ME C231A, EECS C220B
Lab 4: general dynamic system optimization by substitution/simulation
September 19 & 21, 2016

1 Dynamic system

With parameters $\gamma = 10$ and $\tau = 0.2$,

$$\begin{aligned} \dot{x}_1(t) &= \sin(x_1(t)) + \tfrac{\gamma}{1+x_1^2(t)} \arctan(x_2(t)) \\ \dot{x}_2(t) &= \tfrac{1}{\tau}\left(-x_2(t) + u(t)\right) \end{aligned}$$

Write concisely as $\dot{x} = f(x, u)$.

2 Objective

Starting from initial condition $x = 0_2$, minimize the tracking error

$$J := \int_0^{T_F} |x_1(t) - x_{1,d}(t)|^2 \, dt$$

where $x_{1,d}$ is a given, desired trajectory for x_1. The specific trajectory is defined in the script file.

3 Constraints

- Final state condition must be within 5% of the desired value (inequalities are written for the case where $x_{1,d}(T_F) > 0$)

$$0.95 x_{1,d}(T_F) \leq x_1(T_F) \leq 1.05 x_{1,d}(T_F)$$

- For some given $R_B > 0$, the joint input/state constraint

$$\left| \frac{1}{\tau}\left(-x_2(t) + u(t)\right) \right| \leq R_B \quad \forall 0 \leq t \leq T_F$$

  If x_2 represents a simple 1st-order actuator model, this captures a rate constraint on the actuation.

4 Approach: Discretzation

1. Replace model by 1st-order Euler discretization, so

$$x(k+1) = x(k) + T_s f(x_E(k), u(k))$$

with a fixed discretization time, T_s. Let $N := \frac{T_F}{T_s}$. Note that although we have kept the same notation for x and u, these variables are Euler approximations to the continuous-time signals in the original differential equation model.

2. Replace objective with summation,

$$J = T_s \sum_{k=0}^{N} |x_1(k) - x_{1,d}(k)|^2$$

3. Write final constraints with

$$0.95 x_{1,d}(N) \leq x_1(N) \leq 1.05 x_{1,d}(N)$$

4. Replace pointwise constraints with

$$\left| \frac{1}{\tau} \left(-x_1(k) + u(k) \right) \right| \leq R_B \quad \forall k = 0, 1, \ldots, N-1$$

5 `fmincon`

The solver `fmincon` is a general nonlinear optimization solver, that minimizes an objective function subject to inequality and equality constraints, specifically

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to } Ax \leq b, A_e x = b_e, L \leq x \leq U, g(x) \leq 0, g_e(x) = 0$$

where A, A_e, b, b_e, L, U are matrices and vectors of appropriate dimensions, and $f : \mathbb{R} \to \mathbb{R}, g : \mathbb{R}^n \to \mathbb{R}^m, g_e : \mathbb{R}^n \to \mathbb{R}^p$. For f, g and g_e, function handles must be defined. Specifically, for f, the user must define a function handle with one argument (x, the n-dimensional decision variable), and the function must return a single scalar, whose value if $f(x)$, the objective function's value. For the constraint functions, there is only one function handle specified. Again, this function accepts only one input argument (x, the decision variable) and returns two arguments: a vector in $\mathbb{R}^m$ with value $g(x)$ and a vector in $\mathbb{R}^p$ with value $g_e(x)$.

6 Approach #1: decision variables are $u_0, u_1, \ldots, u_{N-1}$

In this case, the objective function and the constraint functions accept the input argument and simulate the dynamic equations from the initial condition, obtaining the state trajectory associated with the given input sequence. Then, with state and input sequence known, the objective and/or constraint function values are calculated and returned

7 Approach #2: decision variables are $u_0, u_1, \ldots, u_{N-1}$ and $x_1, \ldots, x_N$

In this case, the objective function merely accepts the decision variable, separates it into x and u and calculates the defined objective function. The constraint function does the analogous procedure, but also includes $2N$ additional equality constraints, by returning $x_{k+1} - x_k - T_s \cdot f(x_k, u_k)$ (for $k = 0, \ldots, N-1$) in the g_e function as values that should ultimately be equal to 0. In other words, the dynamic equations are enforced as equality constraints on the large set of decision variables, $u_0, u_1, \ldots, u_{N-1}$ and $x_1, \ldots, x_N$.

8 Use of `persistent` variables

Since `fmincon` repeatedly calls the objective function and constraint function, and usually in succession at the same value of the decision variable, we can make the code more efficient by storing (in case #1) the computed state trajectory so that it is not computed twice for each function evaluation.

9 Provided code

Look carefully at the code in `fmcDemo.m` and read the associated published document `fminconDynamicSystemDemo.pdf`. It carries out this example for both Case #1 and #2. The code `fminconDynamicSystemTemplate.m` is general purpose to carry out the approach for Case #1 and Case #2. A user would follow the ideas in `fmcDemo.m` as a starting point for theor own example, leaving the code in `fminconDynamicSystemTemplate.m` unchanged.