A Superlinearly-Convergent Proximal Newton-Type Method for the Optimization of Finite Sums

Anton Rodomanov\textsuperscript{1,2} Dmitry Kropotov\textsuperscript{2,3}

\textsuperscript{1}Higher School of Economics
\textsuperscript{2}Bayesian Methods Research Group
\textsuperscript{3}Lomonosov Moscow State University

Moscow, Russia

22 June 2016
International Conference on Machine Learning (ICML-2016), New York, USA
Consider the minimization of the composite finite average:

$$\min_{x \in \mathbb{R}^d} \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right]$$

Assumptions:
- Each $f_i$ is twice-continuously differentiable and convex
- $h$ is a general convex function (but simple)
- $\phi$ is strongly convex

Examples:
- Linear regression
- Logistic regression
- CRF etc.
- $n$ is very large
Consider the minimization of the composite finite average:

$$\min_{x \in \mathbb{R}^d} \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right]$$

**Assumptions:**
- each $f_i$ is twice-continuously differentiable and convex
- $h$ is a general convex function (but simple)
- $\phi$ is strongly convex
Consider the minimization of the composite finite average:

$$\min_{x \in \mathbb{R}^d} \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right]$$

**Assumptions:**
- each $f_i$ is twice-continuously differentiable and convex
- $h$ is a general convex function (but simple)
- $\phi$ is strongly convex

**Examples:** linear regression, logistic regression, CRF etc.
Consider the minimization of the composite finite average:

$$\min_{x \in \mathbb{R}^d} \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right]$$

**Assumptions:**
- each $f_i$ is twice-continuously differentiable and convex
- $h$ is a general convex function (but simple)
- $\phi$ is strongly convex

**Examples:** linear regression, logistic regression, CRF etc.

- $n$ is very large
Motivation

We are interested in **incremental methods** [Bertsekas, 2011] whose iteration cost is independent of \( n \):
Motivation

We are interested in **incremental methods** [Bertsekas, 2011] whose iteration cost is independent of \( n \):

- **Stochastic methods** for \( \min_x \{ \mathbb{E}_z[f(x; z)] \} = \min_x \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\} \):
  - **Examples**: SGD [Robbins-Monro, 1951], oLBFGS [Schraudolph et al., 2007], AdaGrad [Duchi et al., 2011], SQN [Byrd et al., 2014], Adam [Kingma, 2014] etc.
  - **Iteration**: \( x_{k+1} = x_k - \alpha_k B_k \nabla f_k(x_k) \).
  - **Convergence rate**: sublinear, usually \( O(1/k) \).
Motivation

We are interested in incremental methods [Bertsekas, 2011] whose iteration cost is independent of $n$:

- **Stochastic methods** for $\min_x \{ \mathbb{E}_z[f(x; z)] \} = \min_x \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$:
  - **Examples:** SGD [Robbins-Monro, 1951], oLBFGS [Schraudolph et al., 2007], AdaGrad [Duchi et al., 2011], SQN [Byrd et al., 2014], Adam [Kingma, 2014] etc.
  - **Iteration:** $x_{k+1} = x_k - \alpha_k B_k \nabla f_k(x_k)$.
  - **Convergence rate:** sublinear, usually $O(1/k)$.

- **Methods** for $\min_x \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$:
  - **Examples:** SAG [Le Roux et al., 2012], SVRG [Johnson & Zhang, 2013], FINITO [Defazio et al., 2014b], SAGA [Defazio et al., 2014a], MISO [Mairal, 2015] etc.
  - **Main idea:** variance reduction.
  - **Convergence rate:** linear, $O(c^k)$.
We are interested in **incremental methods** [Bertsekas, 2011] whose iteration cost is independent of $n$:

- **Stochastic methods** for $\min_x \{ \mathbb{E}_z [f(x; z)] \} = \min_x \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$:
  - **Examples**: SGD [Robbins-Monro, 1951], oLBFGS [Schraudolph et al., 2007], AdaGrad [Duchi et al., 2011], SQN [Byrd et al., 2014], Adam [Kingma, 2014] etc.
  - **Iteration**: $x_{k+1} = x_k - \alpha_k B_k \nabla f_i(x_k)$.
  - **Convergence rate**: sublinear, usually $O(1/k)$.

- Methods for $\min_x \left\{ \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$:
  - **Examples**: SAG [Le Roux et al., 2012], SVRG [Johnson & Zhang, 2013], FINITO [Defazio et al., 2014b], SAGA [Defazio et al., 2014a], MISO [Mairal, 2015] etc.
  - **Main idea**: variance reduction.
  - **Convergence rate**: linear, $O(c^k)$.

**Goal**: an incremental method with a superlinear convergence rate.
Main contributions

Our main contributions:

- New method: Newton-type Incremental Method (NIM)
- Theorem establishing superlinear convergence of NIM
**NIM: Idea**

**Problem:** \( \min_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right] \).
NIM: Idea

**Problem:** \( \min \limits_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right] \).

- **Build the second-order Taylor approximation of each** \( f_i \): 
  \( f_i(x) \approx m^i_k(x) := f_i(v^i_k) + \nabla f_i(v^i_k)^\top(x - v^i_k) + \frac{1}{2}(x - v^i_k)^\top \nabla^2 f_i(v^i_k)(x - v^i_k) \).
NIM: Idea

Problem: \[ \min \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \].

- Build the second-order Taylor approximation of each \( f_i \):
  \[ f_i(x) \approx m_k^i(x) := f_i(v_k^i) + \nabla f_i(v_k^i)^\top (x - v_k^i) + \frac{1}{2} (x - v_k^i)^\top \nabla^2 f_i(v_k^i)(x - v_k^i). \]

- Then \( \phi(x) \approx m_k(x) := \frac{1}{n} \sum_{i=1}^{n} m_k^i(x) + h(x) \).
NIM: Idea

Problem: \( \min_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right] \).

- Build the second-order Taylor approximation of each \( f_i \):
  \[
  f_i(x) \approx m^i_k(x) := f_i(v^i_k) + \nabla f_i(v^i_k)^\top (x - v^i_k) + \frac{1}{2} (x - v^i_k)^\top \nabla^2 f_i(v^i_k)(x - v^i_k).
  \]

- Then \( \phi(x) \approx m_k(x) := \frac{1}{n} \sum_{i=1}^{n} m^i_k(x) + h(x) \).

- Find the minimizer of the model \( \bar{x}_k := \arg\min_x m_k(x) \).

- Choose next iterate: \( x_{k+1} = x_k + \alpha(\bar{x}_k - x_k) \).
**Problem:** \( \min_x \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \).

- Build the **second-order Taylor approximation** of each \( f_i \):
  \[
  f_i(x) \approx m_k^i(x) := f_i(v_k^i) + \nabla f_i(v_k^i)^\top (x - v_k^i) + \frac{1}{2} (x - v_k^i)^\top \nabla^2 f_i(v_k^i) (x - v_k^i).
  \]

- Then \( \phi(x) \approx m_k(x) := \frac{1}{n} \sum_{i=1}^{n} m_k^i(x) + h(x) \).

- Find the **minimizer of the model** \( \bar{x}_k := \arg\min_x m_k(x) \).

- Choose next iterate: \( x_{k+1} = x_k + \alpha (\bar{x}_k - x_k) \).

- **(Standard Newton method)** \( v_k^i = x_k \) for all \( i = 1, \ldots, n \).
Problem: \( \min_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right] \).

- Build the second-order Taylor approximation of each \( f_i \):
  \[
  f_i(x) \approx m_k^i(x) := f_i(v_k^i) + \nabla f_i(v_k^i)^\top (x - v_k^i) + \frac{1}{2} (x - v_k^i)^\top \nabla^2 f_i(v_k^i)(x - v_k^i).
  \]

- Then \( \phi(x) \approx m_k(x) := \frac{1}{n} \sum_{i=1}^{n} m_k^i(x) + h(x) \).

- Find the minimizer of the model \( \bar{x}_k := \text{argmin}_x m_k(x) \).

- Choose next iterate: \( x_{k+1} = x_k + \alpha(\bar{x}_k - x_k) \).

- (Standard Newton method) \( v_k^i = x_k \) for all \( i = 1, \ldots, n \).

- (NIM) Update only one \( v_k^i \): choose \( i_k \in \{1, \ldots, n\} \) and set
  \[
  v_{k+1}^i := \begin{cases} 
  x_{k+1} & \text{if } i = i_k, \\
  v_k^i & \text{otherwise}.
  \end{cases}
  \]

- Iteration cost is independent of \( n \).
Recall:

\[ m^i_k(x) = f_i(v^i_k) + \nabla f_i(v^i_k)^\top (x - v^i_k) + \frac{1}{2}(x - v^i_k)^\top \nabla^2 f_i(v^i_k)(x - v^i_k) \]

\[ m_k(x) = \frac{1}{n} \sum_{i=1}^{n} m^i_k(x) + h(x) \]
NIM: Model update

\[ m_k(x) = \frac{1}{n} \sum_{i=1}^{n} \left[ f_i(v_k^i) + \nabla f_i(v_k^i)^\top (x - v_k^i) + \frac{1}{2} (x - v_k^i)^\top \nabla^2 f_i(v_k^i)(x - v_k^i) \right] + h(x). \]
NIM: Model update

\[ m_k(x) = \frac{1}{n} \sum_{i=1}^{n} \left[ f_i(v_k^i) + \nabla f_i(v_k^i)^\top (x - v_k^i) + \frac{1}{2} (x - v_k^i)^\top \nabla^2 f_i(v_k^i)(x - v_k^i) \right] + h(x). \]

**Note:** \( m_k \) is a (composite) quadratic,

\[ m_k(x) = (g_k - u_k)^\top x + \frac{1}{2} x^\top H_k x + h(x) + \text{const}, \]

and determined only by the following three quantities:

\[ H_k := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v_k^i), \quad u_k := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v_k^i)v_k^i, \quad g_k := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v_k^i). \]
\[ m_k(x) = \frac{1}{n} \sum_{i=1}^{n} \left[ f_i(v_k^i) + \nabla f_i(v_k^i)^\top (x - v_k^i) + \frac{1}{2} (x - v_k^i)^\top \nabla^2 f_i(v_k^i)(x - v_k^i) \right] + h(x). \]

**Note:** \( m_k \) is a (composite) quadratic,

\[ m_k(x) = (g_k - u_k)^\top x + \frac{1}{2} x^\top H_k x + h(x) + \text{const}, \]

and determined only by the following three quantities:

\[ H_k := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v_k^i), \quad u_k := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v_k^i)v_k^i, \quad g_k := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v_k^i). \]

Since only one \( v_k^i \) is updated at every iteration, we have for \( i = i_k \)

\[ H_{k+1} = H_k + \frac{1}{n} \left[ \nabla^2 f_i(v_{k+1}^i) - \nabla^2 f_i(v_k^i) \right] \]

\[ u_{k+1} = u_k + \frac{1}{n} \left[ \nabla^2 f_i(v_{k+1}^i)v_{k+1}^i - \nabla^2 f_i(v_k^i)v_k^i \right] \]

\[ g_{k+1} = g_k + \frac{1}{n} \left[ \nabla f_i(v_{k+1}^i) - \nabla f_i(v_k^i) \right]. \]
**NIM: Algorithm**

**Input:** \( x_0, \ldots, x_{n-1} \in \mathbb{R}^d \): initial points; \( \alpha > 0 \): step length.

**Initialize model:** \( v^i := x_{i-1} \) for \( i = 1, \ldots, n \) and

\[
H := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i), \quad u := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i)v^i, \quad g := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i)
\]

for \( k \geq n - 1 \) do

[Rest of the algorithm text continues here]
NIM: Algorithm

**Input:** \( x_0, \ldots, x_{n-1} \in \mathbb{R}^d \): initial points; \( \alpha > 0 \): step length.

**Initialize model:** \( v^i := x_{i-1} \) for \( i = 1, \ldots, n \) and

\[
H := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i), \quad u := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i)v^i, \quad g := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i)
\]

for \( k \geq n - 1 \) do

**Compute minimizer:** \( \bar{x}_k := \arg\min_x \left[ (g - u)^T x + \frac{1}{2} x^T H x + h(x) \right] \)

**Make a step:** \( x_{k+1} := x_k + \alpha (\bar{x}_k - x_k) \)
NIM: Algorithm

**Input:** $x_0, \ldots, x_{n-1} \in \mathbb{R}^d$: initial points; $\alpha > 0$: step length.

**Initialize model:** $v^i := x_{i-1}$ for $i = 1, \ldots, n$ and

$$H := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i), \quad u := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i)v^i, \quad g := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i)$$

for $k \geq n - 1$ do

**Compute minimizer:** $\bar{x}_k := \arg\min_x [(g - u)^T x + \frac{1}{2} x^T H x + h(x)]$

**Make a step:** $x_{k+1} := x_k + \alpha (\bar{x}_k - x_k)$

**Update model** for $i := (k + 1) \mod n + 1$ (cyclic order):

$$H := H + \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1}) - \nabla^2 f_i(v^i) \right]$$
$$u := u + \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1})x_{k+1} - \nabla^2 f_i(v^i)v^i \right]$$
$$g := g + \frac{1}{n} \left[ \nabla f_i(x_{k+1}) - \nabla f_i(v^i) \right]$$
$$v^i := x_{k+1}$$

end for
**NIM: Algorithm**

**Input:** $x_0, \ldots, x_{n-1} \in \mathbb{R}^d$: initial points; $\alpha > 0$: step length.

**Initialize model:** $v^i := x_{i-1}$ for $i = 1, \ldots, n$ and

$$H := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i), \quad u := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i)v^i, \quad g := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i)$$

for $k \geq n - 1$ do

**Compute minimizer:** $\bar{x}_k := \text{argmin}_x [(g - u)^\top x + \frac{1}{2} x^\top H x + h(x)]$

**Make a step:** $x_{k+1} := x_k + \alpha(\bar{x}_k - x_k)$

**Update model** for $i := (k + 1) \mod n + 1$ (cyclic order):

$$H := H + \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1}) - \nabla^2 f_i(v^i) \right]$$
$$u := u + \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1})x_{k+1} - \nabla^2 f_i(v^i)v^i \right]$$
$$g := g + \frac{1}{n} \left[ \nabla f_i(x_{k+1}) - \nabla f_i(v^i) \right]$$
$$v^i := x_{k+1}$$

end for

**Note:** $H, u, g$ and $v^i$ are kept in memory.

**Required memory:** $\mathcal{O}(d^2 + nd)$. 

A. Rodomanov, D. Kropotov 
Superlinear Incremental Method NIM 
22 June 2016 7 / 16
Suppose $\nabla^2 f_i$ are Lipschitz-continuous with constant $M_f$. Assume $x^*$ is a minimizer of $\phi$ with $rac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(x^*) \succeq \mu_f I \succ 0$, and all the initial points are close enough to $x^*$: $\|x_i - x^*\| \leq R$ for $0 \leq i \leq n - 1$. Then the sequence of iterates $\{x_k\}$ of NIM with $\alpha \equiv 1$ converges to $x^*$ at an $R$-superlinear rate, i.e. there exist $\{z_k\}$ and $\{q_k\}$ such that for $k \geq n$

\begin{align*}
\|x_k - x^*\| &\leq z_k, \\
 z_{k+1} &\leq q_k z_k, \\
 q_k &\to 0,
\end{align*}

where $R := \mu_f 2 M_f$, $q_k := \left(1 - \frac{3}{4} n \right)^{2 \lceil k / n \rceil - 1}$. More precisely, the rate of convergence is $n$-step quadratic:

$z_{k+n} \leq M_f \mu_f z_k^2$. 
Suppose $\nabla^2 f_i$ are Lipschitz-continuous with constant $M_f$. Assume $x^*$ is a minimizer of $\phi$ with $\frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(x^*) \succeq \mu_f I \succ 0$, and all the initial points are close enough to $x^*$: $\|x_i - x^*\| \leq R$ for $0 \leq i \leq n - 1$.

Then the sequence of iterates $\{x_k\}$ of NIM with $\alpha \equiv 1$ converges to $x^*$ at an $R$-superlinear rate, i.e. there exist $\{z_k\}$ and $\{q_k\}$ such that for $k \geq n$

$$\|x_k - x^*\| \leq z_k,$$

$$z_{k+1} \leq q_k z_k,$$

$q_k \to 0$,

where

$$R := \frac{\mu_f}{2M_f}, \quad q_k := \left(1 - \frac{3}{4n}\right)^{2[k/n]-1}.$$

More precisely, the rate of convergence is $n$-step quadratic:

$$z_{k+n} \leq \frac{M_f}{\mu_f} z_k^2.$$
Convergence rate (global)

Problem: \( \min_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + h(x) \right] \).

Assume \( h(x) := \frac{\mu}{2} \|x\|^2 \).

Theorem

Denote the condition number of \( \phi \) as \( \kappa_\phi := (L_f + \mu)/\mu \) and the minimizer of \( \phi \) as \( x^* \). Then, for any initial points \( x_0, \ldots, x_{n-1} \), NIM with a constant step length \( \alpha \equiv \kappa_\phi^{-3}(1 + 19\kappa_\phi(n-1))^{-1} \) converges to \( x^* \) at a linear rate:

\[
\phi(x_k) - \phi(x^*) \leq c^k [\phi(x_0) - \phi(x^*)],
\]

where

\[
c := (1 - \kappa_\phi^{-4}(1 + 19\kappa_\phi(n-1))^{-1})^{\frac{1}{1+2(n-1)}}.
\]

N.B.: This result is qualitative.
NIM: Model minimization?

**Input:** \(x_0, \ldots, x_{n-1} \in \mathbb{R}^d\): initial points; \(\alpha > 0\): step length.

**Initialize model:** \(v^i := x_{i-1} \) for \(i = 1, \ldots, n\) and

\[
H := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i), \quad u := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i)v^i, \quad g := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i)
\]

for \(k \geq n - 1\) do

**Compute minimizer:** \(\bar{x}_k := \text{argmin}_x \left[(g - u)^\top x + \frac{1}{2}x^\top Hx + h(x)\right]\)

**Make a step:** \(x_{k+1} := x_k + \alpha(\bar{x}_k - x_k)\)

**Update model** for \(i := (k + 1) \mod n + 1\) (cyclic order):

\[
\begin{align*}
H &:= H + \frac{1}{n} \left[\nabla^2 f_i(x_{k+1}) - \nabla^2 f_i(v^i)\right] \\
u &:= u + \frac{1}{n} \left[\nabla^2 f_i(x_{k+1})x_{k+1} - \nabla^2 f_i(v^i)v^i\right] \\
g &:= g + \frac{1}{n} \left[\nabla f_i(x_{k+1}) - \nabla f_i(v^i)\right] \\
v^i &:= x_{k+1}
\end{align*}
\]

end for

If \(h \equiv 0\), then \(\bar{x}_k = H^{-1}(u - g)\). Otherwise, use an iterative method for finding \(\bar{x}_k\).

Idea: \(\bar{x}_k\) may be computed inexactly (as in inexact Newton methods).
**NIM: Model minimization?**

**Input:** \(x_0, \ldots, x_{n-1} \in \mathbb{R}^d\): initial points; \(\alpha > 0\): step length.

**Initialize model:** \(v^i := x_{i-1}\) for \(i = 1, \ldots, n\) and
\[
H := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i), \quad u := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i)v^i, \quad g := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i)
\]

for \(k \geq n - 1\) do

**Compute minimizer:** \(\bar{x}_k := \arg\min_x [(g - u)^T x + \frac{1}{2} x^T H x + h(x)]\)

**Make a step:** \(x_{k+1} := x_k + \alpha (\bar{x}_k - x_k)\)

**Update model** for \(i := (k + 1) \mod n + 1\) (cyclic order):
\[
H := H + \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1}) - \nabla^2 f_i(v^i) \right]
\]
\[
u^i := x_{k+1} - \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1})x_{k+1} - \nabla^2 f_i(v^i)v^i \right]
\]
\[
g := g + \frac{1}{n} \left[ \nabla f_i(x_{k+1}) - \nabla f_i(v^i) \right]
\]
\[
v^i := x_{k+1}
\]

end for

- If \(h \equiv 0\), then \(\bar{x}_k = H^{-1}(u - g)\).
NIM: Model minimization?

Input: $x_0, \ldots, x_{n-1} \in \mathbb{R}^d$: initial points; $\alpha > 0$: step length.

Initialize model: $v^i := x_{i-1}$ for $i = 1, \ldots, n$ and

$$H := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i), \quad u := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i) v^i, \quad g := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i)$$

for $k \geq n - 1$ do

Compute minimizer: $\bar{x}_k := \arg\min_x \left[ (g - u)^T x + \frac{1}{2} x^T H x + h(x) \right]$

Make a step: $x_{k+1} := x_k + \alpha (\bar{x}_k - x_k)$

Update model for $i := (k + 1) \mod n + 1$ (cyclic order):

$$H := H + \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1}) - \nabla^2 f_i(v^i) \right]$$

$$u := u + \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1}) x_{k+1} - \nabla^2 f_i(v^i) v^i \right]$$

$$g := g + \frac{1}{n} \left[ \nabla f_i(x_{k+1}) - \nabla f_i(v^i) \right]$$

$$v^i := x_{k+1}$$

end for

- If $h \equiv 0$, then $\bar{x}_k = H^{-1}(u - g)$.
- Otherwise, use an iterative method for finding $\bar{x}_k$. 
NIM: Model minimization?

**Input:** $x_0, \ldots, x_{n-1} \in \mathbb{R}^d$: initial points; $\alpha > 0$: step length.

**Initialize model:** $v^i := x_{i-1}$ for $i = 1, \ldots, n$ and

\[
H := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i), \quad u := \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(v^i)v^i, \quad g := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i)
\]

for $k \geq n - 1$ do

**Compute minimizer:** $\tilde{x}_k := \arg\min_x \left[ (g - u)^T x + \frac{1}{2} x^T H x + h(x) \right]$

**Make a step:** $x_{k+1} := x_k + \alpha(\tilde{x}_k - x_k)$

**Update model** for $i := (k + 1) \mod n + 1$ (cyclic order):

\[
H := H + \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1}) - \nabla^2 f_i(v^i) \right],
\]

\[
u^i := \frac{1}{n} \left[ \nabla^2 f_i(x_{k+1})x_{k+1} - \nabla^2 f_i(v^i)v^i \right] \]

\[g := g + \frac{1}{n} \left[ \nabla f_i(x_{k+1}) - \nabla f_i(v^i) \right] \]

end for

- If $h \equiv 0$, then $\tilde{x}_k = H^{-1}(u - g)$.
- Otherwise, use an iterative method for finding $\tilde{x}_k$.
- **Idea:** $\tilde{x}_k$ may be computed inexactlty (as in inexact Newton methods).
NIM: Inexact model minimization

**Problem:** \( \min_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \).

(Assume \( h \equiv 0 \) for simplicity.)

**Model:** \( m_k(x) = (g_k - u_k)^\top x + \frac{1}{2} x^\top H_k x + \text{const} \).

**NIM iteration:** \( x_{k+1} = x_k + \alpha (\bar{x}_k - x_k) \), where \( \bar{x}_k := \arg\min m_k(x) \).

**Inexact minimization:** instead of \( \bar{x}_k \), use \( \hat{x}_k \) such that
\[
\|\nabla m_k(\hat{x}_k)\| \leq \eta_k \|\nabla \phi(x_k)\|,
\eta_k := \left\{ 0.5, \sqrt{\|\nabla \phi(x_k)\|} \right\}.
\]
NIM: Inexact model minimization

**Problem:** \[ \min_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right]. \]

(Assume \( h \equiv 0 \) for simplicity.)

**Model:** \[ m_k(x) = (g_k - u_k)^\top x + \frac{1}{2} x^\top H_k x + \text{const}. \]

**NIM iteration:** \[ x_{k+1} = x_k + \alpha (\bar{x}_k - x_k), \] where \( \bar{x}_k := \arg\min m_k(x) \).

**Inexact minimization:** instead of \( \bar{x}_k \), use \( \hat{x}_k \) such that

\[ \|\nabla m_k(\hat{x}_k)\| \leq \eta_k \|\nabla \phi(x_k)\|, \quad \eta_k := \begin{cases} 0.5, & \sqrt{\|\nabla \phi(x_k)\|} \end{cases}. \]

**Problem:** cannot compute \( \|\nabla \phi(x_k)\| \) (this in incremental optimization!).
NIM: Inexact model minimization

**Problem:** \( \min_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \).

(Assume \( h \equiv 0 \) for simplicity.)

**Model:** \( m_k(x) = (g_k - u_k)^\top x + \frac{1}{2} x^\top H_k x + \text{const} \).

**NIM iteration:** \( x_{k+1} = x_k + \alpha(\bar{x}_k - x_k) \), where \( \bar{x}_k := \text{argmin} \ m_k(x) \).

**Inexact minimization:** instead of \( \bar{x}_k \), use \( \hat{x}_k \) such that
\[
\| \nabla m_k(\hat{x}_k) \| \leq \eta_k \|g_k\|, \quad \eta_k := \left\{ 0.5, \sqrt{\|g_k\|} \right\}.
\]

**Recall:** \( g_k := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v_k^i) \approx \nabla \phi(x_k) \).

Convergence rate remains superlinear!
Problem: \[
\min_x \left[ \phi(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right].
\]
(Assume \( h \equiv 0 \) for simplicity.)

Model: \( m_k(x) = (g_k - u_k)\top x + \frac{1}{2} x\top H_k x + \text{const.} \)

NIM iteration: \( x_{k+1} = x_k + \alpha(\bar{x}_k - x_k) \), where \( \bar{x}_k := \arg\min m_k(x) \).

Inexact minimization: instead of \( \bar{x}_k \), use \( \hat{x}_k \) such that
\[
\|\nabla m_k(\hat{x}_k)\| \leq \eta_k \|g_k\|, \quad \eta_k := \left\{ 0.5, \sqrt{\|g_k\|} \right\}.
\]

Recall: \( g_k := \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(v^i_k) \approx \nabla \phi(x_k) \).

Convergence rate remains superlinear!

For \( h \neq 0 \), all of this can be generalized using the composite gradient mapping (see paper for details).
Order of component selection (cyclic vs randomized)

- What if randomized order is used in NIM instead of cyclic?
Experiments ($\ell_2$-regularized logistic regression): Epochs

Residual in function $L^2$-reg, a9a ($n=32561$, $d=123$)

- NIM
- SAG
- Newton
- LBFGS
- SFO
- SGD

Residual in function $L^2$-reg, covtype ($n=581012$, $d=54$)

- NIM
- SAG
- Newton
- LBFGS
- SFO
- SGD
Experiments ($\ell_2$-regularized logistic regression): Real time

<table>
<thead>
<tr>
<th>L2-reg</th>
<th>$alpha$ ($n=500,000$, $d=500$)</th>
<th>mnist8m ($n=8,100,000$, $d=784$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NIM</td>
<td>SAG</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>1.91s</td>
<td>1.36s</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>13.37s</td>
<td>6.72s</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>28.56s</td>
<td>17.73s</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>36.65s</td>
<td>36.04s</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>46.66s</td>
<td>1.0m</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>53.92s</td>
<td>1.5m</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>57.63s</td>
<td>2.0m</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>1.0m</td>
<td>2.7m</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>1.1m</td>
<td>3.5m</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>1.2m</td>
<td>4.3m</td>
</tr>
</tbody>
</table>

**Inner solver:** Conjugate Gradient Method.
### Experiments ($\ell_1$-regularized logistic regression): Real time

<table>
<thead>
<tr>
<th>L1-reg</th>
<th>alpha ($n=500,000$, $d=500$)</th>
<th>mnist8m ($n=8,100,000$, $d=784$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NIM</td>
<td>SAG</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>26.76s</td>
<td>1.31s</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>44.94s</td>
<td>6.52s</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>55.56s</td>
<td>17.26s</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>1.1m</td>
<td>35.51s</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>1.3m</td>
<td>1.0m</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>1.3m</td>
<td>1.5m</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>1.4m</td>
<td>2.1m</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>1.5m</td>
<td>2.9m</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>1.6m</td>
<td>3.8m</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>1.6m</td>
<td>4.8m</td>
</tr>
</tbody>
</table>

**Inner solver:** Fast Gradient Method [Nesterov, 2013].
The presented Newton-type Incremental Method (NIM) is the first incremental method with a superlinear rate of convergence. Method NIM can be seen as an incremental variant of the standard Newton method. NIM has the same advantages and disadvantages as the classic Newton method:

- Fast superlinear rate of convergence with the unit step length.
- Superlinear convergence is guaranteed only locally.
- Not applicable to high-dimensional problems.

Thank you!