The PATH solver for Complementarity Problems

Steven P. Dirkse *   Michael C. Ferris *

May 25, 1994

Abstract

This document describes the path following nonmonotone stabilization algorithm employed by the PATH solver for complementarity problems. Some implementation details are given as well, along with a description of the available options. For a more complete description, see [1].

1 The Algorithm

The PATH solver is an implementation of a stabilized Newton method for solving the Mixed Complementarity Problem, which is defined as follows:

Definition 1 (MCP) *Given a function $F : \Re^n \to \Re^n$ and bounds $\ell, u \in \overline{\Re}^n$,*

$$\begin{array}{ll} \textit{find} & z \in \Re^n, \;\; w, v \in \Re^n_+ \\ & F(z) = w - v \\ \textit{s.t.} & \ell \leq z \leq u \\ & (z-\ell)^\top w = 0 \\ & (u-z)^\top v = 0 \end{array}$$

The algorithm makes use of the path construction and searching techniques first explored by Ralph [4] and later developed by Dirkse and Ferris [1] . In the sequel, we shall assume that the function F is continuously differentiable on the box $B := [\ell, u]$.

In order to apply a damped Newton method, we must rewrite MCP as a zero-finding problem. This is accomplished by using the normal equations (NE) of Robinson [5], defined as

$$0 = F_B(x) := F(x_B) + (x - x_B),$$

where x_B is the Euclidean projection of x onto the box B. A vector x solves (NE) only if $z = x_B$ solves MCP.

The normal map F_B is a piecewise-smooth map, continuous on $\Re^n$, such that F_B is differentiable on each of the cells of the normal manifold defined by B.

The normal map F_B is approximated by the piecewise-linear normal map A_k defined by

$$A_k(x) := Mx_B + q + x - x_B \tag{1}$$

*Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706. This material is based on research supported by the National Science Foundation Grant CCR–9157632 and the Air Force Office of Scientific Research Grant F49620–94–1–0036

where

$$M := F'(x^k) \quad \text{and} \quad q := F({x^k}_B) - M{x^k}_B.$$

Construction of the Newton point x_N^k (i.e. the zero of A_k) is where the majority of the work in this algorithm is performed. We describe here a pivotal method for computing this point, given first by Ralph [4], which results in a piecewise linear path from the current point x^k to the Newton point.

If we define

$$\begin{aligned} z &:= x_B, \\ w &= (z - x)_+, \\ v &= (x - z)_+, \\ \text{and} \quad r &= -F_B(x^k), \end{aligned}$$

then (NE) corresponds to the following pivotal system:

$$\begin{bmatrix} M & -I & I & r \end{bmatrix} \begin{bmatrix} z \\ w \\ v \\ t \end{bmatrix} = -q, \tag{2}$$

$$\ell \le z \le u, \tag{3}$$

$$w, v \ge 0, \tag{4}$$

$$0 \le t \le 1 \tag{5}$$

where the parameter $t = 1$ when (z, w, v) is defined at x^k and $t = 0$ when (z, w, v) is defined at the Newton point x_N^k. In this system, $v - w = x - x_B$ at all times, where $x = z - w + v$, while the equality $x - x_B = A_k(x)$ holds when $t = 0$, so that x_N^k satisfies the normal equation in A_k. The pivotal method we employ seeks to force t from 1 down to 0. A sequence of pivot steps is performed, each of which reduces t. This forms a piecewise-linear path, parameterized by t, from x^k to x_N^k, where $p^k(t) = z(t) - w(t) + v(t)$.

A number of difficulties can arise while constructing the path. The initial basis may not be invertible, the value of t may not decrease monotonically on the path, and the Newton point $p^k(1)$ may never be reached, due either to a cycling of bases or to ray termination. These issues are addressed in Section 2.

The crucial property of the path p^k is that the norm of the approximation A_k decreases linearly in t on the path, i.e.

$$\left\| A_k(p^k(t)) \right\| = tA_k(x^k).$$

The path described above enables the application of damping techniques to acheive global convergence. In a typical pathsearch step, the path p^k is searched for a point $p^k(t)$ which satisfies

$$\left\| F_B(p^k(t)) \right\| \le (1 - \sigma(1-t)) \left\| F_B(p^k(1)) \right\|, \tag{6}$$

where $\sigma \in (0,1)$. If A_k is a "good" approximation to F_B at x^k, then (6) is satisfied for some $\bar{t} < 1$, since $\left\| A_k(p^k(t)) \right\|$ goes to zero linearly as t goes from 1 to 0. The new iterate x^{k+1} in the Newton process is defined to be $p^k(\bar{t})$. Given the appropriate hypotheses, it can be shown that the sequence $\left\| F_B(x^k) \right\|$ decreases linearly to 0 when $\bar{t}$ is chosen to satisfy (6).

Various non-monotone stabilization (NMS) schemes for Newton's method have been proposed, each seeking to improve efficiency by relaxing the requirement of monotone descent in F_B. The PATH solver implements a scheme of this type, modified to incorporate a pathsearch rather than a linesearch. The NMS scheme implemented makes use of a *watchdog* technique to reduce the number of pathsearches performed, thereby decreasing the number of function evaluations performed, and allows a non-monotonic decrease in the merit function associated with the points chosen as a result of these pathsearches. In the majority of cases, the watchdog technique elects to take a *d-step*. This d-step is acceptable if the point returned by the path generation procedure is suitably close to the current point. The measure of closeness, Δ, decreases as the algorithm progresses. In order to monitor these steps, the non-monotone descent criteria for the merit function are checked at least once every $\bar{n}$ iterations. The current merit function value is compared with a *reference value* $\mathcal{R}$, which is computed from previous function values. Steps in which these checks on the current merit function value occur are called *m-steps*. The points at which these criteria are checked *and satisfied* are called *check* points. An m-step is also taken when a d-step is unacceptable, that is, when it is too large. A watchdog step occurs when descent criteria are violated; when this occurs, the algorithm returns to the most recent check point, re-generates the path from the check point (if necessary), and searches the path for a point that satisfies the non-monotone descent criteria.

2 Implementation Issues

The PATH solver was originally implemented as a standalone solution system, and later modified to interface with a number of front ends, including the GAMS/MCP system for specifying complementarity problems. Regardless of which front end is used, the behavior of the solver can be influenced by setting various parameters and options in an options file. This options file must be in the current directory, and be named `path.opt`. Its structure is similar to the MINOS options file; each option is set using a keyword string and, most often, an option value, such as `basis_update_limit 50`. Only the first 3 characters of each keyword are significant. More than one option can be specified per line. The "*" character indicates that the remainder of the line should be ignored, and blank lines are permitted. The options available are given in Tables 1, 2, and 3 below. In these tables, the default values are indicated in parentheses or by the use of bold face type.

The bulk of the computation is done in the computation of the path to the Newton point. A number of problems can arise in this context. Firstly, given any iterate x^k, the columns corresponding to the basic variables in the pivotal system (2) may not form a basis. When this occurs, we can make an attempt at solving the system using a different (full-rank) matrix as the initial basis, if possible. This basis is found by choosing as many slack columns to be basic as possible. When solving an LCP, this slack basis (and the correct choice of initial point $x = -w$) corresponds exactly to the steps taken by Lemke's method, where the first pivot step taken effectively performs a ray start. The option to use this initial basis at all times (see Table 1) enables the PATH solver to mimic Josephy-Newton's method exactly. However, better results are achieved by letting the final basic variables of the previous subproblem remain basic; this provides a warm basis for the next subproblem, and often reduces the number of pivots required substantially.

The value of t is not guaranteed to decrease monotonically during the algorithm, so that $p^k(t)$ is not a true function of t. In practice, t will sometimes cycle through a repeating range of values or increase to ∞, after decreasing slightly from 1. When ray termination occurs or cycling leads to an iteration interrupt, the problem can be linearized at a point corresponding to the last computed basis or at a point corresponding to the basis with the best (lowest) value of t encountered. When **save_best_basis** is on, the latter behavior occurs.

The majority of work done by the PATH solver is done by the pivotal techniques used to compute the Newton point, and particularly by the basis routines used to do the pivotal algebra. These updating routines handle the bases encountered in a sparse manner, and permit efficient rank-one basis updates. The basis is refactored when the updates begin to take too long relative to a refactorization step, or when an upper limit is reached. A number of options exist to control the basis handling routines and the amount of memory they are allocated for factoring and updating the basis. A DEVEX-type routine (see [3]) is used to calculate the leaving variable in each pivot operation. The slack tolerance used in this calculation is one of several zero tolerances used by the PATH solver.

A number of options exist to control the nonmonotone stabilization techniques used, some of which set the parameters in the NMS acceptance criterion

$$\|F_B(p(t))\| \leq (1 - \sigma(1-t))\mathcal{R} \tag{7}$$

Table 1: Parameters and options

keywords	value	description
basis_elbowroom	c	Basis routines initially allocated $(1+c)\cdot sizeof(M)$ storage. (1.5)
basis_update_limit	k	At most k updates performed before the basis is refactored. (50)
basis_update_luroom	c	Basis routines are allocated $(1+c)\cdot sizeof(LU)$ (needed to store updates). (.75)
initial_lemke_path	y/n	The **first** subproblem will be solved "Lemke-style". (no)
lemke_path_allowed		Subproblems with a rank-deficient initial basis are solved "Lemke-style" - default
lemke_path_banned		**No** subproblems are solved "Lemke-style"
lemke_path_required		**All** subproblems are solved "Lemke-style"
major_iteration_limit	k	Maximum number of subproblems solved. (500)
major_logging_frequency	k	Status line is logged every k major iterations. (1)
minor_iteration_limit	k	Maximum number of pivots allowed per subproblem (1000)
minor_logging_frequency	k	Controls frequency of tprint and basis update logging (20)
save_best_basis	y/n	Basis with lowest t value is saved. (yes)
scale_initial_slacks	y/n	Scale initial values of w and v are using F_B. (yes)
tolerance_converg	c	Convergence is declared when $\|f_B\|_2 < c$. (1e-8)
tolerance_factor	c	A tolerance supplied to the factorization routines. For MINOS 5.4, it's the minimum ratio of pivot element to max element in column. (.1)
tolerance_pivot	c	Pivot elements less than c are treated as zero. (3.667e-11)
tolerance_slack	c	During pivoting, the bounds are relaxed by c in order to choose a leaving variable. (1e-6)
tolerance_small	c	The basis routines set "small" values to zero when factoring/updating. (1e-14)

Table 2: NMS parameters and options

keywords	value	description
nms	y/n	Use non-monotone stabilization techniques; if not, the point returned by the path generator is always accepted. (yes)
nms_memory_size	k	Number of previous merit function values stored. (5)
nms_mstep_freq	k	Number of d-steps allowed between m-steps. (5)
nms_pathsearch_tol	σ	Sets σ in (7) above. (.1)
nms_reference_init_factor	c	The reference value $\mathcal{R}$ is initialized to $c \cdot \left\| F_B(x^0) \right\|$. (20)
nms_reference_min	c	The reference value is constrained to be no less than c. (0)
nms_stepsize_init_factor	c	The initial step size limit is set to $c \cdot \left\| x^0 - x_N^0 \right\|$. (20)
nms_stepsize_init_min	c	The initial step size limit is constrained to be no less than c. (20)
nms_stepsize_shrink_factor	c	The step size limit is decreased by a factor of c at each d-step. (.9)

Finally, a number of verbosity flags exist to control the output of the solver. Perhaps the most important of these is **vflags_options y**, which causes a listing of all the modified options and parameter values to be sent to the log file. To print out all default values, use only this line in an options file.

Table 3: Options file verbosity flags

keywords	value	description
vfl_backtrace	y/**n**	for each backtrace performed, log the steps taken
vfl_initpoint	y/**n**	log z_0 and $f(z_0)$
vfl_leaves	y/**n**	for each pivot, log leaving and entering variables
vfl_mcp	y/**n**	for each major iteration, log M and q
vfl_nms	y/**n**	log actions of the non-monotone stabilization scheme
vfl_options	y/**n**	logs a list of all options values after reading options file
vfl_ratiotest	y/**n**	for each pivot, the vectors involved in the ratio test are logged
vfl_raybase	y/**n**	when ray termination occurs, log the base of the ray
vfl_silent	y/**n**	keeps some warning/error messages from being logged
vfl_soluprint	y/**n**	final problem solution is logged
vfl_sumprint	**y**/n	at solution, summary solver statistics are logged
vfl_tprint	y/**n**	at each pivot, the value of the parameter t is logged
vfl_updates	y/**n**	logs size and memory use statistics at each basis update

Always included in the PATH solver output are the problem size and density and an echo print of the options file. The log line printed (by default) at each major iteration consists of the iteration index, the number of pivot steps (minor iterations) required to solve the subproblem, and cumulative totals for function and gradient evaluations. Also printed is the t value at the endpoint of the path (with 0 corresponding to a Newton point), the 2- and inf-norms of the normal map F_B, and the GAMS equation which corresponds to the reported inf-norm value. Setting `major_logging_frequency 0` suppresses these log lines.

References

[1] S. P. Dirkse and M. C. Ferris. The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. Technical Report 1179, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1993. Submitted for publication.

[2] S. P. Dirkse and M. C. Ferris. MCPLIB: A collection of nonlinear mixed complementarity problems. Technical Report 1215, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1994.

[3] P. M. J. Harris. Pivot selection methods of the Devex LP code. *Mathematical Programming*, 5:1–28, 1973.

[4] D. Ralph. Global convergence of damped Newton's method for nonsmooth equations, via the path search. *Mathematics of Operations Research*, 1994. To appear.

[5] S. M. Robinson. Normal maps induced by linear transformations. *Mathematics of Operations Research*, 17(3):691–714, August 1992.