Lecture 11 Solving Non-linear Programming Problems

Zidong Mark Wang
2015 Fall

Based on material written by Gillig and McCarl; Improved upon by many previous lab instructors; Special thanks to Pei Huang.
Introduction

• We often encounter problems that cannot be solved by LP algorithms, in which the objective function or constraints are in non-linear forms.

• Solving non-linear programming models are generally much complicated than linear programming models based on the types of non-linear models. For example:
  – The objective is linear but constraints are non-linear
  – Non-linear objectives and linear constraints
  – Non-linear objectives and constraints
Approximation of NLP problem

• One important intuition in solving NLP problems is approximation.
• A continuous function \( f(x) \) can be approximated by a polynomial to any accuracy desired.
• Linear approximation of a NLP problem. GAMS handles linear problems faster and more efficient.
• Besides, GAMS has some nice NLP solvers which can be applied to NLP models directly.
Using NLP solvers in GAMS

• As before
  – Define sets
  – Data assignment
  – Define variables
  – Define equations (now non-linear)
  – Model statement

• Except now the last step is: Solve modelname using NLP maximizing/minimizing objective
Price Endogenous Problem Example

• Demand:  \( P_d = a_d - b_d Q_d \)
• Supply:  \( P_s = a_s - b_s Q_s \)

• The mathematical formation for this problem is:

\[
\begin{align*}
\text{Max} & \quad a_d \quad Q_d - \frac{1}{2} b_d Q_d^2 - a_s \quad Q_s - \frac{1}{2} b_s Q_s^2 \\
\text{s.t.} & \quad Q_d - Q_s \quad \leq 0 \\
& \quad Q_d, Q_s \quad \geq 0
\end{align*}
\]

• The problem maximizes welfare (consumer surplus + producer surplus).
GAMS code

SETS        CURVEPARM  CURVE PARAMETERS  /INTERCEPT,SLOPE/
            CURVES      TYPES OF CURVES  /DEMAND,SUPPLY/

TABLE        DATA(CURVES,CURVEPARM)  SUPPLY DEMAND DATA
INTERCEPT    SLOPE
DEMAND       6   -0.30
SUPPLY       1   0.20

POSITIVE VARIABLES  Q(CURVES)  ACTIVITY LEVEL
VARIABLES         OBJ        NUMBER TO BE MAXIMIZED
EQUATIONS         OBJJJ      OBJECTIVE FUNCTION
                  BALANCE    COMMODITY BALANCE;

OBJJJ..  OBJ =E=
          DATA("demand","INTERCEPT")*Q("demand")
          +0.5*DATA("demand","SLOPE")*Q("demand")**2
          - (DATA("supply","INTERCEPT")*Q("supply")
          +0.5*DATA("supply","SLOPE")*Q("supply")**2);

BALANCE..  Q("demand")-q("supply")=L= 0;

MODEL PRICEEND  /ALL/;
SOLVE PRICEEND  USING  NLP  MAXIMIZING OBJ;

\[
\begin{align*}
\text{max} \quad & 6Q_d - 0.15Q_d^2 - Q_s - 0.1Q_s^2 \\
Q_d, & \quad -Q_s \quad \leq 0 \\
Q_d, & \quad Q_s \quad \geq 0
\end{align*}
\]
NLP Solvers

- Gradient based search
  - With a starting point and a direction, the algorithm repeatedly search for optimal.

- Potential problems
  - Solver may only find a local solution.
Starting Points

- Initial values for variables

**Syntax**

```
variablename.L(setdependency)=startingvalue;
```

- Default starting point is zero or the variable lower bound.
- Starting point in right neighborhood is likely to return a desirable solution.
- Initial values close to optimal one reduces work required to find the optimal solution.
- Poor initial values can lead to numerical problems. Starting points can help avoid such.
Starting points - example

Suppose we solve a simple CS-PS maximization problem without specifying starting points

```plaintext
set sd /supply,demand/;
set params /price,quantity,elasticity,const,flex/
parameter signit(sd) /demand 1,supply -1/;

table dsparam(sd,params)
     price  quantity  elasticity
supply    10    20     0.5
demand    10    20     -1.5;
dsparam(sd,"flex")=1/dsparam(sd,"elasticity");
dsparam(sd,"const")
    =dsparam(sd,"price")/(dsparam(sd,"quantity")**dsparam(sd,"flex"));
display dsparam;

POSITIVE VARIABLES  Q(sd);
variable csps;
EQUATIONS          obj,balance;

obj..  csps=e=  sum(sd,signit(sd)*((1/(dsparam(sd,"flex")+1))
             *dsparam(sd,"const")*Q(sd)**(dsparam(sd,"flex")+1)));
balance..  sum(sd,signit(sd)*Q(sd))=1=0;
MODEL PROBLEM /ALL/;
SOLVE PROBLEM USING nlp maximizing csps;
```
- The NLP solver takes 22 iterations to find the solution.

<table>
<thead>
<tr>
<th>SOLVE SUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODEL</td>
</tr>
<tr>
<td>TYPE</td>
</tr>
<tr>
<td>SOLVER</td>
</tr>
<tr>
<td>**** SOLVER STATUS</td>
</tr>
<tr>
<td>**** MODEL STATUS</td>
</tr>
<tr>
<td>**** OBJECTIVE VALUE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESOURCE USAGE, LIMIT</th>
<th>0.015</th>
<th>1000.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITERATION COUNT, LIMIT</td>
<td>22</td>
<td>2000000000</td>
</tr>
<tr>
<td>EVALUATION ERRORS</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CONOPT 3</td>
<td>Jul 4, 2012</td>
<td>23.9.1 WEX 33924.33953 WEI x86_64/MS Windows</td>
</tr>
</tbody>
</table>
If we specify a proper starting points before solving the model.

```plaintext
set sd /supply,demand/;
set params /price,quantity,elasticity,cons,flex/
parameter signit(sd) /demand 1,supply -1/;

table dsparam(sd,params)
<table>
<thead>
<tr>
<th>price</th>
<th>quantity</th>
<th>elasticity</th>
</tr>
</thead>
</table>
supply | 10     | 20       | 0.5        |
demand | 10     | 20       | -1.5       |

dsparam(sd,"flex")=1/dsparam(sd,"elasticity");
dsparam(sd,"cons")
   =dsparam(sd,"price")/(dsparam(sd,"quantity")**dsparam(sd,"flex"));
display dsparam;

POSITIVE VARIABLES   Q(sd);
variable csps;
EQUATIONS           obj, balance;

obj.. csps=e= sum(sd,signit(sd)*(1/(dsparam(sd,"flex")+1))
          *dsparam(sd,"cons")*Q(sd)**(dsparam(sd,"flex")+1));
balance.. sum(sd,signit(sd)*Q(sd))=l=0;
MODEL PROBLEM /ALL/;
Q.. L(sd)=1;
SOLVE PROBLEM USING nlp maximizing csps;
```
The NLP solver only takes 8 iterations to find the solution.
Upper and lower bounds

- Set upper bound and lower bound for specific variables
- More realistic
- Keep algorithm in a range
- Improve solution feasibility and pre-solve performance

**Syntax**

```
variablename.LO(setdependency) = lowerbound;
variablename.UP(setdependency) = upperbound;
```
Linear Approximation

- Generally, GAMS takes much more time to solve a NLP problem than a LP problem.
- We sometimes can linearly approximate the NLP problem, and then solve it as a LP problem.
- A typical example in our class is the MOTAD
MOTAD: Linear Approximation

• Linear Approximation to the expected value variance (E-V) problem, which is generally quadratic
  – LP not appropriate without some adjustments
  – MOTAD depicts tradeoffs in expected income and absolute deviations of income

• Minimization of Total Absolute Deviations
The algebraic form of MOTAD

Max

\[ \text{inc} - \Phi \left\{ \sum_k p_k \left[ (d^+_k)^2 + (d^-_k)^2 \right] \right\}^{0.5} \]

s.t.

\[ \sum_j c_{kj} X_j \]

\[ \sum_j c_{kj} X_j - Inc_k \]

\[ \sum_k p_k Inc_k - \overline{Inc} \]

\[ Inc_k - \overline{Inc} \]

\[ X_j, \]

\[ Inc_k, \overline{Inc} \]

\[ \leq b_i \text{ for all } i \]

\[ = 0 \text{ for all } k \]

\[ = 0 \]

\[ d^+_k + d^-_k = 0 \text{ for all } k \]

\[ d^+_k, d^-_k \geq 0 \text{ for all } j, k \]

\[ > 0 \text{ for all } k \]
SETS
STOCKS POTENTIAL INVESTMENTS /BUYSTOCK1*BUYSTOCK4 /
EVENTS EQUALLY LIKELY RETURN STATES OF NATURE /EVENT1*EVENT10 /

PARAMETERS
PRICES(STOCKS) PURCHASE PRICES OF THE STOCKS
/ BUYSTOCK1 22
  BUYSTOCK2 30
  BUYSTOCK3 28
  BUYSTOCK4 26 /

SCALAR
Funds TOTAL INVESTABLE FUNDS / 500 /
RAP RISK AVERSION PARAMETER / 0.0 /
PI /3.141716/
N SAMPLE SIZE
TRAN TRANSFORMATION COEF MAD TO STD ERROR ;

N = CARD(EVENTS) ;
TRAN = ((PI * N)/(2*(N-1)))**0.5 ;
<table>
<thead>
<tr>
<th>EVENT 1</th>
<th>BUYSTOCK1</th>
<th>BUYSTOCK2</th>
<th>BUYSTOCK3</th>
<th>BUYSTOCK4</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT2</td>
<td>8</td>
<td>4</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>EVENT3</td>
<td>4</td>
<td>8</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>EVENT4</td>
<td>5</td>
<td>9</td>
<td>-2</td>
<td>7</td>
</tr>
<tr>
<td>EVENT5</td>
<td>6</td>
<td>7</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>EVENT6</td>
<td>3</td>
<td>10</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>EVENT7</td>
<td>2</td>
<td>12</td>
<td>-2</td>
<td>6</td>
</tr>
<tr>
<td>EVENT8</td>
<td>5</td>
<td>4</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>EVENT9</td>
<td>4</td>
<td>7</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>EVENT10</td>
<td>3</td>
<td>9</td>
<td>-5</td>
<td>6</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>OBJ</td>
<td>NUMBER TO BE MAXIMIZED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RETURN</td>
<td>RETURNS BY EVENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EVENT(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MEAN</td>
<td>MEAN RETURNS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EQUATIONS</th>
<th>OBJJ</th>
<th>OBJECTIVE FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RETURNDEF</td>
<td>RETURNS DEFINITION</td>
</tr>
<tr>
<td></td>
<td>(EVENTS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AVRET</td>
<td>AVERAGE RETURNS</td>
</tr>
<tr>
<td></td>
<td>INVEST</td>
<td>INVESTMENT FUNDS AVAILABLE</td>
</tr>
<tr>
<td></td>
<td>TAV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEVIATION</td>
<td>DEVIATIONS FROM MEAN INCOME</td>
</tr>
<tr>
<td></td>
<td>(EVENTS)</td>
<td></td>
</tr>
</tbody>
</table>
OBJJ..

OBJ
=E=
MEAN
- TRAN*RAP*SUM(EVENTS, POSDEV(EVENTS) + NEGDEV(EVENTS)) / CARD(EVENTS);

INVESTAV..

SUM(STOCKS, PRICES(STOCKS) * INVEST(STOCKS))
=L= FUNDS;

RETURNDEF(EVENTS)..

SUM(STOCKS, RETURNS(EVENTS, STOCKS) * INVEST(STOCKS))
- RETURN(EVENTS) =E= 0;

AVRET..

SUM(EVENTS, 1/CARD(EVENTS) * RETURN(EVENTS)) - MEAN
=E= 0;

DEViation(EVENTS)..

RETURN(EVENTS) - MEAN - POSDEV(EVENTS) + NEGDEV(EVENTS)
=E= 0;

MODEL MTDPortfol /ALL/;
SOLVE MTDPortfol USING LP MAXIMIZING OBJ;
Questions?