Ch 19: Interior-point methods for nonlinear programming

* Most-powerful for large-scale, together with active-set SQP.
* primal-dual steps
* two classes
* Idea was abandoned due to SQP, now reborn
* Excellent for large-scale problems
* Not in 1999 edition. Added in 2006.
* The other Chapter added is Chapter 9.

<u>Consider:</u> $\min_{x,s} f(x)$

<u>subject to:</u> $C_E(x) = 0$, (equality constr.)

$C_I(x) - s = 0$, (ineq converted to equality).

$s \geq 0$.

<u>Here:</u> * $C_I(x) \geq 0$ is represented by: $C_I(x) - s = 0$.

* l equalities and m inequalities.

19.1 Two interpretations

Continuation approach #1:

KKT for problem:

L_x

$$\nabla f(x) - A_E^T(x)\, y - A_I^T(x)\, z = 0$$

$$Sz - \mu e = 0$$

$$C_E(x) = 0$$

$$C_I(x) - s = 0$$

with $\mu = 0$,

$$s \geq 0, \quad z \geq 0$$

<u>Where:</u> y, z are Lagrange multipliers

$S = \mathrm{diag}(s_1, s_2, \ldots, s_n)$

Generate $(x(\mu), s(\mu), y(\mu), z(\mu))$, the <u>primal-dual central path</u> that converges to (x^*, s^*, y^*, z^*) as $\mu \to 0$.

Barrier approach #2:

$$\min_{x,s} f(x) - \mu \sum_{i=1}^{m} \log s_i$$

subject to: $c_E(x) = 0$

$$c_I(x) - s = 0$$

$\mu > 0$. $\mu_k \to 0$ to produce the solution

KKT conditions:

std form

New $\mathcal{L}_{(x,s)} = 0$
$$\begin{cases} \nabla f(x) - A_E^T(x) y - A_I^T(x) z = 0 & \Leftarrow \nabla_x \\ -\mu S^{-1} e + z = 0 & \Leftarrow \nabla_s \end{cases}$$

$$\left.\begin{aligned} c_E(x) &= 0 \\ c_I(x) - s &= 0 \end{aligned}\right\} \Leftarrow \text{std constraints}$$

<u>Note:</u> $\mathcal{L}(x, s, y, z) = f(x) - y^T c_E(x) - z^T (c_I(x) - s)$
$$- \mu \sum_{i=1}^{m} \log s_i$$

∇_s

Note:

$$\nabla_{s}\left(-\mu\sum_{i=1}^{n}\log(s_i)\right)=$$

$$=-\mu\sum_{i=1}^{n}1/s_i=-\mu S^{-1}e$$

with: $S^{-1}=(\mathrm{diag}(s_1,s_2,\dots,s_n))^{-1}$
$e=(1,1,1,\dots,1)^T$

* Approach #1 differs from Approach #2 in 2nd equs:

$$Sz-\mu e=0$$

vs $-\mu S^{-1}e+z=0$ (non-linear in s_i)

But multiply #2 by S:

$\Rightarrow -\mu e+Sz=0$ same as #1.

Barrier methods in section 19.6.

- Can be designed to stay feasible if a feasible point is obtained.

Apply Newton to #1:

Let $\mathcal{L}(x,s,y,z) = f(x) - y^T c_E(x) - z^T(c_I(x) - s)$

Apply $\nabla^2_{(x,s,y,z)}$ and $\nabla_{(x,s,y,z)}$

to get:

$$\begin{array}{cccc} & & \nabla_y(\nabla_x \mathcal{L}) & \nabla_z(\nabla_x \mathcal{L}) \\ \left[\begin{array}{cccc} \nabla^2_{xx}\mathcal{L} & 0 & -A_E^T(x) & -A_I^T(x) \\ 0 & Z & 0 & 0 \\ A_E(x) & 0 & 0 & 0 \\ A_I(x) & -I & 0 & 0 \end{array}\right] & & & \end{array} \begin{bmatrix} p_x \\ p_s \\ p_y \\ p_z \end{bmatrix}$$

$$= -\begin{bmatrix} \nabla f(x) - A_E^T(x)\, y - A_I^T(x)\, z \\ Sz - \mu e \\ c_E(x) \\ c_I(x) - s \end{bmatrix} \begin{array}{l} \leftarrow \nabla_x \\ \leftarrow \nabla_s \\ \leftarrow \nabla_y \\ \leftarrow \nabla_z \end{array}$$

Note the structure of this Newton step:

$$\begin{bmatrix} \nabla_{xx}^2 \mathcal{L} & \nabla_s(\nabla_x \mathcal{L}) & \nabla_y(\nabla_x \mathcal{L}) & \nabla_z(\nabla_x \mathcal{L}) \\ \nabla_x(\nabla_s \mathcal{L}) & \nabla_{ss}^2 \mathcal{L} & \nabla_y(\nabla_s \mathcal{L}) & \nabla_z(\nabla_s \mathcal{L}) \\ \nabla_x(\nabla_y \mathcal{L}) & \nabla_s(\nabla_y \mathcal{L}) & \nabla_{yy}^2 \mathcal{L} & \nabla_z(\nabla_y \mathcal{L}) \\ \nabla_x(\nabla_z \mathcal{L}) & \nabla_s(\nabla_z \mathcal{L}) & \nabla_y(\nabla_z \mathcal{L}) & \nabla_{zz}^2 \mathcal{L} \end{bmatrix} \begin{bmatrix} p_x \\ p_s \\ p_y \\ p_z \end{bmatrix}$$

$$= - \begin{bmatrix} \nabla_x \mathcal{L} \\ \nabla_s \mathcal{L} \\ \nabla_y \mathcal{L} \\ \nabla_z \mathcal{L} \end{bmatrix}$$

or: $\boxed{\left(\nabla^2_{(x,s,y,z)} \mathcal{L}\right) p = -\nabla_{(x,s,y,z)} \mathcal{L}}$ I

with: $\boxed{\mathcal{L}(x,s,y,z) = f(x) - y^T c_E(x) - z^T(c_I(x) - s)}$ II

This is the primal-dual system

The step becomes:

$$\left.\begin{aligned} x^+ &= x + \alpha_s^{max} p_x \\ s^+ &= s + \alpha_s^{max} p_s \end{aligned}\right\} \text{ using: } \boxed{\alpha_s^{max} = \max\{\alpha \in (0,1] : s + \alpha p_s \geq (1-\tau) s\}} \text{ III}$$

$$\left.\begin{aligned} y^+ &= y + \alpha_z^{max} p_y \\ z^+ &= z + \alpha_z^{max} p_z \end{aligned}\right\} \text{ using: } \boxed{\alpha_z^{max} = \max\{\alpha \in (0,1] : z + \alpha p_z \geq (1-\tau) z\}} \text{ IV}$$

Here, $\tau \in (0,1)$. Typically, $\tau = 0.995$

The α_s^{max}, α_z^{max} come from the "fraction to the boundary rule", preventing $s, z \to 0$ too quickly. Eg: $\underbrace{s + \alpha p_s \geq 0.005\, s \text{ and } z + \alpha p_z \geq 0.005\, z}$.

→ bounding reduction.

⇒ Very simple: $\alpha p_s \geq -\tau s$, $\alpha \in (0, 1]$

Solve for α for "$\geq$" component-wise.

In addition, we need to update "barrier parameters" μ_k:

- Keep constant until KKT is satisfied or
- Update with k

→ Covered in 19.3.

Also, superlinear convergence.

For stopping criteria, return to the KKT conditions:

$$E(x, s, y, z; \mu) = \max\{\|\nabla f(x) - A_E^T(x) y - A_I(x)^T z\|, \|Sz - \mu e\|, \|c_E(x)\|, \|c_I(x) - s\|\}$$

for some norm $\|\cdot\|$. Note max error.

Algorithm 19.1 (Basic interior-point algorithm)

Choose $x_0, s_0 > 0$ and compute $y_0, z_0 > 0$.
Select $\mu_0 > 0$, $\sigma, \tau \in (0,1)$.

$k \leftarrow 0$
Repeat until stopping test (more on this later)
  Repeat until $E(x_k, s_k, y_k, z_k; \mu_k) \leq \mu_k$

  Solve $\nabla^2 \mathcal{L}\, p = -\nabla \mathcal{L}$
  Compute $\alpha_s^{max}, \alpha_z^{max}$ as in III, IV

  Update $x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1}$

  $\mu_{k+1} \leftarrow \mu_k$ (more on this later)
  $k \leftarrow k+1$
  end
  Choose $\mu_k \in (0, \sigma\mu_k)$ (must $\mu_k \to 0$)
end

Thm 19.1: Assume we get out of the inner loop and $\mu_k \to 0$. Suppose f and c are ctsly diff'ble. Then all limit points $\hat{x}$ of $\{x_k\}$ are feasible. If any limit point $\hat{x}$ satisfies LICQ, then the first order opt. conds hold at $\hat{x}$.

19.3 Algorithmic development

* Modify to solve <u>non-convex</u>, <u>non-linear</u> problems from <u>any</u> initial estimate ...

Rewrite $(\nabla^2 \mathcal{L})\, \underline{p} = -\nabla \mathcal{L}$ by:

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & 0 & A_E^T(x) & A_I^T(x) \\ 0 & \Sigma & 0 & -I \\ A_E(x) & 0 & 0 & 0 \\ A_I(x) & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_s \\ -p_y \\ -p_z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A_E^T(x)y - A_I^T(x)z \\ z - \mu S^{-1}e \\ c_E(x) \\ c_I(x) - s \end{bmatrix}$$

<u>Where</u>: $\Sigma = S^{-1}Z$, the <u>primal-dual</u> system which allows a <u>symmetric</u> solver.

<u>Primal vs primal-dual system</u>

Starting from opt conds can yield:

$\Sigma = \mu S^{-2}$, the <u>primal system</u>

... prefer <u>primal-dual</u>!

Solving the primal-dual system:

- Solve by Ch 16 methods

Problems to watch for:

- elements of $\Sigma \to \infty$ (can be handled)
- Avoid: ① s, z very close to zero
  ② $\nabla^2_{xx} \mathcal{L}$, A_E must maintain full-rank
- Safest: → Direct linear algebra methods
  → Pre-conditioning to cluster eigenvalues

Pre-condition:

Good: $\boxed{\tilde{p}_s = S^{-1} p_s} \Rightarrow \boxed{p_s = S \tilde{p}_s}$ ~ to substitute

and multiply 2nd eqn by S to get:

$$S \Sigma p_s = -S(z - \mu S^{-1} e)$$

$$\Rightarrow S \Sigma S \tilde{p}_s = -Sz + \mu e$$

As $\mu \to 0$, get $SZ \approx \mu I$, $S \Sigma S$ approx μI

Perfect: $\boxed{\tilde{p}_s = \Sigma^{1/2} p_s}$ ~ best pre-conditioner

Iterative approaches:

* GMRES, QMR, LSQR directly
  in Golub & Van Loan

Effective approaches:

Iterative + works!

① Use a null-space approach to Solve the primal-dual system

and

② Apply apply projected CG given in algorithm 16.2

Updating the barrier parameter

Fiacco-McCormick approach in algorithm 19.1:

$$\mu_{k+1} = \sigma_k \mu_k, \quad \sigma_k \in (0,1)$$

$$\sigma_k = \begin{cases} 0.1, & \text{when making significant progress} \\ 0.2, & \text{when not} \end{cases}$$

Get superlinear convergence for:

$\sigma_k \to 0$ and $\tau \to 1$

Choose good x_0, μ_0, and Scaling.

Adaptive strategies (better).

$$\mu_{k+1} = \sigma_k \frac{s_k^T z_k}{m}$$

with:

$$\sigma_k = 0.1 \min\left(0.05 \frac{1-\xi_k}{\xi_k}, 2\right)^3$$

$$\xi_k = \frac{\min_i [s_k]_i [z_k]_i}{(s^k)^T z^k / m}$$

reduces μ_k significantly for $\xi_k \simeq 1$.

can also use predictor approaches.

Non-Convexity and Singularity

Modify $\nabla^2 f$ to:

$$\begin{bmatrix} (\nabla_{xx}^2 f + \delta I) & 0 & A_E(x)^T & A_I(x)^T \\ 0 & \Sigma & 0 & -I \\ A_E(x) & 0 & -\gamma I & 0 \\ A_I(x) & -I & 0 & 0 \end{bmatrix}$$

where γ, δ are chosen as given in the algorithm in appendix B.

Step acceptance: Merit functions and filters

* determine whether the step is sufficient to be accepted.
* Can use merit functions or filters.

Exact merit function:

$$\Phi_\nu(x,s) = f(x) - \mu \sum_{i=1}^{m} \log(s_i) + \nu\|C_E(x)\| + \nu\|C_I(x) - s\|$$

Here, ν is a constant multiplying l_1 or l_2 norm.

* $\nu > 0$ is a penalty parameter as given in algorithm 18.5 (p. 554).

* note that we have:

$$\alpha_s \in (0, \alpha_s^{max}), \quad \alpha_z \in (0, \alpha_z^{max}]$$

for the search space for:

$$x^+ = x + \alpha_s p_x, \quad s^+ = s + \alpha_s p_s,$$
$$y^+ = y + \alpha_z p_y, \quad z^+ = z + \alpha_z p_z$$

The idea is to adjust α_s, α_z to ensure sufficient decrease in Φ_ν.

* Recall possible "Maratos effect" (non-decreasing ϕ may be needed ...) (see ch. 15 end for soln)

The objectives for the filters will be:

$$\begin{cases} (i) \ f(x) - \mu \sum_{i=1}^{m} \log s_i \\ (ii) \ \| c_E(x), c_I(x) - s \| \end{cases}$$

* Also, a new p (also called step) will be accepted if better (not dominated) by anything we already have.

* Recall that we may need to employ a ~~phase~~ feasibility restoration phase where we restart the search based on the lowest value of $\| c_E(x), c_I(x) - s \|$.

* Again, we may have "Maratos effect" (see end of ch. 15)

Quasi-Newton Approximations

Basic-idea

Approximate $\nabla^2_{xx}\mathcal{L}$ with B, the Hessian.

* This is <u>not</u> L-BFGS! L-BFGS worked with $H = B^{-1}$. Here, we are working with B only.

* BFGS must be modified as given in algorithm 18.2, page 537, described by <u>damped BFGS</u> updating. ⇒ Rejects negative curvature updates.

* For all algorithms apply:

- Replace $s_k = x_{k+1} - x_k$ by $\Delta x_k = x_{k+1} - x_k$
- Replace $y_k = \nabla f_{k+1} - \nabla f_k$ by Δl_k where:

$$\Delta l_k = \nabla_x \mathcal{L}(x^+, s^+, y^+, z^+) - \nabla_x \mathcal{L}(x, s^+, y^+, z^+)$$

⇒ <u>NB:</u> Δl_k capture $\nabla_x \mathcal{L}$ change <u>after</u> s^+, y^+, z^+ are plugged in

⇒ It comes from finalized x^+, s^+, y^+, z^+.

From page 181, Thm 7.9, have:

$$S_k = \begin{bmatrix} \Delta x_0 & \Delta x_1 & \cdots & \Delta x_k \end{bmatrix} \quad \text{for } k < \hat{m}$$

After $k \geq \hat{m}$, use:

$$S_k = \begin{bmatrix} \Delta x_{k-m} & \cdots & \Delta x_{k-1} \end{bmatrix} \quad \text{of } n \times \hat{m} \text{ size.}$$

Similarly, for Y_k, afterwards, $k \geq \hat{m}$:

$$Y_k = \begin{bmatrix} \Delta l_{k-m} & \Delta l_{k-m+1} & \cdots & \Delta l_{k-1} \end{bmatrix} \quad \text{of } n \times \hat{m} \text{ size.}$$

Then:

$$B = \underbrace{\xi I}_{\substack{\xi > 0 \\ \text{for } B_0}} + W M W^T \quad \text{in ch. 19.}$$

From theorem 7.4:

$$B_k = \underbrace{\delta_k I}_{\Downarrow} - \underbrace{\begin{bmatrix} \delta_k S_k & Y_k \end{bmatrix}}_{\Downarrow} \underbrace{\begin{bmatrix} \delta_k S_k^T S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1}}_{\Downarrow} \underbrace{\begin{bmatrix} \delta_k S_k^T \\ Y_k^T \end{bmatrix}}_{\Downarrow}$$

$\underline{\underline{\text{to}}}$

$$B = \xi I + \underset{n \times (2\hat{m})}{W} \times \underset{(2\hat{m}) \times (2\hat{m})}{M} \times \underset{(2\hat{m} \times n)}{W^T}$$

On page 182, all of the terms need to be updated based on page 19.5 on how to replace s_k, y_k by Δx_k and Δl_k.

From problem 19.14, this representation is used to reformulate (19.12): $(\nabla^2 \mathcal{L}_k) P = -\nabla \mathcal{L}$ by:

$$\nabla^2 \mathcal{L}_{k+1} = \begin{bmatrix} \xi I & 0 & A_E^T & A_I^T \\ 0 & \Sigma & 0 & I \\ A_E & 0 & 0 & 0 \\ A_I & I & 0 & 0 \end{bmatrix} + \underbrace{\begin{bmatrix} W \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{is } U} \underbrace{\begin{bmatrix} MW^T & 0 & 0 & 0 \end{bmatrix}}_{\text{is } V^T}$$

This puts $\nabla^2 \mathcal{L}_k$ in the updating form of:

$$\nabla^2 \mathcal{L}_{k+1} = \nabla^2 \mathcal{L}_k + UV^T$$ (see page 612)

In turn, this allows the application of the Sherman-Morrison-Woodbury formula:

$$\left(\nabla^2 \mathcal{L}_{k+1}\right)^{-1} = \left(\nabla^2 \mathcal{L}_k\right)^{-1} - \left(\nabla^2 \mathcal{L}_k\right)^{-1} U \left(I + V^T \left(\nabla^2 \mathcal{L}_k\right)^{-1} U\right)^{-1} \times V^T \times \left(\nabla^2 \mathcal{L}_k\right)^{-1}$$

Thus, the difficulty comes from having to work with the inverses.

Problem 19.14 shows:

$$\nabla^2 \mathcal{L}_k = C = \begin{bmatrix} D & A^T \\ A^T & 0 \end{bmatrix} \text{ with } D = \begin{bmatrix} \zeta I & 0 \\ 0 & \Sigma \end{bmatrix}, \; A = \begin{bmatrix} A_E & 0 \\ A_I & I \end{bmatrix}$$

so that to compute the step, we need

$$\left(C + UV^T\right) \underbrace{r}_{\text{step}} = - \underbrace{s}_{\nabla \mathcal{L}}$$

see

compact

which is

$$r = -\left(C + UV^T\right)^{-1} = -\left(C^{-1} - C^{-1} U \left(I + V^T C^{-1} V\right)^{-1} V^T C^{-1}\right) s$$

with:

$$C^{-1} = \left.\begin{bmatrix} D^{-1} - D^{-1} A^T \left(A D^{-1} A^T\right)^{-1} A D^{-1} & D^{-1} A^T \left(A D^{-1} A^T\right)^{-1} \\ \left(A D^{-1} A^T\right) A D^{-1} & -\left(A D^{-1} A^T\right)^{-1} \end{bmatrix}\right\} \text{compact}$$

Compact

BFGS is supposedly used in

KNITRO/CG (free download for students)
Source code?

*L-BFGS-B does simple bounds and is also free for download, <u>with</u> source code.

Feasible Interior-point Methods

Suppose that we <u>must</u> satisfy some constraints C_{I_s}.

Basic idea:

<u>Set</u> $\boxed{s^+ \leftarrow c_{I_s}(x^+)}$

and test (x^+, s^+) if is acceptable for merit function ϕ.

If not, reject and pick "shorter" p.

For trust-region methods, you would also reduce the trust-region width.

Note that: * $c_i(x^+) \leq 0 \Rightarrow \phi \rightarrow \infty$

* Also, $x + p_s$ close to bdry is rejected since $-\log(s_i) \rightarrow +\infty$ for $s_i \rightarrow 0$.

Review of directional derivatives

(A.51): $$D(f(x); p) \overset{\text{def}}{=} \lim_{\epsilon \to 0} \frac{f(x+\epsilon p) - f(x)}{\epsilon}$$

will work even if f is <u>not</u> continuously differentiable.

For the l_1-norm:

$$D(\|x\|_1; p) = \lim_{\epsilon \to 0} \frac{\|x+\epsilon p\|_1 - \|x\|_1}{\epsilon}$$

$$= \lim_{\epsilon \to 0} \frac{\sum_{i=1}^{n} |x_i + \epsilon p_i| - \sum_{i=1}^{n} |x_i|}{\epsilon}$$

For $x_i > 0$, $x_i = 0$, $x_i < 0$, we get (for each component):

$$|x_i + \epsilon p_i| = \begin{cases} |x_i| + \epsilon p_i, & x_i > 0 \\ \epsilon |p_i|, & x_i = 0,\ \epsilon > 0 \\ |x_i| - \epsilon p_i, & x_i < 0 \end{cases}$$ for ϵ sufficiently small.

From:

$$|x_i + \epsilon p_i| = \begin{cases} x_i + \epsilon p_i & \text{if } x_i + \epsilon p_i > 0 \\ -x_i - \epsilon p_i & \text{if } x_i + \epsilon p_i < 0 \\ |\epsilon p_i| & \text{for } x_i = 0 \end{cases}$$

Here, $-x_i = |x_i|$

This gives:

$$D(\|x\|_1; p) = \sum_{i \mid x_i < 0} -p_i + \sum_{i \mid x_i > 0} p_i + \sum_{i \mid x_i = 0} |p_i|$$

for any x and p.

However, ∇f, $f(x) = \|x\|_1$, does not exist if any one $x_i = 0$:

If f is contly diff'ble, then

$$\nabla f(x)^T p = D(f(x); p)$$

will work for $f(x) = \|x\|_1$, $x_i > 0$.

19.4 A line-search interior-point method

Algorithm 19.2 will work but needs help with global convergence. On page 589, we have:

* monitor α_s, α_z to stay "large enough"
  If (α_s, α_z are small) and we are using the filter method then
    Apply feasibility restoration stage.
  end

* Another approach is to run a trust region step when this happens.
* The problem comes from the fact that $\alpha_s, \alpha_z \to 0$ at non-stationary points.
* This will not happen with trust-region methods.

⇒ We will focus on trust-region methods and skip algorithm 19.2.

19.5 A trust-region interior-point method

* more complex but works great!

An algorithm for solving the barrier problem.

* Use specialized SQP.
* A sequence of steps leads to (19.39). (bottom-up)

We want to solve the normal subproblem:

$$\min_{v=(v_x,v_s)} \|A_E(x)v_x + c_E(x)\|_2^2 + \|A_I(x)v_x - S v_s + (c_I(x)-s)\|_2^2$$

$$\text{subject to: } \begin{cases} \|(v_x, v_s)\|_2 \leq 0.8\Delta \\ v_s \geq -(\tau/2)e. \end{cases}$$

To solve this, please note:

* $v_x = (v_{x1}, \ldots, v_{xn})^T$ and $v_s = (v_{s1}, \ldots v_{sm})$ number of Ineq. Constr.

make the $\|\cdot\|_2^2$ terms easy!

Recall: $\left\|\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}\right\|_2^2 = v_1^2 + v_2^2$

For the gradients ∇_v, ∇_v^2, we only need to worry about terms like $(av_1+b)^2$ with $\left[(av_1+b)^2\right]' = 2(av_1+b).a$. Easy!

Next, ignore $v_s \geq -(\tau/2)\,e$ initially:

Step 1. Solve the trust-region problem using std methods (eg: dogleg):

$$\min_{v=(v_x,v_s)} \|A_E(x)v_x + c_E(x)\|_2^2 + \|A_I(x)v_x - Sv_s + (c_I(x) - s)\|_2^2$$

subject to: $\|(v_x, v_s)\|_2 \leq 0.8\Delta$

Step 2. Check if $v_s \geq -(\tau/2)\,e$ is satisfied. If not, then backtrack to make sure $v_s \geq -(\tau/2)e$ is satisfied.

Your textbook is not clear how to backtrack! Using step-length ideas, we can find α^* so that $\alpha^* v_s \geq -(\tau/2)e$ and pick α^* closer to 1.

We then form the residuals:

$$r_E = A_E(x) v_x + c_E(x)$$

$$r_I = A_I(x) v_x - S v_s + (c_I(x) - s)$$

Also, define the projection matrix:

$$P = \begin{bmatrix} I & \hat{A}^T \\ \hat{A} & 0 \end{bmatrix} \sim (*)$$

<u>with:</u> $$\hat{A} = \begin{bmatrix} A_E(x) & 0 \\ A_I(x) & -S \end{bmatrix}$$

Reformulate (19.33) as a QP to be solved by Algorithm 16.2.

Recall the basic algorithm is to solve (p. 451):

$$\min_x \; q(x) \stackrel{\text{def}}{=} \tfrac{1}{2} x^T G x + x^T c$$

$$\text{subject to: } Ax = b$$

Here, we want to apply this algorithm to:

$$\min_{p_x, \tilde{p}_s} \; \nabla f^T p_x + \tfrac{1}{2} p_x^T \nabla_{xx}^2 \mathcal{L}\, p_x - \mu e^T \tilde{p}_s + \tfrac{1}{2} \tilde{p}_s^T S \Sigma S \tilde{p}_s$$

Subject to:

$$A_I(x)p_x - S\tilde{p}_s + (c_I(x) - s) = r_I$$

$$A_E(x)p_x + c_E(x) = r_E$$

After forming G, c, A, "x" $= \begin{bmatrix} p_x \\ \tilde{p}_s \end{bmatrix}$ from this problem, with p given in ⊛, we run the Projected CG method to generate "x" $= \begin{bmatrix} p_x \\ \tilde{p}_s \end{bmatrix}$, where we terminate the projected CG by following Steihaug's rules: <u>Stop if:</u>

* $\|(p_x, \tilde{p}_s)\| \leq \Delta$ is about to be violated. We are increasing $\|(p_x, \tilde{p}_s)\|$ as we go.
* Negative curvature is detected or
* Solution is satisfactory (i.e. $r \approx 0$)

After this, return to (19.32):

$$\tilde{p} = \begin{bmatrix} p_x \\ \tilde{p}_s \end{bmatrix} = \begin{bmatrix} p_x \\ S^{-1}p_s \end{bmatrix}$$ to solve for p_s.

Lagrange multipliers estimates and step acceptance

For fixed (x,s), we choose the Lagrange multipliers using:

$$\begin{bmatrix} y \\ z \end{bmatrix} = (\hat{A}\hat{A}^T)^{-1}\hat{A}\begin{bmatrix} \nabla f(x) \\ -\mu e \end{bmatrix}$$ "Least-squares estimate"

with:

$$\hat{A} = \begin{bmatrix} A_E(x) & 0 \\ A_I(x) & -S \end{bmatrix}$$

To enforce the required positivity:

$$z_i \leftarrow \min(10^{-3}, \mu/s_i), \quad i=1,2,\dots,m.$$

To determine if we can accept the generated step, define:

$$\text{ared}(p) = \phi_\nu(x,s) - \phi_\nu(x+p_x, s+p_s) > 0$$

$$\text{pred}(p) = q_\nu(0) - q_\nu(p)$$

with:

$$q_\nu(p) = \nabla f^T p_x + \tfrac{1}{2} p_x^T \nabla_{xx}^2 \mathcal{L}\, p_x - \mu e^T S^{-1} p_s + \tfrac{1}{2} p_s^T \Sigma p_s + \nu m(p)$$

and

$$m(p) = \left\| \begin{bmatrix} A_E(x) p_x + c_E(x) \\ A_I(x) p_x - p_s + c_I(x) - s \end{bmatrix} \right\|_2$$

Here, ν is a penalty parameter that needs to be set so as to require:

$$\text{pred}(\underline{p}) \geq \rho\nu\left(m(0) - m(\underline{p})\right), \text{ some } \rho \in (0,1)$$

So, we want:

$$q_\nu(0) - q_\nu(\underline{p}) \geq \rho\nu\left(m(0) - m(\underline{p})\right) \sim (**)$$

To guarantee (**), <u>if</u> it was satisfied in the previous iteration, then bring ν from that iteration. Else, by (15.36), we need:

$$\nu \geq \frac{\nabla f^T \underline{p}_x + (\sigma/2)\, \underline{p}_x^T \nabla_{xx}^2 \mathcal{L}\, \underline{p}_x}{(1-\rho)\left\| \begin{bmatrix} c_E(x) \\ c_I(x) - s \end{bmatrix} \right\|}$$

to be satisfied by a margin.

Plug-in (**) to verify.

We then require

$$\boxed{\text{ared}(\underline{p}) \geq \eta\, \text{pred}(\underline{p})} \text{ — } (\Delta)$$

with $\eta = 10^{-8}$ (approx.).

If (Δ) is not satisfied, we reject $\underline{p}$.

Algorithm 19.4 (Trust-Region Interior-Point)

Choose $\eta > 0$, $\tau \in (0,1)$, $\sigma \in (0,1)$, $\zeta \in (0,1)$

Set tolerances: ϵ_μ, ϵ_{TOL}:

If using Quasi-Newton, select initial B_0 (symmetric)

Choose $\mu > 0$, $x_0, s_0 > 0$, Δ_0

$k \leftarrow 0$

Final condition ($\mu = 0$)

Repeat until $E(x_k, s_k, y_k, z_k; 0) \leq \epsilon_{TOL}$

Repeat until $E(x_k, s_k, y_k, z_k; \mu) \leq \epsilon_\mu$

Compute y, z using (19-26) (19.37-38 in book)

Compute $\nabla^2_{xx} \mathcal{L}(x_k, s_k, y_k, z_k)$ or update quasi-Newton approximation B_k, with $\Sigma_k = S^{-1} Z$

Compute $v_k = (v_x, v_s)$ see pages (19-22)-(19-23)

Compute $\hat{p}_k$ by projected CG method

Set $p_s = S \tilde{p}_s$

Update the penalty ν_k to satisfy: $\text{pred}(p) \geq \rho \nu (m(0) - m(p))$

Compute $\text{pred}_k(p_k)$, $\text{ared}_k(p_k)$

<u>If</u> $ared_k(p_k) \geq \eta\, pred_k(p_k)$

$$\left.\begin{array}{l} \underline{\text{Set}}\ x_{k+1} \leftarrow x_k + p_x \\ \underline{\text{Set}}\ s_{k+1} \leftarrow s_k + p_s \\ \text{Choose}\ \Delta_{k+1} \geq \Delta_k \end{array}\right\} \text{success}$$

<u>Else</u>

$$\left.\begin{array}{l} \underline{\text{Set}}\ x_{k+1} = x_k \\ \underline{\text{Set}}\ s_{k+1} = s_k \\ \text{Choose}\ \Delta_{k+1} < \Delta_k \end{array}\right\} \text{Failed}$$

<u>endif</u>

$k \leftarrow k + 1$

<u>end</u> (inside loop)

<u>Set</u> $\mu \leftarrow \sigma\mu$, update ϵ_μ.

<u>end</u> (outside loop)

We can stop for $\epsilon_\mu = \mu$

<u>Notes</u>: * This is KNITRO/CG

* May need to address Maratos effect.

P589: Global Convergence of the trust-region approach

Thm 19.2 Suppose that the algorithm is applied (trust-region inner loop only), μ is fixed, $\epsilon_\mu = 0$. Suppose $\{f_k\}$ is bounded below and $\{\nabla f_k\}, \{c_k\}, \{A_k\}, \{B_k\}$ are bounded.

Then, one of the following three happens:

(I) $\{x_k\}$ is not asymptotically feasible
Then, $A_k c_k^- \to 0$ and $\nu_k \to \infty$.

(II) $\{x_k\}$ is asymptotically feasible, but $\{(c_k, A_k)\}$ has $(\bar{y}, \bar{A})$ limit point failing linear independence. Here, $\nu_k \to \infty$ also.

(III) $\{x_k\}$ is asymptotically feasible, and all limit points $\{(c_k, A_k)\}$ satisfy the linear independence constraint qualification $\nu_k \to$ constant, $c_k \geq 0$ for large k and the stationarity conditions are satisfied in the limit.

Proven for $\Sigma = \mu S^{-2}$ in [48]

Basic idea: Monitor ν_k for convergence.

Also, recall theorem 19.1 talks about convergence of the basic method.

19.8 Superlinear Convergence

* decrease μ, decrease ϵ_μ, $\tau \to 1$ sufficiently rapidly.

For line-search:

$$\epsilon_\mu = \theta\mu, \quad \epsilon_{\mu^+} = \theta\mu^+, \quad \theta > 0$$

Then:

$$\mu^+ = \mu^{1+\delta}, \quad \delta \in (0,1)$$
$$\tau = 1 - \mu^\beta \text{ with } \beta > \delta.$$

Do not decrease μ faster than quadratic.

If $\mu^+ = \sigma\mu$, $\sigma \in (0,1)$ gives linear convergence.

* Superlinear only achieved at the end iterations.

19.9 Perspectives and SW

* KNITRO / CG for trust-region algorithm.
  [50] for details on the algorithm.
* KNITRO advertized a lot.