EN530.603 Applied Optimal Control
Homework #7
November 3, 2021

Due: November 10, 2021 (before class)

Professor: Marin Kobilarov

The following are practice problems focusing on basic numerical trajectory optimization, to which we have provided partial solution which you need to understand and modify.

1. *Direct Shooting*: An example of a direct shooting method (i.e. optimizing over a discrete sequence of controls) is provided with an example application to a simple car model (`car_shooting.m`) that exploits the least-squares structure of the cost function. You are required to apply the same method to a new system of your choice, or to the two-link arm example with discrete dynamic function f_k given in `arm_sim.m`.

   Details: This problem is implemented using the Gauss-Newton (GN) least-squares method for optimizing over the controls $\xi \triangleq u_{0:N-1}$. Using the dynamics, each state can be expressed as a function of the controls ξ which is encoded through the functions $x_k = \psi_k(\xi)$ for $k = 0, \ldots, N$. The cost is then expressed as $J(\xi) = \frac{1}{2} g(\xi)^T g(\xi)$, where $g(\xi)$ is given by

$$g(\xi) = \begin{bmatrix} \sqrt{R_0}\,(u_0 - u_d) \\ \sqrt{Q_1}\,(\psi_1(\xi) - x_d) \\ \sqrt{R_1}\,(u_1 - u_d) \\ \vdots \\ \sqrt{Q_{N-1}}\,(\psi_{N-1}(\xi) - x_d) \\ \sqrt{R_{N-1}}\,(u_{N-1} - u_d) \\ \sqrt{Q_f}\,(\psi_N(\xi) - x_f) \end{bmatrix},$$

   for some desired control u_d, desired state x_d, and desired final state x_f. Since $R_k > 0$ the Jacobian $\partial g(\xi)$ is guaranteed to be full rank and one can apply a GN iterative method directly to update $\xi \to \xi + \delta\xi$ where $\delta\xi = -(\partial_\xi g^T \partial_\xi g)^{-1} \partial_\xi g^T g$. In addition, the Jacobian has a lower-triangular structure that can be exploited in the Cholesky GN solution.

2. *Direct Collocation*: Implement a nonlinear programming strategy using direct collocation (i.e. optimization over a discrete sequence of states and controls, as explained in the notes and also in Betts, 1998) to one of the two provided models (car or arm), or to a model of your choice. This can be accomplished by defining the cost and constraints and finding a solution using Matlab `fmincon`. See example `trajopt_sqp_car.m`. Obstacles should be added as inequality constraints.

3. *Differential Dynamic Programing*: A general discrete optimal control code (ddp.zip) is provided along with two examples (2-dof robotic arm and a second-order wheeled vehicle model).

You have two options: 1) *extend* one of the two provided models; or 2) implement a new model of your own choice in a similar manner as the two examples. In both cases you must include meaningful control bounds (by modifying ddp.m) and add environmental obstacles (recall HW5#3). Obstacles should be added as a penalty/repulsive potential term to the cost function. In your plots clearly show that the computed controls do not exceed the specified bounds.

Details: Assume that we need to enforce a constraints of the form

$$c_k(x_k, u_k) \leq 0, \text{ for all } k = 0, \ldots, N-1,$$

where $c_k = (c_k^1, \ldots, c_k^m)$ are m constraint functions. This can be accomplished by adding penatly terms to the trajectory costs L_k, i.e. by using

$$\bar{L}_k(x, u) = L_k(x, u) + \frac{\beta_k}{2}\|g_k(x, u)\|^2,$$

in place of $L_k(x, u)$, where $g_k(x, u) = \max(c_k(x, u), 0)$ for some chosen coefficient $\beta_k > 0$ that controls the "softness" of the constraint (here max is applied independently to each component of the vector c_k). Then the Jacobian with respect to x is

$$\nabla_x \bar{L}_k(x, u) = \nabla_x L_k(x, u) + \beta_k \partial_x g_k(x, u)^T g_k(x, u),$$

and is similarly defined with respect to u. The Hessian is

$$\nabla_x^2 \bar{L}_k(x, u) = \nabla_x^2 L_k(x, u) + \beta_k \partial_x g_k(x, u)^T \partial_x g_k(x, u) + \beta_k \sum_{i=1}^{m} g_k^i(x, u) \nabla_x^2 g_k^i(x, u)$$

and is similarly defined with respect to u. In some cases (when either g_k is small or $\nabla_x^2 g_k^i(x, u)$ has small eigenvalues) the last term above can be ignored.

See example `ddp_pnt_obst.m`

Note: upload your code as a single zip file (please name it as *LastName_FirstName_HW7.zip*) to the File upload link on the class webpage; in addition attach a printout of the code and plots to your homework solutions.