Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
Institut für Dynamische Systeme und Regelungstechnik

151-0310-00 **Problem Set 8**

Topic: **Constrained Nonlinear Programming**

April 27, 2022

Exercise 1 (Single and Multiple Shooting)

In this exercise, we want to control the Van der Pol oscillator, which was introduced in the lecture. In state-space representation, the system dynamics are given by:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u) = \begin{bmatrix} (1 - x_2^2)x_1 - x_2 + u \\ x_1 \end{bmatrix} \tag{1}$$

We formulate the optimal control problem (OCP) as:

$$\begin{aligned} \min_{\boldsymbol{x}(\cdot),u(\cdot)} \quad & \overbrace{\int_0^{t_\text{f}} x_1^2(t) + x_2^2(t) + u^2(t)\, dt}^{J(\boldsymbol{x},u)} \\ \text{s.t.} \quad & \dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), u(t)) \\ & -1 \leq u(t) \leq 1 \\ & x_1(t) \geq -0.25 \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 = [\,0 \;\; 1\,]^\top \end{aligned} \tag{2}$$

in which for the final time t_f we choose 10 s. We discretize (2) using the fourth-order Runge-Kutta method (RK4) and a sampling time T_s of 0.5 s, such that the prediction/control horizon N is 20 time steps long. Your task is to transform the optimal control problem (2) into a constrained nonlinear program (NLP) using MATLAB, CasADi and one of two shooting methods. Subsequently, you will solve the resulting NLP using the interior-point solver IPOPT (which comes with CasADi).

a) Solve the optimal control problem using single shooting. Use the template `ps08_ex1a_main.m`, which defines the structure of the code and contains comments to guide you through the steps.

b) Solve the optimal control problem using multiple shooting. Use the template `ps08_ex1b_main.m`, which defines the structure of the code and contains comments to guide you through the steps.

c) What are the advantages and disadvantages of each of these shooting methods?

Exercise 2 (Solving the OCP using SQP)

In the previous exercise, you solved the NLP directly using an interior-point solver. In this exercise, we want to solve (2) using sequential quadratic programming (SQP). To this end, use only multiple shooting and the same parameters as in the previous exercise. Based on what you learned about SQP in the previous problem set, complete the following tasks in MATLAB:

a) Write a function `[qp_W,qp_gradJ,qp_gradhT,qp_h,lb,ub] = createCasadiFunctions(options)` that calculates all the necessary CasADi functions and bounds that define the quadratic program (QP) at each iteration k. Take your solution from exercise 1 question **b** as a starting point. In the structure `options`, you can put everything that you would like to change later, such as the sampling time or the horizon length.

b) Write a script in which the options are defined, the CasADi functions are created, and the optimization problem is solved using SQP. Use the qpOASES function `qpOASES_sequence` to solve the sequence of QPs with varying matrices[1]. For the lower and upper bounds that you give to qpOASES, notice that the QP problem is a linearization of (2) around $\boldsymbol{x}_k$. Thus, you need to change the bounds on $\boldsymbol{s}_{x_k}$ accordingly each iteration.

The variable `lambda` returned by qpOASES contains both the Lagrange multipliers (also known as dual multipliers or dual variables) of the linear equality constraints (which we denote as $\boldsymbol{\lambda}$) and those corresponding to the lower and upper bounds on the decision variables (which we denote as $\boldsymbol{\mu}$). Whereas you only have to use $\boldsymbol{\lambda}$ to update the QP matrices for the next iteration, you need to consider both when calculating the KKT conditions:

```
kktCond = [gradJ - gradhT'*lambda - mu; h];
```

In this MATLAB expression, `lambda` means $\boldsymbol{\lambda}$.

c) Run your SQP algorithm to solve the OCP. Use a full step length for the search direction. Let the algorithm terminate if

- the 2-norm of the KKT conditions becomes smaller than 10^{-8}, or
- the number of SQP iterations reaches a maximum of 100.

Does the result match that of IPOPT from exercise 1? How many SQP iterations are needed for the algorithm to converge? How does the computation time compare?

[1] Using the function `qpOASES_sequence`, each QP will be hot-started with the solution from the previous one, which in general makes the SQP faster. Refer to the user manual for more information.