Manual of ReSNA

— matlab software for mixed nonlinear second-order cone complementarity problems based on Regularized Smoothing Newton Algorithm —

Shunsuke Hayashi*

September 4, 2013

1 Introduction

ReSNA (Regularized Smoothing Newton Algorithm) is a matlab software for solving the following mixed nonlinear second-order cone complementarity problem (MNSOCCP):

$$\begin{array}{rl} \text{Find} & (x,y,p) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^l \\ \text{such that} & x \in \mathcal{K},\ y \in \mathcal{K},\ x^\top y = 0, \\ & y = F_1(x,p),\ F_2(x,p) = 0, \end{array} \tag{1.1}$$

where $F_1 : \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^n$ and $F_2 : \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^l$ are given continuously differentiable functions, and $\mathcal{K}$ is a Cartesian product of several second-order cones (SOCs), i.e.,

$$\mathcal{K} := \mathcal{K}^{n_1} \times \mathcal{K}^{n_2} \times \cdots \times \mathcal{K}^{n_m}$$

with $n_1 + n_2 + \cdots + n_m = n$ and

$$\mathcal{K}^{n_i} := \begin{cases} \left\{ z \in \mathbb{R} \;\middle|\; z \geq 0 \right\} & (n_i = 1) \\ \left\{ z \in \mathbb{R}^{n_i} \;\middle|\; z_1 \geq \sqrt{z_2^2 + \cdots + z_{n_i}^2} \right\} & (n_i \geq 2). \end{cases}$$

MNSOCCP (1.1) involves many kinds of problems as special cases. When $n_1 = n_2 = \cdots = n_m = 1$, MNSOCCP (1.1) reduces to the following mixed complementary problem:

$$\begin{array}{rl} \text{Find} & (x,p) \in \mathbb{R}^n \times \mathbb{R}^l \\ \text{such that} & x \geq 0,\ F_1(x,p) \geq 0,\ x^\top F_1(x,p) = 0,\ F_2(x,p) = 0. \end{array} \tag{1.2}$$

Also, the following nonlinear second-order cone program (NSOCP)

$$\begin{array}{rl} \text{Minimize} & \theta(z) \\ \text{subject to} & G(z) \in \mathcal{K},\ H(z) = 0 \end{array} \tag{1.3}$$

reduces to MNSOCCP (1.1) via the KKT formulation.

2 How to use ReSNA

`ReSNA.m` can be used as follows.

Usage 1: `[x,y,p] = ReSNA(FUNC,nabFUNC,K,el)`
Usage 2: `[x,y,p] = ReSNA(FUNC,nabFUNC,K,el,x0,y0,p0)`

*Graduate School of Information Sciences (GSIS), Tohoku University (s_hayashi@plan.civil.tohoku.ac.jp)

- `FUNC` — implies the function $F : \mathbb{R}^{n+l} \to \mathbb{R}^{n+l}$ such that

$$F(z) = \begin{pmatrix} F_1(x,p) \\ F_2(x,p) \end{pmatrix} \text{ with } z = \begin{pmatrix} x \\ p \end{pmatrix}.$$

  If function m-file `F.m` plays a role of function F, then put `F.m` in the same folder and let `FUNC = @F`. ("at mark" is required before the name of function m-file.)

- `nabFUNC` — implies $\nabla F : \mathbb{R}^{n+l} \to \mathbb{R}^{(n+l)\times(n+l)}$, i.e., the transposed Jacobian of function F. More precisely,

$$\nabla F(z) = \begin{pmatrix} \nabla_x F_1(x,p) & \nabla_x F_2(x,p) \\ \nabla_p F_1(x,p) & \nabla_p F_2(x,p) \end{pmatrix} \text{ with } z = \begin{pmatrix} x \\ p \end{pmatrix}.$$

  If function m-file `nabF.m` plays a role of function ∇F, then put `nabF.m` in the same folder and let `nabFUNC = @nabF`. If you do not have the closed form of $\nabla F(z)$, let `nabFUNC = []`. In this case, $\nabla F(z)$ is approximated by means of the finite difference method.

- `K` — implies the Cartesian structure of $\mathcal{K}$ in MNSOCCP (1.1). `K` should be given as the row or column vector whose component corresponds to the dimension of each second-order cones. For example, when $\mathcal{K} = \mathcal{K}^3 \times \mathcal{K}^1 \times \mathcal{K}^2$, let `K = [3,1,2]`. When $\mathcal{K} = \mathbb{R}^{10}_+$, let `K = ones(1,10)`. When $\mathcal{K} = \emptyset$, let `K = []` or `K = 0`, whereby ReSNA solves the vector equation $F(p) = 0$.

- `el` — implies the value of l, i.e., the dimension of p or $F_2(x,p)$ in MNSOCCP (1.1). `el` should be given as a nonnegative integer. If p and $F_2(x,p)$ are absent (non-mixed case), let `el = 0`.

- `x0` — implies the initial point $x^{(0)}$ for the regularized smoothing Newton algorithm (Algorithm 4.1 given later). `x0` should be given as a column vector whose length is equal to `sum(K)`. If you omit `x0` or let `x0 = []`, then ReSNA chooses a random vector from $[-1,1]^n$ automatically.

- `y0` — implies the initial point $y^{(0)}$, which can be omitted similarly to `x0`.

- `p0` — implies the initial point $p^{(0)}$ for the regularized smoothing Newton algorithm. `p0` should be given as a column vector whose length is equal to `el`. If you omit `p0` or let `p0 = []`, then ReSNA chooses a random vector from $[-1,1]^l$ automatically.

Parameters in `ReSNA.m`

- `PROGRESS` — decides whether or not ReSNA displays the detailed progress of the iteration. The default value is `'Y'`.

- `tole` — is used for the termination criterion in Step 1. When $\|H_{\mathrm{NR}}(w^{(k)})\| \leq$ `tole`, the algorithm terminates normally and the obtained output is guaranteed to be the solution of MNSOCCP (1.1). The default value is `1e-8`.

- `tole_diff` — is used for approximating the Jacobian matrix by means of the finite difference method. The default value is `1e-8`.

- `eta, eta_bar, rho, sigma, kappa, kappa_bar, kappa_hat` — are the parameters indicated in Algorithm 4.1. Some default values are assigned automatically.

3 Basic idea of ReSNA

In this section, we summarize the rough idea of ReSNA. For the detailed mathematical background, see [2].

For $x = (x^1, \ldots, x^m) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}$ and $y = (y^1, \ldots, y^m) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}$, define function $\Phi_{\text{NR}} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ (called *natural residual*) by

$$\begin{aligned} \Phi_{\text{NR}}(x, y) &:= \begin{pmatrix} \varphi_{\text{NR}}(x^1, y^1) \\ \vdots \\ \varphi_{\text{NR}}(x^m, y^m) \end{pmatrix}, \\ \varphi_{\text{NR}}(x^i, y^i) &:= x^i - P_{\mathcal{K}^{n_i}}(x^i - y^i), \end{aligned}$$

where $P_{\mathcal{K}^{n_i}}(x^i - y^i)$ denotes the Euclidean projection of $x^i - y^i$ onto $\mathcal{K}^{n_i}$. Then, it follows that

$$\Phi_{\text{NR}}(x, y) = 0 \iff x \in \mathcal{K},\ y \in \mathcal{K},\ x^T y = 0.$$

Therefore, letting $H_{\text{NR}} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^{2n+l}$ be defined by

$$H_{\text{NR}}(x, y, p) := \begin{pmatrix} \Phi_{\text{NR}}(x, y) \\ F_1(x, p) - y \\ F_2(x, p) \end{pmatrix},$$

MNSOCCP (1.1) is reformulated as the following vector equation equivalently:

$$H_{\text{NR}}(x, y, p) = 0. \tag{3.1}$$

Since MNSOCCP (1.1) is equivalent to (3.1), we have only to solve (3.1) instead of MNSOCCP (1.1). However, function Φ_{NR} is nondifferentiable, and hence the Newton based method cannot be applied directly. Moreover, the level set of $\|H_{\text{NR}}(x, y, p)\|$ may often be unbounded even when F_1 and F_2 have nice properties. To overcome those difficulties, we introduce the smoothing and the regularization methods.

Smoothing method

Let

$$\begin{aligned} \Phi_\mu(x, y) &:= \begin{pmatrix} \varphi_\mu(x^1, y^1) \\ \vdots \\ \varphi_\mu(x^m, y^m) \end{pmatrix}, \\ \varphi_\mu(x^i, y^i) &:= x^i - P_\mu(x^i - y^i), \\ P_\mu(z) &:= \mu\hat{g}(\lambda_1/\mu)u^{\{1\}} + \mu\hat{g}(\lambda_2/\mu)u^{\{2\}}, \\ \hat{g}(\alpha) &:= \frac{1}{2}\big(\sqrt{\alpha^2 + 4} + \alpha\big), \end{aligned}$$

where λ_1 and λ_2 are the spectral values of z, and $u^{\{1\}}$ and $u^{\{2\}}$ are the spectral vectors of z. (See [1, 2].) Then, Φ_μ satisfies the following properties:

- Φ_μ is continuously differentiable for any fixed $\mu > 0$.
- $\lim_{\mu \searrow 0} \Phi_\mu(x, y) = \Phi_{\text{NR}}(x, y)$ for any fixed $(x, y) \in \mathbb{R}^n \times \mathbb{R}^n$.

Hence, we use Φ_μ instead of Φ_{NR} with letting $\mu \searrow 0$. This is the basic idea of smoothing method.

Regularization method

Let the functions $F_{1,\varepsilon} : \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^n$ and $F_{2,\varepsilon} : \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^l$ be defined by

$$\begin{aligned} F_{1,\varepsilon}(x,p) &:= F_1(x,p) + \varepsilon x, \\ F_{2,\varepsilon}(x,p) &:= F_2(x,p) + \varepsilon p, \end{aligned}$$

respectively, with a positive parameter ε. In general, functions $F_{1,\varepsilon}$ and $F_{2,\varepsilon}$ have better properties than F_1 and F_2 from the viewpoint of global convergence. For example, if $F = \binom{F_1}{F_2}$ is monotone, then $\binom{F_{1,\varepsilon}}{F_{2,\varepsilon}}$ is strongly monotone for any $\varepsilon > 0$.

Regularized smoothing Newton method

Define functions $H_{\mu,\varepsilon} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^{2n+l}$ by

$$H_{\mu,\varepsilon}(x,y,p) := \begin{pmatrix} \Phi_\mu(x,y) \\ F_{1,\varepsilon}(x,p) - y \\ F_{2,\varepsilon}(x,p) \end{pmatrix}. \tag{3.2}$$

Then, we solve the vector equation $H_{\mu,\varepsilon}(x,y,p) = 0$ by Newton's method with letting $(\mu,\varepsilon) \searrow (0,0)$.

4 Algorithm

4.1 Explicit expression of Jacobian and additional functions

Remark 4.1 *From the definition of $H_{\mu,\varepsilon}$, Φ_μ, $\hat{g}$, etc., $\nabla H_{\mu,\varepsilon}(x,y,p) \in \mathbb{R}^{(2n+l)\times(2n+l)}$ can be calculated as*

$$\nabla H_{\mu,\varepsilon}(x,y,p) = \begin{pmatrix} \operatorname{diag}\{I - \nabla P_\mu(x^i - y^i)\}_{i=1}^m & \nabla_x F_1(x,p) + \varepsilon I & \nabla_x F_2(x,p) \\ \operatorname{diag}\{\nabla P_\mu(x^i - y^i)\}_{i=1}^m & -I & 0 \\ 0 & \nabla_p F_1(x,p) & \nabla_p F_2(x,p) + \varepsilon I \end{pmatrix}, \tag{4.1}$$

where $\nabla P_\mu(z)$ is written as

$$\nabla P_\mu(z) = \begin{cases} \hat{g}'(z_1/\mu) I & \text{if } z_2 = 0, \\ \begin{pmatrix} b_\mu & \dfrac{c_\mu z_2^T}{\|z_2\|} \\ \dfrac{c_\mu z_2}{\|z_2\|} & a_\mu I + (b_\mu - a_\mu)\dfrac{z_2 z_2^T}{\|z_2\|^2} \end{pmatrix} & \text{if } z_2 \neq 0, \end{cases} \tag{4.2}$$

with

$$\begin{cases} a_\mu = \dfrac{\hat{g}(\lambda_2/\mu) - \hat{g}(\lambda_1/\mu)}{\lambda_2/\mu - \lambda_1/\mu}, \\ b_\mu = \dfrac{1}{2}(\hat{g}'(\lambda_2/\mu) + \hat{g}'(\lambda_1/\mu)), \\ c_\mu = \dfrac{1}{2}(\hat{g}'(\lambda_2/\mu) - \hat{g}'(\lambda_1/\mu)). \end{cases} \tag{4.3}$$

Definition 4.1

(a) *Let $\tilde{\lambda} : \mathbb{R}^n \to [0, +\infty)$ be defined by*

$$\tilde{\lambda}(z) := \begin{cases} \min\limits_{i \in \mathcal{I}(z)} |\lambda_i(z)| & (\mathcal{I}(z) \neq \emptyset) \\ 0 & (\mathcal{I}(z) = \emptyset), \end{cases} \tag{4.4}$$

where $\lambda_i(z)$ $(i = 1, 2)$ are the spectral values of z, and $\mathcal{I}(z) \subseteq \{1, 2\}$ is the index set defined by $\mathcal{I}(z) := \{i \mid \lambda_i(z) \neq 0\}$.

(b) *Let* $\overline{\mu} : \mathbb{R}^n \times \mathbb{R}^n \to [0, +\infty]$ *be defined by*

$$\overline{\mu}(\alpha, \delta) := \begin{cases} +\infty & (\delta \geq 1/2 \ \text{ or } \ \alpha = 0) \\ \dfrac{1}{2}|\alpha|\sqrt{\delta} & (\delta < 1/2 \ \text{ and } \ \alpha \neq 0)\,. \end{cases}$$

Proposition 4.1 *Let* $\hat{g}$ *be defined as in Section 3. Let* γ_μ *and* γ_0^+ *be defined as in* [2]*, respectively. Then, for any* $\alpha \in \mathbb{R}$*,* $\delta > 0$ *and* $\mu \in (0, \overline{\mu}(\alpha, \delta))$*, we have*

$$|\gamma'_\mu(\alpha) - \gamma_0^+(\alpha)| < \delta. \tag{4.5}$$

4.2 Main algorithm

For convenience, we denote

$$w := \begin{pmatrix} x \\ y \\ p \end{pmatrix}, \quad w^{(k)} := \begin{pmatrix} x^{(k)} \\ y^{(k)} \\ p^{(k)} \end{pmatrix}.$$

Algorithm 4.1 *Choose* η*,* $\rho \in (0, 1)$*,* $\overline{\eta} \in (0, \eta]$*,* $\sigma \in (0, 1/2)$*,* $\kappa > 0$ *and* $\hat{\kappa} > 0$*.*

Step 0 *Choose* $w^{(0)} \in \mathbb{R}^{2n+l}$ *and* $\beta_0 \in (0, \infty)$*. Let* $\mu_0 := \|H_{\text{NR}}(w^{(0)})\|$ *and* $\varepsilon_0 := \|H_{\text{NR}}(w^{(0)})\|$*. Set* $k := 0$*.*

Step 1 *Terminate if* $\|H_{\text{NR}}(w^{(k)})\| = 0$*.*

Step 2

Step 2.0 *Set* $v^{(0)} := w^{(k)} \in \mathbb{R}^{2n+l}$ *and* $j := 0$*.*

Step 2.1 *Find a vector* $\hat{d}^{(j)} \in \mathbb{R}^{2n+l}$ *such that*

$$H_{\mu_k, \varepsilon_k}(v^{(j)}) + \nabla H_{\mu_k, \varepsilon_k}(v^{(j)})^T \hat{d}^{(j)} = 0.$$

Step 2.2 *If* $\|H_{\mu_k, \varepsilon_k}(v^{(j)} + \hat{d}^{(j)})\| \leq \beta_k$*, then let* $w^{(k+1)} := v^{(j)} + \hat{d}^{(j)}$ *and go to* Step 3*. Otherwise, go to* Step 2.3*.*

Step 2.3 *Find the smallest nonnegative integer* m *such that*

$$\Psi_{\mu_k, \varepsilon_k}(v^{(j)} + \rho^m \hat{d}^{(j)}) \leq (1 - 2\sigma\rho^m)\Psi_{\mu_k, \varepsilon_k}(v^{(j)}).$$

Let $m_j := m$*,* $\tau_j := \rho^{m_j}$ *and* $v^{(j+1)} := v^{(j)} + \tau_j \hat{d}^{(j)}$*.*

Step 2.4 *If*

$$\|H_{\mu_k, \varepsilon_k}(v^{(j+1)})\| \leq \beta_k, \tag{4.6}$$

then let $w^{(k+1)} := v^{(j+1)}$ *and go to* Step 3*. Otherwise, set* $j := j + 1$ *and go back to* Step 2.1*.*

Step 3 *Update the parameters as follows*:

$$\mu_{k+1} := \min\left\{\kappa\|H_{\text{NR}}(w^{(k+1)})\|^2,\, \mu_0\overline{\eta}^{k+1},\, \overline{\mu}\Big(\tilde{\lambda}(x^{(k+1)} - y^{(k+1)}), \hat{\kappa}\|H_{\text{NR}}(w^{(k+1)})\|\Big)\right\},$$

$$\varepsilon_{k+1} := \min\left\{\kappa\|H_{\text{NR}}(w^{(k+1)})\|^2, \varepsilon_0\overline{\eta}^{k+1}\right\},$$

$$\beta_{k+1} := \beta_0\eta^{k+1}.$$

Set $k := k + 1$*. Go back to* Step 1*.*

In Step 3, $\tilde{\lambda}$ is the function given by (4.4), and $\overline{\mu}(\alpha, \delta)$ is determined so that $|\gamma'_\mu(\alpha) - \gamma_0^+(\alpha)| < \delta$ for any $\mu \in (0, \overline{\mu}(\alpha, \delta))$. In the inner iterations Steps 2.0 – 2.4, a damped Newton method seeks a point $w^{(k+1)}$ such that $\|H_{\mu_k, \varepsilon_k}(w^{(k+1)})\| \leq \beta_k$. Note that we have $\beta_k \to 0$ as $k \to 0$. Step 3 specifies the updating rule of the parameters, where $\{\beta_k\}$, $\{\mu_k\}$ and $\{\varepsilon_k\}$ converge to 0 since $0 < \overline{\eta} \leq \eta < 1$. Algorithm 4.1 is well-defined in the sense that Steps 2.0 – 2.4 find $v^{(j+1)}$ satisfying (4.6) in a finite number of iterations for each k.

4.3 Convergence properties

In [2], the authors proved that Algorithm 4.1 is globally and quadratically convergent when $l = 0$ (i.e., p and F_2 are absent.) and F_1 is monotone.

References

[1] M. Fukushima, Z.-Q. Luo, and P. Tseng, *Smoothing functions for second-order cone complementarity problems*, SIAM Journal on Optimization, 12 (2001), pp. 436–460.

[2] S. Hayashi, N. Yamashita, and M. Fukushima, *A combined smoothing and regularization method for monotone second-order cone complementarity problems*, SIAM Journal on Optimization, 15 (2005), pp. 593–615.