EN530.603 Applied Optimal Control
Homework #4

October 22, 2014

Due: November 5, 2014 (before class)

Professor: Marin Kobilarov

1. Consider the minimization of

$$J = \frac{1}{2}x(1)^2 + \int_0^1 \frac{1}{2}[x(t)u(t)]^2 dt,$$

subject to the nonlinear dynamics

$$\dot{x} = xu, \qquad x(0) = 1.$$

Derive an optimal feedback control using the HJB equation. In the process, show that the HJB partial differential equation has a solution that is a quadratic function in x.

2. Consider the LQR problem with known disturbance $w(t)$ which requires the minimization of

$$J = \frac{1}{2}x(t_f)^T P_f x(t_f) + \frac{1}{2}\int_{t_0}^{t_f} [x(t)^T Q(t)x(t) + u(t)^T R(t)u(t)]dt,$$

subject to

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + w(t).$$

a) Using the HJB equation, show that a possible value function has the form

$$V(x(t), t) = \frac{1}{2}x(t)^T P(t)x(t) + b(t)^T x(t) + c(t)$$

and show that the associate optimal control is

$$u(t) = K(t)x(t) + k(t),$$

where $K(t)$ and $k(t)$ are the feedback gain matrix and feedforward term, respectively. Derive the differential equations for $\dot{P}, \dot{b}, \dot{c}$ and their boundary conditions which satisfy the HJB equation.

b) Do the discrete-time equivalent of a)

3. Read the paper: J. Betts "Survey of Numerical Methods for Trajectory Optimization", Journal of Guidance, Control, and Dynamics, Vol. 21, No. 2, 1998

4. Nonlinear numerical optimal control implementation:

a) An example of a direct shooting method 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 file arm_sim.m. *Details*: this problem is implemented using the Gauss-Newton (GN) least-squared method for optimizing over the controls $\xi \triangleq u_{0:N-1}$. Using the dynamics each state can be expressed as a function of 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 $\delta\xi = -(\partial g^T \partial g)^{-1} \partial g^T g$. In addition, the Jacobian has a lower-triangular structure that can be exploited in the Cholesky GN solution.

b) 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 HW3#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`

c) Implement a nonlinear programming strategy using direct transcription/collocation (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.

Note: email your code to marin@jhu.edu with a subject line starting with: **EN530.603.F2014.HW4**; in addition attach a printout of your code and *plots* (annotated if necessary) to your homework solutions.