ECE-597: Optimal Control
Homework #1

Due: Tuesday September 11, 2007

Problems

1. (*Rectangle of Maximum Perimeter Inscribed in an Ellipse, from Bryson*) Show that the x, y that maximize $P = 4(x + y)$ subject to the constraint

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} \;=\; 1$$

are given by

$$x = \frac{a^2}{\sqrt{a^2 + b^2}} \qquad y = \frac{b^2}{\sqrt{a^2 + b^2}}$$

2. (*Rectangular Parallelepiped of Maximum Volume Contained in an Ellipsoid, from Bryson*) Show that the x, y, z that maximize $V = 8xyz$ subject to the constraint

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} \;=\; 1$$

are given by

$$x = \frac{a}{\sqrt{3}} \quad y = \frac{b}{\sqrt{3}} \quad z = \frac{c}{\sqrt{3}}$$

3. Consider the problem of finding the vector x of minimum length that satisfies the equation $Ax = b$. Assume this is an *underdetermined system*, i.e., there are more rows than columns in A. Show that the optimal x is given by

$$x \;=\; A^T(AA^T)^{-1}b$$

4. (*General Quadratic Performance Index with Linear Equality Constraints, from Bryson*) Consider the problem of finding the parameter vectors x and u to minimize

$$L \;=\; \frac{1}{2}\left(x^T Q x + u^T R u\right)$$

subject to

$$x + Gu \;=\; c$$

Assume Q and R are symmetric and are both invertible.

i) Form the Hamiltonian and by setting $H_x = H_u = 0$, solving for λ in each equation, and then substituting the result into the constraint we get

$$\begin{aligned} S &= \left(Q^{-1} + GR^{-1}G^T\right)^{-1} \\ x &= Q^{-1}Sc \\ u &= R^{-1}G^TSc \end{aligned}$$

ii) Show that $L_{min} = \frac{1}{2}c^TSc$.
(*Hint: Show that $x^TQx = -\lambda^Tx$ and $u^TRu = -\lambda^TGu$, put these into L, and factor out the λ^T. Finally, use the fact that S is symmetric.*)

Read the Appendix before completing the following problems.

5. Run each of the routines as it is, and verify that you get the correct answer. Try an initial guess of [1 9 0 0] and [1 1 0 0]. Do you get the correct answers? For most optimization routines, you need to have a reasonably good starting point or you will find a *local* minima instead of a *global* minima.

6. Modify all three programs to solve the following problem:

Determine the point on the ellipse

$$\left(\frac{x}{p}\right)^2 + \left(\frac{y-r}{q}\right)^2 = 1$$

closest to the parabola

$$y = sx^2$$

where $p = 3$, $q = 1$, $r = 2$, and $s = 0.1$. Show all work (computation of derivatives) and turn in your code (and answers!) You should try a number of different starting points to try and be sure to find the global minimum. It will probably help if your initial guess satisfies both equations.

7. We would like to solve the following discrete-time problem:

minimize $L(u) = u^TRu$

subject to $x(k+1) = \phi x(k) + \gamma u(k)$ for $k = 0..N-1$

where $u = [u_0\ u_1\ ...\ u_{N-1}]^T$ and ϕ, γ, $x(0)$, $x(N)$, and N are known.
To solve this, we need to make the problem look a bit more like something we know.
(i) Show that:

$$\begin{aligned} x(N) &= \phi^N x(0) + \left[\phi^{N-1}\gamma\ \phi^{N-2}\gamma\ ...\ \phi\gamma\ \gamma\right] u \\ &= \phi^N x(0) + \mathbf{M}u \end{aligned}$$

(ii) Determine an expression for $f(u)$

(iii) Determine all necessary derivatives the three algorithms

(iv) Implement the problems for all three algorithms, and solve it assuming $N = 5$, $x(0) = [0\ 0]^T$, $x(5) = [1.5\ -0.5]^T$, $\phi = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$, and $\gamma = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $R = diag([1:5])$ Note that you will have to change fmincon a bit...

Hint: To find $\mathbf{M}$ *and* ϕ^N *you can use code like*

```
M = gamma;
temp = phi;
for k = 1:N-1
  M = [temp*gamma M];
  temp = temp*phi
end;
beq = xN-temp*x0
```

Appendix

POP, POPN, and fmincon Routines

Example Consider the problem of trying to find the point on the circle $x^2 + y^2 = 1$ closest to the ellipse $(\frac{x}{a})^2 + (\frac{y}{b})^2 = 1$, where $a = 10$ and $b = 2$.

In order to solve this we need to reformulate the problem using different x's and y's for the ellipse and circle. The optimization routines we will use solves for a vector, so let's let $u = [x_1\ x_2\ y_1\ y_2]^T$. We now want to find the point on the unit circle $x_1^2 + y_1^2 = 1$ closest to the ellipse $(\frac{x_2}{a})^2 + (\frac{y_2}{b})^2 = 1$. The function we want to minimize is the distance (squared) between points, so

$$L(u) \quad = \quad (x_1 - x_2)^2 + (y_1 - y_2)^2$$

subject to the constraints

$$f(u) \quad = \quad \left[\begin{array}{c} f_1(u) \\ f_2(u) \end{array} \right] = \left[\begin{array}{l} f_1(u) = x_1^2 + x_2^2 - 1 \\ f_2(u) = (\frac{x_2}{a})^2 + (\frac{y_2}{b})^2 - 1 \end{array} \right]$$

The algorithms we will use need various information, so we'll compute those things now

$$\begin{aligned}
L_u &= \frac{dL}{du} = [2(x_1 - x_2) \ \ -2(x_1 - x_2) \ \ 2(y_1 - y_2) \ \ -2(y_1 - y_2)] \\
L_{uu} &= \frac{d^2L}{du^2} = \left[\begin{array}{c} \frac{\partial}{\partial x_1}\frac{dL}{du} \\ \frac{\partial}{\partial x_2}\frac{dL}{du} \\ \frac{\partial}{\partial y_1}\frac{dL}{du} \\ \frac{\partial}{\partial y_2}\frac{dL}{du} \end{array} \right] = \left[\begin{array}{cccc} 2 & -2 & 0 & 0 \\ -2 & 2 & 0 & 0 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & -2 & 2 \end{array} \right] \\
f_u &= \frac{df}{du} = \left[\begin{array}{c} \frac{df_1}{du} \\ \frac{df_2}{du} \end{array} \right] = \left[\begin{array}{cccc} 2x_1 & 0 & 2y_1 & 0 \\ 0 & \frac{2x_2}{a^2} & 0 & \frac{2y_2}{b^2} \end{array} \right] \\
f_{uu} &= \left[\begin{array}{c} \frac{d^2f_1}{du^2} \\ \frac{d^2f_2}{du^2} \end{array} \right] = \left[\begin{array}{c} \frac{\partial}{\partial x_1}\frac{df_1}{du} \\ \frac{\partial}{\partial x_2}\frac{df_1}{du} \\ \frac{\partial}{\partial y_1}\frac{df_1}{du} \\ \frac{\partial}{\partial y_2}\frac{df_1}{du} \\ \frac{\partial}{\partial x_1}\frac{df_2}{du} \\ \frac{\partial}{\partial x_2}\frac{df_2}{du} \\ \frac{\partial}{\partial y_1}\frac{df_2}{du} \\ \frac{\partial}{\partial y_2}\frac{df_2}{du} \end{array} \right] = \left[\begin{array}{cccc} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{2}{a^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{b^2} \end{array} \right]
\end{aligned}$$

POP Routine

The POP routine determines the optimal value of a parameter u for a function L when there are equality constraints $f(u) = 0$ using a *gradient* algorithm. In order to use this algorithm, you must first write a Matlab function that, given the current value of u, determines the value of L, f, L_u (the derivative of L with respect to u) and f_u (the derivative of f with

respect to u). For best performance u should be normalized so the change of one unit of each element of u has approximately equal significance.

This function should look something like this:

```
function [L, f, Lu, fu] = bobs_pop(u);

... stuff....

return;
```

The arguments to POP are the following:

- the function you just created (in single quotes)
- the initial guess for the value of u that minimizes L and (hopefully) satisfies $f(u) = 0$, though this is not necessary
- the value k, a scalar step size parameter. Choose $k > 0$ for a minimum, $k < 0$ for a maximum. If $|k|$ is too small, convergence will be very slow, while if $|k|$ is too large the algorithm is likely to overshoot the minimum (or maximum)
- the value of η, where $0 < \eta \leq 1$. If η is one the constraints are satisfied in one time step, so smaller values of η allow the program to gradually satisfy the constraints.
- the stopping tolerance. This depends on your problem.
- $mxit$, the maximum number of iterations to try.

To see an illustration of this routine for this problem, look at **bobs_pop_example.m** and **bobs_pop_example_driver.m** on the class website.

POPN Routine

The POPN routine determines the optimal value of a parameter u for a function L when there are equality constraint $f(u) = 0$ using a *Newton-Raphson* algorithm. In order to use this algorithm, you must first write a Matlab function that, given the current value of u, determines the value of L, f, L_u, f_u, L_{uu}, and f_{uu}. For best performance u should be normalized so the change of one unit of each element of u has approximately equal significance. This algorithm will generally converge quickly if the starting point is close enough to the optimum.

This function should look something like this:

```
function [L, f, Lu, fu, Luu, fuu ] = bobs_popn(u);

... stuff....

return;
```

The arguments to POPN are the following:

- the function you just created (in single quotes)
- the initial guess for the value of u that minimizes L and (hopefully) satisfies $f(u) = 0$, though this is not necessary.
- the stopping tolerance. This depends on your problem.
- $mxit$, the maximum number of iterations to try.

To see an illustration of this routine for this problem, look at **bobs_popn_example.m** and **bobs_popn_example_driver.m** on the class website.

fmincon Routine (From the Matlab Optimization Toolbox)

fmincon minimizes constrained nonlinear functions, subject to a variety of possible constraints. See the Matlab doc files for this function, as well as for **optimset**, to set some of the options.

To see an illustration of this routine for this problem, look at **bobs_fmincon_example.m** on the class website.