Solving a Class of Optimal Power Flow Problems in Tree Networks

Amir Beck[1] Yuval Beck[2] Yoash Levron[3] Alex Shtof[3]
Luba Tetruashvili[3]

[1]Tel Aviv University

[2]Holon Institute of Technology

[3]Technion - Israel Institute of Technology

July, 2018

Power networks and problems

Power network (AC)

"A mathematical model describing problems on a large electrical circuit which conveys power from generators to consumers."

Ingredients

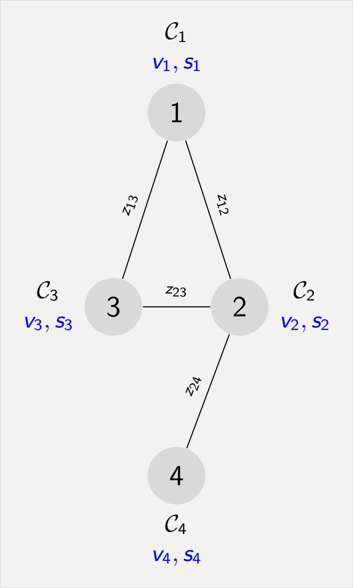
- Model data - an augmented graph
- Decision variables
- Constraints

Two fundamental problems

Feasibility Power Flow

Optimization Optimal Power Flow

Power network (AC)



Model data: $P = (G, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$.

- Topology graph $G = (V, E)$.
- Edge impedances $0 \neq z_{ij} \in \mathbb{C}$.
- Nodal constraints $\mathcal{C}_i \in \mathbb{R} \times \mathbb{C}$.

Decision variables:

Voltage - $\mathbf{v} \in \mathbb{C}^n$, Power - $\mathbf{s} \in \mathbb{C}^n$.

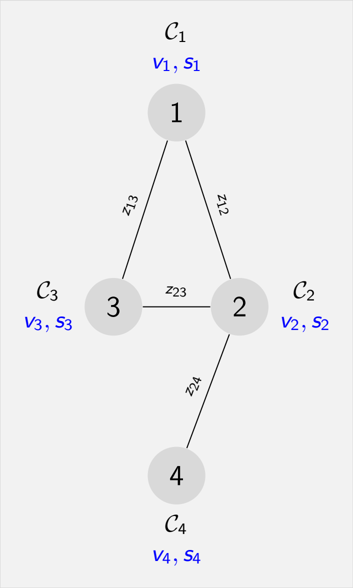
Constraints:

$$s_i = \sum_{j \in N(i)} v_i \frac{v_i^* - v_j^*}{z_{ij}^*} \qquad i \in V$$

$$\arg(v_1) = 0$$

$$(|v_i|, s_i) \in \mathcal{C}_i \qquad i \in V$$

Power network (AC)



Model data: $P = (G, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$.

- Topology graph $G = (V, E)$.
- Edge impedances $0 \neq z_{ij} \in \mathbb{C}$.
- Nodal constraints $\mathcal{C}_i \in \mathbb{R} \times \mathbb{C}$.

Decision variables:

Voltage - $\mathbf{v} \in \mathbb{C}^n$, Power - $\mathbf{s} \in \mathbb{C}^n$.

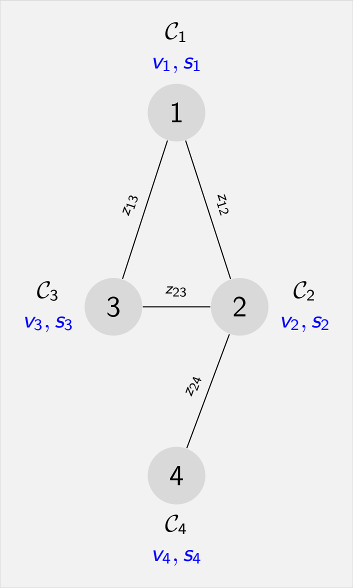
Constraints:

$$s_i = \sum_{j \in N(i)} v_i \frac{v_i^* - v_j^*}{z_{ij}^*} \qquad i \in V$$

$$\arg(v_1) = 0$$

$$(|v_i|, s_i) \in \mathcal{C}_i \qquad i \in V$$

Power network (AC)



Model data: $P = (G, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$.

- Topology graph $G = (V, E)$.
- Edge impedances $0 \neq z_{ij} \in \mathbb{C}$.
- Nodal constraints $\mathcal{C}_i \in \mathbb{R} \times \mathbb{C}$.

Decision variables:

Voltage - $\mathbf{v} \in \mathbb{C}^n$, Power - $\mathbf{s} \in \mathbb{C}^n$.

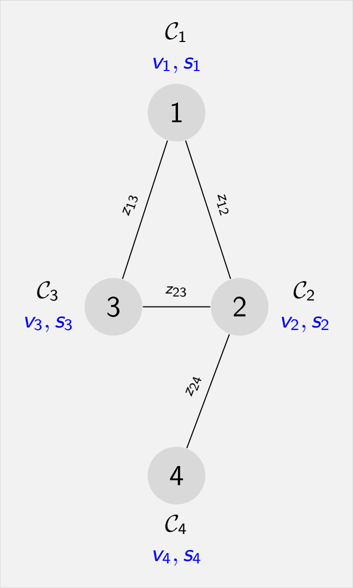
Constraints:

$$s_i = \sum_{j \in N(i)} v_i \frac{v_i^* - v_j^*}{z_{ij}^*} \qquad i \in V$$

$$\arg(v_1) = 0$$

$$(|v_i|, s_i) \in \mathcal{C}_i \qquad i \in V$$

Power network (AC)



Model data: $P = (G, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$.

- Topology graph $G = (V, E)$.
- Edge impedances $0 \neq z_{ij} \in \mathbb{C}$.
- Nodal constraints $\mathcal{C}_i \in \mathbb{R} \times \mathbb{C}$.

Decision variables:

Voltage - $\mathbf{v} \in \mathbb{C}^n$, Power - $\mathbf{s} \in \mathbb{C}^n$.

Constraints:

$$
\begin{aligned}
\mathcal{F}\mathcal{F}(P) = \{(\mathbf{v}, \mathbf{s}) :& \\
& s_i = \sum_{j \in N(i)} v_i \frac{v_i^* - v_j^*}{z_{ij}^*} && i \in V \\
& \arg(v_1) = 0 \\
& (|v_i|, s_i) \in \mathcal{C}_i && i \in V \\
\}.&
\end{aligned}
$$

Common constraints

Common constraints

PQ constraint - known power demand

$$\mathcal{C}_i = [\underline{u}_i, \overline{u}_i] \times \{\hat{s}_i\}.$$

Examples:

$$\mathcal{C}_i = [0.9, 1.1] \times \{-5 - 1\imath\}.$$

$$\mathcal{C}_i = [0.9, +\infty] \times \{-5 - 1\imath\}.$$

Common constraints

PQ constraint - known power demand

$$\mathcal{C}_i = [\underline{u}_i, \overline{u}_i] \times \{\hat{s}_i\}.$$

Examples:

$$\mathcal{C}_i = [0.9, 1.1] \times \{-5 - 1\imath\}.$$

$$\mathcal{C}_i = [0.9, +\infty] \times \{-5 - 1\imath\}.$$

PV constraint - known voltage and power generation

$$\mathcal{C}_i = \{\hat{u}_i\} \times \{\hat{p}_i + \imath[\underline{q}_i, \overline{q}_i]\}.$$

Example:

$$\mathcal{C}_i = [1.05] \times \{10 + \imath[-5, +\infty]\}.$$

The optimal power flow (OPF) problem

Input: A power network $P = (G, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$, and a cost function $f : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{R}$
Objective: Solve the optimization problem

$$\begin{aligned} \min_{\mathbf{v},\mathbf{s}} \quad & f(\mathbf{v}, \mathbf{s}) \\ \text{s.t.} \quad & (\mathbf{v}, \mathbf{s}) \in \mathcal{FF}(P) \end{aligned}$$

The optimal power flow (OPF) problem

Input: A power network $P = (G, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$, and a cost function $f : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{R}$

Objective: Solve the optimization problem

$$\begin{aligned} \min_{\mathbf{v},\mathbf{s}} \quad & f(\mathbf{v}, \mathbf{s}) \\ \text{s.t.} \quad & (\mathbf{v}, \mathbf{s}) \in \mathcal{FF}(P) \end{aligned}$$

Cost function examples:

$$f(\mathbf{v}, \mathbf{s}) = \sum_{i \in Gen} \mathrm{re}(s_i), \quad f(\mathbf{v}, \mathbf{s}) = \sum_{i \in PQ} \left| |v_i| - 0.5 \cdot (\underline{u}_i + \overline{u}_i) \right|$$

The optimal power flow (OPF) problem

Input: A power network $P = (G, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$, and a cost function $f : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{R}$
Objective: Solve the optimization problem

$$\begin{aligned} \min_{\mathbf{v},\mathbf{s}} \quad & f(\mathbf{v}, \mathbf{s}) \\ \text{s.t.} \quad & (\mathbf{v}, \mathbf{s}) \in \mathcal{FF}(P) \end{aligned}$$

Cost function examples:

$$f(\mathbf{v}, \mathbf{s}) = \sum_{i \in Gen} \mathrm{re}(s_i), \quad f(\mathbf{v}, \mathbf{s}) = \sum_{i \in PQ} \left| |v_i| - 0.5 \cdot (\underline{u}_i + \overline{u}_i) \right|$$

A common setup

- Consumers have PQ constraints
- Generators have box constraints.

A non-convex and potentially non-smooth problem.

The bad news

Karsten Lehmann, Alban Grastien, Pascal Van Hentenryck.
AC-Feasibility on Tree Networks is NP-Hard
IEEE Transactions on Power Systems, 2015

Approaches

- Heuristics[1]
  - Interior point methods
  - Convex relaxations
- Global solution methods (e.g. branch and bound)
- Specialized solution[2] methods for sub-classes
  - Tight convex relaxations
  - Our method

[1] Known to work well in in practice, but no theory explaining why.
[2] A method equipped with a theory explaining why it works

Approaches

- Heuristics[1]
  - Interior point methods
  - Convex relaxations
- Global solution methods (e.g. branch and bound)
- Specialized solution[2] methods for sub-classes
  - Tight convex relaxations
  - Our method

Stephen Frank, Ingrida Steponavice, Steffen Rebennack
Optimal power flow: a bibliographic survey I
Energy Systems, 2012

Daniel Bienstock
Progress on solving power flow problems
Optima, 2013

[1] Known to work well in in practice, but no theory explaining why.
[2] A method equipped with a theory explaining why it works

Our setup

Assumptions

Network assumptions

- The topology is a tree $T = (V, E)$ with $\mathrm{root}(T) = 1$ and $|V| > 1$.
- Leaf constraints: $\mathcal{C}_i$ is a compact PQ or PV constraint.
- Root constraint: $\mathcal{C}_1 = [\underline{u}_r, \overline{u}_r] \times \mathbb{C}$.
- Remaining constraints: $\mathcal{C}_i$ is a PQ constraint.
- Non-zero voltages: $(u, s) \in \mathcal{C}_i \implies u > 0$.

Assumptions

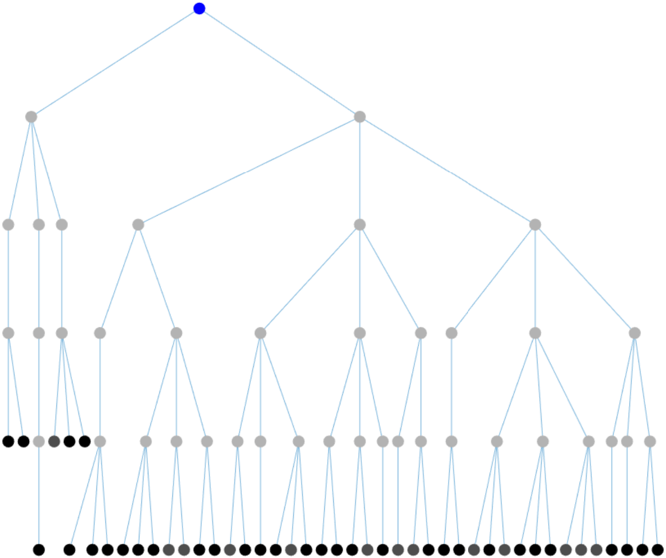
Network assumptions

- The topology is a tree $T = (V, E)$ with $\text{root}(T) = 1$ and $|V| > 1$.
- Leaf constraints: $\mathcal{C}_i$ is a compact PQ or PV constraint.
- Root constraint: $\mathcal{C}_1 = [\underline{u}_r, \overline{u}_r] \times \mathbb{C}$.
- Remaining constraints: $\mathcal{C}_i$ is a PQ constraint.
- Non-zero voltages: $(u, s) \in \mathcal{C}_i \implies u > 0$.

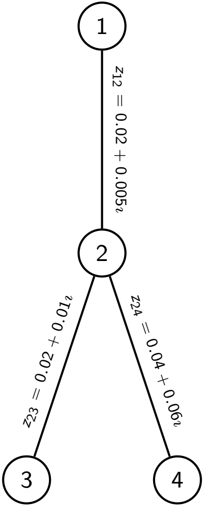
Problem assumptions

- The objective function f is continuous.

Visualization



Example



$$\begin{aligned} \mathcal{C}_1 &= [0.97, \infty] \times \mathbb{C} && \text{Root constraint} \\ \mathcal{C}_2 &= [0.9, 1.1] \times \{-0.2 - 0.1\imath\} && \text{PQ constraint} \\ \mathcal{C}_3 &= [0.9, 1.1] \times \{-0.4 - 0.3\imath\} && \text{PQ constraint} \\ \mathcal{C}_4 &= \{1\} \times (0.25 + \imath[-1, 1]) && \text{PV constraint} \end{aligned}$$

Motivation

Only one degree of freedom. So why the effort?

Motivation

- It is challenging - to the best of our knowledge, no known efficient solution.
- Useful as a computational step.

https://flic.kr/p/5jLLgR

The tree reduction / expansion method

The main result

- A decomposition of the feasible set into a finite union of parameterized curves:

$$\mathcal{F}\mathcal{F}(P) = \bigcup_{i=1}^{m} \text{image}(\gamma_i),$$

  where $\gamma_i : [0, 1] \to \mathbb{C}^n \times \mathbb{C}^n$ are continuous and piecewise smooth functions.
- An algorithm to compute a representation of $\{\gamma_i\}_{i=1}^{m}$.
- An efficient algorithm to compute $\gamma_i(t)$ given i and t.

Solving OPF

- Compute a representation of the curves: $\{\gamma_i\}_{i=1}^m$.
- Choose sampling density $N \in \mathbb{N}$.
- Grid search:

$$(\hat{\mathbf{v}}, \hat{\mathbf{s}}) \in \underset{(\mathbf{v},\mathbf{s})}{\operatorname{argmin}} \Big\{ f(\mathbf{v}, \mathbf{s}) : (\mathbf{v}, \mathbf{s}) = \gamma_i(j/(N-1)), \\ i = 1, \ldots, m,\ j = 0, \ldots, N-1 \Big\}$$

Observation: For any $(\mathbf{v}, \mathbf{s}) \in \mathcal{FF}(P)$, the vector $\mathbf{s}$ is redundant:

$$s_i = \sum_{j \in N(i)} v_i \frac{v_i^* - v_j^*}{z_{ij}^*}, \quad i = 1, \dots, n.$$

We work with $\mathcal{FF}_v(P) = \{\mathbf{v} : (\mathbf{v}, \mathbf{s}) \in \mathcal{FF}(P)\}$.

Observation: Compact PQ and PV constraints are line segments in $\mathbb{R} \times \mathbb{C}$.

Example: $[0.9, 1.1] \times \{-0.4 - 0.3\imath\}$ is the line segment between $(0.9, -0.4 - 0.3\imath)$ and $(1.1, -0.4 - 0.3\imath)$.

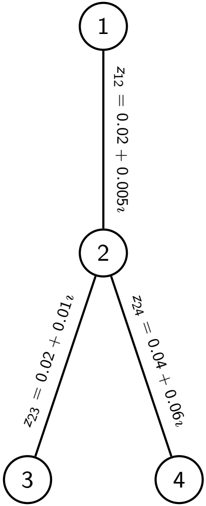
Observation: Compact PQ and PV constraints are line segments in $\mathbb{R} \times \mathbb{C}$.

Example: $[0.9, 1.1] \times \{-0.4 - 0.3\imath\}$ is the line segment between $(0.9, -0.4 - 0.3\imath)$ and $(1.1, -0.4 - 0.3\imath)$.

Observation: Line segments are curves.

Idea: Replace leaf constraints with parameterized curves - functions from $[0, 1]$ to $\mathbb{R} \times \mathbb{C}$.

Example



Input network

$$\begin{aligned}
\mathcal{C}_1 &= [0.97, \infty] \times \mathbb{C} \\
\mathcal{C}_2 &= [0.9, 1.1] \times \{-0.2 - 0.1\imath\} \\
\mathcal{C}_3 &= [0.9, 1.1] \times \{-0.4 - 0.3\imath\} \\
\mathcal{C}_4 &= \{1\} \times (0.25 + \imath[-1, 1])
\end{aligned}$$

Curved network

$$\begin{aligned}
\mathcal{C}_1 &= [0.97, \infty] \times \mathbb{C} \\
\mathcal{C}_2 &= [0.9, 1.1] \times \{-0.2 - 0.1\imath\} \\
\mathcal{C}_3 &= \mathrm{image}(\nu_3, \sigma_3) \\
&\quad \nu_3(t) = 0.9 + 0.2t, \quad \sigma_3(t) = -0.4 - 0.3\imath \\
\mathcal{C}_4 &= \mathrm{image}(\nu_4, \sigma_4) \\
&\quad \nu_4(t) = 1, \quad \sigma_4(t) = 0.25 + \imath(-1 + 2t)
\end{aligned}$$

Curved network

- The topology is a tree $T = (V, E)$ with $\mathrm{root}(T) = 1$.
- Leaf constraints: $\mathcal{C}_i = \mathrm{image}(\nu_i, \sigma_i)$ given continuous $\nu_i : [0, 1] \to \mathbb{R}$ and $\sigma_i : [0, 1] \to \mathbb{C}$.
- Root constraint:
  - If $|V| > 1$ then $\mathcal{C}_1 = [\underline{u}_r, \overline{u}_r] \times \mathbb{C}$.
  - If $|V| = 1$ then $\mathcal{C}_1 = [\underline{u}_r, \overline{u}_r] \times \{0\}$.
- Remaining constraints: $\mathcal{C}_i$ is a PQ constraint.
- Non-zero voltages: $(u, s) \in \mathcal{C}_i \implies u > 0$.

Curved network

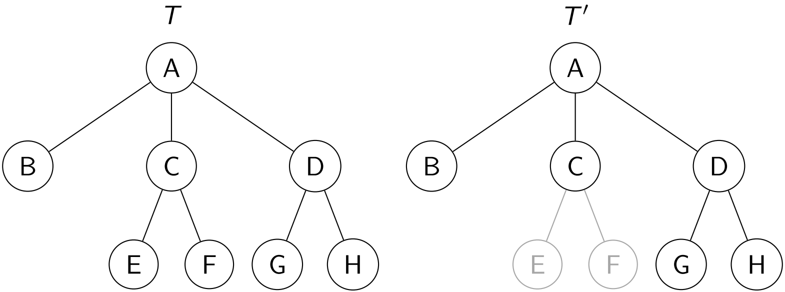
- The topology is a tree $T = (V, E)$ with $\mathrm{root}(T) = 1$.
- Leaf constraints: $\mathcal{C}_i = \mathrm{image}(\nu_i, \sigma_i)$ given continuous $\nu_i : [0, 1] \to \mathbb{R}$ and $\sigma_i : [0, 1] \to \mathbb{C}$.
- Root constraint:
  - If $|V| > 1$ then $\mathcal{C}_1 = [\underline{u}_r, \overline{u}_r] \times \mathbb{C}$.
  - If $|V| = 1$ then $\mathcal{C}_1 = [\underline{u}_r, \overline{u}_r] \times \{0\}$.
- Remaining constraints: $\mathcal{C}_i$ is a PQ constraint.
- Non-zero voltages: $(u, s) \in \mathcal{C}_i \implies u > 0$.

Next: find a relationship between $\mathcal{FF}_v(P)$ and $\mathcal{FF}_v(P')$ for a smaller network P'.

Tree reduction

- Reducible node - a non-leaf whose children are all leaves
- Tree reduction - the operation of removing all children of some reducible node.

Example

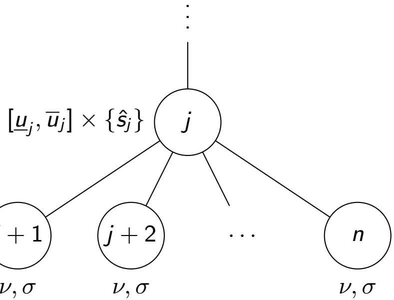


Tree-Reduction Theorem

Let $P = (T, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ be a curved network.

Tree-Reduction Theorem

Let $P = (T, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ be a curved network. Let j be a reducible node, let T' be the resulting reduction, and w.l.o.g its children are $\{j+1, \ldots, n\}$.



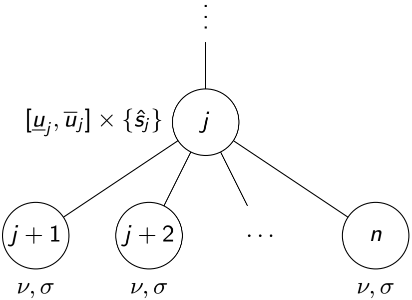
Tree-Reduction Theorem

Let $P = (T, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ be a curved network. Let j be a reducible node, let T' be the resulting reduction, and w.l.o.g its children are $\{j+1, \ldots, n\}$. Define

$$\tilde{\nu}_k(t) = |\nu_k(t) - z_{kj}(\sigma_k(t))^* / \nu_k(t)|, \quad k = j+1, \ldots, n$$

assume $\tilde{\nu}_k$ are invertible, and let

$$U_j = [\underline{u}_j, \overline{u}_j] \cap \text{image}(\tilde{\nu}_{j+1}) \cap \cdots \cap \text{image}(\tilde{\nu}_n).$$



Tree-Reduction Theorem

Let $P = (T, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ be a curved network. Let j be a reducible node, let T' be the resulting reduction, and w.l.o.g its children are $\{j+1, \ldots, n\}$. Define

$$\tilde{\nu}_k(t) = |\nu_k(t) - z_{kj}(\sigma_k(t))^* / \nu_k(t)|, \quad k = j+1, \ldots, n$$

assume $\tilde{\nu}_k$ are invertible, and let

$$U_j = [\underline{u}_j, \overline{u}_j] \cap \mathrm{image}(\tilde{\nu}_{j+1}) \cap \cdots \cap \mathrm{image}(\tilde{\nu}_n).$$

Then,

- If $U_j = \emptyset$ then $\mathcal{F}\mathcal{F}(P) = \emptyset$.

Tree-Reduction Theorem

Let $P = (T, \mathbf{z}, \mathcal{C}_1, \ldots, \mathcal{C}_n)$ be a curved network. Let j be a reducible node, let T' be the resulting reduction, and w.l.o.g its children are $\{j+1, \ldots, n\}$. Define

$$\tilde{\nu}_k(t) = |\nu_k(t) - z_{kj}(\sigma_k(t))^* / \nu_k(t)|, \quad k = j+1, \ldots, n$$

assume $\tilde{\nu}_k$ are invertible, and let

$$U_j = [\underline{u}_j, \overline{u}_j] \cap \mathrm{image}(\tilde{\nu}_{j+1}) \cap \cdots \cap \mathrm{image}(\tilde{\nu}_n).$$

Then,

- If $U_j = \emptyset$ then $\mathcal{FF}(P) = \emptyset$.
- Otherwise, the there exist functions $h_{j+1}, \ldots, h_n$, and a curve $\mathcal{C}'_j$, such that $\mathbf{v} \in \mathcal{FF}_v(P)$ if and only if

$$(v_1, \ldots, v_j) \in \mathcal{FF}_v(T', \mathbf{z}', \mathcal{C}_1, \ldots, \mathcal{C}_{j-1}, \mathcal{C}'_j),$$
$$v_k = h_k(v_j), \quad k = j+1, \ldots, n$$

Tree-Reduction Theorem - the definition of h_k and $\mathcal{C}'_j$

Let

$$\tilde{\sigma}_k(t) = \sigma_k(t) - z_{kj}\frac{|\sigma_k(t)|^2}{(\nu_k(t))^2},$$
$$\phi_k = \tilde{\sigma}_k \circ \tilde{\nu}_k^{-1},$$

Then,

$$h_k(v_j) = v_j - z_{kj}\frac{{\phi_k}^*(|v_j|)}{v_j^*},$$

and $\mathcal{C}'_j = \text{image}(\nu_j, \sigma_j)$ with

$$\nu_j(t) = (1-t)\cdot(\min U_j) + t\cdot(\max U_j),$$
$$\sigma_j(t) = \begin{cases} \hat{s}_j + \displaystyle\sum_{k=j+1}^{n}(\phi_k \circ \nu_j)(t), & j \neq 1, \\ 0 & j = 1. \end{cases}$$

Tree reduction theorem - example

The big picture (j = 2)

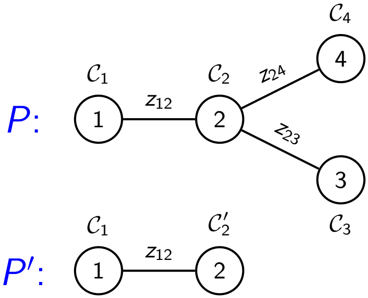
$$(v_1, v_2, v_3, v_4)^T \in \mathcal{F}\mathcal{F}_v(P)$$

$$\Longleftrightarrow$$

$$(v_1, v_2)^T \in \mathcal{F}\mathcal{F}_v(P'),$$

$$v_3 = h_3(v_2),$$

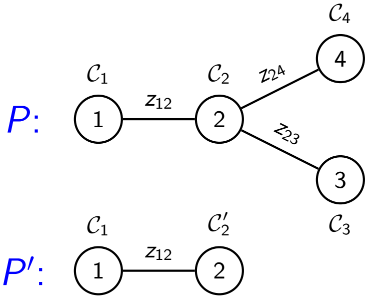
$$v_4 = h_4(v_2).$$



Tree reduction theorem - example

The big picture (j = 2)

$$(v_1, v_2, v_3, v_4)^T \in \mathcal{FF}_v(P)$$
$$\Longleftrightarrow$$
$$(v_1, v_2)^T \in \mathcal{FF}_v(P'),$$
$$v_3 = h_3(v_2),$$
$$v_4 = h_4(v_2).$$



Details:

- Compute $\tilde{\nu}_3$ and $\tilde{\nu}_4$. Verify invertability.
- Compute $U_2 = [\underline{u}_2, \overline{u}_2] \cap \text{image}(\tilde{\nu}_3) \cap \text{image}(\tilde{\nu}_4)$. Verify $U_2 \neq \emptyset$.
- Compute the functions h_3 and h_4, and the curve $\mathcal{C}_2'$.

Tree reduction theorem - example

The big picture (j $=2$)

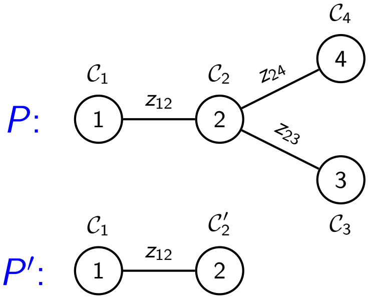
$$(v_1, v_2, v_3, v_4)^T \in \mathcal{FF}_v(P)$$

$$\iff$$

$$(v_1, v_2)^T \in \mathcal{FF}_v(P'),$$

$$v_3 = h_3(v_2),$$

$$v_4 = h_4(v_2).$$



Details:

- Compute $\tilde{\nu}_3$ and $\tilde{\nu}_4$. Verify invertability.
- Compute $U_2 = [\underline{u}_2, \overline{u}_2] \cap \text{image}(\tilde{\nu}_3) \cap \text{image}(\tilde{\nu}_4)$. Verify $U_2 \neq \emptyset$.
- Compute the functions h_3 and h_4, and the curve $\mathcal{C}_2'$.

Tree reduction theorem - example

$$(v_1, v_2, v_3, v_4)^T \in \mathcal{FF}_v(P)$$

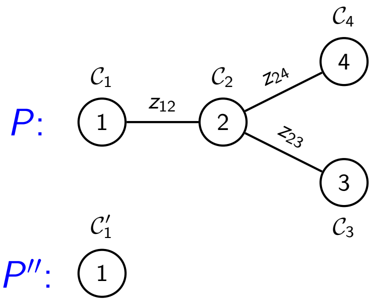
$$\Longleftrightarrow$$

$$v_1 \in \mathcal{FF}_v(P''),$$

$$v_2 = h_2(v_1),$$

$$v_3 = h_3(v_2),$$

$$v_4 = h_4(v_2).$$



Tree reduction theorem - example

$$(v_1, v_2, v_3, v_4)^T \in \mathcal{F}\mathcal{F}_v(P)$$

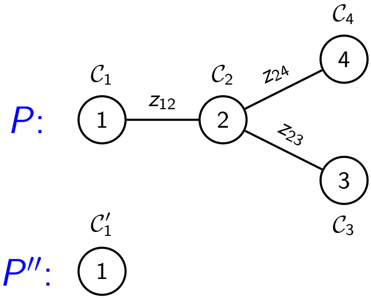
$$\Longleftrightarrow$$

$$v_1 \in \mathcal{F}\mathcal{F}_v(P''),$$

$$v_2 = h_2(v_1),$$

$$v_3 = h_3(v_2),$$

$$v_4 = h_4(v_2).$$

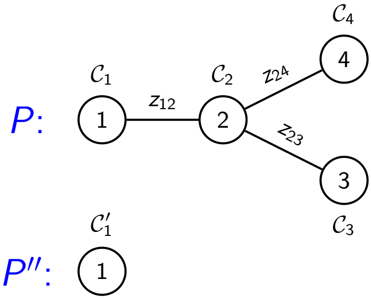


Observation

- $\mathcal{C}_1' = U_1 \times \{0\}$

Tree reduction theorem - example

$$(v_1, v_2, v_3, v_4)^T \in \mathcal{FF}_v(P)$$
$$\Longleftrightarrow$$
$$v_1 \in \mathcal{FF}_v(P''),$$
$$v_2 = h_2(v_1),$$
$$v_3 = h_3(v_2),$$
$$v_4 = h_4(v_2).$$



Observation

- $\mathcal{C}_1' = U_1 \times \{0\}$
- $\implies \mathcal{FF}(P'') = \{(v_1, s_1) : |v_1| \in U_1,\ s_1 = 0,\ \arg(v_1) = 0\}$

Tree reduction theorem - example

$$(v_1, v_2, v_3, v_4)^T \in \mathcal{F}\mathcal{F}_v(P)$$

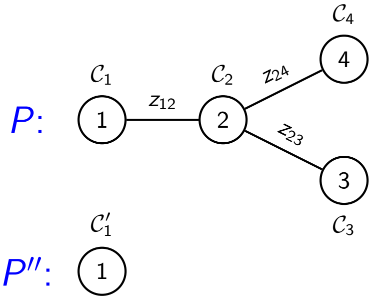
$$\Longleftrightarrow$$

$$v_1 \in \mathcal{F}\mathcal{F}_v(P''),$$

$$v_2 = h_2(v_1),$$

$$v_3 = h_3(v_2),$$

$$v_4 = h_4(v_2).$$



Observation

- $\mathcal{C}_1' = U_1 \times \{0\}$
- $\implies \mathcal{F}\mathcal{F}(P'') = \{(v_1, s_1) : |v_1| \in U_1,\ s_1 = 0,\ \arg(v_1) = 0\}$
- $\implies \mathcal{F}\mathcal{F}_v(P'') = U_1.$

Conclusion

$$(v_1, v_2, v_3, v_4)^T \in \mathcal{F}\mathcal{F}_v(P)$$
$$\Longleftrightarrow$$
$$\begin{aligned} v_1 &\in U_1, \\ v_2 &= h_2(v_1), \\ v_3 &= h_3(v_2), \\ v_4 &= h_4(v_2). \end{aligned}$$

Invertability $\implies$ the feasible set is a curve $\gamma_1 : [0, 1] \to \mathbb{R} \times \mathbb{C}$

The above is an algorithm to evaluate $\gamma_1(t)$

Two-phase meta-algorithm

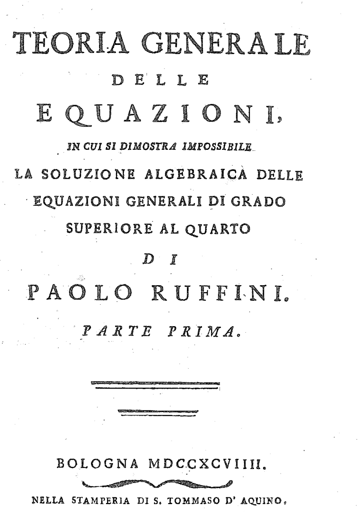
Phase 1 - tree reduction

- Perform a sequence of reductions, until one-node network is obtained.
- Compute h_k functions (ϕ_k in practice) and U_j intervals.
- If any $U_j = \emptyset$ - return "problem infeasible"
- If any $\tilde{\nu}_k$ non-invertible - return "error"

Phase 2 - tree expansion

- Take any $v_1 \in U_1$.
- Use expand v_1 to the full $\mathbf{v}$ vector using h_k functions.
- Compute the $\mathbf{s}$ vector.

Inverting $\tilde{\nu}_k$


TEORIA GENERALE

DELLE

EQUAZIONI,

IN CUI SI DIMOSTRA IMPOSSIBILE

LA SOLUZIONE ALGEBRAICA DELLE

EQUAZIONI GENERALI DI GRADO

SUPERIORE AL QUARTO

DI

PAOLO RUFFINI.

PARTE PRIMA.

BOLOGNA MDCCXCVIIII.

NELLA STAMPERIA DI S. TOMMASO D' AQUINO,

source: Wikipedia

The remedy - spline approximation

Choose approx. density d. Let $\mathbf{t} = \frac{1}{d-1}(0, 1, 2, \dots, d-1)$.

- ν_k and σ_k are approximated by vectors:

$$\boldsymbol{\nu}_k = (\nu_k(t_1), \dots, \nu_k(t_d)),$$
$$\boldsymbol{\sigma}_k = (\sigma_k(t_1), \dots, \sigma_k(t_d)).$$

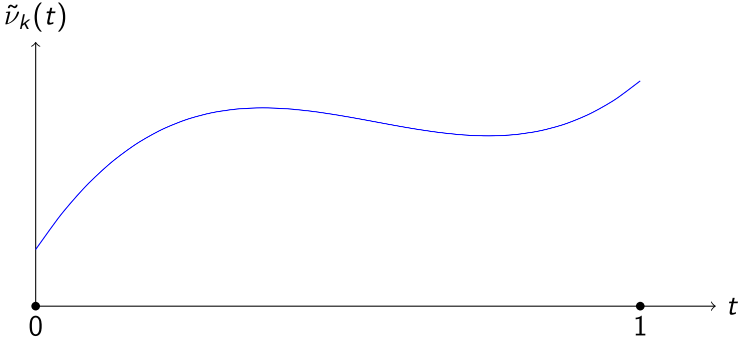
- Approximations of $\tilde{\nu}_k$ and $\tilde{\sigma}_k$ are computed componentwise:

$$\tilde{\boldsymbol{\nu}}_k = \left| \boldsymbol{\nu}_k - z_{kj} \frac{(\boldsymbol{\sigma}_k)^*}{\boldsymbol{\nu}_k} \right|,$$
$$\tilde{\boldsymbol{\sigma}}_k = \dots$$

- $\phi_k = \tilde{\sigma}_k \circ \tilde{\nu}_k^{-1}$ is approximated by a cubic spline interpolant mapping the components of $\tilde{\boldsymbol{\nu}}_k$ to $\tilde{\boldsymbol{\sigma}}_k$.

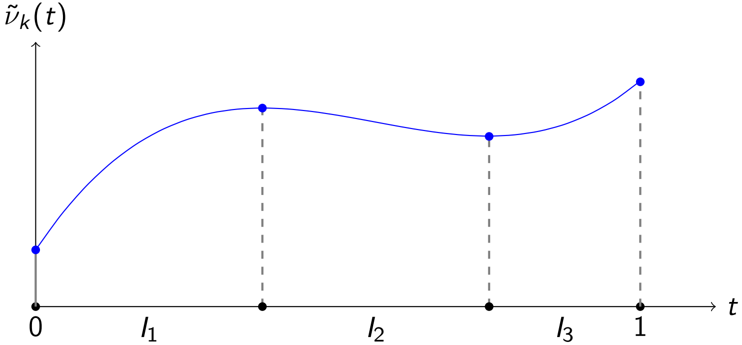
Multiple curves

$$\tilde{\nu}_k(t) \text{ invertible } \iff \text{ it is strictly monotone.}$$



Multiple curves

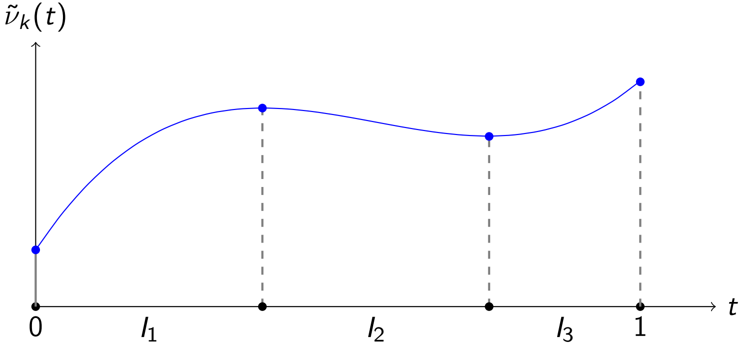
$$\tilde{\nu}_k(t) \text{ invertible} \iff \text{it is strictly monotone.}$$



Clearly, $\tilde{\nu}_k$ is invertible on I_1, I_2, and I_3.

Multiple curves

$\tilde{\nu}_k(t)$ invertible $\iff$ it is strictly monotone.

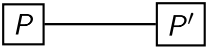


Clearly, $\tilde{\nu}_k$ is invertible on I_1, I_2, and I_3.
Solution: Split the network P into 3 networks.

Multiple curves - cont.

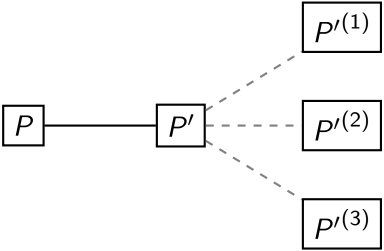
$\boxed{P}$

Multiple curves - cont.



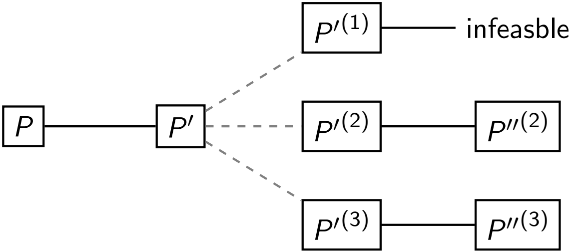
Reduction

Multiple curves - cont.



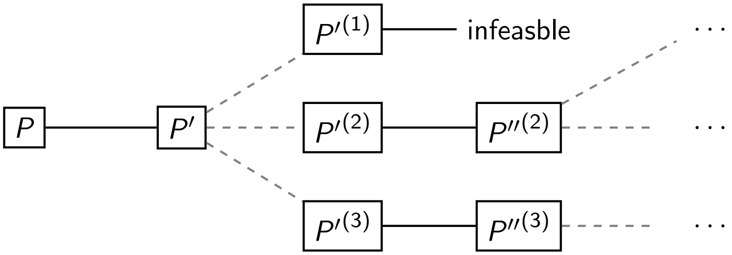
Reduction Split

Multiple curves - cont.



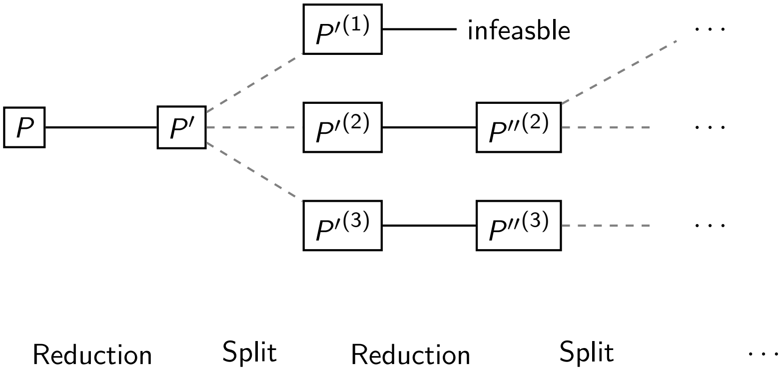
Reduction Split Reduction

Multiple curves - cont.



Reduction Split Reduction Split $\cdots$

Multiple curves - cont.



Eventually: multiple terminal single-node networks

Numerical results

Experiment setup

- Test networks: IEEE radial distribution feeders. `http://sites.ieee.org/pes-testfeeders/`
- Modified to conform to the assumptions
  - PV constraints on leaves only.
  - z_{ij} for each edge $(i, j) \in E$.
- Several tests
  - Accuracy - distance from the feasible set w.r.t the approximation density.
  - Reliability - comparison with MATPOWER[3].
  - Number of parallel networks in practice.
- Software available from `https://github.com/alexshtf/trem_opf_solver`.

[3] A well-known PF and OPF solver. Stability function can be specified using the extension mechanism.

Accuracy

Test setup

- Check several approximation densities $d \in [2^3, 2^{12}]$.
- For each approximation density, sample each curve at $N = 1000$ points.
- For each deviation measure, report the highest value among the samples.

Deviation measures

PQ constraints

$$E_{PQ,v}(\mathbf{v}, \mathbf{s}) = \max_{j \in PQ} d\left(|v_j|, [\underline{u}_j, \overline{u}_j]\right),$$

$$E_{PQ,s}(\mathbf{v}, \mathbf{s}) = \max_{j \in PQ} |s_j - \hat{s}_j|,$$

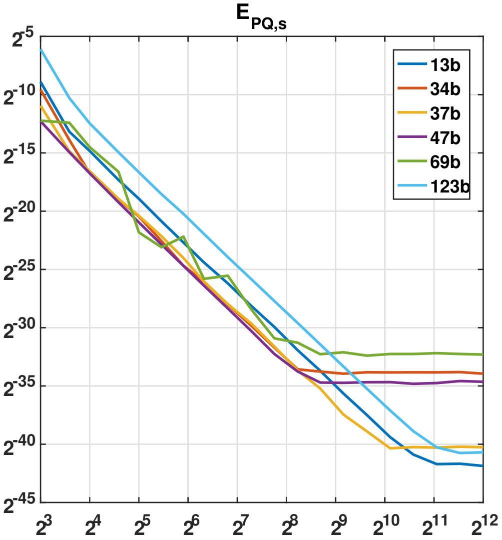
PV constraints

$$E_{PV,v}(\mathbf{v}, \mathbf{s}) = \max_{j \in PV} |\hat{u}_j - |v_j||$$

$$E_{PV,p}(\mathbf{v}, \mathbf{s}) = \max_{j \in PV} |\hat{p}_j - \mathrm{re}(s_j)|$$

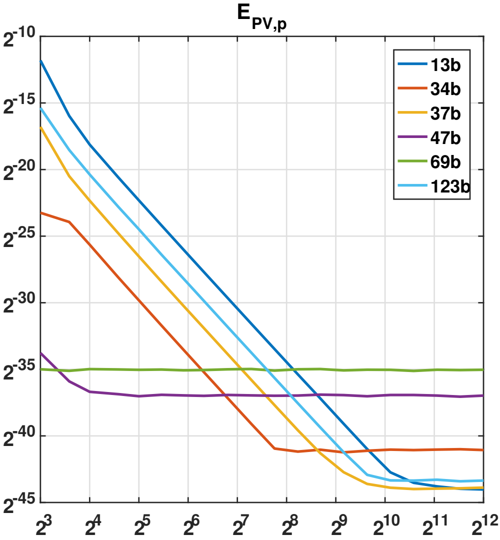
$$E_{PV,q}(\mathbf{v}, \mathbf{s}) = \max_{j \in PV} d\left(\mathrm{im}(s_j), \underline{q}_j, \overline{q}_j\right)$$

Accuracy



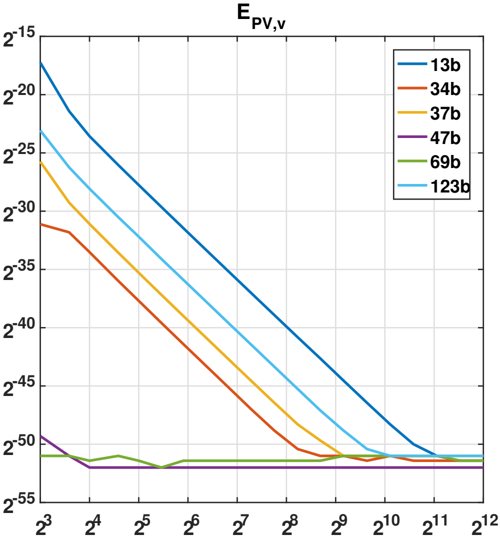
$E_{PQ,s}$
13b
34b
37b
47b
69b
123b
2^{-5}
2^{-10}
2^{-15}
2^{-20}
2^{-25}
2^{-30}
2^{-35}
2^{-40}
2^{-45}
2^3
2^4
2^5
2^6
2^7
2^8
2^9
2^{10}
2^{11}
2^{12}

Accuracy



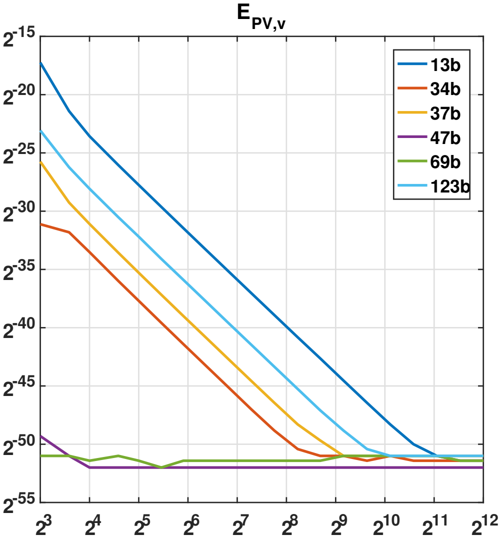
$E_{PV,p}$
2^{-10}
2^{-15}
2^{-20}
2^{-25}
2^{-30}
2^{-35}
2^{-40}
2^{-45}
2^{3}
2^{4}
2^{5}
2^{6}
2^{7}
2^{8}
2^{9}
2^{10}
2^{11}
2^{12}
13b
34b
37b
47b
69b
123b

Accuracy



$E_{PV,v}$
2^{-15}
2^{-20}
2^{-25}
2^{-30}
2^{-35}
2^{-40}
2^{-45}
2^{-50}
2^{-55}
2^{3}
2^{4}
2^{5}
2^{6}
2^{7}
2^{8}
2^{9}
2^{10}
2^{11}
2^{12}
13b
34b
37b
47b
69b
123b

Accuracy



$E_{PQ,v}$ and $E_{PV,q}$ were zero in all our experiments.

Reliability

Test setup

- Perturb the PQ constraints of each network:

$$[\underline{u}_j, \overline{u}_j] \times \{\mathrm{re}(\hat{s}_j) \cdot \alpha_j + (\mathrm{im}(\hat{s}_j) \cdot \beta_j)\imath\},$$

  where $\alpha_j, \beta_j \sim U[0,2]$.
- Generate 5000 random networks from each existing network.
- Solve OPF with the "stability" objective function:

$$f(\mathbf{v}, \mathbf{s}) = \sum_{j \in PQ} \left| |v_j| - \frac{1}{2}(\underline{u}_j + \overline{u}_j) \right|$$

- Solve using both MATPOWER and our solver.
- Gather statistics.

Reliability

13-node	34-node	37-node	47-node	69-node	123-node
6.46%	5.22%	2.56%	2.78%	2.34%	39.64%

The % of random networks, generated from each original network, for which our method found a solution while MATPOWER did not.

Number of networks in parallel

Observation: $\tilde{\nu}_k$ becomes more 'wild' when $|z|$ increases:

$$\tilde{\nu}_k(t) = |\nu_k(t) - z_{kj}(\sigma_k(t))^*/\nu_k(t)|$$

Number of networks in parallel

Observation: $\tilde{\nu}_k$ becomes more 'wild' when $|z|$ increases:

$$\tilde{\nu}_k(t) = |\nu_k(t) - z_{kj}(\sigma_k(t))^*/\nu_k(t)|$$

Setup: Make $\tilde{\nu}_k$ non-invertible my replacing $\mathbf{z}$ with $\alpha\mathbf{z}$ for $\alpha \in [1, 10]$.

Number of networks in parallel

Observation: $\tilde{\nu}_k$ becomes more 'wild' when $|z|$ increases:

$$\tilde{\nu}_k(t) = |\nu_k(t) - z_{kj}(\sigma_k(t))^* / \nu_k(t)|$$

Setup: Make $\tilde{\nu}_k$ non-invertible my replacing $\mathbf{z}$ with $\alpha\mathbf{z}$ for $\alpha \in [1, 10]$.

Results - Maximum (worst) number of networks in parallel

13-node	34-node	37-node	47-node	69-node	123-node
3	5	4	1	2	2

Questions?