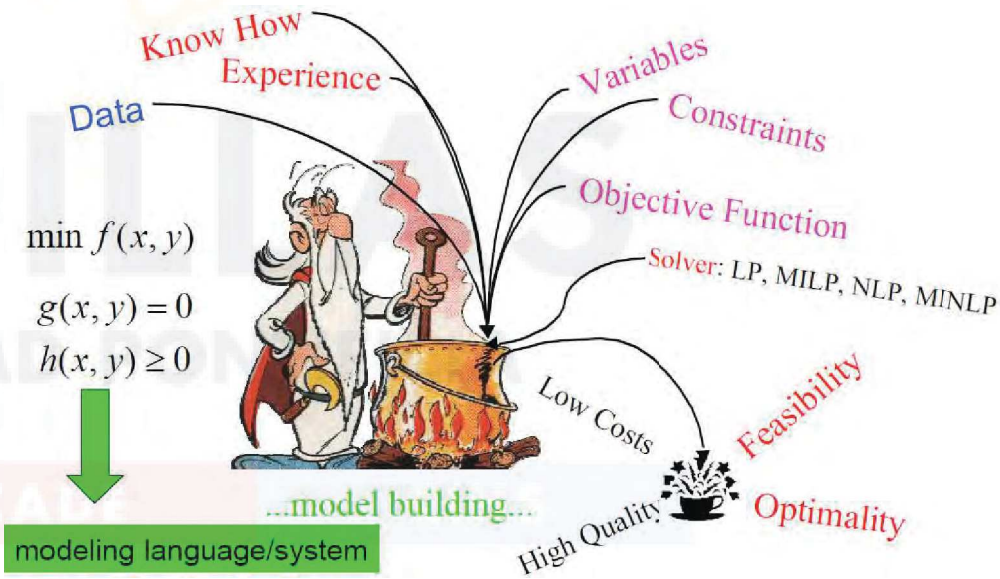
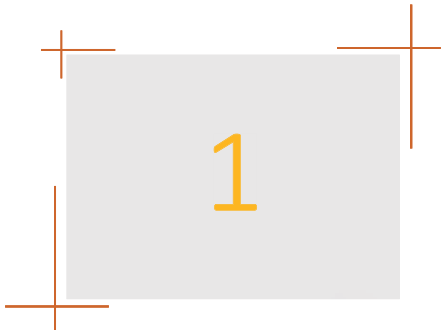
Do not confuse the ingredients of the recipe

- Mathematical formulation
  - LP, MIP, QCP, MCP
- Language
  - GAMS
- Solver
  - CPLEX, GUROBI, PATH
- Solver algorithm
  - Primal simplex, dual simplex, interior point
- Input/output interfaces
  - Text file, CSV, Excel, Matlab, Access
- Operating system
  - Windows, Linux, MacOS
- User-developed algorithm
  - Benders decomposition, Lagrangian relaxation, GA
- Stochastic extensions
  - EMP



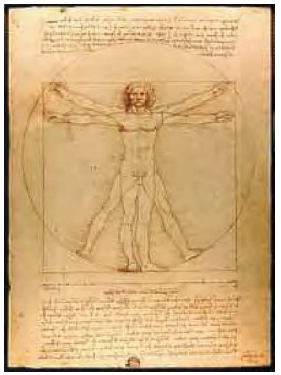
Source: http://www.gams.com/presentations/present_modlang.pdf

1. Programming Style
2. GAMS Code
3. Look and Feel
4. Mathematical Formulation
5. Advanced Algorithms



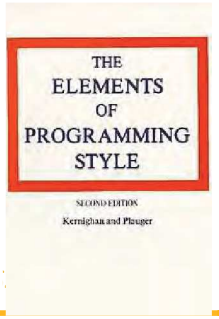
Programming Style

Programming



- Discipline whose control is basic in many engineering projects
  - Science: thinking, discipline, rigorousness and experimentation
  - Art: beauty and elegance

- A good design is fundamental
- Before writing any code the optimization problem must be written algebraically
- Learning by reading
- Coding by gradual refinement, incremental implementation
- Use a mockup for development and verification of the model
- Be careful with the details ("God is in the detail")

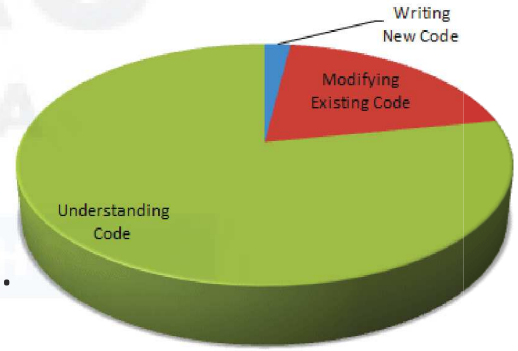


B.W. Kernighan and P.J. Plauger, ***The Elements of Programming Style***, McGraw Hill, New York, 1978 http://en.wikipedia.org/wiki/The_Elements_of_Programming_Style

General recommendations

- Act according to the Pareto principle
  - 20 % takes to create the first prototype
  - 80 % of code development is devoted to maintenance and refinement
- MAINTAINABILITY and reusability are crucial
- Code is developed to be read by humans, not by machines. Write code for understanding the model, not for obscuring it.
- Say what you mean, simply and directly.
- Don't stop with your first draft. Refine it.

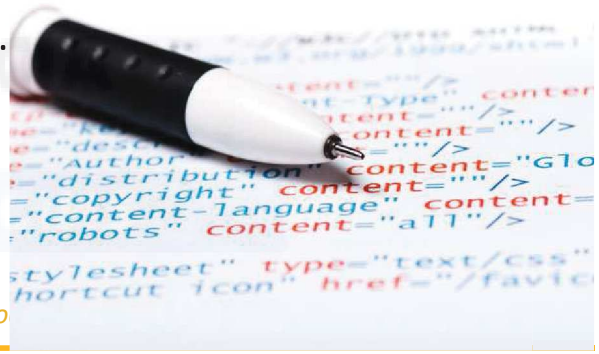
http://www.jetbrains.com/



Good Optimiza http://blog.codinghorror.com/when-understanding-means-rewriting/

Documentation. Comments

- It is a crucial task in code development
  - In particular, GAMS was born to explicitly include documentation into the code.
- Code must be self-documented
- Illustrative comments and well localized
- Make sure comments and code agree.
- Don’t just echo the code with comments - make every comment count.
- Don’t comment bad code or tricks - rewrite it.
- Don’t patch bad code - rewrite it.
- Don’t over-comment.



Code style

- Any project manager ought to define the style before starting up a multiple participant project (or maybe just for his/her own help)
- Systematic and consistent use of uppercase and lowercase letters
  - Use lowercase letters instead of uppercase. We are more used to read lowercase letters.
- Clean code, take care of the aesthetics when coding
  - Aesthetics is as important as the content. Code must be read at a glance.
- Format the code to help the reader understand it.
  - Indent to show the logical structure of a program.
  - Keep coherence in the coding rules (indent in repetitive sentences)
  - Align code to show patterns.
  - Make the reading easier (parallelism among consecutive similar sentences, indent)
- Use meaningful and long names for identifiers. Same use of identifiers in different parts of the code.

Efficiency vs. Clarity

- Make it clear and right before you make it faster
- Keep it simple to make it faster
- Don’t sacrifice clarity for small gains in efficiency

1. Programming Style
2. GAMS Code
3. Look and Feel
4. Mathematical Formulation
5. Advanced Algorithms

2

GAMS Code

Learning by reading first, and then by doing

- GAMS Model Libraries

  (https://www.gams.com/modlibs/)

- Decision Support Models in the Electric Power Industry
  (https://www.iit.comillas.edu/aramos/Ramos_CV.htm#ModelosAyudaDecision)

StarGen Lite (**Short Term Stochastic Daily Unit Commitment Model**) demo GAMS version
StarGen Lite (**Medium Term Stochastic Hydrothermal Coordination Model**) demo GAMS version
StarGen Lite (**Long Term Generation Expansion Planning Model**) demo GAMS version
StarGen Lite (**Probabilistic Production Cost Model**) demo GAMS version

StarNet Lite (**Long Term Transmission Expansion Planning Model**) demo GAMS version

StarMkt Lite (**Cournot Market Equilibrium Model**) demo GAMS version
StarMkt Lite (**Bushnell Market Equilibrium Model**) demo GAMS version
StarMkt Lite (**Long Term Market Generation Expansion Planning Model**) demo GAMS version

StarGen Lite (**Short Term Stochastic Daily Unit Commitment Model**) demo Julia version

Developing in GAMS

- Development environment gamside

aaa.gpr

- Documentation
  - GAMS Documentation Center https://www.gams.com/latest/docs/
  - GAMS Support Wiki https://support.gams.com/
  - Bruce McCarl's GAMS Newsletter https://www.gams.com/community/newsletters-mailing-list/
- Users guide Help > GAMS Users Guide
- Solvers guide Help > Expanded GAMS Guide

- Model: FileName.gms
- Results: FileName.lst
- Process log: FileName.log

Blocks in a GAMS model

- Mandatory
  - variables
  - equations
  - model
  - solve

- Optional
  - sets: (alias)
    - alias (i,j) i and j can be used indistinctly
    - Checking of domain indexes
  - data: scalars, parameters, table

General structure of GAMS sentences

- Commenting
  - Lines with * in the first column
  - `$OnText $OffText` to comment many lines
- No distinction between uppercase and lowercase letters
- Parenthesis (), square bracket [] or braces {} can be used indistinctly to distinguish levels
- Language reserved words appear in bold
- Sentences end with a ;
  - Can be suppressed when the following word is a reserved one

Basic input/output in text format

- Data input from a text file

```
$include FileName.txt
display IdentifierName (shows its content or value)
```

- Data output to a text file

```
file InternalName / ExternalName.txt /
put      InternalName
   put   IdentifierName
putclose InternalName
```

- Specific options to control the output format
  - Put Writing Facility

Functions and operators

- +, -, *, /, ** or power(x,n)
- abs, arctan, sin, cos, ceil, floor, exp, log, log10, max, min, mod, round, sign, sqr, sqrt, trunc, normal, uniform
- gyear, gmonth, gday, ghour, gminute, gsecond, gdow, gleap, jdate, jnow, jstart, jtime
- lt <, gt >, eq =, ne <>, le <=, ge >=
- not, and, or, xor
- diag(set_element,set_element)={1,0}
- sameas(set_element,set_element)={T,F}
- ord, card ordinal and cardinal of a set, SetName.pos ordinal of a set
- sum, prod, smax, smin
- inf, eps are valid as data

Model temporal license

```
abort $[jstart > jdate(2018,5,15)] 'License for this model has expired and it cannot be used any more. Contact the developers'
```

$ Operator in assignments, summations, constraints

- Sets a condition

```
$(value > 0)        $(number1 <> number2)
```

- On the left of an assignment, it does the assignment ONLY if the condition is satisfied

```
if (condition,
    DO THE ASSIGNMENT
) ;
```

- On the right of an assignment, it does the assignment ALWAYS and if the condition is not satisfied it assigns value 0

```
if (condition,
    DO THE ASSIGNMENT
else
    ASSIGNS VALUE 0
) ;
```

Index shifting. Lag and lead

- `t=J,F,M,A,MA,J,JU,AU,S,O,N,D`

  `vReserve(t) + pInflow(t) - vOutflow(t) =e= vReserve(t+1)`

- Vector values out of the domain are 0

  `vReserve('D') + pInflow('D') - vOutflow('D') =e= 0`

- Circular sequence of an index (++, --)

  `t=J,F,M,A,MA,J,JU,AU,S,O,N,D`

  `vReserve(t) + pInflow(t) - vOutflow(t) =e= vReserve(t++1)`

  `vReserve('D') + pInflow('D') - vOutflow('D') =e= vReserve('J')`

- Inverted order sequence of `PP` index even though `t` is traversed in increasing order

  `PP(t+[card(t)-2*ord(t)+1])`

Operations with sets

- Intersection

```
D(a) = B(a) * C(a)
```

- Union

```
D(a) = B(a) + C(a)
```

- Complementary

```
D(a) = NOT C(a)
```

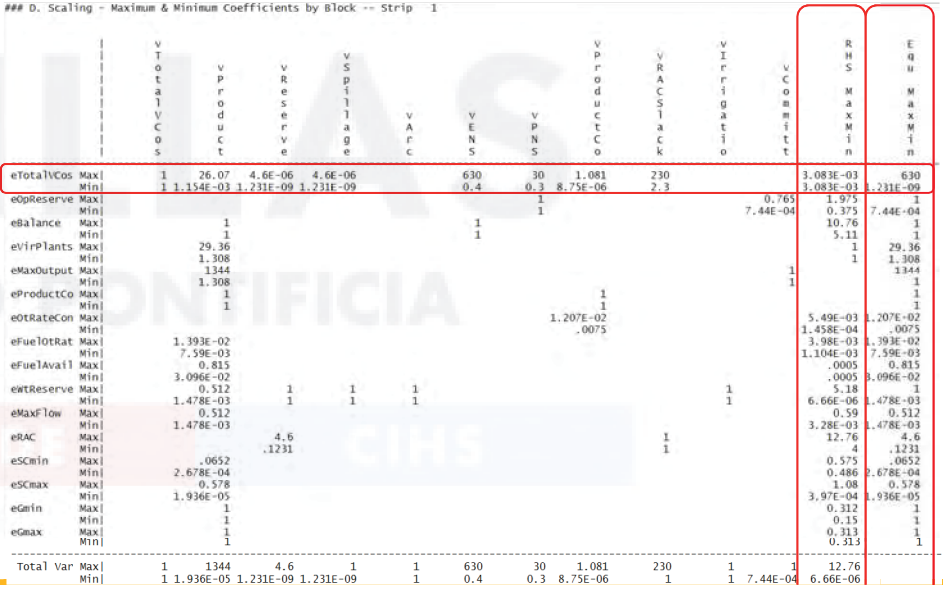
- Difference

```
D(a) = B(a) - C(a)
```

Analytical computation of constraints and variables

- Important to know the estimated size of the optimization problem and dependence with respect to basic elements
- It can be used for detecting formulation errors
- Use LimRow/LimCol
- Suitable to know the constraint matrix structure (GAMSChk)

```
option LP=GAMSChk
```



D. Scaling - Maximum & Minimum Coefficients by Block -- Strip 1

		vTotalVCos	vProduct	vReserve	vSpillage	vArc	vENS	vPNS	vProductCo	vRACSlack	vIrrigation	vCommit	RHS Max	RHS Min	Equ Max	Equ Min
eTotalVCos	Max|	1	26.07	4.6E-06	4.6E-06		630	30	1.081	230			3.083E-03		630	
	Min|	1	1.154E-03	1.231E-09	1.231E-09		0.4	0.3	8.75E-06	2.3			3.083E-03		1.231E-09	
eOpReserve	Max|							1				0.765	1.975		1	
	Min|							1				7.44E-04	0.375		7.44E-04	
eBalance	Max|		1				1						10.76		1	
	Min|		1				1						5.11		1	
eVirPlants	Max|		29.36										1		29.36	
	Min|		1.308										1		1.308	
eMaxOutput	Max|		1344									1			1344	
	Min|		1.308									1			1	
eProductCo	Max|		1						1						1	
	Min|		1						1						1	
eOtRateCon	Max|								1.207E-02				5.49E-03		1.207E-02	
	Min|								.0075				1.458E-04		.0075	
eFuelOtRat	Max|		1.393E-02										3.98E-03		1.393E-02	
	Min|		7.59E-03										1.104E-03		7.59E-03	
eFuelAvail	Max|		0.815										.0005		0.815	
	Min|		3.096E-02										.0005		3.096E-02	
eWtReserve	Max|		0.512	1	1	1						1	5.18		1	
	Min|		1.478E-03	1	1	1						1	6.66E-06		1.478E-03	
eMaxFlow	Max|		0.512										0.59		0.512	
	Min|		1.478E-03										3.28E-03		1.478E-03	
eRAC	Max|			4.6						1			12.76		4.6	
	Min|			.1231						1			4		.1231	
eSCmin	Max|		.0652										0.575		.0652	
	Min|		2.678E-04										0.486		2.678E-04	
eSCmax	Max|		0.578										1.08		0.578	
	Min|		1.936E-05										3.97E-04		1.936E-05	
eGmin	Max|		1										0.312		1	
	Min|		1										0.15		1	
eGmax	Max|		1										0.313		1	
	Min|		1										0.313		1	
Total Var	Max|	1	1344	4.6	1	1	630	30	1.081	230	1	1	12.76			
	Min|	1	1.936E-05	1.231E-09	1.231E-09	1	0.4	0.3	8.75E-06	1	1	7.44E-04	6.66E-06			

Source: MPES

How big is a big optimization problem

- Memory requirements for creating the model (GAMS)
  - 1 GB for every 1 million rows
- Memory requirements for solving the model (solver)
  - Depends on the difficulty in solving the model
    - Integrality gap for MIP problems

Avoid creation of superfluous constraints and variables

- Or how to achieve a compact formulation (small size of the constraint matrix or small density)
- Some redundant constraints can introduce a tighter model, see later
- However, introduce logical conditions (with a $ in GAMS) in the creation of equations or the use of variables to avoid superfluous ones
- Reduction rules: mathematical reasoning or common sense based on the problem context
  - Flows by non existing connections in a network
- Solvers can detect some of these superfluous equations/variables but it is more efficient to avoid their creation (pre-processing)
- Profile, ProfileTol

Debugging an optimization model

- Grammar error
  - Read the error and click in the red line of the log file

- Infeasibility detection
  - Soft (elastic) constraints
    - Introduce a deficit or surplus variable in each equation and penalize them in the objective function. Be careful with penalty parameter
  - Detect the smallest core of infeasible constraints by the LP solver (option *Irreducible Infeasible Subsets iis* in solvers)
    - Once known, they have to be deleted or modified
    - FeasOpt in GUROBI

Efficiency in GAMS code usage (loop)

```
set i / 1*2000 /
alias (i,ii)
parameter X(i,i)

loop ((i,ii),
   X(i,ii) = 4 ;
) ;
```

75.3 s

```
set i / 1*2000 /
alias (i,ii)
parameter X(i,i) ;


X(i,ii) = 4 ;
```

0.3 s

Efficiency in GAMS code usage (index order)

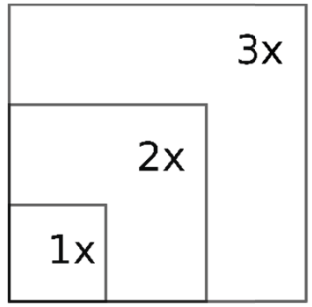
```
option Profile=10, ProfileTol=0.01

set i / 1*200 /
    j / 1*200 /
    k / 1*200 /
parameter X(k,j,i), Y(i,j,k) ;

Y(i,j,k) = 2 ;

X(k,j,i) = Y(i,j,k)
```

4.5 s

```
option Profile=10, ProfileTol=0.01

set i / 1*200 /
    j / 1*200 /
    k / 1*200 /
parameter X(i,j,k), Y(i,j,k) ;

Y(i,j,k) = 2 ;

X(i,j,k) = Y(i,j,k)
```

1.3 s

Scaling



- Solvers are powerful but not magic
- Input data and output results must be in commonly used units
- But internally variables, equations, parameters must be around 1 (i.e., from 0.01 to 100). Ratio of largest to smallest matrix coefficient should be < 10^5
- Scaling can be done:
  - Manually (e.g., from MW to GW, from euros to Meuros). Modelers can typically do better because they know the problem
  - Automatically by the language (ModelName.ScaleOpt=1)
  - By the solver (`ScaInd 1` in CPLEX, `ScaleFlag 2` in GUROBI)
- Specially useful in large-scale LP problems or NLP problems and/or when willing to use the dual variables
- The condition number is a measure for the sensitivity of the solution of a system of linear equations to errors in the data.
  - It is the ratio between the largest and smallest eigenvalues
- Condition numbers <10^6 are good enough. Numerical problems arise for condition numbers >10^8 (*ill-conditioned*)
  - `Quality 1` in CPLEX
  - `Kappa 1` in GUROBI

Models with numerical issues can lead to undesirable results: slow performance, wrong answers or inconsistent behavior. Source: GUROBI

Rule of thumb for selecting an LP algorithm

- Simplex (or dual simplex) method can the best choice for moderate size (up to 100000 x 100000)
- Interior point method is usually the most efficient for huge and difficult problems
  - It is the most *numerically sensitive* algorithm. Numerical issues can cause crossover to stall
  - It can be *threaded* quite efficiently (compared to simplex)
- Difference in solution time can reach one order of magnitude

Variable attributes

Attributes	
VarName.lo	lower bound
VarName.up	upper bound
VarName.fx	fixes the variable to a constant
VarName.Range	range of the variable
VarName.l	initial value before and optimal value after
VarName.m	marginal value (reduced cost)
VarName.Scale	numerical scale factor
VarName.Prior	branching priority in a MIP model (∞ →not discrete)
VarName.SlackUp	slack from upper bound
VarName.SlackLo	slack from lower bound
VarName.Infeas	infeasibility out of bounds

Equation attributes

Attributes	
EquationName.lo	lower bound
EquationName.up	upper bound
EquationName.l	initial value before and optimal value after
EquationName.m	marginal value (dual variable or shadow price)
EquationName.Scale	numerical scaling factor

Model

```
model ModelName / EquationNames /
model ModelName / all /
model ModelName / all – EquationName1 + EquationName2 /
```

Model attributes

Attributes	
ModelName.ResLim	Resource limit
ModelName.SolveOpt	Replace/merge/clear in consecutive solves
ModelName.SolSlack	Show slack variables
ModelName.SolvePrint	0, 1, 2 (to remove the detailed solution from the .lst file)
ModelName.TryLinear	Try linear model first
ModelName.ModelStat	Model status
ModelName.SolveStat	Solve status
ModelName.NumEqu	Number of equations
ModelName.NumVar	Number of variables
ModelName.NumDVar	Number of discrete variables
ModelName.NumNz	Number of non zeros

Attributes	
ModelName.IterUsd	Number of iterations
ModelName.ResUsd	Resource used
ModelName.BRatio	Basis ratio controls the use of previous basis
ModelName.HoldFixed	Fix and eliminate variables
ModelName.IterLim	Iteration limit
ModelName.NodLim	Node limit
ModelName.OptCA	Absolute optimality tolerance
ModelName.OptCR	Relative optimality tolerance
ModelName.OptFile	Use of an option file
ModelName.PriorOpt	Use of priority

GAMS Call Options

GAMS Options	
Suppress	Suppress echo of the code listing
PW	Page width
PS	Page size
Charset	Allows international characters
U1..U10	User parameter

For example, InterfaceName, SolverSelection, SkipExcelInput, SkipExcelOutput,

```
u1="Excel_Interface_Name" u2=0 u3=0 u4=1 --NumberCores=4
```

GAMS to LaTeX

```
sets
   I origins      / VIGO, ALGECIRAS /
   J destinations / MADRID, BARCELONA, VALENCIA /

parameters
   pA(i) origin capacity
       / VIGO      350
         ALGECIRAS 700 /

   pB(j) destination demand
       / MADRID    400
         BARCELONA 450
         VALENCIA  150 /

table pC(i,j) per unit transportation cost
          MADRID BARCELONA VALENCIA
VIGO       0.06     0.12     0.09
ALGECIRAS  0.05     0.15     0.11

variables
   vX(i,j) units transported
   vCost   transportation cost

positive variable vX

equations
   eCost        transportation cost
   eCapacity(i) maximum capacity of each origin
   eDemand  (j) demand supply at destination ;

eCost        .. sum[(i,j), pC(i,j) * vX(i,j)] =e= vCost ;
eCapacity(i) .. sum[   j ,           vX(i,j)] =l= pA(i) ;
eDemand  (j) .. sum[ i   ,           vX(i,j)] =g= pB(j) ;

model mTransport / all /
solve mTransport using LP minimizing vCost
```

$$\min_x \sum_{ij} c_{ij} x_{ij}$$

$$\sum_j x_{ij} \le a_i \quad \forall i$$

$$\sum_i x_{ij} \ge b_j \quad \forall j$$

$$x_{ij} \ge 0$$

Generate the doc file

gams transport.gms **DocFile**=transport

Write the tex file

model2tex transport

Symbols

Sets

Name	Domains	Description
I	*	origins
J	*	destinations

Parameters

Name	Domains	Description
pA	I	origin capacity
pB	J	destination demand
pC	I, J	per unit transportation cost

Variables

Name	Domains	Description
vX	I, J	units transported
vCost		transportation cost

Equations

Name	Domains	Description
eCost		transportation cost
eCapacity	I	maximum capacity of each origin
eDemand	J	demand supply at destination

Equation Definitions

eCost

$$\sum_{I,J} (\mathrm{pC}_{I,J} \cdot \mathrm{vX}_{I,J}) = \mathrm{vCost}$$

eCapacity$_I$

$$\sum_J (\mathrm{vX}_{I,J}) \le \mathrm{pA}_I \qquad \forall I$$

eDemand$_J$

$$\sum (\mathrm{vX}_{I,J}) \ge \mathrm{pB}_J \qquad \forall J$$

$\mathrm{vX}_{I,J} \ge 0 \ \forall I, J$

Each tool has a specific purpose

- GDX (GAMS Data eXchange) utilities to interface with other applications
- Interfaces with other programs
  - Excel (gdx2xls, xls2gms)
  - Access (gdx2access, mdb2gms)
  - SQL (gdx2sqlite, sql2gms)
  - Matlab (gdxmrw)
  - R (gdxrrw)
- APIs
  - .Net
  - Java
  - Python
- Input/output data and simple graphs (Excel, Access)
- Advanced graphs (GNUPlot, Matlab)