UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

ESPECIALIDAD DE MATEMÁTICA

TRABAJO FINAL DE PROGRAMACIÓN NO LINEAL:

ALGORITMOS DE PUNTOS INTERIORES

Un proyecto presentado por Anthony E. Huertas Quispe

Supervisado por:
Hector Carlos Guimaray Huerta

09 de Diciembre del 2015

Índice general

Capítulo 1

Introducción

Un modelo matemático o problema se dice que pertenece a la *programación no lineal* si la función objetivo y/o alguna de las restricciones del problema son una función no lineal de las variables de decisión (*modelo o problema no lineal*). Si la función objetivo y/o alguna de las restricciones son no lineales y las variables sólo pueden tomar valores enteros no negativos (*modelo o problema no lineal entero*), entonces el modelo matemático pertenecería al campo de la programación no lineal entera.

En la década de los 60s, los **métodos de punto interior** (MPI) fueron populares para resolver problemas de optimización, pero tuvieron un desuso en la programación lineal debido a que el método simplex abarcaba como técnica de preferencia. En los 1984, Karmarkar, un matemático y científico de la computación de la India, publicó un anuncio sobre un método de punto interior con tiempo polinomial para programación lineal; desde esa fecha métodos de punto interior tanto para programación lineal como para programación no lineal han estado teniendo mucha influencia como base para desarrollo de varios problemas de optimización.

Para dar una idea de como funcionan los métodos de punto interior pues estos parten de un punto inicial y obtienen las direcciones de búsqueda que dirigen al interior del poliedro, determinado por las restricciones dadas, hacia un óptimo local x^*.

En este informe se detallarán algunos algoritmos, de los tantos por existir, que han sido diseñados para la resolución de problemas específicos. Ha de tener en cuenta que el método de punto interior con mayor uso es el método primal-dual.

El problema de programación no lineal bajo consideración es la siguiente:

$$\begin{aligned} \min_x \ & f(x) \\ \text{s.t } & g(x) \leq 0 \\ & h(x) = 0 \end{aligned} \tag{1.1}$$

donde $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^t$ son funciones suaves.

En esta sección introductoria daremos a conocer algunos conceptos necesarios para la comprensión de los algoritmos referentes a métodos de puntos interiores que serán determinados en las secciones posteriores.

Capítulo 2

Método de Punto Interior-Región de Confianza

Este método de punto interior, es desarrollado de tal forma que posea la capacidad de detección de infactibilidad.

2.1. Certificado de Infactibilidad

En esta sección describimos un mecanismo para determinar algunos de los diseños de los algoritmos de punto interior. La estrategia consiste en confiar en la información otorgada por la componente normal en un método de paso-descomposición.

Tomando como referencia (1.1), definimos

$$w(x) = \begin{bmatrix} g(x)^+ \\ h(x) \end{bmatrix}, \text{ con } g(x)^+ = \text{máx}\{g(x), 0\} \tag{2.1}$$

así que $\|w\|$ puede servir como una medida de infactibilidad.
En un punto x estacionario infactible del problema (1,1), nosotros tenemos que

$$\|w(x)\| > 0 \text{ y } \theta(x) = \left(\|w(x)\| - \min_{\|d\| \leq \Delta} \left\| \begin{bmatrix} [g(x) + A_g(x)d]^+ \\ h(x) + A_h(x)d \end{bmatrix} \right\| \right) = 0 \tag{2.2}$$

para cualquier $\Delta > 0$, donde $A_g(x)$ y $A_h(x)$ denotan las matrices jacobianas de $g(x)$ y $h(x)$, respectivamente.
Por tanto, podremos sospechar que las iteraciones de un algoritmo de optimización no lineal aproximan a un punto estacionario infactible si la sucesión $\|w(x_k)\|$ permanece acotada lejos de cero mientras $\theta(x)$ se aproxima a cero. Estas dos condiciones pueden ser encapsuladas en una desigualdad de la forma

$$\theta(x_k) \leq \delta \|w(x_k)\|, \text{ para } \delta > 0$$

Nosotros basamos nuestro mecanismo entre los modos principal y de infactibilidad en una condición de esta forma, trasladada al marco de referencia del punto interior que incluye holguras.

2.2. Análisis del algoritmo de Punto Interior-Región de Confianza

También conocido como *Trust Region Interior Point Algorithm*. El algoritmo contiene un modo principal (*main mode*) cuyo objetivo es satisfacer las condiciones de optimalidad del programa no lineal (1.1), y un modo de factibilidad (*feasibility mode*) cuyo objetivo exclusivamente es la de mejorar, valga la redundancia, la factibilidad. Para hacer uso de los certificados de infactibilidad descritos previamente, empleamos un método de punto interior con región de confianza a paso-descomposición. Daremos una breve visión sobre este método.

Introduciendo variables de holgura, el programa no lineal (1.1) es transformado en

$$\begin{aligned} \min_{x,s} & f(x) \\ s.t \;\; g(x) + s &= 0 \\ h(x) &= 0 \\ s &\geq 0 \end{aligned} \tag{2.3}$$

La solución a este problema es obtenida resolviendo problemas de barrera de la forma

$$\begin{aligned} \min_{x,s} \varphi(x,s) \stackrel{def}{=} & f(x) - \mu \sum_{i=1}^{m} lns^{(i)} \\ s.t \;\; g(x) + s &= 0 \\ h(x) &= 0 \end{aligned} \tag{2.4}$$

con $\mu \to 0$.

Paso-descomposición se enfoca como sigue. Dado una iteración (x_k, s_k) y un radio de región de confianza Δ_k, el algoritmo calcula un paso normal $v = (v_x, v_s)$ resolviendo el subproblema

$$\begin{aligned} &\min \|g(x_k) + s_k + A_g(x_k)v_x + v_s\|^2 + \|h(x_k) + A_h(x_k)v_x\|^2 \\ &\text{s.t. } \|(v_x, S_k^{-1}v_s)\| \leq \xi\Delta_k \\ &\quad v_s \geq -\xi\kappa s, \end{aligned} \tag{2.5}$$

donde $S_k = diag\{s_k^i\}$, el escalar $\xi \in (0,1)$ es un parámetro de restricción de la región de confianza y $\kappa \in (0,1)$ determina la fracción a la regla de acotación. (Valores típicos: $\xi = 0{,}8, \kappa = 0{,}005$). El paso total $d = (d_x, d_s)$ del algoritmo es obtenido resolviendo

$$\begin{aligned} &\min_d \nabla\varphi(x_k, s_k)^T d + \tfrac{1}{2} d^T W_k d \\ &\text{s.t. } A_g(x_k)d_x + d_s = A_g(x_k)v_x + v_s \\ &\quad A_h(x_k)d_x = A_h(x_k)v_x \\ &\quad \|(d_x, S^{-1}d_s)\| \leq \Delta_k \\ &\quad d_s \geq -\kappa s, \end{aligned} \tag{2.6}$$

donde W_k es una aproximación Hessiana. La nueva iteración es dada por

$$(x_{k+1}, s_{k+1}) = (x_k, s_k) + d \tag{2.7}$$

Definimos la función de mérito como

$$\phi(x, s, \nu) = \varphi(x, s) + \nu\|c(x, s)\|, \tag{2.8}$$

donde $\nu > 0$ es un parámetro de penalidad, y

$$c(x,s) \stackrel{def}{=} \begin{bmatrix} g(x)+s \\ h(x) \end{bmatrix} \tag{2.9}$$

Luego de la nueva iteración en (2.7) calculada, el algoritmo aplica el reinicio de holgura

$$s_{k+1} \leftarrow \max\{s_{k+1}, -g(x_k)^+\},$$

que asegura que, en cada iteración,

$$g(x_{k+1}) + s_{k+1} \geq 0. \tag{2.10}$$

2.2.1. Fase de Factibilidad

En esta fase se mejora la factibilidad. Se aplica un algortimo de punto interior al problema

$$\min_x \|w(x)\|, \tag{2.11}$$

donde $w(x)$ es el vector de violación de restricciones definido en (2.1). Reformulamos (2.11) introduciendo variables de relajamiento $r_g \in \mathbb{R}^m, r_h^+, r_h^- \in \mathbb{R}^t$ como sigue:

$$\begin{array}{ll} \min\limits_{x,r} r^T 1 & \\ \text{s.t. } g(x) - r_g \leq 0 & \\ \quad h(x) - r_h^+ r_h^- = 0 & \text{con } r = \begin{bmatrix} r_g \\ r_h^+ \\ r_h^- \end{bmatrix}. \\ \quad r \geq 0, & \end{array} \tag{2.12}$$

El problema de barrera correspondiente es dado por

$$\begin{array}{ll} \min\limits_{x,r,\overline{s}} \tilde{\varphi}(x,r,\overline{s}) \stackrel{def}{=} r^T 1 - \overline{\mu} \sum\limits_{i=1}^{2m+2t} \ln \overline{s}^{(i)} & \\ \text{s.t. } g(x) - r_g + \overline{s}_g = 0 & \\ \quad h(x) - r_h^+ + r_h^- = 0 & \text{con } \overline{s} = \begin{bmatrix} \overline{s}_g \\ \overline{s}_r \end{bmatrix}, \\ \quad -r + \overline{s}_r = 0, & \end{array} \tag{2.13}$$

donde $\overline{s}_g \in \mathbb{R}^m$ y $\overline{s}_r \in \mathbb{R}^{m+2t}$ son las holguras del modo de factibilidad, y $\overline{\mu} \in \mathbb{R}_+$ es el parámetro de barrera del modo de factibilidad.

La función de Lagrange para este problema es

$$\mathcal{L}(x,r,\overline{s},\overline{\lambda}) = r^T 1 - \overline{\mu} \sum_{i=1}^{2m+2t} \ln \overline{s}^{(i)} + \overline{\lambda}_g^T (g(x) - r_g + \overline{s}_g) + \overline{\lambda}_h^T (h(x) - r_h^+ + r_h^-) + \overline{\lambda}_r^T (-r + \overline{s}_r),$$

con multiplicadores de Lagrange del modo de factibilidad $\overline{\lambda}_g \in \mathbb{R}^m, \overline{\lambda}_h \in \mathbb{R}^t$, y $\overline{\lambda}_r \in \mathbb{R}^{m+2t}$. Las condiciones de optimalidad de primer orden para el problema de barrera (2.13) son

$$F^{\overline{\mu}}(x,r,\overline{s},\overline{\lambda}) = \begin{pmatrix} A_g(x)^T \overline{\lambda}_g + A_h(x)^T \overline{\lambda}_h \\ 1 - \overline{\lambda} - \overline{\lambda}_r \\ \overline{S}_g \overline{\lambda}_g - \overline{\mu} 1 \\ \overline{S}_r \overline{\lambda}_r - \overline{\mu} 1 \\ g(x) - r_g + \overline{s}_g \\ h(x) - r_h^+ + r_h^- \\ -r + \overline{s}_r \end{pmatrix} = 0, \tag{2.14}$$

donde

$$\begin{array}{l}\overline{S}_g = \text{diag}(\overline{s}_g^{(1)}, \cdots, \overline{s}_g^{(m)}),\\ \overline{S}_r = \text{diag}(\overline{s}_r^{(1)}, \cdots, \overline{s}_r^{(m+2t)}),\end{array} \qquad \text{y} \qquad \overline{\lambda} = \begin{bmatrix} \overline{\lambda}_g \\ \overline{\lambda}_h \\ -\overline{\lambda}_h \end{bmatrix}.$$

Es conveniente introducir las notaciones siguientes:

$$\tilde{f}(x,r) = r^T 1, \quad \tilde{g}(x,r) = \begin{bmatrix} g(x) - r_g \\ -r \end{bmatrix}, \quad \tilde{h}(x,r) = h(x) + r_h^+ - r_h^-. \tag{2.15}$$

Esto nos lleva a escribir el problema (2.13) como (2.4),

$$\begin{array}{l} \min\limits_{x,r,\overline{s}} \tilde{f}(x,r) - \overline{\mu} \sum\limits_{i=1}^{2m+2t} \ln \overline{s}^{(i)} \\ \text{s.t. } \tilde{g}(x,r) + \overline{s} = 0 \\ \qquad \tilde{h}(x,r) = 0, \end{array} \tag{2.16}$$

y luego la discusión en esta sección se tornaría en base a (2.16)

2.2.2. Condiciones de cambio

Se determina cuando en el algoritmo ocurre el cambio al modo principal (main mode) o modo factible. Se depende de dos constantes $\overline{\delta} > \delta > 0$, que son preseleccionadas. Se hará uso de la función contraste $c(x,s)$ definida en (2.9).
Haremos referencia a modo principal como *main mode* y modo factible como *feasible mode*

Principales condiciones de cambio

[Main→Feasible] Suponemos que el punto (x,s) esta en modo principal y tenemos calculado un paso $d = (d_x, d_s)$, y suponemos que $s \geq -g(x)$ (ver (2.10)).

$$\begin{array}{ll} \textbf{If} & \|g(x) + s + A_g(x)d_x + d_s\| + \|h(x) + A_h(x)d_x\| \geq \delta \|c(x,s)\| \\ \textbf{then} & \text{inicia el modo de factibilidad (feasible mode)} \end{array} \tag{2.17}$$

[Feasible→Main] Suponemos que el algoritmo esta en modo de factibilidad (feasibility mode) y que esto ha calculado un paso $d = (d_x, d_r, d_s)$.

$$\begin{array}{ll} \textbf{If} & \|g(x) + s + A_g(x)d_x + d_s\| + \|h(x) + A_h(x)d_x\| \leq \overline{\delta} \|c(x,s)\| \\ \textbf{then} & \text{retorna al modo principal (main mode).} \end{array} \tag{2.18}$$

2.2.3. Finalización

El algoritmo termina en tres casos

1. Convergencia a un punto estacionario del programa no lineal (1.1). Esto es medido por el error KKT, que puede ser escrito como

$$F^M(x_k, \lambda_k) = \begin{pmatrix} \nabla f_k + (\lambda_g)_k^T A_g(x_k) + (\lambda_h)_k^T A_h(x_k) \\ g(x_k)^T (\lambda_g)_k \\ g(x_k)^+ \\ h(x_k) \end{pmatrix} \tag{2.19}$$

donde $(\lambda_g)_k \geq 0 \in \mathbb{R}^m$ y $(\lambda_h)_k \in \mathbb{R}^t$ son los multiplicadores de Lagrange en la iteración k.

2. Convergencia a un punto estacionario del problema de factibilidad (2.12). Esta decisión se basa en el error KKT para (2.12), que es dado como

$$F^F(x_k, \overline{\lambda}_k, r_k) = \begin{pmatrix} (\overline{\lambda}_g)_k^T A_g(x_k) + (\overline{\lambda}_h)_k^T A_h(x_k) \\ (g(x_k) - (r_g)_k)^T (\overline{\lambda}_g)_k \\ (g(x_k) - (r_g)_k)^+ \\ h(x_k) - (r_h^-)_k + (r_h^+)_k \\ r_k^T (\overline{\lambda}_r)_k \\ (-r_k)^+ \\ 1 - \overline{\lambda}_k - (\overline{\lambda}_r)_k \end{pmatrix} \tag{2.20}$$

Donde $\overline{\lambda}_k, r_k \geq 0$ son los multiplicadores de Lagrange y las variables de relajación en la iteración k, respectivamente. λ y $\overline{\lambda}$ son definidos en Sección 2.2.1. Cuando $\|w(x)\| > 0$ en un punto estacionario de (2.12), nosotros aceptamos esto con un certificado de infactibilidad local y terminamos.

3. El algoritmo general puede converger a un punto en donde las restricciones de (1.1) no satisfagan LICQ[1]. Este es el tercer caso en donde el algoritmo terminaría.

2.3. Algoritmo de Punto Interior - Región de Confianza

1. **Input**: $(x0, s0), \mu_1 > 0, \gamma \in (0,1), \{\epsilon_l\}_{l\geq 1} \to 0, \tau \approx 0$
2. $l = 1, k_0 = 0, (x_{k_0}, s_{k_0}) = (x_0, s_0)$;
3. $mode = N$;
4. **while** los límites de recursos no son excedidos **do**
5. **if** $\|F^M(x_{k_l}, \lambda_{k_l})\| < \tau$ **then**
6. Convergencia a un punto KKT del problema (1.1)
7. `Exit`
8. **else if** $\|F^F(x_{k_l}, \overline{\lambda}_{k_l}, r_{k_l})\| < \tau$ y $\|w(x_k)\| > \tau$ **then**
9. Convergencia a un punto estacionario infactible
10. `Exit`
11. **else**
12. /* no es punto estacionario, resolver el siguiente subproblema de barrera */
13. **if** mode=M **then**
14. $(x_{k_l}, s_{k_l}, \text{mode})$ = `Solve_Main` $((x_{k_{l-1}}, s_{k_{l-1}}), \mu_l, \varepsilon_l)$ *(ver Algoritmo 1.1)*
15. **else if** mode=F **then**
16. $(x_{k_l}, \overline{s}_{k_l}, \text{mode})$ = `Solve_Feas` $((x_{k_{l-1}}, \overline{s}_{k_{l-1}}), \mu_l, \varepsilon_l)$ *(ver Algoritmo 1.2)*
17. /* A su regreso, *mode* se ha establecido a M o F */
18. **if** mode=M **then**
19. **if** $\|\nabla f_{k_l} + (\lambda_g)_{k_l}^T A_g(x_{k_l}) + (\lambda_h)_{k_l}^T A_h(x_{k_l}), S_{k_l}(\lambda_g)_{k_l} - \mu_l\| \leq \epsilon_l$ **then**
20. Convergencia a un punto que no satisface LICQ
21. `Exit`
22. Fijar $\mu_{l+1} \in (0, \gamma\mu_l)$
23. **else if** mode=F **then**
24. Fijar $\overline{\mu}_{l+1} \in (0, \gamma\overline{\mu}_l)$
25. l=l+1

[1]Linear independence constraint qualification: los gradientes de las restricciones de desigualdad activas y los gradientes de las restricciones de igualdad son linealmente independientes en el punto óptimo x^*

Algoritmo 1.1: Solve_Main

1. **Input**: $(x0, s0), \mu > 0, \epsilon > 0, \lambda_0, \Delta_0 > 0, \eta, \rho, \beta \in (0,1), \nu_{-1} > 0, \delta > 0$
2. **while** los límites de recursos no son excedidos **do**
3. **if** $\|F^{\mu}(x_k, s_k, \lambda_k)\| < \epsilon$ **then**
4. `Return`
5. Calcular el paso normal, $v = (v_x, v_s)$ resolviendo el problema (2.5);
6. /* Si reducción de la restricción linealizada es insuficiente, cambiar al modo de restauración */
7. **if** condición (2.17) se cumple **then**
8. mode=F
9. Fijar holguras modo de restauración $\overline{s}$, multiplicadores $\overline{\lambda}$, y parámetros $\overline{\mu}$, $\overline{\Delta}$;
10. `Solve_Feas` $((x_k, \overline{s}), \overline{\mu}, \epsilon)$;
11. Calcular el paso total, $d = (d_x, d_y)$ resolviendo el problema (2.6)
12. Actualiza el parámetro de penalidad, $\nu_k \geq \nu_{k-1}$ a fin de que $pred_k(d) \leq \rho \nu_k vpred_k(v)$;
13. **if** $\phi(x_k, s_k; \nu_k) - \phi(x_k + d_k, s_k + d_s; \nu_k) < \eta pred_k(d)$ **then**
14. $\Delta_k = \beta \Delta_k$;
15. **else**
16. $x_{k+1} = x_k + d_x$;
17. $s_{k+1} = \text{máx}(s_k + d_s, -g_{k+1})$;
18. Estimar λ_{k+1};
19. Fijar $\Delta_{k+1} \geq \Delta_k$;
20. k=k+1

Algoritmo 1.1: Solve_Feas

1. **Input**: $(x0, s0), \overline{\mu} > 0, \epsilon > 0, \overline{\lambda}_0, \overline{\Delta}_0 > 0, \eta, \rho, \beta \in (0,1), \overline{\nu}_{-1} > 0$
2. **while** los límites de recursos no son excedidos **do**
3. **if** $\|F^{\overline{\mu}}(x_k, r_k, \overline{s}_k, \overline{\lambda}_k)\| < \epsilon$ **then**
4. `Return`
5. Calcular el paso total, $d = (d_x, d_r, d_s)$ usando un procedimiento de punto interior;
6. **if** condición (2.18) se cumple **then**
7. mode=M;
8. Ajustar ν_{-1} si es necesario, resetear Δ;
9. Reiniciar las holguras del modo principal s, y multiplicadores λ;
10. `Solver_Main`$((x_k, s), \mu, \epsilon)$;
11. Actualiza el parámetro de penalidad, $\overline{\nu}_k \geq \overline{\nu}_{k-1}$: $pred_k(d) \leq \rho \overline{\nu}_k vpred_k(v)$;
12. **if** $\phi(x_k, r_k; \overline{s}_k) - \phi(x_k + d_k, r_k + d_r; \overline{s}_k + d_s) < \eta pred_k(d)$ **then**
13. $\overline{\Delta}_k = \beta \overline{\Delta}_k$;
14. **else**
15. $x_{k+1} = x_k + d_x, r_{k+1} = r_k + d_r$;
16. $\overline{s}_{k+1} = \text{máx}(\overline{s}_k + d_s, -\overline{g}_{k+1})$;
17. Estimar $\overline{\lambda}_{k+1}$;
18. Fijar $\overline{\Delta}_{k+1} \geq \overline{\Delta}_k$;
19. k=k+1

Capítulo 3

Método de Punto Interior - Busqueda de Línea de Filtro

En particular, este método proporciona una alternativa atractiva a las estrategias de conjunto activo en la manipulación de problemas con un gran número de restricciones de desigualdad. A diferencia de (1.1) consideraremos nuestro problema de la forma siguiente

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } c(x) = 0 \\ &x_L \leq x \leq x_U, \end{aligned} \tag{3.1}$$

donde $x_L \in [-\infty, \infty)^n$ y $x_U \in (-\infty, \infty]^n$ con $x_L^{(i)} \leq x_U^{(i)}$ son las cotas inferiores y superiores en las variables x. La función objetivo $f : \mathbb{R}^n \to \mathbb{R}$ y las restricciones de igualdad $c : \mathbb{R}^n \to \mathbb{R}^m$, con $m \leq n$, se asumen que son dos veces continuamente diferenciable. Los problemas con restricciones de desigualdad ($d(x) \leq 0$) las reformularemos como en (3.1) con tan solo introduciendo variables de holgura.
Ha de tener en cuenta que la i-ésima componente de un vector $v \in \mathbb{R}^n$ será representada como $v^{(i)}$; además ϵ_{maq} indicará el error de la máquina ($\epsilon \approx 10^{-16}$)

3.1. Análisis del Algoritmo de Punto Interior - Búsqueda de Línea

3.1.1. Enfoque Primal-Dual

Para simplificar notación, describimos primero el método para el problema

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &\text{s.t. } c(x) = 0 \\ &\quad x \geq 0. \end{aligned} \tag{3.2}$$

El cambio necesario para el caso general se verá en sección (2.).

El algoritmo propuesto calcula soluciones para una serie de problemas de barrera

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \varphi_\mu(x) := f(x) - \mu \sum_{i=1}^{n} \ln(x^{(i)}) \\ &\text{s.t. } c(x) = 0 \end{aligned} \tag{3.3}$$

con $\mu \to 0$.

Equivalentemente, se interpreta como aplicar un método homotópico a las ecuaciones primal-dual,

$$\begin{aligned} \nabla f(x) + \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZ1 - \mu 1 &= 0 \end{aligned} \tag{3.4}$$

con parámetro homotópico $\mu \to 0$, $\lambda \in \mathbb{R}^m$ y $z \in \mathbb{R}^n$ son los multiplicadores de Lagrange para las restricciones en (3.2) de $c(x)$ y x, respectivamente; $X = diag(x)$, $Z = diag(z)$. Notar que en (3.4) para $\mu = 0$ junto con "$x, z \geq 0$" son las condiciones KKT para el problema original en (3.2).
El presente método calcula una solución aproximada al problema (3.3) para un valor fijo μ, luego decrece el parámetro de barrera, y continúa de igual forma.
Definimos el error de optimalidad, usando (3,4), como

$$E_\mu(x, \lambda, z) := \text{máx}\left\{ \frac{\|\nabla f(x) + \nabla c(x)\lambda - z\|_\infty}{s_d}, \|c(x)\|_\infty, \frac{\|XZ1 - \mu 1\|_\infty}{s_c} \right\} \tag{3.5}$$

con parámetros de escala $s_d, s_c \geq 1$. $E_0(x.\lambda, z)$, si $\mu = 0$. El algoritmo en general termina si la solución aproximada $(\overline{x}_*, \overline{\lambda}_*, \overline{z}_*)$ satisface

$$E_0(\overline{x}_*, \overline{\lambda}_*, \overline{z}_*) \leq \epsilon_{tol} \tag{3.6}$$

donde $\epsilon_{tol} > 0$ es el error de tolerancia que el usuario otorga.
Para adaptar los criterios de términación a ciertas circunstancias, escojemos los factores de escala de la forma

$$s_d = \text{máx}\left\{ s_{\text{máx}}, \frac{\|\lambda\|_1 + \|z\|_1}{(m+n)} \right\} / s_{\text{máx}} \qquad s_c = \text{máx}\left\{ s_{\text{máx}}, \frac{\|z\|_1}{n} \right\} / s_{\text{máx}}$$

en (3.5). En este sentido, un componente del error de optimalidad es escalada, siempre que el valor promedio de los multiplicadores se vuelvan más grandes que el número fijo $s_{\text{máx}} \geq 1$.

Para lograrse una rápida convergencia local (a una solución local en (3.2) satisfaciendo las condiciones de optimalidad suficientes de segundo orden fuertes) diseñaremos como se sigue; denotando con j el contador de iteración para el bluce externo (*outer loop*), se requiere que la solución aproximada $(\tilde{x}_{*,j+1}, \tilde{\lambda}_{*,j+1}, \tilde{z}_{*,j+1})$ del problema (3.3), para un valor μ_j, satisfaga la tolerancia

$$E_{\mu_j}(\tilde{x}_{*,j+1}, \tilde{\lambda}_{*,j+1}, \tilde{z}_{*,j+1}) \leq k_\epsilon \mu_j$$

para una constante $k_\epsilon > 0$, luego el nuevo parámetro de barrera es obtenido de

$$\mu_{j+1} = \text{máx}\left\{ \frac{\epsilon_{tol}}{10}, \text{mín}\left\{ k_\mu \mu_j, \mu_j^{\theta_\mu} \right\} \right\} \tag{3.7}$$

con constantes $k_\mu \in (0, 1)$ y $\theta_\mu \in (1, 2)$. En este sentido, el parámetro de barrera eventualmente decrecería. Como referencia posterior, escogemos un parámetro *fracción del límite*

$$\tau_j = \text{máx}\{\tau_{\text{mín}}, 1 - \mu_j\} \tag{3.8}$$

donde $\tau_{\text{mín}} \in (0, 1)$ es su mínimo valor.

3.1.2. Solución del Problema de Barrera

Con objeto a resolver el problema (3.3) para un valor fijo dado μ_j de parámetro de barrera.un método de Newton amortiguado es aplicado al problema primal-dual en (3.4). Usaremos k como contador de iteración del bucle interno (*inner loop*). Dado una iteración (x_k, λ_k, z_k) con $x_k, z_k > 0$, hallamos las direcciones $(d_k^x, d_k^\lambda, d_k^z)$, ,haciendo uso de (3.4), en

$$\begin{bmatrix} W_k & A_k & -I \\ A_k^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \\ d_k^z \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) + A_k\lambda_k - z_k \\ c(x_k) \\ X_k Z_k 1 - \mu_j 1 \end{pmatrix}. \tag{3.9}$$

donde $A_k = \nabla c(x_k)$ y W_k denota la Hessiana $\nabla^2_{xx}\mathcal{L}(x_k, \lambda_k, z_k)$ de la función lagrangiana de (3.2),

$$\mathcal{L}(x, \lambda, z) := f(x) + c(x)^T\lambda - z. \tag{3.10}$$

En lugar de resolver el sistema lineal no simétrico (3.9) directamente, el método resuelve primero el sistema lineal

$$\begin{bmatrix} W_k + \sum_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla\varphi_{\mu_j}(x_k) + A_k\lambda_k \\ c(x_k) \end{pmatrix} \tag{3.11}$$

con $\sum_k := X_k^{-1}Z_k$. El vector d_k^z es obtenido de

$$d_k^z = \mu_j X_k^{-1} 1 - z_k - \textstyle\sum_k d_k^x. \tag{3.12}$$

Necesitamos modificar la matriz de iteración, por ello en la implementación, resolvemos el sistema lineal

$$\begin{bmatrix} W_k + \sum_k + \delta_w I & A_k \\ A_k^T & -\delta_c I \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla\varphi_{\mu_j}(x_k) + A_k\lambda_k \\ c(x_k) \end{pmatrix} \tag{3.13}$$

para algún $\delta_w, \delta_c \geq 0$.

de (13) y (12), tamaños de paso $\alpha_k, \overline{\alpha}_k^z \in (0, 1]$ se calculan para las iteraciones

$$\begin{array}{rcl} x_{k+1} & := & x_k + \alpha_k d_k^x \\ \lambda_{k+1} & := & \lambda_k + \alpha_k d_k^\lambda \\ z_{k+1} & := & z_k + \alpha_k^z d_k^z \end{array} \tag{3.14}$$

Desde que x y z son positivos, esta propiedad se mantiene para todas las iteraciones. Esto se alcanza usando la regla de la *fracción al límite*

$$\begin{array}{rcl} \alpha_k^{\text{máx}} & := & \text{máx}\{\alpha \in (0, 1] : x_k + \alpha d_k^x \geq (1 - \tau_j)x_k\} \\ \alpha_k^z & := & \text{máx}\{\alpha \in (0, 1] : z_k + \alpha d_k^z \geq (1 - \tau_j)z_k\} \end{array} \tag{3.15}$$

donde el parámetro $\tau_j \in (0, 1)$ es definido en (3.8).
Reiniciamos

$$z_{k+1}^{(i)} \leftarrow \text{máx}\left\{\text{mín}\left\{z_{k+1}^{(i)}, \frac{k_\sum \mu_j}{x_{k+1}^{(i)}}\right\}, \frac{\mu_j}{k_\sum x_{k+1}^{(i)}}\right\}, \qquad i = 1, \ldots, n, \tag{3.16}$$

para algún valor $k_\sum \geq 1$ fijado después de cada paso. Esto garantiza que cada componente $\sigma_{k+1}^{(i)}$ de $\sum_{k+1}$ esta en el intervalo

$$\sigma_{k+1}^{(i)} \in \left[\frac{1}{k_\sum}/(x_k^{(i)})^2, k_\sum \mu_j/(x_k^{(i)})^2\right]. \tag{3.17}$$

Tal protección son comunes para asegurar la prueba de la convergencia global de métodos primal-dual para NLP.

3.1.3. Método de Búsqueda de Línea de Filtro.

También conocido como *Line-Search Filter Method*, propuesto por Fletcher y Leyffer. Consideramos el punto de prueba $x_k(\alpha_{k,l}) := x_k + \alpha_{k,l} d_k^x$ durante la búsqueda de línea de retroceso que sea aceptable, si

$$\begin{array}{c} \theta(x_k(\alpha_{k,l})) \leq (1-\gamma_\theta)\theta(x_k) \\ \text{o} \quad \varphi_{\mu_j}(x_k(\alpha_{k,l})) \leq \varphi_{\mu_j}(x_k) - \gamma_\varphi \theta(x_k) \end{array} \tag{3.18}$$

se sostiene para constantes fijas $\gamma_\theta, \gamma_\varphi \in (0,1)$, donde $\alpha_{k,l} = 2^{-l}\alpha_k^{\text{máx}} con (l = 0, 1, \ldots)$. Sin embargo, este criterio es reemplazado por requerimiento de un proceso suficiente en la función de barrera objetivo, siempre que para la actual iteración $\theta(x_k) \leq \theta^{\text{mín}}$, para alguna constante $\theta^{\text{mín}} \in (0, \infty]$, y la condición de cambio

$$\nabla\varphi_{\mu_j}(x_k)^T d_k^x < 0 \qquad \text{y} \qquad \alpha_{k,l}[-\nabla\varphi_{\mu_j}(x_k)^T d_k^x]^{s_\varphi} > \delta[\theta(x_k)]^{s_\theta}, \tag{3.19}$$

con constantes $\delta > 0, s_\theta > 1, s_\theta \geq 1$. Si $\theta(x_k) \leq \theta^{\text{mín}}$ y (3.19) es verdad entonces el punto de prueba satisface la condición de Armijo

$$\varphi_{\mu_j}(x_k(\alpha_{k,l})) \leq \varphi_{\mu_j}(x_k) + \eta_\varphi \alpha_{k,l} \nabla\varphi_{\mu_j}(x_k)^T d_k^x. \tag{3.20}$$

donde $\eta_\varphi \in (0, \frac{1}{2})$ para que sea aceptable.
El agoritmo mantiene un *filtro*, un conjunto $\mathcal{F}_k \subseteq \{(\theta, \varphi) \in \mathbb{R}^2 : \theta \geq 0\}$ para cada iteración k. El filtro $\mathcal{F}$ contiene combinación de valores θ de violacion de restricciones y valores φ de función objetivo, que son prohibidos para un punto de prueba exitoso en la iteración k. Se rechaza $x_k(\alpha_{k,l})$, si $(\theta(x_k(\alpha_{k,l})), \varphi_{\mu_j}(x_k(\alpha_{k,l}))) \in \mathcal{F}_k$. El filtro inicial y para iteración k son

$$\mathcal{F}_0 := \{(\theta, \varphi) \in \mathbb{R}^2 : \theta \geq \theta^{\text{máx}}\} \tag{3.21}$$

$$\mathcal{F}_{k+1} := \mathcal{F}_k \cup \left\{\{(\theta, \varphi) \in \mathbb{R}^2 : \theta \geq (1-\gamma_\theta)\theta(x_k)\} \qquad \text{y} \qquad \varphi \geq \varphi_{\mu_j}(x_k) - \gamma_\varphi\theta(x_k)\right\} \tag{3.22}$$

para algún $\theta^{\text{máx}}$. Este procedimiento asegura que el algoritmo no sea cíclico. En algunos casos no es posible encontrar $\alpha_{k,l}$ que satisfaga los criterios de arriba, por ello definimos

$$\alpha_k^{\text{mín}} := \gamma_\alpha . \begin{cases} \text{mín}\left\{\gamma_\theta, \dfrac{\gamma_\varphi\theta(x_k)}{-\nabla\varphi_{\mu_j}(x_k)^T d_k^T}, \dfrac{\delta[\theta(x_k)]^{s_\theta}}{[-\nabla\varphi_{\mu_j}(x_k)^T d_k^x]^{s_\varphi}}\right\} \\ \qquad\qquad \text{si } \nabla\varphi_{\mu_j}(x_k)^T d_k^x < 0 \text{ y } \theta(x_k) \leq \theta^{\text{mín}} \\ \text{mín}\left\{\gamma_\theta, \dfrac{\gamma_\varphi\theta(x_k)}{-\nabla\varphi_{\mu_j}(x_k)^T d_k^T}\right\} \\ \qquad\qquad \text{si } \nabla\varphi_{\mu_j}(x_k)^T d_k^x < 0 \text{ y } \theta(x_k) > \theta^{\text{mín}} \\ \gamma_\theta \\ \qquad\qquad \text{en otro caso.} \end{cases} \tag{3.23}$$

con un *factor de seguridad* $\gamma_\alpha \in (0, 1]$. El algoritmo se revierte a una fase de restauración de factibilidad cuando $\alpha_{k,l} \leq \alpha_k^{\text{mín}}$. Luego el algoritmo intenta buscar $x_{k+1} > 0$ que sea aceptable al actual filtro y para que (3.18) se sostenga.
Para asegurar convergencia global del método es suficiente asegurar convergencia global para cada parámetro de barrera con un valor fijo μ_l. Por tanto, el filtro $\mathcal{F}$ es reiniciado a su valor inicial (3.21) cuando μ_l decrece.

3.1.4. Correcciones de Segundo Orden (SOC)

Se usa para mejorar el paso en donde el punto de prueba es rechazado. El objetivo es reducir la inFactibilidad para un paso $\tilde{d}_k^x$ aplicando un paso del tipo Newton adicional para las restricciones en el punto $x_k + \tilde{d}_k^x$, usando el jacobiano A_k^T en x_k. En el método, si el primer paso de prueba $\alpha_{k,0}$ ha sido rechazado y si $\theta(x_k(\alpha_{k,0})) \geq \theta(x_k)$, la correción de segundo orden $d_k^{x,soc}$ (para el paso $\tilde{d}_k^x = \alpha_{k,0} d_k^x$) es calculado tal que satisfaga

$$A_k^T d_k^{x,soc} + c(x_k + \alpha_{k,0} d_k^x) = 0 \tag{3.24}$$

La nueva dirección corregida es obtenida de

$$d_k^{x,cor} = \alpha_{k,0} d_k^x + d_k^{x,soc}. \tag{3.25}$$

Para evitar factorizaciones de matrices adicionales, se usa como en (3.13) para calcular el paso correcto (3.25) de

$$\begin{bmatrix} W_k + \sum_k + \delta_w I & A_k \\ A_k^T & -\delta_c I \end{bmatrix} \begin{pmatrix} d_k^{x,cor} \\ d_k^{\lambda} \end{pmatrix} = - \begin{pmatrix} \nabla \varphi_{\mu_j}(x_k) + A_k \lambda_k \\ c_k^{soc} \end{pmatrix} \tag{3.26}$$

Aquí, escogemos

$$c_k^{soc} = \alpha_{k,0} c(x_k) + c(x_k + \alpha_{k,0} d_k^x), \tag{3.27}$$

que es obtenido de (3.13),(3.24) y (3.25). Luego aplicamos la regla de fración del límite

$$\alpha_k^{soc} := \text{máx}\{\alpha \in (0,1] : x_k + \alpha d_k^x \geq (1 - \tau_j) x_k\} \tag{3.28}$$

y verificar si $x_k^{soc} := x_k + \alpha_k^{soc} d_k^{x,cor}$ es aceptable al filtro y satisface su criterio de aceptación. x_k^{soc} reemplaza $x(\alpha_k)$ en (3.20).

3.2. Algoritmo de Punto Interior - Búsqueda de Línea de Filtro

Algoritmo 2

1. Punto inicial (x_0, λ_0, z_0) con $x_0, z_0 > 0$; $\mu_0 > 0$; $\epsilon_{tol} > 0$; $s_{\text{máx}} \geq 1$; $k_\epsilon > 0$, $k_\mu \in (0,1)$; $\theta_\mu \in (1,2)$; $\tau_{\text{mín}} \in (0,1)$ $k_\sum > 1$; $\theta_{\text{máx}} \in (\theta(x_0), \infty)$; $\theta_{\text{mín}} > 0; \gamma_\theta, \gamma_\varphi \in (0,1)$; $\delta > 0$; $\gamma_\alpha \in (0,1]$; $s_\theta > 1$; $s_\varphi \geq 1$; $\eta_\varphi \in (0, \frac{1}{2})$; $k_{soc} \in (0,1)$; $p^{\text{máx}} \in \{0,1,2\ldots\}$.

2. Inicializar los contadores de iteración $j \leftarrow 0$ y $k \leftarrow 0$, tan bien como el filtro $\mathcal{F}_0$ en (3.21). Obtener τ_0 de (3.8).

3. **if** $E_0(x_k, \lambda_k, z_k) \leq \epsilon_{tol}$ **then** `STOP`

4. **if** $E_{\mu_j}(x_k, \lambda_k, z_k) \leq k_\epsilon \mu_j$, **then**

5. Calcular μ_{j+1} y τ_{j+1} de (3.7) y (3.8), y hacer $j \leftarrow j + 1$;

6. Reinicializar el filtro $\mathcal{F}_k \leftarrow \{(\theta, \varphi) \in \mathbb{R}^2 : \theta \geq \theta^{\text{máx}}\}$;

7. **if** $k = 0$ repetir el paso 4, en otro caso continuar a 8.

8. Calcular $(d_k^x, d_k^\lambda, d_k^z)$ de (3.13), donde δ_w y δ_c son obtenidos del *algoritmo IC*

9. Retomar la búsqueda de línea.

10. Hacer $\alpha_{k,0} = \alpha_k^{\text{máx}}$ con $\alpha_k^{\text{máx}}$ de (3.15), hacer $l \leftarrow 0$.

11. $x_k(\alpha_{k,l}) := x_k + \alpha_{k,l} d_k^x$.

12. **If** $(\theta(x_k(\alpha_{k,l})), \varphi_{\mu_j}(x_k(\alpha_{k,l})) \in \mathcal{F}_k$, rechazar el paso de prueba e ir a 15.

13. -CASO I: $\theta(x_k) \leq \theta^{\text{mín}}$ y (3.19) se cumple: **if** (3.20) se cumple, aceptar el paso de prueba $x_{k+1} := x_k(\alpha_{k,l})$ e ir a 22. En otro caso, ir a 15.

14. -CASO II: $\theta(x_k) > \theta^{\text{mín}}$ o (3.19) no se cumple: **if** (3.18), aceptar el paso de prueba $x_{k+1} := x_k(\alpha_{k,l})$ e ir a 22. En otro sentido, ir a 15.

15. **if** Si $l > 0$ o $\theta(x_k, 0) < \theta(x_k)$, salir e ir a 21. En otro caso, inicializar contador $p \leftarrow 1$ y c_k^{soc} de (3.27). Inicializar $\theta_{old}^{soc} \leftarrow \theta(x_k)$.

16. Calcular $d_k^{x,cor}$ y d_k^λ de (3.26), α_k^{soc} de (28), y $x_k^{soc} := x_k + \alpha_k^{soc} d_k^{x,cor}$.

17. **if** $(\theta(x_k^{soc})), \varphi_{\mu_j}(x_k^{soc}) \in \mathcal{F}_k$, rechazar el paso de prueba e ir a 21.

18. -CASO I: $\theta(x_k) \leq \theta^{\text{mín}}$ y (3.19) se cumple (para $\alpha_{k,0}$): **if** (3.20) se cumple con $x_k(\alpha_k, l)$ reemplazado por x_k^{soc}, aceptar el paso de prueba $x_{k+1} := x_k^{soc}$ e ir a 22. En otro caso, ir a 20.

19. -CASO II: $\theta(x_k) > \theta^{\text{mín}}$ o (3.19) no se cumple (para $\alpha_{k,0}$): **if** (3.18) se cumple con $x_k(\alpha_k, l)$ reemplazado por x_k^{soc}, aceptar el paso de prueba $x_{k+1} := x_k^{soc}$ e ir a 22. En otro sentido, ir a 20.

20. (SOC). **if** $p = p^{\text{máx}}$ o $\theta(x_k^{soc}) > k_{soc}\theta_{old}^{soc}$, salir y continuar en 21. En otro caso, $p \leftarrow p + 1$, $c_k^{soc} \leftarrow \alpha_k^{soc} c_k^{soc} + c(x_k^{soc})$ y $\theta_{old}^{soc} \leftarrow \theta(x_k^{soc})$. Retornar a 16.

21. Hacer $\alpha_{k,l+1} = \frac{1}{2}\alpha_{k,l}$ y $l \leftarrow l + 1$. **if** $\alpha_{k,l} < \alpha_k^{\text{mín}}$ con $\alpha_k^{\text{mín}}$ definido en (3.23), ir a 25. En otro caso, retornar a 11.

22. Hacer $\alpha_k := \alpha_{k,l}$ (o $\alpha_k := \alpha_k^{soc}$ si punto SOC fue aceptado en (18,19)), y actualizar λ_{k+1} y z_{k+1} de (3.14) con α_k^z de (3.15). Aplicamos (3.16) para corregir z_{k+1} si es necesario.

23. Si (3.19) y (3.20) no se cumplen para α_k, cambiar el filtro usando (3.22). En otro caso, $\mathcal{F}_{k+1} := \mathcal{F}_k$

24. $k \leftarrow k + 1$, retornar a 3.

25. Cambiar el filtro usando (3.22), y calcular la nueva iteración $x_{k+1} > 0$ por decrecimiento de la medida de infactibilidad $\theta(x)$, de modo que x_{k+1} sea aceptable, es decir $(\theta(x_{k+1})) \notin \mathcal{F}_{k+1}$. Luego continuar con la iteración regular en 24.

Algoritmo IC: Inertia Correction

Input: constantes $0 < \overline{\delta}_w^{\text{mín}} < \overline{\delta}_w^{0} < \overline{\delta}_w^{\text{máx}}$; $\overline{\delta}_c > 0$; $0 < k_w^- < 1 < k_w^+ < \overline{k}_w^+$; $k_c \geq 0$.
Inicializar $\delta_w^{last} \leftarrow 0$.
En cada iteración k:

1. Intento de factorizar la matriz no modificada en (3.13) con $\delta_w = \delta_c = 0$. Si la matriz es no singular y su inercia es $(n, m, 0)$, usar la dirección de búsqueda resultante en la búsqueda de línea. En otro caso, continuar en 2.

2. Si la iteración de matriz tiene cero valores propios, hacer $\delta \leftarrow \overline{\delta}_c \mu^{k_c}$, en otro caso hacer $\delta_c \leftarrow 0$.

3. **if** $\delta_w^{last} = 0$, hacer $\delta \leftarrow \overline{\delta}_w^{0}$. En otro caso, hacer $\delta_w \leftarrow \text{máx}\{\overline{\delta}_w^{\text{mín}}, k_w^- \delta_w^{last}\}$.

4. Intento de factorizar la matriz no modificada en (3.13). Si la inercia es ahora $(n, m, 0)$, ahcer $\delta_w^{last} \leftarrow \delta_w$ y usar la dirección de búsqueda resultante en la búsqueda de linea. En otro caso continuar en 5.

5. **if** $\delta_w^{last} = 0$, hacer $\delta_w \leftarrow \overline{k}_w^+ \delta_w$. En otro caso $\delta_w \leftarrow k_w^+ \delta_w$.

6. **if** $\delta_w > \overline{\delta}_w^{last}$, abortar la dirección de búsqueda de cálculo, evitar la búsqueda de línea de retroceso, y cambiar directamente la fase de restauración en (Algoritmo 2-paso 25)

Bibliografía

[1] Benson, H.Y., Shanno, D. F.,Vanderbei, R. J.: Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions. Computational Optimization and Applications, 23 (2), 257 - 272 (2002)

[2] Byrd, R. H., Gilbert, J. Ch., Nocedal, J.: A trust region method based on interior point techniques for nonlinear programming. Mathematical Programming, 89, 149 - 185 (2000)

[3] Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Mathematical Programming, 91 (2), 239 - 269 (2002)

[4] Nocedal, J. Oztoprak, F. WaltzAn, R.: Interior Point Method for Nonlinear Programming with Infeasibility Detection Capabilities. Mathematics Subject Classification (2000)

[5] Nemirovski, A. Todd, M.: Interior-point methods for optimization. Cambridge University Press (2008)

[6] Wachter, A. Biegler, L.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Springer-Verlag (2005)