Newton-Raphson consensus for distributed convex optimization

Luca Schenato

Department of Information Engineering - University of Padova, Italy
URL: http://automatica.dei.unipd.it/people/schenato.html

June 15, 2013 – ECC Workshop on Distributed Optimization in Large Networks and its Applications

Contributors

Angelo Cenedese
University of Padova

Gianluigi Pillonetto
University of Padova

Damiano Varagnolo
KTH, Sweden

Filippo Zanella
a-pole.com, sellf.com

Structure of the presentation

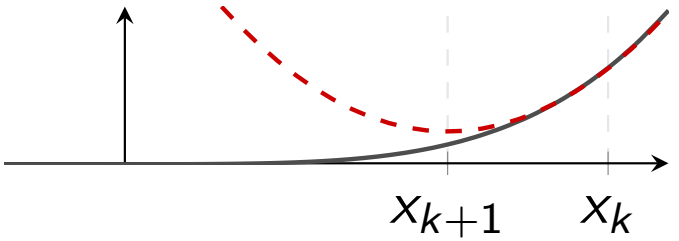
- Centralized Newton-Raphson: an overview
- Distributed Newton-Raphson: the scalar scenario
- Distributed Newton-Raphson: convergence proof ideas
- Distributed Newton-Raphson: the multidimensional scenario
- Simulations
- Conclusions and references

Centralized Newton-Raphson: the rationale behind & properties

Newton-Raphson: scalar case

Goal: find minimum of convex $f(x)$

Idea: approximate function $f(x)$ with a parabola



$$\widehat{f}(x) = \frac{1}{2}a(x-b)^2 + c$$

Match slope and curvature at point x_n:

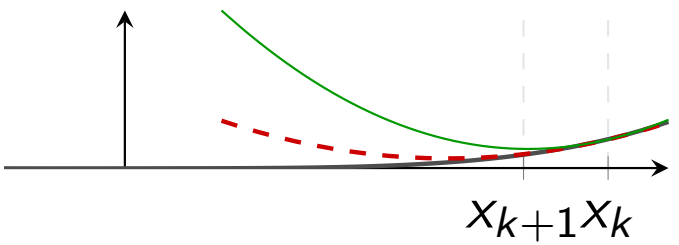
$$\begin{aligned} f(x_k) &= \widehat{f}(x_k) = \tfrac{1}{2}a(x_k-b)^2 + c \\ f'(x_k) &= \widehat{f}'(x_k) = a(x_k-b) \\ f''(x_k) &= \widehat{f}''(x_k) = a \end{aligned} \quad \Rightarrow \quad \begin{aligned} a &= f''(x_k) \\ b &= x_k - \frac{f'(x_k)}{f''(x_k)} \\ c &= * \end{aligned}$$

Jump to the minimum:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Gradient Descent: scalar case

Idea: approximate function $f(x)$ with a parabola with unitary curvature



$$\widehat{f}(x) = \frac{1}{2}(x-b)^2 + c$$

Match slope at x_k:

$$\begin{aligned} f(x_k) &= \widehat{f}(x_k) = \tfrac{1}{2}(x_k - b)^2 + c \\ f'(x_k) &= \widehat{f}'(x_k) = x_k - b \end{aligned} \quad \Rightarrow \quad \begin{aligned} b &= x_k - f'(x_k) \\ c &= * \end{aligned}$$

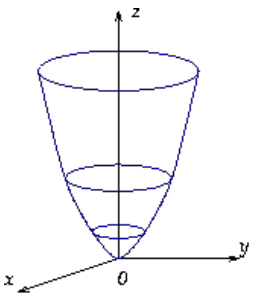
Jump to the minimum:

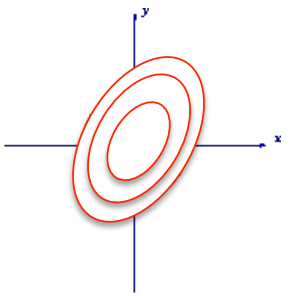
$$x_{k+1} = x_k - f'(x_k)$$

Newton-Raphson: multivariable case

Idea: approximate function $f(x)$ with a parabola

$$\widehat{f}(x) = \tfrac{1}{2}(x-b)^T A(x-b) + c,$$
$$b \in \mathbb{R}^n, A \in \mathbb{R}^{n\times n}$$





Match slope and curvature at point x_k:

$$\begin{aligned} \nabla f(x_k) &= \nabla \widehat{f}(x_k) = A(x_k - b) \\ \nabla^2 f(x_k) &= \nabla^2 \widehat{f}''(x_k) = A \end{aligned} \quad \Rightarrow \quad \begin{aligned} A &= \nabla^2 f(x_k) \\ b &= x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \end{aligned}$$

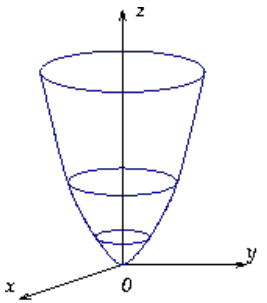
Jump to the minimum:

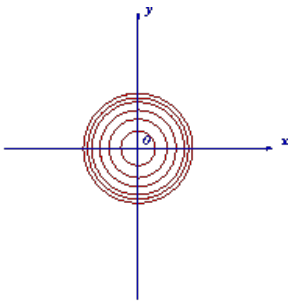
$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

Gradient Descent: multivariable

Idea: approximate function $f(x)$ with a parabola with unitary curvature

$$\widehat{f}(x) = \tfrac{1}{2}\|x - b\|^2 + c$$
$$(A = I)$$





Match slope at x_k:

$$\nabla f(x_k) = \nabla \widehat{f}(x_k) = x_k - b$$

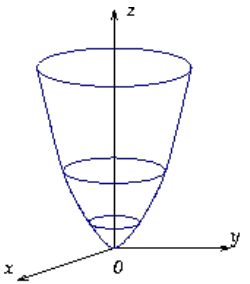
Jump to the minimum:

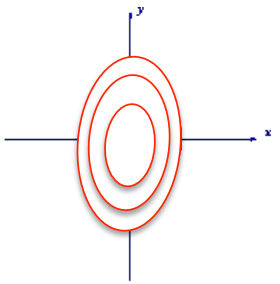
$$x_{k+1} = x_k - \nabla f(x_k)$$

Jacobi: multivariable

Idea: approximate function $f(x)$ with a parabola with parallel axes

$$\widehat{f}(x) = \tfrac{1}{2}(x-b)^T A(x-b) + c,$$
$$A = \operatorname{diag}\{a_1, \ldots, a_n\}$$





Match slope and axis curvature at x_k:

$$\nabla f(x_k) = \nabla \widehat{f}(x_k) = A(x_k - b)$$
$$\left[\nabla^2 f(x_k)\right]_{ii} = a_i$$

Jump to the minimum:

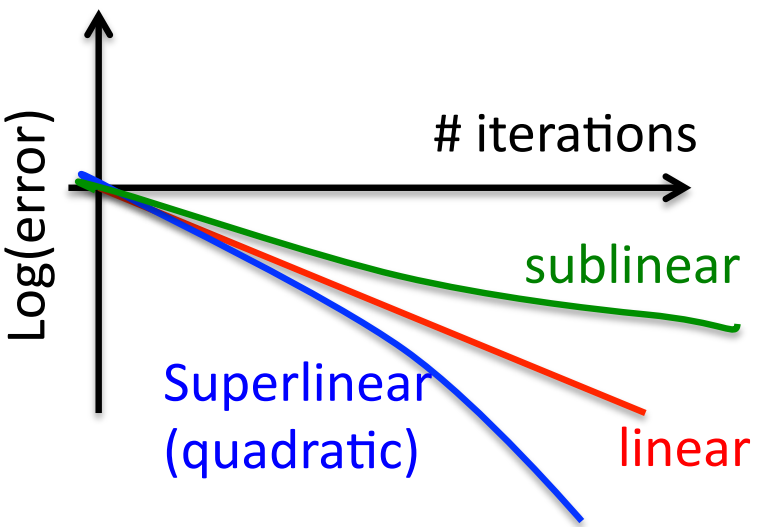
$$x_{k+1} = x_k - \left(\operatorname{diag}(\nabla^2 f(x_k))\right)^{-1} \nabla f(x_k)$$

Centralized Newton-Raphson (NR): properties

- if f is quadratic, then minimization is performed in 1 step
- Newton step is invariant w.r.t. affine changes of coordinates
- if $f \in C^2$, strongly convex, and Hessian is uniformly Lipschitz, i.e.,

$$\left\| \nabla^2 f(\boldsymbol{x}_1) - \nabla^2 f(\boldsymbol{x}_2) \right\|_2 \leq L \left\| \boldsymbol{x}_1 - \boldsymbol{x}_2 \right\|_2$$

then for $x \approx x^*$ convergence rate is ***quadratic (super-linear, doubly exponential)***



Centralized NR in practice

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \varepsilon(\nabla^2 f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k)$$

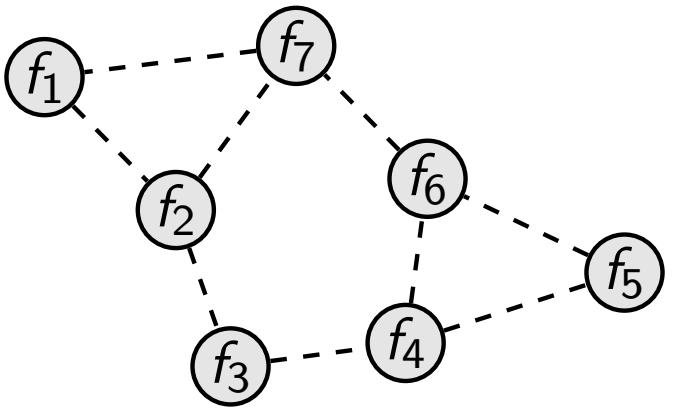
- practical implementations perform line search, e.g. $\varepsilon_k^* = \min_\varepsilon f(\mathbf{x}_{k+1})$. For $\varepsilon = 1$ could diverge if $\mathbf{x}_0$ far away.
- convergence follows two phases: first *damped* (linear convergence) then *quadratic* (optimal $\varepsilon \approx 1$)
- computational burden to obtain $\nabla^2 f(\boldsymbol{x})$ can be alleviated using *quasi*-Newton methods:

$$\Delta \boldsymbol{x} = -B_k^{-1}\nabla f(\boldsymbol{x}_k)$$

where B_k^{-1} is an estimate of the Hessian using $\nabla f(\boldsymbol{x}_{k-1})$

Distributed Newton-Raphson: the scalar scenario

Networked systems



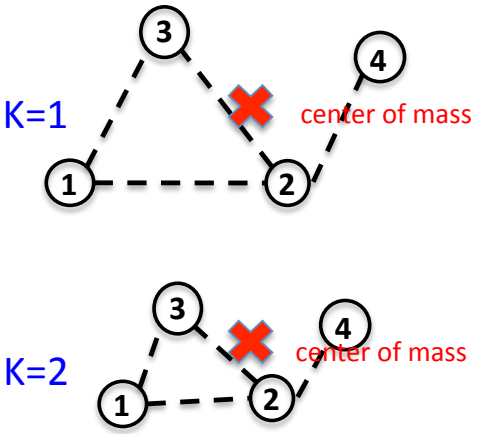
Assumption: neighbours cooperate to find the optimum of an additively separable cost:

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x), \quad x^* = \operatorname{argmin}_x f(x)$$

Average Consensus algorithm

Linear Distributed algorithm to compute averages:

$$x_i \in \mathbb{R}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$



Matrix P doubly stochastic, nonnegative, associated graph strongly connected

$$x(k+1) = Px(k)$$

$$\mathbf{1}^T P = \mathbf{1}^T, P\mathbf{1} = \mathbf{1}, P \geq 0, P^N > 0$$

$$P = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{3}{4} \end{bmatrix}, \Longrightarrow \begin{array}{l} \lim_{k\to\infty} x_i(k) = \frac{1}{N}\sum_{i=1}^{N} x_i(0), \ \forall i \\ \text{exponentially fast rate} = \mathrm{esr}(P) \end{array}$$

Center of mass preserved ! Works also for time-varying $P(k)$: e.g. gossip

Naive application of Consensus: the wrong way !

Centralized Gradient Descent (to simplify notation $x_k = x, x_{k+1} = x^+$):

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x) \Longrightarrow x^+ = x - \frac{1}{N}\sum_{i=1}^{N} f_i'(x)$$

Some notation:

x_i: local copies of estimated minimum, $\mathbf{x} = [x_1 \cdots x_n]^T$
y_i: local copies of estimated global gradient, $\mathbf{y} = [y_1 \cdots y_n]^T$

Naive Distributed Gradient Descent Algorithm:

(1)	$y_i = f_i'(x_i)$	local gradient
(2)	$\mathbf{y}^+ = P\mathbf{y}$	estimated global gradient via communication
(3)	$x_i^+ = x_i - y_i^+$	local descent step

NOT WORKING !!

Naive application of Consensus: the wrong way ! (cont'd)

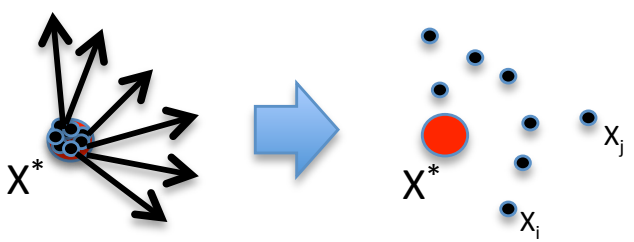
(1) $y_i = f_i'(x_i)$ local gradient

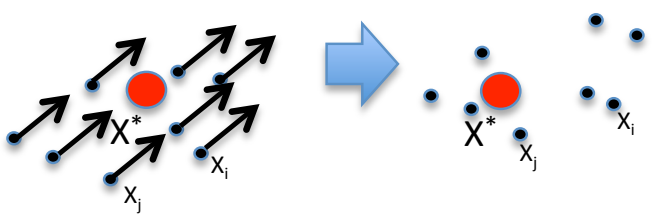
(2) $\mathbf{y}^+ = P\mathbf{y}$ estimated global gradient via communication

(3) $x_i^+ = x_i - y_i^+$ local descent step

Why it does not work:

- even if $x_i = x^* \ \forall i$, unless $P = \frac{1}{N}\mathbf{1}\mathbf{1}^T$ (complete graph), then the x_i^+'s s will spread around $\Longrightarrow$ x^* ***is not an asymptotic equilibrium point***
- even if $P = \frac{1}{N}\mathbf{1}\mathbf{1}^T$ (complete graph), unless $x_i = x_j \forall i, j$, then $x_i^+ \neq x_j^+ \Longrightarrow$ ***they agree on a direction not on a point***





Back to Newton-Raphson intuition

Approximate **each** $f_i(x)$ with a parabola

$$\widehat{f}_i(x) = \frac{1}{2} a_i (x - b_i)^2 + c_i \Longrightarrow \begin{array}{ll} \widehat{f}(x) & = \frac{1}{N} \sum_{i=1}^{n} \left(\frac{1}{2} a_i (x - b_i)^2 + c_i \right) \\ & = \frac{1}{2} a (x - x^*)^2 \end{array}$$

Match slope and curvature at point x_i:

$$\begin{array}{l} f_i'(x_i) = \widehat{f}_i'(x_i) = a_i(x_i - b_i) \\ f_i''(x_i) = \widehat{f}_i''(x_i) = a_i \end{array} \Rightarrow \begin{array}{l} a_i = f_i''(x_i) \\ a_i b_i = f_i''(x_i) x_i - f_i'(x_i) \end{array}$$

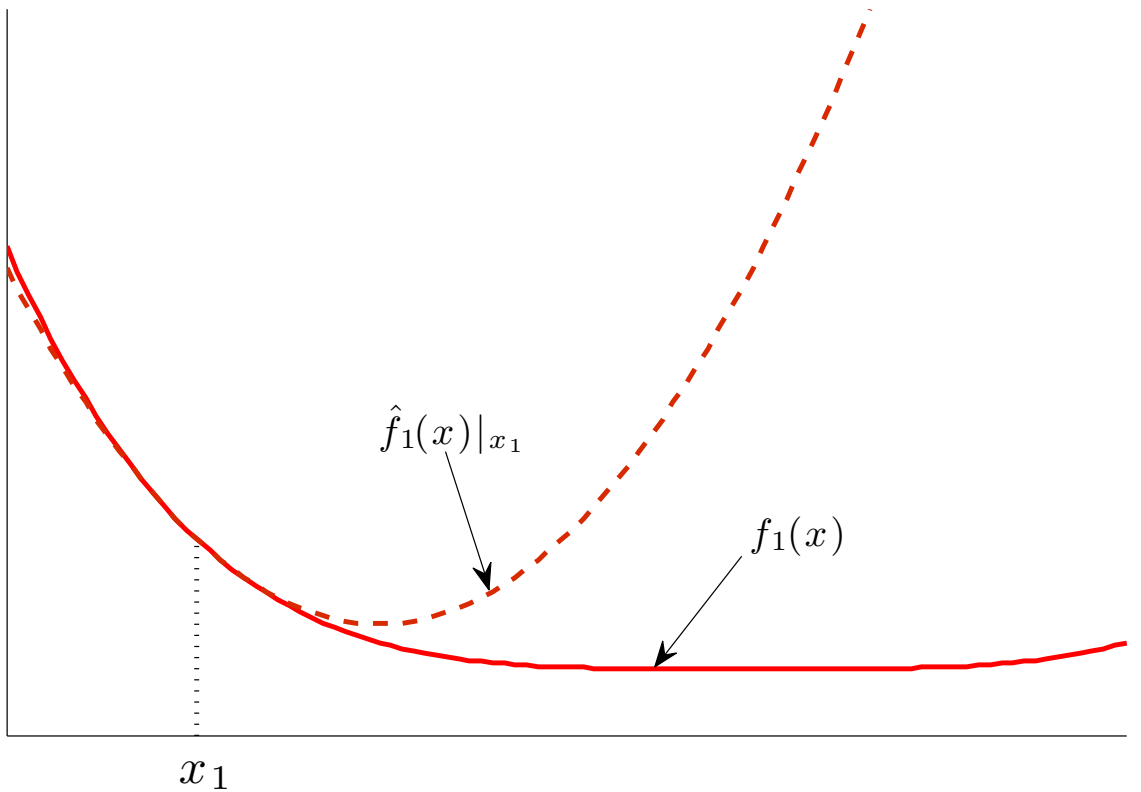
Jump to the minimum of $\widehat{f}(x)$:

$$x_i^+ = x^* = \frac{\sum_{i=1}^{N} a_i b_i}{\sum_{i=1}^{N} a_i} = \frac{\frac{1}{N} \sum_{i=1}^{N} a_i b_i}{\frac{1}{N} \sum_{i=1}^{N} a_i} = \frac{\frac{1}{N} \sum_{i=1}^{N} f_i''(x_i) x_i - f_i'(x_i)}{\frac{1}{N} \sum_{i=1}^{N} f_i''(x_i)}$$

Graphical interpretation

- $a_i b_i = f_i''(x_i)x_i - f_i'(x_i)$
- $a_i = f_i''(x_i)$

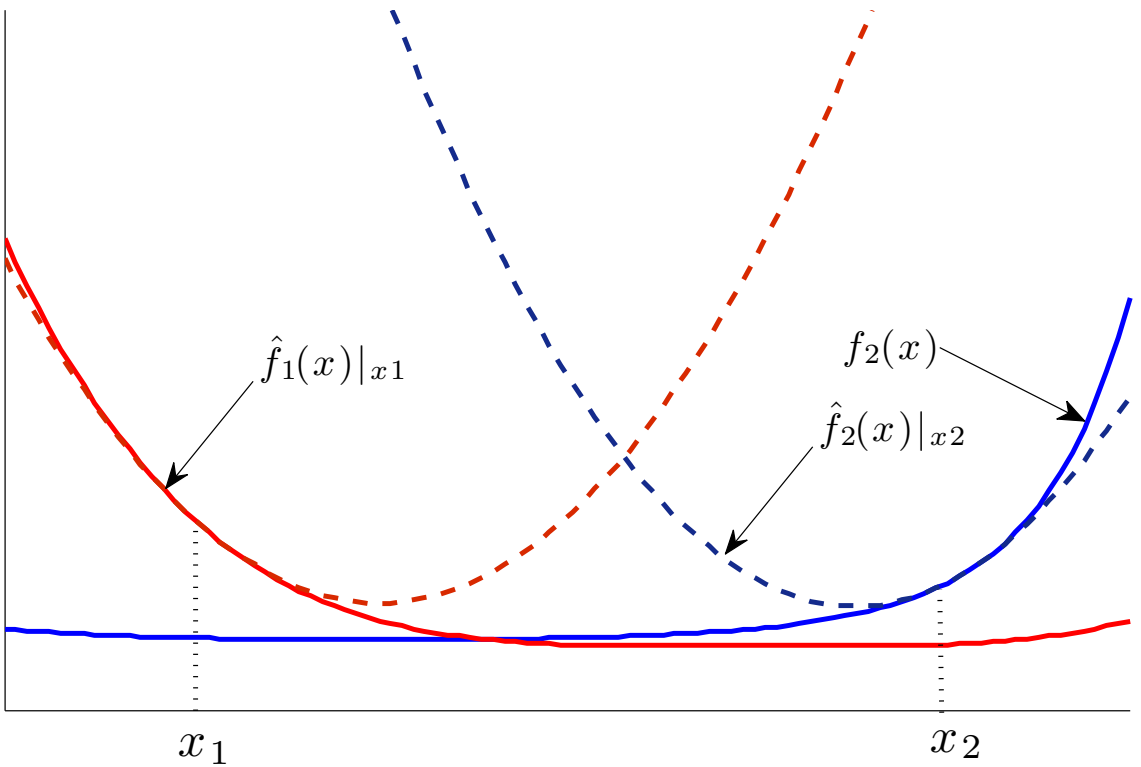
$$\Rightarrow \quad x^* = \frac{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)x_i - f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$$



Graphical interpretation

- $a_i b_i = f_i''(x_i)x_i - f_i'(x_i)$
- $b_i = f_i''(x_i)$

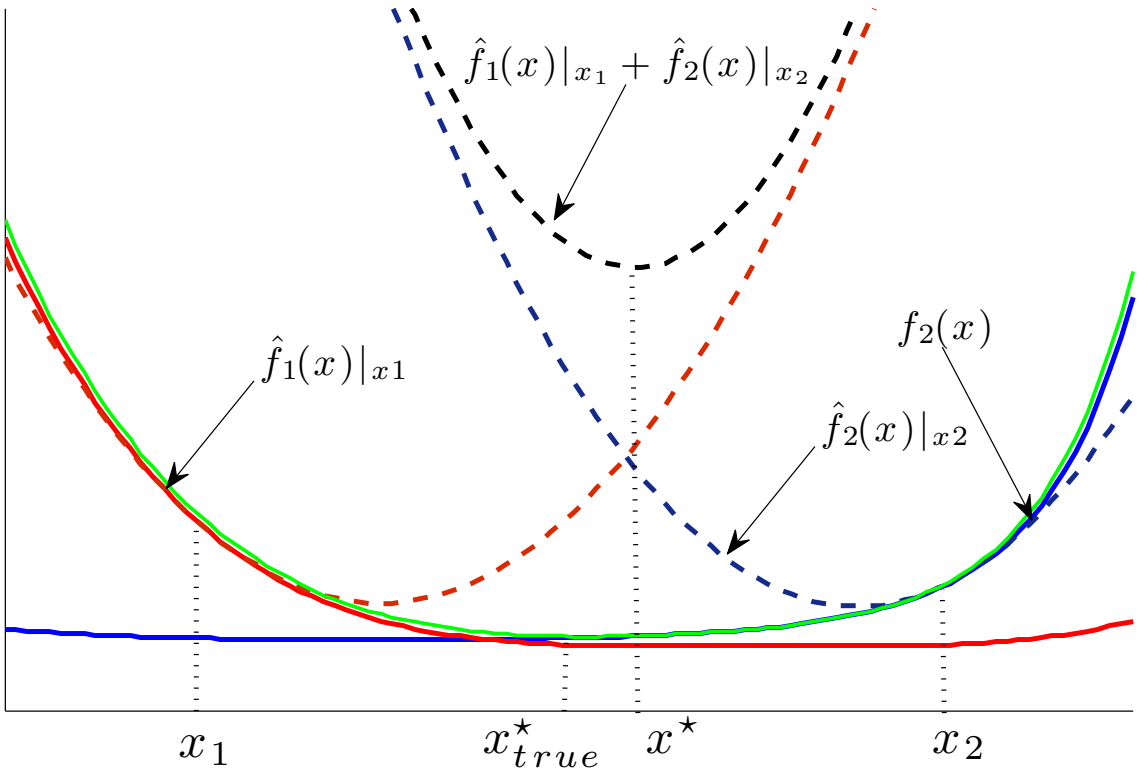
$$\Rightarrow \quad x^* = \frac{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)x_i - f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$$



Graphical interpretation

- $a_i b_i = f_i''(x_i)x_i - f_i'(x_i)$
- $b_i = f_i''(x_i)$

$$\Rightarrow \quad x^* = \frac{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)x_i - f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$$



Centralized vs Distributed Newton-Raphson

Is the minimum of $\widehat{f}(x)$ a good approximation of the true minimum of $f(x)$? Minimum of global $\widehat{f}(x)$:

$$x_i^+ = x^* = \frac{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)x_i - f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$$

Not clear, but if all points are the same, i.e. $x_i = x \ \forall i$, then:

$$x_i^+ = x^+ = x - \frac{\frac{1}{N}\sum_{i=1}^{N} f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)} = x - \frac{f'(x)}{f''(x)}$$

Intuition: If x_i are close to each other, then x^* is a good estimate of the true minimum, therefore $x^* - x_i$ is a good direction for x_i.

Towards a consensus-based Newton-Raphson

Algorithm

1. initialise local variables:
   - $y_i(0) := f_i''(x_i(0))x_i(0) - f_i'(x_i(0))$
   - $z_i(0) := f_i''(x_i(0))$
2. run 2 average consensus (P doubly stochastic):
   - $\mathbf{y}(k+1) = P\mathbf{y}(k)$,
   - $\mathbf{z}(k+1) = P\mathbf{z}(k)$
3. locally compute $x_i(k+1) = \dfrac{y_i(k+1)}{z_i(k+1)}$

Towards a consensus-based Newton-Raphson

Algorithm

1. initialise local variables:
   - $y_i(0) := f_i''(x_i(0))x_i(0) - f_i'(x_i(0))$
   - $z_i(0) := f_i''(x_i(0))$
2. run 2 average consensus (P doubly stochastic):
   - $\mathbf{y}(k+1) = P\mathbf{y}(k)$,
   - $\mathbf{z}(k+1) = P\mathbf{z}(k)$
3. locally compute $x_i(k+1) = \dfrac{y_i(k+1)}{z_i(k+1)}$

Problem:

All local estimate converge to consensus $y_i(k) \to \bar{y}(0), z_i(k) \to \bar{z}(0)$ therefore also $x_i(k) \to x^*(0)$, but $x^*(0)$ depends on initial condition. One could run K steps and then restart algorithm with $y_i(0) \leftarrow f_i''(x_i(K))x_i(K) - f_i'(x_i(K)), \quad z_i(0) \leftarrow f_i''(K)$: ***too slow***

The (synchronous) consensus-based Newton-Raphson

Fixes:

- change initial condition of consensus step to track the changing x_i
- move x_i slowly to let consensus variable (y_i, z_i) to converge

Algorithm

1. define local variables:
   - $g_i(k) := f_i''(x_i(k))x_i(k) - f_i'(x_i(k)), \quad g_i(-1) = 0, \quad y_i(0) = 0$
   - $h_i(k) := f_i''(x_i(k)), \quad h_i(-1) = 0, \quad z_i(0)$
2. run 2 average consensus (P doubly stochastic):
   - $\mathbf{y}(k+1) = P\mathbf{y}(k) + \mathbf{g}(k) - \mathbf{g}(k-1)$,
   - $\mathbf{z}(k+1) = P\mathbf{z}(k) + \mathbf{h}(k) - \mathbf{h}(k-1)$
3. locally compute $x_i(k+1) = (1-\varepsilon)x_i(k) + \varepsilon \dfrac{y_i(k+1)}{z_i(k+1)}$

Why $\boldsymbol{h}(k) - \boldsymbol{h}(k-1)$ and not simply $\boldsymbol{h}(k)$?

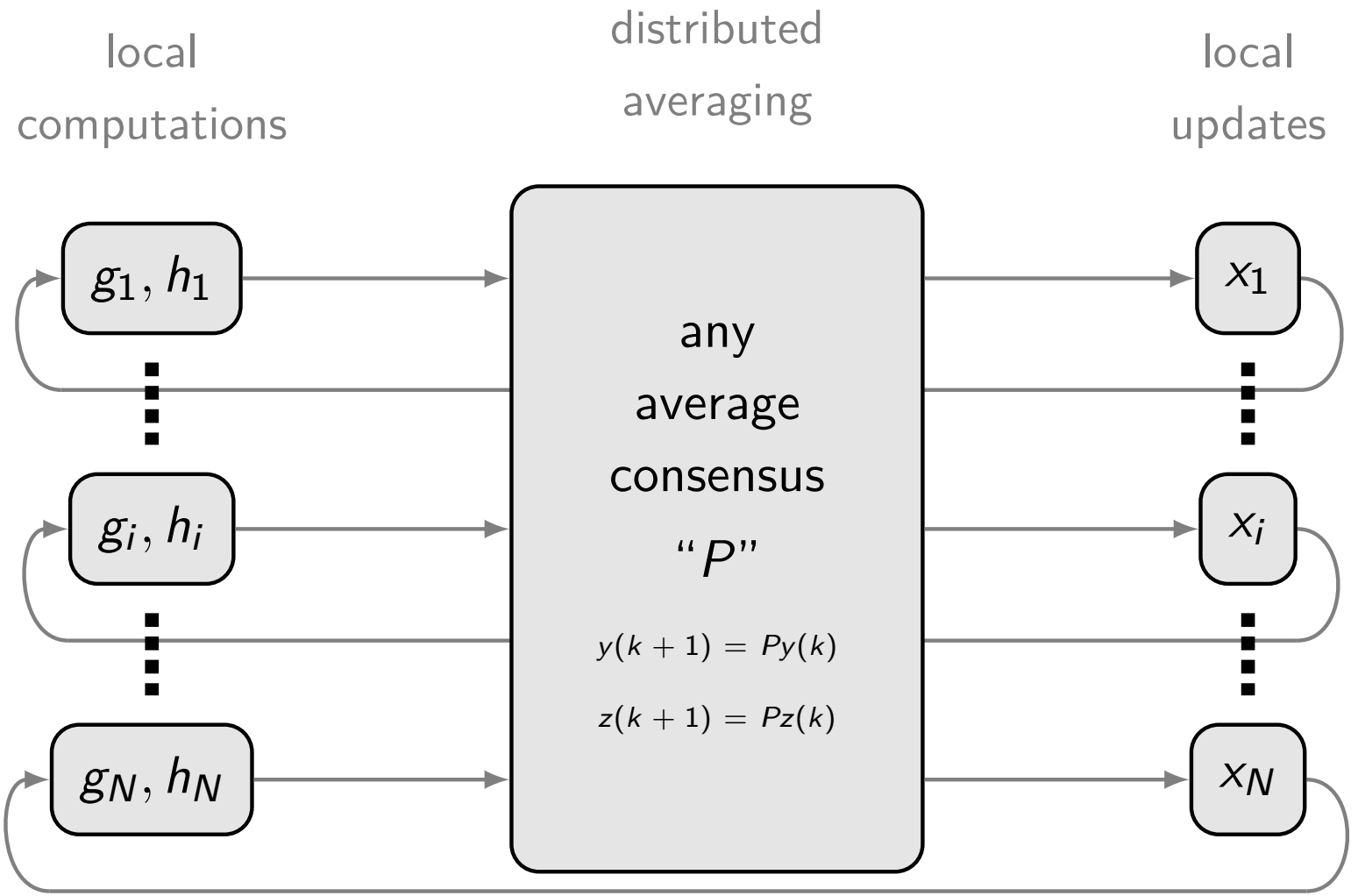
Plain average consensus would lead to integration, differently:

$$\mathbf{z}(k+1) = P\mathbf{z}(k) + \mathbf{h}(k) - \mathbf{h}(k-1)$$
$$\mathbf{z}(0) = 0, \quad \mathbf{h}(-1) = 0$$
$$\Downarrow$$
$$\frac{1}{N}\sum_{i=1}^{N} z_i(k+1) = \frac{1}{N}\sum_{i=1}^{N} h_i(x_i(k)), \quad \forall k!!$$

Therefore, if $z_i(k) - z_j(k) \overset{k\to\infty}{\longrightarrow} 0$, then

$$z_i(k+1) \longrightarrow \frac{1}{N}\sum_{i=1}^{N} h_i(x_i(k)) = \frac{1}{N}\sum_{i=1}^{N} f_i''(x_i(k)), \quad \forall i$$

Block diagram representation



$$g_i(k) = f_i''(x_i(k))x_i(k) - f_i'(x_i(k))$$
$$h_i(k) = f_i''(x_i(k))$$

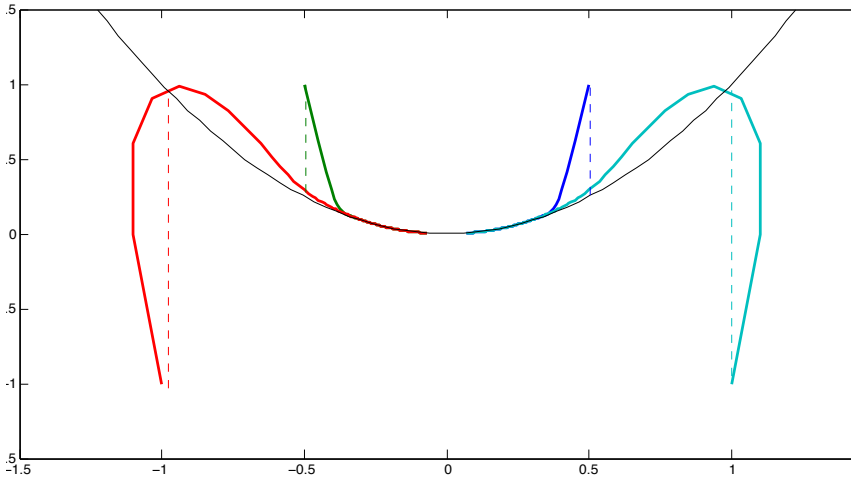
$$x_i(k+1) = (1-\varepsilon)x_i(k) + \varepsilon\frac{y_i(k+1)}{z_i(k+1)}$$

Distributed Newton-Raphson: convergence proof ideas

Singular Perturbation Theory: an example

Coupled dynamics:

$$\begin{aligned} \dot{x} &= -xy^2 && \text{slow dynamics} \\ \varepsilon\dot{y} &= -y + x^2 && \text{fast dynamics} \\ \left(\dot{y} = \tfrac{1}{\varepsilon}(-y + x^2)\right) & \end{aligned}$$



Idea: decouple time scales

- freeze slow dynamics, i.e. $x = \textit{constant}$
- find equilibrium points for fast dynamics, i.e. $y = x^2$
- verify if fast dynamics is asymptotically stable: $\dot{y} = -y$ (OK)
- substitute equilibrium into slow dynamics and verify is systems is asymptotically stable, $\dot{x} = -x^5$
- plus some other technical conditions $\Longrightarrow$ ***coupled system is asymptotically stable if ε sufficiently small***

Convergence based on Singular Perturbation Theory

Algorithm

$$\begin{cases} \mathbf{x}(0) = \mathbf{y}(0) = \mathbf{z}(0) = \mathbf{g}(\mathbf{x}(-1)) = \mathbf{h}(\mathbf{x}(-1)) = \mathbf{0} & \text{initialization} \\ \hline \mathbf{y}(k+1) = P\mathbf{y}(k) + \mathbf{g}(\mathbf{x}(k)) - \mathbf{g}(\mathbf{x}(k-1)) & \text{fast dynamics} \\ \mathbf{z}(k+1) = P\mathbf{z}(k) + \mathbf{h}(\mathbf{x}(k)) - \mathbf{h}(\mathbf{x}(k-1)) & \\ \hline x_i(k+1) = (1-\varepsilon)x_i(k) + \varepsilon \frac{y_i(k+1)}{z_i(k+1)} & \text{slow dynamics} \end{cases}$$

Proof sketch:

Fast dynamics

If $\varepsilon \approx 0$, then $\mathbf{x}(k+1) \approx \mathbf{x}(k) = \mathbf{x}$ (constant)

$\Longrightarrow y_i(k+1) \to \frac{1}{N}\sum_{i=1}^{N} g_i(x_i) = \frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)x_i - f_i'(x) = \bar{g}(\mathbf{x}), \quad \forall i$

$\Longrightarrow z_i(k+1) \to \frac{1}{N}\sum_{i=1}^{N} h_i(x_i) = \frac{1}{N}\sum_{i=1}^{N} f_i''(x_i) = \bar{h}(\mathbf{x}), \quad \forall i$

$\bar{g}(\mathbf{x}), \bar{h}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$

Convergence based on Singular Perturbation Theory

Fast dynamics

If $\varepsilon \approx 0$, then $\mathbf{x}(k+1) \approx \mathbf{x}(k) = \mathbf{x}$ (constant)

$\Longrightarrow y_i(k+1) = \frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)x_i - f_i'(x) = \bar{g}(\mathbf{x}), \quad \forall i$

$\Longrightarrow z_i(k+1) = \frac{1}{N}\sum_{i=1}^{N} f_i''(x_i) = \bar{h}(\mathbf{x}), \quad \forall i$

Slow dynamics: perturbed system

Insert equilibrium point of fast dynamics into slow dynamics:

$x_i(k+1) = (1-\varepsilon)x_i(k) + \varepsilon\frac{\bar{g}(\mathbf{x}(k))}{\bar{h}(\mathbf{x}(k))}, \forall i$

Same forcing term, therefore $\lim_{k\to\infty} x_i(k) - x_j(k) = 0$.

Convergence based on Singular Perturbation Theory

Slow dynamics: perturbed system

Insert equilibrium point of fast dynamics into slow dynamics:

$x_i(k+1) = (1-\varepsilon)x_i(k) + \varepsilon \frac{\bar{g}(\mathbf{x}(k))}{\bar{h}(\mathbf{x}(k))}, \forall i$

Same forcing term, therefore $\lim_{k\to\infty} x_i(k) - x_j(k) = 0$.

Slow dynamics: unperturbed system

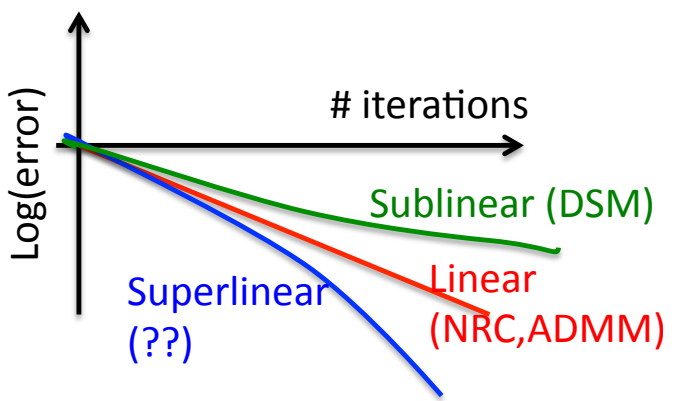
Assume $x_i = x_j = \bar{x}$:

$$\begin{aligned}
\bar{x}^+ &= (1-\varepsilon)\bar{x} + \varepsilon \frac{\bar{g}(\bar{x}\mathbf{1})}{\bar{h}(\bar{x}\mathbf{1})} \\
&= (1-\varepsilon)\bar{x} + \varepsilon \frac{\frac{1}{N}\sum_{i=1}^{N} f_i''(\bar{x})\bar{x} - f_i'(\bar{x})}{\frac{1}{N}\sum_{i=1}^{N} f_i''(\bar{x})} \\
&= (1-\varepsilon)\bar{x} + \varepsilon \left(\bar{x} - \frac{\frac{1}{N}\sum_{i=1}^{N} f_i'(\bar{x})}{\frac{1}{N}\sum_{i=1}^{N} f_i''(\bar{x})} \right) \\
&= \bar{x} - \varepsilon \frac{f'(\bar{x})}{f''(\bar{x})}
\end{aligned}$$

Centralized Newton-Raphson !!

Formal results

- If f_i are ***quadratic*** $\Longrightarrow$ ***Global exponential convergence with rate*** $\mathrm{sr}(P)$ ***for*** $\varepsilon = 1$ for ***any connected graph***
- If graph is ***complete*** $\Longrightarrow$ ***Centralized Newton-Raphson***
- Under mild conditions ($f_i \in \mathcal{C}^3$ and convex) $\Longrightarrow$ ***Local Exponential Stability*** for $0 < \varepsilon < \varepsilon_c$
- Under more restrictive conditions (global boundedness of $\frac{f' \star f'''}{(f'')^2}$ and f'') $\Longrightarrow$ ***Global Exponential Stability*** for $0 < \varepsilon < \varepsilon_c$
- ***Convergence is "only" linear*** due to consensus: need time to pass information around



Distributed Newton-Raphson: the multivariable scenario

The Multivariable consensus-based Newton-Raphson

Derivation of the algorithm

Algorithm

1. define local variables:
   - $g_i(k) := \nabla^2 f_i(x_i(k))x_i(k) - \nabla f_i(x_i(k)), g_i(-1) = y_i(0) = 0, \in \mathbb{R}^n$
   - $H_i(k) := \nabla^2 f_i(x_i(k)), \quad H_i(-1) = Z_i(0) = 0, \quad \in \mathbb{R}^{n \times n}$
2. run 2 average consensus (P doubly stochastic):
   - $\mathbf{y}(k+1) = P\mathbf{y}(k) + \mathbf{g}(k) - \mathbf{g}(k-1)$
   - $\mathbf{Z}(k+1) = P\mathbf{Z}(k) + \mathbf{h}(k) - \mathbf{h}(k-1)$
3. locally compute $x_i(k+1) = (1-\varepsilon)x_i(k) + \varepsilon Z_i(k+1)^{-1} y_i(k+1)$

Need to compute averages and inversions of matrices:

- $O(n^2)$ communication complexity & memory requirements
- $O(n^3)$ computational complexity

.

Distributed Gradient Descent Revised

Approximate **each** $f_i(x)$ with a parabola with ***unitary curvature***:

$$\widehat{f}_i(x) = \frac{1}{2}(x - b_i)^2 + c_i \Longrightarrow \begin{array}{ll} \widehat{f}(x) & = \frac{1}{N}\sum_{i=1}^{n}\left(\frac{1}{2}(x - b_i)^2 + c_i\right) \\ & = \frac{1}{2}(x - x^*)^2 + c \end{array}$$

Match slope x_i:

$$f_i'(x_i) = \widehat{f}_i'(x_i) = (x_i - b_i) \;\Rightarrow\; b_i = x_i - f_i'(x_i)$$

Jump to the minimum of $\widehat{f}(x)$:

$$x_i^+ = x^* = \frac{1}{N}\sum_{i=1}^{N} b_i = \frac{1}{N}\sum_{i=1}^{N} x_i - f_i'(x_i)$$

The (synchronous) consensus-based Gradient Descent

Derivation of the algorithm

The correct algorithm

1. define local variables:
   - $g_i(k) := x_i(k) - f_i'(x_i(k)), \quad g_i(-1) = 0, \quad y_i(0) = 0$
2. run 1 average consensus (P doubly stochastic):
   - $\mathbf{y}(k+1) = P\mathbf{y}(k) + \mathbf{g}(k) - \mathbf{g}(k-1),$
3. locally compute

$$\begin{aligned} x_i(k+1) &= (1-\varepsilon)x_i(k) + \varepsilon y_i(k+1) \\ &= x_i(k) + \varepsilon\left(y_i(k+1) - x_i(k)\right) \end{aligned}$$

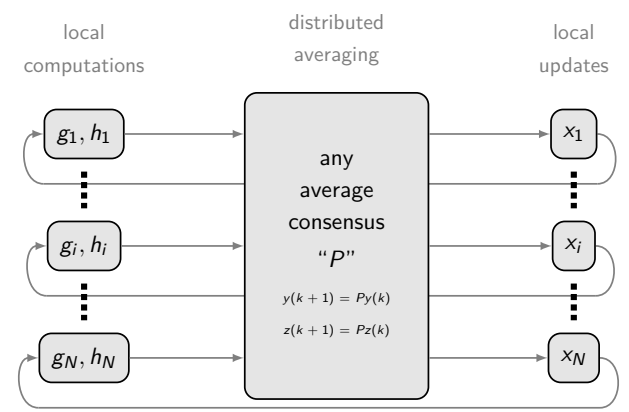
The Naive Gradient Descent algorithm

(1)	$y_i = f_i'(x_i)$	local gradient
(2)	$\mathbf{y}^+ = P\mathbf{y}$	estimated global gradient via communication
(3)	$x_i^+ = x_i - y_i^+$	local descent step

Extensions

- Simplified Multivariable:
  - Distributed Gradient Descent: $O(n)$ complexity, only ∇f needed
  - Distributed Jacobi: $O(n)$ complexity, only $\nabla f, [\nabla^2 f]_{ii}$ needed
- Asynchronous: straightforward implementation. Some uniform persistency requirements for global convergence
- Flexible: by changing the consensus block can be adapted to different scenarios:

- Accelerated: diffusion-based consensus
- Broadcast communication: no need for symmetric gossip (ratio consensus)
- Directed Graphs
- Packet loss



$g_i(k) = f_i''(x_i(k))x_i(k) - f_i'(x_i(k))$
$h_i(k) = f_i''(x_i(k))$

$x_i(k+1) = (1-\varepsilon)x_i(k) + \varepsilon \frac{y_i(k+1)}{z_i(k+1)}$

Distributed Newton-Raphson: simulations

Simulations: SVM Classification with synchronous NR

http://archive.ics.uci.edu/ml/datasets/Spambase

$\chi \in \mathbb{R}^4$: frequency of specific words,

$y \in \{\text{spam, non-spam}\}$

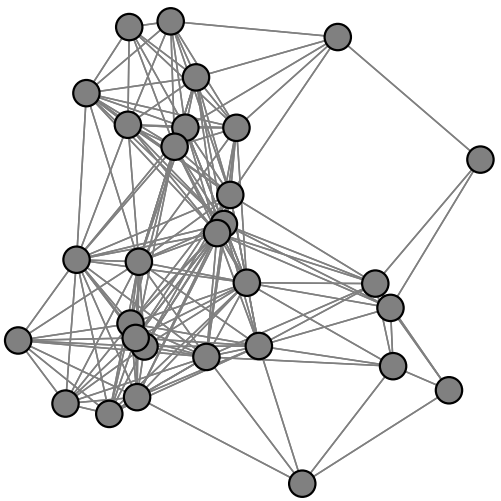
$(\mathbf{x}, x_0) \in \mathbb{R}^5$: separating hyperplane parameters

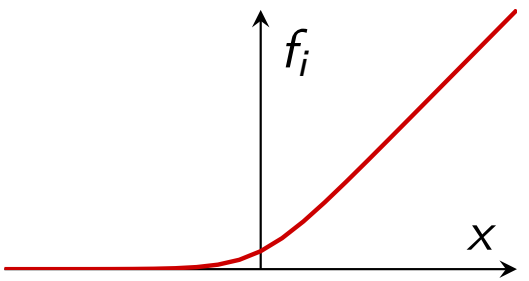
Connected graphs with 30 nodes

Local cost functions:

$$f_i(x) := \sum_{j=1}^{30} \log\left(1+\exp\left(-y_j\left(\chi_j^T \boldsymbol{x} + x_0\right)\right)\right) + \gamma \|\boldsymbol{x}\|_2^2 .$$

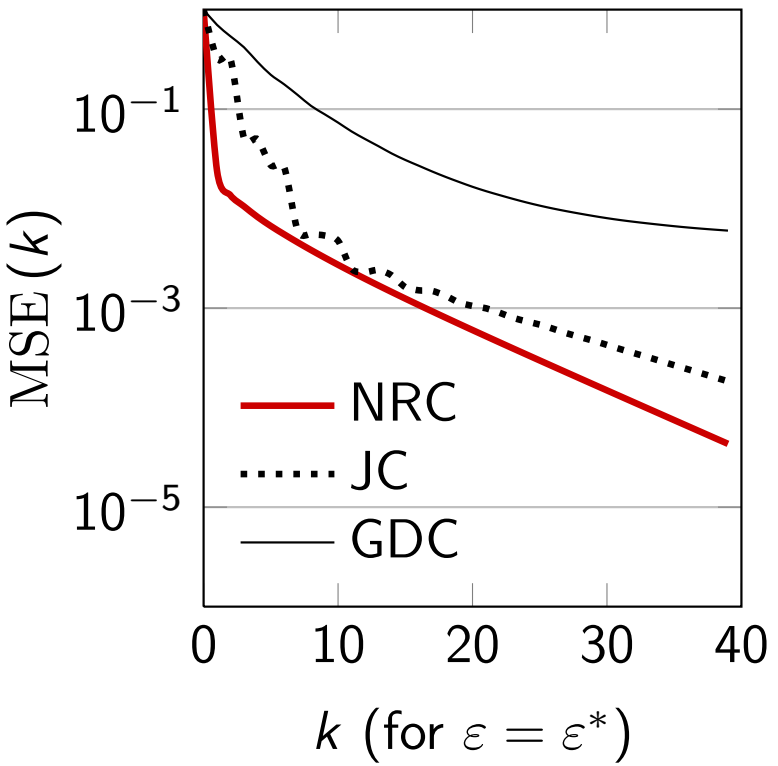
Spam Filters:





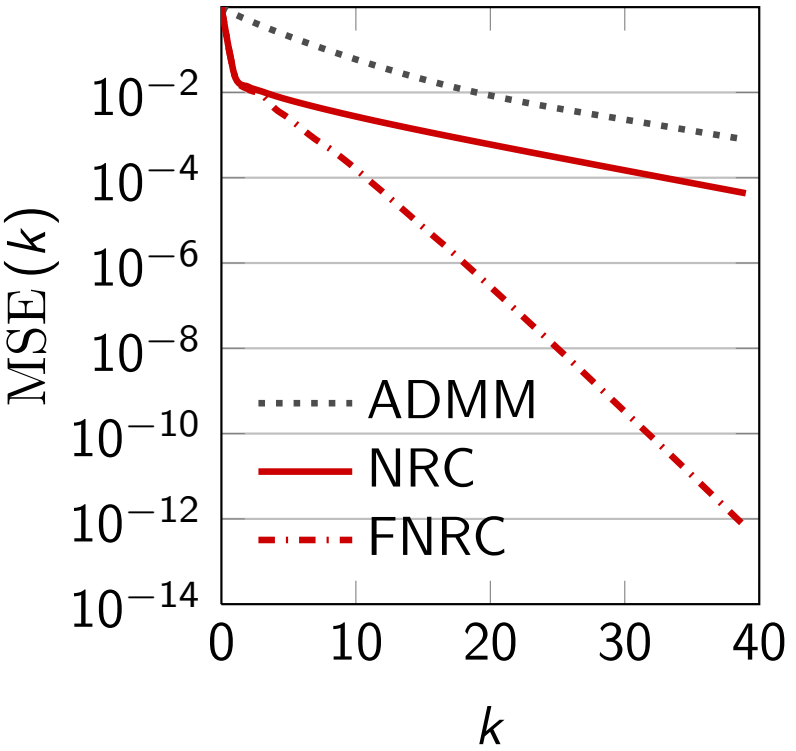
Simulations: SVM Classification with synchronous NR

Consensus-based algorithms:



NRC=Newton-Raphson Consensus
JC= Jacobi Consensus
GDC = Gradient Descent Consensus

Comparison with other algorithms



ADMM=Alternating Direction Multipliers Method
NRC=Newton-Raphson Consensus
FNRC= Newton-Raphson with Fast Consensus (diffusive)

Simulations: Robust Regression with synchronous NR

http://archive.ics.uci.edu/ml/datasets/Housing

$\chi \in \mathbb{R}^4$: size, distance from downtown

$y \in \mathbb{R}$, house price

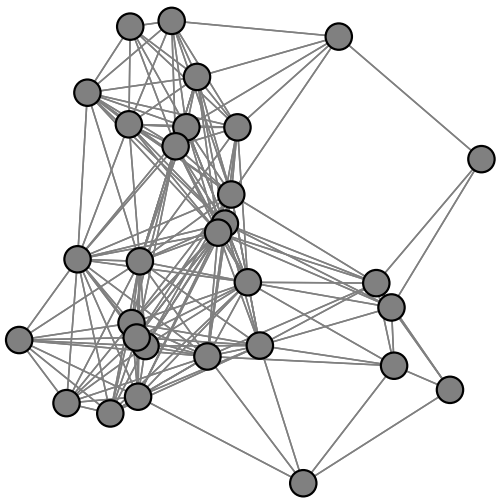
$(\mathbf{x}, x_0) \in \mathbb{R}^5$: parameters to be computed

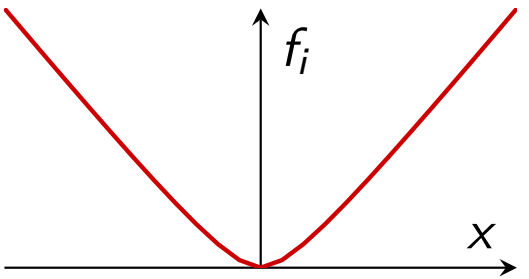
Connected graphs with 30 nodes

Local cost functions:

$$f_i(x) := \sum_{j=1}^{30} \frac{\left(y_j - \chi_j^T \boldsymbol{x} - x_0\right)^2}{\left|y_j - \chi_j^T \boldsymbol{x} - x_0\right| + \beta} + \gamma \|\boldsymbol{x}\|_2^2 .$$

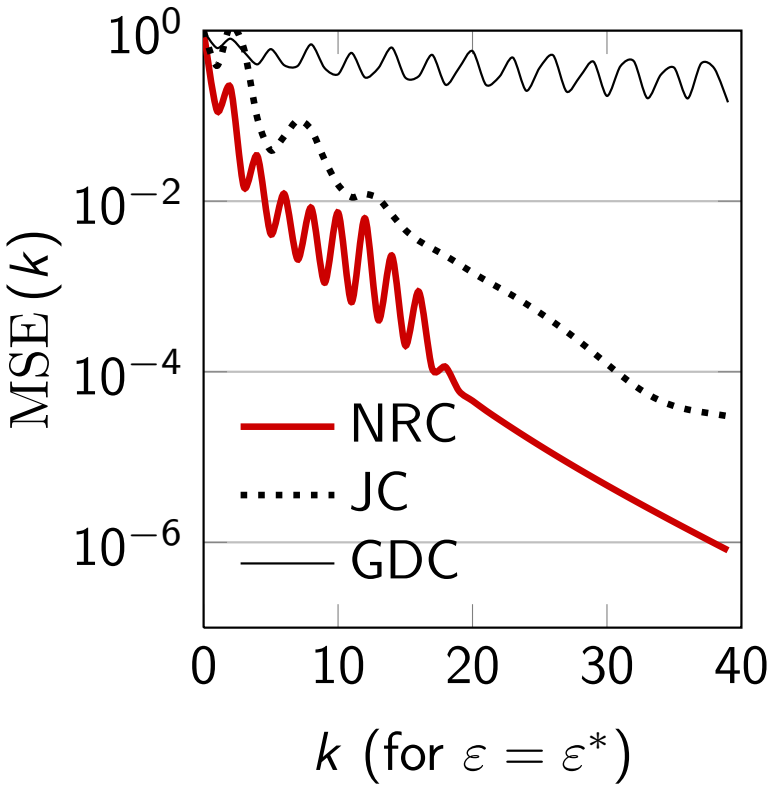
Housing Price Predictors:





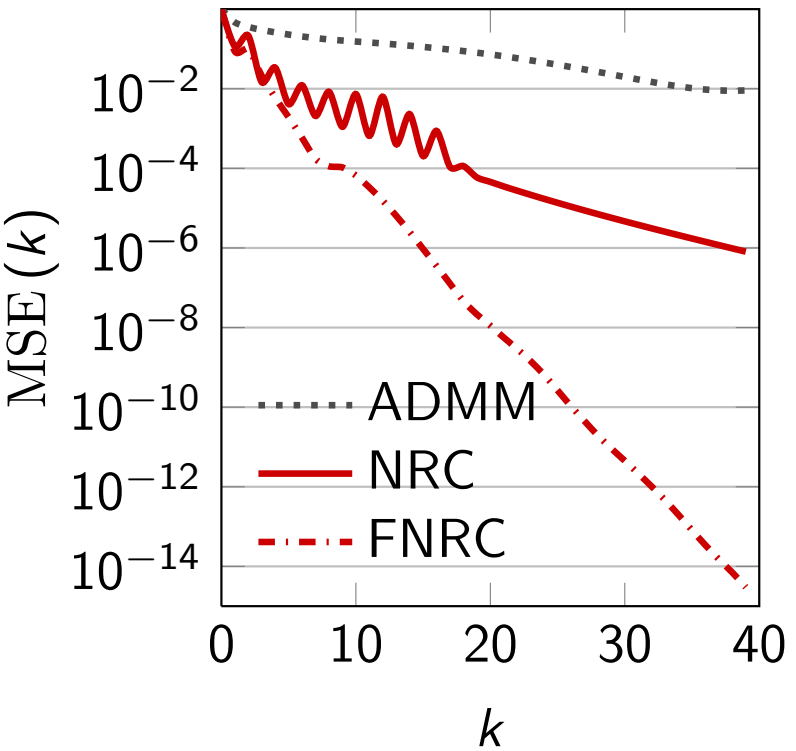
Simulations: Robust Regression with synchronous NR

Consensus-based algorithms:



NRC=Newton-Raphson Consensus
JC= Jacobi Consensus
GDC = Gradient Descent Consensus

Comparison with other algorithms



ADMM=Alternating Direction Multipliers Method
NRC=Newton-Raphson Consensus
FNRC= Newton-Raphson with Fast Consensus (diffusive)

Conclusions and references

Conclusions

Takeaway messages

- new distributed optimisation method
- it takes advantage of standard consensus algorithms (plug-and-play)
- its potentials are still mainly unexplored

Future work

- adaptive local stepsize $\varepsilon_i(k)$
- non-differentiable cost functions
- quasi-Newton methods
- constraints
- distributed interior point methods
- extensive comparisons based on real data with ADMM&co

Publications (1/2)

Synchronous

F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, L. Schenato (2011)
Newton-Raphson consensus for distributed convex optimization
IEEE Conference on Decision and Control (CDC'11)

F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, L. Schenato (2012)
Multidimensional Newton-Raphson consensus for distrib. convex optimization
American Control Conference (ACC'12)

D. Varagnolo, F. Zanella, A. Cenedese, G. Pillonetto, L. Schenato
Newton-Raphson Consensus for Distributed Convex Optimization
IEEE Transactions on Automatic Control (submitted)

Publications (2/2)

Asynchronous

F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, L. Schenato (2012)
Asynchronous Newton-Raphson Consensus for Distributed Convex Optimization
3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys'12)

Convergence rate

F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, L. Schenato (2012)
The convergence rate of Newton-Raphson consensus optimization for quadratic cost functions
IEEE Conference on Decision and Control (CDC'12)

Newton-Raphson consensus for distributed convex optimization

Luca Schenato

Department of Information Engineering - University of Padova, Italy
URL: http://automatica.dei.unipd.it/people/schenato.html

June 15, 2013 – ECC Workshop on Distributed Optimization in Large Networks and its Applications

entirely written in LaTeX 2ε using Beamer and Tik Z

licensed under the Creative Commons BY-NC-SA 2.5 Italy License: