# A convex QP solver based on block-LU updates

**PP06**

**SIAM Conference on Parallel Processing for Scientific Computing**

**San Francisco, CA, Feb 22–24, 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

Active-set methods have an advantage over interior methods in permitting warm starts. We describe a convex QP method intended for use within SNOPT (a sparse SQP package for constrained optimization). An initial KKT system is factorized by any available method (LUSOL, MA57, PARDISO, SuperLU, . . . ). Active-set changes are implemented by block-LU updates that retain sparsity while leaving the original KKT factors intact.

**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 currently solved by SQOPT:

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

- Limited-memory quasi-Newton Hessian
  $H_1 = (I + vu^T) H_0 (I + uv^T)$, etc
- Warm start, few iterations $\quad \Rightarrow$ active-set method
- SQOPT's reduced Hessian $Z^T H_k Z$ can be large
- Need a QP solver that works with KKT systems (like QPA in GALAHAD)

# KKT systems

Active-set QP solvers start with systems of the form

$$K_0 y = d, \qquad K_0 = \begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} = L_0 D_0 {L_0}^T$$

and then add/delete rows and cols of $H$, $A$

Later systems are equivalent to

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

where we want to treat $K_0$ as a black box

# QPA

Active-set QP solver in GALAHAD (Gould and Toint, 2004)

Uses SCU to solve sequence of updated KKT systems

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

SCU maintains factors of Schur complement $C$, where

$$\begin{aligned} K_0 T &= V \\ C &= D - V^T T \end{aligned}$$

- $K_0$ is a black box (QPA uses $L_0 D_0 {L_0}^T$ via MA27 or MA57)
- $T$ is not stored (may be fairly dense)
- One solve with dense $C$, but **two solves with** $K_0$

# Symmetric Block-LU updates

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

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

Solves with $L_0$, $D_0$, $C$, $D_0$, ${L_0}^T$, and products with $Y^T$, $Y$

- $L_0$, $D_0$ are **two black boxes**
  MA57 has separate solves with $L_0$, $D_0$, ${L_0}^T$ (Thanks Iain!)
- $Y$ is likely to be **sparse**
- Small dense $LC = U$ with $L$ square

# Unsymmetric Block-LU updates

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

$$\begin{aligned} K_0 &= L_0 U_0, \\ L_0 Y &= V, \\ {U_0}^T Z &= W, \\ C &= D - Z^T Y \end{aligned}$$

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

- $L_0$, $U_0$ are **two black boxes**
- $Y$ and $Z$ are likely to be **sparse**
- Small dense $LC = U$ with $L$ square
- Same approach for updating simplex-type basis $B_0 = L_0 U_0$

**Special request to PARDISO, SuperLU, . . . developers:**

**Allow separate solves with $L$, $D$, $U$ factors**

(MA57 already does)

# Rank-Revealing Factors

# Partial Pivoting

$$
\begin{array}{ccccccc}
\cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\
 & \ddots & & & & & \vdots \\
 & & \ddots & & & & \vdots \\
 & & & 4.0 & \times & \times & \times \\
 & & & 2.0 & \times & \times & \times \\
 & & & 1.0 & \times & \times & \times \\
 & & & 4.0 & \times & \times & \times \\
 & & & 0.1 & \times & \times & \times
\end{array}
$$

# Rook Pivoting

$$
\begin{matrix}
\cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\
 & \ddots & & & & & \vdots \\
 & & \ddots & & & & \vdots \\
 & & & 6.0 & 1.0 & 0.1 & 6.0 \\
 & & & 2.0 & \times & \times & \times \\
 & & & 1.0 & \times & \times & \times \\
 & & & 4.0 & \times & \times & \times \\
 & & & 0.1 & \times & \times & \times
\end{matrix}
$$

# Complete Pivoting

$$
\begin{array}{ccccccc}
\cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\
 & \ddots & & & & & \vdots \\
 & & \ddots & & & & \vdots \\
 & & & 9.0 & 1.0 & 0.1 & 6.0 \\
 & & & 2.0 & \times & \times & \times \\
 & & & 1.0 & \times & \times & \times \\
 & & & 4.0 & \times & \times & 9.0 \\
 & & & 0.1 & \times & 0.1 & \times
\end{array}
$$

# LUSOL pivoting options

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

# TPP: Threshold Partial Pivoting

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

Require $|L_{ij}| \leq 2.0$ (say)

# TRP: Threshold Rook Pivoting

$$
\begin{array}{ccccccc}
\cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\
 & \ddots & & & & & \vdots \\
 & & \ddots & & & & \vdots \\
 & & & 4.0 & 1.0 & & 7.0 \\
 & & & 2.0 & & & \times \\
 & & & & \times & \times & \\
 & & & 4.0 & \times & \times & \times \\
 & & & & \times & & \times
\end{array}
$$

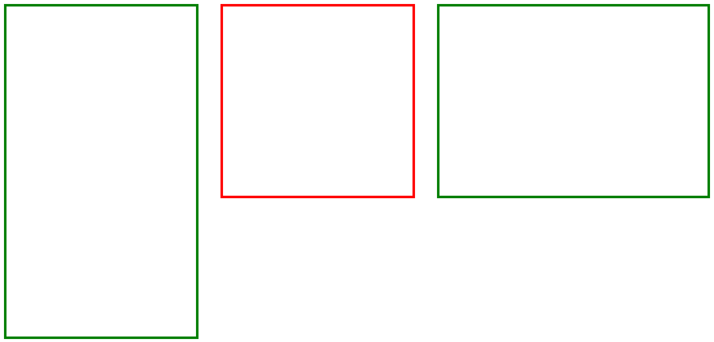
$A = LDU$ Require $|L_{ij}|$ and $|U_{ij}| \le 2.0$ (say)

# TCP: Threshold Complete Pivoting

$$
\begin{array}{ccccccc}
\cdot & \cdots & \cdots & \cdots & \cdots & \cdots & \cdot \\
 & \ddots & & & & & \vdots \\
 & & \ddots & & & & \vdots \\
 & & & 5.0 & 1.0 & & 7.0 \\
 & & & 2.0 & & & \times \\
 & & & & \times & \times & \\
 & & & 4.0 & \times & \times & 9.0 \\
 & & & & \times & & \times
\end{array}
$$

$A = LDU$ Require $|L_{ij}|$ and $|U_{ij}| \leq 2.0$ (say)

# Rank-Revealing Factors

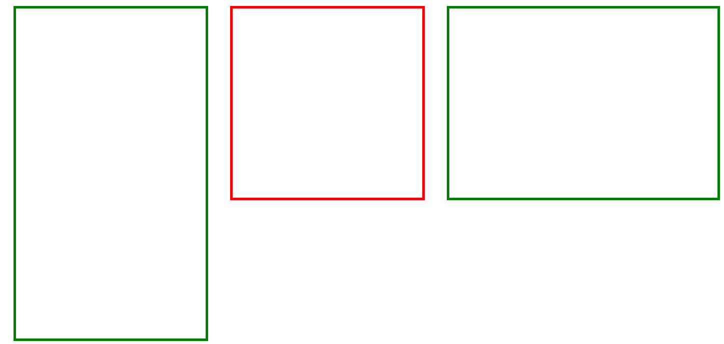
$A = XDY^T =$ 

Demmel et al. (1999):

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

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

# Rank-Revealing Factors

$$A = XDY^T = $$ 

Demmel et al. (1999):

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

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

**Special request to** PARDISO, SuperLU, **. . . developers:**

**Provide options for computing rank-revealing factors**

(LUSOL, MA57 already do)

Need off-diags $\leq 2.5$ say (MA57's pivot tol $\geq 0.4$)

# Numerical results

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

- QP problems with $H = I$
- $A$, $b$, $c$, $l$, $u$ come from LPnetlib collection (Tim Davis)
- MATLAB implementation of proposed QP solver using block-LU updates of KKT system
- `[L0,U0,P,Q] = lu(K0)` via UMFPACK (Tim Davis)
- Update 20 times then refactorize current KKT matrix

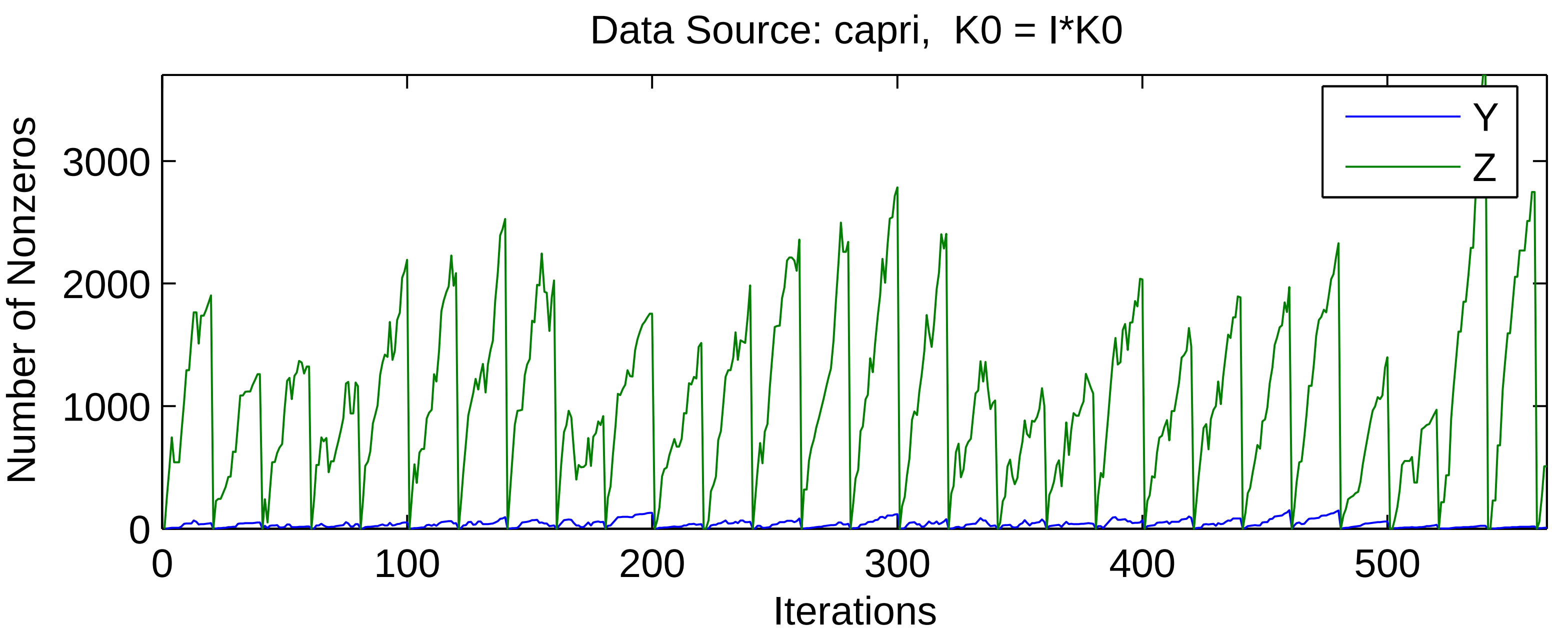
# Block-LU updates

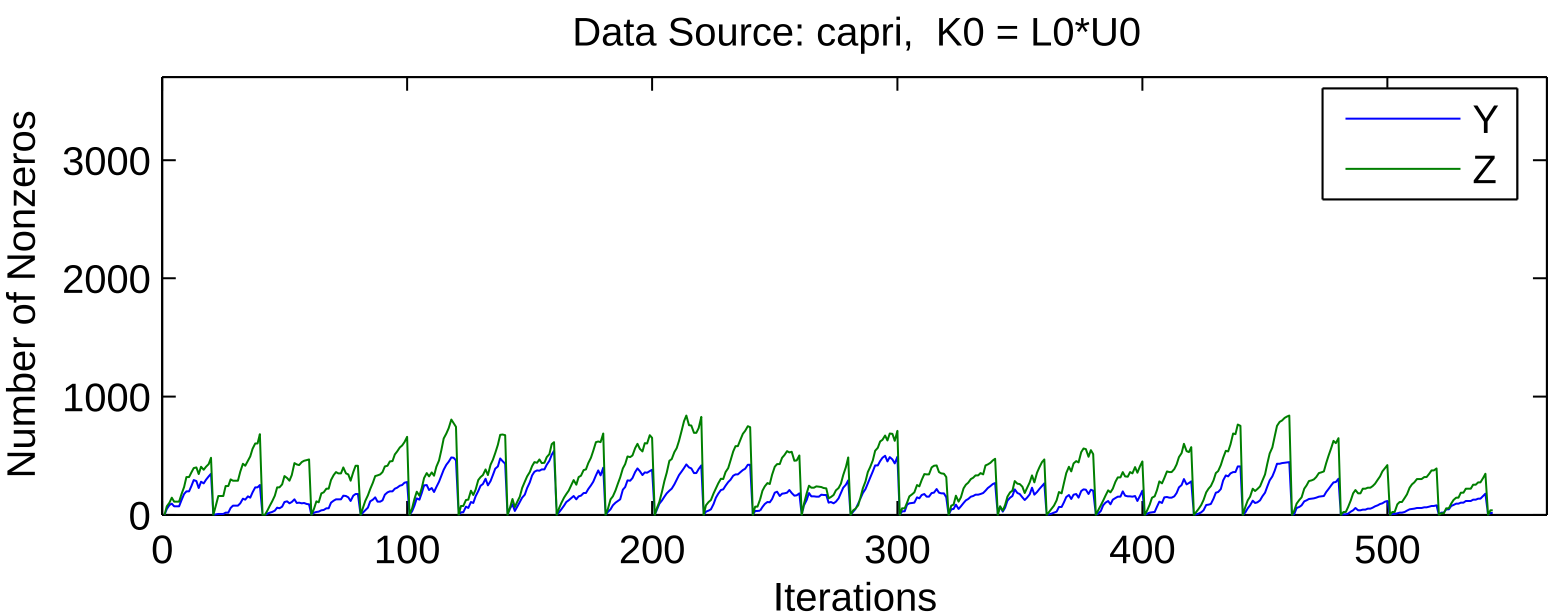
$$\begin{pmatrix} K_0 & V \\ W^T & D \end{pmatrix} = \begin{pmatrix} L_0 & \\ Z^T & I \end{pmatrix} \begin{pmatrix} U_0 & Y \\ & C \end{pmatrix}$$

Compare nonzeros in $Y$ and $Z$:

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

# **Black-box** $K_0$ **vs separate** $L_0$, $U_0$



# Summary

## 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 $K_0 = L_0 U_0$ or $L_0 D_0 L_0^T$
- Our main hope for parallelism in LP/QP/NLP solvers
- **Need separate solves with $L_0$, $D_0$, $U_0$**
- **Need rank-revealing factors of $K_0$**
- Ideally parallel products $Yv$, $Y^T w$, $Zv$, $Z^T w$ (OBLAS?)