Replaceable Solvers in GAMS

Arne Stolbjerg Drud
ARKI Consulting & Development A/S
Bagsvaerd
Denmark
adrud@arki.dk

Overview

- Introduction and Background
- Interfaces between GAMS and Solvers
- Replaceable Solvers
- Model and Solver Status and Quality Control
  - Unbounded Solutions
  - Some Initial Results
  - Definitions of Unboundedness
  - Examples of Unbounded Model that do not terminate “Unbounded”.
  - Results with an updated CONOPT3
- Summary and Conclusions

Introduction and Background

From "High Level Modeling Systems and Nonlinear Programming" by Brooke, Drud, and Meeraus, in Boggs et.al. (eds.), *Numerical Optimization 1984*, SIAM:

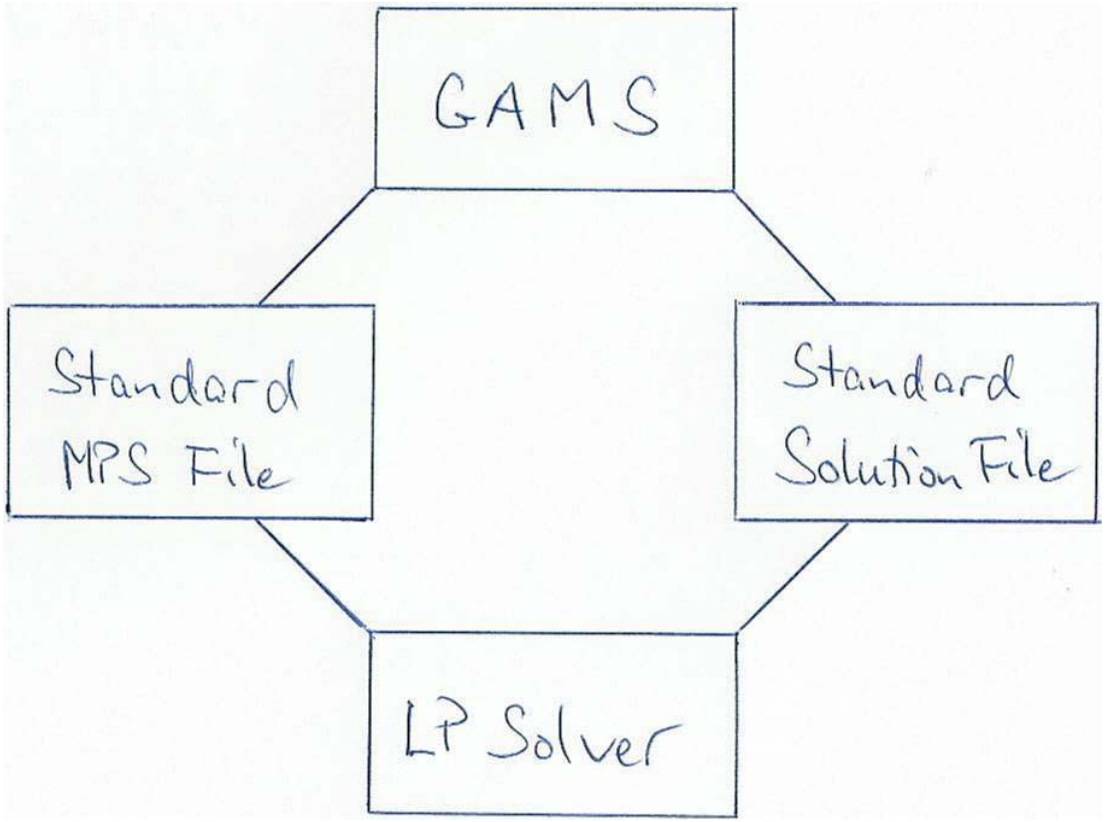
NLP is not used as much as LP because:

- NLP is conceptually harder to understand than LP
- Fewer tools for model formulation and report writing
- NLP solvers are less reliable than LP solvers, and any model reformulations or solver replacements are expensive.

The objective of GAMS has in this context since the very beginning been:

- Bring the NLP magic out in the open by clear algebra
- Provide better tools for model formulation and report writing
- Make it easy to experiment with alternative formulations and alternative NLP solvers.

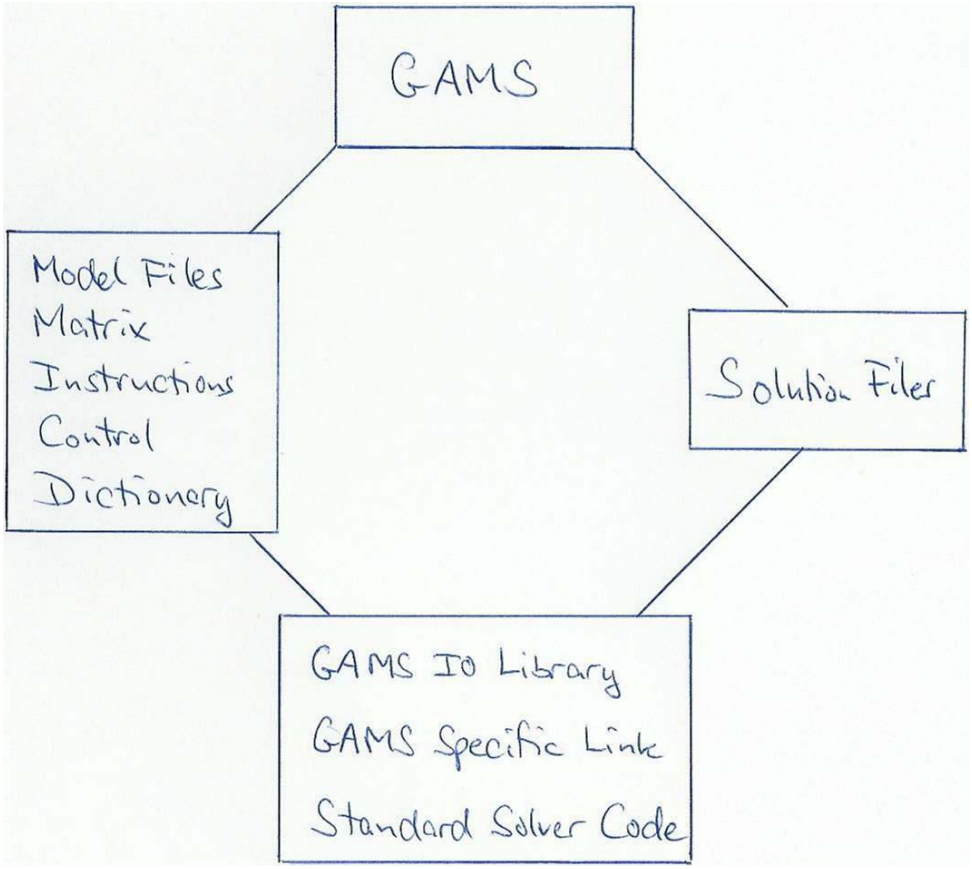
LP Interface (1984)



Differences between NLP Solvers

	GRG2	CONOPT	NPSOL	MINOS
Jacobian	Dense	Sparse	Dense	Sparse
Special Equation Order	No	No	Yes	Yes
Special Variable Order	No	No	No	Yes
Nonlinear elements	Ignored	Marked	Defined by row	Defined by Row and Col
Function routines	Obj. included	Obj. included	Obj. separate	Obj. separate
Linear Terms in Functions	Included	Excluded	Included	Excluded
Numerical Derivatives	Yes	Yes	No	Yes

The NLP Solver Interface



The NLP Solver Interface

- Replaceable NLP Solvers: Different GAMS/NLP solvers are created with different GAMS specific links combined with the standard NLP solver library.
- Content of Model Files must be Complete: All information ever needed by any solver must be present.
- The capabilities of the GAMS IO Library must at the same time satisfy the most demanding solvers and the dumbest solvers => flexible setup.
- GAMS IO Library must work on all machines, all operating systems, always, and it must NEVER crash => very defensive programming.
- Use of (slow) files instead of (modern) DLL interfaces is used to make errors reproducible and to protect GAMS from solver problems/crashes.

The NLP Solver Interface

Additions to the GAMS IO System over time follow the principle, that routines needed in more GAMS Specific Links are moved to the GAMS IO System.
Recent additions:

- Second order information for SQP and Interior Point solvers – is computed from existing information in the interface files using new methods.
- Interval information for preprocessing and global solvers.
- Names for more detailed messages during solution – based on the Dictionary file.
- A new CONVERT 'Solver' that can recreate GAMS models in algebraic form according to alternative conventions, e.g. BARON, AMPL, GAMS, LGO etc.

Replaceable Solvers

The **objective** with replaceable NLP solvers was to **increase reliability**: if one did not work, another could be tried without rewriting the model or model interfaces.

GAMS has actively encouraged users to have multiple NLP solvers with discounted NLP-packages. This has brought advantages to us all:

- Users have increased reliability
- GAMS has less support work
- If none of the NLP solvers work it is easier to convince a user that the model needs some rethinking
- Strong and weak sides of solvers can be identified and utilized
- Solver developers have less immediate support work and can specialize in special types of models without loosing market

Result: Today we have good solvers and we can concentrate on more than just getting an **"Optimal Solution". What about Quality?**

Replaceable Solvers – The Solution Files

GAMS has full control over the Model Files (from GAMS to Solvers).

GAMS has a set of definitions for the behavior of solutions, for example definition of the proper sign of dual variables.

However, GAMS cannot promise the same solution from different solvers:

- Most NLP solvers return a Local Optimum and there may be many Local Optima.
- There can be alternate optimal solutions, even when they a Globally Optimal.
- Solvers may not function correctly.

Replaceable Solvers – The Solution Files

A GAMS solver must return vectors of Primal and Dual solution values PLUS:

Model Status (a description of the numerical solution that has been returned):

- (Locally) Optimal
- (Locally) Infeasible
- Unbounded
- Intermediate Infeasible or Non-optimal
- Error / Failure

Solver Status (a description of how the algorithm behaved):

- Normal Completion
- Resource Interrupt (Iterations, CPU Time, User-Interrupt)
- Interrupt due to too many Function Evaluation Errors
- Cannot Proceed, intermediate Solution exist
- Error or Failure with No solution

Status Information and Advanced Usage

Advanced usage of GAMS, e.g. in decomposition, looping, branch & bound, and conditional model solves, rely on the **Model and Solver Status** information. The combinations of Model Status and Solver Status form a matrix where the entries can be classified as follows (only items relevant for NLP are included here):

	Normal	Resource Interrupt	Function Errors	Cannot Proceed	Error
Optimal	Good	Illegal	Illegal	Illegal	Illegal
Infeasible	Good	Illegal	Illegal	Illegal	Illegal
Unbounded	Good	Illegal	Illegal	Illegal	Illegal
Intermediate	Illegal	Acceptable	Acceptable	Acceptable	Illegal
Error	Illegal	Illegal	Illegal	Illegal	Bad

Status Information - Quality

GAMS internal Quality Control for solvers and solver interfaces cannot test for a unique solution and unique status information since it may not exist. Instead, a number of tests for **internal consistency** have been introduced:

Quality: A solver must **never** return any of the **red combinations**.

Quality: If one of the **green combinations** is returned it must also be **correct.** The tests for **local optimality** are implemented with the **GAMS/EXAMINER** solver, but the tests for **local infeasibility** and **unboundedness** are missing.

Conditional execution will usually be based on an assumption of **green** status information.

Status Information - Quality

The **blue combinations** (“**resource interrupt**”) indicate that the solver was not given enough time or iterations. **Solver developers** look at these cases and include relevant models into their performance tests. We try to solve models more quickly, thereby moving models from the **blue** to one of the **green** cells.

Conditional execution will usually continue the Solve when the status information is **blue**.

The **purple combinations** (“**cannot proceed**”) will usually indicate a solver problem in which case the relevant **Solver developer** need to look at them. However, **purple** cases can also be caused by non-smooth models (DNLP) being solved by a smooth solver (NLP) or by solution moving towards “**singularites**”.

Sensible conditional execution will usually not be possible when the status information is **purple**. A Continued Solve may sometimes cure the problem.

Status Information - Quality

The **yellow combinations** will usually indicate a modeling problem and the listing file should mark the offending equations. Some solvers may try to move around areas with undefined functions (domain violations), but the responsibility is not well defined.

Sensible conditional execution will usually not be possible.

Few consistency tests are possible for **blue**, **purple**, and **yellow** combinations: Essentially only “Intermediate Feasible” vs. “Intermediate Infeasible”.

Status Information – Solver Status Normal

In the following we will discuss the **green** cases.

We usually agree what a **Locally Optimal** solution means and the **EXAMINER** solver has implemented tests based on the Karush-Kuhn-Tucker conditions.

A **Locally Infeasible** solution can usually be translated into a **Locally Optimal** solution for some 'Phase 1 Objective' on a related model. However, the weights used in the objective function are often not available. GAMS does not have requirements for locally optimal solutions, in particular not for the dual solution, and we can therefore not implement any tests.

An **Unbounded** solution is less well defined (even for LP). We will discuss this case in the following.

Question: If we start from a Solution with Solver Status "Normal" using another NLP solver, will we always get the same solution back?

Unbounded Solutions
The Difference Between LP and NLP

An unbounded LP model has an unbounded ray in a direction with improving objective.

Example:

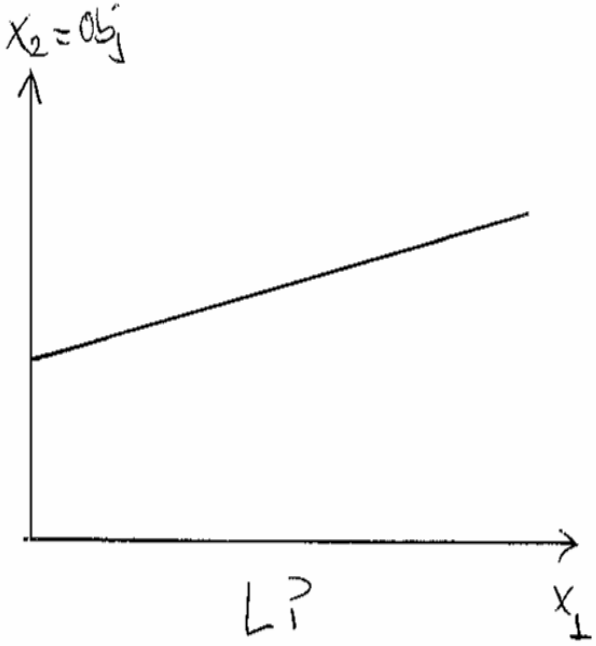
Max	x2
Subject to	x2 <= 0.5*x1
	x1 ≥ 0, x2 ≥ 0

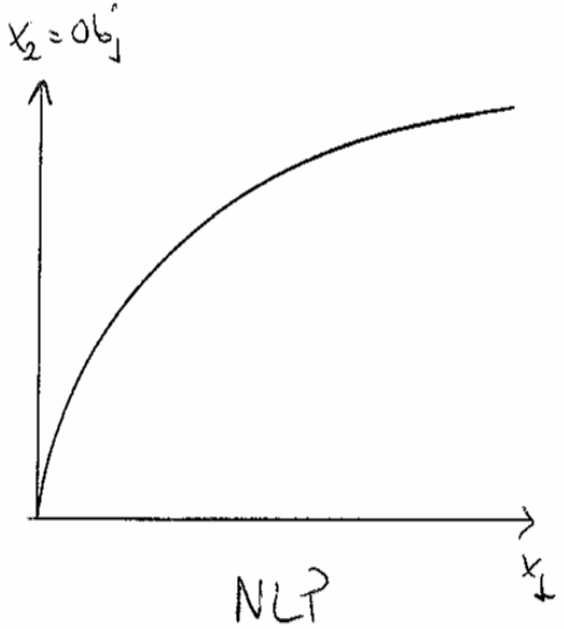
Unbounded Solutions
The Difference Between LP and NLP

An unbounded NLP model may not have an unbounded ray in a direction with improving objective.

Example:

Max	x2
Subject to	x2 <= sqrt(x1)
	$x1 \geq 0$, $x2 \geq 0$


$X_2 = Obj$
LP
X_1


$X_2 = Obj$
NLP
X_1

Unbounded Solutions
The Difference Between LP and NLP

An **LP model** can be proven to be unbounded with a pair of n-vectors: A feasible point, x, and an unbounded feasible ray, d, emerging from this point. The numerical work involved in defining x and d can all be done close to the origin, i.e. without actually moving towards infinity and without working with very large numbers.

How is an **NLP model** proven to be unbounded? For a general non-convex NLP we must go all the way towards infinity. On the way towards infinity, all kinds of numerical problems may pop up.

Some Initial Results

8 families of mathematically unbounded models with a total of 38 instances have been defined. These instances have been solved using MINOS, SNOPT, CONOPT2, and CONOPT3 under GAMS distribution 21.1 (from June 2003).

The classification of the 158 solves (based on default tolerances and options) were:

	Normal	Resource Interrupt	Function Errors	Cannot Proceed	Error
Optimal	**25**				
Infeasible	**2**				
Unbounded	**41**				
Intermediate		**31**	**8**	**45**	
Error					

With only 41 = 26% correct classifications we must conclude that detection of Unbounded Solutions is not a strong side of the selected solvers.

Some Initial Results (cont)

We have translated the same 38 instances into AMPL using GAMS/CONVERT and solved them using KNITRO, LOQO, FILTER, and LANCELOT via the NEOS Server and using default options and tolerances (version August 2003).

AMPL and these solvers use different classifications of the solution so we have performed a manual translation into the GAMS classification. Not all cases were clear: A solution declared optimal by a solver but with an objective value of 1.e20 is here classified as optimal. The classification of the 158 solves were:

	Normal	Resource Interrupt	Function Errors	Cannot Proceed	Error
Optimal	**41**				
Infeasible	**5**				
Unbounded	**15**				
Intermediate		**74**	**4**	**12**	
Error					**1**

Some Initial Results (cont)

The classification of the 38 solves for CONOPT3 (again based on default tolerances) were:

	Normal	Resource Interrupt	Function Errors	Cannot Proceed	Error
Optimal	**10**				
Infeasible	**0**				
Unbounded	**6**				
Intermediate		**8**	**2**	**12**	
Error					

Some Initial Results (cont)

Many of the 'Cannot Proceed' cases are caused by violation of a default upper limit on the size of Jacobian elements (around 1.e7), intended to prevent users from trying to solve very poorly scaled models.

Since unbounded models can have very large Jacobian elements, we have tried to increase this limit is to 1.e30. The results were only slightly better:

	Normal	Resource Interrupt	Function Errors	Cannot Proceed	Error
Optimal	**11**				
Infeasible	**0**				
Unbounded	**13**				
Intermediate		**7**	**2**	**5**	
Error					

Definitions of Unbounded NLPs

Definition used in CONOPT: An NLP model is unbounded if, given a fixed large M, there exist a feasible point, x with $|x| \geq M$, and a feasible tangent direction in x, d, pointing out of the M-box such that $d_{obj} > 0$.

This definition is natural for a feasible-path NLP solver (and many other types of NLP solvers):

Replace all 'Infinity' bounds with a large value (M) and solve the model. If any of the M-bounds become binding with a nonzero marginal then the model is considered unbounded. The test can be applied to the final solution or to intermediate solutions.

Note that the direction vector d cannot be returned to the user in a Modeling System like GAMS without hiding it, e.g. in the vector of marginals. Only part of the "Proof of Unboundesness", namely x, is made available.

Why are Unbounded Models not declared Unbounded?

Problem 1: The solver stops and declares the final solution **"Locally Optimal"**.

Example:

Max x_2

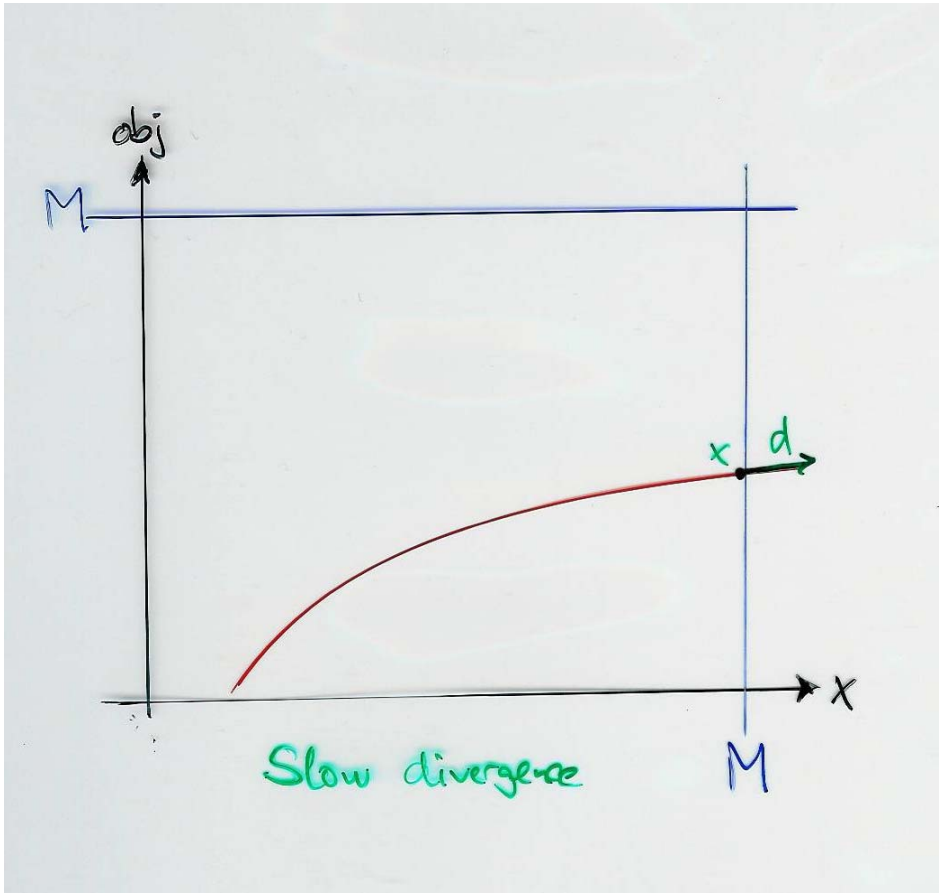
s.t. $x_2 = x_1**p, x_1 \geq 0$ with small positive p **- OR -**

s.t. $x_2 = \log(x_1), x_1 \geq 1$

When x_1 is large the derivative becomes smaller than the optimality tolerance and the solution is declared "Locally Optimal".

The only way to reduce the number of false "Locally Optimal" classifications is to reduce the optimality tolerance which seems unrealistic.

There is not much hope that we can "fix" this problem. Users must learn that Mathematical Unbounded and Numerical Unbounded is not the same.


obj
M
x
d
X
Slow divergence
M

Why are Unbounded Models not declared Unbounded?

Problem 2: The solver stops on **"Iteration Interrupt".**

The following model is clearly unbounded:

$$\begin{aligned} \text{Max} \quad & x_3 = x_1 + x_2 \\ \text{s.t.} \quad & x_2 = x_1^{**}1.5 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \end{aligned}$$

Change the constraint to the mathematically equivalent

$$x_2^{**}(2*p) = x_1^{**}(3*p)$$

for some integer values of p (1, 2, 3, and 4) and the model is still unbounded.

The constraints become increasingly nonlinear with larger p values. In addition, x_2 is implicitly defined and progress can be extremely slow.

Why are Unbounded Models not declared Unbounded?

Problem 3: The solver stops and declares the final solution **"Locally Infeasible"**.

This can be caused by very large intermediate values in the calculation of the constraints combined with small absolute tolerances. The previous example shows this situation.

Aggressive scaling seems to be the only solution to these problems.

Why are Unbounded Models not declared Unbounded?

Problem 4: The solver stops with **"Cannot Proceed"** or **"Terminated by Solver".**

Example 1: Max x_4

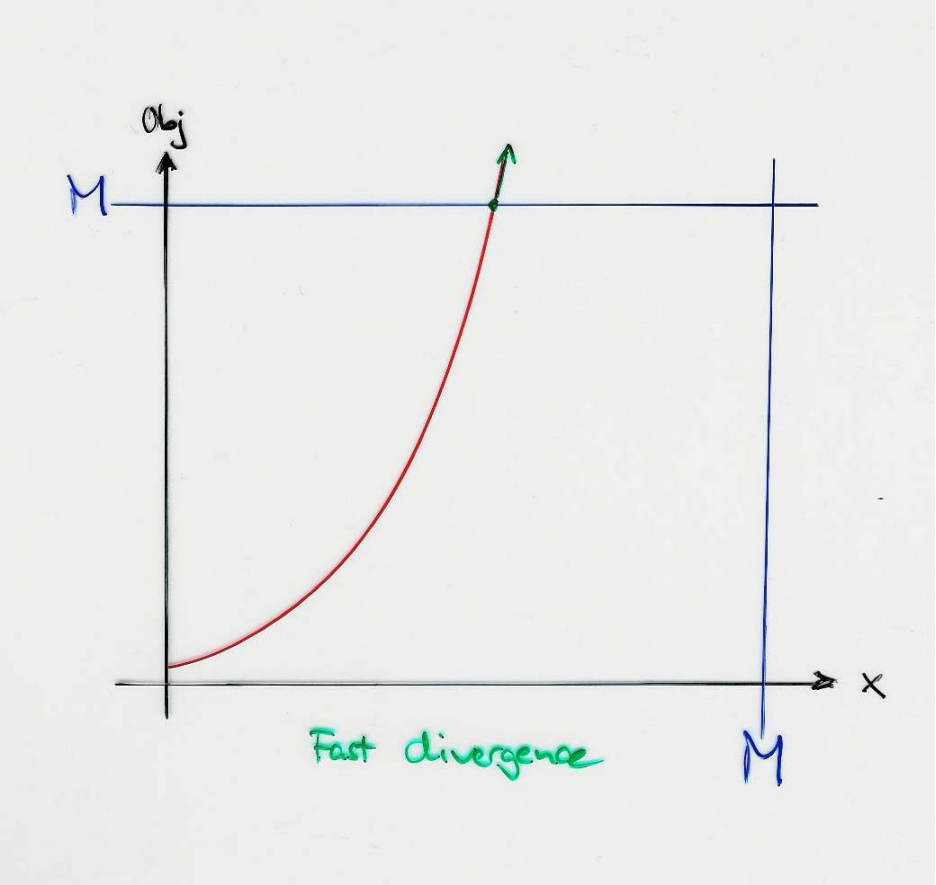
s.t. $x_4 = x_3 - 0.99*x_1 - 0.01*x_2$

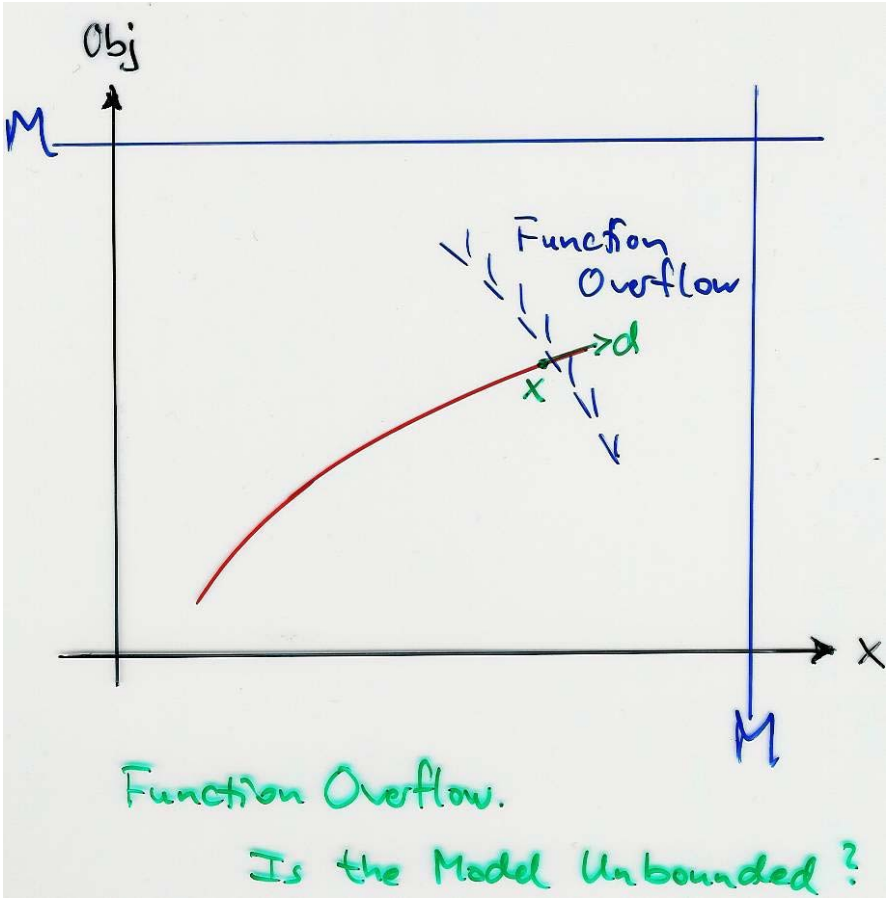
$x_3 = 1.01 * (0.5*x_1**Rho+0.5*x_2**Rho)**(1/Rho)$

The model represents a production function with efficiency > 1, unbounded inputs available at fixed prices, and unbounded demand at a fixed price.

For Rho = 5 CONOPT3 stops with (x_1,x_2,x_3,x_4) = (1.0, 1.1e13, 9.4e12, 9.3e12) and the message is **"Slow Progress".**

The problem is hidden in a message saying "**671 cases of x**c overflow**". A small increase in x_2 results in a function evaluation error return code from GAMS' evaluator instead of the numerical value of the nonlinear function. CONOPT3 cannot improve the objective even though gradients suggest that better solutions exist. But x is still far from 'Infinity'.


Obj
M
X
Fast divergence
M


Obj
M
Function
Overflow
d
x
X
M
Function Overflow.
Is the Model Unbounded ?

Why are Unbounded Models not declared Unbounded?

Problem 4: The solver stops with **"Cannot Proceed"** or **"Terminated by Solver".**

Example 2: max x_2
s.t. $x_2 = -\log(x_1)$

CONOPT3 terminates with the feasible solution $(x_1,x_2) = (7.3e\text{-}32, 71.688)$.

In this point there is a derivative of 1.4e31 which is above an absolute internal limit included to prevent numerical problems. CONOPT3 returns **"Terminated by Solver".**

Mathematically the model is unbounded, but x_2 can not exceed 1000 if we use ordinary floating point arithmetic. **So is the model really unbounded?**

Why are Unbounded Models not declared Unbounded?

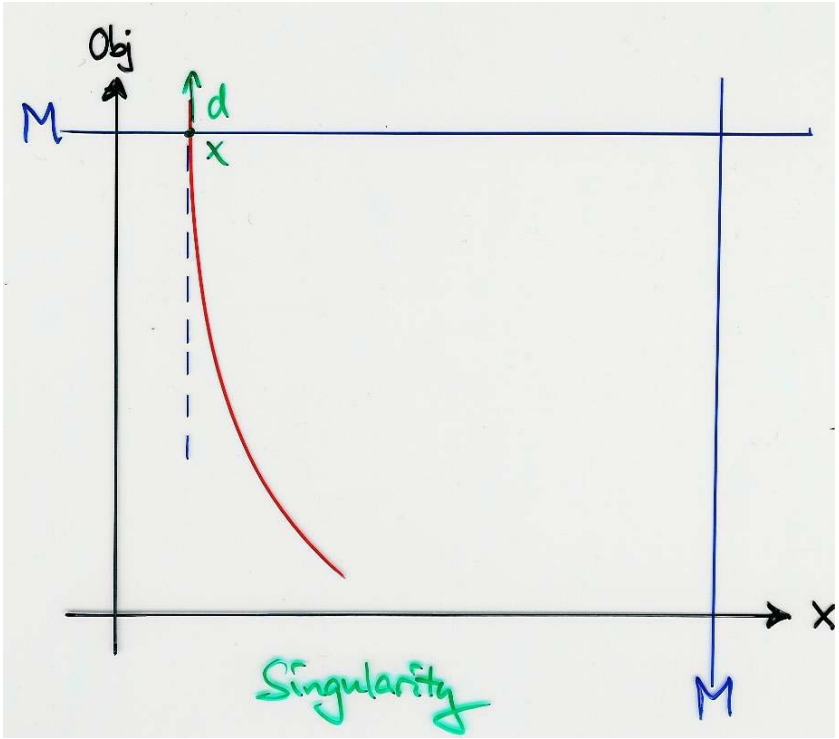
Problem 5: The solver stops on **"Function Evaluation Error Limit"** or something similar.

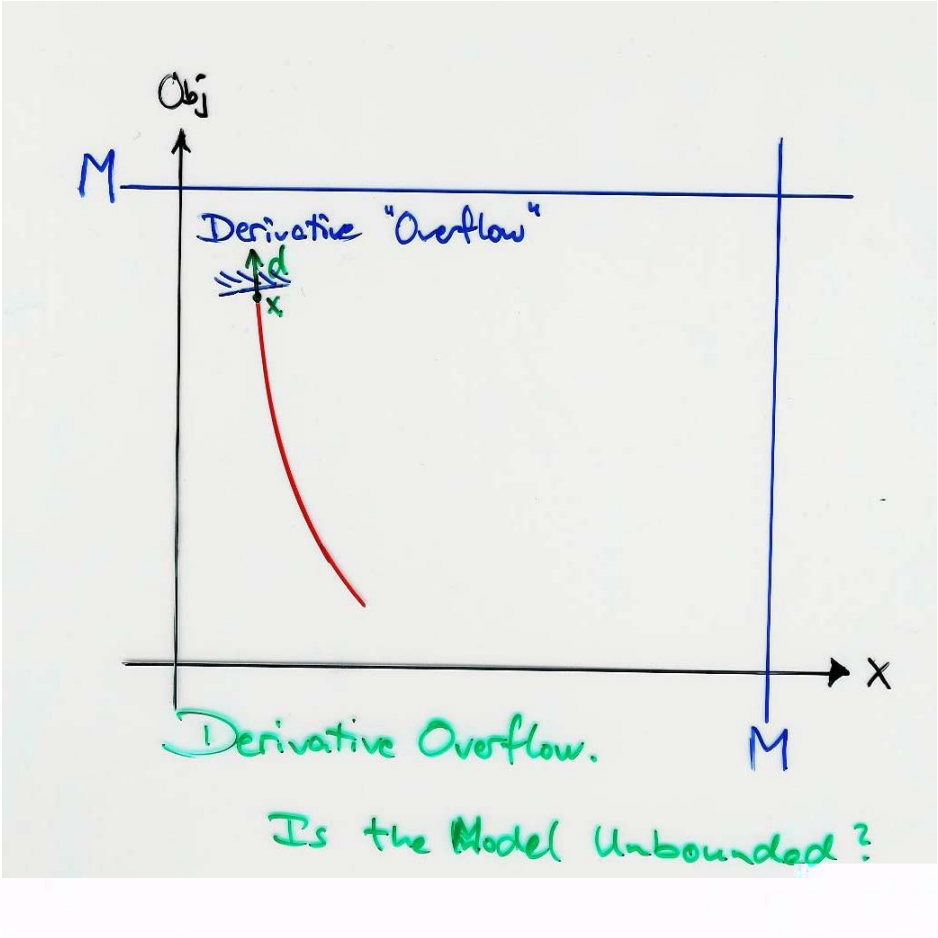
$$\begin{aligned} \max \quad & x_2 \\ \text{s.t.} \quad & x_2 = x_1^{**}(-0.5) \\ & x_1 \geq 0 \end{aligned}$$

The current distribution of CONOPT3 finds the intermediate feasible point (x_1,x_2) = (8.7e-19,1.07e+9).

Since x_1 is very close to the lower bound it is set to zero. When CONOPT3 tries to compute the derivative, it is not defined and it stops. The solution that is returned is not even feasible.

This is of course a bug that is being fixed, but it may not be an unusual one.


Obj
M
d
x
X
Singularity
M


Obj
M
Derivative "Overflow"
d
x
X
Derivative Overflow.
M
Is the Model Unbounded?

Results with an Updated CONOPT3

Based on **"Stress Tests"** using simple clearly unbounded models similar to the ones shown above we have made a number of changes in CONOPT3, mainly:

- Many Bugs have been identified and removed
- More aggressive scaling
- Adjustment of Filtering for small values

The classification of the 38 solves has changed from:

	Normal	Resource Interrupt	Function Errors	Cannot Proceed	Error
Optimal	**11**				
Infeasible	**0**				
Unbounded	**13**				
Intermediate		**7**	**2**	**5**	
Error					

to

	Normal	Resource Interrupt	Function Errors	Cannot Proceed	Error
Optimal	**2**				
Infeasible	**0**				
Unbounded	**25**				
Intermediate		**0**	**1**	**10**	
Error					

Summary and Conclusion

- Mainly due to the use of modeling systems like GAMS we have had a tremendous development in NLP over the last 20 years.
- Practical use of NLP has become common.
- NLP solvers have become much more reliable and much faster.
- The usual cases are well covered and we should now be ready to look at other termination states than "**Optimal**" and define what we mean and how and when solutions can be used.
- "**Unbounded Solution**" is an overlooked area in NLP and the current state is **BAD**.

Summary and Conclusion

- Definitions for bad cases need more work. Will other solvers give the same solution when restarted from an Infeasible or Unbounded solution?
- Do we need other classifications for Normal Termination than Optimal, Infeasible, and Unbounded? For example: “Boundary of Computable Region” or “Close to Singularity” (better short names are needed).
- Better Error Communication between NLP solvers and GAMS. “Function not defined” should distinguish between “Domain Error” and “Overflow”. We may also need a new function that can return Distance to Singularity, Distance to Discontinuity, and Distance to Overflow measured in a particular direction.
- Should the return information include the “Proof of unboundedness” (or an Unboundedness Certificate) which include a tangent direction, d?
- There is still a lot of work ahead!