Newton-Raphson consensus for distributed convex optimization

Luca Schenato

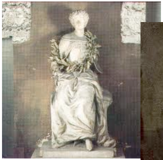
Department of Information Engineering - University of Padova
URL: http://automatica.dei.unipd.it/people/schenato.html

April 21, 2014
ISL Seminar
Stanford

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

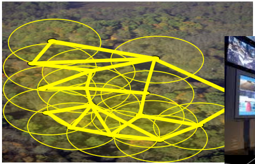
University of Padova

- Founded 1222: 2nd oldest university
- 60K students out of 200K citizens
- First Ph.d. woman in 1678: Elena Piscopia
- Alumni: Galileo, Copernicus, Riccati, Bernoulli
- Department of Information Engineering (EE&CS&BIOENG) 3K students

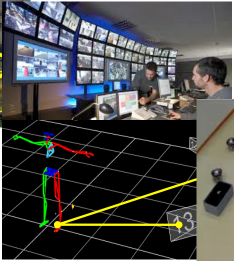


Target applications: the MAgIC Lab

Wireless Sensor Actuator Networks



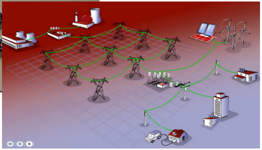
Smart Camera Networks



M.A.g.I.C.
Multi Agent Intelligent Control

Robotic Networks

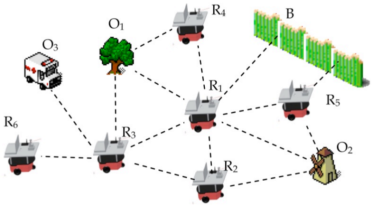
Smart Energy Grids



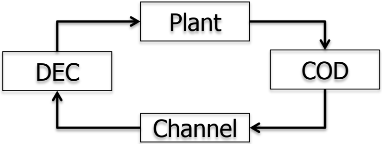
Networked Control Systems: physically distributed dynamical systems interconnected by a communication network

Research Lines

- Research line 1: multi-agent systems:
  - Consensus algorithms
  - Distributed estimation
  - Distributed optimization



- Research line 2: control subject to communication constraints:
  - Packet loss
  - Random delay
  - Sensor fusion



Contributors

Angelo Cenedese
Univ. of Padova

Gianluigi Pillonetto
Univ. of Padova

Damiano Varagnolo
Luleå University, Sweden

Filippo Zanella
Sellf, A-Pole

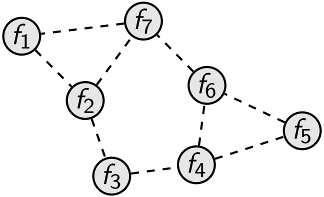
Presentation outline

- Motivations
- State-of-the-art
- Centralized Newton-Raphson: a quick overview
- Consensus-based Newton-Raphson
- Convergence properties (theory + simulations)
- Future directions

Presentation outline

- Motivations
- State-of-the-art
- Centralized Newton-Raphson: a quick overview
- Consensus-based Newton-Raphson
- Convergence properties (theory + simulations)
- Future directions

Cooperative Distributed Optimisation

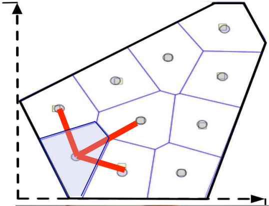


Assumption: neighbours cooperate to find minimizer of network cost:

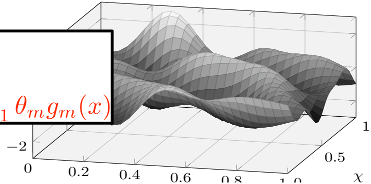
$$f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x), \quad x^* = \mathrm{argmin}_x f(x)$$

- Global estimation: $x \in \mathbb{R}^n$, each node wants $\hat{x}_i = x^*, \forall i = 1, \ldots, N$. Typically n independent of N: support vector machine, robotic map building.
- Local estimation: $f_i(x) = f_i(x_i, \{x_j\}_{j \in \mathcal{N}_i})$, each nodes just wants $\hat{x}_i = x_i^*$. Typically $n \geq N$: smart grid state estimation, robotic localization

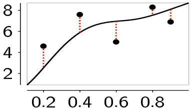
Global estimation: Robotic Map Building



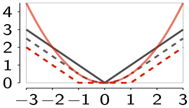
Parametric Model:
$g(x) = \sum_{m=1}^{M} \theta_m g_m(x)$



Noisy data:
$\{(x_i, y_i)\}_{i=1}^{N}$
$y_i = g(x_i) + v_i$



Cost function
$f(r) = |r|$



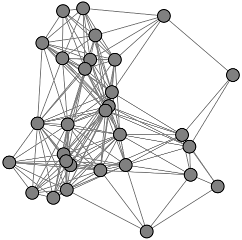
Global estimation: SVM Classification

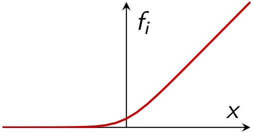
D. Varagnolo, F. Zanella, A. Cenedese, G. Pillonetto, L. Schenato. "Newton-Raphson Consensus for Distributed Convex Optimization". IEEE Transactions on Automatic Control (submitted)

$\chi \in \mathbb{R}^4$: 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:





Global estimation: Robust Regression

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

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

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

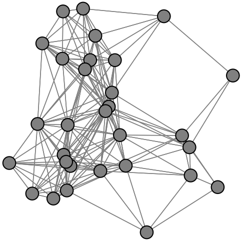
$(\mathbf{x}, x_0) \in \mathbb{R}^5$: parameters to be computed

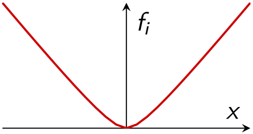
Connected graphs with 30 nodes

Local cost functions:

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

Housing Price Predictors:





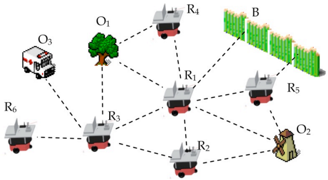
Local estimation: Localization

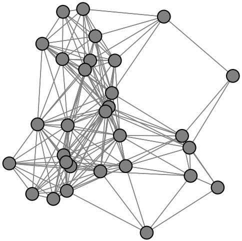
A. Carron, M. Todescato, R. Carli, L. Schenato. "An asynchronous consensus-based algorithm for estimation from noisy relative measurements". IEEE Transactions on Control of Network Systems (submitted)

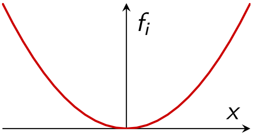
$x_i \in \mathbb{R}^2$: robot position
$x = (x_1, \ldots, x_N) \in \mathbb{R}^{2N}$
$z_{ij} \in \mathbb{R}^2$, vector noisy distance of node i and j, i.e. $z_{ij} = x_i - x_j +$ noise
Local cost functions:

$$f_i(x) := \sum_{j \in \mathcal{N}_i} \|x_i - x_j - z_{ij}\|^2.$$

Range-bearing measurements:





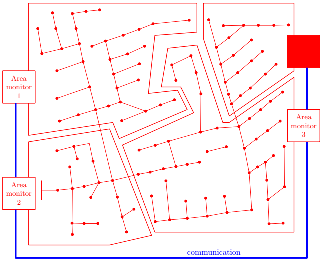


Local estimation: Smart Grid Estimation from noisy PMUs

S. Bolognani, R. Carli, M. Todescato, "State estimation in power distribution networks with poorly synchronized measurements", IEEE Transactions on Smart Grids (submitted)

$x_i \in \mathbb{C}$: node voltage
$x = (x_1, \ldots, x_N) \in \mathbb{C}^N$
$m_i^u \in \mathbb{C}$, noisy measured voltage at bus i
$m_i^c \in \mathbb{C}$, noisy measured current at bus i
L: weighted Laplacian of the network

Macro-area monitoring:



$$m = Hx + \eta, \quad R_\eta = \mathbb{E}[\eta\eta^T]$$

$$m = \begin{bmatrix} Re[m^u] \\ Im[m^u] \\ Re[m^c] \\ Im[m^c] \end{bmatrix}, H = \begin{bmatrix} I & 0 \\ 0 & I \\ Re[L] & -Im[L] \\ Im[L] & Re[L] \end{bmatrix}$$

Local cost functions:

$$\min_x \quad (m - Hx)^T R_\eta^{-1} (m - Hx) = \min_{x_{A_1},..,x_{A_s}} \sum_{h=1}^{s} J_h(x_{A_h}, \{x_{A_\ell}\}_{\ell \in \mathcal{N}_{A_h}})$$

J_h are quadratic functions

Ideal algorithm features

- Distributed: only local communication
- **Asynchronous**: no global communication nor updates synchronization
- Robust to (random) time-delays
- Robust to packet losses
- Broadcast communication: no ACK/NACK or full duplex
- Asymptotically optimal
- Anonymous
- Suitable for time-varying graphs

Presentation outline

- Motivations
- State-of-the-art
- Centralized Newton-Raphson: a quick overview
- Consensus-based Newton-Raphson
- convergence properties (theory + simulations)
- future directions

State-of-the-art

Distributed optimization methods: 3 main categories

- Primal decompositions methods
  (e.g. distributed subgradients)
- Dual decompositions methods
  (e.g. alternating direction method of multipliers)
- Heuristic methods
  (e.g. swarm optimization, genetic algorithms)

Primal decomposition methods (centralized)

Subgradient methods [Shor, 1985]

$$x_{k+1} = x_k - \alpha_k \cdot g\left(x_k\right)$$

with

- $g\left(x_k\right) :=$ subgradient of $f(\cdot)$ at x_k
- $\alpha_k :=$ stepsize

Convergence properties

- α_k typically needs to be diminishing for non-smooth f
- $g(\cdot)$ may be required to be bounded
- ...

Primal decomposition methods (distributed)

Distributed subgradient methods [Nedic Ozdaglar, 2009]

$$\begin{aligned} x_i(k)^+ &= x_i(k) - \alpha g_i(x_i(k)) \\ x_i(k+1) &= \textstyle\sum_{j=1}^{N} a_{ij}(k) x_j^+(k) \\ \hat{x}_i(k) &= \frac{1}{k} \textstyle\sum_{h=1}^{k} x_i(h) \end{aligned}$$

with

- $g_i(x_i(k)) :=$ local subgradient of local cost $f_i(\cdot)$ at $x_i(k)$
- α local stepsize
- $\sum_{j=1}^{N} a_{ij}(k) x_j(k) :=$ aver. consensus step on local estimates $x_j(k)$

Convergence properties [Nedic Ozdaglar, 2009]

E.g., for *bounded subgradients* and $\alpha_i(k) = \alpha$ then

$$\lim \inf_{k \to +\infty} f(\hat{x}_i(k)) \leq f^* + \delta$$

Dual decomposition methods (centralized)

Method of Multipliers [Bertsekas, 1982]

Primal reformulation:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \end{array}$$

$$\Updownarrow$$

$$\begin{array}{ll} \text{minimize} & f(x) + \frac{\rho}{2} \|Ax - b\|_2^2 \\ \text{subject to} & Ax = b \end{array}$$

yelds to dual Lagrangian

1. $x_{k+1} = \arg\min_x \left(f(x) + \lambda_k^T (Ax - b) + \frac{\rho}{2} \|Ax - b\|_2^2 \right)$
2. $\lambda_{k+1} = \lambda_k + \rho(Ax_k - b)$

Dual decomposition methods (distributed)

Alternating Direction Method of Multipliers [Bertsekas Tsitsiklis, 1997]

$$\begin{array}{ll} \text{minimize} & f_1(x) + f_2(z) \\ \text{subject to} & A_1 x + A_2 z - b = 0 \end{array}$$

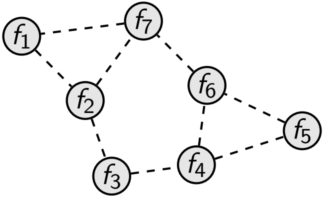
Augmented Lagrangian:

$$L_\rho(x, x_2, \lambda) := \quad f_1(x) + f_2(z) + \lambda^T (A_1 x + A_2 z - b) + \\ + \frac{\rho}{2} \|A_1 x + A_2 z - b\|_2^2$$

Algorithm

1. $\mathbf{x(k+1)} = \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, z(k), \lambda(k))$
2. $\mathbf{z(k+1)} = \arg\min_{\mathbf{x_2}} L_\rho(x(k+1), \mathbf{z}, \lambda(k))$
3. $\lambda(k+1) = \lambda(k) + \rho\,(A_1 x(k+1) + A_2 z(k+1) - b)$

ADMM for distributed optimization



Global estimation

$$\min_x \sum_{i=1}^{N} f_i(x) \iff \begin{array}{ll} \min_{\{x_i\}_{i=1}^N, \{z_{ij}\}_{(i,j)\in\mathcal{E}}} & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} & x_i = z_{ij}, \forall (i,j) \in \mathcal{E} \end{array}$$

z_{ij}: Bridge variables. Constraints written as $A_1 x + A_2 z - b = 0$.
Lagrangian:

$$L_\rho(\{x_i\}, \{\lambda_{ij}\}) := \sum_{i=1}^N f_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \lambda_{ij}^T (x_i - z_{ij}) + \frac{\rho}{2} \sum_{(i,j)\in\mathcal{E}} \|x_i - z_{ij}\|^2$$

Drawbacks of the considered algorithms

Primal based strategies

- may be slow (sublinear convergence $1/k$)
- may not converge to the minimizer

Dual based strategies

- may be computationally expensive
- require topological knowledge
- implementation to handle time-varying graphs, time delays, packet losses, etc. may require effort

Related recent work

- Primal: Gharesifard and Cortes 2014, Lu and Tang 2012, Wang and Elia 2010, Kia et al. 2014
- Dual: Boyd et al. 2010, Duchi et al. 2012, Zhu and Martinez, 2012, Johansson et al. 2009, Wei and Ozdaglar 2013

Presentation outline

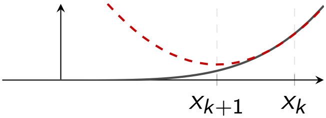
- Motivations
- State-of-the-art
- Centralized Newton-Raphson: a quick overview
- Consensus-based Newton-Raphson
- convergence properties (theory + simulations)
- future directions

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:

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

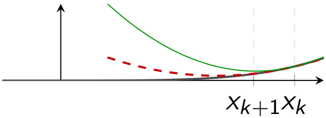
Jump to the minimum:

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

Gradient Descent: scalar case

Idea: approximate function $f(x)$ with a parabola with curvature equal to one

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



Match slope at x_k:

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

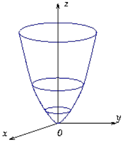
Jump to the minimum:

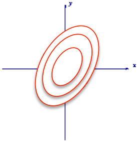
$$x_{k+1} = x_k - f'(x_k)$$

Newton-Raphson: multivariable case

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

$$\widehat{f}(x) = \frac{1}{2}(x-b)^T A(x-b) + c,$$
$$b \in \mathbb{R}^n, A > 0 \in \mathbb{R}^{n \times n}$$





Match slope and curvature at point x_k:

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

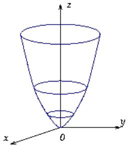
Jump to the minimum:

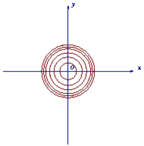
$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

Gradient Descent: multivariable

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

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





Match slope at x_k:

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

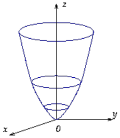
Jump to the minimum:

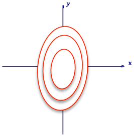
$$x_{k+1} = x_k - \nabla f(x_k)$$

Jacobi: multivariable

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

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





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:

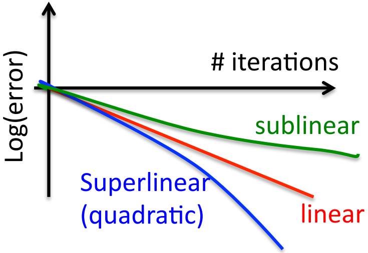
$$x_{k+1} = x_k - \left(\mathrm{diag}(\nabla^2 f(x_k))\right)^{-1} \nabla f(x_k)$$

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 $\nabla f(\boldsymbol{x}_{k-1})$

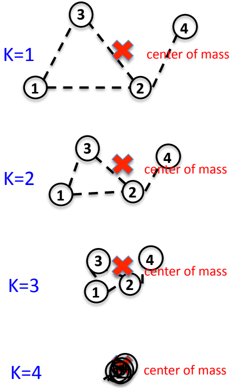
Presentation outline

- Motivations
- State-of-the-art
- Centralized Newton-Raphson: a quick overview
- Consensus-based Newton-Raphson
- convergence properties (theory + simulations)
- future directions

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

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

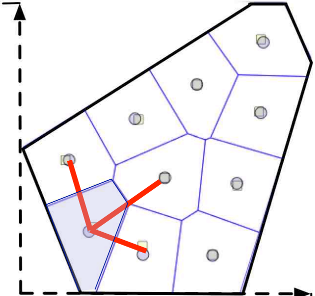
$$\mathbf{1}^T P = \mathbf{1}^T, P\mathbf{1} = \mathbf{1}, P \geq 0, P^N > 0$$

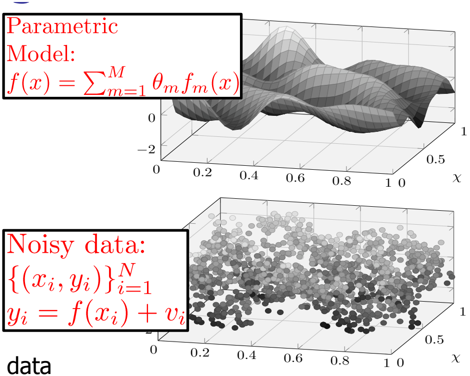
$$P = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{3}{4} \end{bmatrix}, \Longrightarrow$$

$\lim_{k\to\infty} x_i(k) = \frac{1}{N}\sum_{i=1}^N x_i(0), \ \forall i$
exponentially fast rate$=\mathrm{esr}(P)$

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

Map-building in robotic networks





- Issues:
  - Each robot collects local data
  - Local communication with robot
  - Patrolled area dynamically change

Map building as distributed least squares

Estimate

$$f(x) = \sum_{m=1}^{M} \theta_m f_m(x)$$

with unknown parameters $\theta_1, \dots, \theta_M$ from noisy measurements

$$y_i = \sum_{m=1}^{M} \theta_m f_m(x_i) + v_i, \quad i = 1, \dots, N$$

By stacking all measurements

F_i^T

$$\begin{bmatrix} y(x_1) \\ y(x_2) \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1(x_1) & \dots & f_M(x_1) \\ \vdots & \vdots & \vdots \\ f_1(x_N) & \dots & f_M(x_N) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_M \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix}$$

or equivalently:

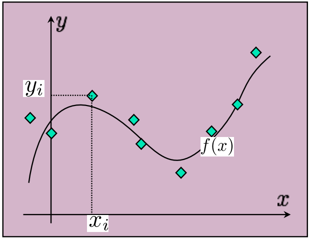
$$y = F\theta + v$$

Goal:

$$\hat{\theta} = \text{argmin}_\theta \sum_{i=1}^{N} v_i^2 = \text{argmin}_\theta ||F\theta - b||^2 = (F^T F)^{-1} F^T y$$

can be written as

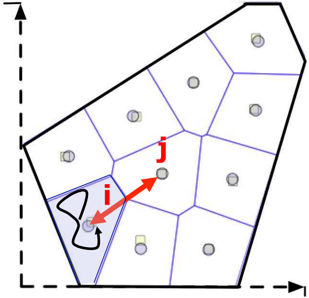
$$\hat{\theta} = (\sum_{i=1}^{N} F_i F_i^T)^{-1} (\sum_{i=1}^{N} F_i y_i) = (\frac{1}{N} \sum_{i=1}^{N} F_i F_i^T)^{-1} (\frac{1}{N} \sum_{i=1}^{N} F_i y_i)$$



Least-squares as ratio of two averages of local quantities

(Xiao,Boyd,Lall, IPSN05), (Bolognani,Del Favero, Schenato, Varagnolo JRNC10)

Consensus based map-building



Strategy for each robot i:

1) Initialize statistics:

$$Z_0^i = 0 \in R^{M\times M}$$
$$z_0^i = 0 \in R^{M}$$

2) Collect data and build local statistics:

$$Z_{t+1}^i = Z_i^t + F_t^i {F_t^i}^T$$
$$z_{t+1}^i = z_i^t + F_t^i y_t^i$$

3) Choose neighbor j and do gossip consensus:

$$Z_{t+1}^j = Z_{t+1}^i = \frac{1}{2} Z_t^i + \frac{1}{2} Z_t^j$$
$$z_{t+1}^j = z_{t+1}^i = \frac{1}{2} z_i^t + \frac{1}{2} z_j^t$$

4) Estimate map:

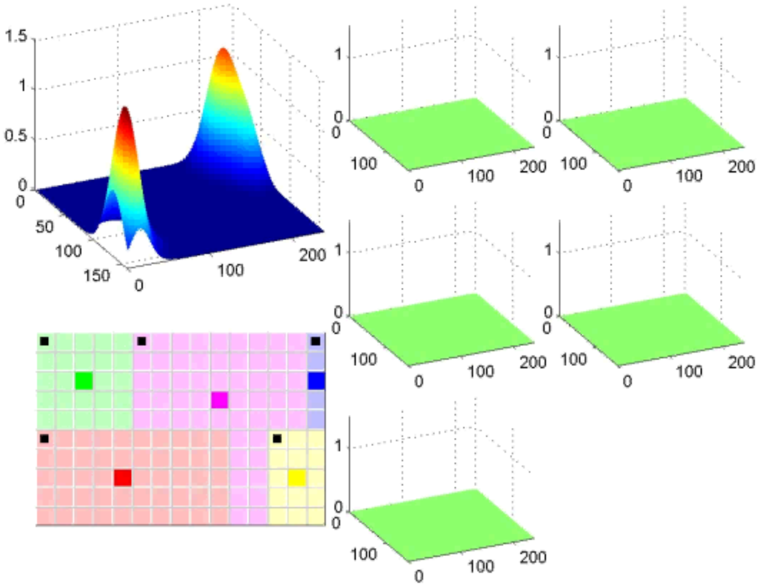
$$\hat{\theta}_t^i = (Z_t^i)^{-1} z_t^i$$

5) Repeat steps 2,3,4 (non necessarely in oder)

$$F_t^i := \begin{bmatrix} f_1(x_i(t)) \\ f_2(x_i(t)) \\ \vdots \\ f_M(x_i(t)) \end{bmatrix}$$

- Pros:
  - Asynchronous
  - Communication graph can change
- Cons:
  - Exchange of $O(M^2)$ data
  - Parametric model ←→ curse of dimensionality

Simulation: coverage with adaptive map-building



How to deal with non-quadratic cost functions?

Estimate

$$f(x) = \sum_{m=1}^{M} \theta_m f_m(x)$$

with unknown parameters $\theta_1, \ldots, \theta_M$ from noisy measurements

$$y_i = \sum_{m=1}^{M} \theta_m f_m(x_i) + v_i, \quad i = 1, \ldots, N$$

F_i^T

By stacking all measurements

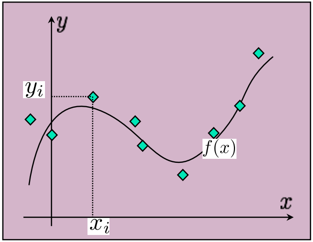
$$\begin{bmatrix} y(x_1) \\ y(x_2) \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1(x_1) & \ldots & f_M(x_1) \\ \vdots & \vdots & \vdots \\ f_1(x_N) & \ldots & f_M(x_N) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_M \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix}$$

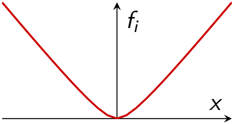
or equivalently:

$$y = F\theta + v$$

Goal:

$$\hat{\theta} = \text{argmin}_\theta \sum_{i=1}^{N} f(v_i) \neq \text{argmin}_\theta ||F\theta - b||^2 = (F^T F)^{-1} F^T y$$





Naive application of Consensus: the wrong way !

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

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

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)	$x_i^+ = x_i - \varepsilon y_i^+$	local descent step

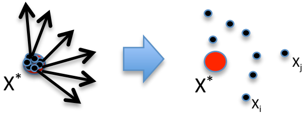
NOT WORKING !!

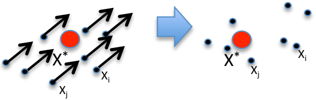
Naive application of Consensus: the wrong way ! (cont'd)

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

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

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

Match slope and curvature at point x_i:

$$\begin{aligned} f_i'(x_i) &= \widehat{f}_i'(x_i) = a_i(x_i - b_i) \\ f_i''(x_i) &= \widehat{f}_i''(x_i) = a_i \end{aligned} \quad \Rightarrow \quad \begin{aligned} a_i &= f_i''(x_i) \\ a_i b_i &= f_i''(x_i)x_i - f_i'(x_i) \end{aligned}$$

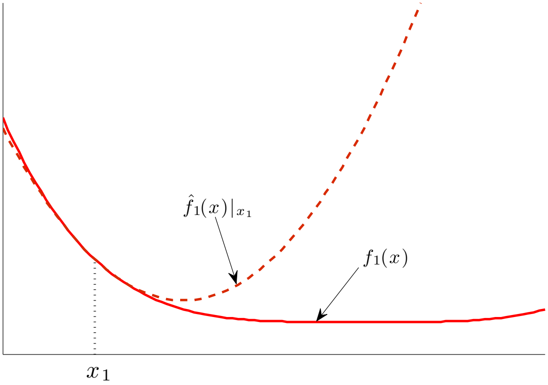
Jump to the minimum of $\widehat{f}(x)$:

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

Graphical interpretation

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

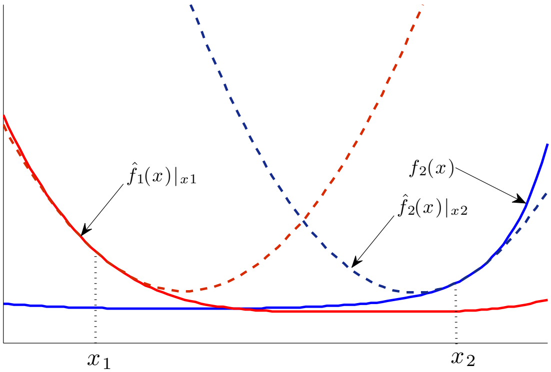
$$\Rightarrow \quad x^* = \frac{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)x_i - f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$$



Graphical interpretation

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

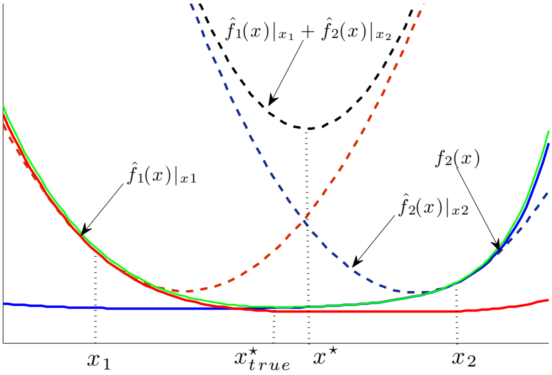
$$\Rightarrow \quad x^* = \frac{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)x_i - f_i'(x_i)}{\frac{1}{N}\sum_{i=1}^{N} f_i''(x_i)}$$



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 $x_i(k+1) = \dfrac{y_i(k+1)}{z_i(k+1)}$

$$\text{If } f_i(x_i) = \frac{1}{2}a_i(x_i - b_i)^2 \Longrightarrow \begin{cases} f_i''(x_i)x_i - f_i'(x_i) = a_i b_i \\ f_i''(x_i) = a_i \end{cases}, \forall x_i, \forall i$$

(Xiao,Boyd,Lall, IPSN05), (Bolognani,Del Favero, Schenato, Varagnolo JRNC10)

Towards a consensus-based Newton-Raphson

Algorithm

1. initialise local variables:
   - $y_i(0) := f_i''(x_i(0))x_i(0) - f_i'(x_i(0)) = a_i b_i$
   - $z_i(0) := f_i''(x_i(0)) = a_i$
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 $x_i(k+1) = \dfrac{y_i(k+1)}{z_i(k+1)}$

$$\text{If } f_i(x_i) = \tfrac{1}{2}a_i(x_i - b_i)^2 \Longrightarrow \begin{cases} f_i''(x_i)x_i - f_i'(x_i) = a_i b_i \\ f_i''(x_i) = a_i \end{cases}, \forall x_i, \forall i$$

(Xiao,Boyd,Lall, IPSN05), (Bolognani,Del Favero, Schenato, Varagnolo JRNC10)

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

Tracking of the current average

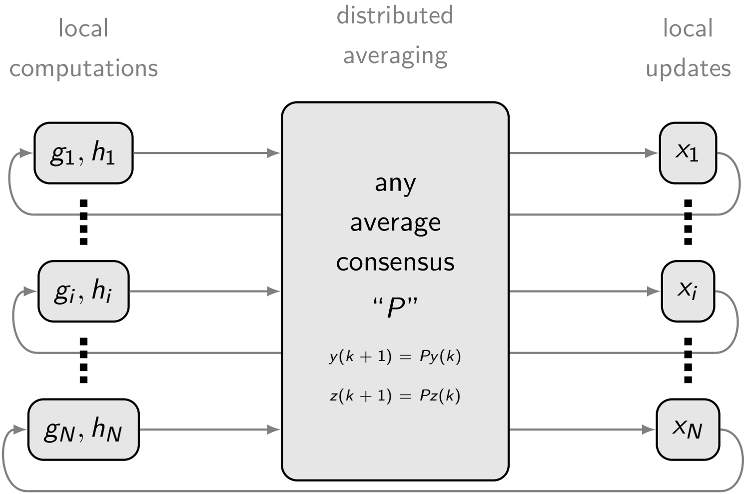
Plain average consensus would lead to integration, differently:

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

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



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

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

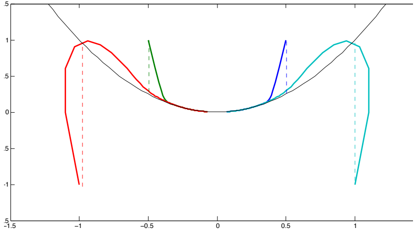
Presentation outline

- Motivations
- State-of-the-art
- Centralized Newton-Raphson: a quick overview
- Consensus-based Newton-Raphson
- Convergence properties (theory + simulations)
- future directions

Singular Perturbation Theory: an example

Coupled dynamics:

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



Idea: decouple time scales

- freeze slow dynamics, i.e. $x = constant$
- 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

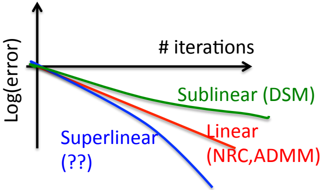
Assume $x_i = x_j = \bar{x}$:

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

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 (uniformly Lipschitz, strongly convex, bounded interconnection perturbations) $\Longrightarrow$ ***Global Exponential Stability*** for $0 < \varepsilon < \varepsilon_c$
- ***Convergence is "only" linear*** due to consensus: it needs time to pass information around



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***:

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

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)	$x_i^+ = x_i - \varepsilon y_i^+$	local descent step

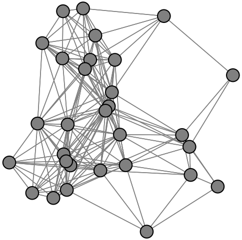
Simulations: SVM Classification with synchronous NR

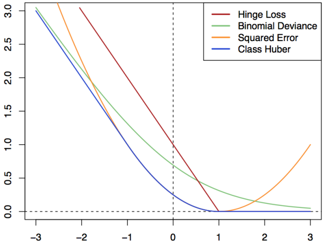
http://archive.ics.uci.edu/ml/datasets/Spambase

$\chi \in \mathbb{R}^4$: 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:

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

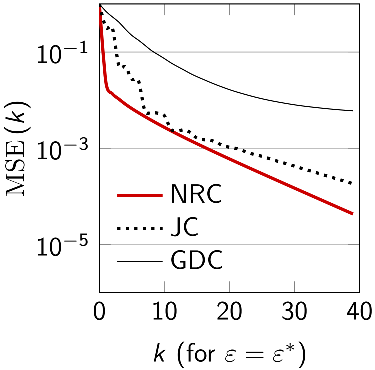
Spam Filters:





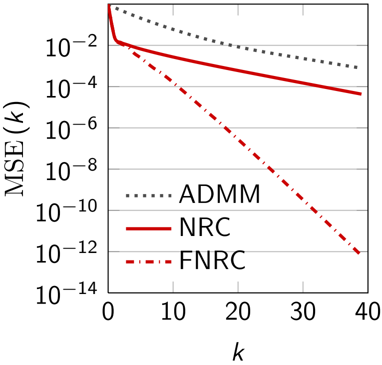
Simulations: SVM Classification with synchronous NR

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

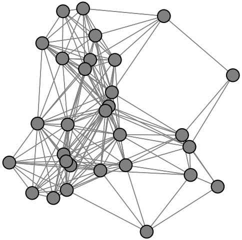
$(\mathbf{x}, x_0) \in \mathbb{R}^5$: parameters to be computed

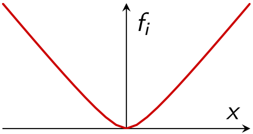
Connected graphs with 30 nodes

Local cost functions:

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

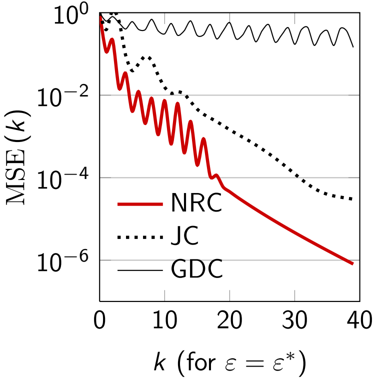
Housing Price Predictors:





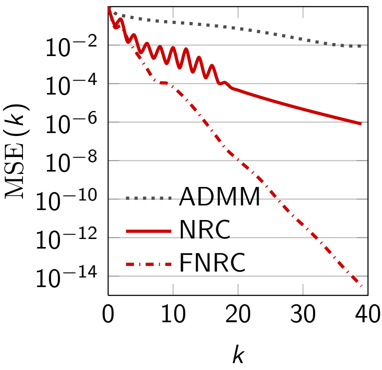
Simulations: Robust Regression with synchronous NR

Consensus-based algorithms:



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

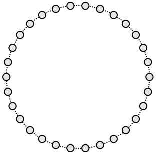
Comparison with other algorithms

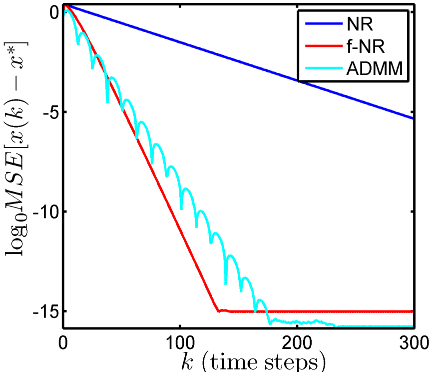


ADMM=Alternating Direction Multipliers Method
NRC=Newton-Raphson Consensus
FNRC= Newton-Raphson with Fast Consensus (diffusive)

Simulations: synthetic data

- circulant graph, $N = 30$
- $f_i(\boldsymbol{x}) = \exp\left(\left(\boldsymbol{x} - \boldsymbol{b}_i\right)^T A_i \left(\boldsymbol{x} - \boldsymbol{b}_i\right)\right)$





Convergence speed: ADMM vs NRC

Quadratic function with unit curvature:

$$f_i(x) = \frac{1}{2}(x - \theta_i)^2 \Longrightarrow x^* = \frac{1}{N}\sum_{i=1}^{N}\theta_i$$

Distributed computation via consensus (same as Newton-Raphson consensus):

$$\begin{aligned} \hat{x}(t+1) &= P\hat{x}(t), \quad P \sim \mathcal{G} \\ \hat{x}(0) &= \theta \end{aligned}$$

Rate of convergence:

$$\text{rate}: \ \ \rho_P = 1 - \sigma_P$$

where ρ_P is essential spectral gap and σ_P is spectral gap of P.

Convergence speed: ADMM vs NRC

Average consensus with memory (diffusive methods):

$$\begin{aligned} \hat{x}(t+1) &= \beta P \hat{x}(t) + (1-\beta)\hat{x}(t-1) \\ \hat{x}(0) &= \hat{x}(-1) = \theta \end{aligned}$$

If β chosen optimally:

$$\beta = \beta^* := \frac{2}{1+\sqrt{1-\rho_P^2}} \Longrightarrow \mathrm{rate} : \approx 1 - \sqrt{2\sigma_P}$$

Interpretation:

- Standard consensus: P feedback
- Consensus with memory: PD feedback and heavy-ball methods

Convergence speed: ADMM vs NRC

Equivalent optimization problem:

$$\min_x \sum_{i=1}^{N} \frac{1}{2}(x-\theta_i)^2 \Leftrightarrow \begin{array}{l} \min_{x_1,\ldots,x_N,z_1,\ldots,z_N} \sum_{i=1}^{N} \frac{1}{2}(x_i-\theta_i)^2 \\ s.t. \;\; x_i = z_j, \quad \forall i=1,\ldots,N, \forall j \in \mathcal{N}_i^+ \end{array}$$

ADMM approach

$$\mathcal{L}(x,z,\eta) = \sum_{i=1}^{N} f_i(x_i) + \sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i^+} \eta_{ij}(x_i - z_j) + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i^+} c_{ij}(x_i - z_j)^2$$

to get:

$$\begin{aligned} x_i(t+1) &= \frac{\theta_i + \sum_{j\in\mathcal{N}_i^+} c_{ij} z_j(t) - \sum_{j\in\mathcal{N}_i^+} \eta_{ij}(t)}{1 + \sum_{j\in\mathcal{N}_i^+} c_{ij}} \\ z_i(t+1) &= \frac{\sum_{j\in\mathcal{N}_i^+} c_{ji} x_j(t+1) + \sum_{j\in\mathcal{N}_i^+} \eta_{ji}(t)}{\sum_{j\in\mathcal{N}_i^+} c_{ji}} \\ \eta_{ij}(t+1) &= \eta_{ij}(t) + c_{ij}(x_i(t+1) - z_j(t+1)) \end{aligned}$$

Convergence speed: ADMM vs NRC

Previous dynamics can be written as:

$$C = \eta P \implies x(t+1) \quad = \quad Mx(t) - Nx(t-1)$$

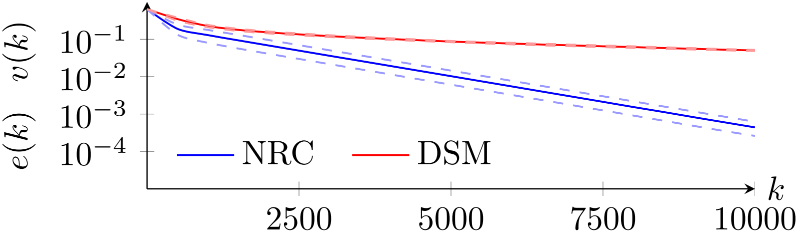
where

$$M = \frac{2\eta}{1+\eta} P^2 + \frac{1}{1+\eta} I, \quad N = \frac{\eta}{1+\eta} P^2$$

and η is a free parameter. If η chosen optimally :

$$\eta = \eta^* \implies \text{rate} : \approx 1 - \sqrt{2\sigma_P}$$

Asynchronous implementation



Presentation outline

- Motivations
- State-of-the-art
- Centralized Newton-Raphson: a quick overview
- Consensus-based Newton-Raphson
- convergence properties (theory + simulations)
- Future directions

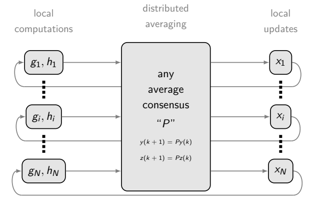
Comparisons

	DSM	ADMM	NRC
diff. functions	NO	NO	YES
rate (diff. functions)	sublinear	linear	linear
comm. complexity	$O(N)$	$O(N)$	$O(N^2)$
comp. complexity	small	medium-high	medium-high
glob. stable	yes	yes	no
asynchronous	yes	maybe	yes
time var. graph	yes	maybe	yes

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



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

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

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

Questions ?

THANK YOU

Publications on Newton-Raphson Convex Optimization (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 on Newton-Raphson Convex Optimization (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
URL: http://automatica.dei.unipd.it/people/schenato.html

April 21, 2014
ISL Seminar
Stanford

entirely written in LaTeX2ε using Beamer and Tik Z