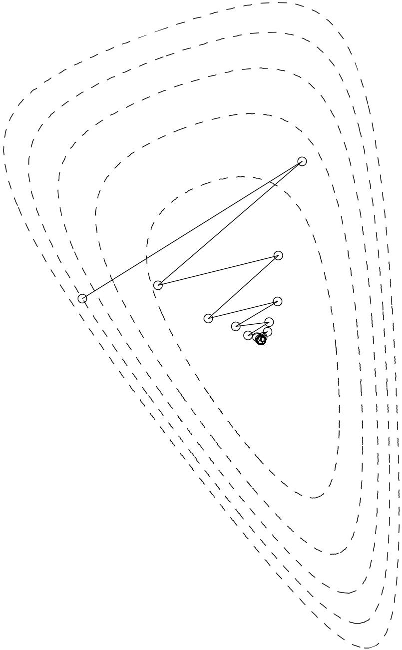


Optimization Algorithms

Appendix

Phase I Optimization, Bound Constraints, Primal-Dual Newton method

Marc Toussaint
Technical University of Berlin
Winter 2024/25

Phase I Optimization

Phase I: Finding a feasible initialization

- We might not have a feasible $x \in \mathbb{R}^n$ to initialize the NLP solver
  - No issue for squared penalty and AugLag
  - Also primal-dual can be ok (although it is usually realized as interior point method)
  - LogBarrier requires feasible initialization (e.g., also within SQP)

- *Phase I Optimization* means finding a feasible initial x by solving another optimization problem

Phase I: formulation to minimize infeasibility

- Standard approach: introduce single or multiple variables of infeasibility
- Single (maximum) infeasibility variable

$$\min_{(x,s)\in\mathbb{R}^{n+1}} s \ \text{ s.t. } \ \forall_i: \ g_i(x) \le s, \ s \ge 0$$

  – Given initial infeasible x, initialize $s = \max_i g_i(x) > 0$

- Individual infeasibility variables

$$\min_{(x,s)\in\mathbb{R}^{n+m}} \sum_{i=1}^{m} s_i \ \text{ s.t. } \ \forall_i: \ g_i(x) \le s_i, \ s_i \ge 0$$

  – Given initial infeasible x, initialize $s_i = \max\{g_i(x), 0\}$

Bound Constraints

Bound Constraints

- A **bound constrained** NLP, with bounds $l, u \in \mathbb{R}^n$, $\quad l \leq u$

$$\min_{l \leq x \leq u} f(x) \;\; \text{s.t.} \;\; g(x) \leq 0,\; h(x) = 0$$

- Other words:
  - *simply constrained problem* or NLP with simple constraints (Bertsekas)
  - box or rectangle constraints

Bound Constraints

- A **bound constrained** NLP, with bounds $l, u \in \mathbb{R}^n, \quad l \le u$

$$\min_{l \le x \le u} f(x) \ \text{ s.t. } \ g(x) \le 0,\ h(x) = 0$$

- Other words:
  - *simply constrained problem* or NLP with simple constraints (Bertsekas)
  - box or rectangle constraints

- Since we know how to deal with constraints g, h, we only discuss:

$$\min_{l \le x \le u} f(x)$$

Bound Constraints – Motivation

- Do we need to handle them specially? Not necessarily
  - **Treat bounds just like any other inequality**
  - Sound, we know what we're doing – **recommended, if possible**

Bound Constraints – Motivation

- Do we need to handle them specially? Not necessarily
  - **Treat bounds just like any other inequality**
  - Sound, we know what we're doing – **recommended, if possible**

- However, **reasons to treat bounds directly:**
  - The primal-dual Newton method requires Newton steps that respect bounds
  - Sometimes undesirable to have an AugLag or LogBarrier with inner/outer loop, only to account for bounds
  - Simpler/more direct solutions to handling bounds other than general (non-linear) inequalities?

Bound Constraints – Motivation

- Do we need to handle them specially? Not necessarily
  - **Treat bounds just like any other inequality**
  - Sound, we know what we're doing – **recommended, if possible**

- However, **reasons to treat bounds directly:**
  - The primal-dual Newton method requires Newton steps that respect bounds
  - Sometimes undesirable to have an AugLag or LogBarrier with inner/outer loop, only to account for bounds
  - Simpler/more direct solutions to handling bounds other than general (non-linear) inequalities?

- Note: Naively clipping ("projecting") all queries in a line search can go badly wrong!

References

- Mainstream: Projected gradient (or rather "projected line search")
  - not focus here, mention briefly
  - (SLIDES) Leyffer, S. Bound Constrained Optimization - GIAN Short Course on Optimization: Applications, Algorithms, and Computation. 30.

- **Our focus:** Bound-constrained Newton method
  - Maintain the strength of Newon method as inner loop in AugLag, primal-dual, etc
  - D.P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. SIAM Journal on Control and Optimization 20, 221-246 (1982).
  - Facchinei, F., Júdice, J. & Soares, J. An active set Newton algorithm for large-scale nonlinear programs with box constraints. SIAM Journal on Optimization 8, 158–186 (1998).
  - Cheng, W., Chen, Z. & Li, D. An active set truncated Newton method for large-scale bound constrained optimization. Computers & Mathematics with Applications 67, 1016–1023 (2014).

Bound Constraints & Newton

- Recap basic Newton method:

Input: initial $x \in \mathbb{R}^n$, functions $f(x), \nabla f(x), \nabla^2 f(x)$, tolerance θ, parameters (defaults: $\varrho_\alpha^+ = 1.2, \varrho_\alpha^- = 0.5, \varrho_{\textsf{ls}} = 0.01, \lambda$)

initialize stepsize $\alpha = 1$, fixed damping λ
repeat
  compute δ to solve $(\nabla^2 f(x) + \lambda \mathbf{I})\, \delta = -\nabla f(x)$
  while $f(x + \alpha\delta) > f(x) + \varrho_{\textsf{ls}} \nabla f(x)^\top (\alpha\delta)$ **do** *// line search*
    $\alpha \leftarrow \varrho_\alpha^- \alpha$ *// decrease stepsize*
  end while
  $x \leftarrow x + \alpha\delta$ *// step is accepted*
  $\alpha \leftarrow \min\{\varrho_\alpha^+ \alpha, 1\}$ *// increase stepsize*
until $\|\alpha\delta\|_\infty < \theta$

- Naive approach: clipping: query $y = \text{clip}(x + \alpha\delta)$
  - with $\text{clip}(x) \equiv \min(\max(x, l), u)$ elem-wise

Bound Constraints & Newton

- Recap basic Newton method:

Input: initial $x \in \mathbb{R}^n$, functions $f(x), \nabla f(x), \nabla^2 f(x)$, tolerance θ, parameters (defaults: $\varrho_\alpha^+ = 1.2, \varrho_\alpha^- = 0.5, \varrho_{\text{ls}} = 0.01, \lambda$)

initialize stepsize $\alpha = 1$, fixed damping λ
repeat
  compute δ to solve $(\nabla^2 f(x) + \lambda \mathbf{I})\, \delta = -\nabla f(x)$
  while $f(x + \alpha\delta) > f(x) + \varrho_{\text{ls}} \nabla f(x)^\top (\alpha\delta)$ **do** *// line search*
    $\alpha \leftarrow \varrho_\alpha^- \alpha$ *// decrease stepsize*
  end while
  $x \leftarrow x + \alpha\delta$ *// step is accepted*
  $\alpha \leftarrow \min\{\varrho_\alpha^+ \alpha, 1\}$ *// increase stepsize*
until $\|\alpha\delta\|_\infty < \theta$

- Naive approach: clipping: query $y = \text{clip}(x + \alpha\delta)$
  - with $\text{clip}(x) \equiv \min(\max(x, l), u)$ elem-wise
- Can go badly wrong – understanding why and when is the key to do it properly

Example

- Core case to consider (from Bertsekas):

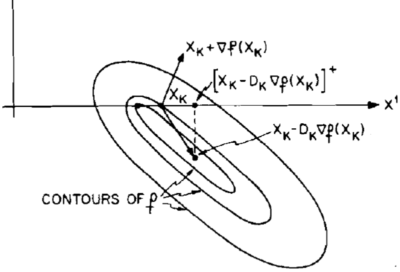


FIG. 1

- Example problem: $x \in \mathbb{R}^2$

$$\min \frac{1}{2} x^\top A x \;\; \text{s.t.} \;\; x_1 \geq \frac{1}{2} \,, \quad \text{with } A = \begin{pmatrix} 200 & -160 \\ -160 & 200 \end{pmatrix}$$

Example

- Core case to consider (from Bertsekas):

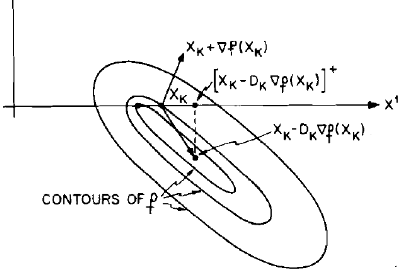


FIG. 1

- Example problem: $x \in \mathbb{R}^2$

$$\min \frac{1}{2} x^\top A x \ \text{ s.t. } \ x_1 \ge \frac{1}{2} \,, \quad \text{with } A = \begin{pmatrix} 200 & -160 \\ -160 & 200 \end{pmatrix}$$

- The standard Newton direction is bad! Naively clipping (projecting line search queries) sends in the wrong direction!

Active Set Identification

- The key is to (try to) identify the active set!
  - This is consistent to our general understanding of the complexity of constrained optimization: If the active inequalities were known apriori, everything would be much simpler! (Recall complexity of Simplex.) This is the same for the simple bound inequalities.
  - For general inequalities, we had the LogBarrier relaxing the hard decision of active constraints, and AugLag using the indicator $[g_i(x) \geq 0 \vee \lambda_i > 0]$
- Bertsekas proposes to define the active set as:

$$I^+(x) = \{i : \ 0 \geq x_i \geq \epsilon, \nabla f_i(x) \geq 0\}$$

(where he assumes $l = 0$, i.e., $x \geq 0$ as bounds)

- Facchinei proposes:

$$L(x) := \{i : \ x_i \leq l_i + a_i(x) \nabla f_i(x)\} \tag{1}$$

$$U(x) := \{i : \ xi \geq u_i + b_i(x) \ \nabla f_i(x)\} \tag{2}$$

Hessian Modification for Active Set

- Assuming we had the active set identified, how can we modify the Newton method?

Hessian Modification for Active Set

- Assuming we had the active set identified, how can we modify the Newton method?

- (Active variables could be hard-assigned to bound.)
- We compute Newton step only for the free variables!
  - The free variables form a *hyperplane* – we want a Newton step only in this hyperplane
  - Following Bertsekas: Let H be the original Hessian, we **delete correlations** of active bound variables to free variables, by **deleting off-diagonal** entries for the active variables

$$H \leftarrow \text{remove}_i(H) \quad : \quad \begin{pmatrix} A & \vdots & B \\ \cdots & h_{ii} & \cdots \\ B^\top & \vdots & C \end{pmatrix} \leftarrow \begin{pmatrix} A & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} & B \\ 0 \cdots 0 & h_{ii} & 0 \cdots 0 \\ B^\top & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} & C \end{pmatrix}$$

The curvature along i remains, but it becomes decorrelated from all other variables

Newton method with Bound Constraints

Input: initial $x \in \mathbb{R}^n$, functions $f(x), \nabla f(x), \nabla^2 f(x)$, bounds l, u, parameters $\theta, \varrho_\alpha^+, \varrho_\alpha^-, \varrho_{\text{ls}}, \lambda$

initialize stepsize $\alpha = 1$, fixed damping λ

$x \leftarrow \text{clip}(x)$ *// otherwise the first $\nabla f(x), \nabla^2 f(x)$ are horribly wrong*

repeat

  compute $g \leftarrow \nabla f(x), H \leftarrow \nabla^2 f(x)$

  Identify $I = \{i : (x = l \wedge g_i > 0) \vee (x = u \wedge g_i < 0)\}$ *// no ϵ; assume previous clip*

  $H \leftarrow \text{remove}_I(H)$ *// delete correlations*

  compute δ to solve $(H + \lambda \mathbf{I})\, \delta = -g$

  while $f(y) > f(x) + \varrho_{\text{ls}} \nabla f(x)^\top (y - x)$, for $y = \text{clip}(x + \alpha\delta)$, **do** *// line search*

    $\alpha \leftarrow \varrho_\alpha^- \alpha$ *// decrease stepsize*

  end while

  $x \leftarrow y$ *// step is accepted*

  $\alpha \leftarrow \min\{\varrho_\alpha^+ \alpha, 1\}$ *// increase stepsize*

until $\|\alpha\delta\|_\infty < \theta$

– since we clip within line search, clipped x_i are exactly on bound and identified in next iteration

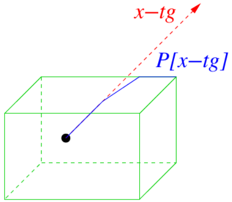
– δ can point away from bound (depending on g_i only), to free a previously bound x_i

- Line search sometimes an issue, when bound variable was not yet identified
- Facchinei mentions a “nonmonotone stabilization technique proposed in [27]”, which seems very interesting alternative to naive Wolfe in bound-constrained case!

Projected-Gradient Methods

- Nice tutorial reference:
  - (SLIDES) Leyffer, S. Bound Constrained Optimization - GIAN Short Course on Optimization: Applications, Algorithms, and Computation. 30.

- Let $\delta = -\nabla f(x)$ (gradient directly)
  - Consider the full line (infinite half-line) projected (clipped)
  - Identify the piece-wise linear pieces of this path
  - Find minimizer along this full path



Primal-Dual interior-point Newton Method

Primal-Dual interior-point Newton Method

- In the unconstraint case, Newton methods find a point x for which $\nabla f(x) = 0$
- The KKT conditions generalize the condition $\nabla f(x) = 0$ to the constraint case, and can be interpreted as saddle point conditions $L(x, \kappa, \lambda)$
- We think of the KKT conditions as an equation system $r(x, \kappa, \lambda) = 0$, and use a Newton method for solving it

- This leads to a **primal-dual** algorithm that adapts (x, κ, λ) concurrently.
  The Newton steps are done in the $(x, \kappa, \lambda) \in \mathbb{R}^{n+l+m}$ space.

Primal-Dual interior-point Newton Method

- We consider the KKT equation system

$$\begin{aligned}\nabla f(x) + \lambda^\top \tfrac{\partial}{\partial x} g(x) + \kappa^\top \tfrac{\partial}{\partial x} h(x) &= 0 \\ h(x) &= 0 \\ \mathrm{diag}(\lambda) g(x) + \mu \mathbf{1}_m &= 0\end{aligned}$$

  - With the 1st, 2nd, and *relaxed* 4th KKT condition
  - The ineq feasibility $g(x) \le 0$ and $\lambda \ge 0$ is implicit.

- We re-write this as

$$r(x,\kappa,\lambda) = 0 \;, \quad r(x,\kappa,\lambda) \overset{\text{def}}{=} \begin{pmatrix} \nabla[f(x) + \lambda^\top g(x) + \kappa^\top h(x)] \\ h(x) \\ \mathrm{diag}(\lambda)\; g(x) + \mu \mathbf{1}_m \end{pmatrix}$$

Primal-Dual interior-point Newton Method

- We compute the regularized Newton step δ in (x, κ, λ)-space as

$$\delta = -[\tfrac{\partial}{\partial_{x\kappa\lambda}} r(x, \kappa, \lambda) + \hat{\lambda}\mathbf{I}]^{\text{-1}}\ r(x, \lambda)$$

- With the **KKT Jacobian** $\frac{\partial}{\partial_{x\kappa\lambda}} r \in \mathbb{R}^{(n+l+m)\times(n+l+m)}$ replacing the role of the Hessian:

$$\frac{\partial}{\partial_{x\kappa\lambda}} r(x, \kappa, \lambda) = \begin{pmatrix} \nabla^2[f(x) + \lambda^\top g(x) + \kappa^\top h(x)] & \frac{\partial}{\partial x} h(x)^\top & \frac{\partial}{\partial x} g(x)^\top \\ \frac{\partial}{\partial x} h(x) & 0 & 0 \\ \mathrm{diag}(\lambda)\ \frac{\partial}{\partial x} g(x) & 0 & \mathrm{diag}(g(x)) \end{pmatrix}$$

Primal-Dual interior-point Newton Method

- We compute the regularized Newton step δ in (x, κ, λ)-space as

$$\delta = -[\tfrac{\partial}{\partial_{x\kappa\lambda}} r(x,\kappa,\lambda) + \hat{\lambda}\mathbf{I}]^{\text{-1}}\ r(x,\lambda)$$

- With the **KKT Jacobian** $\frac{\partial}{\partial_{x\kappa\lambda}} r \in \mathbb{R}^{(n+l+m)\times(n+l+m)}$ replacing the role of the Hessian:

$$\frac{\partial}{\partial_{x\kappa\lambda}} r(x,\kappa,\lambda) = \begin{pmatrix} \nabla^2[f(x) + \lambda^\top g(x) + \kappa^\top h(x)] & \frac{\partial}{\partial x}h(x)^\top & \frac{\partial}{\partial x}g(x)^\top \\ \frac{\partial}{\partial x}h(x) & 0 & 0 \\ \mathrm{diag}(\lambda)\ \frac{\partial}{\partial x}g(x) & 0 & \mathrm{diag}(g(x)) \end{pmatrix}$$

- Pseudo code $\rightarrow$ just like Newton method, but with δ as above

Primal-Dual interior-point Newton Method

- The method uses the Hessians $\nabla^2 f(x), \nabla^2 g_i(x), \nabla^2 h_j(x)$
  - One can approximate the constraint Hessians $\nabla^2 g_i(x), \nabla^2 h_j(x) \approx 0$
  - Gauss-Newton approximation: $f(x) = \phi(x)^\top \phi(x)$ only requires $\nabla\phi(x)$

- *No need for nested iterations, as with penalty/barrier methods!*

- The above formulation allows for a duality gap μ
  - Choosing $\mu = 0$ is not robust
  - We adapt μ on the fly, before each Newton step:
  - First evaluate the current duality measure $\tilde\mu = -\frac{1}{m} \sum_{i=1}^m \lambda_i g_i(x)$, then choose $\mu = \frac{1}{2}\tilde\mu$ to half that
  - See also Boyd sec 11.7.3.

- The dual feasibility $\lambda_i \geq 0$ needs to be handled explicitly by the root finder!
  - Specialized method for bound-constrained optimization