<table>
<thead>
<tr>
<th>Contribution</th>
<th>Properties</th>
<th>Proposed Framework</th>
<th>Numerical Experiments</th>
<th>Summary</th>
</tr>
</thead>
</table>

Self-Correcting Variable-Metric Algorithms for Nonsmooth Optimization

**Frank E. Curtis**, Lehigh University

joint work with

**Daniel P. Robinson**, Johns Hopkins University

International Conference on Continuous Optimization (ICCOPT)
Tokyo, Japan

8 August 2016
Outline

Contribution

Self-Correcting Properties of BFGS-type Updating

Proposed Framework

Numerical Experiments

Summary
Outline

Contribution

Self-Correcting Properties of BFGS-type Updating

Proposed Framework

Numerical Experiments

Summary
Consider unconstrained optimization problems of the form

\[
\min_{x \in \mathbb{R}^n} f(x),
\]

where \( f \) is

- locally Lipschitz in \( \mathbb{R}^n \) and
- differentiable in an open, dense subset of \( \mathbb{R}^n \),

but

- nonsmooth and (potentially) nonconvex.
Balance between first- and second-order methods

For deterministic, smooth optimization, a nice balance achieved by quasi-Newton:

\[ x_{k+1} \leftarrow x_k - \alpha_k W_k g_k, \]

where

- \( \alpha_k > 0 \) is a stepsize;
- \( g_k \leftarrow \nabla f(x_k) \);
- \{W_k\} is updated dynamically.

We all know:

- local rescaling based on iterate/subgradient displacements
- only first-order derivatives required
- no linear system solves required
- global convergence guarantees (say, with line search)
- superlinear local convergence rate

How can we carry these ideas to nonsmooth settings?
What has been done?

Many have observed improved performance with quasi-Newton schemes

“Unadulterated” BFGS
  - Lemaréchal (1982)
  - Lewis, Overton (2012)

BFGS (with restricted updates)
  - Curtis, Que (2015)

**Issue:** global convergence guarantees muddled by
  - “Hessian” approximations† tending to singularity
  - intertwined \(\{x_k\}, \{\alpha_k\}, \{g_k\}, \text{ and } \{W_k\}\)

To our knowledge, none have tried to exploit **self-correcting** properties of BFGS

† “Hessian” and “inverse Hessian” used loosely in nonsmooth settings
Propose a quasi-Newton method for nonsmooth optimization

- **unifying framework** covering
  - cutting plane / bundle methods (convex only)
  - gradient sampling methods (nonconvex)
- exploit **self-correcting** properties of BFGS-type updates
  - Powell (1976)
  - Ritter (1979, 1981)
  - Werner (1978)
  - Byrd, Nocedal (1989)
- properties of **Hessians** offer useful bounds for **inverse Hessians**
- global convergence guarantees
- improved practical performance

**Remember:** Forget about superlinear convergence (not relevant here!)
Outline

Contribution

Self-Correcting Properties of BFGS-type Updating

Proposed Framework

Numerical Experiments

Summary

Self-Correcting Variable-Metric Algorithms for Nonsmooth Optimization
BFGS-type updates

Inverse Hessian and Hessian approximation updating formulas \((s_k^T v_k > 0)\):

\[
W_{k+1} \leftarrow \left( I - \frac{v_k s_k^T}{s_k^T v_k} \right)^T W_k \left( I - \frac{v_k s_k^T}{s_k^T v_k} \right) + \frac{s_k s_k^T}{s_k^T v_k} \\
H_{k+1} \leftarrow \left( I - \frac{s_k s_k^T H_k}{s_k^T H_k s_k} \right)^T H_k \left( I - \frac{s_k s_k^T H_k}{s_k^T H_k s_k} \right) + \frac{v_k v_k^T}{s_k^T v_k}
\]

- These satisfy secant-type equations

\[
W_{k+1} v_k = s_k \quad \text{and} \quad H_{k+1} s_k = v_k,
\]

but these are not relevant for this talk.

- Choosing \(v_k \leftarrow y_k := g_{k+1} - g_k\) yields standard BFGS, but we consider

\[
v_k \leftarrow \beta_k s_k + (1 - \beta_k) \tilde{y}_k \quad \text{for some} \quad \beta_k \in [0, 1] \quad \text{and} \quad \tilde{y}_k \in \mathbb{R}^n.
\]

This scheme is important to preserve self-correcting properties.

Consider the matrices (which only depend on $s_k$ and $H_k$, not $g_k$!)

\[ P_k := \frac{s_k s_k^T H_k}{s_k^T H_k s_k} \quad \text{and} \quad Q_k := I - P_k. \]

Both $H_k$-orthogonal projection matrices (i.e., idempotent and $H_k$-self-adjoint).

- $P_k$ yields $H_k$-orthogonal projection onto span($s_k$).
- $Q_k$ yields $H_k$-orthogonal projection onto span($s_k$)$^\perp H_k$. 

Consider the matrices (which only depend on \( s_k \) and \( H_k \), not \( g_k \)!

\[
P_k := \frac{s_k s_k^T H_k}{s_k^T H_k s_k} \quad \text{and} \quad Q_k := I - P_k.
\]

Both \( H_k \)-orthogonal projection matrices (i.e., idempotent and \( H_k \)-self-adjoint).

- \( P_k \) yields \( H_k \)-orthogonal projection onto \( \text{span}(s_k) \).
- \( Q_k \) yields \( H_k \)-orthogonal projection onto \( \text{span}(s_k)^\perp H_k \).

Returning to the Hessian update:

\[
H_{k+1} \leftarrow \left( I - \frac{s_k s_k^T H_k}{s_k^T H_k s_k} \right)^T H_k \left( I - \frac{s_k s_k^T H_k}{s_k^T H_k s_k} \right) + \frac{v_k v_k^T}{s_k^T v_k}.
\]

- Curvature projected out along \( \text{span}(s_k) \)
- Curvature corrected by \( \frac{v_k v_k^T}{s_k^T v_k} = \left( \frac{v_k v_k^T}{\|v_k\|^2} \right) \left( \frac{\|v_k\|^2}{v_k^T W_{k+1} v_k} \right) \) (inverse Rayleigh).
Self-correcting properties of Hessian update

Since curvature is constantly projected out, what happens after many updates?
Self-correcting properties of Hessian update

Since curvature is constantly projected out, what happens after many updates?

**Theorem 1 (Byrd, Nocedal (1989))**

Suppose that, for all $k$, there exists $\{\eta, \theta\} \subset \mathbb{R}^{++}$ such that

$$\eta \leq \frac{s_k^T v_k}{\|s_k\|_2^2} \quad \text{and} \quad \frac{\|v_k\|_2^2}{s_k^T v_k} \leq \theta. \quad (\star)$$

Then, for any $p \in (0, 1)$, there exist constants $\{\iota, \kappa, \lambda\} \subset \mathbb{R}^{++}$ such that, for any $K \geq 2$, the following relations hold for at least $\lceil pK \rceil$ values of $k \in \{1, \ldots, K\}$:

$$\iota \leq \frac{s_k^T H_k s_k}{\|s_k\|_2 \|H_k s_k\|_2} \quad \text{and} \quad \kappa \leq \frac{\|H_k s_k\|_2}{\|s_k\|_2} \leq \lambda.$$ 

**Proof technique.**

Building on work of Powell (1976), involves bounding growth of

$$\gamma(H_k) = \text{tr}(H_k) - \ln(\det(H_k)).$$
Rather than focus on superlinear convergence results, we care about the following.

**Corollary 2**

Suppose the conditions of Theorem 1 hold. Then, for any \( p \in (0, 1) \), there exist constants \( \{\mu, \nu\} \subset \mathbb{R}_{++} \) such that, for any \( K \geq 2 \), the following relations hold for at least \( \lceil pK \rceil \) values of \( k \in \{1, \ldots, K\} \):

\[
\mu \|\bar{g}_k\|^2 \leq \bar{g}_k^T W_k \bar{g}_k \quad \text{and} \quad \|W_k \bar{g}_k\|^2 \leq \nu \|\bar{g}_k\|^2
\]

Here \( \bar{g}_k \) is the vector such that the iterate displacement is

\[
x_{k+1} - x_k = s_k = -W_k \bar{g}_k
\]

**Proof sketch.**

Follows simply after algebraic manipulations from the result of Theorem 1, using the facts that \( s_k = -W_k \bar{g}_k \) and \( W_k = H_k^{-1} \) for all \( k \).
# Outline

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Proposed Framework</th>
<th>Numerical Experiments</th>
<th>Summary</th>
</tr>
</thead>
</table>

## Contribution

Self-Correcting Properties of BFGS-type Updating

## Proposed Framework

## Numerical Experiments

## Summary
Subproblems in nonsmooth optimization algorithms

With sets of points, scalars, and (sub)gradients

\[ \{x_{k,j}\}_{j=1}^{m}, \{f_{k,j}\}_{j=1}^{m}, \{g_{k,j}\}_{j=1}^{m}, \]

nonsmooth optimization methods involve the primal subproblem

\[
\min_{x \in \mathbb{R}^n} \left( \max_{j \in \{1,\ldots,m\}} \{ f_{k,j} + g_{k,j}^T (x - x_{k,j}) \} + \frac{1}{2} (x - x_k)^T H_k (x - x_k) \right)
\]

s.t. \( \|x - x_k\| \leq \delta_k \),

but, with \( G_k \leftarrow [g_{k,1} \cdots g_{k,m}] \), it is typically more efficient to solve the dual

\[
\sup_{(\omega,\gamma) \in \mathbb{R}_+^m \times \mathbb{R}^n} -\frac{1}{2} (G_k \omega + \gamma)^T W_k (G_k \omega + \gamma) + b_k^T \omega - \delta_k ||\gamma||^* \]

s.t. \( \mathbb{1}_m^T \omega = 1 \).

The primal solution can then be recovered by

\[
x_k^* \leftarrow x_k - W_k (G_k \omega_k + \gamma_k). \]

**Algorithm**  Self-Correcting BFGS for Nonsmooth Optimization

1: Choose $x_1 \in \mathbb{R}^n$.
2: Choose a symmetric positive definite $W_1 \in \mathbb{R}^{n \times n}$.
3: Choose $\alpha \in (0, 1)$

4: **for** $k = 1, 2, \ldots$ **do**

5: Solve (P)–(D) such that setting

\[
G_k \leftarrow [g_{k,1} \cdots g_{k,m}],
\]

\[
s_k \leftarrow -W_k(G_k \omega_k + \gamma_k),
\]

and $x_{k+1} \leftarrow x_k + s_k$

6: yields

\[
f(x_{k+1}) \leq f(x_k) - \frac{1}{2} \alpha(G_k \omega_k + \gamma_k)^T W_k (G_k \omega_k + \gamma_k).
\]

7: Choose $\tilde{y}_k \in \mathbb{R}^n$.
8: Set $\beta_k \leftarrow \min\{\beta \in [0, 1] : v(\beta) := \beta s_k + (1 - \beta) \tilde{y}_k \text{satisfies } (\star)\}$.

9: Set $v_k \leftarrow v(\beta_k)$.

10: Set

\[
W_{k+1} \leftarrow \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right) W_k \left(I - \frac{v_k s_k^T}{s_k^T v_k}\right)^T + \frac{s_k s_k^T}{s_k^T v_k}.
\]

11: **end for**
Instances of the framework

Cutting plane / bundle methods
- Points added incrementally until sufficient decrease obtained
- Finite number of additions until accepted step

Gradient sampling methods
- Points added randomly / incrementally until sufficient decrease obtained
- Sufficient number of iterations with “good” steps

In any case: convergence guarantees require \( \{W_k\} \) to be uniformly positive definite and bounded on a sufficient number of accepted steps
Outline

Contribution

Self-Correcting Properties of BFGS-type Updating

Proposed Framework

Numerical Experiments

Summary
Matlab implementation

Random instances of max-of-affine plus strongly convex quadratic, i.e.,

\[ f(x) = \max_{i \in \{1, \ldots, m\}} \{ a_i^T x + b_i \} + c^T x + \frac{1}{2} x^T Q x \]

with \( n = m = 100 \); varying numbers of “active” affine functions at \( x_\star = 0 \)

Algorithms:

- **BFGS**: BFGS w/ Wolfe line search
- **B**: Bundle method
- **B-SC**: ... w/ self-correcting BFGS (guarantees)
- **B-free**: ... w/ unadulterated BFGS
- **GS**: Gradient sampling (guarantees)
- **GS-SC**: ... w/ self-correcting BFGS (guarantees)
- **GS-free**: ... w/ unadulterated BFGS
Relative performance measures: $\kappa(Q) = 100$

function evaluations:

<table>
<thead>
<tr>
<th># act.</th>
<th>BFGS</th>
<th>B</th>
<th>B-SC</th>
<th>B-free</th>
<th>GS</th>
<th>GS-SC</th>
<th>GS-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>2.7861</td>
<td>1.6154</td>
<td>0.6976</td>
<td>79.111</td>
<td>1.0801</td>
<td>1.0801</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1.9192</td>
<td>1.2771</td>
<td>1.0580</td>
<td>158.698</td>
<td>1.0149</td>
<td>1.0127</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1.4433</td>
<td>1.0293</td>
<td>1.0462</td>
<td>218.103</td>
<td>1.0975</td>
<td>1.0975</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0.9760</td>
<td>0.7573</td>
<td>0.9222</td>
<td>241.187</td>
<td>1.0042</td>
<td>1.0042</td>
</tr>
</tbody>
</table>

gradient evaluations:

<table>
<thead>
<tr>
<th># act.</th>
<th>BFGS</th>
<th>B</th>
<th>B-SC</th>
<th>B-free</th>
<th>GS</th>
<th>GS-SC</th>
<th>GS-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>3.4729</td>
<td>2.0136</td>
<td>0.8695</td>
<td>16.001</td>
<td>1.0858</td>
<td>1.0858</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>3.0148</td>
<td>2.0063</td>
<td>1.6620</td>
<td>32.704</td>
<td>1.0406</td>
<td>1.0375</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2.6174</td>
<td>1.8667</td>
<td>1.8973</td>
<td>47.674</td>
<td>1.1433</td>
<td>1.1433</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1.9266</td>
<td>1.4950</td>
<td>1.8205</td>
<td>54.882</td>
<td>1.0098</td>
<td>1.0098</td>
</tr>
</tbody>
</table>

GS very poor, but adding BFGS yields great improvements; B-SC and B-free better than B; self-correcting BFGS improves both bundle and gradient sampling methods.
Relative performance measures: $\kappa(Q) = 100$

function evaluations:

<table>
<thead>
<tr>
<th># act.</th>
<th>BFGS</th>
<th>B</th>
<th>B-SC</th>
<th>B-free</th>
<th>GS</th>
<th>GS-SC</th>
<th>GS-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>2.7861</td>
<td>1.6154</td>
<td>0.6976</td>
<td>79.111</td>
<td>1.0801</td>
<td>1.0801</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1.9192</td>
<td>1.2771</td>
<td>1.0580</td>
<td>158.698</td>
<td>1.0149</td>
<td>1.0127</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1.4433</td>
<td>1.0293</td>
<td>1.0462</td>
<td>218.103</td>
<td>1.0975</td>
<td>1.0975</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0.9760</td>
<td>0.7573</td>
<td>0.9222</td>
<td>241.187</td>
<td>1.0042</td>
<td>1.0042</td>
</tr>
</tbody>
</table>

gradient evaluations:

<table>
<thead>
<tr>
<th># act.</th>
<th>BFGS</th>
<th>B</th>
<th>B-SC</th>
<th>B-free</th>
<th>GS</th>
<th>GS-SC</th>
<th>GS-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>3.4729</td>
<td>2.0136</td>
<td>0.8695</td>
<td>16.001</td>
<td>1.0858</td>
<td>1.0858</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>3.0148</td>
<td>2.0063</td>
<td>1.6620</td>
<td>32.704</td>
<td>1.0406</td>
<td>1.0375</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2.6174</td>
<td>1.8667</td>
<td>1.8973</td>
<td>47.674</td>
<td>1.1433</td>
<td>1.1433</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1.9266</td>
<td>1.4950</td>
<td>1.8205</td>
<td>54.882</td>
<td>1.0098</td>
<td>1.0098</td>
</tr>
</tbody>
</table>

- **GS** very poor, but adding BFGS yields great improvements
- **B-SC** and **B-free** better than **B**
- self-correcting BFGS improves both bundle and gradient sampling methods
Relative performance measures: $\kappa(Q) = 1000$

function evaluations:

<table>
<thead>
<tr>
<th># act.</th>
<th>BFGS</th>
<th>B</th>
<th>B-SC</th>
<th>B-free</th>
<th>GS</th>
<th>GS-SC</th>
<th>GS-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>5.9193</td>
<td>5.5070</td>
<td>0.4741  (3)</td>
<td>111.425</td>
<td>0.9806</td>
<td>0.9831</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>3.8184</td>
<td>3.6010</td>
<td>0.5912  (2)</td>
<td>158.768</td>
<td>1.0490</td>
<td>1.0494</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>3.2655</td>
<td>3.0035</td>
<td>1.0220  (0)</td>
<td>193.947</td>
<td>1.0008</td>
<td>1.0235</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>2.9943</td>
<td>2.8077</td>
<td>1.4598  (6)</td>
<td>303.429</td>
<td>0.9943</td>
<td>0.9943</td>
</tr>
</tbody>
</table>

gradient evaluations:

<table>
<thead>
<tr>
<th># act.</th>
<th>BFGS</th>
<th>B</th>
<th>B-SC</th>
<th>B-free</th>
<th>GS</th>
<th>GS-SC</th>
<th>GS-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>6.9029</td>
<td>6.4220</td>
<td>0.5529  (3)</td>
<td>27.890</td>
<td>0.9924</td>
<td>0.9945</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4.7267</td>
<td>4.4575</td>
<td>0.7318  (2)</td>
<td>39.922</td>
<td>1.0424</td>
<td>1.0398</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>4.3938</td>
<td>4.0412</td>
<td>1.3751  (0)</td>
<td>47.516</td>
<td>1.0026</td>
<td>1.0277</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>4.4746</td>
<td>4.1958</td>
<td>2.1814  (6)</td>
<td>72.748</td>
<td>0.9930</td>
<td>0.9930</td>
</tr>
</tbody>
</table>

- similar conclusions, but **B-free** now unreliable (11 failures of 80 problems)
Minimum and maximum eigenvalues
Minimum and maximum eigenvalues
Minimum and maximum eigenvalues
Minimum and maximum eigenvalues
Outline

Contribution

Self-Correcting Properties of BFGS-type Updating

Proposed Framework

Numerical Experiments

Summary
Contributions

Proposed a quasi-Newton method for nonsmooth optimization

- **unifying framework** covering
  - cutting plane / bundle methods (convex only)
  - gradient sampling methods (nonconvex)
- exploit **self-correcting** properties of BFGS-type updates
- properties of **Hessians** offer useful bounds for inverse Hessians
- global convergence guarantees
- improved practical performance
  - different effects in cutting plane / bundle vs. gradient sampling...
  - worthwhile to explore this further...

Paper forthcoming...