# Distributed quasi-Newton method and its application to the optimal reactive power flow problem

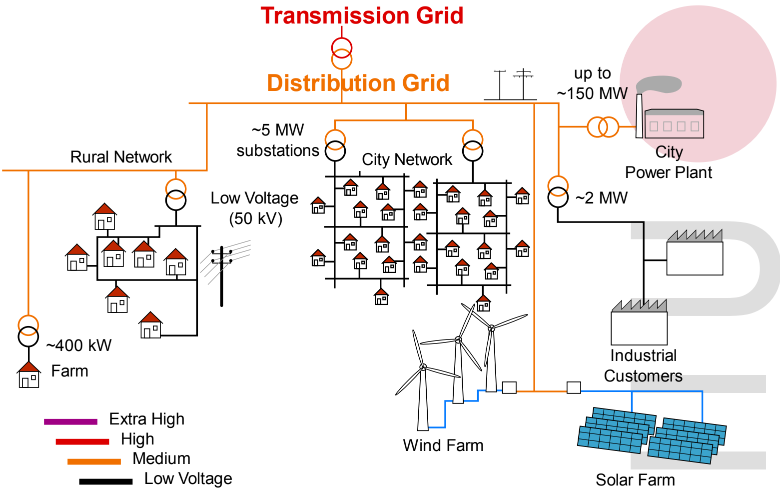
Saverio Bolognani

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

# Power distribution networks



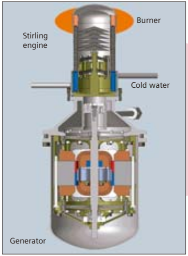


# Reactive power compensators



## Distributed Generation

The electronic interface of every micro generator can act as a compensator: micro-hydroelectric, combined heat and power, wind, solar, waste thermal generation.

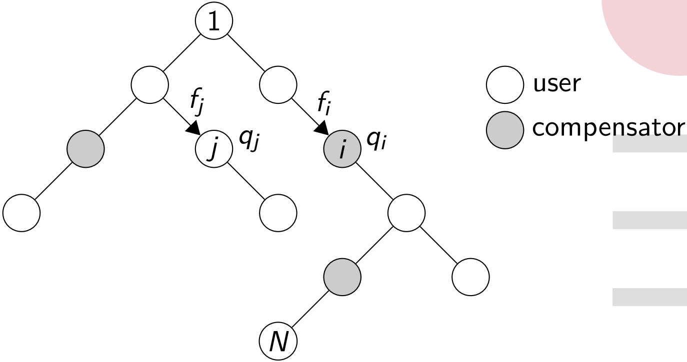



- the distribution network is partially unknown and unmonitored
- these agents can connect and disconnect
- because of the stochastic character of the energy sources and the large number of DG units, a centralized dispatchment is too complex
- security of the energy supply may be jeopardized if a great amount of data is handled online by a single control center.

# Simplified model



Consider a tree describing the low-mid voltage distribution network.




$q_i$ is the injected reactive power, $f_i$ is the reactive power flow.

# Optimization problem



The optimization problem of having minimal power losses on the network corresponds to having minimal reactive power flows

$$\min \; F(f_2, \dots, f_N) = \sum_{i=2}^{N} f_i^2 k_i .$$

subject to

- $\sum_{i \in \mathcal{C} \cup \mathcal{U}} q_i = 0$ 1 constraint - reactive power conservation
- $f_i = f_i(q_1, \dots, q_N)$ $N-1$ constraints - power flow equations

or in matricial form

$$\begin{aligned} \min \quad & \mathbf{f}^T \frac{\mathbf{K}}{2} \mathbf{f} \\ \text{subject to} \quad & \mathbf{f} = \mathbf{A}\mathbf{q} + \mathbf{B}\bar{\mathbf{q}} \\ & \mathbf{1}_{N_\mathcal{C}}^T \mathbf{q} + \mathbf{1}_{N_\mathcal{U}}^T \bar{\mathbf{q}} = 0 \end{aligned}$$

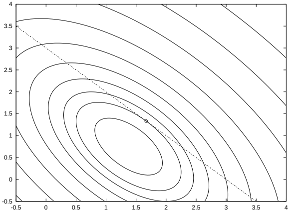
# Optimization problem

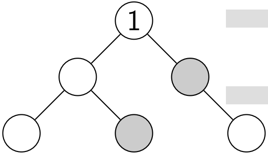
By eliminating the second constraint, one obtains the quadratic problem

$$\begin{aligned} \min \quad & J(\mathbf{q}) = \mathbf{q}^T \frac{\mathbf{M}}{2} \mathbf{q} + \mathbf{q}^T \mathbf{m} \\ \text{subject to} \quad & \mathbf{1}_{N_\mathcal{C}}^T \mathbf{q} = c. \end{aligned}$$

which has the closed form solution

$$\mathbf{q}^* = -\mathbf{M}^{-1} \left[ \mathbf{m} - \frac{\left( c + \mathbf{1}_{N_\mathcal{C}}^T \mathbf{M}^{-1} \mathbf{m} \right) \mathbf{1}_{N_\mathcal{C}}}{\mathbf{1}_{N_\mathcal{U}}^T \mathbf{M}^{-1} \mathbf{1}_{N_\mathcal{C}}} \right].$$



# Distributing the problem



Why this problem can still be interesting?

- unknown hessian $\mathbf{M}$ (depends on the topology)
- unknown constant $c$ (depends on the demands)
- unknown vector $\mathbf{m}$ (depends on the demands).

# Distributing the problem



Why this problem can still be interesting?

- unknown hessian $\mathbf{M}$ (depends on the topology)
- unknown constant $c$ (depends on the demands)
- unknown vector $\mathbf{m}$ (depends on the demands).

**Gradient driven optimization algorithms**

Most of the algorithms for the solution of convex optimization problems are driven by the gradient, and assume that the gradient is available.

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \boldsymbol{\Gamma}\mathbf{g}(\mathbf{q}(t))$$

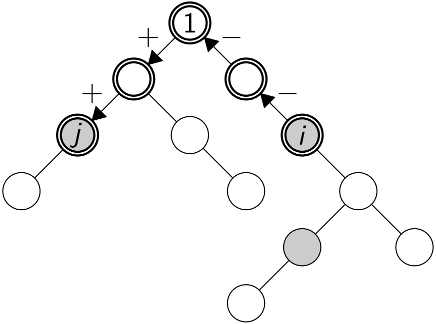
# Distributed gradient estimation



The gradient can be rewritten as

$$\mathbf{g} = \mathbf{A}^T\mathbf{K}\mathbf{A}\mathbf{q} + \mathbf{A}^T\mathbf{K}\mathbf{B}\bar{\mathbf{q}} = \mathbf{M}\mathbf{q} + \mathbf{m} = \mathbf{A}^T\mathbf{K}\mathbf{f} = \begin{bmatrix} \cdots \\ \sum_{i\in\mathcal{E}-\mathcal{P}_i} k_i f_i \\ \cdots \end{bmatrix},$$

$$\mathbf{g}_i - \mathbf{g}_j = \sum_{\ell\in\mathcal{P}_{ij}} \delta_\ell(i,j) k_\ell f_\ell \approx \mathbf{v}_i - \mathbf{v}_j.$$




The gradient can then be estimated element-wise and up to a constant from the steady state of the system:

$$\mathbf{g}_i = \mathbf{v}_i + \xi.$$

# Distributing the problem



$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{\Gamma}\mathbf{g}$$

If a communication constraint is enforced via a graph $\mathcal{G}$, then $\mathbf{\Gamma}$ cannot be a generic gain matrix.

**Sparse $\mathbf{\Gamma}$**

The simplest approach consists in enforcing sparsity of $\mathbf{\Gamma}$ so that it is consistent with $\mathcal{G}$.

However, a sparse $\mathbf{\Gamma}$ is unlikely to solve the problem efficiently, because

- the global constraint couples the agents' states
- non-separable cost functions couple the agents' optimal choice
- nobody knows the whole system and can design $\mathbf{\Gamma}$

# Example: Newton descent

If the network topology is fully known and communication constraints are relaxed, it is possible to implement a constrained Newton algorithm that guarantees 1-step convergence:

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \boldsymbol{\Gamma}\mathbf{g} = \mathbf{q}(t) - \mathbf{M}^{-1}\mathbf{g} + \frac{\mathbf{1}^T\mathbf{M}^{-1}\mathbf{g}}{\mathbf{1}^T\mathbf{M}^{-1}\mathbf{1}}\mathbf{M}^{-1}\mathbf{1}$$

$$\boldsymbol{\Gamma} = \mathbf{M}^{-1} - \frac{\mathbf{M}^{-1}\mathbf{1}\mathbf{1}^T\mathbf{M}^{-1}}{\mathbf{1}^T\mathbf{M}^{-1}\mathbf{1}}$$

# Example: Newton descent



If the network topology is fully known and communication constraints are relaxed, it is possible to implement a constrained Newton algorithm that guarantees 1-step convergence:

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{\Gamma}\mathbf{g} = \mathbf{q}(t) - \mathbf{M}^{-1}\mathbf{g} + \frac{\mathbf{1}^T\mathbf{M}^{-1}\mathbf{g}}{\mathbf{1}^T\mathbf{M}^{-1}\mathbf{1}}\mathbf{M}^{-1}\mathbf{1}$$

$$\mathbf{\Gamma} = \mathbf{M}^{-1} - \frac{\mathbf{M}^{-1}\mathbf{1}\mathbf{1}^T\mathbf{M}^{-1}}{\mathbf{1}^T\mathbf{M}^{-1}\mathbf{1}}$$

Or, if an approximation for $\mathbf{M}^{-1}$ is available, one can implement an approximate Newton step

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{H}\mathbf{g} + \frac{\mathbf{1}^T\mathbf{H}\mathbf{g}}{\mathbf{1}^T\mathbf{H}\mathbf{1}}\mathbf{H}\mathbf{1}$$

# Decomposition: descent + projection



In the approximate Newton descent step it is easy to recognize two parts

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{Hg} + \frac{\mathbf{1}^T\mathbf{Hg}}{\mathbf{1}^T\mathbf{H1}}\mathbf{H1}$$

# Decomposition: descent + projection



In the approximate Newton descent step it is easy to recognize two parts

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{H}\mathbf{g} + \frac{\mathbf{1}^T\mathbf{H}\mathbf{g}}{\mathbf{1}^T\mathbf{H}\mathbf{1}}\mathbf{H}\mathbf{1}$$

- an unconstrained descent step (requires knowledge of the system)

# Decomposition: descent + projection



In the approximate Newton descent step it is easy to recognize two parts

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{Hg} + \frac{\mathbf{1}^T\mathbf{Hg}}{\mathbf{1}^T\mathbf{H1}}\mathbf{H1}$$

- an unconstrained descent step (requires knowledge of the system)
- a projection step (requires knowledge of the others' choice)

# Sparse approximations + consensus



$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{Hg} + \frac{\mathbf{1}^T\mathbf{Hg}}{\mathbf{1}^T\mathbf{H1}}\mathbf{H1}$$

# Sparse approximations + consensus



$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{H}\mathbf{g} + \frac{\mathbf{1}^T\mathbf{H}\mathbf{g}}{\mathbf{1}^T\mathbf{H}\mathbf{1}}\mathbf{H}\mathbf{1}$$

## Sparse **H**

By choosing a sparse approximation **H** for $\mathbf{M}^{-1}$, the computation of **Hg** and **H1** depends only on neighbors' data.

# Sparse approximations + consensus



$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{Hg} + \frac{\mathbf{1}^T\mathbf{Hg}}{\mathbf{1}^T\mathbf{H1}}\mathbf{H1}$$

## Sparse $\mathbf{H}$

By choosing a sparse approximation $\mathbf{H}$ for $\mathbf{M}^{-1}$, the computation of $\mathbf{Hg}$ and $\mathbf{H1}$ depends only on neighbors' data.

## Consensus algorithm

By running average consensus algorithms on the vectors $[\mathbf{H}_i\mathbf{g}\ \mathbf{H}_i\mathbf{1}]^T$, nodes agree on the projection step.

# Sparse approximations + consensus



$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{Hg} + \frac{\mathbf{1}^T\mathbf{Hg}}{\mathbf{1}^T\mathbf{H1}}\mathbf{H1}$$

## Sparse H

By choosing a sparse approximation $\mathbf{H}$ for $\mathbf{M}^{-1}$, the computation of $\mathbf{Hg}$ and $\mathbf{H1}$ depends only on neighbors' data.

## Consensus algorithm

By running average consensus algorithms on the vectors $[\mathbf{H}_i\mathbf{g}\ \mathbf{H}_i\mathbf{1}]^T$, nodes agree on the projection step.

This approach enables a whole class of methods in the form

$$\mathbf{q}_i(t+1) = \mathbf{q}_i(t) - \gamma_i(\mathbf{q}_j, \mathbf{g}_j(\mathbf{q}), j \in \mathcal{N}_i; \eta_i, \mathbf{x})$$

# Quasi-Newton method



## Quasi-Newton methods

In these methods an estimate of the hessian's inverse is updated at every step so that

- it satisfies the secant condition $\mathbf{H}(t+1)\Delta\mathbf{g}(t) = \Delta\mathbf{q}(t)$ (where $\mathbf{H}$ is the estimate of the inverse of the hessian, and $\mathbf{d}$ is the projection of the gradient of the constraint)
- it minimizes $\|\mathbf{H}(t+1) - \mathbf{H}(t)\|$.

# Quasi-Newton method



## Quasi-Newton methods

In these methods an estimate of the hessian's inverse is updated at every step so that

- it satisfies the secant condition $\mathbf{H}(t+1)\Delta\mathbf{g}(t) = \Delta\mathbf{q}(t)$ (where $\mathbf{H}$ is the estimate of the inverse of the hessian, and $\mathbf{d}$ is the projection of the gradient of the constraint)
- it minimizes $\|\mathbf{H}(t+1) - \mathbf{H}(t)\|$.

A quasi-Newton method (Broyden's method) can be applied to our constrained optimization problem.

# Quasi-Newton method



$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{G}\mathbf{d}(t)$$

$$\mathbf{G}(t+1) = \mathbf{G}(t) + \frac{[\Delta\mathbf{q} - \mathbf{G}\Delta\mathbf{d}]\Delta\mathbf{d}^T}{\Delta\mathbf{d}^T\Delta\mathbf{d}}$$

where $\mathbf{d} = \Omega\mathbf{g}$.

# Quasi-Newton method



$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{G}\mathbf{d}(t)$$

$$\mathbf{G}(t+1) = \mathbf{G}(t) + \frac{[\Delta\mathbf{q} - \mathbf{G}\Delta\mathbf{d}]\Delta\mathbf{d}^T}{\Delta\mathbf{d}^T\Delta\mathbf{d}}$$

where $\mathbf{d} = \Omega\mathbf{g}$.
Update equation for the single node:

$$\mathbf{q}_i(t+1) = \mathbf{q}_i(t) - \mathbf{G}_i\mathbf{d}(t)$$

$$\mathbf{G}_i(t+1) = \mathbf{G}_i(t) + \frac{[\Delta\mathbf{q}_i - \mathbf{G}_i\Delta\mathbf{d}]\Delta\mathbf{d}^T}{\Delta\mathbf{d}^T\Delta\mathbf{d}}.$$

# Quasi-Newton method



$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{G}\mathbf{d}(t)$$

$$\mathbf{G}(t+1) = \mathbf{G}(t) + \frac{[\Delta\mathbf{q} - \mathbf{G}\Delta\mathbf{d}]\Delta\mathbf{d}^T}{\Delta\mathbf{d}^T\Delta\mathbf{d}}$$

where $\mathbf{d} = \Omega\mathbf{g}$.
Update equation for the single node:

$$\mathbf{q}_i(t+1) = \mathbf{q}_i(t) - \mathbf{G}_i\mathbf{d}(t)$$

$$\mathbf{G}_i(t+1) = \mathbf{G}_i(t) + \frac{[\Delta\mathbf{q}_i - \mathbf{G}_i\Delta\mathbf{d}]\Delta\mathbf{d}^T}{\Delta\mathbf{d}^T\Delta\mathbf{d}}.$$

**Finite time convergence**

We proved that this method converges in at most $2N$ steps.

# Distributed quasi-newton method



## Communication constraints

Suppose that communication constraints are now enforced: the update equation must keep the estimate **H** sparse.

$$\mathbf{H}(t+1) = \mathbf{H}(t) + \mathcal{P}_{\mathcal{E}}\left[\mathbf{D}^{+}(\Delta\mathbf{q} - \mathbf{H}\Delta\mathbf{g})\Delta\mathbf{g}^{T}\right],$$

where

$$(\mathcal{P}_{\mathcal{E}}(\mathbf{A}))_{ij} = \begin{cases} \mathbf{A}_{ij} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\left(\mathbf{D}^{+}\right)_{ij} = \begin{cases} 1/\mathbf{g}^{(i)\,T}\mathbf{g}^{(i)} & \text{if } \mathbf{g}^{(i)} \neq 0 \\ 0 & \text{if } \mathbf{g}^{(i)} = 0. \end{cases}$$

# Distributed quasi-newton method



Let's complete the algorithm by introducing the projection step:

$$\mathbf{q}(t+1) = \mathbf{q}(t) - \mathbf{H}(t)\mathbf{g}(t) + \frac{\mathbf{1}^T\mathbf{H}(t)\mathbf{g}(t)}{\mathbf{1}^T\mathbf{H}(t)\mathbf{1}}\mathbf{H}\mathbf{1}$$

obtaining

## Distributed quasi-Newton method

$$\mathbf{q}_i(t+1) = \mathbf{q}_i(t) - \mathbf{H}_i(t)^T\mathbf{g}^{(i)}(t) + x\mathbf{H}_i(t)^T\mathbf{1}^{(i)}$$

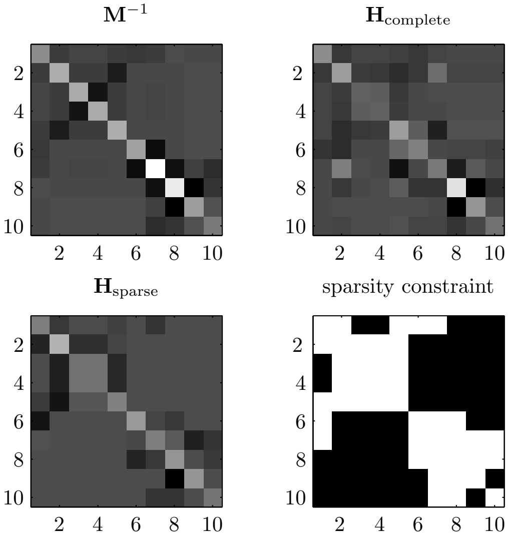
$$\mathbf{H}_i(t+1) = \mathbf{H}_i(t) + \left[\Delta\mathbf{q}_i - \mathbf{H}_i(t)^T\Delta\mathbf{g}^{(i)}(t)\right]\frac{\mathbf{g}^{(i)}(t)}{\mathbf{g}^{(i)}(t)^T\mathbf{g}^{(i)}(t)}$$

where $x = \bar{z}_1/\bar{z}_2$, $z$ being the result of consensus algorithm on

$$z^{(i)}(0) = \begin{bmatrix}\mathbf{H}_i(t)^T\mathbf{g}^{(i)}(t)\\ \mathbf{H}_i(t)^T\mathbf{1}^{(i)}\end{bmatrix}.$$

# Numerical simulations

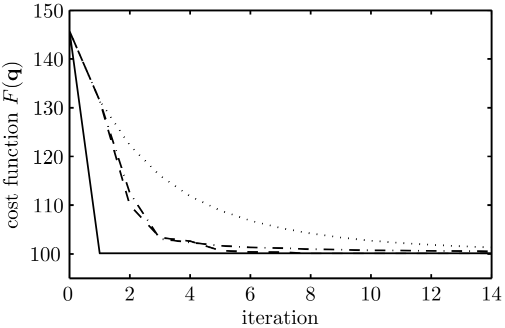




DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

# Numerical simulations






Newton (solid), quasi-Newton (dashed), distribute quasi-Newton (dot-dashed), steepest descent (dotted).

Bolognani, S., and Zampieri, S. (2010).
Distributed Quasi-Netwon Method and its Application to the Optimal Reactive Power Flow Problem.
In *Proceedings of NECSYS 2010*, Annecy, France.

# Thanks!

**Saverio Bolognani**
Department of Information Engineering
University of Padova (Italy)

saverio.bolognani@dei.unipd.it
`http://www.dei.unipd.it/~sbologna`