

Recent Developments in Methods for Nonlinear Optimization

Philip E. Gill

UC San Diego

2011 PEER Annual Meeting
Berkeley, California, September 30–October 1, 2011

From Wikipedia, 2011:

*“In mathematics and computational science, **mathematical optimization** (alternatively, **optimization** or **mathematical programming**) refers to the selection of a best element from some set of available alternatives.*

In the simplest case, this means solving problems in which one seeks to maximize (or to minimize) a real function by systematically choosing the values of real or integer variables from within an allowed set. . . ”

Continuous nonlinear optimization

Given functions that define $f(x)$ and $c(x)$ (and their derivatives) at any x , solve

$$\begin{array}{ll} \text{minimize} & f(x) \\ & x \in \mathbb{R}^n \\ \text{subject to} & \ell \leq \left\{ \begin{array}{c} x \\ c(x) \\ Ax \end{array} \right\} \leq u \end{array}$$

The class of optimization problems considered in this talk:

- f and c are arbitrary *smooth* functions
- n and m are *large* (maybe in the tens or hundreds of thousands)
- A and the derivatives of f and c are *sparse*

A trick learned from LP—add slack variables

$$\begin{array}{ll}
 \text{minimize} & f(x) \\
 \text{subject to} & c(x) - s_C = 0, \quad Ax - s_A = 0 \\
 & \ell \leq \left\{ \begin{array}{c} x \\ s_C \\ s_A \end{array} \right\} \leq u
 \end{array}$$

Without loss of generality, we focus on the generic problem:

$$\begin{array}{ll}
 \text{minimize} & f(x) \\
 \text{subject to} & c(x) = 0, \quad x \geq 0
 \end{array}$$

Some applications...

- Optimal design of tensegrity structures
- Optimal control with PDE or ODE constraints
 - optimization of space vehicles, satellites, etc
 - shape optimization
 - inverse problems in medicine and geophysics
- Model parameter synchronization
- Robot path planning

Two fundamental classes of method

- The field of numerical optimization has been in existence since 1948
- Over this time, two fundamentally different approaches have emerged:
 - *Active-set methods* and *path-following methods*
- Two successful representatives from these approaches are:
 - *Sequential quadratic Programming (SQP)* methods
 - *Interior methods*
- The balance of “superiority” is constantly shifting

Outline

- 1 Context
- 2 Overview of SQP methods
- 3 Overview of interior methods
- 4 The interior-point ascendancy
- 5 The Empire strikes back...
- 6 The future ...

Outline

- 1 Context
- 2 Overview of SQP methods
- 3 Overview of interior methods
- 4 The interior-point ascendancy
- 5 The Empire strikes back...
- 6 The future ...

Outline

- 1 Context
- 2 Overview of SQP methods
- 3 Overview of interior methods
- 4 The interior-point ascendancy
- 5 The Empire strikes back...
- 6 The future ...

Outline

- 1 Context
- 2 Overview of SQP methods
- 3 Overview of interior methods
- 4 The interior-point ascendancy
- 5 The Empire strikes back...
- 6 The future ...

Outline

- 1 Context
- 2 Overview of SQP methods
- 3 Overview of interior methods
- 4 The interior-point ascendancy
- 5 The Empire strikes back...
- 6 The future ...

Outline

- 1 Context
- 2 Overview of SQP methods
- 3 Overview of interior methods
- 4 The interior-point ascendancy
- 5 The Empire strikes back...
- 6 The future ...

SQP Methods

Overview of SQP methods

We start with only *equality constraints*:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0$$

The gradient of the Lagrangian function $\mathcal{L}(x, \pi) = f(x) - c(x)^T \pi$ is:

$$\nabla \mathcal{L}(x, \pi) = \begin{pmatrix} g(x) - J(x)^T \pi \\ -c(x) \end{pmatrix}$$

with $g(x)$ the gradient of f , and $J(x)$ the constraint Jacobian.

An optimal solution (x^*, π^*) is a *stationary point* of $\mathcal{L}(x, \pi)$, i.e.,

$$\nabla \mathcal{L}(x^*, \pi^*) = 0$$

Overview of SQP methods

We start with only *equality constraints*:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0$$

The gradient of the Lagrangian function $\mathcal{L}(x, \pi) = f(x) - c(x)^T \pi$ is:

$$\nabla \mathcal{L}(x, \pi) = \begin{pmatrix} g(x) - J(x)^T \pi \\ -c(x) \end{pmatrix}$$

with $g(x)$ the gradient of f , and $J(x)$ the constraint Jacobian.

An optimal solution (x^*, π^*) is a **stationary point** of $\mathcal{L}(x, \pi)$, i.e.,

$$\nabla \mathcal{L}(x^*, \pi^*) = 0$$

The vector (x^*, π^*) solves the **nonlinear equations**

$$\nabla \mathcal{L}(x, \pi) = \begin{pmatrix} g(x) - J(x)^T \pi \\ -c(x) \end{pmatrix} = 0$$

$n + m$ nonlinear equations in the $n + m$ variables x and π .

Apply *Newton's method* to find a solution of $\nabla \mathcal{L}(x, \pi) = 0$.

Newton's method converges at a *second-order rate*.

The vector (x^*, π^*) solves the **nonlinear equations**

$$\nabla \mathcal{L}(x, \pi) = \begin{pmatrix} g(x) - J(x)^T \pi \\ -c(x) \end{pmatrix} = 0$$

$n + m$ nonlinear equations in the $n + m$ variables x and π .

Apply **Newton's method** to find a solution of $\nabla \mathcal{L}(x, \pi) = 0$.

Newton's method converges at a **second-order rate**.

Newton's method

$$\left(\text{"Jacobian"} \right) \left(\begin{array}{l} \text{"Change in} \\ \text{variables"} \end{array} \right) = - \left(\text{"Residual"} \right)$$

These equations are:

$$\begin{pmatrix} H(x_k, \pi_k) & -J(x_k)^T \\ -J(x_k) & 0 \end{pmatrix} \begin{pmatrix} p_k \\ q_k \end{pmatrix} = - \begin{pmatrix} g(x_k) - J(x_k)^T \pi_k \\ -c(x_k) \end{pmatrix}$$

with $H(x, \pi) = \nabla_{xx}^2 \mathcal{L}(x, \pi)$, the *Lagrangian Hessian*.

The next Newton estimate is $(x_k + p_k, \pi_k + q_k)$.

Newton's method

$$\left(\text{"Jacobian"} \right) \left(\begin{array}{c} \text{"Change in"} \\ \text{variables"} \end{array} \right) = - \left(\text{"Residual"} \right)$$

These equations are:

$$\begin{pmatrix} H(x_k, \pi_k) & -J(x_k)^T \\ -J(x_k) & 0 \end{pmatrix} \begin{pmatrix} p_k \\ q_k \end{pmatrix} = - \begin{pmatrix} g(x_k) - J(x_k)^T \pi_k \\ -c(x_k) \end{pmatrix}$$

with $H(x, \pi) = \nabla_{xx}^2 \mathcal{L}(x, \pi)$, the *Lagrangian Hessian*.

The next Newton estimate is $(x_k + p_k, \pi_k + q_k)$.

Equivalence to quadratic programming

(p_k, q_k) is the primal-dual solution of the *quadratic subproblem*:

$$\begin{array}{ll} \underset{p}{\text{minimize}} & g(x_k)^T p + \frac{1}{2} p^T H(x_k, \pi_k) p \\ \text{subject to} & c(x_k) + J(x_k) p = 0 \end{array}$$

The iterates $\{(x_k, \pi_k)\}$ converge at a *second-order rate*.

Inequality constraints

Given (x_k, π_k) , the SQP subproblem is:

$$\begin{array}{ll} \underset{p}{\text{minimize}} & g(x_k)^T p + \frac{1}{2} p^T H(x_k, \pi_k) p \\ \text{subject to} & c(x_k) + J(x_k) p = 0, \quad x_k + p \geq 0 \end{array}$$

The QP is solved using an iterative method.

⇒ inner/outer iteration structure.

Inequality constraints

Given (x_k, π_k) , the SQP subproblem is:

$$\begin{aligned} & \underset{p}{\text{minimize}} && g(x_k)^T p + \frac{1}{2} p^T H(x_k, \pi_k) p \\ & \text{subject to} && c(x_k) + J(x_k) p = 0, \quad x_k + p \geq 0 \end{aligned}$$

The QP is solved using an iterative method.

⇒ inner/outer iteration structure.

The QP optimality conditions imply

$$\begin{pmatrix} H_F & -J_F^T \\ -J_F & 0 \end{pmatrix} \begin{pmatrix} p_F \\ q_k \end{pmatrix} = - \begin{pmatrix} (g(x_k) - J(x_k)^T \pi_k)_F \\ -c(x_k) \end{pmatrix}$$

where “ F ” denotes elements associated with the free variables

⇒ SQP is Newton's method on the free variables.

The QP optimality conditions imply

$$\begin{pmatrix} H_F & -J_F^T \\ -J_F & 0 \end{pmatrix} \begin{pmatrix} p_F \\ q_k \end{pmatrix} = - \begin{pmatrix} (g(x_k) - J(x_k)^T \pi_k)_F \\ -c(x_k) \end{pmatrix}$$

where “ F ” denotes elements associated with the free variables

⇒ SQP is Newton's method on the free variables.

QP methods

A sequence of equality-constraint QPs is solved, each defined by fixing a **subset** of the variables on their bounds.

Sequence of related systems with matrix

$$K_F = \begin{pmatrix} H_F & -J_F^T \\ -J_F & 0 \end{pmatrix}$$

At each QP step, K_F changes by a row and column.

These changes are reflected in a *factorization* of K_F .

QP methods

A sequence of equality-constraint QPs is solved, each defined by fixing a **subset** of the variables on their bounds.

Sequence of related systems with matrix

$$K_F = \begin{pmatrix} H_F & -J_F^T \\ -J_F & 0 \end{pmatrix}$$

At each QP step, K_F changes by a row and column.

These changes are reflected in a **factorization** of K_F .

If the fixed set from one QP is used to start the next QP, the subproblems usually require one QP iteration near the solution.

With a good starting point, SQP requires very few QP iterations

Issues associated with the formulation of SQP methods:

- Global convergence
 - Is (x_{k+1}, π_{k+1}) “better” than (x_k, π_k) ?
- Ill-posed subproblems
 - QP subproblem may not be feasible
 - Ill-conditioned or singular equations
- Computational efficiency
 - Sequence of linear equations with *changing structure*
- Nonconvex problems
 - Replace QP by a “convexified” QP

“Pros and cons” of SQP methods

SQP methods are . . .

- difficult to implement with second derivatives
- very efficient for a sequence of related problems
- good at certifying infeasible problems
- based on customized matrix updating methods
 - very reliable
 - hard to change for new computer architectures

Interior Methods

The optimality conditions are:

$$\begin{aligned}g(x) - J(x)^T \pi - z &= 0, \\c(x) &= 0, & x &\geq 0 \\x \cdot z &= 0, & z &\geq 0\end{aligned}$$

where $x \cdot z$ is a vector with components $x_j z_j$.

$$\begin{aligned}g(x) - J(x)^T \pi - z &= 0, \\c(x) &= 0, & x &\geq 0 \\x \cdot z &= 0, & z &\geq 0\end{aligned}$$

These conditions define nonlinear equations $F(x, \pi, z) = 0$, with

$$F(x, \pi, z) = \begin{pmatrix} g(x) - J(x)^T \pi - z \\ c(x) \\ x \cdot z \end{pmatrix}$$

These are $2n + m$ nonlinear equations in the $2n + m$ unknowns (x, π, z) . Newton's method?

Define a small positive μ (say $\mu = 10^{-2}$).

Perturb the optimality conditions to give

$$\begin{aligned}g(x) - J(x)^T \pi - z &= 0, \\c(x) &= 0, & x &\geq 0 \\x \cdot z &= \mu e, & z &\geq 0\end{aligned}$$

where e is the n -vector of ones $(1, 1, \dots, 1)$.

The perturbation forces x_i and z_i to satisfy $x_i > 0$ and $z_i > 0$.

The *Newton equations* for the change to a given (x, π, z) are:

$$\begin{pmatrix} H & -J^T & -l \\ J & 0 & 0 \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} = - \begin{pmatrix} g - J^T \pi - z \\ c \\ x \cdot z - \mu e \end{pmatrix}$$

where $X = \text{diag}(x_1, x_2, \dots, x_n)$, $Z = \text{diag}(z_1, z_2, \dots, z_m)$.

The next iterate is

$$\begin{pmatrix} x \\ \pi \\ z \end{pmatrix} + \alpha \begin{pmatrix} p \\ q \\ r \end{pmatrix} \quad \text{where} \quad x + \alpha p > 0 \quad \text{and} \quad z + \alpha r > 0$$

Once an *approximate solution* of $F_\mu(x, \pi, z) = 0$ is found, μ is reduced and the process is repeated.

The *Newton equations* for the change to a given (x, π, z) are:

$$\begin{pmatrix} H & -J^T & -I \\ J & 0 & 0 \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} = - \begin{pmatrix} g - J^T \pi - z \\ c \\ x \cdot z - \mu e \end{pmatrix}$$

where $X = \text{diag}(x_1, x_2, \dots, x_n)$, $Z = \text{diag}(z_1, z_2, \dots, z_m)$.

The next iterate is

$$\begin{pmatrix} x \\ \pi \\ z \end{pmatrix} + \alpha \begin{pmatrix} p \\ q \\ r \end{pmatrix} \quad \text{where} \quad x + \alpha p > 0 \quad \text{and} \quad z + \alpha r > 0$$

Once an *approximate solution* of $F_\mu(x, \pi, z) = 0$ is found, μ is reduced and the process is repeated.

Symmetric form of the Newton equations

$$\begin{pmatrix} H & -J^T & -I \\ J & 0 & 0 \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} = - \begin{pmatrix} g - J^T \pi - z \\ c \\ x \cdot z - \mu e \end{pmatrix}$$

Eliminating r gives

$$\begin{pmatrix} H + D & -J^T \\ -J & 0 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = - \begin{pmatrix} g - \mu X^{-1} e - J^T \pi \\ -c \end{pmatrix}$$

with $r = z - \mu X^{-1} e - Dp$, and $D = X^{-1}Z$.

Issues associated with the formulation of interior methods:

- Global convergence
 - Is $(x_{k+1}, \pi_{k+1}, z_{k+1})$ “better” than (x_k, π_k, z_k) ?
- Ill-posed subproblems
 - the problem may not be feasible
 - Ill-conditioned or singular equations
- Computational efficiency
 - linear equations with *fixed structure* are solved *from scratch*
- Nonconvex problems
 - Use Newton system from a “convexified” problem

“Pros and cons” of interior methods

Interior methods ...

- are best when second derivatives are provided
- have difficulty exploiting a good solution
- have difficulty certifying infeasible constraints
- solve a sequence of systems with *fixed structure*
 - they can exploit solvers designed for modern computer architectures
 - reliability depends on the solver
- factor a matrix with every constraint present
- *are blazingly fast on one-off problems that aren't too hard*

The interior-point ascendancy...

The early 2000s saw a shift in the relative efficiency of interior methods and SQP methods

Two reasons:

- The advent of efficient automatic differentiation packages
 - many modeling languages started to provide second derivatives automatically
- The evolution of computer architectures

The early 2000s saw a shift in the relative efficiency of interior methods and SQP methods

Two reasons:

- The advent of efficient automatic differentiation packages
 - many modeling languages started to provide second derivatives automatically
- The evolution of computer architectures

Efficient software for linear equations

Computer hardware is changing

- Moore's Law is fading

"The number of transistors on a microchip will double every 18 months"

- Moore's Law has been "updated":

"the number of cores (cpus) on a processor will double every 18 months"

- it's already happening...

- 2008 Mac G5: 4 quad-core processors = 16 cpus
- 2011 Mac Book: dual 16-core processors = 32 cpus
- 2013 dual 132-core = 264 cpus
- > 2008 potentially hundreds of cpus using GPUs

Efficient software for linear equations

Computer hardware is changing

- Moore's Law is fading

"The number of transistors on a microchip will double every 18 months"

- Moore's Law has been "updated":

"the number of cores (cpus) on a processor will double every 18 months"

- it's already happening...

- 2008 Mac G5: 4 quad-core processors = 16 cpus
- 2011 Mac Book: dual 16-core processors = 32 cpus
- 2013 dual 132-core = 264 cpus
- > 2008 potentially hundreds of cpus using GPUs

Efficient software for linear equations

Computer hardware is changing

- Moore's Law is fading

"The number of transistors on a microchip will double every 18 months"

- Moore's Law has been "updated":

"the number of cores (cpus) on a processor will double every 18 months"

- it's already happening...

- 2008 Mac G5: 4 quad-core processors = 16 cpus
- 2011 Mac Book: dual 16-core processors = 32 cpus
- 2013 dual 132-core = 264 cpus
- > 2008 potentially hundreds of cpus using GPUs

Efficient software for linear equations

Computer hardware is changing

- Moore's Law is fading

"The number of transistors on a microchip will double every 18 months"

- Moore's Law has been "updated":

"the number of cores (cpus) on a processor will double every 18 months"

- it's already happening...

- 2008 Mac G5: 4 quad-core processors = 16 cpus
- 2011 Mac Book: dual 16-core processors = 32 cpus
- 2013 dual 132-core = 264 cpus
- > 2008 potentially hundreds of cpus using GPUs

20 years of progress

PILOT 1442 rows, 3652 columns, 43220 nonzeros

| Year | Itns | Cpu secs | Architecture |
|------|-------|-------------------|--------------------------|
| 1987 | – | 8.7×10^4 | DEC Vaxstation II |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 2005 | 17738 | 22.2 | dual-core Xeon |
| 2006 | 16865 | 9.7 | dual-core Opteron 2.4Ghz |
| 2007 | 16865 | 8.1 | dual-core Opteron 3.1Ghz |
| 2008 | 16865 | 8.7 | quad-core Opteron 3.1Ghz |

ILOG CPLEX interior method (2009): 2 secs

20 years of progress

PILOT 1442 rows, 3652 columns, 43220 nonzeros

| Year | Itns | Cpu secs | Architecture |
|------|-------|-------------------|--------------------------|
| 1987 | – | 8.7×10^4 | DEC Vaxstation II |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 2005 | 17738 | 22.2 | dual-core Xeon |
| 2006 | 16865 | 9.7 | dual-core Opteron 2.4Ghz |
| 2007 | 16865 | 8.1 | dual-core Opteron 3.1Ghz |
| 2008 | 16865 | 8.7 | quad-core Opteron 3.1Ghz |

ILOG CPLEX interior method (2009): 2 secs

20 years of progress

PILOT 1442 rows, 3652 columns, 43220 nonzeros

| Year | Itns | Cpu secs | Architecture |
|------|-------|-------------------|--------------------------|
| 1987 | – | 8.7×10^4 | DEC Vaxstation II |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 2005 | 17738 | 22.2 | dual-core Xeon |
| 2006 | 16865 | 9.7 | dual-core Opteron 2.4Ghz |
| 2007 | 16865 | 8.1 | dual-core Opteron 3.1Ghz |
| 2008 | 16865 | 8.7 | quad-core Opteron 3.1Ghz |

ILOG CPLEX interior method (2009): 2 secs

The custom-built updating methods used by SQP are difficult to modernize or adapt to matrices with special structure.

Interior methods can use “off-the-shelf” solvers.

- solvers designed for structured systems and/or advanced architectures

Currently, interior methods are generally faster for “one-off” problems.

The Empire strikes back...

Many important applications require the solution of a *sequence of related optimization problems*, e.g.,

- ODE and PDE-based optimization with mesh refinement
- Real-time optimal control

The common feature is that we would like to benefit from *good approximate solutions*.

Many important applications require the solution of a *sequence of related optimization problems*, e.g.,

- ODE and PDE-based optimization with mesh refinement
- Real-time optimal control

The common feature is that we would like to benefit from *good approximate solutions*.

Shifting from updating to factorization

Reformulate the QP method to shift the emphasis from *sparse matrix updating* to *sparse matrix factorization*.

The QP equations may be written in the form $K_j v = f$, with

$$\begin{pmatrix} K_0 & W \\ W^T & D \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

ONE solve with dense Schur-complement $C = D - W^T K_0^{-1} W$

TWO solves with the sparse/structured K_0

Shifting from updating to factorization

Reformulate the QP method to shift the emphasis from *sparse matrix updating* to *sparse matrix factorization*.

The QP equations may be written in the form $K_j v = f$, with

$$\begin{pmatrix} K_0 & W \\ W^T & D \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

ONE solve with dense Schur-complement $C = D - W^T K_0^{-1} W$

TWO solves with the sparse/structured K_0

What is possible now?

- **General-purpose** codes for nonlinear optimization
 - up to 50,000 variables, 10,000 constraints
 - requires sparse Hessian and Jacobian
 - active-set methods vs “interior” methods
- **Special-purpose** codes for nonlinear optimization
 - up to 22 million variables
 - requires parallel processing

The future. . .

New developments

Combine the best features of interior and SQP methods

- “Crossover” methods
- Iterative methods with active-set preconditioning
- SQP methods with automatic regularization

Thanks for listening!

References

Anders Forsgren, Philip E. Gill & Margaret H. Wright, *Interior Methods for Nonlinear Optimization*, SIAM Review, **44**, 525–597, 2002.

Philip E. Gill & Elizabeth Wong, *Sequential Quadratic Programming Methods*, in J. Lee & S. Leyffer (eds.), *Mixed-Integer Nonlinear Optimization: Algorithmic Advances and Applications*, The IMA Volumes in Mathematics and its Applications, Springer Verlag, Berlin, Heidelberg and New York, 2011.

Philip E. Gill & Daniel P. Robinson, *Regularized primal-dual sequential quadratic programming methods*, Report NA 11-1, Department of Mathematics, University of California, San Diego, 2011.