Implementation of a KKT-based active-set QP solver

ISMP 2006

19th International Symposium on Mathematical Programming

Rio de Janeiro, Brazil, July 30–August 4, 2006

Hanh Huynh and Michael Saunders

SCCM Program
Stanford University
Stanford, CA 94305-9025
hhuynh@stanford.edu

Dept of Management Sci & Eng
Stanford University
Stanford, CA 94305-4026
saunders@stanford.edu

Abstract

Sparse SQP methods such as SNOPT need a QP solver that permits warm starts each major iteration and can handle many degrees of freedom. An active-set QP method with direct KKT solves seems the only option.

We discuss some implementation issues such as updating the KKT factors, scaling the QP Hessian, and recovering from KKT singularity.

Acknowledgements:

Philip Gill, UC San Diego

Comsol, Inc

Why a new QP solver?

SNOPT

Sparse SQP solver for NLP (Gill, Murray & Saunders 2005)

Sequence of QP subproblems:

$$\mathrm{QP}_k \quad \underset{x}{\text{minimize}} \quad g_k^T x + \tfrac{1}{2} x^T H_k x$$
$$\text{subject to} \quad \text{linearized constraints and bounds}$$

Limited-memory quasi-Newton Hessian

$H_0 = I$ or diagonal

$H_1 = (I + vu^T) H_0 (I + uv^T)$, etc

Warm start, few iterations $\Rightarrow$ active-set method SQOPT

SNOPT

Sparse SQP solver for NLP (Gill, Murray & Saunders 2005)

Sequence of QP subproblems:

$$\boxed{\begin{array}{lll} \mathrm{QP}_k & \underset{x}{\text{minimize}} & g_k^T x + \frac{1}{2} x^T H_k x \\ & \text{subject to} & \text{linearized constraints and bounds} \end{array}}$$

Limited-memory quasi-Newton Hessian

$H_0 = I$ or diagonal

$H_1 = (I + vu^T)H_0(I + uv^T)$, etc

Warm start, few iterations $\Rightarrow$ active-set method SQOPT

SQOPT's reduced Hessian $Z^T H_k Z$ can be large

SNOPT

Sparse SQP solver for NLP (Gill, Murray & Saunders 2005)

Sequence of QP subproblems:

$$\boxed{\begin{array}{lll} \mathsf{QP}_k & \underset{x}{\text{minimize}} & g_k^T x + \frac{1}{2} x^T H_k x \\ & \text{subject to} & \text{linearized constraints and bounds} \end{array}}$$

Limited-memory quasi-Newton Hessian

$H_0 = I$ or diagonal

$H_1 = (I + vu^T)H_0(I + uv^T)$, etc

Warm start, few iterations $\Rightarrow$ active-set method SQOPT

SQOPT's reduced Hessian $Z^T H_k Z$ can be large

CG on $Z^T H_k Z$ works unexpectedly well sometimes

SNOPT

Sparse SQP solver for NLP (Gill, Murray & Saunders 2005)

Sequence of QP subproblems:

$$\text{QP}_k \quad \underset{x}{\text{minimize}} \quad g_k^T x + \tfrac{1}{2} x^T H_k x$$
$$\text{subject to} \quad \text{linearized constraints and bounds}$$

Limited-memory quasi-Newton Hessian

$H_0 = I$ or diagonal

$H_1 = (I + vu^T)H_0(I + uv^T)$, etc

Warm start, few iterations $\Rightarrow$ active-set method SQOPT

SQOPT's reduced Hessian $Z^T H_k Z$ can be large

CG on $Z^T H_k Z$ works unexpectedly well sometimes

Need a QP solver that works with KKT systems

Linear Algebra Q1
Updating basis factors

Updating a basis

Aim Use SuperLU, PARDISO, . . . as **black box solvers** for B_0

Product-form update $B_k = B_0 T_1 T_2 \dots T_k$

Simple, but perhaps dense, unstable

Updating a basis

Aim Use SuperLU, PARDISO, ... as **black box solvers** for B_0

Product-form update

$$B_k = B_0 T_1 T_2 \dots T_k$$

Simple, but perhaps dense, unstable

Schur-complement update (Bisschop & Meeraus 1977)

Initially: $$B_0 x = b_0$$

Later: $$\begin{pmatrix} B_0 & V_k \\ W_k^T & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_k \\ 0 \end{pmatrix}$$

2 solves with B_0
1 solve with $C_k = W_k^T B_0^{-1} V_k$ (small),
sparse products $V_k v$, $W_k^T w$

Linear Algebra Q2
Updating KKT factors

Updating KKT systems

for QP active-set solver

Aim Use MA57, PARDISO, . . . as **black box solvers** for K_0

Initially: $$K_0 y = d$$

Later:

$$\begin{pmatrix} K_0 & V_k \\ V_k^T & \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$$

Existing work: Gould & Toint, Gondzio

QPA in GALAHAD

Active-set QP solver (Gould and Toint 2001)

Sequence of updated KKT systems

$$\begin{pmatrix} K_0 & V \\ V^T & \end{pmatrix} = \begin{pmatrix} I & \\ V^T K_0^{-1} & I \end{pmatrix} \begin{pmatrix} K_0 & V \\ & C \end{pmatrix}$$

$K_0 = L_0 D_0 L_0^T$ via MA27 or MA57

$C = -V^T K_0^{-1} V$ factored by SCU (small)

QPA in GALAHAD

Active-set QP solver (Gould and Toint 2001)

Sequence of updated KKT systems

$$\begin{pmatrix} K_0 & V \\ V^T & \end{pmatrix} = \begin{pmatrix} I & \\ V^T K_0^{-1} & I \end{pmatrix} \begin{pmatrix} K_0 & V \\ & C \end{pmatrix}$$

$K_0 = L_0 D_0 L_0^T$ via MA27 or MA57

$C = -V^T K_0^{-1} V$ factored by SCU (small)

2 solves with K_0, 1 solve with C, products Vv, $V^T w$

QPA in GALAHAD

Active-set QP solver (Gould and Toint 2001)

Sequence of updated KKT systems

$$\begin{pmatrix} K_0 & V \\ V^T & \end{pmatrix} = \begin{pmatrix} I & \\ V^T K_0^{-1} & I \end{pmatrix} \begin{pmatrix} K_0 & V \\ & C \end{pmatrix}$$

$K_0 = L_0 D_0 L_0^T$ via MA27 or MA57

$C = -V^T K_0^{-1} V$ factored by SCU (small)

2 solves with K_0, 1 solve with C, products Vv, $V^T w$

1 solve with K_0 if $K_0^{-1} V$ were stored (but it is fairly dense)

Symmetric Block-LU updates

$$\begin{pmatrix} K_0 & V \\ V^T & \end{pmatrix} = \begin{pmatrix} L_0 & \\ Y^T D_0^{-1} & I \end{pmatrix} \begin{pmatrix} D_0 L_0^T & Y \\ & C \end{pmatrix}$$

$$L_0 Y = V, \qquad C = -Y^T D_0^{-1} Y$$

$K_0 = L_0 D_0 L_0^T$ via MA27, MA57, PARDISO, . . .

C factored by LUMOD ($LC = U$, L square, small)

Symmetric Block-LU updates

$$\begin{pmatrix} K_0 & V \\ V^T & \end{pmatrix} = \begin{pmatrix} L_0 & \\ Y^T D_0^{-1} & I \end{pmatrix} \begin{pmatrix} D_0 L_0^T & Y \\ & C \end{pmatrix}$$

$$L_0 Y = V, \qquad C = -Y^T D_0^{-1} Y$$

$K_0 = L_0 D_0 L_0^T$ via MA27, MA57, PARDISO, . . .

C factored by LUMOD ($LC = U$, L square, small)

Solves with L_0, D_0, C, D_0, L_0^T, products with Y, Y^T

MA57 treats L_0, D_0 as **two black boxes** (Thanks Iain!)

Y is likely to be **well-conditioned** and **sparse**

Unsymmetric Block-LU updates

$$\begin{pmatrix} K_0 & V \\ W^T & \end{pmatrix} = \begin{pmatrix} L_0 & \\ Z^T & I \end{pmatrix} \begin{pmatrix} U_0 & Y \\ & C \end{pmatrix}$$

$$\begin{aligned} L_0 Y &= V, \\ U_0^T Z &= W, \\ C &= -Z^T Y \end{aligned}$$

$K_0 = L_0 U_0$ via LUSOL, SuperLU, UMFPACK, PARDISO, ...

Solves with L_0, U_0, C,   products with Y, Z^T

L_0, U_0 are **two black boxes**

Y and Z are likely to be **sparse**

QPBLU

Active-set QP solver (Hanh Huynh's thesis)

QP $\quad \underset{x}{\text{minimize}} \quad c^T x + \frac{1}{2} x^T H x$

$\quad\quad \text{subject to} \quad Ax = b, \quad l \le x \le u$

- MATLAB prototype, F90 version under way
- Block-LU updates of KKT factors
- $K_0 = L_0 U_0$ with black-box solvers for L_0, U_0
- SNOPT's $H_1 = (I + vu^T) H_0 (I + uv^T)$, etc handled by block-LU updates

Experiments with Matlab QPBLU

QP	$\underset{x}{\text{minimize}}$	$c^T x + \frac{1}{2} x^T x$
	subject to	$Ax = b, \quad l \le x \le u$

- QP problems with $H = I$
- A, b, c, l, u come from LPnetlib collection (Tim Davis)
- `[L0,U0,P,Q] = lu(K0)` via UMFPACK (Tim Davis)
- 20 Block-LU updates then factorize current KKT

Block-LU updates

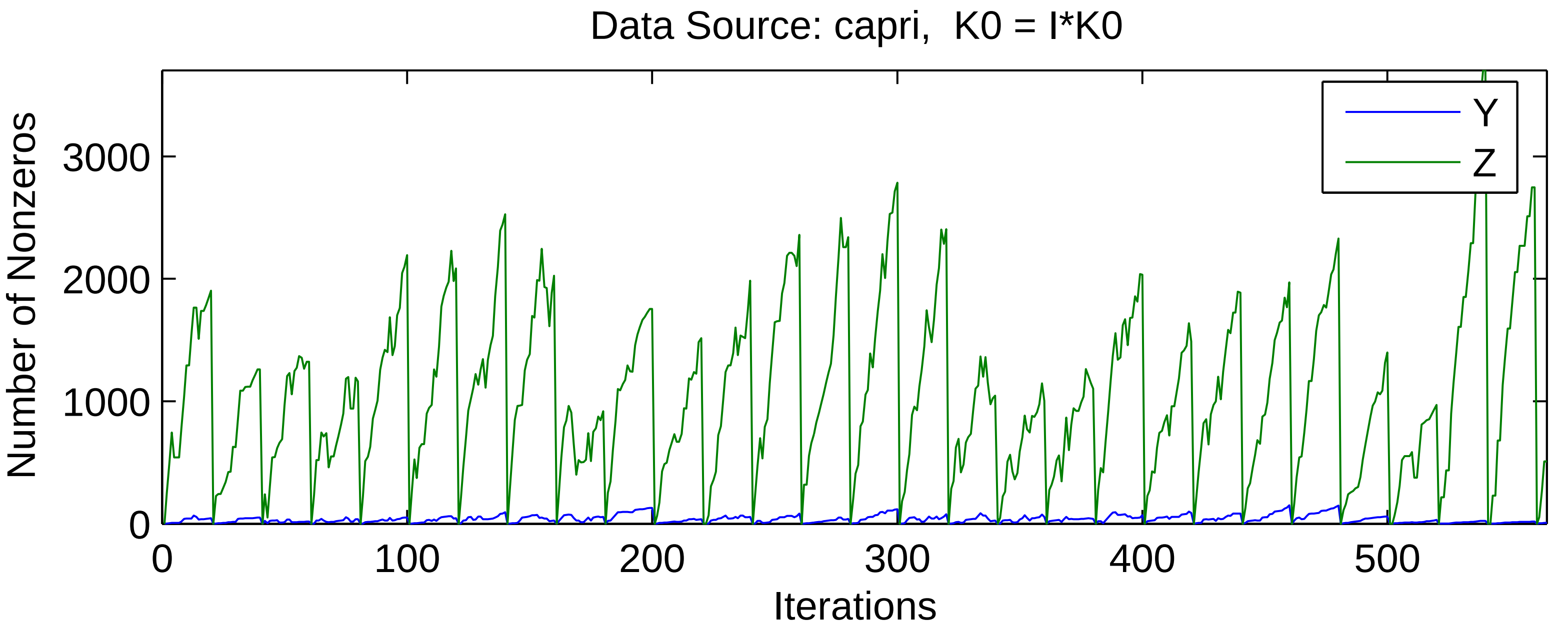
$$\begin{pmatrix} K_0 & V \\ W^T & \end{pmatrix} = \begin{pmatrix} L_0 & \\ Z^T & I \end{pmatrix} \begin{pmatrix} U_0 & Y \\ & C \end{pmatrix}$$

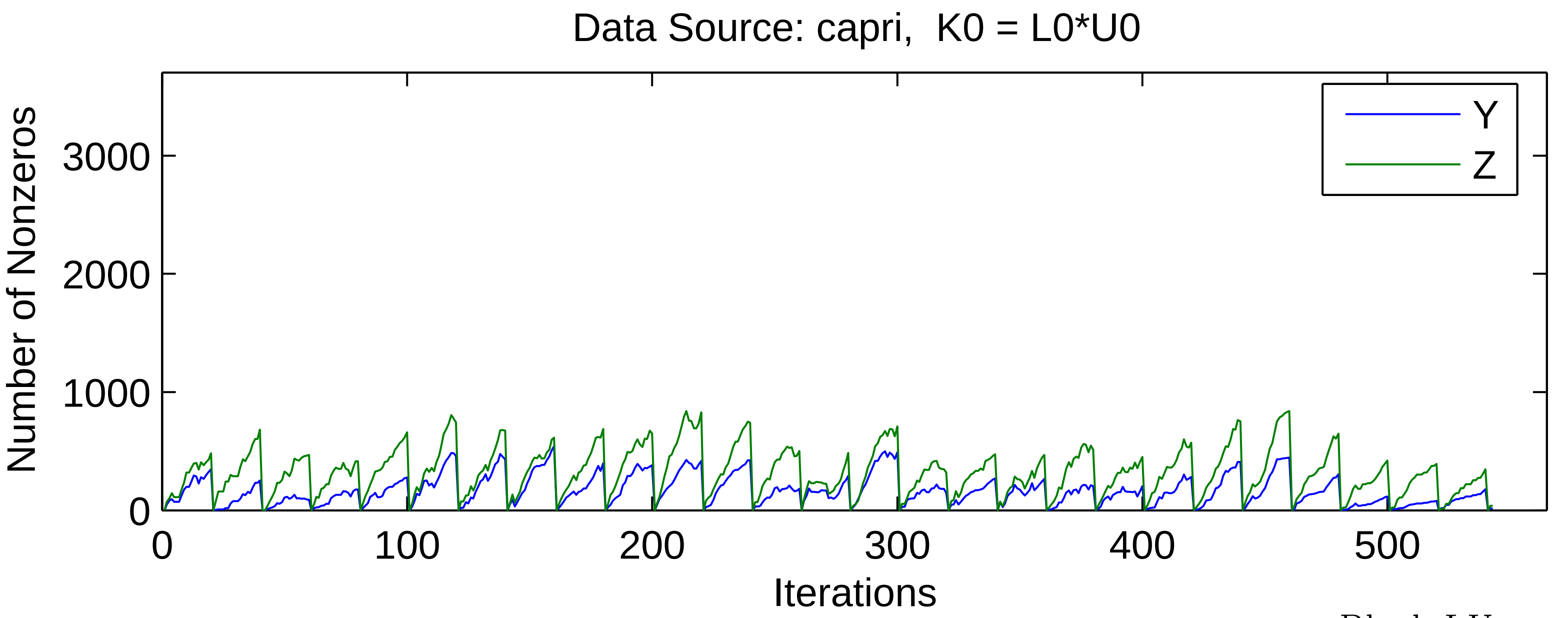
Two possible implementations:

$L_0 = I,\ U_0 = K_0$	Separate L_0 and U_0
$\begin{aligned} Y &= V \\ U_0^T L_0^T Z &= W \end{aligned}$	$\begin{aligned} L_0 Y &= V \\ U_0^T Z &= W \end{aligned}$

Compare nonzeros in Y and Z

Black-box K_0 vs separate L_0, U_0





Linear Algebra Q3
Rank-revealing factors

LUSOL

Three pivoting options:

TPP	Threshold Partial Pivoting
TRP	Threshold Rook Pivoting
TCP	Threshold Complete Pivoting

TPP: Threshold Partial Pivoting

$$
\begin{array}{ccccccc}
\cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\
 & \ddots & & & & & \vdots \\
 & & \ddots & & & & \vdots \\
 & & & 2.0 & \times & & \times \\
 & & & 3.0 & & & \times \\
 & & & & \times & \times & \\
 & & & 4.0 & \times & \times & \times \\
 & & & & \times & & \times
\end{array}
$$

Require $|L_{ij}| \leq \tau$, $\qquad \tau = 2.0$ say (not 1.0)

TRP: Threshold Rook Pivoting

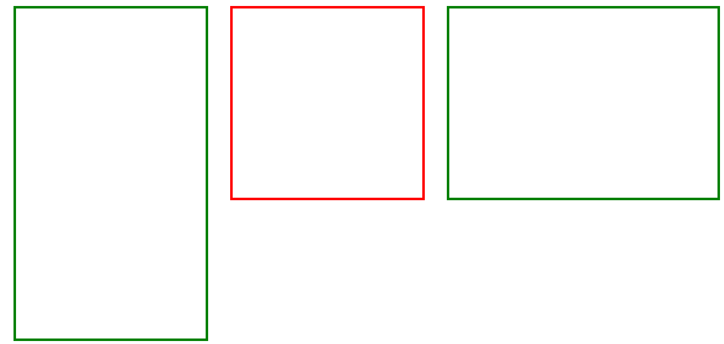
$$
\begin{array}{ccccccc}
\cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\
 & \ddots & & & & & \vdots \\
 & & \ddots & & & & \vdots \\
 & & & {\color{green}4.0} & {\color{blue}1.0} & & {\color{red}8.0} \\
 & & & {\color{blue}3.0} & & & \times \\
 & & & & \times & \times & \\
 & & & {\color{blue}4.0} & \times & \times & \times \\
 & & & & \times & & \times
\end{array}
$$

TCP: Threshold Complete Pivoting

$$
\begin{matrix}
\cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\
 & \ddots & & & & & \vdots \\
 & & \ddots & & & & \vdots \\
 & & & 4.5 & 1.0 & & 8.0 \\
 & & & 3.0 & & & \times \\
 & & & & \times & \times & \\
 & & & 4.0 & \times & \times & 9.0 \\
 & & & & \times & & \times
\end{matrix}
$$

Rank-Revealing Factors

$$A = XDY^T =$$

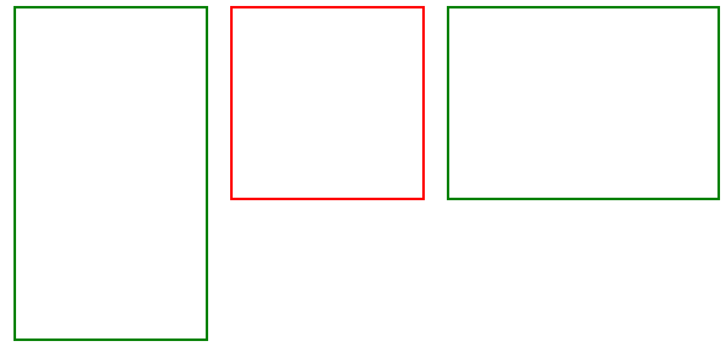


Demmel et al. (1999):

$$X,\ Y \text{ well-conditioned} \quad \Rightarrow \quad \mathrm{cond}(A) \approx \mathrm{cond}(D)$$

- SVD $\qquad UDV^T$
- QR with column interchanges $\qquad QDR$
- LU with Rook Pivoting $\qquad LDU$
- LU with Complete Pivoting $\qquad LDU$

Rank-Revealing Factors

$$A = XDY^T =$$ 

Demmel et al. (1999):

$$X,\ Y \text{ well-conditioned} \;\Rightarrow\; \operatorname{cond}(A) \approx \operatorname{cond}(D)$$

- SVD $\quad UDV^T$
- QR with column interchanges $\quad QDR$
- LU with Rook Pivoting $\quad LDU$
- LU with Complete Pivoting $\quad LDU$
- MA27, MA57 (L_{ij} bounded, D block-diag) $\quad LDL^T$

Linear Algebra Q4
KKT repair

Two-stage KKT repair

$$K = \begin{pmatrix} H & A^T \\ A & \end{pmatrix}$$

`[L,U,p,q] = lusol(A)` with Threshold Rook Pivoting
detects singularities in A

`[L,D,p] = ma57(K)` with strict pivot tolerance
detects singularities in K

Linear Algebra Q5
Condition of K_0

Scaling H

As we know from least squares, $\begin{pmatrix} \alpha I & A^T \\ A & \end{pmatrix}$ is better conditioned if

$$\alpha \approx \sigma_{\min}(A).$$

Hence, QPBLU solves

$$\boxed{\begin{array}{lll} \text{QP} & \underset{x}{\text{minimize}} & \alpha(c^T x + \tfrac{1}{2} x^T H x) + \omega \times \text{suminf} \\ & \text{subject to} & Ax = b, \quad l \le x \le u \end{array}}$$

$K_0 = \begin{pmatrix} \alpha H_0 & A_0^T \\ A_0 & \end{pmatrix}$ is better conditioned (if we can guess good α).

Schur-complements $C_k = -V_k^T K_0^{-1} V_k$ also.

Summary

QPBLU active-set QP solver

KKT solves: $K_0 = \begin{pmatrix} H_0 & A_0^T \\ A_0 & \end{pmatrix}$

Block-LU updates: $\begin{pmatrix} K_0 & V \\ W^T & D \end{pmatrix} = \begin{pmatrix} L_0 & \\ Z^T & I \end{pmatrix} \begin{pmatrix} U_0 & Y \\ & C \end{pmatrix}$

Black-box solvers for $L_0,\ U_0$

$Y,\ Z$ sparse, so worth storing

One hope for parallelism in LP/QP/NLP solvers

Summary

QPBLU active-set QP solver

KKT solves: $K_0 = \begin{pmatrix} H_0 & A_0^T \\ A_0 & \end{pmatrix}$

Block-LU updates: $\begin{pmatrix} K_0 & V \\ W^T & D \end{pmatrix} = \begin{pmatrix} L_0 & \\ Z^T & I \end{pmatrix} \begin{pmatrix} U_0 & Y \\ & C \end{pmatrix}$

Black-box solvers for $L_0,\ U_0$

$Y,\ Z$ sparse, so worth storing

One hope for parallelism in LP/QP/NLP solvers

Sparse LU, LDL' solvers

LUSOL, MA27, MA57 already suitable

Need $|L_{ij}| \le 2$ (say) to be rank-revealing

Summary

QPBLU active-set QP solver

KKT solves: $K_0 = \begin{pmatrix} H_0 & A_0^T \\ A_0 & \end{pmatrix}$

Block-LU updates: $\begin{pmatrix} K_0 & V \\ W^T & D \end{pmatrix} = \begin{pmatrix} L_0 & \\ Z^T & I \end{pmatrix} \begin{pmatrix} U_0 & Y \\ & C \end{pmatrix}$

Black-box solvers for L_0, U_0

Y, Z sparse, so worth storing

One hope for parallelism in LP/QP/NLP solvers

Sparse LU, LDL' solvers

LUSOL, MA27, MA57 already suitable

Need $|L_{ij}| \leq 2$ (say) to be rank-revealing

Request to SuperLU, PARDISO, UMFPACK, ... **developers**

Separate solves with L_0, D_0, U_0

At least one factor well-conditioned (tell us which one!)

Options for rank-revealing factors of K_0