Einführung in die mathematische Modellierung mit der Software GAMS

Knut Haase, Matthes Koch, Sven Müller

Institut für Verkehrswirtschaft

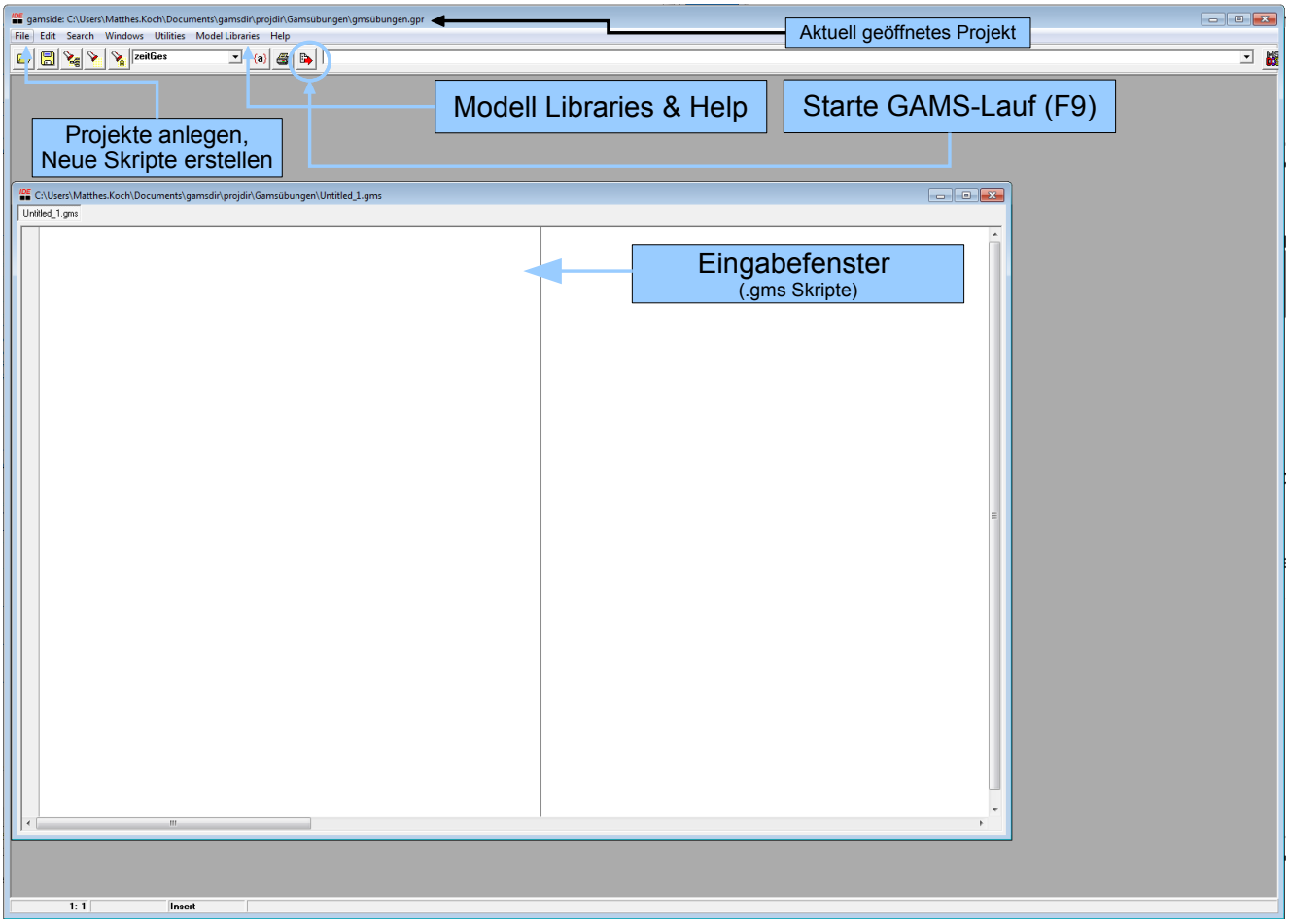
23. April 2016

Übersicht

Einführung

- GAMS: **G**eneral **A**lgebraic **M**odeling **S**ystem
- algebraische Modellierungssprache
- Formulieren und Lösen mathematischer Modelle
- ursprgl. entwickelt an der World Bank in den 1970er Jahren
- aktuell: Version 24.7 (April 2016)
- Demoversion: `http://www.gams.com/download`
  Modellgrößen beschränkt (vgl. Downloadseite)
- weitere bekannte Modellierungssprachen:
  - AMPL (A Modeling Language for Mathematical Programming)
  - OML (Optimization Modeling Language)
  - LINGO
  - uvm...

Einführung - GAMSIDE



Einführung - Dateitypen

Zu einem GAMS-Projekt gehören folgende Dateien:

- **.gpr Projektdatei**
- **.gms Gams-Skript - hierin schreiben wir mathematische Modelle**
- **.lst Gams-Listing - Ergebnisdatei eines GAMS-Aufrufs**
- .lxi Inhaltsverzeichnis des Listing-Files (GAMSIDE)
- .log Gams-Log - Logdatei
- .gdx *g*ams *d*ata e*x*change - externe Datensätze
- .gch Diagrammdateien
- .inc Includedateien - bei Modularisierung des Skriptes
- .opt Optionsdateien für Solver
- .txt Textdateien, z.B. als externe Daten, geschriebene Ausgabedateien u.a.

Diese Dateien werden in einem gemeinsamen Projektordner mit der .gpr-Datei abgelegt.

Einführung - Projekte und Dateitypen

Projekte - .gpr

- Projekt: Sammlung zusammengehöriger Dateien (Inputdaten, Modelle, Outputdateien)
- Pfad bzw. Verzeichnis der Projektdatei ist default für alle von GAMS im Rahmen der Projektdatei erzeugten Dateien
- Endung von Projektdateien: .gpr
- Menü `File` → `Project` → `New` oder `Open`
- aktuell geöffnetes Projekt wird in der Titelleiste der GAMSIDE (blauer Bereich ganz oben) angezeigt

Einführung - Projekte und Dateitypen

Skripte - .gms

- Enhält das Modell und alle zugehörigen Deklarationen
- Endung von Projektdateien: .gms
- `File` → `New`
- `File` → `Save as` (dort, wo aktuelle .gpr liegt)
- Daten können in der .gms Datei enthalten sein oder aus anderen Quellen integriert werden.

Einführung - GAMSIDE

Nützliche Hilfsmittel:

- unter `Help` befinden sich viele hilfreiche Anleitungen, insb.
  - GAMS Users Guide
  - Expanded GAMS Guide (McCarl)
  - Anleitungen zu den in GAMS verwendeten Solvern: docs/solvers
- Model Libraries: große Auswahl bereits programmierter Modelle

Orientierung

Beispiel

$$
\begin{array}{lllll}
max & F & = & 90x_1 + 50x_2 + 70x_3 + 40x_4 & (1) \\
 & & & \text{s.t.} & \\
 & & & 2x_1 - 1x_2 + 1x_3 + 2x_4 \leq 30 & (2) \\
 & & & 1x_1 + 2x_2 + 3x_3 - 1x_4 \leq 24 & (3) \\
 & & & 3x_1 + 2x_2 - 1x_3 + 2x_4 \leq 36 & (4) \\
 & & & x_1, x_2, x_3, x_4 \geq 0 & (5)
\end{array}
$$

Modell abstrahieren:

- Ziel: Trennung von konkreten Daten und Modellstruktur
- Daten durch Symbole repräsentieren

Beispiel

$$
\begin{aligned}
max \quad F \quad &= \quad c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 && && (6)\\
&\quad \text{s.t.} \\
&\quad a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 && \leq b_1 && (7)\\
&\quad a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 && \leq b_2 && (8)\\
&\quad a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 && \leq b_3 && (9)\\
&\quad x_1, x_2, x_3, x_4 && \geq 0 && (10)
\end{aligned}
$$

(7) - (9) besitzen dieselbe Struktur → Zusammenfassung zu einem Restriktionsblock (12)

$$
\begin{aligned}
max \quad F \quad &= \quad \sum_j c_j x_j && && (11)\\
&\quad \text{s.t.} \\
&\quad \sum_j a_{ij} x_j && \leq b_i \quad \forall i \in I && (12)\\
&\quad x_j && \geq 0 \quad \forall j \in J && (13)
\end{aligned}
$$

Das so verallgemeinerte **Modell** besitzt:

Mengen

- J Spalten (Auswahlalternativen) $j = 1, 2, 3, 4$
- I Zeilen $i = 1, 2, 3$

Parameter

- c_j Zielfunktionskoeffizient der Alternative j
- $a_{i,j}$ Kapazitätsparameter bzgl. Alternative j in Restriktion i
- b_i Kapazitätsgrenze von Restriktion (Ressource) i

Variablen

- F Zielfunktionsvariable
- x_j gewählte Menge der Alternative j

Gleichungsblöcke

Übertragung formaler Modelle in GAMS Syntax:

Mathematisches Modell	**GAMS Modell**	
Mengen	**sets**	(Daten)
Parameter	**parameters**	(Daten)
	tables	(Daten)
	scalars	(Daten)
Variablen	**variables**	(Modellstruktur)
Zielfunktion	**equations**	(Modellstruktur)
Restriktionen	**equations**	(Modellstruktur)

→ Einfache Änderung der Inputdaten ohne Änderung der Modellstruktur

Syntax

Erzeugen von entities (z.B. Parameter):

```
TypDesEntities NameDesEntities Kommentar /Wertzuweisung/;
```

- Deklaration (`TypDesEntities NameDesEntities`) und Definition (`/Wertzuweisung/`) können wie hier zusammen oder aber seperat erfolgen (Ausnahme: equations/variables immer seperat)
- Mehrere entities gleichen Typs können seperat mit je einer Anweisung oder zusammen in einer Anweisung deklariert werden (Trennung durch Komma oder Zeilenumbruch)
- Bezeichner (`NameDesEntities`) müssen mit einem Buchstaben beginnen, dem bis zu 30 weitere Zeichen folgen können
- keine Unterscheidung zwischen Groß- und Kleinschreibung
- WICHTIG: Bezeichner (engl. Identifyer) müssen eindeutig sein.

Weitere Hinweise

- Leerzeichen und Leerzeilen zur besseren Lesbarkeit beliebig einsetzbar
- mehrzeilige Anweisungen bzw. mehrere Anweisungen auf einer Zeile sind erlaubt
- alle Anweisungen vorerst immer durch Semikolon abschließen, um Fehler zu vermeiden
- Kommas in Dezimalzahlen werden durch Punkte dargestellt
- Kommentare zur Dokumentation:
  - direkt bei der Deklaration der sets, parameter, equations usw.
  - gesamte Zeile: * am Zeilenanfang
- Bestimmte Schlüsselworte und Symbole sind reserviert und dürfen nicht anderweitig verwendet werden.

Mengen

- J Spalten (Auswahlalternativen) $j = 1, 2, 3, 4$
- I Zeilen $i = 1, 2, 3$

```
* Mit dem *-Symbol am Zeilenbeginn können Kommentare eingefügt werden (grau)
sets
 I Menge der Restriktionen /1,2,3/
 J Menge der Spalten /1*4/
;
```

Hinweis:

Deklarationen und Definitionen sind nicht case-sensitive. Als Schlüsselwort wird set und sets erkannt.

```
SeT
 i Menge der Restriktionen /1,2,3/
 J Menge der Spalten /1*4/
;
```

Mengenelemente können auch Strings (Zeichenketten) sein - bspw. Städtenamen.

Parameter (1)

- c_j Zielfunktionskoeffizient der Alternative j
- b_i Kapazitätsgrenze von Restriktion (Ressource) i

```
parameter
 c(j) Zielfunktionskoeffizient / 1 90, 2 50, 3 70, 4 40 /
 b(i) Kapazitätsgrenze / 1 30
                         2 24
                         3 36 /
;
```

Trennung der einzelnen Einträge durch Kommasetzung oder Leerzeilen möglich.

Parameter (2)

- $a_{i,j}$ Kapazitätsparameter bzgl. Alternative j in Restriktion i

```
table a(i,j) Kapazitätskoeffizienten
          1       2       3       4
1         2       -1      1       2
2         1       2       3       -1
3         3       2       -1      2
;
```

Ebenfalls möglich:

```
parameter a(i,j) Koeffizientenmatrix
         /1.1 2, 1.2 -1, 1.3  1, 1.4  2
          2.1 1, 2.2  2, 2.3  3, 2.4 -1
          3.1 3, 3.2  2, 3.3 -1, 3.4  2 / ;
```

Variablen

- F Zielfunktionsvariable, $-\infty \leq F \leq \infty$
- x_j gewählte Menge der Alternative j, $x_j \geq 0 \quad \forall i \in I$

```
free variables
 F Zielfunktionswert
;

positive variables
 x(j) gewählte Menge von Alternative j
;
```

Variablentyp	Lower Bound (.lo)	Upper Bound (.up)	Anmerkung
free	$-\infty$	$+\infty$	Beide Schranken veränderbar
positive	0	$+\infty$	obere Schranke veränderbar
negative	$-\infty$	0	untere Schranke veränderbar
binary	0	1	diskret, Schranken sind fix
integer	0	100	diskret, beide Schranken veränderbar

Gleichungen

Deklaration einer Gleichung mit einem Bezeichner und optional einem Kommentar erfolgt separat **vor** der Definition; Die Deklaration muss mit einem Semikolon abgeschlossen werden.

$$max\ F \ = \sum_j c_j x_j \tag{14}$$

s.t.

$$\sum_j a_{ij} x_j \quad \leq \quad b_i \qquad \forall i \in I \tag{15}$$

```
equations
 Zielfkt Zielfunktion
 Kapazit Kapazitätsrestriktion
;

Zielfkt.. F =E= sum(j, c(j) * x(j) );
Kapazit(i).. sum(j, a(i,j) * x(j) ) =L= b(i);
```

Equations

Hinweise zu Equations

- Deklaration separat vor Definition
- Definition einer Gleichung durch `..`
- Die Domain eines Gleichungsblocks wird wie bei anderen entities durch eine Klammer **()** nach dem Bezeichner angegeben. Im Beispiel wird Gleichung `Kapazit` für alle Elemente in `i` aufgestellt ($\forall i \in I$). Der Block `Kapazit(i)` enthält also Gleichungen:
  - `Kapazit(1)`
  - `Kapazit(2)`
  - `Kapazit(3)`

Summen, Produkte und Operatoren

Summen werden durch das Schlüsselwort `sum()` definiert. Innerhalb der Klammer steht zuerst der Index, über den summiert wird. Darauf folgt der Summenterm. Produkte werden analog gebildet.

```
*Beispiel zur Summenbildung: sum()
 sum(j, x(j) - y(j) );
* Eine Summe über 2 Mengen -
* Indices müssen eingeklammert werden:
 sum((k,l), z(k,l) * p(k) );
*Beispiel zur Produktbildung:
 prod(j, p(j) * r(j) );
```

Operatoren

GAMS-Syntax	Operator	Bedeutung
=L=	$\leq$	kleiner-gleich
=G=	$\geq$	größer-gleich
=E=	$=$	ist-gleich

Modell

- Equations können zu einem Model zusammengefasst werden.
- Wie bei anderen entities kann ein Kommentar vor der Definition eingefügt werden.
- Die Wertzuweisung erfolgt durch auflisten der equations, oder durch /all/.

```
model Beispiel1 /Zielfkt,Kapazit/;
*ebenso möglich:
*model Beispiel1 /all/;
```

Es können mehrere Modelle in einem .gms - Skript definiert werden.

Umsetzung in GAMS - Beispiel1.gms

```
* Mit dem *-Symbol am Zeilenbeginn können Kommentare eingefügt werden (grau)
sets
 I Menge der Restriktionen /1,2,3/
 J Menge der Spalten /1*4/
;

parameter
 c(j) Zielfunktionskoeffizient / 1 90, 2 50, 3 70, 4 40 /
 b(i) Kapazitätsgrenze / 1 30
                        2 24
                        3 36 /
;

table a(i,j) Kapazitätskoeffizienten
         1       2       3       4
1        2       -1      1       2
2        1       2       3       -1
3        3       2       -1      2
;

free variables
 F Zielfunktionswert
;

positive variables
 x(j) gewählte Menge von Alternative j
;

equations
 Zielfkt Zielfunktion
 Kapazit Kapazitätsrestriktion
;

Zielfkt.. F =E= sum(j, c(j) * x(j) );
Kapazit(i).. sum(j, a(i,j) * x(j) ) =L= b(i);

model Beispiel1 /Zielfkt,Kapazit/;

solve Beispiel1 maximizing F using lp;
```

Orientierung

Lösen des Optimierungsproblems

- Übergabe des so definierten Optimierungsmodells an einen Solver
- Im Solver sind Lösungsverfahren implementiert (z.B. Simplex, Schnittebenenverfahren u.a.)

```
solve Beispiel1 maximizing F using lp;
```

Schlüsselwort zum Aufruf des Solvers: solve

dahinter Bestimmung:

- des zu berechnenden Modells
- der Optimierungsrichtung, möglich:
  - maximizing
  - minimizing
- der zu optimierenden Zielfunktionsvariable
- des Modelltyps

Modelltypen und Solver

Kürzel	Modelltyp
lp	**linear programming**
nlp	nonlinear programming
mip	**mixed integer programming**
rmip	**relaxed mixed integer programming**
minlp	mixed integer nonlinear programming
rminlp	relaxed mixed integer nonlinear programming
mcp	mixed complementarity problems
cns	constrained nonlinear systems

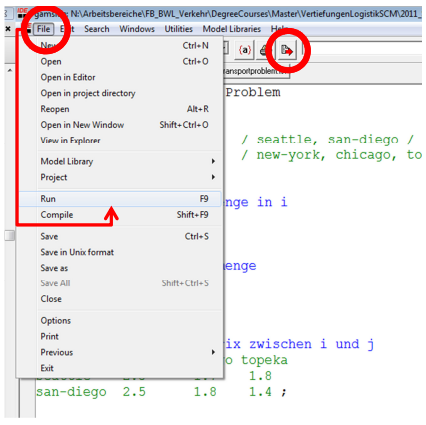
- Zunächst Betrachtung von Modellen der linearen Programmierung (lp) und der ganzzahligen Programmierung (mip)
- Der bei uns dafür verwendete Standardsolver ist `IBM Cplex`
- Beim Lösen von **mip**-Modellen sind weitere Einstellungen möglich (vgl. Folie 51)
- Mehr Informationen über Solver finden sich unter `Help` → `Solver Manual`.

Compile, Run

- Kompilieren: Shift + F9

  Menü: `File` → `Compile`
  - Das Skript wird kompiliert und auf Fehler überprüft.
- Gams starten: F9

  Menü: File → Run
  - Das Skript wird kompiliert und auf Fehler überprüft. Das Modell wird für den Solver vorbereitet. Danach wird der Solver aufgerufen.



Orientierung

Übersicht

- Während des Gamslaufes wird im *Process-Window* ein Protokoll des Gamslaufes angezeigt (`.log` Datei)
- Nach Beendigung des Laufes wird standardmäßig das **Listing** angezeigt (`.lst` - Datei im Verzeichnis der Projektdatei)
- Sämtliche unmittelbar relevanten Größen lassen sich daraus ablesen. Das Listing besteht aus
  - einem echo-print, der die Model-Syntax nochmal wiedergibt
  - dem Equation Listing
  - dem Column Listing
  - der Model Statistics
  - dem Solution Report
  - optional dem Display

Equation Listing

```
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Equation Listing    SOLVE Beispiel1 Using LP From line 41


---- Zielfkt  =E=  Zielfunktion

Zielfkt..  F - 90*x(1) - 50*x(2) - 70*x(3) - 40*x(4) =E= 0 ; (LHS = 0)


---- Kapazit  =L=  Kapazitätsrestriktion

Kapazit(1)..  2*x(1) - x(2) + x(3) + 2*x(4) =L= 30 ; (LHS = 0)

Kapazit(2)..  x(1) + 2*x(2) + 3*x(3) - x(4) =L= 24 ; (LHS = 0)

Kapazit(3)..  3*x(1) + 2*x(2) - x(3) + 2*x(4) =L= 36 ; (LHS = 0)
```

- Darstellung der ersten drei ausformulierten Gleichungen (default = 3) jedes Blocks
- linke Seite der Gleichungen (LHS): Variablen mit Parametern; rechte Seite: Konstanten
- Initialisierung: alle Variablen x_j auf kleinst möglichem Wert (0)
- Falls eine Gleichung durch die Initialisierung unzulässig wird, wird zusätzlich (INFES = infeasible) angezeigt

Column Listing

Darstellung der ersten drei konkreten Variablen (default = 3) je deklarierter Variable

(a) Angabe der Wertebereiche und der Startwerte der Variablen
- .LO = Lower Bound
- .L = Level
- .UP = Upper Bound
- .M = Marginal

(b) Angabe der Koeffizienten der Variablen in allen Equations (Spalte in der Koeffizientenmatrix)

```
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Column Listing      SOLVE Beispiel1 Using LP From line 41


---- F  Zielfunktionswert

F
                (.LO, .L, .UP, .M = -INF, 0, +INF, 0)
        1       Zielfkt

                                                                  (a)

---- x  gewählte Menge von Alternative j

x(1)
                (.LO, .L, .UP, .M = 0, 0, +INF, 0)
      -90       Zielfkt
        2       Kapazit(1)
        1       Kapazit(2)                                        (b)
        3       Kapazit(3)

x(2)
                (.LO, .L, .UP, .M = 0, 0, +INF, 0)
      -50       Zielfkt
       -1       Kapazit(1)
        2       Kapazit(2)
        2       Kapazit(3)

x(3)
                (.LO, .L, .UP, .M = 0, 0, +INF, 0)
      -70       Zielfkt
        1       Kapazit(1)
        3       Kapazit(2)
       -1       Kapazit(3)

REMAINING ENTRY SKIPPED
```

Model Statistics

```
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m
Model Statistics    SOLVE Beispiel1 Using LP From line 41


MODEL STATISTICS

BLOCKS OF EQUATIONS           2     SINGLE EQUATIONS            4
BLOCKS OF VARIABLES           2     SINGLE VARIABLES            5
NON ZERO ELEMENTS            17


GENERATION TIME      =        0.156 SECONDS      4 Mb  WEX235-235 Aug 17, 2010


EXECUTION TIME       =        0.156 SECONDS      4 Mb  WEX235-235 Aug 17, 2010
```

- `Blocks of Equations`: Anzahl Gleichungsblöcke (hier: 1 Zielfunktion (11) + 1 Kapazitätsbeschränkung (12))
- `Single Equations`: Anzahl an Gleichungen (hier: 1 Zielfunktion (11) + 3 ausformulierte Restriktionsgleichungen (2) - (4))
- `Blocks of Variables` : Anzahl der deklarierten Variablen (hier: F und x_j)
- `Single Variables`: Anzahl ausformulierter Variablen (hier: F und $x_1, ..., x_4$)
- `Non Zeor Elements`: Anzahl von 0 verschiedener Elemente der Koeffizientenmatrix (hier: 3 x 4 aus den Nebenbedingungen + 1 x 5 aus der ZF)
- `Generation time`: Zeit, die GAMS benötigt hat, um das Modell für den Solver aufzubereiten

Solution Report

Nachdem im Listing zuerst das Modell vor dem Solveraufruf dargestellt wurde, folgt im `Solution Report` des Listings die Auswertung der Solver Ergebnisse.

Es werden also Variablen und Gleichungen mit den vom Solver ermittelten Variablenwerten ausgewertet.

Zum Solution Report gehören:

a) Solve Summary

b) SolEqu: Solution Report Equations

c) SolVar: Solution Report Variables

a) Solution Report - Solve Summary

```
       G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
       Solution Report     SOLVE Beispiel1 Using LP From line 41

                    S O L V E      S U M M A R Y

            MODEL   Beispiel1           OBJECTIVE  F
            TYPE    LP                  DIRECTION  MAXIMIZE
            SOLVER  CPLEX               FROM LINE  41

       **** SOLVER STATUS     1 Normal Completion
       **** MODEL STATUS      1 Optimal
       **** OBJECTIVE VALUE             1513.3333

        RESOURCE USAGE, LIMIT          0.129      1000.000
        ITERATION COUNT, LIMIT         4    2000000000

       IBM ILOG CPLEX   Aug 18, 2010 23.5.2 WEX 19143.19383 WEI x86_64/MS Windows
       Cplex 12.2.0.0, GAMS Link 34
```

- Model, Direction, Objective, Type entsprechen dem Solve-Statement des .gms Skriptes vgl. Folie 27
- Solver Status, Model Status: Rückmeldung des Solvers - verschiedene Statusmeldungen möglich - vgl. GAMS Guide - Help → expanded gams guide. Hier wurde das Modell optimal gelöst.
- Objective Value: Zielfunktionswert
- resource usage: Zeitverbrauch des Solvers beim Lösen des Modells (CPU s)
- iteration count: Benötigte Iterationen des Solvers zum Lösen des Modells

b) Solution Report Equations

```
                     LOWER     LEVEL     UPPER    MARGINAL

---- EQU Zielfkt         .         .         .        1.000

  Zielfkt  Zielfunktion

---- EQU Kapazit  Kapazitätsrestriktion

     LOWER     LEVEL     UPPER    MARGINAL

1     -INF     30.000    30.000    18.148
2     -INF     24.000    24.000    21.481
3     -INF     36.000    36.000    12.593
```

- Angabe der unteren und oberen Schranken der Gleichungen - im Bsp. entsprechen Upper Bound den Konstanten b_i
- Angabe des Levels: Wert der linken Seite der Gleichung
- Angabe der Marginals: **Simplexmultiplikatoren** der Restriktion
  - Wie verändert sich der Zielfunktionswert bei einer marginalen Veränderung der Konstanten → Veränderung um $\Delta \cdot \pi_i$
  - Hier würde der ZF steigen, da Kapazitätsgrenze der Restriktion erhöht würde → ein besserer Zielfunktionswert wäre möglich

c) Solution Report Variables

```
                     LOWER     LEVEL     UPPER    MARGINAL

---- VAR F              -INF   1513.333     +INF        .

  F  Zielfunktionswert

---- VAR x  gewählte Menge von Alternative j

     LOWER     LEVEL     UPPER    MARGINAL

1      .         .        +INF     -5.556
2      .       7.333      +INF        .
3      .       8.000      +INF        .
4      .      14.667      +INF        .
```

- Untere (LOWER) und obere (UPPER) Beschränkung der Variable
- LEVEL: Wert der Variablen in der Lösung (optimaler Wert)
- MARGINAL: Reduzierte Kosten
  - wie verändert sich der ZFW, wenn die Variable um eine marginale Einheit Δ erhöht wird
  - Veränderung um $\Delta \cdot \bar{c}_j$

Display Anweisung

Ausgabe gewünschter Werte am Ende des Listings

```
solve Beispiel1 maximizing F using lp;

display
 F.l, x.l, x.m, a;
```

```
----     44 VARIABLE F.L                  =     1513.333  Zielfunktionswert

----     44 VARIABLE x.L  gewählte Menge von Alternative j

2  7.333,    3  8.000,    4 14.667


----     44 VARIABLE x.M  gewählte Menge von Alternative j

1 -5.556


----     44 PARAMETER a  Kapazitätskoeffizienten

           1          2          3          4

1      2.000     -1.000      1.000      2.000
2      1.000      2.000      3.000     -1.000
3      3.000      2.000     -1.000      2.000
```

Fehlerhinweise - Logfile

Logfile (.log-Datei):

- Fehlermeldungen werden mit Zeilenangabe in der Reihenfolge ihres Auftretens im logfile-Fenster angezeigt.
- Doppelklick auf die rot markierte Fehlermeldung führt zur fehlerhaften Zeile in der .gms-Datei.
- Ein Fehler führt oft zu mehreren Fehlermeldungen →**die erste Fehlermeldung ist die wichtigste**

Echo Print (im Listing):

- Fehlernummer erscheint im Echo Print direkt in der Zeile unterhalb der (scheinbar) fehlerhaften Zeile an entsprechender Stelle und die Zeile wird durch **** markiert.
- Am Ende des Echo Prints werden alle aufgetretenen Fehlernummern mit Beschreibung des Fehlers aufgeführt.

Orientierung

Sets - Elemente definieren

- Mengenelemente können durch Strings definiert werden
- Werden Elementbezeichnungen mit Zahlen verwendet, muss lediglich das erste und das letzte Element verbunden durch * angegeben werden

```
sets
 Bezirke Menge der Bezirke /Pankow, Mitte, Charlottenburg/;
sets
 Staedte Menge der Staedte /Stadt5*Stadt12/;

display Staedte, Bezirke;
```

```
----      9 SET Staedte  Menge der Staedte

Stadt5 ,    Stadt6 ,    Stadt7 ,    Stadt8 ,    Stadt9 ,    Stadt10,    Stadt11
Stadt12


----      9 SET Bezirke  Menge der Bezirke

Pankow        ,    Mitte         ,    Charlottenburg
```


Sets - Mehrere Namen für ein Set

```
set      i      Knotenmenge    / 1, 2, 3, 4, 5, 6 /  ;

alias( i, k, l) ;
```

$\longrightarrow \quad i = k = l = \{1, 2, 3, 4, 5, 6\}$

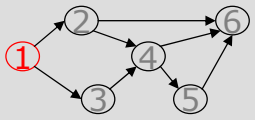
Trennung der Namen durch Komma

- Einem set dürfen beliebig viele weitere Namen gegeben werden.
- Die Reihenfolge der Bezeichner in der alias-Anweisung ist egal.
- **Beachte**: Nur ein Bezeichner darf vorher schon deklariert worden sein.

Sets - Subsets definieren

```
sets    i     Knotenmenge          /1*6/
        q(i)  Startknoten          /1/ ;
```

Bezeichner der übergeordneten Menge

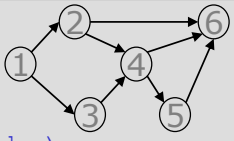


q ist eine Teilmenge (subset) von i - es enthält hier ein Element der übergeordneten Menge. Es empfiehlt sich die übergeordnete Menge in Klammern anzugeben, um Fehler zu vermeiden, die durch eine fehlerhafte Angabe der Elemente der Teilmenge entstehen.

Bei der Compilierung wird überprüft, ob Elemente der Teilmenge wirklich Elemente der übergeordneten Menge sind. Würde q hier bspw. das Element „7“ zugewiesen, käme es zu einer Fehlermeldung.

Sets - Zuweisung von Elementen nach der Deklaration

```
sets        i     Knotenmenge    /1*6/
            q(i)  Quelle(Startknoten)
            s(i)  Senke
            r(i)  Restknoten (weder Quelle noch Senke) ;
```




```
q("1")= yes ;
s("6")= yes ;
```

Zugehörigkeit eines Elements zu einer Menge:
ja (yes) (=Hinzufügen) oder nein (no) (=Entfernen)

Bezug auf einzelne Elemente: Name des Elements in " " oder ' '

Bei Subsets Bezug auf alle Elemente des übergeordneten Sets:
Bezeichner des übergeordneten Sets

```
r(i)= yes ;
```
$\longrightarrow r = \{1, 2, 3, 4, 5, 6\}$

```
r("1")= no ;
```
$\longrightarrow$ Entfernen der Quelle $\longrightarrow r = \{2, 3, 4, 5, 6\}$

```
r("6")= no ;
```
$\longrightarrow$ Entfernen der Senke $\longrightarrow r = \{2, 3, 4, 5\}$

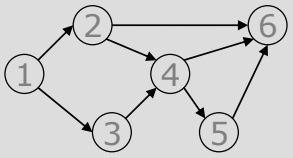
Sets - Mehrdimensionale Mengen

```
sets        i       Knotenmenge   / 1, 2, 3, 4, 5, 6 /

            p(i,i)  Pfeilmenge    / 1.2
                                    1.3
                                    2.4
                                    2.6
                                    3.4
                                    4.5
                                    4.6
                                    5.6 / ;
```

Index 1 Index 2

Verbindung der Elemente durch .



Ein Pfeil verbindet zwei Knoten (=zwei Elemente aus der Menge Knoten) miteinander.

Zusammenfassende Schreibweise, z.B.

```
set         p(i,i)  Pfeilmenge
                                / 1.(2,3)
                                  2.(4,6)
                                  3.4
                                  4.(5,6)
                                  5.6 / ;
```

Bis zu 10 Dimensionen möglich

Parameter - mehrdimensionale Daten in Tabellenform

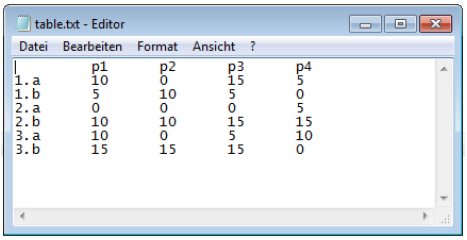
```
set
 i /1*3/
 j /a,b/
 k /p1*p4/
;
table tab(i,j,k) Eine Tabelle mit 3 Dimensionen
          p1      p2      p3      p4
1.a       10      0       15      5
1.b       5       10      5       0
2.a       0       0       0       5
2.b       10      10      15      15
3.a       10      0       5       10
3.b       15      15      15      0
;
```

- Wertetabellen werden **ohne** Backslashes (/.../) definiert
- Tabellenformat muss eingehalten werden - Dateneinträge dürfen Spaltenstruktur nicht überlappen
- mehrere Dimensionen können durch . verknüpft werden. Im Beispiel ist $tab_{\mathbf{1},a,\mathbf{p1}} = 10$.

Parameter - mehrdimensionale Daten in Tabellenform

```
set
 i /1*3/
 j /a,b/
 k /p1*p4/
;
table tab(i,j,k) Eine Tabelle mit 3 Dimensionen
$include table.txt
;
```

- Es kann nur eine table pro table-Anweisung deklariert werden
- Tables können bis zu 10 Dimensionen haben
- Große Datenmengen können in .txt Dateien ausgelagert werden
- Tabellenspalten durch Tabstop (Tabulatortaste) trennen



Weitere Möglichkeiten zur Einbindung externer Daten →Help-Menü

Parameter - Wertzuweisung (dynamische Parameter)

```
table      d(s,k) Distanz zwischen Standort s und Kunde k
         DD          F          H          K          S
B        200        540        290        570        630
HH       500        490        150        420        670
M        460        390        630        570        220   ;

scalar     f      Frachtkosten pro Stück und 100 km  /0.5/ ;

parameter  c(s,k) Transportkosten pro Stück zw. s und k ;

c(s,k) = d(s,k) / 100 * f  ;
```

Berechnung von c für jede Kombination aus s und k

auf konkrete Elemente wird über 'Element' Bezug genommen

Variablen - Verändern des Wertebereiches

Suffix	Bedeutung
.lo	untere Schranke (lower bound)
.up	obere Schranke (upper bound)
.fx	Fixierung: untere und obere Schranke sind identisch

Variable .Suffix Index bzw. Indizes, hier: gilt nur für bestimmte Variablen

```
x.up('B','DD') = 0;
```
Kunde in DD darf nicht von Standort B beliefert werden

```
x.lo('M','K') = 100;
```
Kunde in K muss mindestens 100 ME aus M erhalten

```
x.fx('HH','H') = 50;
```
Kunde in H erhält genau 50 ME aus HH

Lösen gemischt ganzzahliger Modelle

- Modelle, die ganzzahlige Variablen beinhalten werden als MIP - mixed integer program - bezeichnet
- derartige Modelle erfordern i.d.R. einen wesentlich höheren Rechenaufwand als lineare Porgramme - u.U. müssen sehr viele LPs gelöst werden

Solve-Aufruf eines MIP:

```
solve modelname maximizing F using mip ;
```

Die LP-Relaxation (relaxed mixed integer program) eines ganzzahligen Programs wird als rmip bezeichnet. Dabei wird die Ganzzahligkeitseigenschaft der Variablen verworfen.

```
solve modelname maximizing F using rmip ;
```

Summen, Produkte und Operatoren

Summen werden durch das Schlüsselwort sum() definiert. Innerhalb der Klammer steht zuerst der Index, über den summiert wird. Darauf folgt der Summenterm. Produkte werden analog gebildet.

```
*Beispiel zur Summenbildung: sum()
 sum(j, x(j) - y(j) );
* Eine Summe über 2 Mengen -
* Indices müssen eingeklammert werden:
 sum((k,l), z(k,l) * p(k) );
*Beispiel zur Produktbildung:
 prod(j, p(j) * r(j) );
```

Operatoren

GAMS-Syntax	Operator	Bedeutung
=L=	$\leq$	kleiner-gleich
=G=	$\geq$	größer-gleich
=E=	$=$	ist-gleich

Funktionen - card() und ord()

Die ord()-Funktion ermittelt die Position eines Elements in der Menge.

```
sets
 bz /Pankow,Marzahn,Kreuzberg/ ;
parameter
 pos(bz) Position der Elemente in der Menge ;

*Wird für jedes Element in bz ausgeführt:
pos(bz) = ord(bz) ;
*Anzeige:
display pos ;
```

```
----      9 PARAMETER pos  Position der Elemente in der Menge

Pankow    1.000,    Marzahn   2.000,    Kreuzberg 3.000
```

- Zugriff auf einzelne Elemente einer Menge z.B. in Summen oder Schleifen
- ord() kann nicht auf „dynamische“ sets angewendet werden
  - dynamische Elementzuweisung durch yes/no (vgl. Folie 45)

Funktionen - card() und ord()

Die card()-Funktion ermittelt die Anzahl der Elemente einer Menge.

```
sets
 bz /Pankow,Marzahn,Kreuzberg/ ;
parameter
 anz Anzahl der Elemente in der Menge ;

anz = card(bz) ;
*Anzeige:
display anz ;
```

```
----      8 PARAMETER anz                 =        3.000  Anzahl der Elemente i
                                                          n der Menge
```

- Formal wird die Anzahl der Elemente auch als Kardinalität bezeichnet
- Kardinalität der Menge J: $|J|$

Funktionen - Max, Min

Berechnung des Maximums und Minimums:

```
scalar
 MA Maximum
 MI Minimum ;

MA = max(1592,(3540/2),7);
MI = min(-15,12,-6);
display MA,MI;
```

```
----      7 PARAMETER MA                  =     1770.000  Maximum
            PARAMETER MI                  =      -15.000  Minimum
```

Funktionen - SMax, SMin

Oftmals ist das Maximum bzw. Minimum eines Parameters oder einer Variable gesucht.

- Maximum-Funktion über einen/mehrere Indices:

```
set i Beispielmenge /1*5/ ;
parameter
 c(i) Beispieldaten /1 10, 2 5, 3 100, 4 -3, 5 13/ ;
scalar
 SMA Daten-Maximum
 SMI Daten-Minimum
;

SMA = smax(i, c(i) ) ;
SMI = smin(i, c(i) ) ;

display
sma,smi ;
```

```
----     21 PARAMETER SMA                =      100.000  Daten-Maximum
           PARAMETER SMI                =       -3.000  Daten-Minimum
```


Funktionen - Runden und Zufallszahlen

- Uniform(x,y) erzeugt eine gleichverteilte (Pseudo)-Zufallszahl zwischen x und y
- eine ganzzahlige Zufallszahl liefert die Funktion Uniformint(x,y)

```
----      6 PARAMETER r  Beispielparameter
A 1.717,    B 8.433,    C 5.504,    D 3.011
----      8 PARAMETER r  Beispielparameter
A 3.000,    B 2.000,    C 3.000,    D 9.000
----     10 PARAMETER r  Beispielparameter
A  0.700,    B  5.000,    C 10.000,    D  5.800
```

Die Funktion round(x,y) rundet die Zahl x auf y Stellen. Mit floor(x,y) und ceil(x,y) kann auch gezielt ab bzw. aufgerundet werden.

Funktionen - Runden und Zufallszahlen

Hinweis zu Zufallszahlen:

Es werden sog. Pseudozufallszahlen erstellt. Ruft man das Skript mehrmals nacheinander auf so werden immer wieder diesselben Zufallszahlen erzeugt. Der Zufallszahlengenerator kann durch

Option Seed=number;

eingestellt werden. Der voreingestellte Standardwert für number ist 3141. Wird der „seed“ verändert, so werden neue (Pseudo)-Zufallszahlen erstellt.

$-Bedingungen

Mit $(...) können Bedingungen umgesetzt werden. Beispielsweise kann die Wertzuweisung eines Parameters mit einer Bedingung verbunden werden.

```
set
 i /1*5/ ;
parameter
 c(i) Parameter ;

c(i)$(ord(i) > 2) = 5 ;
display c ;
```

```
----      7 PARAMETER c  Parameter

3 5.000,    4 5.000,    5 5.000
```

Die Wertzuweisung auf der linken Seite wird nur dann vorgenommen, wenn die $( )-Bedingungen erfüllt ist. Der Wert von c_i wird also nur für diejenigen Elemente der Menge i zugewiesen, deren Position in der Menge i größer als 2 ist. Für c_1 und c_2 wird die Wertzuweisung demnach nicht aufgestellt (vgl.ord()-Funktion: Folie 53).

$-Bedingungen

Auch Gleichungen können an Bedingungen geknüpft werden.

Im Beispiel soll eine Restriktion nur für das letzte Element einer Menge aufgestellt werden:

```
equations
 Gleichung1 Dies ist eine Gleichung ;

Gleichung1(i)$(ord(i) = card(i)).. sum( (i,j), x(i,j) ) =E= 1 ;
```


$-Bedingungen - Operatoren

Logische Operatoren

Operator	Bedeutung
not	nicht
and	und
or	oder
xor	exklusiv oder

Numerische Operatoren

Operator		Bedeutung
lt	<	kleiner als
le	<=	kleiner oder gleich
eq	=	gleich
ne	<>	ungleich
ge	>=	größer oder gleich
gt	>	größer

Beispiel

```
c(j)$( (ord(j) > 6) AND (ord(j) <> 10) ) = 10 ;
```

Optimalitätskriterium

Um die Lösungszeit eines MIPs zu verringern, kann angegeben werden, wie groß die Lücke der zu ermittelnden Lösung zu der theoretisch noch bestmöglichen ganzzahligen Lösung sein darf. Sobald im Branch-&-Bound Prozess eine zulässige ganzzahlige Lösung gefunden wird, deren Zielfunktionswert eine geringere Differenz zum noch theoretisch bestmöglichen Ergebnis aufweist, wird der Lösungsprozess mit der Meldung 'solution satisfies tolerances' abgebrochen.

Angabe der Lücke, dem sog. solution gap, vor dem solve Statement:

```
*Angabe eines absoluten gaps: optca (criterium absolute)
option optca = 5000;
*maximale Abweichung der ermittelten Lösung von der Optimallösung:
*5000
```

Bei Angabe einer maximalen relativen Lücke:

```
*Angabe eines relativen gaps: optcr (criterium relative)
option optcr = 0.1;
*maximale Abweichung der ermittelten Lösung von der Optimallösung:
*10 Prozent
```


Optimalitätskriterium

Die Standardeinstellung ist `optcr = 0.1`. Soll bis zum Nachweis der Optimalität gerechnet werden muss *optcr = 0* gesetzt werden.

```
option optcr = 0;
```

```
MIP status(101): integer optimal solution
Fixed MIP status(1): optimal
Proven optimal solution.

MIP Solution:        96394.100000
Final Solve:         96394.100000

Best possible:       96394.100000
Absolute gap:            0.000000
Relative gap:            0.000000
```

Die maximale relative Abweichung des Zielfunktionswertes von der bestmöglichen Lösung ist auf 0% gesetzt. Somit wird die optimale Lösung berechnet. Der Rechenaufwand dafür kann sehr groß sein.

```
option optcr = 0;
Option optca=4000;
```

```
MIP status(102): integer optimal, tolerance
Fixed MIP status(1): optimal
Solution satisfies tolerances.

MIP Solution:        96394.100000
Final Solve:         96394.100000

Best possible:       92485.800000
Absolute gap:         3908.300000
Relative gap:            0.040545
```

Die maximale absolute Abweichung des Zielfunktionswertes von der bestmöglichen Lösung beträgt 4000. Sobald eine ganzzahlige Lösung gefunden wird, die eine Optimalitätslücke von höchstens 4000 aufweist, wird die Berechnung beendet. Um `optca` zu setzen muss zunächst die Standardeinstellung für `optcr` (Standard: optcr=0.1) aufgehoben werden.

Optionen - Rechenzeit und Iterationen

```
S O L V E      S U M M A R Y

     MODEL   schiffsbeladung     OBJECTIVE  F
     TYPE    MIP                 DIRECTION  MAXIMIZE
     SOLVER  CPLEX               FROM LINE  56

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE            13519.9117

 RESOURCE USAGE, LIMIT          0.109      1000.000
 ITERATION COUNT, LIMIT         7          10000
```

Maximal zulässiger Zeitbedarf zum Lösen (default: 1000 Sekunden)

Maximal zulässige Anzahl von Iterationen zum Lösen (default: 10000 Iterationen)

Ausschnitt A_4_5_big.lst

vom Solver zur Lösung benötigte Zeit (CPU-Zeit)

vom Solver zur Lösung benötigte Iterationen

Optionen - Rechenzeit und Iterationen

Bei größeren Problemen kann eine Erhöhung notwendig sein, da bspw. innerhalb der vorgegebenen Zeit keine Lösung gefunden wird.

Bsp. für Zeitüberschreitung:

```
Cplex 10.0.1, GAMS Link 31
Cplex licensed for 1 use of lp, qp, mip and barrier, with 2 parallel threads.

Resource limit exceeded, no integer solution found.

No solution returned
```

Ausschnitt A_4_5_big.lst

Optionen - Rechenzeit und Iterationen

Verändern der Zeit- bzw. Iterationsbeschränkung durch model.reslim und model.iterlim vor dem solve Statement:

```
aufgabe.reslim=20000;
solve schiffsbeladung maximizing F using mip ;
```

Erhöhung der Rechenzeit auf 20.000 Sekunden

```
RESOURCE USAGE, LIMIT          0.109      20000.000
```

Ausschnitt A_4_5_big.lst

```
aufgabe.iterlim=50000;
solve schiffsbeladung maximizing F using mip ;
```

Erhöhung der Iterations-anzahl auf 50.000

```
ITERATION COUNT, LIMIT         7         50000
```

Ausschnitt A_4_5_big.lst

Hinweis zum Setzen von Optionen

Optionen können in den meisten Fällen sowohl für alle Modelle des aktuellen GAMS-Laufes als auch für spezifische Modelle gesetzt werden. Bspw. kann die Zahl der Iterationen entweder über:

```
option iterlim = 10000 ;
```

für alle Modelle, oder über:

```
modelname.iterlim = 10000 ;
```

für ein spezifisches Modell (modelname) gesetzt werden.

Optionen - Sonstiges

Veränderung der Anzahl der im Listing angezeigten Spalten (Variablen) bzw. Zeilen (Equations) durch

```
option limcol = number ;
```

```
option limrow = number ;
```


Sonstiges

- $ sowie * müssen immer an den Zeilenanfang (nicht einrücken)
- Mehrzeilge Kommentare können zwischen `$Ontext` und `$Offtext` geschrieben werden
- Titel können durch `$title Beispieltitel` gesetzt werden
- $Exit beendet das Skript in der entsprechenden Zeile

If-Else Verzweigung

```
if( y<5,
            x = 10 ;
elseif y>5,
            x = 2 ;
else        x = 0 ;
) ;
```

- `elseif` ist optional und kann beliebig oft verwendet werden
- mehrere Bedingungen am besten gut einklammern
- keine Deklaration und keine Definition von equations innerhalb if, aber z.B. solve zum mehrfachen Lösen von Modellen

Schleifen: loop

```
loop( j,
     Anweisungen ;
) ;
```

Menge, für deren Elemente die Anweisungen ausgeführt werden sollen

```
loop((i,j),
     Anweisungen ;
) ;
```

mehrere Indizes durch Komma trennen und in Klammern setzen

```
loop( j$(y.l(j)=1),
        Anweisungen ;
        loop( i,
              Anweisungen ;
        ) ;
) ;
```

geschachtelte Schleife, wobei die Anweisungen nur für die Elemente von j ausgeführt werden, für die die Variable y(j) den Wert 1 hat (bedingte loop)

- wiederholte Anweisungen über einen Index
- keine Deklaration und keine Definition von equations innerhalb loop

Schleifen: for

Mehrmalige Ausführung mit einem Zähler

```
scalar zaehler, p / 0 /;
      von Startwert   bis   Endwert
for( zaehler = 1 to 10,
                        p = p + 10 ;
) ;
      Zähler (scalar, kein set)
display p ;
```

default: Erhöhung des Zählers pro Durchlauf um 1

→ Anweisung wird 10mal ausgeführt

```
----      16 PARAMETER p                 =      100.000
```

```
scalar zaehler, p / 0 /;
for( zaehler = 1 to 10 by 2,
                        p = p + 10 ;
) ;
```

Erhöhung des Zählers pro Durchlauf um 2

→ Anweisung wird 5mal ausgeführt

```
----      16 PARAMETER p                 =       50.000
```

- keine Deklaration und keine Definition von equations innerhalb for

Schleifen: while

Mehrmalige bedingte Ausführung

```
parameter    p  /0/ ;

                wiederhole die folgenden Anweisungen solange
                die Bedingung erfüllt ist
while( p<=20,
      a(s) = a(s) + p *a(s) ;
      solve transport minimizing F using lp ;
      a(s) = a(s)/(1+p) ;
      p = p + 5;
) ;
```

Anweisungen, die pro Durchlauf ausgeführt werden

- keine Deklaration und keine Definition von equations innerhalb while, aber z.B. solve zum mehrfachen Lösen von Modellen

Ausgabedateien - Beispiel Transportplanung

- Mit GAMS können .txt - Ausgabedateien erstellt werden.
- Deklaration + Definition einer Ausgabedatei:

```
file                out       / Ausgabe_Transport.txt /   ;
```

Schlüsselwort zur Deklaration von Dateien	Eindeutiger Bezeichner (Name)	Dateiname mit Endung und ggf. Pfadangabe; ohne Pfad: Speicherung am Speicherort der aktuellen .gpr-Datei

- Festlegung, in welche Datei geschrieben werden soll

```
put                 out    ;
```

Schlüsselwort zum Schreiben in Dateien	Bezeichner der Datei, in die ab sofort geschrieben werden soll

Ausgabedateien - Beispiel

```
file out /Ausgabe_Transport.txt/;
put out;

put "Transportproblem"  //;
put a.ts /;

loop(s,
     put @3, s.tl , @10, a(s)/;
);
put /;
```

Ausgabe von Text erfolgt in "..." bzw. '...'

Zeilenumbruch /

Schreibe Kommentar zum parameter a

„für jedes Element aus der Menge s wiederhole"

Schreibe das, was folgt, in die 3. Spalte der aktuellen Zeile (@Spaltennr.)

Name des betrachteten Elements (.tl)

Wert des Parameters a(s) des betrachteten Elements

Leerzeile

Trennung der Einträge hinter put durch Komma (zu empfehlen) oder Leerzeichen

Ausgabedateien - Beispiel Fortsetzung

```
put b.ts /;
loop(k,
     put @3, k.tl , @10, b(k)/;
);
put /;

put "Transportkosten: " z.l /;
put /;

put "Transportmengen" //;

put "von":8, "nach":8, "Menge":>8 //;
loop((s,k),
          put s.tl:8, k.tl:8, x.l(s,k):8:2 /;
);
```

Zielfunktionswert (Variable) schreiben

Anzahl der Stellen, die für diesen Eintrag reserviert werden (Feldbreite)

Ausrichtung des Eintrags innerhalb der reservierten Feldbreite, hier: rechtsbündig

Anzahl an Dezimalstellen von der Feldbreite, hier: 2 von 8 Stellen (nur bei numerischen Einträgen)

Ausgabedateien - Beispiel Ergebnis

Entstehende Ausgabedatei

```
Transportproblem

Kapazität am Standort s
  B            350.00
  HH           500.00
  M            550.00

Nachfrage des Kunden k
  DD           300.00
  F            300.00
  K            250.00
  S            150.00
  H            350.00

Transportkosten:     1910.00

Transportmengen

von      nach       Menge

B        DD        300.00
B        F           0.00
B        K           0.00
B        S           0.00
B        H          50.00
HH       DD          0.00
HH       F           0.00
HH       K         200.00
HH       S           0.00
HH       H         300.00
M        DD          0.00
M        F         300.00
M        K          50.00
M        S         150.00
M        H           0.00
```

Ausgabe_Transport.txt

Ausgabedateien - Beispiel Fortsetzung

Transportmengen als Tabelle - Quellcode

```
put "":10:0;
loop(k,
      put k.tl:>10 ;
);
put /;
scalar i ;
for(i=1 to 60,
             put "-";
);
put /;
loop(s,
      put s.tl:10:0  ;
      loop(k,
            put x.l(s,k):10:2 ;
      );
      put /;
);
```

Tabellenkopf erzeugen

Strich über gesamte Tabellenbreite erzeugen

geschachtelte loop-Anweisung zur Ausgabe einer Tabelle

Äußerer loop: Zeilen

Innerer loop: Spalten

Ausgabedateien - Beispiel Fortsetzung

Transportmengen als Tabelle - Ausgabedatei

	DD	F	K	S	H
B	300.00	0.00	0.00	0.00	50.00
HH	0.00	0.00	200.00	0.00	300.00
M	0.00	300.00	50.00	150.00	0.00

Ausgabedateien - Hinweise

- Normalen Text zwischen zwei Anführungszeichen ("Text") schreiben
- Elementnamen: Menge.tl
- Elementkommentare: Menge.ts
- Wert/Schranken von variables/equations: Bezeichner.l/.m/.lo/.up(Index)
- Möglichkeit zur einfachen Ausgabe formatierter Dateien zum Einlesen in Excel oder Datenbanken (leerzeichen-, tabstopp- bzw. kommagetrennt)

Ausgabedateien - Standardformatierung

Standardformat	Feldbreite	Ausrichtung
Text	0 Stellen	linksbündig
Namen	12 Stellen	linksbündig
numerische Objekte	12 Stellen	rechtsbündig
Set-Elemente	12 Stellen	rechtsbündig

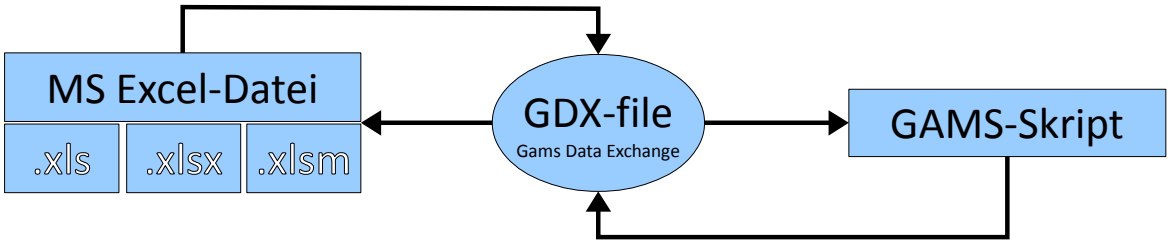
Anpassung für einzelne Output-Elemente: (vgl. Folie 77)

```
Objekt:  Ausrichtung Feldbreite:Dezimalstellen
```

Ausgabe-objekt	Optional: > Rechtsbündig < Linksbündig <> zentriert	Anzahl der Stellen des reservierten Platzes	Anzahl der Dezimalstellen an der Gesamtzahl der Stellen

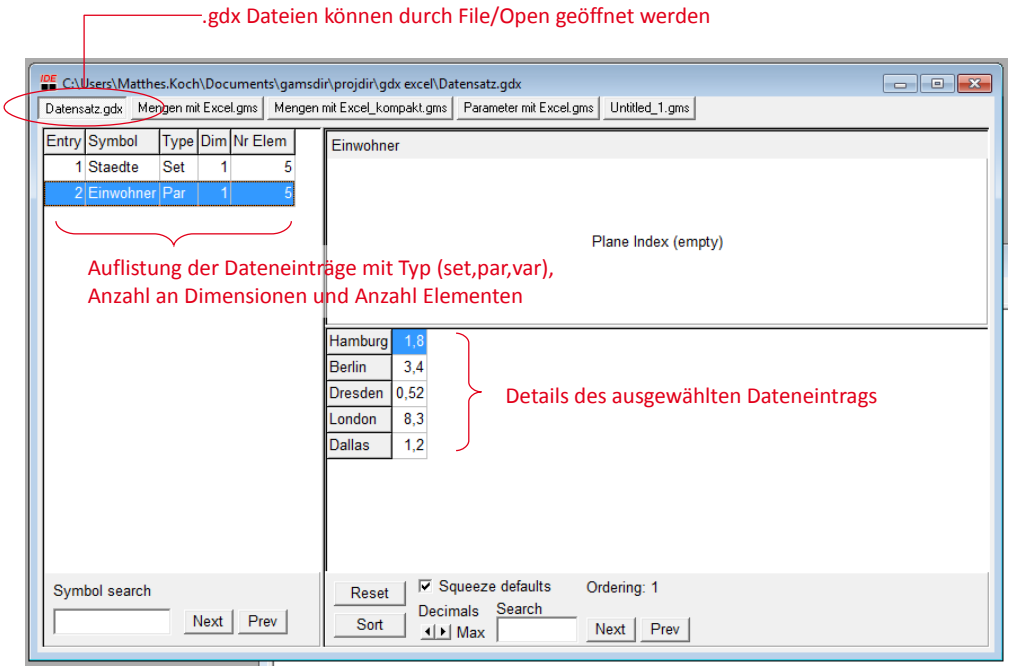
Gams Data Exchange

- Mit GAMS können Daten aus MS Excel eingelesen und in eine Excel-Datei geschrieben werden.
- Dazu wird die Funktion GDXXRW verwendet
- Der Datenaustausch erfolgt über .gdx (gams data exchange) Dateien



GDX Daten

GDX-Dateien speichern Daten in für GAMS lesbarer Form. Beispiel für eine GDX Datei:



Einlesen und Schreiben von GDX Daten

Vor der Verwendung müssen die GDX-Daten dem Gamsskript bekannt gemacht werden. Dafür wird die gdx-Datei geöffnet, die entsprechenden Daten werden gelesen und mit $gdxin wird die GDX-Datei wieder geschlossen. Die Deklaration der Daten erfolgt vorher separat!

```
set Staedte ;
Parameter Einwohner(Staedte) ;
*Laden aus gdx:
$gdxin Datensatz.gdx
$load Staedte Einwohner
$gdxin

display Staedte, Einwohner;
```

```
----      8 SET Staedte

Hamburg,    Berlin ,    Dresden,    London ,    Dallas


----      8 PARAMETER Einwohner

Hamburg 1.800,    Berlin  3.400,    Dresden 0.520,    London  8.300
Dallas  1.200
```

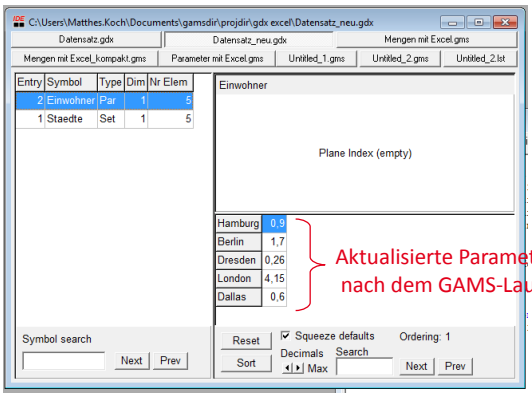

Einlesen und Schreiben von GDX Daten

Sollen Daten nach dem GAMS-Lauf in einer GDX-Datei gespeichert werden, so kann folgende Funktion verwendet werden:

```
Einwohner(Staedte) = Einwohner(Staedte)/2;

execute_unload "Datensatz_neu.gdx" Staedte Einwohner
```

Werden dabei keine Daten angegeben, so werden alle derzeit geladenen Daten nach .gdx entladen.

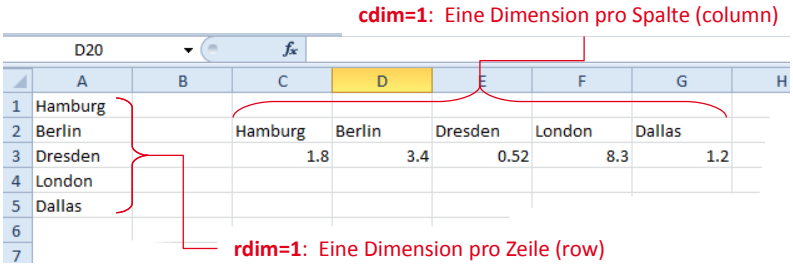


Datenaustausch

Der Datenaustausch von GDX zu MS Excel erfolgt über das Gams-Programm GDXXRW.

```
execute 'gdxxrw.exe Datensatz.gdx o=meinWB.xlsx set=Staedte rng=Tabelle1!A1:A5 rdim=1 cdim=0' ;
execute 'gdxxrw.exe Datensatz.gdx o=meinWB.xlsx par=Einwohner rng=Tabelle1!C2:G2 rdim=0 cdim=1';
```

Ergebnis in Tabelle1 von meinWB.xlsx:



Mit `o=Workbookname` wird das zu beschreibende Workbook angegeben. Wird die Angabe weggelassen, so wird ein Workbook mit dem Namen der GDX Datei verwendet. Sollte die Datei noch nicht existieren, wird sie angelegt. Die zu beschreibende Datei darf nicht in Excel geöffnet sein (Schreibschutz).

Datenaustausch

Der Datenaustausch von MS Excel zu GDX wird ebenfalls über GDXXRW realisiert.

```
$call 'gdxxrw meinWB.xlsx o=Input.gdx set=Staedte rng=Tabelle1!A1:A5 rdim=1 cdim=0' ;
```

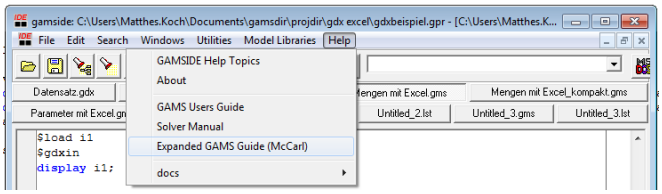
Es wird die .gdx Datei erstellt (überschrieben) und mit Werten gefüllt.

- Liest man Mengen ein, kann in der Tabellenzelle neben/unter dem Mengenelement ein Kommentar angegeben werden
- es gibt verschiedene Optionen zum Einlesen:
  - `SE=0` Skip Empty: erspart die Angabe eines genauen rng (wie oben A1:A5) - es ist möglich nur die erste Zelle des Bereiches anzugeben (z.B. `rng=Tabelle1!A1`). Sobald die durch SE angegebene Anzahl an leeren Zellen erreicht ist, wird das Einlesen des aktuellen Datenelements beendet
  - `values=yn` für Mengenelemente: es werden nur Elemente akzeptiert, deren Kommentarzelle „YES“,“Y“ oder „1“ enthält. Mit „0“,“N“,“NO“ können damit einzelne Set-Elemente ausgeschaltet werden.
- enthält die Tabelle Duplikate, kann `dset=...` anstelle von `set=...` verwendet werden, um Mengenelemente einzulesen.

Datenaustausch - Fortgeschrittene Methoden

- kompakte Schreibweise des GDXXRW-Aufrufes durch Verwendung eines Index-Blattes innerhalb der zu lesenden Excelmappe
- Zusammenführung mehrerer GDX-Dateien zu einer GDX-Datei (gdx merge)
- Mehrdimensionale Parameter und Mengen

Ausführliche Darstellung fortgeschrittener Methoden des Datenaustausches finden sich mit Beispielen im Expanded Gams Guide von Bruce McCarl



- Tipp: Für statische Daten ist das Einbinden einer .txt Datei mit $Include der einfachste Weg externe Daten einzubinden. Excel-Tabellen lassen sich als .txt Dateien exportieren

Quellen

- Förster, A. (2009): *Foliensatz zur Gams-Übung*, Technische Universität Dresden
- McCarl, B. A. (2006): Expanded GAMS Guide (`http://www.gams.com`)
- GAMS Development Corporation (Hrsg.) (2006), GAMS - A User's Guide, Books on Demand, Norderstedt (bzw. `http://www.gams.com` o. GAMS - Menü Help)
- GAMS IDE Documentation (`http://www.gams.com` bzw. GAMS-Menü Help-Help Topics)
- GAMS Homepage: `http://www.gams.com`