1 Introduction

The article is aimed at assisting users interested in writing KKT conditions of their formulation and convert the optimization problem into complimentarity problem. Though GAMS allows users to solve non-linear problem (NLP) as an Mixed Complimentarity Problem (MCP) using the EMP solver, it is often advantageous for users to explicitly model the KKT conditions and the respective marginals in their model. This article focuses on providing a structured approach towards reformulating NLP model as MCP model by systematically introducing the KKT conditions, with marginals and their bounds. The article can be extended to help debug an existing KKT system in model.

Consider the simple example described in the PATH solver manual using the example of a Linear Transportation model. The traditional model of fixed demand and price is made more realistic by making the variables endogenous, i.e. the price of commodity is market affects the demand of the commodity, and the model is solved as a complimentarily problem. At times the complexity of the model leads to errors in formulation of optimality (KKT) conditions, leading to incorrect or infeasible solutions. Finding the source of the error can be a particularly tedious task. The systematic scheme described in the article can help users formulate error free MCP models for their NLP systems.

For a standard MCP tasked with finding a solution vector $z \in R$ that is complimentary to a function $F(z) : R^n \mapsto R^n$, lower bounds $l \in R \cup -\infty^n$ and upper bounds $u \in R \cup \infty^n$ for each $i \in 1, ..., n$, if one of the following three condition holds hen function F is said to complementary to the variable z and its bounds:

$$\begin{gathered} F_i(z) = 0 \text{ and } l_i \leq z_i \leq u_i \text{ or} \\ F_i(z) > 0 \text{ and } z_i = l_i \text{ or} \\ F_i(z) < 0 \text{ and } z_i = u_i \end{gathered}$$

This is written in compact form as: $F(z) \perp L \leq z \leq U$

Where the symbol $\perp$ means "perpendicular to". For a NLP model, the optimization constraints and their respective marginals must be perpendicular to each other for an optimum solution to exist. Consider the following generalized NLP formulation.

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & g_i(x) \leqslant 0 & i = 1, 2...m \\ & h_k(x) = 0 & k = 1, 2...p \\ & d_l(x) \geqslant = 0 & l = 1, 2...q \\ & L \leq x \leq U & \\ & f(x) : R^n \mapsto R, g(x) : R^n \mapsto R^m & \\ & h(x) : R^n \mapsto R^p, d(x) : R^n \mapsto R^p & \end{array} \tag{1}$$

The Lagrange function, KKT conditions and their multipliers in complimentarity form can be written as

$$\begin{array}{l} L(x, u, v, w) = f(x) - < u, g(x) > - < v, h(x) > - < w, d(x) > \\ \bigtriangledown_x L \perp L_x \leq x \leq U_x \\ -\bigtriangledown_u L \perp u \geq 0 \\ -\bigtriangledown_v L \perp v free \\ -\bigtriangledown_w L \perp w \leq 0 \end{array} \tag{2}$$

It should be noted that the gradient of Lagrangian w.r.t to the marginals from NLP is the constraint itself. Thus an NLP can be solved as an MCP using the KKT conditions. However, formulating the KKT conditions might prove difficult for complex systems, and require verification of their accuracy. In the following sections, we provide the framework for modeling the KKT conditions by using concept of dummy complementarity equations in the KKT formulation. The process includes the following steps:

1. Solve NLP model formulation, and save the results using Save and Restart Feature
2. Reformulate the model as an MCP by adding dummy KKT conditions into the model.

3. Solve the MCP using solution from step 1 as initial point. The MCP should start at the solution itself
4. Replace one dummy equation at a time with one KKT condition. Resolve the NLP and save the results
5. Restart the problem as MCP with iterlim=0. Objective value should match the results from NLP.

Follow steps 3 and 4 until all KKT conditions have been successfully incorporated.

2 Example : Maximum Revenue- NLP formulation

Consider a simple case of a steel factory trying to maximize the revenue under budget constraints, with man-hours(h) and raw materials steel (s) as the decision variables. The revenue is a function of decision variables given by

$$R(h,s) = 200h^{(2/3)}s^{(1/3)}$$

where, budget = $ 20,000, cost of manpower = $ 20 /hr , cost of raw material = $ 170 / tonn The objective is to maximize revenue under the budget and manpower constraints.
As a standard minimization model, the problem can be formulated as:

$$\begin{aligned} \min \quad & R(h,s) = -200h^{2/3}s^{1/3} \\ \text{s.t.} \quad & 20h + 170s \leq 20000 \\ & L < h, s < U \end{aligned} \tag{3}$$

The GAMS program below gives the optimum values of

```
Maximum Revenue, R -51854.82
Man hours, h 666.67
Tons of raw material, s 39.22
```

The results of the program are saved using command line option *"save filename"*

```
$title : sd_nlp.gms MAX REVENUE MODEL, NLP FORMULATION

$ontext
The model below showcases using GAMS to identify / modify the KKT conditions for a given
NLP formulation. Our starting point is a basic minimization NLP model.
$offtext

scalar
    labor 'cost of labor dollar per hour' /20/
    steel 'cost of steel per ton' /170/
    budget 'total budget for production' /20000/
    ;
variables
    h 'man hours in production' /lo 1, up 50000/
    s 'tons of raw material' /lo 1, up 50000/
    R 'Revenue'
    ;
Equations
    con1 'constraint on budget'
    obj 'objective function'
    ;
obj.. R =e= - 200 * h**(2/3) * s**(1/3) ;
con1.. labor*h + steel*s =l= budget;

h.l=10;
```

```
s.l=10;

model khan /con1,obj/;
solve khan using NLP minimizing R;
```

For the given system, the KKT conditions are given as

$$\begin{aligned} L = &- 200h^{2/3}s^{1/3} - con1\_m[20h + 170s - 20000] \\ \bigtriangledown_h L = &- 200 * (2/3)h^{(-1/3)}s^{(1/3)} - con1 \\ &_m * (20) \\ \bigtriangledown_s L : &- 200 * (1/3)h^{(2/3)} * (1/3) * s^{(-2/3)} - con1 \\ &_m * (170) \\ \bigtriangledown_{con1\_m} L : &20 * h + 170 * s - 20000 = 0 \end{aligned} \tag{4}$$

It should be noted that the third KKT equation which is gradient of Lagrangean w.r.t to the marginal, is the inequality budget constraint from original NLP model. The above equations are solved as an MCP model using the results saved in the initial run using command line option *'restart filename'*. In the model below, an error has been made in one of the KKT conditions. Execution of the model terminates due to the abort statement, with declaration *'We did not start at a solution'* as shown in the subsequent log.

```
variables
    con1_m 'marginal value for con1' /up 0/
    ;
Equations
    dLdh 'gradL wrt h'
    dLds 'gradL wrt s'
    ;

con1_m.l=con1.m;
dLdh.. - 200*(2/3) * ([h**(2/3)]/h) * s**(1/3) - con1_m*(200) =n=0;
dLds.. - 200 * h**(2/3) * (1/3)*s**(-2/3) - con1_m*(170) =n=0;

model kkt /dLdh.h,dLds.s,con1.con1_m/;
kkt.iterlim=0;
solve kkt using MCP;

abort $(kkt.objval >1e-5) 'We did not start at a solution';
```

```
--- Job sd_kkt.gms Start 07/26/18 15:56:02 25.1.1 r66732 WEX-WEI x86 64bit/MS Windows
GAMS 25.1.1 Copyright (C) 1987-2018 GAMS Development. All rights reserved
Licensee: Chintan Bhomia, Single User License G180612/0001CN-GEN
          GAMS Development, Fairfax DC14199
          cbhomia@gams.com
--- Starting continued compilation
--- Workfile was generated under GAMS version WEX251-251
--- sd_kkt.gms(17) 2 Mb
--- Starting execution: elapsed 0:00:00.008[LST:27]
--- sd_kkt.gms(42) 3 Mb
--- Generating MCP model kkt[LST:27]
--- sd_kkt.gms(43) 5 Mb
--- 3 rows 3 columns 8 non-zeroes
--- 25 nl-code 4 nl-non-zeroes
--- sd_kkt.gms(43) 3 Mb
--- Executing PATH: elapsed 0:00:00.015[LST:79]
```

```
Reading dictionary...
Reading row data...
Evaluating functions...
Checking model...
Calculating Jacobian...

PATH 25.1.1 r66732 Released May 19, 2018 WEI x86 64bit/MS Windows

3 row/cols, 8 non-zeros, 88.89% dense.

Path 4.7.04 (Sat May 19 15:07:52 2018)
Written by Todd Munson, Steven Dirkse, and Michael Ferris

Major Iteration Log
major minor func grad residual step type prox inorm (label)
    0 0 1 1 3.2033e+02 I 0.0e+00 3.2e+02 (dLdh)

FINAL STATISTICS
Inf-Norm of Complementarity . . 1.5533e+05 eqn: (dLdh)
Inf-Norm of Normal Map. . . . . 4.6669e+02 eqn: (dLdh)
Inf-Norm of Minimum Map . . . . 4.6669e+02 eqn: (dLdh)
Inf-Norm of Fischer Function. . 3.2033e+02 eqn: (dLdh)
Inf-Norm of Grad Fischer Fcn. . 2.7429e+04 eqn: (con1)
Two-Norm of Grad Fischer Fcn. . 2.7429e+04

 ** EXIT - iteration limit.

Major Iterations. . . . 0
Minor Iterations. . . . 0
Restarts. . . . . 0
Crash Iterations. . . . 0
Gradient Steps. . . . . 0
Function Evaluations. . 1
Gradient Evaluations. . 1
Basis Time. . . . . 0.000000
Total Time. . . . . 0.000000
Residual. . . . . 3.203316e+02

--- Restarting execution
--- sd_kkt.gms(43) 2 Mb
--- Reading solution for model kkt[LST:94]
--- Executing after solve: elapsed 0:00:00.085[LST:155]
--- sd_kkt.gms(45) 3 Mb
*** Error at line 45: Execution halted: abort$1 'We should start at a solution'[LST:160]
--- sd_kkt.gms(45) 3 Mb 1 Error
*** Status: Execution error(s)[LST:177]
--- Job sd_kkt.gms Stop 07/26/18 15:56:02 elapsed 0:00:00.087
```

In case *abort* statement is not used, the MCP formulationl will return a solution different from the NLP formulation, indicating error in one of the KKT conditions. From simple observation, we can see that a typographical error was made with the multiplier of *con1_m* in equation *dLdh*, which when corrected provides results consistent with the NLP formulation.

However, mere observation does not help with larger, more complex systems. A strategy, as described in Section 1 is required to effectively correct the model. In the given case, the MCP formulation can be first

solved using dummy KKT conditions shown below. These conditions can then be replaced one at a time with the real KKT equations derived earlier.

```
dLdh.. 37 =n=0 ; h.fx=h.L;

dLds.. 37 + con1_m=n=0; s.fx=s.L;

model kkt /dLdh.h,dLds.s,con1.con1_m/;
```

2.1 Rules for writing dummy KKT conditions

Writing dummy condition, as straightforward as it may seem requires consideration of few salient points to avoid errors from the GAMS compiler.

1. A new variable is needed to represent marginals of each constraint from NLP (eg. $con1\_m$ from above example)
2. Number of dummy equations must equal number variables in the NLP formulation
3. Each dummy KKT condition must be accompanied by its differentiating variable fixed at the var.L value from the save run of NLP
4. Each KKT condition must be modeled with the differentiating variable for the condition. i.e. $dLdh \perp h$, and is modeled as $dLdh.h$ in the model statement
5. All variables from both NLP and MCP formulation must appear in one of the dummy KKT conditions
6. Replacing a dummy equation must be accompanied by un-fixing the value of variable, per the concept of complimentarity.

Since order of replacing dummy quation is irrelevant, it is easy to miss variable in model. For simplicity and to ensure that all variables are part of the model at all times, the variables can be placed in a single dummy equation replaced at the very end.

3 KKT Development for complex systems

In this section, we implement the proposed methodology and inferences from previous example to solve a more complex NLP problem as MCP. The NLP example has been designed as a minimization model, which helps maintaining the sign convention. The complexity applies the methodology over range of equations and contains leads/lags in the formulation. The NLP Model and the output is given below.

```
$title: refer EX2.gms, KKT conditions development
$ontext
This is an advanced model for showcasing development of KKT conditions for an NLP
$offtext

sets
  j 'set j' / j1 * j6 /
  j2(j) 'subset for j' / j2, j4*j6 /
  i 'set i' / i1 * i3 /
  i2(i) 'subset for i' / i1, i2 /
  ;
scalar
  eMin / 2.5 /
  sLow / -100 /
  vMax / 20 /
```

```
  ;
parameter
  c(j) /
    j1 2
    j2 -2
    j3 2
    j4 -2
    j5 2
    j6 -2
  /
  s0(i) /
    i1 10
    i2 20
    i3 -10
  /
  ;
table A(i,j)
    j1 j2 j3 j4 j5 j6
i1 1 4 2 2
i2 4 1 2
i3 -2 -2 -1 4 ;

variable
  z 'objective var'
  ttt(j) 'variable over set j'
  sss(i) 'variable over set i'
  ;
positive variable
  x(j) 'positive variable over set j'
  ;
equation
  objDef
  sssdef(i)
  tttdef(j)
  eSum
  sSum
  allBnd
  ;

objDef.. sum{j, c(j)*x(j)} + sum{i, sqr(sss(i)-s0(i))} =E= z;
sssdef(i).. sss(i) =E= sum{j, A(i,j) * ttt(j)};
tttdef(j).. ttt(j) =E= 4*x(j) + x(j+1);
eSum.. sum{j2(j), exp(x(j)-1)} - eMin =G= 0;
sSum.. sum{i2(i), sss(i)} - sLow =G= 0;
allBnd.. sum{j, x(j) + ttt(j)} + sum{i, sss(i)} -vMax =L= 0;

model nonlinear 'NLP model' / all /;
solve nonlinear using nlp min z;

file EX2 /EX2.txt/
put EX2;
put "Objective Function" , z.l / /;
put 'Var ttt(j)'/;
loop(j, put 'ttt(', @5,j.tl,@7')' ,@14, ttt.l(j)/);
```

```
put /;
put 'Var sss(i)' /;
loop(i, put 'sss(', @5,i.tl,@7')' ,@14, sss.l(i)/);
put /;
put 'Var x(j)' /;
loop(j, put 'x(', @3,j.tl,@5')' ,@14, x.l(j)/);



display "solution: "
display $ord
```

```
Objective Function 75.41

Var ttt(j)
ttt(j1) 0.18
ttt(j2) 0.85
ttt(j3) 0.54
ttt(j4) 1.28
ttt(j5) 5.13
ttt(j6) 0.00

Var sss(i)
sss(i1) 4.64
sss(i2) 13.69
sss(i3) -7.90

Var x(j)
x(j1) 0.00
x(j2) 0.18
x(j3) 0.14
x(j4) 0.00
x(j5) 1.28
x(j6) 0.00
```

As seen in the model, the problem consists of five constraints and one objective variable. We define multiplier variables per the convention for each equations as follows:

- $sssdef\_m(i)$ for equation $sssdef(i)$, free variable initialized at $sssdef.m(i)$
- $tttdef\_m(j))$ for equation $tttdef(j)$, free variable initialized at $tttdef.m(i)$
- $esum\_m$ for equation $esum$, positive variable initialized at $esum.m$
- $ssum\_m$ for equation $ssum$, positive variable initialized at $Ssum.m$
- $allbnd\_m$ for equation $allbnd$, negative variable initialized at $allbnd.m$

The Lagrangian $(L)$of the function is given as

$$L(sss, ttt, x, sss\_m, ttt\_m, esum\_m, ssum\_m, allbnd\_m) = obj - \sum_i sssdef\_m(i).sssdef(i) - \sum_j tttdef\_m(j).tttdef(j) - (esum\_ \tag{5}$$

The gradients of Lagrangian with respect to the marginals are

$$dLdttt(j) = \sum_i sssdef\_m(i)A(i,j) - tttdef\_m(j) - allbnd\_m$$

$$dLdsss(i) = 2(sss(i) - s0(i)) - sssdef(i) - ssum\_m - esum\_m, \quad ssum\_m \forall i2 \in i$$

$$dLdx(j) = c(j) + 4tttdef\_m(j) + tttdef\_m(j-1) - esum\_m \exp^{(x(j)-1)} - allbnd\_m, \quad \exp^{(x(j)-1)} \forall j2 \in j \tag{6}$$

The gradients of Lagrangian with respect to the variables are the constraints themselves. The model with dummy KKT conditions is shown below. When executed with saved solution from NLP model, the model exists at iteration 0 with the message "solution found' as shown by the log below.

```
$title: refer EX2KKT0.gms, KKT conditions development
$ontext
This is an advanced model for showcasing development of KKT conditions for an NLP.
All dummy equations are used.
$offtext


Variables
     sssdef_m(i) 'multipler for equation sssdef(i)'
     tttdef_m(j) 'multipler for equations tttdef(j)'
     esum_m 'multipler for esum' /lo 0 /
     ssum_m 'multipler for ssum' /lo 0/
     allbnd_m 'multipler for allbnd' /up 0/
     ;

Equations
     dLdttt(j) 'gradient of Lagrangian w.r.t ttt(j)'
     dLdsss(i) 'gradient of Lagrangian w.r.t sss(i)'
     dLdx(j) 'gradient of Lagrangian w.r.t x(j)'
     ;

*initializing the marginals/lagrangian multipliers
sssdef_m.l(i)= sssdef.m(i);
tttdef_m.l(j) =tttdef.m(j);
esum_m.l = esum.m;
ssum_m.l = ssum.m;
allbnd_m.l = allbnd.m;

* dummy equations with differentiating variable values

dLdttt(j).. 37 =n= 0;
              ttt.fx(j)=ttt.l(j);

dLdsss(i).. sssdef_m(i) + 37 =n= 0;
              sss.fx(i) = sss.l(i);

dLdx(j).. tttdef_m(j)+ esum_m + ssum_m + allbnd_m =n= 0;
```

```
              x.fx(j) = x.l(j);

model nlpkkt / dLdttt.ttt , dLdsss.sss, dLdx.x, sssdef.sssdef_m , tttdef.tttdef_m, esum.
    esum_m, ssum.ssum_m, allbnd.allbnd_m / ;

nlpkkt.iterlim=0;

solve nlpkkt using MCP;

abort $(nlpkkt.objval >1e-5) 'We should start at a solution';
```

```
--- Job EX2KKT0.gms Start 07/26/18 15:46:49 25.1.1 r66732 WEX-WEI x86 64bit/MS Windows
GAMS 25.1.1 Copyright (C) 1987-2018 GAMS Development. All rights reserved
Licensee: Chintan Bhomia, Single User License G180612/0001CN-GEN
          GAMS Development, Fairfax DC14199
          cbhomia@gams.com
--- Starting continued compilation
--- Workfile was generated under GAMS version WEX251-251
--- EX2KKT0.gms(37) 3 Mb
--- Starting execution: elapsed 0:00:00.004
--- EX2KKT0.gms(91) 4 Mb
--- Generating MCP model nlpkkt
--- EX2KKT0.gms(93) 6 Mb
--- 21 rows 27 columns 79 non-zeroes
--- 17 nl-code 4 nl-non-zeroes
--- EX2KKT0.gms(93) 4 Mb
--- Executing PATH: elapsed 0:00:00.011
Reading dictionary...
Reading row data...
Evaluating functions...
Checking model...
Calculating Jacobian...

PATH 25.1.1 r66732 Released May 19, 2018 WEI x86 64bit/MS Windows

12 row/cols, 0 non-zeros, 0.00% dense.

Path 4.7.04 (Sat May 19 15:07:52 2018)
Written by Todd Munson, Steven Dirkse, and Michael Ferris

Major Iteration Log
major minor func grad residual step type prox inorm (label)
    0 0 1 1 1.3422e-10 I 0.0e+00 1.3e-10 (eSum)

FINAL STATISTICS
Inf-Norm of Complementarity . . 4.1996e-09 eqn: (eSum)
Inf-Norm of Normal Map. . . . . 1.3422e-10 eqn: (eSum)
Inf-Norm of Minimum Map . . . . 1.3422e-10 eqn: (eSum)
Inf-Norm of Fischer Function. . 1.3422e-10 eqn: (eSum)
Inf-Norm of Grad Fischer Fcn. . 0.0000e+00 eqn: (sssdef(i1))
Two-Norm of Grad Fischer Fcn. . 0.0000e+00

 ** EXIT - solution found.
```

```
Major Iterations. . . . 0
Minor Iterations. . . . 0
Restarts. . . . . . . . 0
Crash Iterations. . . . 0
Gradient Steps. . . . . 0
Function Evaluations. . 1
Gradient Evaluations. . 1
Basis Time. . . . . . . 0.000000
Total Time. . . . . . . 0.016000
Residual. . . . . . . . 1.342206e-10

--- Restarting execution
--- EX2KKT0.gms(93) 2 Mb
--- Reading solution for model nlpkkt
--- Executing after solve: elapsed 0:00:00.193
--- EX2KKT0.gms(95) 3 Mb
*** Status: Normal completion
--- Job EX2KKT0.gms Stop 07/26/18 15:46:49 elapsed 0:00:00.194
```

3.1 Replacing Dummy Equations

The dummy KKT equations representing gradient w.r.t the marginals are replaced one at a time per the process described in previous section. We first replace the equation $dLdttt(j)$ with the real KKT equation as shown in the model below (using restart feature). Also, the associated complementarity variable $ttt(i)$ is no longer fixed, which is the key to switching dummy equation from inactive to active. When executed with saved solution from NLP model, the model exists at iteration 0 with the message "solution found' , i.e. the equation defining KKT condition $dLttt(i)$ is correct.

It should be noted that the multiplier $sssdef\_m(i)$, defined in equation $dLdsss(i)$ is now removed as it is already part of the model in the new $dLdttt(j)$ equation. However, keeping $sssdef\_m(i)$ in the dummy equation would not affect the solution in any way because the variable $sss(j)$ for the equation $dLdsss(i)$ is still fixed, allowing the dummy equation to take any value in the LHS (attribute to =n= equality).

```
$title: refer EX2KKT1.gms, KKT conditions development
$ontext
This is an advanced model for showcasing development of KKT conditions for an NLP.
One dummy equations is replaced with KKT condition.
$offtext

Variables
     sssdef_m(i) 'multipler for equation sssdef(i)'
     tttdef_m(j) 'multipler for equations tttdef(j)'
     esum_m 'multipler for esum' /lo 0 /
     ssum_m 'multipler for ssum' /lo 0/
     allbnd_m 'multipler for allbnd' /up 0/
     ;

Equations
     dLdttt(j) 'gradient of Lagrangian w.r.t ttt(j)'
     dLdsss(i) 'gradient of Lagrangian w.r.t sss(i)'
     dLdx(j) 'gradient of Lagrangian w.r.t x(j)'
     ;

*initializing the marginals/lagrangian multipliers
```

```
sssdef_m.l(i)= sssdef.m(i);
tttdef_m.l(j) =tttdef.m(j);
esum_m.l = esum.m;
ssum_m.l = ssum.m;
allbnd_m.l = allbnd.m;

* dummy equations with differentiating variable values

* old dummy equation --> dLdttt(j).. 37 =n= 0; ttt.fx(j)=ttt.l(j);
dLdttt(j).. sum(i,sssdef_m(i)*A(i,j)) - tttdef_m(j) - allbnd_m =n= 0;

* dummy equations
dLdsss(i).. 37 =n= 0;
              sss.fx(i) = sss.l(i);

dLdx(j).. esum_m + ssum_m+ allbnd_m =n= 0;
              x.fx(j) = x.l(j);

model nlpkkt / dLdttt.ttt , dLdsss.sss, dLdx.x, sssdef.sssdef_m , tttdef.tttdef_m, esum.
    esum_m, ssum.ssum_m, allbnd.allbnd_m / ;

nlpkkt.iterlim=20;

solve nlpkkt using MCP;

abort $(nlpkkt.objval >1e-5) 'We should start at a solution'
```

```
--- Job EX2KKT1.gms Start 07/26/18 16:32:20 25.1.1 r66732 WEX-WEI x86 64bit/MS Windows
GAMS 25.1.1 Copyright (C) 1987-2018 GAMS Development. All rights reserved
Licensee: Chintan Bhomia, Single User License G180612/0001CN-GEN
          GAMS Development, Fairfax DC14199
          cbhomia@gams.com
--- Starting continued compilation
--- Workfile was generated under GAMS version WEX251-251
--- EX2KKT1.gms(36) 3 Mb
--- Starting execution: elapsed 0:00:00.009[LST:46]
--- EX2KKT1.gms(102) 4 Mb
--- Generating MCP model nlpkkt[LST:46]
--- EX2KKT1.gms(104) 6 Mb
--- 27 rows 27 columns 96 non-zeroes
--- 17 nl-code 4 nl-non-zeroes
--- EX2KKT1.gms(104) 4 Mb
--- Executing PATH: elapsed 0:00:00.016[LST:274]
Reading dictionary...
Reading row data...
Evaluating functions...
Checking model...
Calculating Jacobian...

PATH 25.1.1 r66732 Released May 19, 2018 WEI x86 64bit/MS Windows

18 row/cols, 46 non-zeros, 14.20% dense.

Path 4.7.04 (Sat May 19 15:07:52 2018)
```

```
Written by Todd Munson, Steven Dirkse, and Michael Ferris

Zero: 6 Single: 6 Double: 0

 ** EXIT - solution found.

Major Iterations. . . . 0
Minor Iterations. . . . 0
Restarts. . . . . 0
Crash Iterations. . . . 0
Gradient Steps. . . . . 0
Function Evaluations. . 0
Gradient Evaluations. . 0
Basis Time. . . . . 0.000000
Total Time. . . . . 0.000000
Residual. . . . . 0.000000e+00
Postsolved residual: 8.8861e-16

--- Restarting execution
--- EX2KKT1.gms(104) 2 Mb
--- Reading solution for model nlpkkt[LST:289]
--- Executing after solve: elapsed 0:00:00.082[LST:433]
--- EX2KKT1.gms(106) 3 Mb
*** Status: Normal completion[LST:449]
--- Job EX2KKT1.gms Stop 07/26/18 16:32:20 elapsed 0:00:00.083
```

The remaining dummy equations are removed per the process described above, to obtain the final MCP model shown below. The model is initialized by the results obtained from solving it as an NLP.

```
$title: refer EX2KKT3.gms, KKT conditions development
$ontext
This is an advanced model for showcasing development of KKT conditions for an NLP.
All dummy equations are replaced with KKT conditions
$offtext

Variables
     sssdef_m(i) 'multipler for equation sssdef(i)'
     tttdef_m(j) 'multipler for equations tttdef(j)'
     esum_m 'multipler for esum' /lo 0 /
     ssum_m 'multipler for ssum' /lo 0/
     allbnd_m 'multipler for allbnd' /up 0/
     ;

Equations
     dLdttt(j) 'gradient of Lagrangian w.r.t ttt(j)'
     dLdsss(i) 'gradient of Lagrangian w.r.t sss(i)'
     dLdx(j) 'gradient of Lagrangian w.r.t x(j)'
     ;

*initializing the marginals/lagrangian multipliers
sssdef_m.l(i)= sssdef.m(i);
tttdef_m.l(j) =tttdef.m(j);
esum_m.l = esum.m;
ssum_m.l = ssum.m;
allbnd_m.l = allbnd.m;
```

```
*dummy equations with differentiating variable values

dLdttt(j).. sum(i,sssdef_m(i)*A(i,j)) - tttdef_m(j) - allbnd_m =n= 0;

dLdsss(i).. 2*(sss(i)-s0(i)) - sssdef_m(i) - ssum_m$(i2(i)) - allbnd_m =n=0;
* ssum_m is active only for the equations where i2 is part of i

dLdx(j).. c(j) + 4*tttdef_m(j) + tttdef_m(j-1)- [esum_m*exp(x(j)-1)]$j2(j) - allbnd_m =n=
     0;
*exp over set j2. hence the differentiation term is over j2 as well

model nlpkkt / dLdttt.ttt , dLdsss.sss, dLdx.x, sssdef.sssdef_m , tttdef.tttdef_m, esum.
    esum_m, ssum.ssum_m, allbnd.allbnd_m / ;

nlpkkt.iterlim=0;

solve nlpkkt using MCP;

abort $(nlpkkt.objval >1e-5) 'We should start at a solution';
```

```
--- Job EX2KKT3.gms Start 07/26/18 16:41:02 25.1.1 r66732 WEX-WEI x86 64bit/MS Windows
GAMS 25.1.1 Copyright (C) 1987-2018 GAMS Development. All rights reserved
Licensee: Chintan Bhomia, Single User License G180612/0001CN-GEN
          GAMS Development, Fairfax DC14199
          cbhomia@gams.com
--- Starting continued compilation
--- Workfile was generated under GAMS version WEX251-251
--- EX2KKT3.gms(39) 3 Mb
--- Starting execution: elapsed 0:00:00.008[LST:49]
--- EX2KKT3.gms(105) 4 Mb
--- Generating MCP model nlpkkt[LST:49]
--- EX2KKT3.gms(107) 6 Mb
--- 27 rows 27 columns 111 non-zeroes
--- 45 nl-code 12 nl-non-zeroes
--- EX2KKT3.gms(107) 4 Mb
--- Executing PATH: elapsed 0:00:00.015[LST:284]
Reading dictionary...
Reading row data...
Evaluating functions...
Checking model...
Calculating Jacobian...

PATH 25.1.1 r66732 Released May 19, 2018 WEI x86 64bit/MS Windows

27 row/cols, 111 non-zeros, 15.23% dense.

Path 4.7.04 (Sat May 19 15:07:52 2018)
Written by Todd Munson, Steven Dirkse, and Michael Ferris

Major Iteration Log
major minor func grad residual step type prox inorm (label)
    0 0 1 1 5.2222e-09 I 0.0e+00 5.2e-09 (dLdx(j3))
```

```
FINAL STATISTICS
Inf-Norm of Complementarity . . 5.2205e-09 eqn: (dLdx(j3))
Inf-Norm of Normal Map. . . . . 5.2205e-09 eqn: (dLdx(j3))
Inf-Norm of Minimum Map . . . . 5.2205e-09 eqn: (dLdx(j3))
Inf-Norm of Fischer Function. . 5.2205e-09 eqn: (dLdx(j3))
Inf-Norm of Grad Fischer Fcn. . 2.0882e-08 eqn: (tttdef(j3))
Two-Norm of Grad Fischer Fcn. . 2.2149e-08

 ** EXIT - solution found.

Major Iterations. . . . 0
Minor Iterations. . . . 0
Restarts. . . . . 0
Crash Iterations. . . . 0
Gradient Steps. . . . . 0
Function Evaluations. . 1
Gradient Evaluations. . 1
Basis Time. . . . . 0.000000
Total Time. . . . . 0.000000
Residual. . . . . 5.222184e-09

--- Restarting execution
--- EX2KKT3.gms(107) 2 Mb
--- Reading solution for model nlpkkt[LST:299]
--- Executing after solve: elapsed 0:00:00.081[LST:451]
--- EX2KKT3.gms(109) 3 Mb
*** Status: Normal completion[LST:467]
--- Job EX2KKT3.gms Stop 07/26/18 16:41:02 elapsed 0:00:00.082
```