PATHNLP

Contents

1 Introduction

This document describes the GAMS/PATHNLP solver for non-linear programs and the options unique to this solver.

PATHNLP solves an NLP by internally constructing the Karush-Kuhn-Tucker (KKT) system of first-order optimality conditions associated with the NLP and solving this system using the PATH solver for complementarity problems. The solution to the original NLP is extracted from the KKT solution and returned to GAMS. All of this takes place automatically - no special syntax or user reformulation is required.

Typically, PATHNLP works very well for convex models. It also has a comparative advantage on models whose solution via reduced gradient methods results in a large number of superbasic variables, since the PATH solver won't construct a dense reduced Hessian in the space of the superbasic variables as reduced gradient solvers do. For nonconvex models, however, PATHNLP is not as robust as the reduced gradient methods.

The theory relating NLP to their KKT systems is well-known: assuming differentiability without convexity, and assuming a constraint qualification holds, then a solution to the NLP must also be a solution to the KKT system. If we also assume convexity, then a solution to the KKT system is also a solution to the NLP - no further constraint qualification is required.

In case PATH fails to find a solution to the KKT system for the NLP, a phase I / phase II method is used in which the phase I objective is simply the feasibility error and the original objective is ignored. If a feasible point is found in phase I then phase II, an attempt to solve the KKT system for the NLP using the current feasible point, is entered.

PATHNLP is installed automatically with your GAMS system. Without a license, it will run in student or demonstration mode (i.e. it will solve small models only). If your GAMS license includes PATH, this size restriction is removed.

2 Usage

If you have installed the system and configured PATHNLP as the default NLP solver, all NLP models without a specific solver option will be solved with PATHNLP. If you installed another solver as the default, you can explicitly request that a particular model be solved using PATHNLP by inserting the statement

```
option NLP = pathnlp;
```

somewhere before the `solve` statement. Similar comments hold for the other model types (LP, RMINLP, QCP, etc.) PATHNLP can handle.

The standard GAMS model options `iterlim, reslim` and `optfile` can be used to control PATHNLP. A description of these options can be found in Chapter 1, "Basic Solver Usage". In general this is enough knowledge to solve your models. In some cases, however, you may want to use some of the PATHNLP options to gain further performance improvements or for other reasons. The rules for using an option file are described in Chapter 1, "Basic Solver Usage" The options used to control PATH can also be used to control PATHNLP. There are also some options unique to PATHNLP described below.

3   Options

The table that follows contains the options unique to PATHNLP. For details on the options PATHNLP shares with the other PATH links, see the chapter on the PATH solver.

Option	Description	Default
`allow_reform`	Many models have an objective variable and equation that can be substituted out of the model, e.g. `z =E= f(x);` If this option is set, PATHNLP will substitute out the objective variable and equation where possible.	yes
`output_memory`	If set, the output will include a report on the memory allocated for use by PATHNLP.	no
`output_time`	If set, the output will include a report on the time used use by PATHNLP.	no
`skip_kkt`	If set, PATHNLP will skip the initial attempt to solve the KKT system for the NLP and go directly into a phase I / phase II method that first attempts to get feasible and then attempts to solve the KKT system starting from the feasible point found in phase I.	no