# CS 450: Numerical Anlaysis

## Lecture 17

## Chapter 6 Numerical Optimization

## Constrained Optimization and Quadratic Programming

Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

March 14, 2018

# Constrained Optimization Problems

- We now return to the general case of *constrained* optimization problems:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0} \quad \text{and} \quad \boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$$

  *When $f$ is quadratic, while $h$, $g$ is linear, this is a quadratic optimization problem.*

- Generally, we will seek to reduce constrained optimization problems to a series of unconstrained optimization problems:
  - *sequential quadratic programming: solve an unconstrained quadratic program at each iteration,*
  - *penalty-based methods: solve a series of more complicated (more ill-conditioned) unconstrained optimization problems,*
  - *active set methods: define sequence of optimization problems with inequality constrained ignored or treated as equality constraints.*

# Lagrangian Duality

- The Lagrangian function with constraints $\boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}$ and $\boldsymbol{h}(\boldsymbol{x}) \leq \mathbf{0}$ is

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \lambda^T \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}) \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix}$$

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}_1^T \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{\lambda}_2^T \boldsymbol{g}(\boldsymbol{x})$$

- The Lagrangian dual problem is an unconstrained optimization problem:

$$\max_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda}), \quad q(\boldsymbol{\lambda}) = \begin{cases} \min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) & \text{if } \boldsymbol{\lambda} \geq \mathbf{0} \\ -\infty & \text{otherwise} \end{cases}$$

*Note that the unconstrained optimality condition* $\nabla q(\boldsymbol{\lambda}^*) = \mathbf{0}$, *implies*

$$\max \left( \boldsymbol{\lambda}^*, \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}) \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix} \right) = \mathbf{0}$$

*when* $\lambda_i^* = 0$, *we say the* $i$*th constraint is inactive.*

# Constrained Optimality

- In equality-constrained optimization $\boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}$, minimizers $\boldsymbol{x}^*$ are on the border of the feasible region (set of points satisfying constraints), in which case we must ensure any direction of decrease of $f$ from $\boldsymbol{x}^*$ leads to an infeasible point, which gives us the condition:

$$\exists \boldsymbol{\lambda} \in \mathbb{R}^n, \quad -\nabla f(\boldsymbol{x}^*) = \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}^*)\boldsymbol{\lambda}$$

*$\boldsymbol{\lambda}$ are referred to as the Lagrange multipliers.*

- Seek critical points in the Lagrangian function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$, described by the nonlinear equation,

$$\nabla \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x})\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix} = \mathbf{0}$$

*Seeking $\boldsymbol{\lambda}$ that maximizes the global minimum of $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ defines the dual optimization problem.*

# Sequential Quadratic Programming

- *Sequential quadratic programming (SQP)* corresponds to using Newton's method to solve the nonlinear equations,

$$\nabla \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x})\boldsymbol{\lambda} \\ \boldsymbol{g}(\boldsymbol{x}) \end{bmatrix} = \boldsymbol{0}$$

*At each iteration, we compute* $\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{\lambda}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{\lambda}_k \end{bmatrix} + \boldsymbol{s}_k$ *by solving*

$$\boldsymbol{J}_{\mathcal{L}}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)\boldsymbol{s}_k = -\nabla \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) = -\begin{bmatrix} \nabla f(\boldsymbol{x}_k) + \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}_k)\boldsymbol{\lambda}_k \\ \boldsymbol{g}(\boldsymbol{x}_k) \end{bmatrix}$$

*where*

$$\boldsymbol{J}_{\mathcal{L}}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) = \begin{bmatrix} \boldsymbol{B}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) & \boldsymbol{J}_g^T(\boldsymbol{x}_k) \\ \boldsymbol{J}_g(\boldsymbol{x}_k) & \boldsymbol{0} \end{bmatrix} \quad \textit{with} \quad \boldsymbol{B}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{H}_f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \boldsymbol{H}_{g_i}(\boldsymbol{x})$$

# Quadratic Programming Problems

- An equality-constrained quadratic programming problem has the form

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

*Its first-order optimality condition in the unconstrained case is*

$$\boldsymbol{0} = \nabla f(\boldsymbol{x}) = \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c} \quad \Rightarrow \quad \boldsymbol{Q}\boldsymbol{x} = -\boldsymbol{c}$$

*and in the constrained case, becomes*

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{b} \end{bmatrix}$$

*This allows us to observe, that at the $k$th iteration SQP solves a QP with $\boldsymbol{Q} = \boldsymbol{B}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$, $\boldsymbol{A} = \boldsymbol{J}_g(\boldsymbol{x}_k)$, $\boldsymbol{c} = \nabla f(\boldsymbol{x}_k) + \boldsymbol{J}_{\boldsymbol{g}}^T(\boldsymbol{x}_k)$, and $\boldsymbol{b} = \boldsymbol{g}(\boldsymbol{x}_k)$.*

# Steepest Descent for Quadratic Programming

- Near the minima, all smooth nonlinear programming problems look like quadratic programming problems, where $\boldsymbol{Q}$ converges to the Hessian at the minima, $\boldsymbol{H}_f(\boldsymbol{x}^*)$.
- Consequently, we can analyze local convergence of methods by considering their convergence for a QP, e.g. for steepest descent:

  *Assume $\boldsymbol{b} = \boldsymbol{0}$ and $\boldsymbol{c} = \boldsymbol{0}$, then we have $\nabla f(\boldsymbol{x}) = \boldsymbol{Q}\boldsymbol{x}$. Consequently, steepest descent will seek a step-size $\alpha_k$ to perform the step*

  $$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{Q}\boldsymbol{x}_k = (\boldsymbol{I} - \alpha_k \boldsymbol{Q})\boldsymbol{x}_k$$

  *for this fixed-point iteration to converge, it suffices that $\alpha_k < 2/\lambda_{max}(\boldsymbol{Q})$. The optimal value can be shown to be $\alpha^* = 2/(\lambda_{max}(\boldsymbol{Q}) + \lambda_{min}(\boldsymbol{Q}))$, leading to convergence rate*

  $$||\boldsymbol{e}_{k+1}|| = \frac{\kappa(\boldsymbol{Q}) - 1}{\kappa(\boldsymbol{Q}) + 1}||\boldsymbol{e}_k||$$

# Gradient Methods with Extrapolation

- We can improve the constant in the linear rate of convergence of steepest descent, by leveraging extrapolation methods, which consider two previous iterates (using *momentum*):

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$$

- The heavy ball, which uses constant $\alpha_k = \alpha$ and $\beta_k = \beta$, achieves better convergence than steepest descent:

$$||\boldsymbol{e}_{k+1}|| = \frac{\sqrt{\kappa(\boldsymbol{Q})} - 1}{\sqrt{\kappa(\boldsymbol{Q})} + 1} ||\boldsymbol{e}_k||$$

*Nesterov's gradient optimization method is another instance of an extrapolation method, and provides further optimality guarantees.*

# Conjugate Gradient Method

- The *conjugate gradient method* is capable of making the optimal choice of $\alpha_k$ and $\beta_k$ at each iteration of an extrapolation method:

$$(\alpha_k, \beta_k) = \operatorname*{argmin}_{\alpha_k, \beta_k} \left[ f\left( \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) \right) \right]$$

  *For quadratic programming problems, conjugate gradient is an optimal 1st order method, converging in $n$ iterations.*

- Generally conjugate gradient methods perform a sequence of line minimizations in $n$ directions that are $\boldsymbol{Q}$-orthogonal:

  *A parallel tangents implementation of the method proceeds as follows*

  - *Perform a step of steepest descent to generate $\hat{\boldsymbol{x}}_k$ from $\boldsymbol{x}_k$*
  - *Generate $\boldsymbol{x}_{k+1}$ by minimizing over the line passing through $\boldsymbol{x}_k$ and $\hat{\boldsymbol{x}}_k$*

  *Each conjugate direction is guaranteed to be $\boldsymbol{Q}$-orthogonal to previous directions by a recurrence argument similar to that of Lanczos iteration.*