Numerical Optimization

A Workshop
At
Department of Mathematics
Chiang Mai University
August 4-15, 2009

Instructor: **Vira Chankong**

Electrical Engineering and Computer Science
Case Western Reserve University

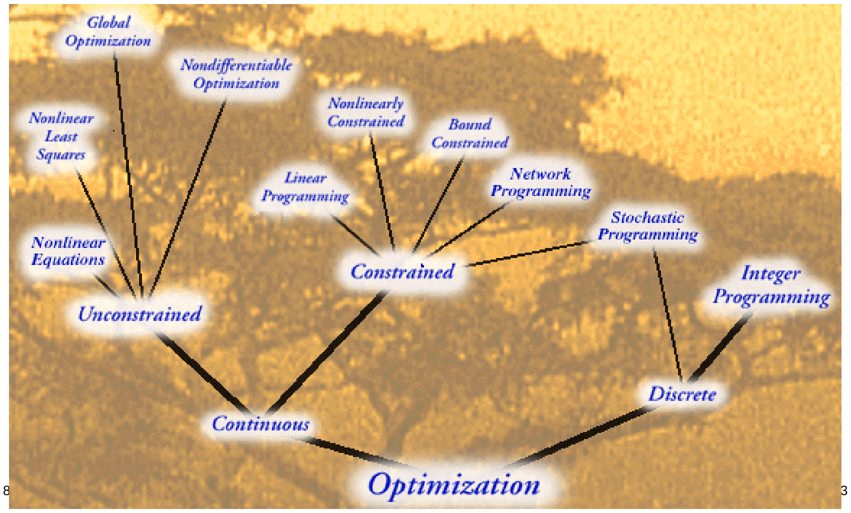
Phone: 216 368 4054, Fax: 216 368 3123
E-mail: **vira@case.edu**

Session:

Methods For Constrained Nonlinear Optimization Problems

Vira Chankong
Case Western Reserve University
Electrical Engineering and Computer Science

NEOS Guide Optimization Tree



Constrained Optimization (Nonlinear Programming)

NLP: $\min_{\mathbf{x} \in R^n} f(\mathbf{x})$

$$s.t. \quad h_j(\mathbf{x}) = 0, \quad j \in J_E$$

$$g_j(\mathbf{x}) \leq 0, \quad j \in J_I$$

or $\min_{\mathbf{x} \in R^n} f(\mathbf{x})$

$$s.t. \quad c_j(x) = 0, \quad j \in J$$

$$l_i \leq x_i \leq u_i, \quad i = 1, .., n$$

Desirable Properties of Numerical Methods

- **Terminate at a right solution, quickly and cheaply** every time
  - **Converge:** Find *solution* every time; Always stop at the right point
  - **Speed:** Find *solution* quickly (low # of iterations)
  - **Cheap: Low cost per iteration (time: # of function evaluations; and storage)**
- **Appropriate handling of optimality v.s. feasibility**

Optimality v.s. Feasibility

2 strategies:

- **Start feasible ($x^{(1)}$ feasible), stay feasible ($x^{(k)}$ feasible), and work for optimality --- Feasible (primal) methods**
- **Start at a best convenient point ($x^{(1)}$ infeasible), stay on "best" but relaxed course ($x^{(k)}$ infeasible), and work to achieve feasibility---Infeasible (dual) methods**

Pros and Cons

Feasible methods

Pros:

- **Can stop anytime, and $x^{(k)}$ is always usable since it is feasible (although not necessarily optimal)**

Cons:

- **Less flexible to move—generally take longer and more costly**

Infeasible methods

Pros:

- **More flexible to move—generally more efficient and less costly**

Cons:

- **Cannot stop until done, and $x^{(k)}$ is not usable since it is normally infeasible**

8/6/2009 7

Classes of Methods

To find search direction $\mathbf{d}^{(k)}$: Solve a simpler problem

- Active Set-Strategy:
  - Gradient projection
  - Reduced gradient--Convert to equality constraints, eliminate variables, and solve bound constrained problems in reduced dimension
- Convert to unconstrained problems—penalty/barrier/Augmented Lagrangian
- Use Linear/Quadratic approximations and solve series of LPs or QPs—SLP/SQP
- Projective Transformation—interior point methods

8/6/2009 8

Common Classes of Methods

- Reduced Gradient (Feasible)
- Penalty/Augmented Lagrangian (Infeasible)
- Successive Quadratic Programming (SQP) (and Sequential Linear Programming (SLP)) (Infeasible)
- Interior-point (Feasible/Infeasible)

Current Software and Optimizers

Optimizer	Methodology	In Software
GRG, GRG2, LSGRG2	**Reduced Gradient**	Excel Solver, LINGO/GINO, GAMS, NAG, IMSL
CONOPT	**Reduced Gradient**	GAMS, AMPL, AIMMS, MPL
MINOS	Projected Augmented Lagrangian	GAMS, AMPL
LANCELOT	Augmented Lagrangian	Stand-alone LANCELOT in various platforms
SQP	**SQP**	MATLAB, OPTIMA, SQP, MATHCAD
NPSOL,NLPQL, SNOPT, SQOPT	**SQP**	Callable by GAMS, AMPL or stand-alone

Constrained Optimization (Nonlinear Programming)

- CO - nonlinear programming in the GAUSS language.
- CONOPT - nonlinear programming.
- DONLP2 - nonlinear programming.
- DOT - Design Optimization Tools.
- Excel and Quattro Pro Solvers - spreadsheet-based linear, integer and nonlinear programming.
- FSQP - nonlinear and minmax constrained optimization, with feasible iterates.
- GINO - nonlinear programming.
- GRG2 - nonlinear programming.
- HARWELL Library - linear and nonlinear programming, nonlinear equations, data fitting.
- ILOG - constraint-based programming and nonlinear optimization.
- LANCELOT - large-scale problems.
- LINGO - linear, integer, nonlinear programming with modeling language.
- LOQO - Linear programming, unconstrained and constrained nonlinear optimization.
- LSGRG2 - nonlinear programming.
- MINOS - linear programming and nonlinear optimization.

8/6/2009 11

Constrained Optimization (Nonlinear Programming)

- MOSEK - linear programming and convex nonlinear optimization.
- NLPJOB - Mulicriteria optimization.
- NLPQL - nonlinear programming.
- NLPQLB - nonlinear programming with constraints.
- NLPSPR - nonlinear programming.
- NPSOL - nonlinear programming.
- NOVA - nonlinear programming.
- OPTIMA Library - optimization and sensitivity analysis.
- PROC NLP - various nonlinear optimization capabilities.
- OPTPACK - constrained and unconstrained optimization.
- SNOPT - large-scale quadratic and nonlinear programming problems.
- SQP - nonlinear programming.
- SPRNLP - sparse and dense nonlinear programming.
- SYNAPS Pointer - multidiscplinary design optimization software.
- What's Best - Excel add-in for linear, integer, nonlinear programming.

8/6/2009 12

Quadratic Programming

- BQPD - quadratic programming.
- CPLEX - linear, quadratic, and network linear programming.
- FortMP - integer quadratic programming.
- LINDO - linear, mixed-integer and quadratic programming.
- LOQO - linear programming, unconstrained and constrained nonlinear optimization.
- LSSOL - least squares problems.
- MOSEK - linear programming and convex optimization (including convex quadratic programming).
- OSL - linear, quadratic and mixed-integer programming.
- PORT 3 - minimization, least squares, etc.
- PROC NLP - various nonlinear optimization capabilities.
- SQOPT - large-scale linear and convex quadratic programming.
- SNOPT - large-scale linear, quadratic, and nonlinear programming problems (including *nonconvex* quadratic programming.
- QL - convex quadratic programming.
- QPOPT - linear and quadratic problems

Nonlinear Least Squares

- DFNLP - nonlinear data fitting.
- HARWELL Library - linear and nonlinear programming, nonlinear equations, data fitting.
- LANCELOT - large-scale problems.
- LOQO - Linear programming, unconstrained and constrained nonlinear optimization.
- MINPACK-1 - nonlinear equations and least squares.
- MODFIT - parameter estimation in dynamic systems.
- NLSSOL - constrained nonlinear least squares problems.
- ODRPACK - NLS and ODR problems
- PDEFIT - parameter estimation in partial differential equations.
- PORT 3 - minimization, least squares, etc.
- PROC NLP - nonlinear minimization or maximization.
- SPRNLP - sparse nonlinear least squares.
- SYSFIT - parameter estimation in systems of nonlinear equations.
- TENSOLVE - nonlinear equations and least squares.
- VE10 - nonlinear least squares.

Nonlinear Equations

- CONTIN - systems of nonlinear equations.
- GAUSS - matrix programming language.
- HARWELL Library - linear and nonlinear programming, nonlinear equations, data fitting.
- HOMPACK - nonlinear equations and polynomials.
- LANCELOT - large-scale problems.
- LOQO - Linear programming, unconstrained and constrained nonlinear optimization.
- MINPACK-1 - nonlinear equations and least squares.
- NITSOL - systems of nonlinear equations.
- OPTIMA Library - optimization and sensitivity analysis.
- PETSc - parallel solution of nonlinear equations and unconstrained minimization problems.

Libraries with Optimization Capabilities

- HARWELL Library - linear and nonlinear programming, nonlinear equations, data fitting.
- IMSL - Fortran and C Library.
- NAG C Library - nonlinear and quadratic programming, minimization
- NAG Fortran Library - nonlinear and quadratic programming, minimiz ation

Optimization Systems/ Modeling Languages

- The AIMMS modeling language.
- The AMPL modeling language.
- DATAFORM - model management system.
- EASY FIT - parameter estimation in dynamic systems.
- Excel and Quattro Pro Solvers - spreadsheet-based linear, integer and nonlinear programming.
- EZMOD - modeling for decision support systems.
- GAMS - modeling language.
- GAUSS - language, oriented toward data analysis and statistical applications.
- LINGO - linear, integer, nonlinear programming with modeling language.
- MATLAB - optimization toolbox.
- MODLER - linear programming modeling language.
- MPL - modeling system.
- MPSIII - mathematical programming system (includes DATAFORM).
- OPL Studio - optimization language and solver environment.
- OPTIMAX - component software for optimization.
- PLAM - algebraic modeling language for mixed integer programming, constraint logic programming, etc.
- SPEAKEASY - numerical problems and operations research.
- PCOMP - modelling language with automatic differentiation.
- PROC NLP - nonlinear minimization or maximization.
- What'sBest - Excel add-in for linear, integer, and nonlinear programming.

Engineering Design Optimization Packages

- CONSOL-OPTCAD - engineering system design.
- COMPACT - design optimization.
- DOC - design optimization control program.
- GENESIS - structural optimization software.
- OPTDES - design optimization tool.
- SIMUSOLV - modeling software.
- SOCS - sparse optimal control; calls the SPRNLP package for nonlinear programming.
- ULTRAMAX - design and process optimization.

More References on Software and Methods

- Optimization Software Guide (Jorge J. Moré and Stephen J. Wright, SIAM Publications, 1993).
- **NEOS—Network Optimization Software**
  http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide
- **Nonlinear Optimization,** Jorge Nocedal and Stephen J. Wright, Springer, NY, 1999

8/6/2009 19

Constrained Optimization

- Basic Ideas still the same as unconstrained case:
  - Iterative: Beginning at an initial $x^{(1)}$, generate sequence $x^{(1)}, x^{(2)},.., x^{(k)},..$ until stop at x^*
  - At a current iterate $x^{(k)}$,
    - Determine a search direction $d^{(k)}$
    - Determine a stepsize $\alpha^{(k}$ along $d^{(k)}$
  - Update: $x^{(k+1)} = x^{(k)} + \alpha^{(k)}d^{(k)}$
- Only this time we need to consider "feasibility" when searching for search direction and stepsize.

8/6/2009 20

Constrained Optimization: Key Issues

- When to stop? –Characterization of solution points
- How do we make progress? -- Determining search direction $\mathbf{d}^{(k)}$ and stepsize $\alpha^{(k}$
- How do we measure progress toward achieving feasibility and optimality?

8/6/2009 21

Characterizing Optimal Points. For Constrained Problems

$$\text{NLP:} \quad \min_{\mathbf{x}\in R^n} f(\mathbf{x})$$
$$s.t. \quad h_j(\mathbf{x}) = 0, \quad j \in J_E$$
$$g_j(\mathbf{x}) \le 0, \quad j \in J_I$$

Unconstrained:	$J_E = \phi$ and $J_I = \phi$
Equality Constrained:	$J_E \neq \phi$ and $J_I = \phi$
Inequality Constrained:	$J_E = \phi$ and $J_I \neq \phi$
Mixed Inequality Constrained:	$J_E \neq \phi$ and $J_I \neq \phi$

8/6/2009 22

Characterizing Optimal Points: Unconstrained Problems

See Notes on "Unconstrained Problems:

In a nutshell;

- If x* is a local minimizer of *f*, then
  $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*)$ is *positive semi-definite* (*psd*)
- If $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*)$ is *positive definite* (*pd*), then x* is a strict local minimizer of *f*
- If *f* is convex $\nabla^2 f(\mathbf{x}^*)$ is *pd for all* **x,** then any local minimizer is global

8/6/2009 23

Characterizing Optimal Points: Equality Constrained Problems

For EP: $\min f(\mathbf{x})$ s.t. $\mathbf{x} \in R^n$, $h_j(\mathbf{x}) = 0, j \in J_E$

For some *Lagrange multipliers* λ_j, $j \in J_E$, let the *Lagrangian*

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j \in J_E} \lambda_j h_j(\mathbf{x}),$$

If at $\mathbf{x}^*, \nabla h_j(\mathbf{x}^*)$, $j \in J_E$ are linearly independent (or some other constraint qualification) and if there exist λ_j^*, $j \in J_E$ such that

i) $\nabla L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0$, and $h_j(\mathbf{x}^*) = 0$, $j \in J_E$

ii) $\mathbf{s}^T \nabla^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*)\mathbf{s} > 0$ for $\mathbf{s} \neq 0$ in $T = \{\mathbf{s} \in R^n \mid \nabla h_j(\mathbf{x}^*)\mathbf{s} = 0, j \in J_E\}$

(i.e. $\nabla^2 L(\mathbf{x}^*)$ is *pd* in the tangent space T at $\mathbf{x}^*$)

Then $\mathbf{x}^*$ is a strict local minimizer of $f(\mathbf{x})$ subject to the equality constraints. Moreover $\mathbf{x}^*$ is a unique global minimizer if $f(\mathbf{x})$ is convex and each $h_j(\mathbf{x})$ is linear--a convex programming problem.

8/6/2009 24

Characterizing Optimal Points:
Mixed Inequality Constrained Problems

For NLP: $\min f(\mathbf{x})$ s.t. $\mathbf{x} \in R^n$, $h_j(\mathbf{x}) = 0, j \in J_E$; $g_j(\mathbf{x}) \le 0, j \in J_I$

For some *multipliers* $\lambda_j,\ j \in J_E$, and $\mu_j,\ j \in J_I$, let the *Lagrangian*

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j \in J_E} \lambda_j h_j(\mathbf{x}) + \sum_{j \in J_I} \mu_j g_j(\mathbf{x})$$

Karush-Khun-Tucker (KKT) Theorem:

If at $\mathbf{x}^*, \nabla h_j(\mathbf{x}^*),\ j \in J_E$ and $\nabla g_j(\mathbf{x}^*),\ j \in J_I$ are linearly independent (or some other constraint qualification) and if there exist $\lambda_j^*,\ j \in J_E$ and $\mu_j^*,\ j \in J_I$ such that

i) $\nabla L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0$

ii) $g_j(\mathbf{x}^*) \le 0, j \in J_I$

iii) $\mu_j^* g_j(\mathbf{x}^*) = 0, j \in J_I$

iv) $\mu_j^* \ge 0, j \in J_I$

Then $\mathbf{x}^*$ is a KKT point of the NLP.

Characterizing Optimal Points:
Mixed Inequality Constrained Problems

For NLP: $\min f(\mathbf{x})$ s.t. $\mathbf{x} \in R^n$, $h_j(\mathbf{x}) = 0, j \in J_E$; $g_j(\mathbf{x}) \le 0, j \in J_I$

Lagrangian

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j \in J_E} \lambda_j h_j(\mathbf{x}) + \sum_{j \in J_I} \mu_j g_j(\mathbf{x})$$

SECOND-ORDER SUFFICIENCY: If

1) $\mathbf{x}^*$ is KKT point with multipliers $\boldsymbol{\lambda}^*$, and

2) $\mathbf{s}^T \nabla^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*)\mathbf{s} > 0$ for $\mathbf{s} \ne 0$ in the tangent space T, where

$$T = \left\{\mathbf{s} \in R^n \mid \nabla h_j(\mathbf{x}^*)\mathbf{s} = 0,\ j \in J_E, \nabla g_j(\mathbf{x}^*)\mathbf{s} = 0,\ j \in J_I \text{ with } \mu_j^* > 0, \nabla g_j(\mathbf{x}^*)\mathbf{s} \le 0,\ j \in J_I \text{ with } \mu_j^* = 0\right\}$$

Then $\mathbf{x}^*$ is a strict local minimizer of NLP.

Moreover it is a unique global minimizer if $f(\mathbf{x})$ is convex, each $h_j(\mathbf{x})$ is linear, and each $g_j(\mathbf{x})$ is convex--a convex programming problem.

Making Progress

To find a new search direction $\mathbf{d}^{(k)}$: Solve a simpler problem

- Convert to unconstrained problems—penalty/barrier/Augmented Lagrangian
- Convert to equality constraints, eliminate variables, and solve bound constrained problems in reduced dimension—reduced gradient/gradient projection
- Use Linear/Quadratic approximations and solve series of LPs or QPs—SLP/SQP
- Projective Transformation—interior point methods

8/6/2009 27

Penalty Function Method

NLP: $\min f(\mathbf{x})$ s.t. $\mathbf{x} \in X = \left\{ \mathbf{x} \in R^n \mid g_j(\mathbf{x}) \le 0, j \in J_I \right\}$

Convert to unconstrained problem using penalty:

$$q(\mathbf{x}:c) = f(\mathbf{x}) + cp(\mathbf{x})$$

where $p(\mathbf{x}) \begin{cases} > 0 \text{ when } \mathbf{x} \notin X \\ = 0 \text{ when } \mathbf{x} \in X \end{cases}$ for example, $p(\mathbf{x}) = \frac{1}{2} \sum_{j \in J_I} \left(\max(0, g_j(x) \right)^2$

c = penalty coefficient (large)

Hence, if c is large enough, minimizing $q(\mathbf{x}:c)$ with respect to $\mathbf{x}$ (unconstrained) should yield a solution $\mathbf{x}^*$ to the original NLP such that $p(\mathbf{x}^*) = 0$, i.e $\mathbf{x}^* \in$ X.

8/6/2009 28

Penalty Function Method SUMT: Fiacco-McCormick (1968, 1990)

0: Choose $\mathbf{x}^{(0)}$, and c_0, set $k = 0$

1: Solve: min $q(\mathbf{x}:c_k) = f(\mathbf{x}) + c_k p(\mathbf{x})$ to get $\mathbf{x}^{(k)}$ using $\mathbf{x}^{(k-1)}$ as a starting point.

2: Let $c_{k+1} > c_k$ (e.g. $c_{k+1} = 2c_k$), set $k = k+1$, and repeat (1) until $p(\mathbf{x}^{(k)}) < \varepsilon$

(i.e. $p(\mathbf{x}^{(k)} \approx 0 \Rightarrow \mathbf{x}^{(k)} \in X$)

Penalty Function Method SUMT: Fiacco-McCormick (1968, 1990)

- Begin at a moderate c_0 and gradually increase c_k to avoid dealing with ill-conditioned problem from the beginning, By starting from the previous solution point $\mathbf{x}^{(k-1)}$, which is assumed closed to $\mathbf{x}^{(k)}$, we can deal with ill conditioned better
- q reflects two things that we always want to achieve—feasibility and optimality—it is sometime known as **merit function** used to measure "progress"
- The method approaches $\mathbf{x}^*$ from the outside—infeasible method

SUMT: Key properties

:

- *Merit function* $q(\mathbf{x}:c)$ is monotone non-decreasing: $q(\mathbf{x}^{(k)}:c_k) \le q(\mathbf{x}^{(k+1)}:c_{k+1})$
- Infeasibility measure is monotone non-increasing: $p(\mathbf{x}^{(k)}) \ge p(\mathbf{x}^{(k+1)})$
- Objective function is monotone non-decreasing: $f(\mathbf{x}^{(k+1)}) \ge f(\mathbf{x}^{(k)})$
- The algorithm converges to $\mathbf{x}^*$.
- Drawback: Need a large c to find $\mathbf{x}^*$, but get ill-conditioned when c_k becomes large

Barrier Function Method

NLP: $\min f(\mathbf{x})$ s.t. $\mathbf{x} \in X = \{\mathbf{x} \in R^n \mid g_j(\mathbf{x}) \le 0, j \in J_I\}$

Convert to unconstrained problem using penalty:

$$r(\mathbf{x}:c) = f(\mathbf{x}) + (1/c)b(\mathbf{x})$$

where $b(\mathbf{x}) \begin{cases} > 0 \text{ when } \mathbf{x} \in \text{interior of } X \\ = \infty \text{ when } \mathbf{x} \text{ near boundary of } X \end{cases}$ and c = large coefficient

e.g., $b(\mathbf{x}) = \sum_{j \in J_I} \ln(-g_j(\mathbf{x}))$ or $\sum_{j \in J_I} \frac{1}{g_j(\mathbf{x})}$

Thus we can minimize unconstrained $r(\mathbf{x}:c)$, starting from an interior $\mathbf{x}^{(0)}$ and a low c_0 and successively increase c_k ($c_{k+1} > c_k$ >…) until c_k is large enough, and this should yield a solution $\mathbf{x}^*$ to the original NLP. Note that the sequence $\mathbf{x}^{(k)}$ **should remain interior**, if $\mathbf{x}^{(0)}$ is. Hence this is a feasible method.

Augmented Lagrangian Method

Consider NLP: min $f(\mathbf{x})$ s.t. $\mathbf{x} \in R^n$, $h_j(\mathbf{x}) = 0, j \in J_E$

Note: any inequality $g_j(\mathbf{x}) \leq 0$ can be converted to equality as $g_j(\mathbf{x}) + v^2 = 0$ or $g_j(\mathbf{x}) + v = 0$, $v \geq 0$.

Convert to unconstrained problem using augmented Lagragian:

$$l(\mathbf{x}, \boldsymbol{\lambda}, \rho) = f(\mathbf{x}) + \sum_{j \in J_E} \lambda_j h_j(\mathbf{x}) + \tfrac{1}{2}\rho \sum_{j \in J_I} \mu_j \left|h_j(\mathbf{x})\right|^2$$

Hence, if $\boldsymbol{\lambda}$ and ρ are chosen properly, minimizing unconstrained $l(\mathbf{x}: \lambda, \rho)$ should yield a solution $\mathbf{x}^*$ to the original NLP with $h_j(\mathbf{x}^*) = 0$

Augmented Lagrangian Method

Why is this good? It has been shown that:

The last term of $l(\mathbf{x},\lambda,\rho)$ has the effect of "CONVEXIFYING: the problem by making $l(\mathbf{x},\lambda,\rho)$ locally convex around $\mathbf{x}^*$. This lead to the following very important results:

a) If $\mathbf{x}^*$ is a local minimizer of $l(\mathbf{x},\lambda,\rho)$ for some value of $(\lambda^{(k)},\rho^{(k)})$, such that $l(\mathbf{x}, \lambda^{(k)},\rho^{(k)})$ is locally convex and that $\nabla^2 l(\mathbf{x}, \lambda^{(k)},\rho^{(k)})$ is *pd* (second-order sufficient conditions), then $\mathbf{x}^*$ is a minimizer of the original NLP

b) If $\mathbf{x}^*$ is regular point (gradients of all active constraints are active) and a solution of the NLP with multipliers λ^*, such that the second-order sufficiency conditions apply , then there is $\rho^* < \infty$ such that for all $\rho \geq \rho^*$, $\mathbf{x}^*$ is a local minimizer of $l(\mathbf{x}, \lambda^*,\rho)$.

CASE SCHOOL OF ENGINEERING

Augmented Lagrangian Method

Why is this good?

The result (b) in the previous page, indicates that ρ does not need *to* be as high as that used in the penalty function, hence avoiding the ill-conditioned effect.

How do we choose a right (λ^*, ρ^*)**:** ρ^* is a little easier to select, but a right λ^* requires some work.

The following is a typical implementation:

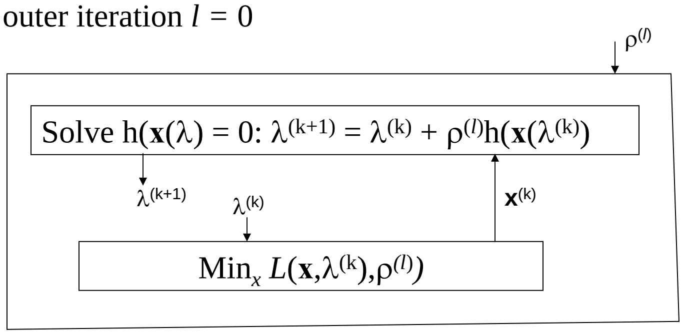
8/6/2009 35

CASE SCHOOL OF ENGINEERING

Augmented Lagrangian Method

Typical implementation:

Start with a low ρ_0 (since we are going to update it by doubling it) and a proper trial $\lambda^{(0)}$. Set inner iteration k =0, and outer iteration l = 0



8/6/2009 36

Reduced Gradient Method

LNLP: $\min f(\mathbf{x})$

s.t. $\mathbf{Ax} = \mathbf{b}$

$\mathbf{x} \geq 0$

Simplify by eliminating variables:

Assume: $\mathbf{A} = m \times n$, $m < n$, and $rank(\mathbf{A}) = m$

With row-column permutation if needed, collect m independent columns of **A** and form

$\mathbf{A} = (\mathbf{B}:\mathbf{C})$

where $\mathbf{B} = m \times m$ nonsinglar (basic) matrix

$\mathbf{C} = m \times (n\text{-}m)$ (nonbasic) matrix

Reduced Gradient Method

Let $\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix}$, $\mathbf{y} > 0$: $\mathbf{y}$ = m-basic variables; z = (n-m)-basic variables

$$\mathbf{Ax} = \mathbf{b} \Rightarrow (\mathbf{B}:\mathbf{C})\begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \mathbf{By} + \mathbf{Cz} = \mathbf{b}$$

$$\Rightarrow \mathbf{y} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{Cz} =$$

$\therefore$ LNLP $\equiv \min \quad f(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{Cz}, \mathbf{z}) = \hat{f}(\mathbf{z})$

s.t. $\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{Cz} \geq 0$ can be ignored temporarily since $\mathbf{y} > 0$

$\mathbf{z} \geq 0$

Around $\mathbf{z}^{(k)}$, LNLP becomes: P: $\min \hat{f}(\mathbf{z})$

$\mathbf{z} \geq 0$

$\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{Cz} \geq 0$

Reduced Gradient Method

P: $\min \hat{f}(\mathbf{z})$

$\mathbf{z} \geq 0$ (1)

$\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{C}\mathbf{z} \geq 0$ (2)

This is obviously easier to solve than LNLP:

$\dim(\mathbf{z}) < \dim(\mathbf{x})$; $\mathbf{z}^{(k)}$ is a feasible point of P;

and (1)&(2) are simpler constraints.

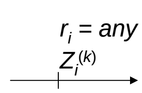
Applying a modified steepest descent (to accommodate $\mathbf{z} \geq 0$) to P:

Reduced Gradient is

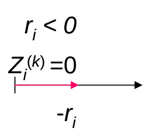
$$\mathbf{r} = \nabla \hat{f}(\mathbf{z}) = \frac{\partial \hat{f}(\mathbf{z})}{\partial z} = \frac{\partial f(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{C}\mathbf{z}, \mathbf{z})}{\partial z} = \frac{\partial f(\mathbf{y},\mathbf{z})}{\partial y}\left(-\mathbf{B}^{-1}\mathbf{C}\right) + \frac{\partial f(\mathbf{y},\mathbf{z})}{\partial z}$$

$$\mathbf{r} = -\nabla_y f(\mathbf{y},\mathbf{z})\mathbf{B}^{-1}\mathbf{C} + \nabla_z f(\mathbf{y},\mathbf{z})$$

Reduced Gradient Method

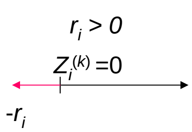


$\Rightarrow \Delta z_i = -r_i$



$\Rightarrow \Delta z_i = -r_i$

$$\Delta z = \begin{pmatrix} \Delta z_1 \\ .. \\ \Delta z_n \end{pmatrix} \text{ using } \Delta z_i \text{ as found}$$



$\Rightarrow \Delta z_i = 0$

Δz can be used as a search direction

We can show that if $\Delta z = 0$, $\mathbf{x}^{(k)}$ is a KKT point.

Reduced Gradient Method

To find a step size $\alpha^{(k)}$:

Compute $\Delta\mathbf{y} = -\mathbf{B}^{-1}\mathbf{C}\,\Delta\mathbf{z}$

Compute: $\alpha_1 = \min_{\Delta y_i < 0}\left(\frac{y_i^{(k)}}{-\Delta y_i}\right)$

Compute: $\alpha_2 = \min_{\Delta z_i < 0}\left(\frac{z_i^{(k)}}{-\Delta z_i}\right)$

Then compute $\alpha_3 = \min(\alpha_1, \alpha_2)$

Finally, do line search to find $\alpha^{(k)} = \min_{0<\alpha<\alpha_3}\left(f(\mathbf{y}^{(k)} + \alpha\Delta\mathbf{z}, \mathbf{z}^{(k)} + \alpha\Delta\mathbf{z}\right)$

8/6/2009 41

Successive Quadratic Programming (SQP)

Basic Idea:

1) Approximate $f(\mathbf{x})$ by a quadratic

and $h_j(\mathbf{x})$ and $g_j(\mathbf{x})$ by linear functions

At $\mathbf{x}^{(k)}$, solve

$$\text{QP}^{(k)}: \min f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)}) + \tfrac{1}{2}(\mathbf{x}-\mathbf{x}^{(k)})^T \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)})$$

$$\text{s.t. } h_j(\mathbf{x}^{(k)}) + \nabla h_j(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)}) = 0,\ j \in J_E$$

$$g_j(\mathbf{x}^{(k)}) + \nabla g_j(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)}) = 0,\ j \in J_I$$

Note that $\mathbf{x} - \mathbf{x}^{(k)} = \mathbf{d}^{(k)}$

8/6/2009 42

Successive Quadratic Programming (SQP)

So

QP$^{(k)}$: $\min f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})\mathbf{d}^{(k)} + \frac{1}{2}\mathbf{d}^{(k)T}\nabla^2 f(\mathbf{x}^{(k)})\mathbf{d}^{(k)}$

$$\text{s.t. } h_j(\mathbf{x}^{(k)}) + \nabla h_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)} = 0,\ j \in J_E$$

$$g_j(\mathbf{x}^{(k)}) + \nabla g_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)} = 0,\ j \in J_I$$

Solve QP$^{(k)}$ by a suitable method to get $\mathbf{d}^{(k)}$

Update $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$

Basic Idea of SQP:
An Illustration

Example 1

NLP: $\min_{\mathbf{x}\in R^2} f(\mathbf{x}) = \frac{6x_1}{x_2} + \frac{x_2}{x_1^2}$

$$s.t.\quad h(\mathbf{x}) = x_1 x_2 - 2 = 0$$

$$g(\mathbf{x}) = -x_1 - x_2 + 1 \le 0$$

$$\mathbf{x}^{(0)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Basic Idea of SQP:
An Illustration (cont.)

$$\nabla f(x_1,x_2)=\begin{pmatrix}\frac{6}{x_2}-\frac{2x_2}{x_1^3} & \frac{-6x_1}{x_2^2}+\frac{1}{x_1^2}\end{pmatrix}, \nabla^2 f(x_1,x_2)=\begin{pmatrix}\frac{6x_2}{x_1^4} & \frac{-6}{x_2^2}-\frac{2}{x_1^3}\\ \frac{-6}{x_2^2}-\frac{2}{x_1^3} & \frac{12x_1}{x_2^3}\end{pmatrix}$$

$$\nabla h(x_1,x_2)=\begin{pmatrix}x_2 & x_1\end{pmatrix}; \quad \nabla g(x_1,x_2)=\begin{pmatrix}-1 & -1\end{pmatrix}$$

$$f(\mathbf{x}^{(0)})=12.25; \nabla f(\mathbf{x}^{(0)})=\begin{pmatrix}\frac{23}{4} & \frac{-47}{4}\end{pmatrix}, \quad \nabla^2 f(\mathbf{x}^{(0)})=\begin{pmatrix}\frac{3}{8} & \frac{-25}{4}\\ \frac{-25}{4} & 24\end{pmatrix}$$

$$h(\mathbf{x}^{(0)})=0; \nabla h(\mathbf{x}^{(0)})=\begin{pmatrix}1 & 2\end{pmatrix}; \quad g(\mathbf{x}^{(0)})=-2; \nabla g(\mathbf{x}^{(0)})=\begin{pmatrix}-1 & -1\end{pmatrix}$$

Basic Idea of SQP:
An Illustration (cont.)

$$\text{QP}^{(0)}: \quad \min_{\mathbf{x}\in R^2} q(\mathbf{x})=12.25+\begin{pmatrix}\frac{23}{4} & \frac{-47}{4}\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}+\begin{pmatrix}d_1 & d_2\end{pmatrix}\begin{pmatrix}\frac{3}{8} & \frac{-25}{4}\\ \frac{-25}{4} & 24\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}$$

$$s.t. \quad \hat{h}(\mathbf{x})=0+\begin{pmatrix}1 & 2\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}=0$$

$$\hat{g}(\mathbf{x})=-2+\begin{pmatrix}-1 & -1\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}\le 0$$

$$\Rightarrow \mathbf{d}^{(0)}=\begin{pmatrix}-0.92\\ 0.46\end{pmatrix} \Rightarrow \mathbf{x}^{(1)}=\mathbf{x}^{(0)}+\mathbf{d}^{(0)}=\begin{pmatrix}2\\ 1\end{pmatrix}+\begin{pmatrix}-0.92\\ 0.46\end{pmatrix}=\begin{pmatrix}1.08\\ 1.46\end{pmatrix}$$

Basic Idea of SQP:
An Illustration (cont.)

$$\text{QP}^{(1)}: \quad \min_{\mathbf{x}\in R^2} q(\mathbf{x}) = \begin{pmatrix}1.78 & -2.18\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix} + \begin{pmatrix}d_1 & d_2\end{pmatrix}\begin{pmatrix}6.46 & -4.40\\ -4.40 & 4.16\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}$$

$$s.t. \quad \hat{h}(\mathbf{x}) = -0.42 + \begin{pmatrix}1.46 & 1.08\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix} = 0$$

$$\hat{g}(\mathbf{x}) = -1.54 + \begin{pmatrix}-1 & -1\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix} \le 0$$

$$\Rightarrow \mathbf{d}^{(1)} = \begin{pmatrix}-0.03\\ 0.43\end{pmatrix} \Rightarrow \mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \mathbf{d}^{(1)} = \begin{pmatrix}1.08\\ 1.46\end{pmatrix} + \begin{pmatrix}-0.03\\ 0.43\end{pmatrix} = \begin{pmatrix}1.05\\ 1.89\end{pmatrix}, h(\mathbf{x}^{(1)}) = .01$$

$$\text{Continue until: } \mathbf{x}^{(4)} = \begin{pmatrix}1.00014\\ 1.99971\end{pmatrix}, h(\mathbf{x}^{(4)}) = -0.62\times 10^{-6}$$

Basic Idea of SQP:
An Illustration (cont.)

Example 2

$$\text{NLP:} \quad \min_{\mathbf{x}\in R^2} f(\mathbf{x}) = x_1 x_2$$

$$s.t. \quad h(\mathbf{x}) = \frac{6x_1}{x_2} + \frac{x_2}{x_1^2} - 5 = 0$$

$$g(\mathbf{x}) = -x_1 - x_2 + 1 \le 0$$

$$\mathbf{x}^{(0)} = \begin{pmatrix}2\\ 1\end{pmatrix}$$

Basic Idea of SQP:
An Illustration (cont.)

$$\nabla f(x_1,x_2)=\begin{pmatrix}x_2 & x_1\end{pmatrix}, \nabla^2 f(x_1,x_2)=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$$

$$\nabla h(x_1,x_2)=\begin{pmatrix}\frac{6}{x_2}-\frac{2x_2}{x_1^3} & \frac{-6x_1}{x_2^2}+\frac{1}{x_1^2}\end{pmatrix};\quad \nabla g(x_1,x_2)=\begin{pmatrix}-1 & -1\end{pmatrix}$$

$$f(\mathbf{x}^{(0)})=2;\nabla f(\mathbf{x}^{(0)})=\begin{pmatrix}1 & 2\end{pmatrix},\quad \nabla^2 f(\mathbf{x}^{(0)})=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$$

$$h(\mathbf{x}^{(0)})=\frac{29}{4};\nabla h(\mathbf{x}^{(0)})=\begin{pmatrix}\frac{23}{4} & \frac{-47}{4}\end{pmatrix};\quad g(\mathbf{x}^{(0)})=-2;\nabla g(\mathbf{x}^{(0)})=\begin{pmatrix}-1 & -1\end{pmatrix}$$

Basic Idea of SQP:
An Illustration (cont.)

$$\text{QP}^{(0)}:\quad \min_{\mathbf{x}\in R^2} q(\mathbf{x})=2+\begin{pmatrix}1 & 2\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}+\begin{pmatrix}d_1 & d_2\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}$$

$$s.t.\quad \widehat{h}(\mathbf{x})=\frac{29}{4}+\begin{pmatrix}\frac{23}{4} & \frac{-47}{4}\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}=0$$

$$\widehat{g}(\mathbf{x})=-2+\begin{pmatrix}-1 & -1\end{pmatrix}\begin{pmatrix}d_1\\ d_2\end{pmatrix}\le 0$$

$$\Rightarrow \mathbf{d}^{(0)}=\begin{pmatrix}-1.75\\ -0.24\end{pmatrix}\Rightarrow \mathbf{x}^{(1)}=\mathbf{x}^{(0)}+\mathbf{d}^{(0)}=\begin{pmatrix}2\\ 1\end{pmatrix}+\begin{pmatrix}-1.75\\ -0.24\end{pmatrix}=\begin{pmatrix}0.24\\ 0.76\end{pmatrix}$$

Basic Idea of SQP: An Illustration (cont.)

Here the method does not work well, since *h* is very sharp at $\mathbf{x}^*= (1,2)^T$

⇒ Curvature of constraints are also important in determining how well we can approach $\mathbf{x}^*$

⇒ Need to improve on the basic method

8/6/2009 51

Successive Quadratic Programming (SQP)

Advantages:

- Simple and efficient, if it works
- Linear approximation helps define direction
- Quadratic approximation helps define step size

Disadvantages:

- Approximation may be inaccurate
- Does not always work as planned (direction and/or step size may be no good particularly if Hessian is not pd.

8/6/2009 52

Successive Quadratic Programming (SQP)

Strategies for improvement:

- Include curvature of constraints to get better approx. Either
  - Approx high curvature nonlinear constraints as quadratics
  - Include Hessian of constraints in objective function—quadratic approx of Lagrangian

8/6/2009 53

Successive Quadratic Programming (SQP)

Strategies for improvement:

This is a constrained version of Newton's method: It has all disadvantages of Newton's

- Improve by using line search using merit function
- Use Quasi-Newton to approximate Hessian of objective function to reduce computational costs and ensure pd.

8/6/2009 54

SQP: Strategies for improvement:

Include curvature of constraints to get better approximation: Strategy 1

Include Hessian of constraints in objective function—quadratic approx of Lagrangian

At $\mathbf{x}^{(k)}$, solve

$$\text{QP}^{(k)}: \min L(\mathbf{x}^{(k)},\lambda^{(k)})+\nabla L(\mathbf{x}^{(k)},\lambda^{(k)})\mathbf{d}^{(k)}+\tfrac{1}{2}\mathbf{d}^{(k)T}\nabla^2 L(\mathbf{x}^{(k)},\lambda^{(k)})\mathbf{d}^{(k)}$$

$$\text{s.t. } h_j(\mathbf{x}^{(k)})+\nabla h_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)}=0,\ j\in J_E$$

$$g_j(\mathbf{x}^{(k)})+\nabla g_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)}\le 0,\ j\in J_I$$

Note that $\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}+\mathbf{d}^{(k)}$

8/6/2009 55

SQP: Strategies for improvement:

Include curvature of constraints to get better approximation: Strategy 2

Approx high curvature nonlinear constraints as quadratics

At $\mathbf{x}^{(k)}$, solve

$$\text{QP}^{(k)}: \min L(\mathbf{x}^{(k)},\lambda^{(k)})+\nabla L(\mathbf{x}^{(k)},\lambda^{(k)})\mathbf{d}^{(k)}+\tfrac{1}{2}\mathbf{d}^{(k)T}\nabla^2 L(\mathbf{x}^{(k)},\lambda^{(k)})\mathbf{d}^{(k)}$$

$$\text{s.t. } h_j(\mathbf{x}^{(k)})+\nabla h_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)}+\tfrac{1}{2}\mathbf{d}^{(k)T}\nabla^2 h_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)}=0,\ j\in J_E$$

$$g_j(\mathbf{x}^{(k)})+\nabla g_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)}\ \tfrac{1}{2}\mathbf{d}^{(k)T}\nabla^2 g_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)}\le 0,\ j\in J_I$$

Note that $\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}+\mathbf{d}^{(k)}$

8/6/2009 56

SQP: Strategies for improvement:

In any case, this is a constrained version of Newton's with all its disadvantages. Strategies for improvement

a) Improve by using line search using merit function

$$P(\mathbf{x}, R) = f(\mathbf{x}) + R\left\{\sum_{i=1}^{k}\left(h_i(\mathbf{x})\right)^2 + \sum_{i=1}^{l}\left(\max\left(0, g_i(\mathbf{x})\right)\right)^2\right\}$$

Use this merit function to find step size $\alpha^{(k)}$, and then $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$

8/6/2009 57

SQP: Strategies for improvement:

This is a constrained version of Newton's with all its disadvantages. Strategies for improvement

b) Use Quasi-Newton to approximate the Hessian of objective function to reduce computational costs and ensure positive definiteness.

At $\mathbf{x}^{(k)}$, solve

$$\text{QP}^{(k)}: \min f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})\mathbf{d}^{(k)} + \tfrac{1}{2}\mathbf{d}^{(k)T}\mathbf{H}^{(k)}\mathbf{d}^{(k)}$$

$$\text{s.t. } h_j(\mathbf{x}^{(k)}) + \nabla h_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)} = 0,\ j \in J_E$$

$$g_j(\mathbf{x}^{(k)}) + \nabla g_j(\mathbf{x}^{(k)})\mathbf{d}^{(k)} \le 0,\ j \in J_I$$

Note that $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$

where $\mathbf{H}^{(k)}$ is updated by BFGS or DFP-like formular, so that $\mathbf{H}^{(k)}$ is always positive definite and $\mathbf{H}^{(k)} \to \nabla^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*)$

8/6/2009 58

SQP: Implementation

To use SQP, we need an efficient method to solve Quadratic Programs: How?

$$\text{QP: } \min a + \mathbf{q}^T\mathbf{d} + \tfrac{1}{2}\mathbf{d}^T\mathbf{Q}\mathbf{d}$$
$$\text{s.t. } \mathbf{Ad} = \mathbf{b}$$
$$\mathbf{Gd} \le \mathbf{c}$$
$$\mathbf{d} \ge \mathbf{0}$$

1) If **Q** is *pd* –easy: Use Wolfe's method based on LP simplex method

2) If **Q** is *psd*—Use Lemke's method

3) If **Q** is *id*—Use Active set Strategy

8/6/2009 59

SQP: Implementation

$$\text{QP: } \min c + \mathbf{q}^T\mathbf{d} + \tfrac{1}{2}\mathbf{d}^T\mathbf{Q}\mathbf{d}$$
$$\text{s.t. } \mathbf{Ad} = \mathbf{b}$$
$$\mathbf{d} \ge \mathbf{0}$$

All methods require solving the KKT conditions:
Assume that we have only equality constraints:

1) Any local solution is a global solution—amazing for QP even if it is not convex.

2) Hence, any solution of QP must be a KKT point and vice versa.

8/6/2009 60

SQP: Implementation

QP: $\min\ c + \mathbf{q}^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{Q}\mathbf{d}$

s.t. $\mathbf{Ad} = \mathbf{b}$

$\mathbf{d} \geq \mathbf{0}$

KKT Conditions: $\mathbf{d}^*$ is a KKT point of the QP if and only if there exist multipliers $\boldsymbol{\lambda}^*$ such that:

$$\begin{pmatrix} \mathbf{Q} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{d}^* \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} -\mathbf{q} \\ \mathbf{b} \end{pmatrix} \quad or$$

Noting that $\mathbf{d}^* = \mathbf{d} + \mathbf{p},\ \mathbf{c} = \mathbf{Ad} - \mathbf{b}, \mathbf{g} = \mathbf{Qd} + \mathbf{q}$ we have

$$\begin{pmatrix} \mathbf{Q} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix}\begin{pmatrix} -\mathbf{p} \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{c} \end{pmatrix} \qquad (3)$$

All methods solve (3) in one way or another.

SQP: Implementation

For example:

$\min f(x_1, x_2, x_3) = 3x_1^2 + 2x_1x_2 + x_1x_3 + 2x_2x_3 + 2.5x_2^2 + 2x_3^2 - 8x_1 - 3x_2 - 3x_3$

s.t. $x_1 + x_3 = 3;\ x_2 + x_3 = 0,\ x_i \geq 0,\ \ i = 1,2,3$

$\Rightarrow f(\mathbf{x}) = 0 + \mathbf{q}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}$, where

$$\mathbf{q} = \begin{pmatrix} -8 \\ -3 \\ -2 \end{pmatrix}, \mathbf{Q} = \begin{pmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{pmatrix}, \mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

$\mathbf{d} \geq \mathbf{0}$

Solving the KKT conditions (3) yields:

$$\mathbf{x}^* = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix},\ \boldsymbol{\lambda}^* = \begin{pmatrix} 3 \\ -2 \end{pmatrix}$$

SQP: Implementation

Solving the KKT condition below:

$$\begin{pmatrix} \mathbf{Q} & \mathbf{-A^T} \\ \mathbf{A} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{-p} \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{c} \end{pmatrix} \text{ or } \mathbf{K}\begin{pmatrix} \mathbf{-p} \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{c} \end{pmatrix} \qquad (3)$$

1) Direct solution: Using symmetric indefinite factorization:

$$\mathbf{P}^T\mathbf{KP} = \mathbf{LBL}^T$$

where $\mathbf{P}$ = permutation matrix;

$\mathbf{L}$ = unit lower triangular

$\mathbf{B}$ = Block diagonal with 1x1 or 2x2 blocks

Solve $\mathbf{Ly} = \mathbf{P}^T\begin{pmatrix} \mathbf{g} \\ \mathbf{c} \end{pmatrix}$ to get $\mathbf{y}$

Solve $\mathbf{B}\hat{\mathbf{y}} = \mathbf{y}$ to get $\hat{\mathbf{y}}$

Solve $\mathbf{L}^T\bar{\mathbf{y}} = \hat{\mathbf{y}}$ to get $\bar{\mathbf{y}}$

Set $\begin{pmatrix} \mathbf{-p} \\ \boldsymbol{\lambda}^* \end{pmatrix} = \mathbf{P}\bar{\mathbf{y}}$

Half the cost of sparse Gaussian Elimination

8/6/2009 63

SQP: Implementation

Solving the KKT condition below:

$$\begin{pmatrix} \mathbf{Q} & \mathbf{-A^T} \\ \mathbf{A} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{-p} \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{c} \end{pmatrix} \text{ or } \mathbf{K}\begin{pmatrix} \mathbf{-p} \\ \boldsymbol{\lambda}^* \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{c} \end{pmatrix} \qquad (3)$$

2) Range-Space Method: Q is assumed pd:

$$\left(\mathbf{AQ^{-1}A^T}\right)\boldsymbol{\lambda}^* = \left(\mathbf{AQ^{-1}g} - \mathbf{c}\right)$$

3) Null Space Method:

$$\mathbf{p} = \mathbf{Yp_y} + \mathbf{Zp_z}$$

$$\mathbf{AYp_y} = \mathbf{-c}$$

$$\mathbf{-GYp_y} - \mathbf{GZp_z} + \mathbf{A^T}\boldsymbol{\lambda}^* = \mathbf{g}$$

$$\mathbf{Z^TGZp_z} = -(\mathbf{Z^TGYp_y} + \mathbf{Z^Tg})$$

$$(\mathbf{AY})^\mathbf{T}\boldsymbol{\lambda}^* = \mathbf{Y^T}(\mathbf{g} + \mathbf{Gp})$$

4) Method based on conjugacy

8/6/2009 64

Projective Transformation: Interior Point Methods

Interior Point Method

KKT: $\nabla f_0(\mathbf{x}) + \sum_{i=1}^{m} y_i \nabla f_i(\mathbf{x}) + \mathbf{A}^T\mathbf{z} = 0 \qquad (1)$

$f_i(\mathbf{x}) + s_i = 0, \quad i = 1,..,m \qquad (2a)$

$\mathbf{Ax} = \mathbf{b} \qquad (2b)$

$y_i s_i = 0, \quad i = 1,..,m \qquad (3)$

$y_i \geq 0, s_i \geq 0, \quad i = 1,..,m \qquad (4)$

Again at iteration k with $(\mathbf{x}^{(k)}, \mathbf{s}^{(k)}, \mathbf{y}^{(k)}, \mathbf{z}^{(k)})$ and $\left(\mathbf{s}^{(k)}, \mathbf{z}^{(k)}\right)$>0 and the duality gap $\tau^{(k)}$

we solve the relaxed KKT for the new serach direction($\Delta\mathbf{x}, \Delta\mathbf{s}, \Delta\mathbf{y}, \Delta\mathbf{z}$):

KKT$^{(k)}$: $\nabla f_0(\mathbf{x}^{(k)} + \Delta\mathbf{x}) + \sum_{i=1}^{m} (y_i + \Delta y_i) \nabla f_i(\mathbf{x}^{(k)} + \Delta\mathbf{x}) + \mathbf{A}^T(\mathbf{z}^{(k)} + \Delta\mathbf{z}) = 0 \qquad (1)$

$f_i(\mathbf{x}^{(k)} + \Delta\mathbf{x}) + (s_i^{(k)} + \Delta s_i) = 0, \quad i = 1,..,m \qquad (2a)$

$\mathbf{A}(\mathbf{x}^{(k)} + \Delta\mathbf{x}) = \mathbf{b} \qquad (2b)$

$\left(\mathbf{Y}^{(k)} + \Delta\mathbf{Y}\right)\left(\mathbf{S}^{(k)} + \Delta\mathbf{S}\right)\mathbf{e} = \tau^{(k)}\mathbf{e} \qquad (3)$

Notice again that with $\tau^{(k)} > 0$, the nonnegativity condition (4) is automatically satisfied.

Interior Point Method

A typical strategy is to solve (1)-(3) above using a variant of Newton's method and perform a simple line search to find stepsize to ensure strict nonnegativity.
The Newton method requires solving a linearized version of the KKT:

$$\text{KKT}^{(k)}: \begin{pmatrix} \nabla^2 f_0(\mathbf{x}^{(k)}) + \sum_{i=1}^{m} y_i^{(k)} \nabla^2 f_i(\mathbf{x}^{(k)}) & 0 & \nabla \mathbf{f}(\mathbf{x}^{(k)}) & \mathbf{A}^T \\ \nabla \mathbf{f}(\mathbf{x}^{(k)})^T & \mathbf{I} & 0 & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ 0 & \mathbf{Y}^{(k)} & \mathbf{S}^{(k)} & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ \mathbf{r}_4 \end{pmatrix}$$

where $\mathbf{r}_1 = -\nabla f_0(\mathbf{x}^{(k)}) - \sum_{i=1}^{m} y_i^{(k)} \nabla f_i(\mathbf{x}^{(k)}) - \mathbf{A}^T \mathbf{z}^{(k)}$; $\mathbf{r}_2 = -\mathbf{f}(\mathbf{x}^{(k)}) - \mathbf{s}^{(k)}$; $\mathbf{r}_3 = -\mathbf{A}\mathbf{x}^{(k)}$

and $\mathbf{r}_4 = -\mathbf{Y}^{(k)}\mathbf{S}^{(k)}\mathbf{e}$ for the prediction step, and

$= -\mathbf{Y}^{(k)}\mathbf{S}^{(k)}\mathbf{e} - \Delta\mathbf{Y}_{aff}\Delta\mathbf{S}_{aff}\mathbf{e} + \rho_k \mu_k \mathbf{e}$ for the corrected centering step

Also, $\mathbf{f}(\mathbf{x}^{(k)}) = \begin{pmatrix} f_1(\mathbf{x}^{(k)}) \\ . \\ f_m(\mathbf{x}^{(k)}) \end{pmatrix}$ and $\nabla \mathbf{f}(\mathbf{x}^{(k)}) = \left(\nabla f_1(\mathbf{x}^{(k)}) : \nabla f_2(\mathbf{x}^{(k)}) : \ldots : \nabla f_m(\mathbf{x}^{(k)}) \right)$

Interior Point Method

Two basic strategies:

1. The primal approach: Solve a Newton system and keep the primal feasibility
   This is equivalent to solving the Barrier problem:

   $$\min\ f_0(\mathbf{x}) + \tau\left(-\sum_{i=1}^{m} \log(-f_i(\mathbf{x}))\right),\ \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}$$

   The adjusted KKT system to be solved reflects the relaxed KKT system for the above Barrier problem. This will be discussed later.
2. The primal-dual approach which consists of the predition step and centering correction step similar to before. This is described in detail next. For convenient, we will write the relaxed linearized KKT to be solved as:

$$\text{KKT}^{(k)}: \begin{pmatrix} \nabla_{xx}^2 L^{(k)} & 0 & \mathbf{F}^T & \mathbf{A}^T \\ \mathbf{F} & \mathbf{I} & 0 & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ 0 & \mathbf{Y}^{(k)} & \mathbf{S}^{(k)} & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ \mathbf{r}_4 \end{pmatrix}$$

where $\nabla_{xx}^2 L^{(k)} = \nabla^2 f_0(\mathbf{x}^{(k)}) + \sum_{i=1}^{m} y_i^{(k)} \nabla^2 f_i(\mathbf{x}^{(k)})$ and $\mathbf{F} = \nabla \mathbf{f}(\mathbf{x}^{(k)})^T$

Note: $L(\mathbf{x}, \mathbf{s}, \mathbf{y}, \mathbf{z}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} y_i (f_i(\mathbf{x}) + s_i) + \mathbf{z}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$

Predictor-Corrector Primal-Dual Version

1. Given ($\mathbf{x}^{(0)}$, $\mathbf{s}^{(0)}$, $\mathbf{y}^{(0)}$, $\mathbf{z}^{(0)}$) with $\mathbf{s}^{(0)} > \mathbf{0}$, $\mathbf{y}^{(0)} > \mathbf{0}$ set $k = 0$.

2. Check for optimality: STOP if all of the following are true:

- dual feasibility: $\left\|\mathbf{r}_1^{(k)}\right\| = \left\|\nabla f_0(\mathbf{x}^{(k)}) + \sum_{i=1}^{m} y_i^{(k)} \nabla f_i(\mathbf{x}^{(k)}) + \mathbf{A}^T \mathbf{z}^{(k)}\right\| \le \tau_k$
- primal feasibility: $\left\|\mathbf{r}_2^{(k)}\right\| = \left\|\mathbf{f}(\mathbf{x}^{(k)}) + \mathbf{s}^{(k)}\right\| \le \tau_k$

  $\left\|\mathbf{r}_3^{(k)}\right\| = \left\|\mathbf{A}\mathbf{x}^{(k)}\right\| \le \tau_k$
- duality gap: $\left(\mathbf{y}^{(k)}\right)^T \mathbf{s}^{(k)} \le m\tau_k$

Note: $\tau_k = \sigma_k \mu_k$

Predictor-Corrector Primal-Dual Version

3. Solve

$$\begin{pmatrix} \nabla_{xx}^2 L^{(k)} & 0 & \mathbf{F}^T & \mathbf{A}^T \\ \mathbf{F} & \mathbf{I} & 0 & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ 0 & \mathbf{Y}^{(k)} & \mathbf{S}^{(k)} & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ \mathbf{r}_4 \end{pmatrix}$$

where $\mathbf{r}_1 = -\nabla f_0(\mathbf{x}^{(k)}) - \sum_{i=1}^{m} y_i^{(k)} \nabla f_i(\mathbf{x}^{(k)}) - \mathbf{A}^T \mathbf{z}^{(k)}$; $\mathbf{r}_2 = -\mathbf{f}(\mathbf{x}^{(k)}) - \mathbf{s}^{(k)}$;

and $\mathbf{r}_3 = -\mathbf{A}\mathbf{x}^{(k)}$; $\mathbf{r}_4 = -\mathbf{Y}^{(k)}\mathbf{S}^{(k)}\mathbf{e}$

to get predicted Newton's direction $\begin{pmatrix} \Delta\mathbf{x}^{aff} \\ \Delta\mathbf{s}^{aff} \\ \Delta\mathbf{y}^{aff} \\ \Delta\mathbf{z}^{aff} \end{pmatrix}$

Predictor-Corrector Primal-Dual Version

4. Compute predicted stepsizes: $\alpha_{aff}^{primal} = \min\left(1, \min_{i:\Delta s_i^{aff}<0} \frac{-s_i^{(k)}}{\Delta s_i^{aff}}\right);$

$$\alpha_{aff}^{dual} = \min\left(1, \min_{i:\Delta z_i^{aff}<0} \frac{-y_i^{(k)}}{\Delta y_i^{aff}}\right)$$

Then $\alpha_{aff} = \min\left(\alpha_{aff}^{primal}, \alpha_{aff}^{dual}\right)$

Compute $\mathbf{x}^{aff} = \mathbf{x}^{(k)} + \alpha_{aff}\Delta\mathbf{x}^{aff}; \mathbf{s}^{aff} = \mathbf{s}^{(k)} + \alpha_{aff}\Delta\mathbf{s}^{aff}$

$\mathbf{y}^{aff} = \mathbf{y}^{(k)} + \alpha_{aff}\Delta\mathbf{y}^{aff}; \mathbf{z}^{aff} = \mathbf{z}^{(k)} + \alpha_{aff}\Delta\mathbf{z}^{aff}; \mathbf{w}^{aff} = \mathbf{w}^{(k)} + \alpha_{aff}\Delta\mathbf{w}^{aff}$

Compute estimated duality gap measure $\mu_{aff} = \frac{\left(\mathbf{y}^{aff}\right)^T \mathbf{s}^{aff}}{m};$

and $\mu_k = \frac{\left(\mathbf{y}^{(k)}\right)^T \mathbf{s}^{(k)}}{m}$

and estimated centering parameter $\sigma_k = \left(\frac{\mu_{aff}}{\mu_k}\right)^3$

Predictor-Corrector Primal-Dual Version

5. Solve

$$\begin{pmatrix} \nabla_{xx}^2 L^{(k)} & 0 & \mathbf{F}^T & \mathbf{A}^T \\ \mathbf{F} & \mathbf{I} & 0 & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ 0 & \mathbf{Y}^{(k)} & \mathbf{S}^{(k)} & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ \mathbf{r}_4 \end{pmatrix}$$

where $\mathbf{r}_1 = -\nabla f_0(\mathbf{x}^{(k)}) - \sum_{i=1}^{m} y_i^{(k)} \nabla f_i(\mathbf{x}^{(k)}) - \mathbf{A}^T\mathbf{z}^{(k)}$; $\mathbf{r}_2 = -\mathbf{f}(\mathbf{x}^{(k)}) - \mathbf{s}^{(k)}$;

and $\mathbf{r}_3 = -\mathbf{A}\mathbf{x}^{(k)}$; $\mathbf{r}_4 = -\mathbf{Y}^{(k)}\mathbf{S}^{(k)}\mathbf{e} - \Delta\mathbf{Y}_{aff}\Delta\mathbf{S}_{aff}\mathbf{e} + \sigma_k\mu_k\mathbf{e}$

to get corrected centering direction $\begin{pmatrix} \Delta\mathbf{x}^{(k)} \\ \Delta\mathbf{s}^{(k)} \\ \Delta\mathbf{y}^{(k)} \\ \Delta\mathbf{z}^{(k)} \end{pmatrix}$

Predictor-Corrector Primal-Dual Version

6. Compute full stepsizes: $\alpha_{\max} = \min\left(1, \min_{i:\Delta s_i^{aff}<0} \frac{-s_i^{(k)}}{\Delta s_i^{(k)}}, \min_{i:\Delta z_i^{aff}<0} \frac{-y_i^{(k)}}{\Delta y_i^{(k)}}\right)$

Use the shortened stepsizes to ensure strict interior i.e. $\mathbf{s}^{(k)} > 0$ and $\mathbf{y}^{(k)} > 0$):

$\alpha_k = \min(1, \eta\alpha_{\max})$ where $0.9 \le \eta < 1$

Compute $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \Delta\mathbf{x}^{(k)}$; $\mathbf{s}^{(k+1)} = \mathbf{s}^{(k)} + \alpha_k \Delta\mathbf{s}^{(k)}$

$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \alpha_k \Delta\mathbf{y}^{(k)}$; $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \alpha_k \Delta\mathbf{z}^{(k)}$

Repeat Step 2.

Note that because of the coupling between the primal and dual variables through (1), a common step-size must be used in steps 4 and 6 above.

Implementation

Again, the most expensive steps are Steps 3 and 5, which involve solving a system of linear equations of the form:

$$\begin{pmatrix} \nabla_{xx}^2 L^{(k)} & 0 & \mathbf{F}^T & \mathbf{A}^T \\ \mathbf{F} & \mathbf{I} & 0 & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ 0 & \mathbf{Y}^{(k)} & \mathbf{S}^{(k)} & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ \mathbf{r}_4 \end{pmatrix}$$

1. Note that due to convexity, $\nabla_{xx}^2 L^{(k)}$ is *psd*. This along with the assumed "strict feasibility", strong duality holds and the system above always has a solution. In addition, the direction generated should be a descent direction (i.e. the merit function deceases along the generated direction.) So the line search used which is a simple form of backtracking line search should produces a good acceptable size.
2. As before, one can use the last rows to eliminate $\Delta\mathbf{z}$ to get a reduced system which can be solved using symmetric indefinite factorization. See the next slide.

Implementation

The most effective ways to solve the above system begin with the elimination of $\Delta\mathbf{z}$ yielding the augmented systems:

$$\begin{pmatrix} \nabla^2_{xx}L^{(k)} & \mathbf{F}^T & \mathbf{A}^T \\ \mathbf{F} & -\mathbf{Y}^{-1}\mathbf{S} & 0 \\ \mathbf{A} & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 - \mathbf{Y}^{-1}\mathbf{r}_4 \\ \mathbf{r}_3 \end{pmatrix}$$

Clearly the coefficient matrix is symmetric and sparse (if A is sparse and each f_i depends on only a few variable).

The augmented system can be solved efficiently using the *sparse symmetric indefinite factorization* as discussed earlier.

8/6/2009 75

Implementation

Further elimination of $\Delta\mathbf{z}$ yields the a more compact augmented system:

$$\begin{pmatrix} \nabla^2_{xx}L^{(k)} + \mathbf{F}^T\mathbf{Y}\mathbf{S}^{-1}\mathbf{F} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_1 + \mathbf{F}^T\mathbf{Y}\mathbf{S}^{-1}\mathbf{r}_2 - \mathbf{F}^T\mathbf{S}^{-1}\mathbf{r}_4 \\ \mathbf{r}_2 - \mathbf{Y}^{-1}\mathbf{r}_4 \end{pmatrix}$$

Again the coefficient matrix is symmetric and sparse (if Q, A, G are).

The augmented system can be solved efficiently using the *sparse symmetric indefinite factorization*.

If A=0, the above system is a normal equation with positive definite coefficient which can be solved using Cholesky (or sparse Cholesky) factorization, or by the Conjugate Gradient method or projected Conjugate gradient method.

8/6/2009 76