Problem Set 6

Due: 4:30PM, Friday April 5, 2002

Problem 1. Bus systems (last time, we promise) 100 points

Problem description

The analytical equations we’ve been using to find the optimal bus systems in problem sets 1 and 5 are approximate solutions of a system of nonlinear equations. In this homework we will obtain numerical solutions of these equations.

We use a simpler set of equations, based on Cartesian rather than polar coordinates. We have a set of parallel bus routes operating in a region of uniform density. All trips are bound to or originate from a point beyond the region (typically a downtown area) that the bus routes serve. The bus routes are spaced a distance $g$ apart, operate at a headway $h$ and charge a fare $f$. Users are uniformly distributed in the area of dimension $X$ times $Y$, and walk in a perpendicular direction to the nearest bus route. We ignore bus stop spacing along the routes, the dimensions of the street grid, etc. All of those can be handled but make the model more complex. This is essentially the same model as that of homework 1 and 5 except that the bus routes are local, not express; the users walk rather than drive to the bus stop; and Cartesian coordinates are used; all these simplify the model.

To minimize deficit (or maximize profit), we maximize the difference of revenues minus costs. Revenues are

$$\text{Revenue} = TpXYf(a_0 - a_2(kh + g/(4j)) - a_4f)$$
\[ \text{Costs} = 2XTcY/(ghv) \]

where

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Value</th>
<th>Definition</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p)</td>
<td>3.59</td>
<td>Trip density</td>
<td>Trips/mi²/day</td>
</tr>
<tr>
<td>(j)</td>
<td>0.05</td>
<td>Walk speed</td>
<td>Miles/minute</td>
</tr>
<tr>
<td>(k)</td>
<td>0.4</td>
<td>Wait/headway ratio</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>50</td>
<td>Bus operating cost</td>
<td>Cents/minute</td>
</tr>
<tr>
<td>(T)</td>
<td>1050</td>
<td>Length of day</td>
<td>Minutes</td>
</tr>
<tr>
<td>(V)</td>
<td>0.167</td>
<td>Bus speed</td>
<td>Miles/minute</td>
</tr>
<tr>
<td>(a_0)</td>
<td>0.41</td>
<td>Bus market share if equal service as auto</td>
<td></td>
</tr>
<tr>
<td>(a_2)</td>
<td>0.0081</td>
<td>Bus wait time coefficient</td>
<td></td>
</tr>
<tr>
<td>(a_4)</td>
<td>0.0014</td>
<td>Bus fare coefficient</td>
<td></td>
</tr>
<tr>
<td>(X)</td>
<td>4.0</td>
<td>Width of analysis area</td>
<td>Miles</td>
</tr>
<tr>
<td>(Y)</td>
<td>6.0</td>
<td>Length of analysis area</td>
<td>Miles</td>
</tr>
</tbody>
</table>

Note that the values of most of the parameters are different than in previous homeworks.

The total number of trips by all modes of transport (bus, auto, etc.) is \(TpXY\), or the trip density times the area and the time period. The number of bus trips is the bus market share times the total trips; the bus market share is \(a_0-a_2(kh+g/(4j))-a_4f\). This is a linear approximation that estimates the market share as a function of the headway \(h\), the route spacing \(g\) (which determines the average walking distance to the bus route), and the fare \(f\). The bus revenue is the number of bus trips times the fare \(f\).

The cost of the bus service is derived as: There are \(X/g\) routes, each operating \(T/h\) trips, each taking \(Y/v\) minutes to complete, multiplied by a round trip factor (2) and the operating cost per minute \(c\).

The fare \(f\) is in cents. The headway \(h\) is in minutes between bus departures. The route spacing \(g\) is in miles between routes.

In this homework we will find the optimal values of route spacing \(g\), headway \(h\) and fare \(f\). To do so, we take the derivative of the profit (deficit) function \(Q\) with respect to \(g\), \(h\) and \(f\). (These are partial derivatives but we use the ‘d’ symbol due to html not supporting the partial derivative symbol.)

\[ Q = TpXYf(a_0-a_2(kh+g/(4j))-a_4f) - 2XTcY/(ghv) \]
\[ \frac{dQ}{dg} = -T_p X Y a_2/(4j) + 2X T_c Y/(v h g^2) = 0 \]  \hspace{1cm} (1) \\
\[ \frac{dQ}{dh} = -T_p X Y a_2 k + 2X T_c Y/(v g h^3) = 0 \]  \hspace{1cm} (2) \\
\[ \frac{dQ}{df} = T_p X Y (a_0 - a_2(kh + g/(4j)) - 2a_4 f) = 0 \]  \hspace{1cm} (3)

Note that \( T, X \) and \( Y \) drop out of all the equations. Also note that equations (1) and (2) are very similar and yield a linear relationship between route spacing \( g \) and headway \( h \): 

\[ h = g/(4jk) \]  \hspace{1cm} (4) 

Use (4) to eliminate \( h \) from (1):

\[ \frac{dQ}{dg} = -p a_2/(4j) + 8j k c/(v g^3) = 0 \]  \hspace{1cm} (5)

Use (4) to eliminate \( h \) from (3):

\[ \frac{dQ}{df} = a_0 - a_2 g/(2j) - 2a_4 f = 0 \]  \hspace{1cm} (6)

We can solve (5) and (6) analytically, approximately to obtain:

\[ g = (64j^2 k a_4 c/(p v a_0 a_2))^{1/3} \]  \hspace{1cm} (7) \\
\[ f = a_0/(2a_4) - (k c a_2^2)/(j a_4^2 p v a_0))^{1/3} \]  \hspace{1cm} (8)

By using (4) we can also find an approximate solution for headway \( h \):

\[ h = (a_4 c/(p a_0 a_2 k^2 v))^{1/3} \]  \hspace{1cm} (9)

The approximate solutions (7)-(9) have errors. They are usable for initial analysis, but if exact answers are needed, we must solve equations (1)-(3) numerically. Since (2) and (3) are linearly dependent, we only need to solve equations (5) and (6), and can then use (4) to obtain the optimal headway \( h \).

Let’s examine the system of two nonlinear equations that we must solve:

\[-p a_2/(4j) + 8j k c/(v g^3) = 0 \]  \hspace{1cm} (0') \\
\[ a_0 - a_2 g/(2j) - 2a_4 f = 0 \]  \hspace{1cm} (1')

Let the equations be numbered 0 and 1: we will need an array and it will be convenient to number it starting at 0. Let the variables \( g \) be \( x_0 \) and \( f \) be \( x_1 \), to fit the general method for solving this system. Rewrite (0) and (1) using \( x_0 \) and \( x_1 \), and simplify coefficients:

\[ f_0(x_0,x_1) = -0.25p a_2 x_1/j + 8j k c/(v x_0^3) = 0 \]  \hspace{1cm} (0'') \\
\[ f_1(x_0,x_1) = a_0 - 0.5a_2 x_0/j - 2a_4 x_1 = 0 \]  \hspace{1cm} (1'')
We will use the two-dimensional Newton’s method to solve this system. Newton’s method requires the derivatives of each equation with respect to each variable. Again, these are partial derivatives but we use the symbol ‘d’:

\[
\begin{align*}
\frac{df_0}{dx_0} &= -24 j kc/v x_0^4 \\
\frac{df_0}{dx_1} &= -0.25pa2/j \\
\frac{df_1}{dx_0} &= -0.5a2/j \\
\frac{df_1}{dx_1} &= -2a4 
\end{align*}
\]  

The equations (0’) and (1’) can be solved with a two-dimensional version of Newton’s method, using the derivatives (a)-(d).

**Newton’s method in two dimensions**

We have two functions \(f_0(x_0, x_1)\) and \(f_1(x_0, x_1)\) and we wish to approximate a solution to the system \(f_0 = 0\) and \(f_1 = 0\). In the neighborhood of \(x_i\) each function \(f_i\) can be expanded using a Taylor series:

\[
\begin{align*}
f_0(x_0 + dx_0, x_1 + dx_1) &= f_0(x_0, x_1) + (df_0/dx_0) \, dx_0 + (df_0/dx_1) \, dx_1 + \ldots \quad (e) \\
f_1(x_0 + dx_0, x_1 + dx_1) &= f_1(x_0, x_1) + (df_1/dx_0) \, dx_0 + (df_1/dx_1) \, dx_1 + \ldots \quad (f)
\end{align*}
\]

By setting \(f_0(x_0 + dx_0, x_1 + dx_1) = 0\) and \(f_1(x_0 + dx_0, x_1 + dx_1) = 0\), we obtain, in matrix notation:

\[
\begin{bmatrix}
\frac{df_0}{dx_0} & \frac{df_0}{dx_1} \\
\frac{df_1}{dx_0} & \frac{df_1}{dx_1}
\end{bmatrix}
\begin{bmatrix}
dx_0 \\
dx_1
\end{bmatrix}
= \begin{bmatrix}
-f_0(x_0, x_1) \\
f_1(x_0, x_1)
\end{bmatrix}
\]

You can see that this is the two-dimensional extension to the Newton’s method covered in lecture.

We now have to solve the matrix equation (g) for \(dx_0\) and \(dx_1\), the step sizes to move toward the solutions of the two equations.

While we cover methods for solving general linear systems this week in class, for this homework we can use a simple method you should recall from algebra called Cramer’s rule. In two dimensions it is particularly simple. For a system:

\[
\begin{bmatrix}
a_{00} & a_{01} \\
a_{10} & a_{11}
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1
\end{bmatrix}
= \begin{bmatrix}
b_0 \\
b_1
\end{bmatrix}
\]

\[
x_0 = (a_{11} \, b_0 - a_{01} \, b_1) / (a_{00} \, a_{11} - a_{01} \, a_{10}) \quad (j)
\]

\[
x_1 = (-a_{10} \, b_0 + a_{00} \, b_1) / (a_{00} \, a_{11} - a_{01} \, a_{10}) \quad (l)
\]
Since the partial derivatives \( \frac{df_i}{dx_j} \) and the function values \( f_i(x_0, x_1) \) are known, we can solve for the step sizes \( dx_0 \) and \( dx_1 \) using (k) and (l).

**Assignment**

Write a program to solve equations (0’’) and (1’’) numerically for optimal values of route spacing \( g \) and fare \( f \). Use equation (4) to find the headway \( h \). Details:

1. Write a 2D Newton’s class and method similar to the 1D Newton’s method covered in lecture:
   - Define an interface that contains the appropriate methods that Newton’s method requires. A suggested interface is given below, but you are free to use your own.

   In the function below, note the arguments \( jj \) and \( kk \). The extra arguments \( jj \) and \( kk \) take on the values 0 and 1, corresponding to the two dimensions of our inputs.

   We need 1 extra argument in the \( f() \) method to choose which dimension of the function we are evaluating.

   We need 2 extra arguments in \( df() \) to choose both the dimension we are partially differentiating (\( jj \)), and the dimension we are partially differentiating with respect to (\( kk \)).

   ```java
   public interface MathFunction2D
   {
       // Function value for function j
       public double f(int jj, double[] x);

       // Partial derivative, function j, variable k
       public double df(int jj, int kk, double[] x);
   }
   ```

   - Set the maximum iterations to 50 and the tolerance to \( 10^{-15} \). The tolerance is silly for this problem, but it’s generally good practice to set it low.
   - The code is quite short for the method: it computes the function and derivative values and invokes Cramer’s rule to solve for the step size \( dx \), until the absolute value of both functions is less than the tolerance.
   - Note that Cramer’s rule solves for the step \( dx \), not the new \( x \); you need to add the step to the current estimate of \( x \) to get the next estimate. Be careful with sign conventions; note that the function values in equation (g) are negative, for example. Check that the denominator in equations (k) and (l) is not too small (or zero).
   - You can implement Cramer’s rule within the body of the 2D Newton’s method because it’s so simple. In a more general, \( n \)-dimensional Newton’s method, you would use a Gaussian elimination method to solve the linear system.
   - Don’t use any integer loop counters called \( j \) or \( k \); they will conflict with the double variables \( j \) and \( k \).

2. Write a class to generate the functions, derivatives and initial guesses. You’ll find it convenient to place these in one class since they share many parameters. The class should:
- Implement the MathFunction2D (or your own) interface.
- Implement the methods to compute the approximate analytic solutions using (7), (8), and (9). Use the approximate solutions from (7) and (8) as your initial guesses for your 2D Newton's method.
- Write any setXXX() and getXXX() methods needed (see below)

3. Write a test class with a main() method (or include the main() in another class):

   - Allow the user to input a range of densities $p$ to obtain optimal route spacing, headway and fare. You can assume an increment of 0.5 units in density. A typical density range would be from 2.0 to 4.0.
   - For each density, output the approximate and numerical solutions for route spacing, headway and fare. Also show the initial guesses, the number of iterations and the value of the two functions at the end of the numerical methods.
   - Also, test Newton’s method with initial guesses for both variables of:
     1. 0
     2. 0.0001
     3. 10
   - You need not write a separate main to perform these initial guess tests: you may just temporarily modify your existing main(). Though you don’t need to submit this version of the code, you must include a brief comment at the end of the regular main() method, describing what happens and why.

Extra Credit (GUI solution) 40 points

In problem sets 6-10, you may implement a graphical user interface in Swing for 40 extra credit points. You may do this in one and only one problem set; please place a comment at the top of your main() method stating that you are doing so.

For the extra credit part of problem set 6, implement a GUI that allows the user to choose a single density (using a slider control, ranging from 2.0 to 4.0) and then displays the numerical results of the program as text in the GUI. Display the same quantities as specified in the problem statement. If you wish, you may draw the routes and their spacing as shown in the figure in the problem set but this is not required.

An example of what it should look like is as follows:
Turnin

Turnin Requirements

- Problem 1: Email only. No hardcopy required.
- Problems 2 & 3: Hardcopy and electronic copy of ALL source code (all .java files).
- Place a comment with your name, username, section, TA's name, assignment number, and list of people with whom you have discussed the problem set on ALL files you submit.
- DO NOT turn in electronic or hardcopies of compiled byte code (.class files).

Electronic Turnin

To electronically turn in your problem sets, run Netscape

Then go to the 1.00 web page at:

http://command.mit.edu/1.00Spring02

Click on the "Submit Assignment" button. Be sure to set the Selection Bar to Problem Set 1 or your files may be lost. Finally, go back to the home page and click on the "View" section and be sure that your files were received. If you submit a file twice, the latest version will be graded.

Penalties

- Missing Hardcopy: -10% off problem score if missing hardcopy.
- Missing Electronic Copy: -30% off problem score if missing electronic copy.

Late Turnin: -20% off problem score if 1 day late. More than 1 day late = NO CREDIT