# Adaptive Low Complexity Algorithms for Unconstrained Minimization

Carmine Di Fiore, Stefano Fanelli, Paolo Zellini
mailto:difiore@mat.uniroma2.it

*Cortona, September 2004*

1. The minimization problem and classical solvers
2. Previous contribution: $\mathcal{L}QN$ descent methods
3. New contribution: Adaptive $\mathcal{L}QN$ descent methods

# The minimization problem and classical solvers

$f(\mathbf{x}_*) = \min_{\mathbf{x}\in R^n} f(\mathbf{x}), \quad \text{find } \mathbf{x}_*$

# The minimization problem and classical solvers

$f(\mathbf{x}_*) = \min_{\mathbf{x}\in R^n} f(\mathbf{x}), \quad$ find $\mathbf{x}_*$

Descent methods

generate a minimizing sequence $\{\mathbf{x}_k\}_{k=0}^{+\infty}$ by the iterative scheme:

$$
\begin{array}{l}
\mathbf{x}_0 \in R^n, \quad \mathbf{g}_0 = \nabla f(\mathbf{x}_0), \quad \mathbf{d}_0 = -\mathbf{g}_0 \\
\textit{For} \;\; k = 0, 1, \ldots \\
\left\{ \begin{array}{l}
\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k \quad \lambda_k > 0 \\
\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1}) \\
B_{k+1} = n \times n \text{ matrix, positive definite (pd)} \\
\mathbf{d}_{k+1} = \underbrace{-B_{k+1}^{-1}\mathbf{g}_{k+1}}_{\text{descent direction}}
\end{array} \right.
\end{array}
$$

## The Newton descent method

- $B_{k+1} = \nabla^2 f(\mathbf{x}_{k+1})$
- A quadratic rate of convergence
- $O(n^3)$ arithmetic operations to compute $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$

## The Newton descent method

- $B_{k+1} = \nabla^2 f(\mathbf{x}_{k+1})$
- A quadratic rate of convergence
- $O(n^3)$ arithmetic operations to compute $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$

## Quasi-Newton (QN) descent methods

- $B_{k+1}$ defined in terms of $\nabla f$
- A superlinear rate of convergence
- Convergence under weak analytical assumptions
- $O(n^2)$ arithmetic operations to compute $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$
- $O(n^2)$ memory allocations for implementation

The Newton descent method

- $B_{k+1} = \nabla^2 f(\mathbf{x}_{k+1})$
- A quadratic rate of convergence
- $O(n^3)$ arithmetic operations to compute $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$

Quasi-Newton (QN) descent methods

- $B_{k+1}$ defined in terms of $\nabla f$
- A superlinear rate of convergence
- Convergence under weak analytical assumptions
- $O(n^2)$ arithmetic operations to compute $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$
- $O(n^2)$ memory allocations for implementation

*Main example:* the *BFGS* method (Broyden et al.'70)

## *BFGS*

$$\mathbf{x}_0 \in R^n, \quad \mathbf{d}_0 = -\mathbf{g}_0$$
$$\textit{For} \quad k = 0, 1, \ldots$$
$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k \qquad \lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0 \\ B_{k+1} = \varphi\left(B_k, \underbrace{\mathbf{x}_{k+1} - \mathbf{x}_k}_{\mathbf{s}_k}, \underbrace{\mathbf{g}_{k+1} - \mathbf{g}_k}_{\mathbf{y}_k}\right) \\ \mathbf{d}_{k+1} = -B_{k+1}^{-1} \mathbf{g}_{k+1} \end{cases}$$

## BFGS

$$\mathbf{x}_0 \in R^n, \quad \mathbf{d}_0 = -\mathbf{g}_0$$
$$For \quad k = 0, 1, \ldots$$
$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k \qquad \lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0 \\ B_{k+1} = \varphi\left(B_k, \underbrace{\mathbf{x}_{k+1} - \mathbf{x}_k}_{\mathbf{s}_k}, \underbrace{\mathbf{g}_{k+1} - \mathbf{g}_k}_{\mathbf{y}_k}\right) \\ \mathbf{d}_{k+1} = -B_{k+1}^{-1}\mathbf{g}_{k+1} \end{cases}$$

$\varphi$ properties $\Rightarrow$

- *$B_{k+1}$ inherites positive definiteness from $B_k$*

  Proof: $B$ pd & $\mathbf{s}^T\mathbf{y} > 0 \Rightarrow \varphi(B, \mathbf{s}, \mathbf{y})$ pd

- $B_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{g}_{k+1} - \mathbf{g}_k$

  Proof: $\varphi(B, \mathbf{s}, \mathbf{y})\mathbf{s} = \mathbf{y}$

The updating function $\varphi$ in $B_{k+1} = \varphi\left(B_k, \mathbf{s}_k, \mathbf{y}_k\right)$ is

$$\varphi\left(B, \mathbf{s}, \mathbf{y}\right) = B + \frac{1}{\mathbf{y}^T\mathbf{s}}\mathbf{y}\mathbf{y}^T - \frac{1}{\mathbf{s}^T B\mathbf{s}} B\mathbf{s}\mathbf{s}^T B$$

$\Rightarrow$ *BFGS is a secant method:*

$$B_{k+1}\underbrace{(\mathbf{x}_{k+1} - \mathbf{x}_k)}_{\mathbf{s}_k} = \underbrace{\mathbf{g}_{k+1} - \mathbf{g}_k}_{\mathbf{y}_k} \quad \textit{secant equation}$$

Proof *(independent on B)*:

$$\begin{aligned} \varphi(B, \mathbf{s}, \mathbf{y})\mathbf{s} &= \left(B + \tfrac{1}{\mathbf{y}^T\mathbf{s}}\mathbf{y}\mathbf{y}^T - \tfrac{1}{\mathbf{s}^T B\mathbf{s}} B\mathbf{s}\mathbf{s}^T B\right)\mathbf{s} \\ &= B\mathbf{s} + \tfrac{1}{\mathbf{y}^T\mathbf{s}}\mathbf{y}(\mathbf{y}^T\mathbf{s}) - \tfrac{1}{\mathbf{s}^T B\mathbf{s}} B\mathbf{s}(\mathbf{s}^T B\mathbf{s}) \\ &= \mathbf{y} \end{aligned}$$

Quasi-Newton (QN) descent methods *for large scale problems*

- $B_{k+1}$ defined in terms of $\nabla f$
- A *fast* rate of convergence
- Convergence under weak analytical assumptions
- *less than* $O(n^2)$ arithmetic operations to compute $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$
- *less than* $O(n^2)$ memory allocations for implementation

*Classical example:* the *Limited memory BFGS* method (Nocedal et al. '80)

## Quasi-Newton (QN) descent methods *for large scale problems*

- $B_{k+1}$ defined in terms of $\nabla f$
- A *fast* rate of convergence
- Convergence under weak analytical assumptions
- *less than* $O(n^2)$ arithmetic operations to compute $\mathbf{x}_{k+1}$ from $\mathbf{x}_k$
- *less than* $O(n^2)$ memory allocations for implementation

*Classical example:* the *Limited memory BFGS* method (Nocedal et al. '80)

*A recent proposal:* the $\mathcal{L}QN$ method (Di Fiore, Fanelli, Zellini et al. '00)

# Previous contribution: $\mathcal{L}QN$ descent methods

Replace the matrix $B_k$ in

$$B_{k+1} = \varphi(B_k, \mathbf{s}_k, \mathbf{y}_k)$$

with a matrix $A_k$ of a *low complexity space $\mathcal{L}$*

# Previous contribution: $\mathcal{L}QN$ descent methods

Replace the matrix $B_k$ in

$$B_{k+1} = \varphi(B_k, \mathbf{s}_k, \mathbf{y}_k)$$

with a matrix $A_k$ of a *low complexity space* $\mathcal{L}$

Choice of $\mathcal{L}$

$B_k \in sd\,U$ for some unitary matrix $U$, where

$$sd\,U = \{\ Ud(\mathbf{z})U^* \ :\ \mathbf{z} \in C^n\ \}, \quad d(\mathbf{z}) = \begin{bmatrix} z_1 & 0 & \cdots & 0 \\ 0 & z_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & z_n \end{bmatrix}$$

$\Rightarrow$ choose $\mathcal{L} = sd\,U$, $U =$*fast unitary transform* ($U = $ Fourier, Hartley, ...)

Choice of $A_k$ in $\mathcal{L}$

$A_k =$ the *best least squares fit* to $B_k$ in $\mathcal{L} = sd\ U$, i.e.
$A_k = \mathcal{L}_{B_k}$ where

$$\|\mathcal{L}_{B_k} - B_k\|_F = \min_{X \in \mathcal{L}} \|X - B_k\|_F$$

The $\mathcal{L}QN$ algorithm

$$\mathbf{x}_0 \in R^n, \quad \mathbf{d}_0 = -\mathbf{g}_0$$
$$\textit{For} \quad k = 0, 1, \ldots$$
$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k \qquad \lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0 \\ B_{k+1} = \varphi(\ \mathcal{L}_{B_k}\ ,\ \underbrace{\mathbf{x}_{k+1} - \mathbf{x}_k}_{\mathbf{s}_k},\ \underbrace{\mathbf{g}_{k+1} - \mathbf{g}_k}_{\mathbf{y}_k}) \\ \mathbf{d}_{k+1} = -B_{k+1}^{-1} \mathbf{g}_{k+1} \end{cases}$$

$$\underline{B_{k+1} = \varphi(\mathcal{L}_{B_k}, \mathbf{s}_k, \mathbf{y}_k)}$$

- *$B_{k+1}$ inherites positive definiteness from $B_k$*

Proof: $B$ pd $\Rightarrow$ $\mathcal{L}_B$ pd

$\underline{B_{k+1} = \varphi(\mathcal{L}_{B_k}, \mathbf{s}_k, \mathbf{y}_k)}$

- $B_{k+1}$ *inherites positive definiteness from* $B_k$

  Proof: $B$ pd $\Rightarrow$ $\mathcal{L}_B$ pd

- $B_{k+1}\mathbf{s}_k = \mathbf{y}_k$, i.e. $\mathcal{L}QN$ *is a secant method*

$B_{k+1} = \varphi(\mathcal{L}_{B_k}, \mathbf{s}_k, \mathbf{y}_k)$

- *$B_{k+1}$ inherites positive definiteness from $B_k$*

  Proof: $B$ pd $\Rightarrow$ $\mathcal{L}_B$ pd
- $B_{k+1}\mathbf{s}_k = \mathbf{y}_k$, i.e. *$\mathcal{L}QN$ is a secant method*
- $B_{k+1}$ projected on $\mathcal{L}$ gives rise the Eigenvalue Updating Formula

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \frac{1}{\mathbf{s}_k^T \mathbf{y}_k} |U^* \mathbf{y}_k|^2 - \frac{1}{\mathbf{z}_k{}^T |U^* \mathbf{s}_k|^2} d(\mathbf{z}_k)^2 |U^* \mathbf{s}_k|^2 \qquad \text{(EUF)}$$

where $\mathcal{L}_{B_k} = U d(\mathbf{z}_k) U^*$.
(EUF) and the Sherman-Morrison formula imply that *each step of $\mathcal{L}QN$ can be performed via two matrix-vector products $U \cdot \mathbf{z}$ and some inner products*

$B_{k+1} = \varphi(\mathcal{L}_{B_k}, \mathbf{s}_k, \mathbf{y}_k)$

- *$B_{k+1}$ inherites positive definiteness from $B_k$*

  Proof: $B$ pd $\Rightarrow$ $\mathcal{L}_B$ pd
- $B_{k+1}\mathbf{s}_k = \mathbf{y}_k$, i.e. *$\mathcal{L}QN$ is a secant method*
- $B_{k+1}$ projected on $\mathcal{L}$ gives rise the Eigenvalue Updating Formula

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \frac{1}{\mathbf{s}_k^T \mathbf{y}_k}|U^*\mathbf{y}_k|^2 - \frac{1}{\mathbf{z}_k{}^T|U^*\mathbf{s}_k|^2} d(\mathbf{z}_k)^2 |U^*\mathbf{s}_k|^2 \qquad \text{(EUF)}$$

where $\mathcal{L}_{B_k} = Ud(\mathbf{z}_k)U^*$.
(EUF) and the Sherman-Morrison formula imply that *each step of $\mathcal{L}QN$ can be performed via two matrix-vector products $U \cdot \mathbf{z}$ and some inner products*

**Main result**: $U =$ fast transform $\Rightarrow$

Space complexity: $O(n)$= memory allocations for $U$

Time complexity (per step): $O(n \log n)$= cost of $U \cdot \mathbf{z}$

## $\mathcal{L}QN$ rate of convergence

*Theory :* linear rate of convergence

$\mathcal{L}QN$ rate of convergence

*Theory :* linear rate of convergence

*Experiments :* fast rate of convergence, competitive with *L-BFGS*

## $\mathcal{L}QN$ rate of convergence

*Theory :* linear rate of convergence

*Experiments :* fast rate of convergence, competitive with *L-BFGS*

- The Ionosphere data set ($n = 1408$)

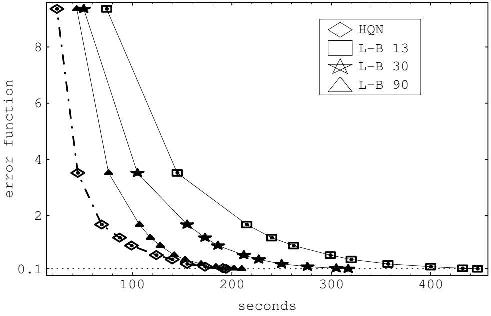


Figure: $\mathcal{L}QN$ and *L-BFGS* applied to a function of 1408 variables

## New contribution: Adaptive $\mathcal{L}QN$ descent methods

In the updating formula

$$B_{k+1} = \varphi(\mathcal{L}_{B_k}, \mathbf{s}_k, \mathbf{y}_k)$$

*adapt the space $\mathcal{L}$ to the current iteration*

# New contribution: Adaptive $\mathcal{L}QN$ descent methods

In the updating formula

$$B_{k+1} = \varphi(\mathcal{L}_{B_k}, \mathbf{s}_k, \mathbf{y}_k)$$

*adapt the space $\mathcal{L}$ to the current iteration*

The adaptive criterion A $\mathcal{L}QN$ drawback with respect to *BFGS* is that the updated matrix $\mathcal{L}_{B_k}$ does not solve the previous secant equation $X\mathbf{s}_{k-1} = \mathbf{y}_{k-1}$

Let $\mathcal{L}_{\mathbf{sy}}$ be the matrix of $\mathcal{L} = sd\, U$ s.t.

$$\mathcal{L}_{\mathbf{sy}}\, \mathbf{s}_{k-1} = \mathbf{y}_{k-1} \qquad (\mathcal{L}_{\mathbf{sy}} \neq \mathcal{L}_{B_k})$$

$$\Rightarrow \qquad \mathcal{L}_{\mathbf{sy}} = U\mathrm{diag}\left(\frac{[U^*\mathbf{y}_{k-1}]_i}{[U^*\mathbf{s}_{k-1}]_i}\right)U^*$$

# New contribution: Adaptive $\mathcal{L}QN$ descent methods

In the updating formula

$$B_{k+1} = \varphi(\mathcal{L}_{B_k}, \mathbf{s}_k, \mathbf{y}_k)$$

*adapt the space $\mathcal{L}$ to the current iteration*

The adaptive criterion A $\mathcal{L}QN$ drawback with respect to *BFGS* is that the updated matrix $\mathcal{L}_{B_k}$ does not solve the previous secant equation $X\mathbf{s}_{k-1} = \mathbf{y}_{k-1}$

Let $\mathcal{L}_{\mathbf{sy}}$ be the matrix of $\mathcal{L} = sd\, U$ s.t.

$$\mathcal{L}_{\mathbf{sy}}\, \mathbf{s}_{k-1} = \mathbf{y}_{k-1} \qquad (\mathcal{L}_{\mathbf{sy}} \neq \mathcal{L}_{B_k})$$

$$\Rightarrow \qquad \mathcal{L}_{\mathbf{sy}} = U\mathrm{diag}\left(\frac{[U^*\mathbf{y}_{k-1}]_i}{[U^*\mathbf{s}_{k-1}]_i}\right)U^*$$

*AIM: $\mathcal{L}_{B_k}$ close to $\mathcal{L}_{\mathbf{sy}}$* during the minimization procedure

# New contribution: Adaptive $\mathcal{L}QN$ descent methods

In the updating formula

$$B_{k+1} = \varphi(\mathcal{L}_{B_k}, \mathbf{s}_k, \mathbf{y}_k)$$

*adapt the space $\mathcal{L}$ to the current iteration*

The adaptive criterion A $\mathcal{L}QN$ drawback with respect to *BFGS* is that the updated matrix $\mathcal{L}_{B_k}$ does not solve the previous secant equation $X\mathbf{s}_{k-1} = \mathbf{y}_{k-1}$

Let $\mathcal{L}_{\mathbf{sy}}$ be the matrix of $\mathcal{L} = sd\ U$ s.t.

$$\mathcal{L}_{\mathbf{sy}}\ \mathbf{s}_{k-1} = \mathbf{y}_{k-1} \qquad (\mathcal{L}_{\mathbf{sy}} \neq \mathcal{L}_{B_k})$$

$$\Rightarrow \qquad \mathcal{L}_{\mathbf{sy}} = U\mathrm{diag}\left(\frac{[U^*\mathbf{y}_{k-1}]_i}{[U^*\mathbf{s}_{k-1}]_i}\right)U^*$$

*AIM: $\mathcal{L}_{B_k}$ close to $\mathcal{L}_{\mathbf{sy}}$* during the minimization procedure

$\rightarrow$ *$\mathcal{L}_{\mathbf{sy}}$ positive definite like $\mathcal{L}_{B_k}$* $U$ =fast transform s.t. $\dfrac{[U^*\mathbf{y}_{k-1}]_i}{[U^*\mathbf{s}_{k-1}]_i} > 0$

## The adaptive $\mathcal{L}QN$ algorithm

Like the $\mathcal{L}QN$ algorithm, but

## The adaptive $\mathcal{L}QN$ algorithm

Like the $\mathcal{L}QN$ algorithm, but

$$
\begin{array}{l}
\dots \;\; \dots \;\; \lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0 \\
B_{k+1} = \varphi(\mathcal{L}_{B_k},\ \mathbf{s}_k,\ \mathbf{y}_k) \\
\text{if } \mathcal{L}_{\mathbf{s}_k \mathbf{y}_k} \text{ is pd then} \{ \\
\quad \mathbf{d}_{k+1} = -B_{k+1}^{-1} \mathbf{g}_{k+1} \\
\}
\end{array}
$$

## The adaptive $\mathcal{L}QN$ algorithm

Like the $\mathcal{L}QN$ algorithm, but

$$\begin{array}{l}
\ldots\ \ldots\ \lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0 \\
B_{k+1} = \varphi(\mathcal{L}_{B_k},\ \mathbf{s}_k,\ \mathbf{y}_k) \\
\text{if } \mathcal{L}_{\mathbf{s}_k \mathbf{y}_k} \text{ is pd then} \{ \\
\quad \mathbf{d}_{k+1} = -B_{k+1}^{-1} \mathbf{g}_{k+1} \\
\} \text{ else } \{ \\
\quad \mathbf{d}_{k+1} = -(\mathcal{L}_{B_{k+1}})^{-1} \mathbf{g}_{k+1} \qquad \leftarrow \text{temporary descent direction}
\end{array}$$

## The adaptive $\mathcal{L}QN$ algorithm

Like the $\mathcal{L}QN$ algorithm, but

$$\ldots\ \ldots\ \lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0$$

$$B_{k+1} = \varphi(\mathcal{L}_{B_k},\ \mathbf{s}_k,\ \mathbf{y}_k)$$

if $\mathcal{L}_{\mathbf{s}_k \mathbf{y}_k}$ is pd then{

$$\mathbf{d}_{k+1} = -B_{k+1}^{-1}\mathbf{g}_{k+1}$$

} else {

$$\mathbf{d}_{k+1} = -(\mathcal{L}_{B_{k+1}})^{-1}\mathbf{g}_{k+1} \qquad \leftarrow \text{temporary descent direction}$$

*define a fast transform* $U$ s.t. $\mathcal{L}_{\mathbf{s}_k \mathbf{y}_k}$ is pd

## The adaptive $\mathcal{L}QN$ algorithm

Like the $\mathcal{L}QN$ algorithm, but

$$\begin{array}{l}
\ldots\ \ldots\ \lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0 \\
B_{k+1} = \varphi(\mathcal{L}_{B_k},\ \mathbf{s}_k,\ \mathbf{y}_k) \\
\text{if } \mathcal{L}_{\mathbf{s}_k\mathbf{y}_k} \text{ is pd then}\{ \\
\quad \mathbf{d}_{k+1} = -B_{k+1}^{-1}\mathbf{g}_{k+1} \\
\}\text{ else }\{ \\
\quad \mathbf{d}_{k+1} = -(\mathcal{L}_{B_{k+1}})^{-1}\mathbf{g}_{k+1} \qquad \leftarrow \text{temporary descent direction} \\
\quad \textit{define a fast transform } U \text{ s.t. } \mathcal{L}_{\mathbf{s}_k\mathbf{y}_k} \text{ is pd} \\
\quad \text{set } \mathcal{L} = sd\, U \\
\}
\end{array}$$

## The adaptive $\mathcal{L}QN$ algorithm

Like the $\mathcal{L}QN$ algorithm, but

$$
\begin{array}{l}
\ldots\ \ldots\ \lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0 \\
B_{k+1} = \varphi(\mathcal{L}_{B_k},\ \mathbf{s}_k,\ \mathbf{y}_k) \\
\text{if } \mathcal{L}_{\mathbf{s}_k \mathbf{y}_k} \text{ is pd then} \{ \\
\quad \mathbf{d}_{k+1} = -B_{k+1}^{-1} \mathbf{g}_{k+1} \\
\} \text{ else } \{ \\
\quad \mathbf{d}_{k+1} = -(\mathcal{L}_{B_{k+1}})^{-1} \mathbf{g}_{k+1} \qquad \leftarrow \text{temporary descent direction} \\
\quad \textit{define a fast transform } U \text{ s.t. } \mathcal{L}_{\mathbf{s}_k \mathbf{y}_k} \text{ is pd} \\
\quad \text{set } \mathcal{L} = sd\, U \\
\}
\end{array}
$$

How to define such $U$ ?

Definition of $U$

$\mathcal{L}_{\mathbf{sy}} = U\mathrm{diag}\left(\frac{[U^*\mathbf{y}_k]_i}{[U^*\mathbf{s}_k]_i}\right)U^*$ is positive definite *iff*

$U$ is such that

$$\frac{[U^*\mathbf{y}_k]_i}{[U^*\mathbf{s}_k]_i} > 0 \quad \forall i \qquad \text{(Crit)}$$

Definition of $U$

$\mathcal{L}_{\mathbf{sy}} = U \mathrm{diag}\left(\frac{[U^*\mathbf{y}_k]_i}{[U^*\mathbf{s}_k]_i}\right) U^*$ is positive definite *iff*

$U$ is such that

$$\frac{[U^*\mathbf{y}_k]_i}{[U^*\mathbf{s}_k]_i} > 0 \quad \forall i \qquad \text{(Crit)}$$

**Main results:** Under our hypothesis on $\lambda_k$ $(\lambda_k \mid \mathbf{s}_k^T \mathbf{y}_k > 0)$ a matrix $U$ satisfying (Crit) *exists* and can be obtained as the *product of two Householder matrices:*

$$U = H(\mathbf{u})H(\mathbf{p}), \quad H(\mathbf{z}) = I - \frac{2}{\|\mathbf{z}\|^2}\mathbf{z}\mathbf{z}^*$$

($\mathbf{u}, \mathbf{p}$ suitable vectors), $\Rightarrow$

Space complexity: $O(n)$= memory allocations for $U$

Time complexity (per step): $O(n)$= cost of $U \cdot \mathbf{z}$ (better than $\mathcal{L}QN$)

## Rate of convergence of *adaptive* $\mathcal{L}QN$

*Experiments :* fast rate of convergence, competitive with $\mathcal{L}QN$

## Rate of convergence of *adaptive* $\mathcal{L}QN$

*Experiments :* fast rate of convergence, competitive with $\mathcal{L}QN$

- The Ionosphere data set $(n = 1408)$

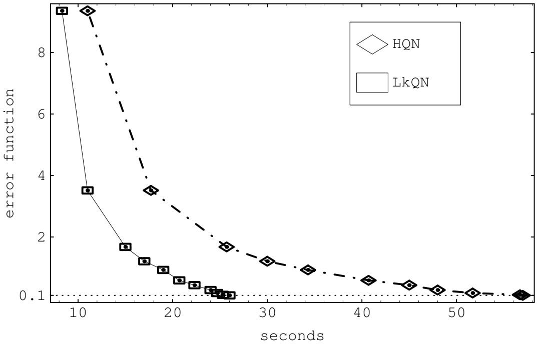


Figure: $\mathcal{L}QN$ and *adaptive* $\mathcal{L}QN$ applied to a function of 1408 variables

- The Iris plant data set $(n = 315)$

*Number of iterations to obtain* $f(\mathbf{x}_k) < 0.1$

| $f$ | $\mathbf{x}_0^1$ | $\mathbf{x}_0^2$ | $\mathbf{x}_0^3$ | $\mathbf{x}_0^4$ |
|---|---|---|---|---|
| $\mathcal{L}QN$ | 10930 | 13108 | 3854 | 7663 |
| *adaptive* $\mathcal{L}QN$ | 3430 | 1663 | 3647 | 1525 |

- The Iris plant data set $(n = 315)$

*Number of iterations to obtain* $f(\mathbf{x}_k) < 0.1$

| $f$ | $\mathbf{x}_0^1$ | $\mathbf{x}_0^2$ | $\mathbf{x}_0^3$ | $\mathbf{x}_0^4$ |
|---|---|---|---|---|
| $\mathcal{L}QN$ | 10930 | 13108 | 3854 | 7663 |
| *adaptive* $\mathcal{L}QN$ | 3430 | 1663 | 3647 | 1525 |

*Number of iterations to obtain* $f(\mathbf{x}_k) < 0.01$

| $f$ | $\mathbf{x}_0^1$ | $\mathbf{x}_0^2$ | $\mathbf{x}_0^3$ | $\mathbf{x}_0^4$ |
|---|---|---|---|---|
| $\mathcal{L}QN$ | 24085 | 42344 | 6184 | 33250 |
| *adaptive* $\mathcal{L}QN$ | 19961 | 2886 | 8306 | 3111 |

Two strategies

- Secant equation: $\mathcal{L}_{\mathbf{sy}}\mathbf{s}_k = \mathbf{y}_k$
- Best least squares approximation: $\|\mathcal{L}_{B_k} - B_k\|_F = \min_{X \in \mathcal{L}} \|X - B_k\|_F$

How to apply both strategies ?

The *adaptive* $\mathcal{L}QN$ algorithm illustrated is a possible solution

*Work in progress:* look for other solutions