used in previous exercises. A single shooting discretization of the OCP in (1), then results in the following Nonlinear Program (NLP):

$$\begin{aligned}
\min_{U} \quad & \frac{1}{2}\sum_{k=0}^{N-1} T_s\,\|u_k\|_R^2 && \text{(3a)}\\
\text{s.t.} \quad & -20 \;\le\; u_k \;\le\; 20, \qquad k=0,\dots,N-1, && \text{(3b)}\\
& 0 \;=\; \phi_{\text{sim}}(\bar{x}_0,U)-x^{\text{ref}}, && \text{(3c)}
\end{aligned}$$

with control trajectory $U=[u_0^\top,\dots,u_{N-1}^\top]^\top$.

4.1 Try to complete the template code in the file `swing_up.m`, implementing the single shooting method as described above. For this, you mainly have to complete the MATLAB functions `cost_single.m` and `constr_single.m`, respectively to evaluate the objective and the nonlinear terminal constraint.

4.2 Go through the code to make sure that you understand how it works. Note that, similar to the previous exercise, first order derivatives are provided to `fmincon` in order to improve its convergence behaviour. Run the script and interpret the obtained solution to the optimal control problem. **HINT**: Always check the information which `fmincon` (or any other solver) outputs to you.

Note that the MATLAB function `hessian_single` is provided to let `fmincon` use the Gauss-Newton Hessian approximation (see lecture on 'Newton-type optimization').

Multiple shooting method For various reasons, we typically prefer to apply a direct multiple shooting discretization instead. For this, let us define a local function $\phi_k(\cdot)$ over each shooting interval which represents a numerical approximation for the solution $x(t_{k+1})$ of the following initial value problem:

$$\dot{x}(\tau) = f(x(\tau), u_k), \quad \tau \in [t_k, t_{k+1}], \tag{4}$$

where $x(t_k) = x_k$. The resulting NLP reads as follows

$$\begin{aligned}
\min_{X,U} \quad & \frac{1}{2} \sum_{k=0}^{N-1} T_s \, \|u_k\|_R^2 && \text{(5a)}\\
\text{s.t.} \quad & 0 \;=\; x_0 - \bar{x}_0, && \text{(5b)}\\
& 0 \;=\; \phi(x_k, u_k) - x_{k+1}, \quad k = 0, \ldots, N-1, && \text{(5c)}\\
& 0 \;=\; x_N - x^{\text{ref}}, && \text{(5d)}\\
& -20 \;\leq\; u_k \;\leq\; 20, \qquad k = 0, \ldots, N-1, && \text{(5e)}
\end{aligned}$$

with state trajectory $X = [x_0^\top, \ldots, x_N^\top]^\top$ and control trajectory $U = [u_0^\top, \ldots, u_{N-1}^\top]^\top$.

4.3 Similar to the working code for single shooting, try to solve the same problem using direct multiple shooting. For this, you will have to make a new version of both the MATLAB functions `cost_multiple.m` and `constr_multiple.m`, respectively for the objective and constraint functions.

For nonlinear optimization algorithms, it is generally important to have accurate (first and possibly higher order) derivative information. In addition, the multiple shooting NLP from Eq. (5) has a clear sparsity structure, which `fmincon` cannot detect automatically. It is this structure which allows us in general to implement multiple shooting (nearly) as efficient as single shooting! It is therefore important that these MATLAB functions additionally provide first order derivative information. Try to complete the templates provided on our event webpage.

4.4 You can now call `fmincon` in the following way to solve the multiple shooting based NLP:

```
    sol_multiple = fmincon(@cost_multiple,Z_MS,[],[],[],[],...
        LB_MS,UB_MS,@constr_multiple,options);
```

Note that you will have to set up new bound values and define a proper initialization for the multiple shooting optimization variables, which include all controls and states over the full horizon Z_MS $= [x_0, u_0, \ldots, x_{N-1}, u_{N-1}, x_N]$. HINT: This order is important for the sparsity of the problem, mentioned earlier.

4.5 Do you see any difference in the number of iterations needed by `fmincon`? Unlike for single shooting, note that you could now additionally try out different initializations for the state trajectory X.

4.6 **Extra**: Instead of the rather restrictive terminal equality constraint, try to include a terminal cost of the form $x_N^\top P x_N$ (which dominates over the penalty on the control values). The terminal constraint can be removed. For this task, you can restrict to the multiple shooting implementation for which you will also need to update the function `hessian_multiple`.