PENNON-AMPL User's Guide (Version 1.3)

Michal Kočvara Michael Stingl

www.penopt.com

August 29, 2003

The problem

We solve optimization problems with nonlinear objective subject to nonlinear inequalities and equalities as constraints:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \qquad g_i(x) \le 0, \qquad i=1,\dots,m_g \\ &\qquad\quad\;\; h_i(x) = 0, \qquad i=1,\dots,m_h\,. \end{aligned} \tag{NLP}$$

Here f, g_i and h_i are C^2 functions from $\mathbb{R}^n$ to $\mathbb{R}$.

The algorithm

To simplify the presentation of the algorithm, *we only consider inequality constraints*. For the treatment of the equality constraints, see [1].

The algorithm is based on a choice of penalty/barrier function $\varphi_g : \mathbb{R} \to \mathbb{R}$ that penalize the inequality constraints. This function satisfies a number of properties (see [1]) that guarantee that for any $p_i > 0,\ i = 1,\dots,m_g$, we have

$$g_i(x) \le 0 \iff p_i\varphi_g(g_i(x)/p_i) \le 0, \quad i=1,\dots,m\ .$$

This means that, for any $p_i > 0$, problem (NLP) has the same solution as the following "augmented" problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \qquad p_i\varphi_g(g_i(x)/p_i) \le 0, \qquad i=1,\dots,m_g\,. \end{aligned} \tag{NLP$_\phi$}$$

The Lagrangian of (NLP$_\phi$) can be viewed as a (generalized) augmented Lagrangian of (NLP):

$$F(x,u,p) = f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi_g(g_i(x)/p_i)\,; \tag{1}$$

here $u \in \mathbb{R}^{m_g}$ are Lagrangian multipliers associated with the inequality constraints.

The algorithm combines ideas of the (exterior) penalty and (interior) barrier methods with the Augmented Lagrangian method.

Algorithm 1 *Let x^1 and u^1 be given. Let $p_i^1 > 0$, $i = 1,\dots,m_g$. For $k = 1, 2, \dots$ repeat till a stopping criterium is reached:*

$$\begin{aligned} (i) \qquad & \text{Find } x^{k+1} \text{ such that } \|\nabla_x F(x^{k+1},u^k,p^k)\| \le K \\ (ii) \qquad & u_i^{k+1} = u_i^k \varphi_g'(g_i(x^{k+1})/p_i^k), \quad i=1,\,\dots,m_g \\ (iii) \qquad & p_i^{k+1} < p_i^k,\ i=1,\dots,m_g\,. \end{aligned}$$

The approximate unconstrained minimization in Step (i) is performed either by the Newton method with line-search or by one of two variants of the Trust Region method (for details, see [1]). The minimization is optionally stopped when either

$$\|\nabla_x F(x^{k+1}, u^k, p^k)\|_2 \leq \alpha$$

or

$$\|\nabla_x F(x^{k+1}, u^k, p^k)\|_2 \leq \alpha \cdot f(x^0)$$

or

$$\|\nabla_x F(x^{k+1}, u^k, p^k)\|_{H^{-1}} \leq \alpha \|\nabla_x F(x^k, u^k, p^k)\|_{H^{-1}}$$

with optional parameter α; by default, $\alpha = 10^{-1}$.

The multipliers calculated in Step (ii) are restricted in order to satisfy:

$$\mu < \frac{u_i^{k+1}}{u_i^k} < \frac{1}{\mu}$$

with some positive $\mu \leq 1$; by default, $\mu = 0.3$.

The update of the penalty parameter p in Step (iii) is performed in the following way: During the first three iterations we do not update the penalty vector p at all. After this kind of "warm start", the penalty vector is updated by some constant factor dependent on the initial penalty parameter π. The penalty update is stopped, if p_{eps} (by default 10^{-6}) is reached.

Algorithm 1 is stopped when both of the inequalities holds:

$$\frac{|f(x^k) - F(x^k, u^k, p)|}{1 + |f(x^k)|} < \epsilon\,, \qquad \frac{|f(x^k) - f(x^{k-1})|}{1 + |f(x^k)|} < \epsilon\,,$$

where ϵ is by default 10^{-7} (parameter `precision`).

Program call

PENNON-AMPL is called in the standard AMPL style, i.e., either by a sequence like

```
> model sample.mod;
> data sample.dat;
> options solver pennon;
> options pennon_options 'convex=1 outlev=2';  (for instance)
> solve;
```

within the AMPL environment or from the command line by

```
> pennon stub.nl 'convex=1 outlev=2'
```

Program options

The options are summarized in Table 1.

Recommendations

- Whenever you know that the problem is convex, use `convex=1`.
- When you have problems with convergence of the algorithm, try to
  - increase (decrease) `uinit`, e.g., `uinit=10000`.
  - swith to Trust Region algorithm by `ncmode=1`
  - decrease `alpha`, e.g., `alpha=1e-3`

Table 1: PENNON-AMPL options

option	meaning	default
`alpha`	stopping parameter α for the Newton/Trust region method in the inner loop	1.0E-1
`autoscale`	automatic scaling 0 . . . on 1 . . . off	0
`convex`	convex problem? 0 . . . generally nonconvex 1 . . . convex	0
`hessianmode`	check density of the Hessian 0. . . automatic 1 . . . dense	0
`ignoreinit`	ignore initial solutions 0. . . do not ignore 1 . . . do ignore	0
`maxit`	maximum number of outer iterations	100
`mu`	restriction factor μ of multiplier update	0.3
`ncmode`	nonconvex mode 0. . . Modified Newton 1. . . Trust region	0
`nwtiters`	maximum number of iterations in the inner loop (Newton or Trust region method)	100
`nwtstopcrit`	stopping criterium for the inner loop 0. . . $\Vert\nabla L(x^{k+1})\Vert_2 < \alpha$ 1. . . $\Vert\nabla L(x^{k+1})\Vert_2 < \alpha \cdot f_0$ 2. . . $\Vert\nabla L(x^{k+1})\Vert_{H^{-1}} < \alpha \cdot \Vert\nabla L(x^k)\Vert_{H^{-1}}$	2
`objno`	objective number in the AMPL .mod file	1
`outlev`	output level 0 . . . silent mode 1 . . . brief output 2 . . . full output	1
`penalty`	penalty function 0. . . logarithmic barrier + quadratic penalty 1 . . . reciprocal barrier	0
`penup`	penalty update	0.5
`peps`	minimal penalty	1.0E-6
`pinit`	initial penalty	1.0E0
`precision`	required final precision	1.0E-7
`timing`	timing destination 0 . . . no 1 . . . stdout 2 . . . stderr 3 . . . both	0
`uinit`	initial multiplier scaling factor	1.0
`umin`	minimal multiplier	1.0E-10
`version`	report PENNON version 0 . . . yes 1 . . . no	0
`wantsol`	solution report without -AMPL. Sum of 0 . . . do not write .sol file 1 . . . write .sol file 2 . . . print primal variable 4 . . . print dual variable 8 . . . do not print solution message	0

References

[1] M. Kočvara and M. Stingl. PENNON—a code for convex nonlinear and semidefinite programming. *Optimization Methods and Software*, 8(3):317–333, 2003.