User guide for the **GNE** package

Christophe Dutang

August 7, 2012

As usual, the **GNE** package is loaded via the `library` function. In the following, we assume that the line below has been called

```
> library(GNE)
```

1 Introduction

Definition 1 (GNEP) *We define the generalized Nash equilibrium problem GNEP(N, θ_i, X_i) as the solutions $x^\star$ of the N sub-problems*

$$\forall i = 1, \dots, N, x_i^\star \ \ solves \ \min_{y_i} \ \ \theta_i(y_i, x_{-i}^\star) \ \ such\ that \ \ x_i^\star \in X_i(x_{-i}^\star),$$

where $X_i(x_{-i})$ is the action space of player i given others player actions x_{-i}.

If we have parametrized action space $X_i(x_{-i}) = \{y_i, g_i(y_i, x_{-i}) \leq 0\}$, we denote the GNEP by GNEP(N, θ_i, g_i).

We denote by $X(x)$ the action set $X(x) = X_1(x_{-1}) \times \cdots \times X_N(x_{-N})$. For standard NE, this set does not depend on x.

The following example seems very basic, but in fact it has particular features, one of them is to have four solutions, i.e. four GNEs. Let $N = 2$. The objective functions are defined as

$$\theta_1(x) = (x_1 - 2)^2(x_2 - 4)^4 \ \text{ and } \ \theta_2(x) = (x_2 - 3)^2(x_1)^4,$$

for $x \in \mathbb{R}^2$, while the constraint functions are given by

$$g_1(x) = x_1 + x_2 - 1 \leq 0 \ \text{ and } \ g_2(x) = 2x_1 + x_2 - 2 \leq 0.$$

Objective functions can be rewritten as $\theta_i(x) = (x_i - c_i)^2(x_{-i}d_i)^4$, with $c = (2, 3)$ and $d = (4, 0)$. First-order derivatives are

$$\nabla_j \theta_i(x) = 2(x_i - c_i)(x_{-i}d_i)^4\delta_{ij} + 4(x_i - c_i)^2(x_{-i}d_i)^3(1 - \delta_{ij}),$$

and

$$\nabla_j g_1(x) = 1 \ \text{ and } \ \nabla_j g_2(x) = 2\delta_{j1} + \delta_{j2}.$$

Second-order derivatives are

$$\nabla_k\nabla_j\theta_i(x) = 2(x_{-i}d_i)^4\delta_{ij}\delta_{ik} + 8(x_i - c_i)(x_{-i}d_i)^3\delta_{ij}(1-\delta_{ik})$$
$$+8(x_i - c_i)(x_{-i}d_i)^3(1-\delta_{ij})\delta_{ik} + 12(x_i - c_i)^2(x_{-i}d_i)^2(1-\delta_{ij})(1-\delta_{ik}),$$

and

$$\nabla_k\nabla_j g_1(x) = \nabla_k\nabla_j g_2(x) = 0.$$

2 GNEP as a nonsmooth equation

2.1 Notation and definitions

From Facchinei et al. (2009), assuming differentiability and a constraint qualification hold, the first-order necessary conditions of player i's subproblem state there exists a Lagrangian multiplier $\lambda^i \in \mathbb{R}^{m_i}$ such that

$$\nabla_{x_i}\theta_i(x^\star) + \sum_{1\le j\le m_i} \lambda_j^{i\star}\nabla_{x_i} g_j^i(x^\star) = 0 \qquad (\in \mathbb{R}^{n_i}).$$
$$0 \le \lambda^{i\star},\ -g^i(x^\star) \ge 0,\ g^i(x^\star)^T\lambda^{i\star} = 0 \qquad (\in \mathbb{R}^{m_i}).$$

Regrouping the N subproblems, we get the following system.

Definition 2 (eKKT) *For the N optimization subproblems for the functions $\theta_i : \mathbb{R}^n \mapsto \mathbb{R}$, with constraints $g_i : \mathbb{R}^n \mapsto \mathbb{R}^{m_i}$, the KKT conditions can be regrouped such that there exists $\lambda \in \mathbb{R}^m$ and*

$$\tilde{L}(x,\lambda) = 0 \quad \textit{and} \quad 0 \le \lambda \perp -G(x) \le 0,$$

where L and G are given by

$$\tilde{L}(x,\lambda) = \begin{pmatrix} \nabla_{x_1}\theta_1(x) + Jac\,g^1(x)^T\lambda^1 \\ \vdots \\ \nabla_{x_N}\theta_N(x) + Jac\,g^N(x)^T\lambda^N \end{pmatrix} \in \mathbb{R}^n \quad \textit{and} \quad G(x) = \begin{pmatrix} g^1(x) \\ \vdots \\ g^N(x) \end{pmatrix} \in \mathbb{R}^m,$$

with $Jac\,g_i(x)^T\lambda_i = \sum_{1\le j\le m_i}\lambda_j^i\nabla_{x_i}g_j^i(x)$. The extended KKT system is denoted by eKKT(N, θ_i, g_i).

Using complementarity function $\phi(a,b)$ (e.g. $\min(a,b)$), we get the following nonsmooth equation

$$\Phi(z) = \begin{pmatrix} \tilde{L}(x,\lambda) \\ \phi_\cdot(-G(x),\lambda) \end{pmatrix} = 0,$$

where $\phi_\cdot$ is the component-wise version of the function ϕ and $\tilde{L}$ is the Lagrangian function of the extended system. The generalized Jacobian is given in Appendix A.1.

2.2 A classic example

Returning to our example, we define the Φ as

$$\Phi(x) = \begin{pmatrix} 2(x_1-2)(x_2-4)^4 + \lambda_1 \\ 2(x_2-3)(x_1)^4 + \lambda_2 \\ \phi(\lambda_1, 1-x_1-x_2) \\ \phi(\lambda_2, 2-2x_1-x_2) \end{pmatrix},$$

where ϕ denotes a complementarity function. In R, we use

```
> myarg <- list(C=c(2, 3), D=c(4,0))
> dimx <- c(1, 1)
> #Gr_x_j O_i(x)
> grobj <- function(x, i, j, arg)
+ {
+         dij <- 1*(i == j)
+         other <- ifelse(i == 1, 2, 1)
+         res <- 2*(x[i] - arg$C[i])*(x[other] - arg$D[i])^4*dij
+         res + 4*(x[i] - arg$C[i])^2*(x[other] - arg$D[i])^3*(1-dij)
+ }
> dimlam <- c(1, 1)
> #g_i(x)
> g <- function(x, i)
+         ifelse(i == 1, sum(x[1:2]) - 1, 2*x[1]+x[2]-2)
> #Gr_x_j g_i(x)
> grg <- function(x, i, j)
+         ifelse(i == 1, 1, 1 + 1*(i == j))
```

Note that the triple dot arguments ... is used to pass arguments to the complementarity function.

Elements of the generalized Jacobian of Φ have the following form

$$\partial\Phi(x) = \left\{ \begin{pmatrix} 2(x_2-4)^4 & 8(x_1-2)(x_2-4)^3 & 1 & 0 \\ 8(x_2-3)(x_1)^3 & 2(x_1)^4 & 0 & 1 \\ -\phi'_b(\lambda_1, 1-x_1-x_2) & -\phi'_b(\lambda_1, 1-x_1-x_2) & \phi'_a(\lambda_1, 1-x_1-x_2) & 0 \\ -2\phi'_b(\lambda_2, 2-2x_1-x_2) & -\phi'_b(\lambda_2, 2-2x_1-x_2) & 0 & \phi'_a(\lambda_2, 2-2x_1-x_2) \end{pmatrix} \right\},$$

where ϕ'_a and ϕ'_b denote elements of the generalized gradient of the complementarity function. The corresponding R code is

```
> #Gr_x_k Gr_x_j O_i(x)
> heobj <- function(x, i, j, k, arg)
+ {
+         dij <- 1*(i == j)
+         dik <- 1*(i == k)
+         other <- ifelse(i == 1, 2, 1)
```

```
+         res <- 2*(x[other] - arg$D[i])^4*dij*dik
+         res <- res + 8*(x[i] - arg$C[i])*(x[other] - arg$D[i])^3*dij*(1-dik)
+         res <- res + 8*(x[i] - arg$C[i])*(x[other] - arg$D[i])^3*(1-dij)*dik
+         res + 12*(x[i] - arg$C[i])^2*(x[other] - arg$D[i])^2*(1-dij)*(1-dik)
+ }
> #Gr_x_k Gr_x_j g_i(x)
> heg <- function(x, i, j, k) 0
```

2.2.1 Usage example

Therefore, to compute a generalized Nash equilibrium, we use

```
> z0 <- rexp(sum(dimx)+sum(dimlam))
> GNE.nseq(z0, dimx, dimlam, grobj=grobj, myarg, heobj=heobj, myarg,
+         constr=g, grconstr=grg, heconstr=heg,
+         compl=phiFB, gcompla=GrAphiFB, gcomplb=GrBphiFB, method="Newton",
+         control=list(trace=0))
```

```
GNE: 1.472266 -0.4722277 422.8954 16.06543
with optimal norm 0.9645926
after  100 iterations with exit code 4 .
Output message: Iteration limit exceeded
Function/grad/hessian calls: 687 100
Optimal (vector) value: 0.6738681 -0.4966524 3.873489e-05 0.4792463
```

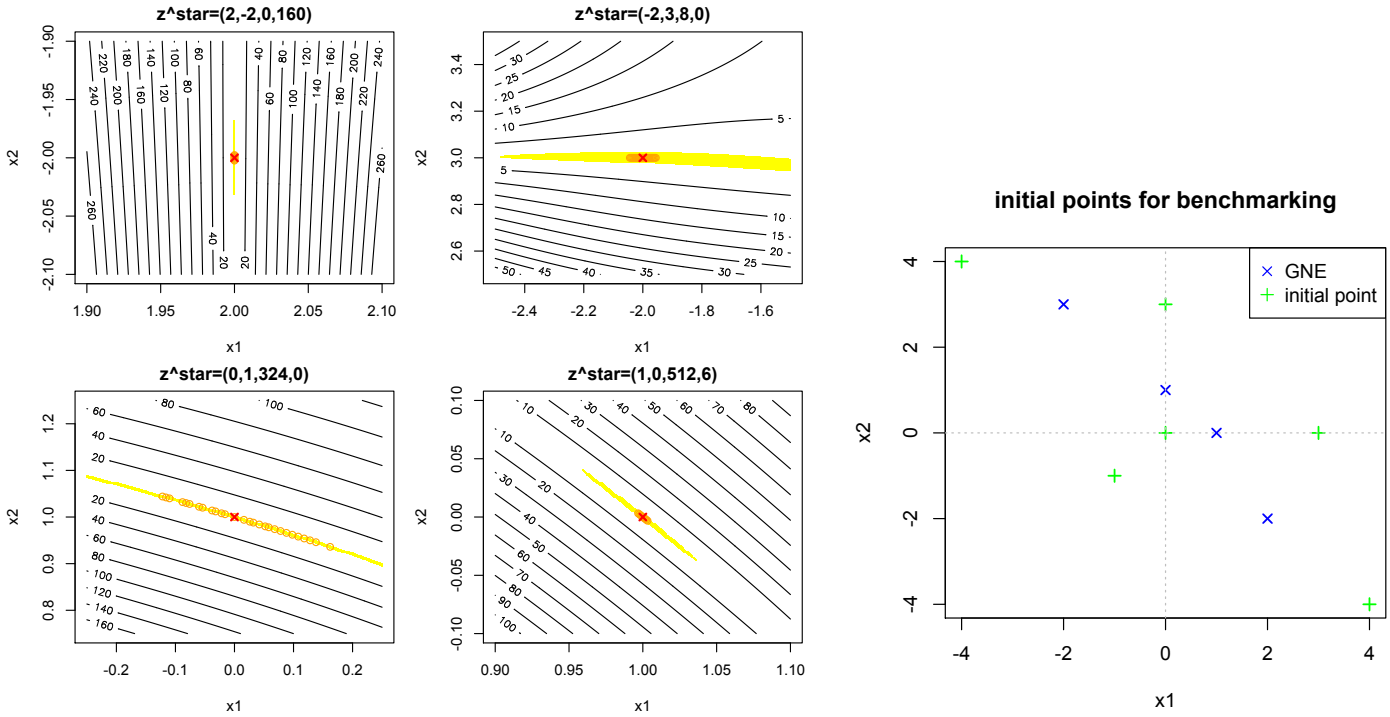
Recalling that the true GNEs are

```
> #list of true GNEs
> trueGNE <- rbind(c(2, -2, 0, 5*2^5),
+         c(-2, 3, 8, 0),
+         c(0, 1, 4*3^4, 0),
+         c(1, 0, 2^9, 6))
> colnames(trueGNE) <- c("x1", "x2", "lam1", "lam2")
> rownames(trueGNE) <- 1:4
> print(trueGNE)
```

```
  x1 x2 lam1 lam2
1  2 -2    0  160
2 -2  3    8    0
3  0  1  324    0
4  1  0  512    6
```

2.2.2 Localization of the GNEs

On figure 1a, we draw contour plots of the function $\frac{1}{2}||\Phi(z)||^2$ with respect to x_1 and x_2, given λ_1 and λ_2. The second figure 1b just plots the initial points and the 6 GNEs.



(a) The 4 GNEs

(b) The 6 initial points

Figure 1: Contour plots of the norm of Φ

2.3 Benchmark of the complementarity functions and the computation methods

Using the following function, we compare all the different methods with different initial points and different complementarity functions. We consider the following complementarity functions.

- $\phi_{Min}(a,b) = \min(a,b)$,
- $\phi_{FB}(a,b) = \sqrt{a^2+b^2} - (a+b)$,
- $\phi_{Man}(a,b) = f(|a-b|) - f(a) - f(b)$ and $f(t) = t^3$,
- $\phi_{LT}(a,b) = (a^q + b^q)^{\frac{1}{q}} - (a+b)$ and $q = 4$,
- $\phi_{KK}(a,b) = (\sqrt{(a-b)^2 + 2\lambda ab} - (a+b))/(2-\lambda)$ and $\lambda = 3/2$.

2.3.1 Initial point $z_0 = (4, -4, 1, 1)$

We work on the initial point $z_0 = (4, -4, 1, 1)$, close the GNE $(2, -2, 0, 160)$. Clearly, we observe the Mangasarian complementarity function ϕ_{Man} does not converge except in the pure Newton method, for which the sequence converges to $(-2, 3, 8, 0)$ quite far from the initial point. So the "Man" sequence converged by a chance! For ϕ_{Min} function, when it converges, the GNEs found are $(2, -2, 0, 160)$ or $(1, 0, 512, 6)$. ϕ_{FB} and ϕ_{KK} associated sequences converge mostly to $(2, -2, 0, 160)$. In terms of function/Jacobian calls, ϕ_{FB} is significantly better when used with the Newton scheme.

	$\phi_{Min}(a,b) = \min(a,b)$							$\phi_{FB}(a,b) = \sqrt{a^2+b^2} - (a+b)$						
	fctcall	jaccall	x_1	x_2	λ_1	λ_2	$\|\|\Phi(z)\|\|$	fctcall	jaccall	x_1	x_2	λ_1	λ_2	$\|\|\Phi(z)\|\|$
Newton - pure	5	5	1	0	512	6	0	6	6	2	-2	0	160	0
Newton - geom. LS	343	67	1	0	512	6	0	6	6	2	-2	0	160	0
Newton - quad. LS	292	100					2	6	6	2	-2	0	160	0
Newton - Powell TR	64	57	1	0	512	6	0	12	6	2	-2	0	160	0
Newton - Dbl. TR	63	58	1	0	512	6	0	12	6	2	-2	0	160	0
Broyden - pure	100	1					164	100	1					188
Broyden - geom. LS	403	6	1	0	512	6	0	1079	26					2
Broyden - quad. LS	291	6					1	467	3					1
Broyden - Powell TR	22	2	2	-2	0	160	0	114	2					1
Broyden - Dbl. TR	20	2	2	-2	0	160	0	115	2					1
	fctcall	jaccall	x_1	x_2	λ_1	λ_2	$\|\|\Phi(z)\|\|$	fctcall	jaccall	x_1	x_2	λ_1	λ_2	$\|\|\Phi(z)\|\|$
Newton - pure	113	113	-2	3	8	0	0	48	48	0	1	325	0	0
Newton - geom. LS	203	25					33	727	100					2
Newton - quad. LS	91	27					37	85	39	2	-2	0	160	0
Newton - Powell TR	75	67					3	152	100	0	1	309	0	0
Newton - Dbl. TR	62	53					3	147	100	0	1	304	0	0
Broyden - pure	200	1					506	49	1	1	0	512	6	0
Broyden - geom. LS	167	6					82	29	3	2	-2	0	160	0
Broyden - quad. LS	86	5					78	20	3	2	-2	0	160	0
Broyden - Powell TR	215	14					3	28	2	2	-2	0	160	0
Broyden - Dbl. TR	246	15					3	29	2	2	-2	0	160	0
	$\phi_{Man}(a,b) = f(\|a-b\|) - f(a) - f(b)$ and $f(t) = t^3$							$\phi_{KK}(a,b) = (\sqrt{(a-b)^2 + 2\lambda ab} - (a+b))/(2-\lambda)$ and $\lambda = 3/2$						

Table 1: With initial point $z_0 = (4, -4, 1, 1)$ close to $(2, -2, 0, 160)$

2.3.2 Initial point $z_0 = (-4, 4, 1, 1)$

We work on the initial point $z_0 = (-4, 4, 1, 1)$, close the GNE $(-2, 3, 8, 0)$. Again, we observe the Mangasarian complementarity function ϕ_{Man} does not converge. All other sequences converge the closest GNE $(-2, 3, 8, 0)$. ϕ_{Min} sequence with Newton scheme is particularly good, then comes ϕ_{FB} and finally ϕ_{KK}.

2.3.3 Initial point $z_0 = (3, 0, 1, 1)$

We work on the initial point $z_0 = (3, 0, 1, 1)$ close to the GNE $(1, 0, 512, 6)$. As always, the "Man" sequence converges by chance with the pure Newton method to a GNE $(-2, 3, 8, 0)$. Otherwise the other sequences, namely "Min", "FB" and "KK" converges to the expected GNE. As the previous subsection, Broyden updates of the Jacobian is less performant than the true Jacobian (i.e. Newton scheme). The convergence speed order is preserved.

2.3.4 Initial point $z_0 = (0, 3, 1, 1)$

We work on the initial point $z_0 = (0, 3, 1, 1)$ close to the GNE $(0, 1, 324, 0)$. As always, the "Man" sequence converges by chance with the pure Newton method to a GNE $(-2, 3, 8, 0)$. Others sequences have difficulty to converge the closest GNE. Local methods (i.e. pure) find the GNE $(0, 1, 324, 0)$, while global version converges to $(1, 0, 512, 6)$. It is logical any method will have difficulty to choose between these two GNEs, because they are close.

2.3.5 Initial point $z_0 = (-1, -1, 1, 1)$

We work on the initial point $z_0 = (-1, -1, 1, 1)$ equidistant to the GNEs $(0, 1, 324, 0)$ and $(1, 0, 512, 6)$. Despite being closer to these GNEs, the pure Newton version of the "Man" sequence converges unconditionally to the GNE $(-2, 3, 8, 0)$. All other sequences converges to the GNE $(0, 1, 324, 0)$ except for the Broyden version of the "KK" sequence, converging to the farthest GNEs. In terms of function calls, the Newton line search version of the "Min" sequence is the best, followed by the Newton trust region version of the "FB" sequence.

2.3.6 Initial point $z_0 = (0, 0, 1, 1)$

We work on the initial point $z_0 = (0, 0, 1, 1)$ equidistant to the GNEs $(0, 1, 324, 0)$ and $(1, 0, 512, 6)$. Both the "Man" and the "Min" sequences do not converge. The "Min" sequence diverges because the Jacobian at the initial point is exactly singular. Indeed, we have

```
> z0 <- c(0, 0, 1, 1)
> jacSSR(z0, dimx, dimlam, heobj=heobj, myarg, constr=g, grconstr=grg,
+         heconstr=heg, gcompla=GrAphiMin, gcomplb=GrBphiMin)
```

```
     [,1] [,2] [,3] [,4]
[1,]  512 1024    1    0
[2,]    0    0    0    2
[3,]   -1   -1    1    0
[4,]    0    0    0    1
```

For the "FB" and "KK" sequences, we do not have this problem.

```
> jacSSR(z0, dimx, dimlam, heobj=heobj, myarg, constr=g, grconstr=grg,
+         heconstr=heg, gcompla=GrAphiFB, gcomplb=GrBphiFB)
```

```
            [,1]         [,2]       [,3]       [,4]
[1,] 512.0000000 1024.0000000  1.0000000  0.0000000
[2,]   0.0000000    0.0000000  0.0000000  2.0000000
```

```
[3,]   0.2928932    0.2928932 -0.2928932  0.0000000
[4,]   0.1055728    0.2111456  0.0000000 -0.5527864
```

```
> jacSSR(z0, dimx, dimlam, heobj=heobj, myarg, constr=g, grconstr=grg,
+         heconstr=heg, gcompla=GrAphiKK, gcomplb=GrBphiKK, argcompl=3/2)
```

```
            [,1]         [,2]       [,3]       [,4]
[1,] 512.0000000 1024.0000000  1.0000000  0.0000000
[2,]   0.0000000    0.0000000  0.0000000  2.0000000
[3,]   0.2679492    0.2679492 -0.2679492  0.0000000
[4,]   0.1101776    0.2203553  0.0000000 -0.4881421
```

So the sequence converge to a GNE, either $(0, 1, 324, 0)$ or $(-2, 3, 8, 0)$. Again the "KK" sequence converges faster.

2.3.7 Conclusions

In conclusion to this analysis with respect to initial point, the computation method and the complementarity function, we observe the strong difference in terms of convergence, firstly and in terms of convergence speed. Clearly the choice of the complementarity function is crucial, the Luo-Tseng and the Mangasarian are particularly inadequate in our example. Regarding the remaining three complementarity functions (the minimum, the Fisher-Burmeister and the Kanzow-Kleinmichel functions) generally converge irrespectively of the computation method. However, the "KK" sequences are particularly efficient and most of the time the Newton trust region method is the best in terms of function/Jacobian calls.

2.4 Special case of shared constraints with common multipliers

Let $h : \mathbb{R}^n \mapsto \mathbb{R}^{m_l}$ be a constraint function shared by all players. The total constraint function and the Lagrange multiplier for the ith player is

$$\tilde{g}^i(x) = \begin{pmatrix} g^i(x) \\ h(x) \end{pmatrix} \quad \text{and} \quad \tilde{\lambda}^i = \begin{pmatrix} \lambda^i \\ \mu \end{pmatrix},$$

where $\mu \in \mathbb{R}^l$. This could fall within the previous framework, if we have not required the bottom part of $\tilde{\lambda}^i$ to be common among all players. The Lagrangian function of the ith player is given by

$$L^i(x, \lambda^i, \mu) = O_i(x) + \sum_{k=1}^{m_i} g_k^i(x)\lambda_k^i + \sum_{p=1}^{l} h_p(x)\mu_p.$$

Definition 3 (eKKTc) *For the N optimization subproblems for the functions $\theta_i : \mathbb{R}^n \mapsto \mathbb{R}$, with constraints $g_i : \mathbb{R}^n \mapsto \mathbb{R}^{m_i}$ and shared constraint $h : \mathbb{R}^n \mapsto \mathbb{R}^l$, the KKT conditions can be regrouped such that there exists $\lambda \in \mathbb{R}^m$ and*

$$\bar{L}(x, \lambda, \mu) = 0 \quad \textit{and} \quad 0 \leq \lambda, 0 \leq \mu \perp -g(x) \leq 0,$$

where L and G are given by

$$\bar{L}(x,\lambda,\mu) = \begin{pmatrix} \nabla_{x_1} L^1(x,\lambda^1,\mu) \\ \vdots \\ \nabla_{x_I} L^I(x,\lambda^I,\mu) \end{pmatrix} \in \mathbb{R}^n \quad \textit{and} \quad g(x) = \begin{pmatrix} g^1(x) \\ \vdots \\ g^N(x) \\ h(x) \end{pmatrix} \in \mathbb{R}^m.$$

The extended KKT system is denoted by eKKTc(N, θ_i, g_i, h).

The generalized Jacobian is given in Appendix A.2.

2.5 Constrained-equation reformulation of the KKT system

This subsection aims to present methods specific to solve constrained (nonlinear) equations, first proposed by Dreves et al. (2011) in the GNEP context. The root function $H : \mathbb{R}^n \times \mathbb{R}^{2m} \mapsto \mathbb{R}^n \times \mathbb{R}^{2m}$ is defined as

$$H(x,\lambda,w) = \begin{pmatrix} \tilde{L}(x,\lambda) \\ g(x) + w \\ \lambda \circ w \end{pmatrix},$$

where the dimensions n, m correspond to the GNEP notation ($\lambda = (\lambda^1, \ldots, \lambda^N)$) and $(a, \bar{\sigma})$ is given by $((0_n, \mathbb{1}_m), 1)$. The potential function is given by

$$p\,(u) = \zeta \log\left(||x||_2^2 + ||\lambda||_2^2 + ||w||_2^2\right) - \sum_{k=1}^{m} \log(\lambda_k) - \sum_{k=1}^{m} \log(w_k),$$

where $u = (x, \lambda, w) \in \mathbb{R}^n \times \mathbb{R}^m_+ \times \mathbb{R}^m_+$ and $\zeta > m$. The Jacobian is given in Appendix A.3.

When there is a constraint function h shared by all players, the root function is given by

$$\widetilde{H}(x,\tilde{\lambda},\tilde{w}) = \begin{pmatrix} \bar{L}(x,\tilde{\lambda}) \\ \tilde{g}(x) + \tilde{w} \\ \tilde{\lambda} \circ \tilde{w} \end{pmatrix}, \quad \text{with} \quad \tilde{\lambda} = \begin{pmatrix} \lambda^1 \\ \vdots \\ \lambda^N \\ \mu \end{pmatrix}, \tilde{w} = \begin{pmatrix} w^1 \\ \vdots \\ w^N \\ y \end{pmatrix} \quad \text{and} \quad \tilde{g}(x) = \begin{pmatrix} g^1(x) \\ \vdots \\ g^N(x) \\ h(x) \end{pmatrix}.$$

The Jacobian is given in Appendix A.4.

2.5.1 A classic example

Using the classic example presented above, we get

Therefore, to compute a generalized Nash equilibrium, we use

```
> z0 <- rexp(sum(dimx)+2*sum(dimlam))
> GNE.ceq(z0, dimx, dimlam, grobj=grobj, myarg, heobj=heobj, myarg,
+         constr=g, grconstr=grg, heconstr=heg,
+         method="IP", control=list(trace=0))
```

```
GNE: NA
with optimal norm NA
after  NA iterations with exit code 100 .
Output message: Error in the non smooth problem: Error in ceq.IP(xinit, dimx, dimlam, Hfinal, jacH
  internal error in line search function.
 .
Function/grad/hessian calls: NA
Optimal (vector) value: NA
```

3 GNEP as a fixed point equation

4 GNEP as a gap minimization problem

5 List of examples

5.1 Example of Facchinei et al. (2007)

We consider a two-player game defined by

$$O_1(x) = (x_1 - 1)^2 \text{ and } O_2(x) = (x_2 - 1/2)^2,$$

with a shared constraint function

$$g(x) = x_1 + x_2 - 1 \leq 0.$$

Solutions are given by $(\alpha, 1-\alpha)$ with $\alpha \in [1/2, 1]$ with Lagrange multipliers given by $\lambda_1 = 2 - 2\alpha$ and $\lambda_2 = 2\alpha - 1$. But there is a unique normalized equilibrium for which $\lambda_1 = \lambda_2 = 1/2$. The nonsmooth reformulation of the KKT system uses the following terms

$$\nabla_1 O_1(x) = 2(x_1 - 1), \nabla_2 O_2(x) = 2(x_2 - 1/2), \text{ and } \nabla_1 g(x) = \nabla_2 g(x) = 1.$$

and

$$\nabla_i^2 O_i(x) = 2, \nabla_j \nabla_k O_i(x) = 0, \text{ and } \nabla_j \nabla_k g(x) = 0.$$

5.2 The Duopoly game from Krawczyk & Uryasev (2000)

We consider a two-player game defined by

$$O_i(x) = -(d - \lambda - \rho(x_1 + x_2))x_i,$$

with

$$g_i(x) = -x_i \leq 0,$$

where $d = 20$, $\lambda = 4$, $\rho = 1$. Derivatives are given by

$$\nabla_j O_i(x) = -(-\rho x_i + (d - \lambda - \rho(x_1 + x_2))\delta_{ij}) \text{ and } \nabla_j g_i(x) = -\delta_{ij},$$

and

$$\nabla_k \nabla_j O_i(x) = -(-\rho\delta_{ik} - \rho\delta_{ij}) \text{ and } \nabla_k \nabla_j g_i(x) = 0.$$

There is a unique solution given by $x^\star = (d - \lambda)/(3\rho)$.

5.3 The River basin pollution game from Krawczyk & Uryasev (2000)

We consider a two-player game defined by

$$O_i(x) = -(d_1 - d_2(x_1 + x_2 + x_3) - c_{1i} - c_{2i}x_i)x_i,$$

and

$$g(x) = \begin{pmatrix} \sum_{l=1}^{3} u_{l1}e_l x_l - K_1 \\ \sum_{l=1}^{3} u_{l2}e_l x_l - K_2 \end{pmatrix}.$$

Derivatives are given by

$$\nabla_j O_i(x) = -(-d_2 - c_{2i}\delta_{ij})x_i - (d_1 - d_2(x_1 + x_2 + x_3) - c_{1i} - c_{2i}x_i)\delta_{ij} \text{ and } \nabla_j g(x) = \begin{pmatrix} u_{j1}e_j \\ u_{j2}e_j \end{pmatrix},$$

and

$$\nabla_k \nabla_j O_i(x) = -(-d_2\delta_{ik} - d_2\delta_{ij} - 2c_{2i}\delta_{ij}\delta_{ik}) \text{ and } \nabla_k \nabla_j g(x) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

References

Dreves, A., Facchinei, F., Kanzow, C. & Sagratella, S. (2011), 'On the solutions of the KKT conditions of generalized Nash equilibrium problems', SIAM Journal on Optimization **21**(3), 1082–1108. 9, 13

Facchinei, F., Fischer, A. & Piccialli, V. (2007), 'On generalized Nash games and variational inequalities', Operations Research Letters **35**(2), 159–164. 10

Facchinei, F., Fischer, A. & Piccialli, V. (2009), 'Generalized Nash equilibrium problems and Newton methods', Math. Program., Ser. B **117**, 163–194. 2

Krawczyk, J. & Uryasev, S. (2000), 'Relaxation algorithms to find Nash equilibria with economic applications', Environmental Modeling and Assessment **5**(1), 63–73. 10, 11

A Appendix for the nonsmooth reformulation

A.1 Semismooth reformulation – General case

The generalized Jacobian of the complementarity formulation has the following form

$$J(z) = \left(\begin{array}{ccc|cc} \mathrm{Jac}_{x_1} L_1(x,\lambda^1) & \dots & \mathrm{Jac}_{x_N} L_1(x,\lambda^1) & \mathrm{Jac}_{x_1} g^1(x)^T & 0 \\ \vdots & & \vdots & & \ddots \\ \mathrm{Jac}_{x_1} L_N(x,\lambda^N) & \dots & \mathrm{Jac}_{x_N} L_N(x,\lambda^N) & 0 & \mathrm{Jac}_{x_N} g^N(x)^T \\ \hline -D_1^a(x,\lambda^1)\mathrm{Jac}_{x_1} g^1(x) & \dots & -D_1^a(x,\lambda^1)\mathrm{Jac}_{x_N} g^1(x) & D_1^b(x,\lambda^1) & 0 \\ \vdots & & \vdots & & \ddots \\ -D_N^a(x,\lambda^N)\mathrm{Jac}_{x_1} g^N(x) & \dots & -D_N^a(x,\lambda^N)\mathrm{Jac}_{x_N} g^N(x) & 0 & D_N^b(x,\lambda^N) \end{array}\right).$$

The diagonal matrices D_i^a and D_i^b are given by

$$D_i^a(x,\lambda^i) = \mathrm{diag}[a^i(x,\lambda^i)] \quad \text{and} \quad D_i^b(x,\lambda^i) = \mathrm{diag}[b^i(x,\lambda^i)],$$

with $a^i(x,\lambda^i), b^i(x,\lambda^i) \in \mathbb{R}^{m_i}$ defined as

$$(a_j^i(x,\lambda_j^i), b_j^i(x,\lambda_j^i)) = \begin{cases} \left(\phi_a'(-g_j^i(x),\lambda_j^i), \phi_b'(-g_j^i(x),\lambda_j^i)\right) & \text{if } (-g_j^i(x),\lambda_j^i) \neq (0,0), \\ (\xi_{ij},\zeta_{ij}) & \text{if } (-g_j^i(x),\lambda_j^i) = (0,0), \end{cases}$$

where ϕ_a' (resp. ϕ_b') denotes the derivative of ϕ with respect to the first (second) argument a (b) and $(\xi_{ij},\zeta_{ij}) \in \bar{B}(p_\phi, c_\phi)$, the closed ball at p_ϕ of radius c_ϕ.

A.2 Semismooth reformulation – Shared constraint case

The generalized Jacobian of the complementarity formulation has the following form $J(z) =$

$$\left(\begin{array}{ccc|ccc|c} \mathrm{Jac}_{x_1} L_1(x,\lambda^1,\mu) & \dots & \mathrm{Jac}_{x_N} L_1(x,\lambda^1,\mu) & \mathrm{Jac}_{x_1} g^1(x)^T & & 0 & \mathrm{Jac}_{x_1} h(x)^T \\ \vdots & & \vdots & & \ddots & & \vdots \\ \mathrm{Jac}_{x_1} L_N(x,\lambda^N,\mu) & \dots & \mathrm{Jac}_{x_N} L_N(x,\lambda^N,\mu) & 0 & & \mathrm{Jac}_{x_N} g^N(x)^T & \mathrm{Jac}_{x_N} h(x)^T \\ \hline -D_1^a(x,\lambda^1)\mathrm{Jac}_{x_1} g^1(x) & \dots & -D_1^a(x,\lambda^1)\mathrm{Jac}_{x_N} g^1(x) & D_1^b(x,\lambda^1) & & 0 & 0 \\ \vdots & & \vdots & & \ddots & & \vdots \\ -D_N^a(x,\lambda^N)\mathrm{Jac}_{x_1} g^N(x) & \dots & -D_N^a(x,\lambda^N)\mathrm{Jac}_{x_N} g^N(x) & 0 & & D_N^b(x,\lambda^N) & 0 \\ \hline -D_h^a(x,\mu)\mathrm{Jac}_{x_1} h(x) & \dots & -D_h^a(x,\mu)\mathrm{Jac}_{x_N} h(x) & 0 & \dots & 0 & D_h^b(x,\mu) \end{array}\right).$$

The diagonal matrices D_a and D_b are given by

$$D_h^a(x,\mu) = \mathrm{diag}[\tilde{a}(x,\mu)] \quad \text{and} \quad D_h^b(x,\mu) = \mathrm{diag}[\tilde{b}(x,\mu)],$$

with $\tilde{a}(x,\mu), \tilde{b}(x,\mu) \in \mathbb{R}^l$ defined as

$$(\tilde{a}_j(x,\mu), \tilde{b}_j(x,\mu)) = \begin{cases} (\phi_a'(-h_j(x),\mu_j), \phi_b'(-h_j(x),\mu_j)) & \text{if } (-h_j(x),\mu_j) \neq (0,0), \\ (\tilde{\xi}_j, \tilde{\zeta}_j) & \text{if } (-h_j(x),\mu_j) = (0,0), \end{cases}$$

where $(\tilde{\xi}_j, \tilde{\zeta}_j) \in \bar{B}(p_\phi, c_\phi)$.

A.3 Semismooth reformulation – General case

For the line-search, the gradient ∇p is given by

$$\nabla p(x,\lambda,w) = \begin{pmatrix} \frac{2\zeta}{||x||_2^2+||\lambda||_2^2+||w||_2^2} x \\ \frac{2\zeta}{||x||_2^2+||\lambda||_2^2+||w||_2^2} \lambda - \lambda^{-1} \\ \frac{2\zeta}{||x||_2^2+||\lambda||_2^2+||w||_2^2} w - w^{-1} \end{pmatrix},$$

where λ and w have positive components and terms λ^{-1} and w^{-1} correspond to the component-wise inverse vector. Compared to the semismooth reformulation, the root function H is now C^1. The Jacobian is given by

$$\mathrm{Jac}\, H(x,\lambda,w) = \begin{pmatrix} \mathrm{Jac}_x \tilde{L}(x,\lambda) & \mathrm{diag}\left[\left(\nabla_{x_i} g^i(x)\right)_i\right] & 0 \\ \mathrm{Jac}_x g(x) & 0 & I \\ 0 & \mathrm{diag}[w] & \mathrm{diag}[\lambda] \end{pmatrix}.$$

As reported in Dreves et al. (2011), the computation of the direction $d_k = (d_{x,k}, d_{\lambda,k}, d_{w,k})$ can be simplified due to the special structure of the above Jacobian matrix. The system reduces to a linear system of n equations to find $d_{x,k}$ and the $2m$ components $d_{\lambda,k}, d_{w,k}$ are simple linear algebra. Using the classic chain rule, the gradient of the merit function is given by

$$\nabla \psi(x,\lambda,w) = \mathrm{Jac}\, H(x,\lambda,w)^T \nabla p(H(x,\lambda,w)).$$

Again the computation of this gradient can be simplified due to the sparse structure of $\mathrm{Jac}\, H$.

A.4 Semismooth reformulation – Shared constraint case

The Jacobian is given by

$$\operatorname{Jac}\widetilde{H}(x,\tilde{\lambda},\tilde{w}) = \begin{pmatrix} \operatorname{Jac}_x\bar{L}(x,\tilde{\lambda}) & \operatorname{Jac}_{\tilde{\lambda}}\bar{L}(x,\tilde{\lambda}) & 0 \\ \operatorname{Jac}_x\tilde{g}(x) & 0 & I \\ 0 & \operatorname{diag}[\tilde{w}] & \operatorname{diag}[\tilde{\lambda}] \end{pmatrix},$$

where

$$\operatorname{Jac}_{\tilde{\lambda}}\bar{L}(x,\tilde{\lambda}) = \begin{pmatrix} \nabla_{x_1} g^1(x) & 0 & & \nabla_{x_1} h(x) \\ 0 & \ddots & 0 & \vdots \\ & 0 & \nabla_{x_N} g^N(x) & \nabla_{x_N} h(x) \end{pmatrix},$$

and

$$\operatorname{Jac}_x\tilde{g}(x) = \begin{pmatrix} \operatorname{Jac}_x\tilde{g}^1(x) \\ \cdots \\ \operatorname{Jac}_x\tilde{g}^N(x) \\ \operatorname{Jac}_x\tilde{h}(x) \end{pmatrix}.$$