Anticipatory Power Flow

Formulation, Computation, and Applications

Christian Y. Cahig

Mindanao State University – Iligan Institute of Technology

christian.cahig@outlook.com christian-cahig

The steady-state grid as a weighted graph $(\mathcal{N}, \mathcal{E})$

N buses as nodes, E branches as edges,

- $\mathcal{N}$: key locations (*e.g.*, substation)
- $\mathcal{E}$: bus-to-bus interfaces (*e.g.*, lines, cables, transformers)

Branch admittances act as edge weights

- How well a branch admits power flow

Bus admittance matrix $\boldsymbol{Y} = \boldsymbol{G} + \mathrm{j}\boldsymbol{B}$ act as graph Laplacian

- Sparsity reflects grid topology
- Elements computed from admittances

☞ $\boldsymbol{Y}$ encodes grid intrinsics

The steady-state grid as a weighted graph $(\mathcal{N}, \mathcal{E})$

N buses as nodes, E branches as edges,

- $\mathcal{N}$: key locations (*e.g.*, substation)
- $\mathcal{E}$: bus-to-bus interfaces (*e.g.*, lines, cables, transformers)

Branch admittances act as edge weights

- How well a branch admits power flow

Bus admittance matrix $\boldsymbol{Y} = \boldsymbol{G} + \mathrm{j}\boldsymbol{B}$ act as graph Laplacian

- Sparsity reflects grid topology
- Elements computed from admittances

☞ $\boldsymbol{Y}$ encodes grid intrinsics

The steady-state grid as a weighted graph $(\mathcal{N}, \mathcal{E})$

N buses as nodes, E branches as edges,

- $\mathcal{N}$: key locations (*e.g.*, substation)
- $\mathcal{E}$: bus-to-bus interfaces (*e.g.*, lines, cables, transformers)

Branch admittances act as edge weights

- How well a branch admits power flow

Bus admittance matrix $\boldsymbol{Y} = \boldsymbol{G} + \mathrm{j}\boldsymbol{B}$ act as graph Laplacian

- Sparsity reflects grid topology
- Elements computed from admittances

☞ $\boldsymbol{Y}$ encodes grid intrinsics

The steady-state grid as a weighted graph $(\mathcal{N}, \mathcal{E})$

Buses are entry and exit points for power

- Demand units draw $\boldsymbol{p}_{\mathsf{d}} + \mathrm{j}\boldsymbol{q}_{\mathsf{d}} \in \mathbb{C}^{U}$
- Supply units inject $\boldsymbol{p}_{\mathsf{u}} + \mathrm{j}\boldsymbol{q}_{\mathsf{u}} \in \mathbb{C}^{D}$
- $\boldsymbol{C}_{\mathsf{u}}$, $\boldsymbol{C}_{\mathsf{d}}$: connection matrices
- $\boldsymbol{d} := (\boldsymbol{p}_{\mathsf{d}}, \boldsymbol{q}_{\mathsf{d}})$ and $\boldsymbol{c} := (\boldsymbol{p}_{\mathsf{u}}, \boldsymbol{q}_{\mathsf{u}})$

☞ Net nodal injections: $\boldsymbol{C}_{\mathsf{u}} (\boldsymbol{p}_{\mathsf{u}} + \mathrm{j}\boldsymbol{q}_{\mathsf{u}}) - \boldsymbol{C}_{\mathsf{d}} (\boldsymbol{p}_{\mathsf{d}} + \mathrm{j}\boldsymbol{q}_{\mathsf{d}})$

Bus voltages $\boldsymbol{v}\underline{/\boldsymbol{\delta}} \equiv \boldsymbol{v}\mathrm{e}^{\mathrm{j}\boldsymbol{\delta}}$ are state variables

- For a grid with $\boldsymbol{Y}$, its state is computable given $\boldsymbol{s} := (\boldsymbol{v}, \boldsymbol{\delta})$

☞ Net nodal injections: $\mathrm{Diag}(\boldsymbol{v}\underline{/\boldsymbol{\delta}})\,\mathrm{conj}(\boldsymbol{Y}\boldsymbol{v}\underline{/\boldsymbol{\delta}})$

The steady-state grid as a weighted graph $(\mathcal{N}, \mathcal{E})$

Buses are entry and exit points for power

- Demand units draw $\boldsymbol{p}_{\mathsf{d}} + \mathrm{j}\boldsymbol{q}_{\mathsf{d}} \in \mathbb{C}^U$
- Supply units inject $\boldsymbol{p}_{\mathsf{u}} + \mathrm{j}\boldsymbol{q}_{\mathsf{u}} \in \mathbb{C}^D$
- $\boldsymbol{C}_{\mathsf{u}}$, $\boldsymbol{C}_{\mathsf{d}}$: connection matrices
- $\boldsymbol{d} := (\boldsymbol{p}_{\mathsf{d}}, \boldsymbol{q}_{\mathsf{d}})$ and $\boldsymbol{c} := (\boldsymbol{p}_{\mathsf{u}}, \boldsymbol{q}_{\mathsf{u}})$

☞ Net nodal injections: $\boldsymbol{C}_{\mathsf{u}} (\boldsymbol{p}_{\mathsf{u}} + \mathrm{j}\boldsymbol{q}_{\mathsf{u}}) - \boldsymbol{C}_{\mathsf{d}} (\boldsymbol{p}_{\mathsf{d}} + \mathrm{j}\boldsymbol{q}_{\mathsf{d}})$

Bus voltages $\boldsymbol{v}\underline{/\boldsymbol{\delta}} \equiv \boldsymbol{v}\mathrm{e}^{\mathrm{j}\boldsymbol{\delta}}$ are state variables

- For a grid with $\boldsymbol{Y}$, its state is computable given $\boldsymbol{s} := (\boldsymbol{v}, \boldsymbol{\delta})$

☞ Net nodal injections: $\mathrm{Diag}(\boldsymbol{v}\underline{/\boldsymbol{\delta}})\,\mathrm{conj}(\boldsymbol{Y}\boldsymbol{v}\underline{/\boldsymbol{\delta}})$

Power flow equations (PFE)

The standard model of steady-state grid physics

Nodal power balance

$$\mathrm{Diag}(\boldsymbol{v}\underline{/\boldsymbol{\delta}})\,\mathrm{conj}(\boldsymbol{Y}\boldsymbol{v}\underline{/\boldsymbol{\delta}}) = \boldsymbol{C}_{\mathsf{u}}\left(\boldsymbol{p}_{\mathsf{u}} + \mathrm{j}\boldsymbol{q}_{\mathsf{u}}\right) - \boldsymbol{C}_{\mathsf{d}}\left(\boldsymbol{p}_{\mathsf{d}} + \mathrm{j}\boldsymbol{q}_{\mathsf{d}}\right)$$

or, equivalently, as a system of $2N$ nonlinear equations,

$$\begin{bmatrix} \mathrm{Diag}(\boldsymbol{v} \odot \cos\boldsymbol{\delta}) & \mathrm{Diag}(\boldsymbol{v} \odot \sin\boldsymbol{\delta}) \\ \mathrm{Diag}(\boldsymbol{v} \odot \sin\boldsymbol{\delta}) & -\mathrm{Diag}(\boldsymbol{v} \odot \cos\boldsymbol{\delta}) \end{bmatrix} \begin{bmatrix} \boldsymbol{G} & -\boldsymbol{B} \\ \boldsymbol{B} & \boldsymbol{G} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \odot \cos\boldsymbol{\delta} \\ \boldsymbol{v} \odot \sin\boldsymbol{\delta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{\mathsf{u}}\boldsymbol{p}_{\mathsf{u}} - \boldsymbol{C}_{\mathsf{d}}\boldsymbol{p}_{\mathsf{d}} \\ \boldsymbol{C}_{\mathsf{u}}\boldsymbol{q}_{\mathsf{u}} - \boldsymbol{C}_{\mathsf{d}}\boldsymbol{q}_{\mathsf{d}} \end{bmatrix}$$

- Staple requirement in power flow analysis (PFA) computations
- PFE parameters: $\boldsymbol{Y}$ as grid intrinsics, $\boldsymbol{d}$ as external stimuli
- $(\boldsymbol{c}, \boldsymbol{s})$ are power-flow feasible for $(\boldsymbol{Y}, \boldsymbol{d})$ if they satisfy the PFE for $(\boldsymbol{Y}, \boldsymbol{d})$

Power flow equations (PFE)

The standard model of steady-state grid physics

Nodal power balance

$$\mathrm{Diag}(\boldsymbol{v}\angle\boldsymbol{\delta})\,\mathrm{conj}(\boldsymbol{Y}\boldsymbol{v}\angle\boldsymbol{\delta}) = \boldsymbol{C}_{\mathsf{u}}\left(\boldsymbol{p}_{\mathsf{u}} + \mathrm{j}\boldsymbol{q}_{\mathsf{u}}\right) - \boldsymbol{C}_{\mathsf{d}}\left(\boldsymbol{p}_{\mathsf{d}} + \mathrm{j}\boldsymbol{q}_{\mathsf{d}}\right)$$

or, equivalently, as a system of $2N$ nonlinear equations,

$$\begin{bmatrix} \mathrm{Diag}(\boldsymbol{v}\odot\cos\boldsymbol{\delta}) & \mathrm{Diag}(\boldsymbol{v}\odot\sin\boldsymbol{\delta}) \\ \mathrm{Diag}(\boldsymbol{v}\odot\sin\boldsymbol{\delta}) & -\mathrm{Diag}(\boldsymbol{v}\odot\cos\boldsymbol{\delta}) \end{bmatrix} \begin{bmatrix} \boldsymbol{G} & -\boldsymbol{B} \\ \boldsymbol{B} & \boldsymbol{G} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}\odot\cos\boldsymbol{\delta} \\ \boldsymbol{v}\odot\sin\boldsymbol{\delta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{\mathsf{u}}\boldsymbol{p}_{\mathsf{u}} - \boldsymbol{C}_{\mathsf{d}}\boldsymbol{p}_{\mathsf{d}} \\ \boldsymbol{C}_{\mathsf{u}}\boldsymbol{q}_{\mathsf{u}} - \boldsymbol{C}_{\mathsf{d}}\boldsymbol{q}_{\mathsf{d}} \end{bmatrix}$$

- Staple requirement in power flow analysis (PFA) computations
- PFE parameters: $\boldsymbol{Y}$ as grid intrinsics, $\boldsymbol{d}$ as external stimuli
- $(\boldsymbol{c}, \boldsymbol{s})$ are power-flow feasible for $(\boldsymbol{Y}, \boldsymbol{d})$ if they satisfy the PFE for $(\boldsymbol{Y}, \boldsymbol{d})$

Power flow equations (PFE)

The standard model of steady-state grid physics

Nodal power balance

$$\mathrm{Diag}(\boldsymbol{v}\underline{/\boldsymbol{\delta}})\,\mathrm{conj}(\boldsymbol{Y}\boldsymbol{v}\underline{/\boldsymbol{\delta}}) = \boldsymbol{C}_{\mathsf{u}}\,(\boldsymbol{p}_{\mathsf{u}} + \mathrm{j}\boldsymbol{q}_{\mathsf{u}}) - \boldsymbol{C}_{\mathsf{d}}\,(\boldsymbol{p}_{\mathsf{d}} + \mathrm{j}\boldsymbol{q}_{\mathsf{d}})$$

or, equivalently, as a system of $2N$ nonlinear equations,

$$\begin{bmatrix} \mathrm{Diag}(\boldsymbol{v}\odot\cos\boldsymbol{\delta}) & \mathrm{Diag}(\boldsymbol{v}\odot\sin\boldsymbol{\delta}) \\ \mathrm{Diag}(\boldsymbol{v}\odot\sin\boldsymbol{\delta}) & -\mathrm{Diag}(\boldsymbol{v}\odot\cos\boldsymbol{\delta}) \end{bmatrix} \begin{bmatrix} \boldsymbol{G} & -\boldsymbol{B} \\ \boldsymbol{B} & \boldsymbol{G} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}\odot\cos\boldsymbol{\delta} \\ \boldsymbol{v}\odot\sin\boldsymbol{\delta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{\mathsf{u}}\boldsymbol{p}_{\mathsf{u}} - \boldsymbol{C}_{\mathsf{d}}\boldsymbol{p}_{\mathsf{d}} \\ \boldsymbol{C}_{\mathsf{u}}\boldsymbol{q}_{\mathsf{u}} - \boldsymbol{C}_{\mathsf{d}}\boldsymbol{q}_{\mathsf{d}} \end{bmatrix}$$

- Staple requirement in power flow analysis (PFA) computations
- PFE parameters: $\boldsymbol{Y}$ as grid intrinsics, $\boldsymbol{d}$ as external stimuli
- $(\boldsymbol{c}, \boldsymbol{s})$ are power-flow feasible for $(\boldsymbol{Y}, \boldsymbol{d})$ if they satisfy the PFE for $(\boldsymbol{Y}, \boldsymbol{d})$

Power flow manifold (PFM)

A geometric intuition for the PFE

PFE as a manifold in $(\boldsymbol{c}, \boldsymbol{s})$–space

Expressed as $\phi(\boldsymbol{c}, \boldsymbol{s}; \boldsymbol{Y}, \boldsymbol{d}) = \mathbf{0}_{2N}$, the PFE describe a manifold of all points $(\boldsymbol{c}, \boldsymbol{s})$ that are power-flow feasible for $(\boldsymbol{Y}, \boldsymbol{d})$.

- PFE parameters $(\boldsymbol{Y}, \boldsymbol{d})$ dictate the “shape” of PFM
- Power-flow feasible $(\boldsymbol{c}, \boldsymbol{s})$ for $(\boldsymbol{Y}, \boldsymbol{d}) \implies (\boldsymbol{c}, \boldsymbol{s})$ is on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$
- Computation over PFE for $(\boldsymbol{Y}, \boldsymbol{d}) \implies$ finding a point on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$

Power flow manifold (PFM)

A geometric intuition for the PFE

PFE as a manifold in $(\boldsymbol{c}, \boldsymbol{s})$–space

Expressed as $\phi(\boldsymbol{c}, \boldsymbol{s}; \boldsymbol{Y}, \boldsymbol{d}) = \mathbf{0}_{2N}$, the PFE describe a manifold of all points $(\boldsymbol{c}, \boldsymbol{s})$ that are power-flow feasible for $(\boldsymbol{Y}, \boldsymbol{d})$.

- PFE parameters $(\boldsymbol{Y}, \boldsymbol{d})$ dictate the "shape" of PFM
- Power-flow feasible $(\boldsymbol{c}, \boldsymbol{s})$ for $(\boldsymbol{Y}, \boldsymbol{d}) \implies (\boldsymbol{c}, \boldsymbol{s})$ is on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$
- Computation over PFE for $(\boldsymbol{Y}, \boldsymbol{d}) \implies$ finding a point on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$

PFE as a manifold in $(\boldsymbol{c}, \boldsymbol{s})$–space

Expressed as $\phi(\boldsymbol{c}, \boldsymbol{s}; \boldsymbol{Y}, \boldsymbol{d}) = \mathbf{0}_{2N}$, the PFE describe a manifold of all points $(\boldsymbol{c}, \boldsymbol{s})$ that are power-flow feasible for $(\boldsymbol{Y}, \boldsymbol{d})$.

- PFE parameters $(\boldsymbol{Y}, \boldsymbol{d})$ dictate the “shape” of PFM
- Power-flow feasible $(\boldsymbol{c}, \boldsymbol{s})$ for $(\boldsymbol{Y}, \boldsymbol{d}) \Longrightarrow$ $(\boldsymbol{c}, \boldsymbol{s})$ is on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$
- Computation over PFE for $(\boldsymbol{Y}, \boldsymbol{d}) \Longrightarrow$ finding a point on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$

Finding a point on the PFM

- Standard power flow (SPF) as completing a point on PFM
  - Fix some coordinates with slack-PV-PQ model (§2.4.1)
    - ★ Find $k \in \mathcal{N}$ with supply unit, set δ_k as reference
    - ★ Assume v_m and $p_{\mathsf{u},n}$ for all $m \in \mathcal{N}$ with supply units, and all unit n at $m \neq k$
  - Complete the other coordinates from PFE

Finding a point on the PFM

- Standard power flow (SPF) as completing a point on PFM
  - Fix some coordinates with slack-PV-PQ model (§2.4.1)
    - Find $k \in \mathcal{N}$ with supply unit, set δ_k as reference
    - Assume v_m and $p_{\mathrm{u},n}$ for all $m \in \mathcal{N}$ with supply units, and all unit n at $m \neq k$
  - Complete the other coordinates from PFE
- Continuation power flow (CPF) as finding points through PFMs
  - Estimating stability limits: $\boldsymbol{\phi}(\boldsymbol{Y}, \boldsymbol{d}_i) = \mathbf{0}_{2N}$ with $\boldsymbol{d}_i = \lambda_i \boldsymbol{d}_{\mathrm{base}}$ [1–6]
  - Branch-outage contingency: $\boldsymbol{\phi}(\boldsymbol{Y}_i, \boldsymbol{d}) = \mathbf{0}_{2N}$ by [7–10]

Finding a point on the PFM

- Standard power flow (SPF) as completing a point on PFM
  - Fix some coordinates with slack-PV-PQ model (§2.4.1)
    - Find $k \in \mathcal{N}$ with supply unit, set δ_k as reference
    - Assume v_m and $p_{\mathrm{u},n}$ for all $m \in \mathcal{N}$ with supply units, and all unit n at $m \neq k$
  - Complete the other coordinates from PFE

- Continuation power flow (CPF) as finding points through PFMs
  - Estimating stability limits: $\boldsymbol{\phi}(\boldsymbol{Y}, \boldsymbol{d}_i) = \mathbf{0}_{2N}$ with $\boldsymbol{d}_i = \lambda_i \boldsymbol{d}_{\mathrm{base}}$ [1–6]
  - Branch-outage contingency: $\boldsymbol{\phi}(\boldsymbol{Y}_i, \boldsymbol{d}) = \mathbf{0}_{2N}$ by [7–10]

- Optimal power flow (OPF) as finding the best point on PFM
  - Best point minimizes operating cost $g(\boldsymbol{c}, \boldsymbol{s})$ subject to
    - Supply capacity: $\underline{\boldsymbol{c}} \leq \boldsymbol{c} \leq \overline{\boldsymbol{c}}$
    - Allowable state: $\boldsymbol{A}\boldsymbol{s} \leq \boldsymbol{b}$
  - Nonconvex (due to PFE) and NP-hard [11–13]

PFA computation in the online setting

- Snapshot data of past operating point: $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ in response to $\widetilde{\boldsymbol{d}}$
  - ☞ Direct or computed from measurements

- Expected conditions for upcoming dispatch
  - ▶ Updated intrinsics $\boldsymbol{Y}$ from online parameter estimation [14, 15]
  - ▶ Anticipated demand $\boldsymbol{d}$ from forecast
  - ▶ Supply limits $\underline{\boldsymbol{c}}$, $\overline{\boldsymbol{c}}$ (*e.g.*, schedule, ramp)
  - ☞ $\boldsymbol{d} \neq \widetilde{\boldsymbol{d}}$, *i.e.*, $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ not on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$

- Invariant roster of supply & demand units (*i.e.*, fixed C_{u} and C_{d})

PFA computation in the online setting

- Snapshot data of past operating point: $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ in response to $\widetilde{\boldsymbol{d}}$
  - ☞ Direct or computed from measurements

- Expected conditions for upcoming dispatch
  - Updated intrinsics $\boldsymbol{Y}$ from online parameter estimation [14, 15]
  - Anticipated demand $\boldsymbol{d}$ from forecast
  - Supply limits $\underline{\boldsymbol{c}}$, $\overline{\boldsymbol{c}}$ (*e.g.*, schedule, ramp)
  - ☞ $\boldsymbol{d} \neq \widetilde{\boldsymbol{d}}$, *i.e.*, $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ not on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$

- Invariant roster of supply & demand units (*i.e.*, fixed C_{u} and C_{d})

PFA computation in the online setting

- Snapshot data of past operating point: $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ in response to $\widetilde{\boldsymbol{d}}$
  - ☞ Direct or computed from measurements

- Expected conditions for upcoming dispatch
  - ▶ Updated intrinsics $\boldsymbol{Y}$ from online parameter estimation [14, 15]
  - ▶ Anticipated demand $\boldsymbol{d}$ from forecast
  - ▶ Supply limits $\underline{\boldsymbol{c}}$, $\overline{\boldsymbol{c}}$ (*e.g.*, schedule, ramp)
  - ☞ $\boldsymbol{d} \neq \widetilde{\boldsymbol{d}}$, *i.e.*, $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ not on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$

- Invariant roster of supply & demand units (*i.e.*, fixed $\boldsymbol{C}_{\mathrm{u}}$ and $\boldsymbol{C}_{\mathrm{d}}$)

PFA computation in the online setting

- Snapshot data of past operating point: $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ in response to $\widetilde{\boldsymbol{d}}$
  - ☞ Direct or computed from measurements
- Expected conditions for upcoming dispatch
  - ▸ Updated intrinsics $\boldsymbol{Y}$ from online parameter estimation [14, 15]
  - ▸ Anticipated demand $\boldsymbol{d}$ from forecast
  - ▸ Supply limits $\underline{\boldsymbol{c}}$, $\overline{\boldsymbol{c}}$ (*e.g.*, schedule, ramp)
  - ☞ $\boldsymbol{d} \neq \widetilde{\boldsymbol{d}}$, *i.e.*, $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ not on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$
- Invariant roster of supply & demand units (*i.e.*, fixed C_{u} and C_{d})

PFA computation in the online setting

- Snapshot data of past operating point: $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ in response to $\widetilde{\boldsymbol{d}}$
  - ☞ Direct or computed from measurements
- Expected conditions for upcoming dispatch
  - ▸ Updated intrinsics $\boldsymbol{Y}$ from online parameter estimation [14, 15]
  - ▸ Anticipated demand $\boldsymbol{d}$ from forecast
  - ▸ Supply limits $\underline{\boldsymbol{c}}$, $\overline{\boldsymbol{c}}$ (*e.g.*, schedule, ramp)
  - ☞ $\boldsymbol{d} \neq \widetilde{\boldsymbol{d}}$, *i.e.*, $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$ not on PFM for $(\boldsymbol{Y}, \boldsymbol{d})$
- Invariant roster of supply & demand units (*i.e.*, fixed $\boldsymbol{C}_{\mathrm{u}}$ and $\boldsymbol{C}_{\mathrm{d}}$)

Using the snapshot point $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$, how do we find a point $(\boldsymbol{c}, \boldsymbol{x})$ on the PFM for $(\boldsymbol{Y}, \boldsymbol{d})$?

Pillars of anticipatory power flow (APF)

1. $\boldsymbol{c}$ as control, $\boldsymbol{s}$ as state, and $\boldsymbol{s}$ as an implicit function of $\boldsymbol{c}$ via PFE
   - OPF notion from '60s [16, §2.4]; used in recent works on online OPF [17, 18]
   - Compute $\boldsymbol{c}$ voltage-free, then solve for $\boldsymbol{s}$ from PFE

Pillars of anticipatory power flow (APF)

1. $\boldsymbol{c}$ as control, $\boldsymbol{s}$ as state, and $\boldsymbol{s}$ as an implicit function of $\boldsymbol{c}$ via PFE
   - OPF notion from '60s [16, §2.4]; used in recent works on online OPF [17, 18]
   - Compute $\boldsymbol{c}$ voltage-free, then solve for $\boldsymbol{s}$ from PFE

2. To avoid rotational degeneracy, pick any bus $\hat{n}$ as reference and $\delta_{\hat{n}} \leftarrow \delta_{\text{ref}}$
   - PFE are sinusoids of phase angle differences, not of phase angles
   - Slack-PV-PQ is mathematically unnecessary [19, §2.2.1]
   - Non-reference voltage phase angles: $\boldsymbol{\vartheta} \in \mathbb{R}^{N-1}$

Pillars of anticipatory power flow (APF)

1. $\boldsymbol{c}$ as control, $\boldsymbol{s}$ as state, and $\boldsymbol{s}$ as an implicit function of $\boldsymbol{c}$ via PFE
   - OPF notion from '60s [16, §2.4]; used in recent works on online OPF [17, 18]
   - Compute $\boldsymbol{c}$ voltage-free, then solve for $\boldsymbol{s}$ from PFE

2. To avoid rotational degeneracy, pick any bus $\hat{n}$ as reference and $\delta_{\hat{n}} \leftarrow \delta_{\text{ref}}$
   - PFE are sinusoids of phase angle differences, not of phase angles
   - Slack-PV-PQ is mathematically unnecessary [19, §2.2.1]
   - Non-reference voltage phase angles: $\boldsymbol{\vartheta} \in \mathbb{R}^{N-1}$

3. To make solving PFE well-determined, add a distributed slack $\kappa \in \mathbb{R}$
   - Based on SPF notion distributed slack bus, dating back to '90s [20]
   - κ shared via slack distribution ratios $\boldsymbol{\kappa}$ based on supply limits (§3.3)
   - With $\boldsymbol{x} := \left[\boldsymbol{v}; \boldsymbol{\vartheta}; \kappa\right]$ as state variables, restate PFE as $\boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{Y}, \boldsymbol{d}, \boldsymbol{c}, \delta_{\hat{n}}) = \boldsymbol{0}_{2N}$

Solving for the anticipated supply injections

Extended economic dispatch (ED+)

$$\begin{aligned}
\min_{\boldsymbol{p},\boldsymbol{q}} \quad & \underbrace{\|\boldsymbol{p}\|_2 + \|\boldsymbol{q}\|_2}_{f_{\text{loss}}} + \underbrace{\mu_{\text{p}}\|\boldsymbol{p} - \widetilde{\boldsymbol{p}}_{\text{u}}\|_2 + \mu_{\text{q}}\|\boldsymbol{q} - \widetilde{\boldsymbol{q}}_{\text{u}}\|_2}_{f_{\text{reg}}} \\
\text{s. t.} \quad & \mathbf{1}^{\mathsf{T}}\boldsymbol{p} = p_{\text{need}}, \quad \mathbf{1}^{\mathsf{T}}\boldsymbol{q} = q_{\text{need}}, \quad \underline{\boldsymbol{p}_{\text{u}}} \le \boldsymbol{p} \le \overline{\boldsymbol{p}_{\text{u}}}, \quad \underline{\boldsymbol{q}_{\text{u}}} \le \boldsymbol{q} \le \overline{\boldsymbol{q}_{\text{u}}}, \\
\text{with} \quad & p_{\text{need}} = \operatorname{clip}\left(\mathbf{1}^{\mathsf{T}}\underline{\boldsymbol{p}_{\text{u}}}, \mathbf{1}^{\mathsf{T}}\boldsymbol{p}_{\text{d}} + p_{\text{h}} + p_{\text{o}}, \mathbf{1}^{\mathsf{T}}\overline{\boldsymbol{p}_{\text{u}}}\right), \quad q_{\text{need}} = \operatorname{clip}\left(\mathbf{1}^{\mathsf{T}}\underline{\boldsymbol{q}_{\text{u}}}, \mathbf{1}^{\mathsf{T}}\boldsymbol{q}_{\text{d}} + q_{\text{h}} + q_{\text{o}}, \mathbf{1}^{\mathsf{T}}\overline{\boldsymbol{q}_{\text{u}}}\right), \\
& p_{\text{h}} + \mathrm{j}q_{\text{h}} = \operatorname{shunt}\left(\widetilde{\boldsymbol{v}}, \widetilde{\boldsymbol{\delta}}; \boldsymbol{Y}\right), \quad \text{and} \quad p_{\text{o}} + \mathrm{j}q_{\text{o}} = \operatorname{loss}\left(\widetilde{\boldsymbol{v}}, \widetilde{\boldsymbol{\delta}}; \boldsymbol{Y}\right).
\end{aligned}$$

- Vanilla ED but considers reactive powers
  - Supply regularization $\mu_{\text{p}}, \mu_{\text{q}} \ge 0$
  - Adjust for non-demand consumption: $\operatorname{shunt}(\cdot)$ (§3.2.1) and $\operatorname{loss}(\cdot)$ (§3.2.2)

Solving for the anticipated supply injections

Extended economic dispatch (ED+)

$$\min_{\boldsymbol{p},\boldsymbol{q}} \quad \underbrace{\|\boldsymbol{p}\|_2 + \|\boldsymbol{q}\|_2}_{f_{\text{loss}}} + \underbrace{\mu_{\text{p}}\|\boldsymbol{p} - \widetilde{\boldsymbol{p}}_{\text{u}}\|_2 + \mu_{\text{q}}\|\boldsymbol{q} - \widetilde{\boldsymbol{q}}_{\text{u}}\|_2}_{f_{\text{reg}}}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathsf{T}}\boldsymbol{p} = p_{\text{need}}, \quad \mathbf{1}^{\mathsf{T}}\boldsymbol{q} = q_{\text{need}}, \quad \underline{\boldsymbol{p}_{\text{u}}} \le \boldsymbol{p} \le \overline{\boldsymbol{p}_{\text{u}}}, \quad \underline{\boldsymbol{q}_{\text{u}}} \le \boldsymbol{q} \le \overline{\boldsymbol{q}_{\text{u}}},$$

$$\text{with} \quad p_{\text{need}} = \text{clip}\left(\mathbf{1}^{\mathsf{T}}\underline{\boldsymbol{p}_{\text{u}}}, \mathbf{1}^{\mathsf{T}}\boldsymbol{p}_{\text{d}} + p_{\text{h}} + p_{\text{o}}, \mathbf{1}^{\mathsf{T}}\overline{\boldsymbol{p}_{\text{u}}}\right), \quad q_{\text{need}} = \text{clip}\left(\mathbf{1}^{\mathsf{T}}\underline{\boldsymbol{q}_{\text{u}}}, \mathbf{1}^{\mathsf{T}}\boldsymbol{q}_{\text{d}} + q_{\text{h}} + q_{\text{o}}, \mathbf{1}^{\mathsf{T}}\overline{\boldsymbol{q}_{\text{u}}}\right),$$

$$p_{\text{h}} + \mathrm{j}q_{\text{h}} = \text{shunt}\left(\widetilde{\boldsymbol{v}}, \widetilde{\boldsymbol{\delta}}; \boldsymbol{Y}\right), \quad \text{and} \quad p_{\text{o}} + \mathrm{j}q_{\text{o}} = \text{loss}\left(\widetilde{\boldsymbol{v}}, \widetilde{\boldsymbol{\delta}}; \boldsymbol{Y}\right).$$

- Vanilla ED but considers reactive powers
  - Supply regularization $\mu_{\text{p}}, \mu_{\text{q}} \ge 0$
  - Adjust for non-demand consumption: $\text{shunt}(\cdot)$ (§3.2.1) and $\text{loss}(\cdot)$ (§3.2.2)

Solving for the anticipated supply injections

Extended economic dispatch (ED+)

$$\min_{\boldsymbol{p},\boldsymbol{q}} \quad \underbrace{\left\|\boldsymbol{p}\right\|_2 + \left\|\boldsymbol{q}\right\|_2}_{f_{\mathsf{loss}}} + \underbrace{\mu_{\mathsf{p}}\left\|\boldsymbol{p} - \widetilde{\boldsymbol{p}}_{\mathsf{u}}\right\|_2 + \mu_{\mathsf{q}}\left\|\boldsymbol{q} - \widetilde{\boldsymbol{q}}_{\mathsf{u}}\right\|_2}_{f_{\mathsf{reg}}}$$

$$\text{s. t.} \quad \mathbf{1}^{\mathsf{T}}\boldsymbol{p} = p_{\mathsf{need}}, \quad \mathbf{1}^{\mathsf{T}}\boldsymbol{q} = q_{\mathsf{need}}, \quad \underline{\boldsymbol{p}_{\mathsf{u}}} \leq \boldsymbol{p} \leq \overline{\boldsymbol{p}_{\mathsf{u}}}, \quad \underline{\boldsymbol{q}_{\mathsf{u}}} \leq \boldsymbol{q} \leq \overline{\boldsymbol{q}_{\mathsf{u}}},$$

$$\text{with} \quad p_{\mathsf{need}} = \operatorname{clip}\left(\mathbf{1}^{\mathsf{T}}\underline{\boldsymbol{p}_{\mathsf{u}}}, \mathbf{1}^{\mathsf{T}}\boldsymbol{p}_{\mathsf{d}} + p_{\mathsf{h}} + p_{\mathsf{o}}, \mathbf{1}^{\mathsf{T}}\overline{\boldsymbol{p}_{\mathsf{u}}}\right), \quad q_{\mathsf{need}} = \operatorname{clip}\left(\mathbf{1}^{\mathsf{T}}\underline{\boldsymbol{q}_{\mathsf{u}}}, \mathbf{1}^{\mathsf{T}}\boldsymbol{q}_{\mathsf{d}} + q_{\mathsf{h}} + q_{\mathsf{o}}, \mathbf{1}^{\mathsf{T}}\overline{\boldsymbol{q}_{\mathsf{u}}}\right),$$

$$p_{\mathsf{h}} + \mathrm{j}q_{\mathsf{h}} = \operatorname{shunt}\left(\widetilde{\boldsymbol{v}}, \widetilde{\boldsymbol{\delta}}; \boldsymbol{Y}\right), \quad \text{and} \quad p_{\mathsf{o}} + \mathrm{j}q_{\mathsf{o}} = \operatorname{loss}\left(\widetilde{\boldsymbol{v}}, \widetilde{\boldsymbol{\delta}}; \boldsymbol{Y}\right).$$

- Vanilla ED but considers reactive powers
  - Supply regularization $\mu_{\mathsf{p}}, \mu_{\mathsf{q}} \geq 0$
  - Adjust for non-demand consumption: $\operatorname{shunt}(\cdot)$ (§3.2.1) and $\operatorname{loss}(\cdot)$ (§3.2.2)
- Convex: checked with CVX 2.2 [21], has a convex QP form (§3.2.3)
  - ☞ Lots of well-established polynomial-time algorithms

Solving for the anticipated bus voltages

APF equations (PFE+)

$$\boldsymbol{\phi}(\boldsymbol{x};\boldsymbol{Y},\boldsymbol{d},\boldsymbol{c},\delta_{\hat{n}}) := \underbrace{\boldsymbol{e}(\boldsymbol{v},\boldsymbol{\vartheta};\boldsymbol{Y},\delta_{\hat{n}}) - \kappa \begin{bmatrix} \boldsymbol{C}_{\mathsf{u}}\boldsymbol{\kappa} \\ \boldsymbol{0}_N \end{bmatrix}}_{\boldsymbol{\psi}(\boldsymbol{x};\boldsymbol{Y},\delta_{\hat{n}})} - \begin{bmatrix} \boldsymbol{C}_{\mathsf{u}}\boldsymbol{p}_{\mathsf{u}} \\ \boldsymbol{C}_{\mathsf{u}}\boldsymbol{q}_{\mathsf{u}} \end{bmatrix} - \begin{bmatrix} \boldsymbol{C}_{\mathsf{d}}\boldsymbol{p}_{\mathsf{d}} \\ \boldsymbol{C}_{\mathsf{d}}\boldsymbol{q}_{\mathsf{d}} \end{bmatrix} = \boldsymbol{0}_{2N}$$

- A $2N$–dimensional root-finding task
  - ☞ Lots of derivative-based algorithms with fast convergence guarantees
  - See §3.3.2 for the APF Jacobian $\boldsymbol{J}(\boldsymbol{x}) := \partial_{\boldsymbol{x}}\boldsymbol{\psi}(\boldsymbol{x})$

Solving for the anticipated bus voltages

APF equations (PFE+)

$$\boldsymbol{\phi}(\boldsymbol{x};\boldsymbol{Y},\boldsymbol{d},\boldsymbol{c},\delta_{\hat{n}}) := \underbrace{\boldsymbol{e}(\boldsymbol{v},\boldsymbol{\vartheta};\boldsymbol{Y},\delta_{\hat{n}}) - \kappa \begin{bmatrix} \boldsymbol{C}_{\mathsf{u}}\boldsymbol{\kappa} \\ \boldsymbol{0}_N \end{bmatrix}}_{\boldsymbol{\psi}(\boldsymbol{x};\boldsymbol{Y},\delta_{\hat{n}})} - \begin{bmatrix} \boldsymbol{C}_{\mathsf{u}}\boldsymbol{p}_{\mathsf{u}} \\ \boldsymbol{C}_{\mathsf{u}}\boldsymbol{q}_{\mathsf{u}} \end{bmatrix} - \begin{bmatrix} \boldsymbol{C}_{\mathsf{d}}\boldsymbol{p}_{\mathsf{d}} \\ \boldsymbol{C}_{\mathsf{d}}\boldsymbol{q}_{\mathsf{d}} \end{bmatrix} = \boldsymbol{0}_{2N}$$

- A $2N$–dimensional root-finding task
  - ☞ Lots of derivative-based algorithms with fast convergence guarantees
    - See §3.3.2 for the APF Jacobian $\boldsymbol{J}(\boldsymbol{x}) := \partial_{\boldsymbol{x}}\boldsymbol{\psi}(\boldsymbol{x})$
- $\boldsymbol{c}$ is a parameter of PFE+
  - ☞ Different $\boldsymbol{c}$'s give different $\boldsymbol{x}$'s
  - ☞ Compare APF points by their κ's

Some fast solvers and algorithms for APF

For solving ED+

- SeDuMi 1.3.4 [22–24]
- SDPT3 4.0 [25–27]

☞ Free and open-source

☞ Shipped as part of CVX 2.2

Some fast solvers and algorithms for APF

For solving ED+

- SeDuMi 1.3.4 [22–24]
- SDPT3 4.0 [25–27]

☞ Free and open-source

☞ Shipped as part of CVX 2.2

For solving PFE+

- Powell hybrid method [28–30]
- Levenberg-Marquardt algorithm [31, 32]

☞ Quadratic local convergence [33, §10.3, 11.2]

☞ Ready-to-use in modern solvers

Some fast solvers and algorithms for APF

For solving ED+

- SeDuMi 1.3.4 [22–24]
- SDPT3 4.0 [25–27]

☞ Free and open-source

☞ Shipped as part of CVX 2.2

For solving PFE+

- Powell hybrid method [28–30]
- Levenberg-Marquardt algorithm [31, 32]

☞ Quadratic local convergence [33, §10.3, 11.2]

☞ Ready-to-use in modern solvers

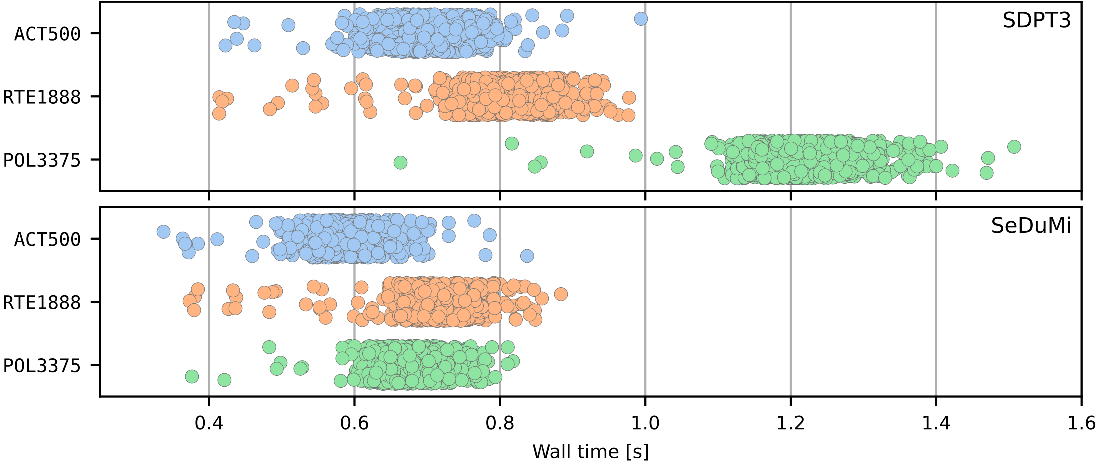
APFLib: a MATLAB library built on CVX and as an extension for MATPOWER

christian-cahig/Masterarbeit-DemoApps

Run time evaluations for solving ED+

Based on Intel Core i7-10750H CPU @ 2.60GHz w/ 16GB RAM

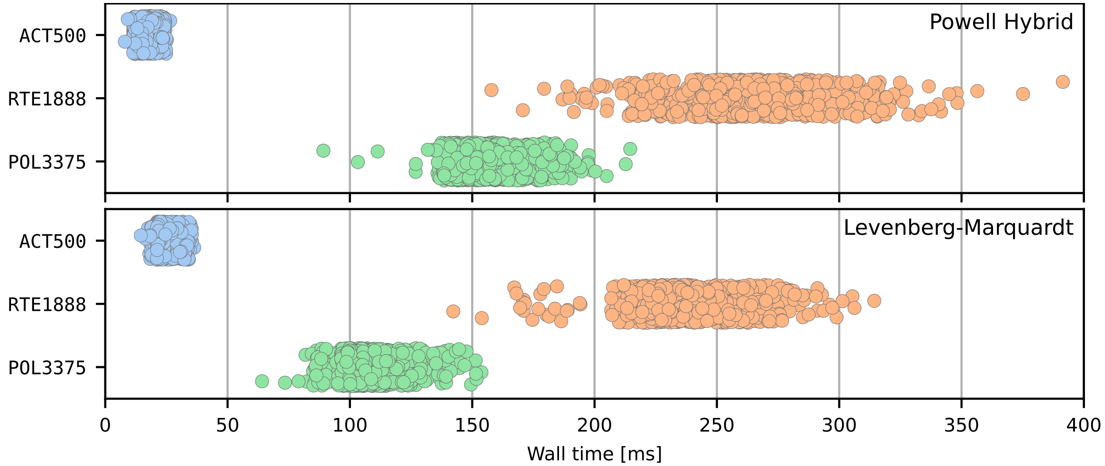
`ACT500`, `RTE1888`, and `POL3375` have 90, 298, and 596 supply units, respectively.



Run time evaluations for solving PFE+

Based on Intel Core i7-10750H CPU @ 2.60GHz w/ 16GB RAM

`ACT500`, `RTE1888`, and `POL3375` have 500, 1888, and 3374 buses, respectively.

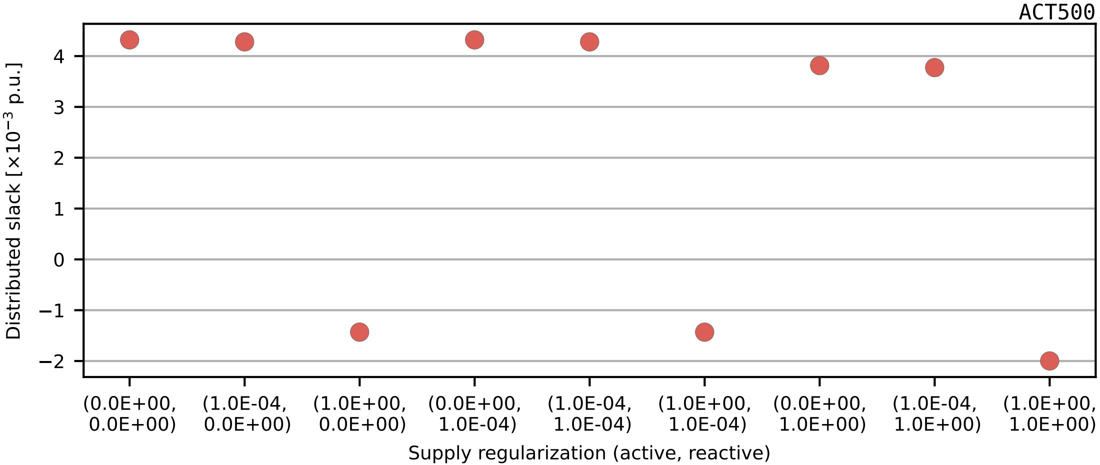


Effect of supply regularization on the APF point

All else fixed, what happens to $(\boldsymbol{c}, \boldsymbol{x})$ when $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) \in \left\{0, 10^{-4}, 1\right\} \times \left\{0, 10^{-4}, 1\right\}$?

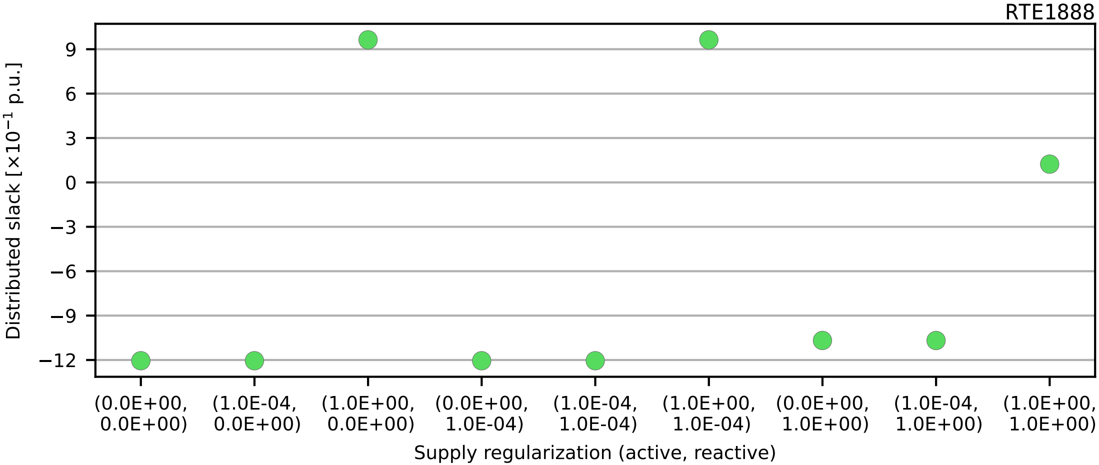
Effect of supply regularization on the APF point

All else fixed, what happens to $(\boldsymbol{c}, \boldsymbol{x})$ when $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) \in \{0, 10^{-4}, 1\} \times \{0, 10^{-4}, 1\}$?



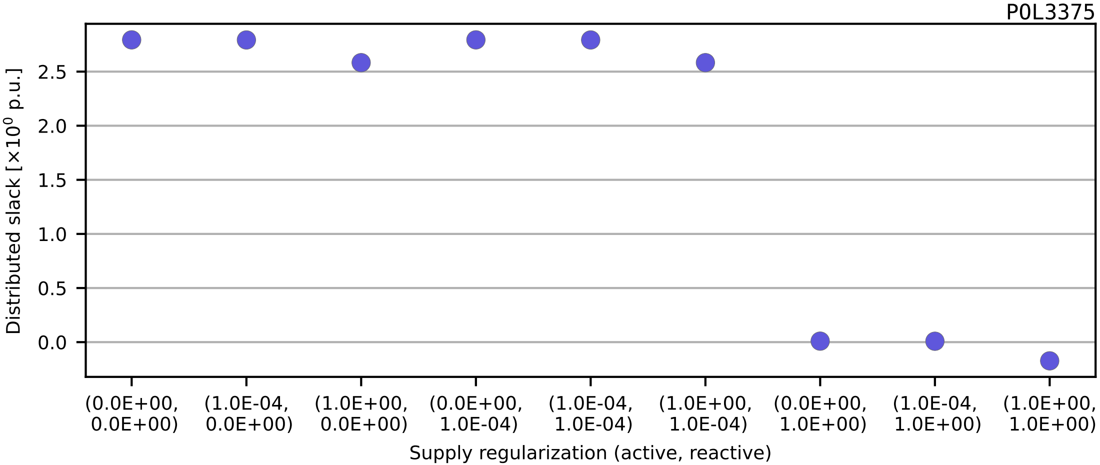
Effect of supply regularization on the APF point

All else fixed, what happens to $(\boldsymbol{c}, \boldsymbol{x})$ when $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) \in \{0, 10^{-4}, 1\} \times \{0, 10^{-4}, 1\}$?



Effect of supply regularization on the APF point

All else fixed, what happens to $(\boldsymbol{c}, \boldsymbol{x})$ when $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) \in \{0, 10^{-4}, 1\} \times \{0, 10^{-4}, 1\}$?



Effect of supply regularization on the APF point

Quadrimodal effect

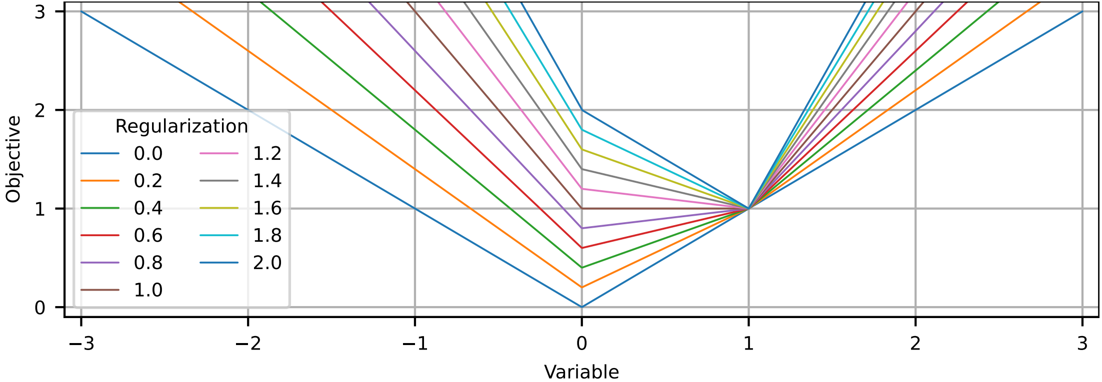
The APF point $(\boldsymbol{c}, \boldsymbol{x})$ will be in one of four neighbourhoods in the upcoming-dispatch PFM.

Effect of supply regularization on the APF point

Quadrimodal effect

The APF point $(\boldsymbol{c}, \boldsymbol{x})$ will be in one of four neighbourhoods in the upcoming-dispatch PFM.

Consider ED+ in 1D: minimize $|x| + \mu|x-1|$ s.t. $-3 \leq x \leq 3$, where $\mu \geq 0$



Effect of supply regularization on the APF point

Quadrimodal effect

The APF point $(\boldsymbol{c}, \boldsymbol{x})$ will be in one of four neighbourhoods in the upcoming-dispatch PFM.

Corollary (Big-μ trick for finding four APF points)

Regularizing ED+ with $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) \in \{0, \mu\} \times \{0, \mu\}$*, for some* $\mu \gg 0$*, yields four* $\boldsymbol{c}$*'s, and, by PFE+, four* $\boldsymbol{x}$*'s. These four independent APF instances can be run in parallel.*

APF for OPF: Providing solvers with warm-start points

OPF solvers are iterative: $(\boldsymbol{c}_{k+1}, \boldsymbol{s}_{k+1}) \leftarrow \text{update}(\boldsymbol{c}_k, \boldsymbol{s}_k)$

- User-specified starting point $(\boldsymbol{c}_0, \boldsymbol{s}_0)$
- Interior-point methods are SOTA [34], especially in large scale [35–37]

APF for OPF: Providing solvers with warm-start points

OPF solvers are iterative: $(\boldsymbol{c}_{k+1}, \boldsymbol{s}_{k+1}) \leftarrow \mathrm{update}(\boldsymbol{c}_k, \boldsymbol{s}_k)$

- User-specified starting point $(\boldsymbol{c}_0, \boldsymbol{s}_0)$
- Interior-point methods are SOTA [34], especially in large scale [35–37]

Snapshot-starting a solver

- $(\boldsymbol{c}_0, \boldsymbol{s}_0) \leftarrow (\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$

☞ Search does not start on PFM

APF for OPF: Providing solvers with warm-start points

OPF solvers are iterative: $(\boldsymbol{c}_{k+1}, \boldsymbol{s}_{k+1}) \leftarrow \mathrm{update}(\boldsymbol{c}_k, \boldsymbol{s}_k)$

- User-specified starting point $(\boldsymbol{c}_0, \boldsymbol{s}_0)$
- Interior-point methods are SOTA [34], especially in large scale [35–37]

Snapshot-starting a solver

- $(\boldsymbol{c}_0, \boldsymbol{s}_0) \leftarrow (\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$

☞ Search does not start on PFM

Warm-starting a solver with APF point $(\boldsymbol{c}, \boldsymbol{x})$

- $\boldsymbol{c}_0 \leftarrow (\boldsymbol{p}_{\mathsf{u}} + \kappa\boldsymbol{\kappa}, \boldsymbol{q}_{\mathsf{u}})$
- $\boldsymbol{s}_0 \leftarrow (\boldsymbol{v}, \boldsymbol{\delta})$

☞ Search starts on PFM

APF for OPF: Providing solvers with warm-start points

Solve instances on `POL3375` via MIPS 1.4 [38]

- 408 APF instances
- Get snapshot- & warm-started optima
  - $\boldsymbol{c}^\star_\mathsf{s}$, $\boldsymbol{v}^\star_\mathsf{s}$, $\boldsymbol{\varphi}^\star_\mathsf{s}$, $g^\star_\mathsf{s}$
  - $\boldsymbol{c}^\star_\mathsf{w}$, $\boldsymbol{v}^\star_\mathsf{w}$, $\boldsymbol{\varphi}^\star_\mathsf{w}$, $g^\star_\mathsf{w}$
- Compare solutions in terms of
  - $\epsilon_\mathsf{c} := \left\|\boldsymbol{c}^\star_\mathsf{s} - \boldsymbol{c}^\star_\mathsf{w}\right\|_\infty$ (in 100-MVA units)
  - $\epsilon_\mathsf{v} := \left\|\boldsymbol{v}^\star_\mathsf{s} - \boldsymbol{v}^\star_\mathsf{w}\right\|_\infty$ (in 400-kV units)
  - $\epsilon_\mathsf{a} := \left\|\boldsymbol{\varphi}^\star_\mathsf{s} - \boldsymbol{\varphi}^\star_\mathsf{w}\right\|_\infty$ (in radians)
  - $\epsilon_\mathsf{g} := g^\star_\mathsf{s} - g^\star_\mathsf{w}$ (in USD)

APF for OPF: Providing solvers with warm-start points

Solve instances on `POL3375` via MIPS 1.4 [38]

- 408 APF instances
- Get snapshot- & warm-started optima
  - $\boldsymbol{c}_{\mathsf{s}}^{\star}$, $\boldsymbol{v}_{\mathsf{s}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{s}}^{\star}$, $g_{\mathsf{s}}^{\star}$
  - $\boldsymbol{c}_{\mathsf{w}}^{\star}$, $\boldsymbol{v}_{\mathsf{w}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{w}}^{\star}$, $g_{\mathsf{w}}^{\star}$
- Compare solutions in terms of
  - $\epsilon_{\mathsf{c}} := \left\| \boldsymbol{c}_{\mathsf{s}}^{\star} - \boldsymbol{c}_{\mathsf{w}}^{\star} \right\|_{\infty}$ (in 100-MVA units)
  - $\epsilon_{\mathsf{v}} := \left\| \boldsymbol{v}_{\mathsf{s}}^{\star} - \boldsymbol{v}_{\mathsf{w}}^{\star} \right\|_{\infty}$ (in 400-kV units)
  - $\epsilon_{\mathsf{a}} := \left\| \boldsymbol{\varphi}_{\mathsf{s}}^{\star} - \boldsymbol{\varphi}_{\mathsf{w}}^{\star} \right\|_{\infty}$ (in radians)
  - $\epsilon_{\mathsf{g}} := g_{\mathsf{s}}^{\star} - g_{\mathsf{w}}^{\star}$ (in USD)

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (0, 0)$

Metric	Minimum	Maximum
ϵ_{g}	-1.7847×10^{-2}	1.90488×10^{-2}
ϵ_{v}	4.79369×10^{-9}	6.89567×10^{-5}
ϵ_{a}	6.47442×10^{-10}	5.92979×10^{-6}
ϵ_{c}	6.05418×10^{-4}	8.78352×10^{-1}

APF for OPF: Providing solvers with warm-start points

Solve instances on `POL3375` via MIPS 1.4 [38]

- 408 APF instances
- Get snapshot- & warm-started optima
  - $\boldsymbol{c}_{\mathsf{s}}^{\star}$, $\boldsymbol{v}_{\mathsf{s}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{s}}^{\star}$, $g_{\mathsf{s}}^{\star}$
  - $\boldsymbol{c}_{\mathsf{w}}^{\star}$, $\boldsymbol{v}_{\mathsf{w}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{w}}^{\star}$, $g_{\mathsf{w}}^{\star}$
- Compare solutions in terms of
  - $\epsilon_{\mathsf{c}} := \left\| \boldsymbol{c}_{\mathsf{s}}^{\star} - \boldsymbol{c}_{\mathsf{w}}^{\star} \right\|_{\infty}$ (in 100-MVA units)
  - $\epsilon_{\mathsf{v}} := \left\| \boldsymbol{v}_{\mathsf{s}}^{\star} - \boldsymbol{v}_{\mathsf{w}}^{\star} \right\|_{\infty}$ (in 400-kV units)
  - $\epsilon_{\mathsf{a}} := \left\| \boldsymbol{\varphi}_{\mathsf{s}}^{\star} - \boldsymbol{\varphi}_{\mathsf{w}}^{\star} \right\|_{\infty}$ (in radians)
  - $\epsilon_{\mathsf{g}} := g_{\mathsf{s}}^{\star} - g_{\mathsf{w}}^{\star}$ (in USD)

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (0, 0)$

Metric	Minimum	Maximum
ϵ_{g}	-1.7847×10^{-2}	1.90488×10^{-2}
ϵ_{v}	4.79369×10^{-9}	6.89567×10^{-5}
ϵ_{a}	6.47442×10^{-10}	5.92979×10^{-6}
ϵ_{c}	6.05418×10^{-4}	8.78352×10^{-1}

APF for OPF: Providing solvers with warm-start points

Solve instances on POL3375 via MIPS 1.4 [38]

- 408 APF instances
- Get snapshot- & warm-started optima
  - $\boldsymbol{c}_{\mathsf{s}}^{\star}$, $\boldsymbol{v}_{\mathsf{s}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{s}}^{\star}$, $g_{\mathsf{s}}^{\star}$
  - $\boldsymbol{c}_{\mathsf{w}}^{\star}$, $\boldsymbol{v}_{\mathsf{w}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{w}}^{\star}$, $g_{\mathsf{w}}^{\star}$
- Compare solutions in terms of
  - $\epsilon_{\mathsf{c}} := \left\|\boldsymbol{c}_{\mathsf{s}}^{\star} - \boldsymbol{c}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in 100-MVA units)
  - $\epsilon_{\mathsf{v}} := \left\|\boldsymbol{v}_{\mathsf{s}}^{\star} - \boldsymbol{v}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in 400-kV units)
  - $\epsilon_{\mathsf{a}} := \left\|\boldsymbol{\varphi}_{\mathsf{s}}^{\star} - \boldsymbol{\varphi}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in radians)
  - $\epsilon_{\mathsf{g}} := g_{\mathsf{s}}^{\star} - g_{\mathsf{w}}^{\star}$ (in USD)

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (0, 0)$

Metric	Minimum	Maximum
ϵ_{g}	-1.7847×10^{-2}	1.90488×10^{-2}
ϵ_{v}	4.79369×10^{-9}	6.89567×10^{-5}
ϵ_{a}	6.47442×10^{-10}	5.92979×10^{-6}
ϵ_{c}	6.05418×10^{-4}	8.78352×10^{-1}

APF for OPF: Providing solvers with warm-start points

Solve instances on `POL3375` via MIPS 1.4 [38]

- 408 APF instances
- Get snapshot- & warm-started optima
  - $\boldsymbol{c}^\star_{\mathsf{s}}$, $\boldsymbol{v}^\star_{\mathsf{s}}$, $\boldsymbol{\varphi}^\star_{\mathsf{s}}$, $g^\star_{\mathsf{s}}$
  - $\boldsymbol{c}^\star_{\mathsf{w}}$, $\boldsymbol{v}^\star_{\mathsf{w}}$, $\boldsymbol{\varphi}^\star_{\mathsf{w}}$, $g^\star_{\mathsf{w}}$
- Compare solutions in terms of
  - $\epsilon_{\mathsf{c}} := \left\| \boldsymbol{c}^\star_{\mathsf{s}} - \boldsymbol{c}^\star_{\mathsf{w}} \right\|_\infty$ (in 100-MVA units)
  - $\epsilon_{\mathsf{v}} := \left\| \boldsymbol{v}^\star_{\mathsf{s}} - \boldsymbol{v}^\star_{\mathsf{w}} \right\|_\infty$ (in 400-kV units)
  - $\epsilon_{\mathsf{a}} := \left\| \boldsymbol{\varphi}^\star_{\mathsf{s}} - \boldsymbol{\varphi}^\star_{\mathsf{w}} \right\|_\infty$ (in radians)
  - $\epsilon_{\mathsf{g}} := g^\star_{\mathsf{s}} - g^\star_{\mathsf{w}}$ (in USD)

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (0, 0)$

Metric	Minimum	Maximum
ϵ_{g}	-1.7847×10^{-2}	1.90488×10^{-2}
ϵ_{v}	4.79369×10^{-9}	6.89567×10^{-5}
ϵ_{a}	6.47442×10^{-10}	5.92979×10^{-6}
ϵ_{c}	6.05418×10^{-4}	8.78352×10^{-1}

APF for OPF: Providing solvers with warm-start points

Solve instances on POL3375 via MIPS 1.4 [38]

- 408 APF instances
- Get snapshot- & warm-started optima
  - $\boldsymbol{c}_{\mathsf{s}}^{\star}$, $\boldsymbol{v}_{\mathsf{s}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{s}}^{\star}$, $g_{\mathsf{s}}^{\star}$
  - $\boldsymbol{c}_{\mathsf{w}}^{\star}$, $\boldsymbol{v}_{\mathsf{w}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{w}}^{\star}$, $g_{\mathsf{w}}^{\star}$
- Compare solutions in terms of
  - $\epsilon_{\mathsf{c}} := \left\|\boldsymbol{c}_{\mathsf{s}}^{\star} - \boldsymbol{c}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in 100-MVA units)
  - $\epsilon_{\mathsf{v}} := \left\|\boldsymbol{v}_{\mathsf{s}}^{\star} - \boldsymbol{v}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in 400-kV units)
  - $\epsilon_{\mathsf{a}} := \left\|\boldsymbol{\varphi}_{\mathsf{s}}^{\star} - \boldsymbol{\varphi}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in radians)
  - $\epsilon_{\mathsf{g}} := g_{\mathsf{s}}^{\star} - g_{\mathsf{w}}^{\star}$ (in USD)

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (0, 0)$

Metric	Minimum	Maximum
ϵ_{g}	-1.7847×10^{-2}	1.90488×10^{-2}
ϵ_{v}	4.79369×10^{-9}	6.89567×10^{-5}
ϵ_{a}	6.47442×10^{-10}	5.92979×10^{-6}
ϵ_{c}	6.05418×10^{-4}	8.78352×10^{-1}

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (1, 1)$

Metric	Minimum	Maximum
ϵ_{g}	-1.21094×10^{-2}	2.14678×10^{-2}
ϵ_{v}	1.82573×10^{-9}	2.4873×10^{-5}
ϵ_{a}	2.88286×10^{-10}	3.47572×10^{-6}
ϵ_{c}	1.23703×10^{-4}	2.88003×10^{-1}

APF for OPF: Providing solvers with warm-start points

Solve instances on `POL3375` via MIPS 1.4 [38]

- 408 APF instances
- Get snapshot- & warm-started optima
  - $\boldsymbol{c}_{\mathsf{s}}^{\star}$, $\boldsymbol{v}_{\mathsf{s}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{s}}^{\star}$, $g_{\mathsf{s}}^{\star}$
  - $\boldsymbol{c}_{\mathsf{w}}^{\star}$, $\boldsymbol{v}_{\mathsf{w}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{w}}^{\star}$, $g_{\mathsf{w}}^{\star}$
- Compare solutions in terms of
  - $\epsilon_{\mathsf{c}} := \left\|\boldsymbol{c}_{\mathsf{s}}^{\star} - \boldsymbol{c}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in 100-MVA units)
  - $\epsilon_{\mathsf{v}} := \left\|\boldsymbol{v}_{\mathsf{s}}^{\star} - \boldsymbol{v}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in 400-kV units)
  - $\epsilon_{\mathsf{a}} := \left\|\boldsymbol{\varphi}_{\mathsf{s}}^{\star} - \boldsymbol{\varphi}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in radians)
  - $\epsilon_{\mathsf{g}} := g_{\mathsf{s}}^{\star} - g_{\mathsf{w}}^{\star}$ (in USD)

☞ $(g_{\mathsf{w}}^{\star}, \boldsymbol{v}_{\mathsf{w}}^{\star}, \boldsymbol{\varphi}_{\mathsf{w}}^{\star}) \approxeq (g_{\mathsf{s}}^{\star}, \boldsymbol{v}_{\mathsf{s}}^{\star}, \boldsymbol{\varphi}_{\mathsf{s}}^{\star})$ attained from distinct $\boldsymbol{c}_{\mathsf{w}}^{\star}$ and $\boldsymbol{c}_{\mathsf{s}}^{\star}$

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (0, 0)$

Metric	Minimum	Maximum
ϵ_{g}	-1.7847×10^{-2}	1.90488×10^{-2}
ϵ_{v}	4.79369×10^{-9}	6.89567×10^{-5}
ϵ_{a}	6.47442×10^{-10}	5.92979×10^{-6}
ϵ_{c}	6.05418×10^{-4}	8.78352×10^{-1}

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (1, 1)$

Metric	Minimum	Maximum
ϵ_{g}	-1.21094×10^{-2}	2.14678×10^{-2}
ϵ_{v}	1.82573×10^{-9}	2.4873×10^{-5}
ϵ_{a}	2.88286×10^{-10}	3.47572×10^{-6}
ϵ_{c}	1.23703×10^{-4}	2.88003×10^{-1}

APF for OPF: Providing solvers with warm-start points

Solve instances on POL3375 via MIPS 1.4 [38]

- 408 APF instances
- Get snapshot- & warm-started optima
  - $\boldsymbol{c}_{\mathsf{s}}^{\star}$, $\boldsymbol{v}_{\mathsf{s}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{s}}^{\star}$, $g_{\mathsf{s}}^{\star}$
  - $\boldsymbol{c}_{\mathsf{w}}^{\star}$, $\boldsymbol{v}_{\mathsf{w}}^{\star}$, $\boldsymbol{\varphi}_{\mathsf{w}}^{\star}$, $g_{\mathsf{w}}^{\star}$
- Compare solutions in terms of
  - $\epsilon_{\mathsf{c}} := \left\|\boldsymbol{c}_{\mathsf{s}}^{\star} - \boldsymbol{c}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in 100-MVA units)
  - $\epsilon_{\mathsf{v}} := \left\|\boldsymbol{v}_{\mathsf{s}}^{\star} - \boldsymbol{v}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in 400-kV units)
  - $\epsilon_{\mathsf{a}} := \left\|\boldsymbol{\varphi}_{\mathsf{s}}^{\star} - \boldsymbol{\varphi}_{\mathsf{w}}^{\star}\right\|_{\infty}$ (in radians)
  - $\epsilon_{\mathsf{g}} := g_{\mathsf{s}}^{\star} - g_{\mathsf{w}}^{\star}$ (in USD)

☞ $(g_{\mathsf{w}}^{\star}, \boldsymbol{v}_{\mathsf{w}}^{\star}, \boldsymbol{\varphi}_{\mathsf{w}}^{\star}) \approxeq (g_{\mathsf{s}}^{\star}, \boldsymbol{v}_{\mathsf{s}}^{\star}, \boldsymbol{\varphi}_{\mathsf{s}}^{\star})$ attained from distinct $\boldsymbol{c}_{\mathsf{w}}^{\star}$ and $\boldsymbol{c}_{\mathsf{s}}^{\star}$

At $(\mu_{\mathsf{p}}, \mu_{\mathsf{q}}) = (1, 0)$

Metric	Minimum	Maximum
ϵ_{g}	-2.44578×10^{-2}	2.24346×10^{-2}
ϵ_{v}	7.57086×10^{-10}	6.40276×10^{-5}
ϵ_{a}	1.99231×10^{-10}	4.99960×10^{-6}
ϵ_{c}	6.91804×10^{-4}	1.16527×10^{0}

APF for OPF: Providing solvers with warm-start points

It's just the nonconvexity of OPF

The APF point can be sufficiently far from the snapshot point that, for the same algorithm, these starting points lead to distinct optima.

APF for OPF: Providing solvers with warm-start points

It's just the nonconvexity of OPF

The APF point can be sufficiently far from the snapshot point that, for the same algorithm, these starting points lead to distinct optima.

Corollary

In its current form, APF is a crude method for finding multiple OPF solutions.

APF for OPF: Providing solvers with warm-start points

It's just the nonconvexity of OPF

The APF point can be sufficiently far from the snapshot point that, for the same algorithm, these starting points lead to distinct optima.

Corollary

In its current form, APF is a crude method for finding multiple OPF solutions.

Open problems

1. Given a solver, a snapshot point S, and a APF point A, how can we tell that starting the solver at S and at A will or will not give us distinct optima?
2. How to make APF a more disciplined method of finding multiple OPF solutions?

APF for amortized OPF: Differentiating through PFE+

Neural nets to learn solution maps of OPF instances with fixed $\boldsymbol{Y}$ but varying $\boldsymbol{d}$

- Design challenge: differentiably incorporate PFE

APF for amortized OPF: Differentiating through PFE+

Neural nets to learn solution maps of OPF instances with fixed $\boldsymbol{Y}$ but varying $\boldsymbol{d}$

- Design challenge: differentiably incorporate PFE
- OPF-DNN: violation-based Lagrangian relaxation [39]
  - $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c}, \boldsymbol{s} \xrightarrow{\text{computations}} \ell = \cdots + \lambda \left\| \phi(\boldsymbol{c}, \boldsymbol{s}) \right\|_1$
  - ! Only encourages PFE compliance

APF for amortized OPF: Differentiating through PFE+

Neural nets to learn solution maps of OPF instances with fixed $\boldsymbol{Y}$ but varying $\boldsymbol{d}$

- Design challenge: differentiably incorporate PFE
- OPF-DNN: violation-based Lagrangian relaxation [39]
  - $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c}, \boldsymbol{s} \xrightarrow{\text{computations}} \ell = \cdots + \lambda \left\| \phi(\boldsymbol{c}, \boldsymbol{s}) \right\|_1$
  - ! Only encourages PFE compliance
- DeepOPF: SPF + zeroth-order gradient estimation [40]
  - $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \hat{\boldsymbol{p}}_{\mathrm{u}}, \hat{\boldsymbol{v}} \xrightarrow{\text{SPF}} \boldsymbol{c}, \boldsymbol{s} \xrightarrow{\text{computations}} \ell$
  - ! Inexact gradient could hurt training

APF for amortized OPF: Differentiating through PFE+

Neural nets to learn solution maps of OPF instances with fixed $\boldsymbol{Y}$ but varying $\boldsymbol{d}$

- Design challenge: differentiably incorporate PFE
- OPF-DNN: violation-based Lagrangian relaxation [39]
  - $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c}, \boldsymbol{s} \xrightarrow{\text{computations}} \ell = \cdots + \lambda \left\| \boldsymbol{\phi}(\boldsymbol{c}, \boldsymbol{s}) \right\|_1$
  - ! Only encourages PFE compliance
- DeepOPF: SPF + zeroth-order gradient estimation [40]
  - $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \hat{\boldsymbol{p}}_{\mathrm{u}}, \hat{\boldsymbol{v}} \xrightarrow{\text{SPF}} \boldsymbol{c}, \boldsymbol{s} \xrightarrow{\text{computations}} \ell$
  - ! Inexact gradient could hurt training
- DC3: SPF + penalty + differentiating through SPF [41]
  - $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \hat{\boldsymbol{p}}_{\mathrm{u}}, \hat{\boldsymbol{v}} \xrightarrow{\text{SPF}} \boldsymbol{c}, \boldsymbol{s} \xrightarrow{\text{computations}} \ell = \cdots + \lambda \left\| \boldsymbol{\phi}(\boldsymbol{c}, \boldsymbol{s}) \right\|_2^2$
  - ! Differentiating through SPF is (very) complicated

APF for amortized OPF: Differentiating through PFE+

Long-term mission

Treating $\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)$ as anticipating $\boldsymbol{c}$, we have $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c} \xrightarrow{\text{solve PFE+}} \boldsymbol{x} \xrightarrow{\text{computations}} \ell(\boldsymbol{c}, \boldsymbol{x}(\boldsymbol{c}))$ for any appropriate loss ℓ. Backpropagation simply follows $\mathrm{d}\ell = \big(\partial_{\boldsymbol{c}}\ell + \partial_{\boldsymbol{x}}\ell\, \partial_{\boldsymbol{c}}\boldsymbol{x}\big)\partial_{\boldsymbol{\theta}}\boldsymbol{c}\, \mathrm{d}\boldsymbol{\theta}$.

APF for amortized OPF: Differentiating through PFE+

Long-term mission

Treating $\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)$ as anticipating $\boldsymbol{c}$, we have $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c} \xrightarrow{\text{solve PFE+}} \boldsymbol{x} \xrightarrow{\text{computations}} \ell(\boldsymbol{c}, \boldsymbol{x}(\boldsymbol{c}))$ for any appropriate loss ℓ. Backpropagation simply follows $\mathrm{d}\ell = \big(\partial_{\boldsymbol{c}}\ell + \partial_{\boldsymbol{x}}\ell\, \partial_{\boldsymbol{c}}\boldsymbol{x}\big)\partial_{\boldsymbol{\theta}}\boldsymbol{c}\, \mathrm{d}\boldsymbol{\theta}$.

- $\partial_{\boldsymbol{\theta}}\boldsymbol{c}$, $\partial_{\boldsymbol{c}}\ell(\cdot)$, and $\partial_{\boldsymbol{x}}\ell(\cdot)$ are trivial for modern autodiff engines

APF for amortized OPF: Differentiating through PFE+

Long-term mission

Treating $\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)$ as anticipating $\boldsymbol{c}$, we have $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c} \xrightarrow{\text{solve PFE+}} \boldsymbol{x} \xrightarrow{\text{computations}} \ell(\boldsymbol{c}, \boldsymbol{x}(\boldsymbol{c}))$ for any appropriate loss ℓ. Backpropagation simply follows $\mathrm{d}\ell = \big(\partial_{\boldsymbol{c}}\ell + \partial_{\boldsymbol{x}}\ell\, \partial_{\boldsymbol{c}}\boldsymbol{x}\big)\partial_{\boldsymbol{\theta}}\boldsymbol{c}\,\mathrm{d}\boldsymbol{\theta}$.

- $\partial_{\boldsymbol{\theta}}\boldsymbol{c}$, $\partial_{\boldsymbol{c}}\ell(\cdot)$, and $\partial_{\boldsymbol{x}}\ell(\cdot)$ are trivial for modern autodiff engines

☞ Contribution: How to differentiate through PFE+
  - Computing the backward APF Jacobian $\boldsymbol{H}(\cdot) := \partial_{\boldsymbol{c}}\boldsymbol{x}(\cdot)$
  - Computing the backward APF gradient $\boldsymbol{g}(\cdot)$, *i.e.*, $\boldsymbol{g}^{\mathsf{T}}(\cdot) \equiv \partial_{\boldsymbol{x}}\ell(\cdot)\,\boldsymbol{H}(\cdot)$

APF for amortized OPF: Differentiating through PFE+

Long-term mission

Treating $\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)$ as anticipating $\boldsymbol{c}$, we have $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c} \xrightarrow{\text{solve PFE+}} \boldsymbol{x} \xrightarrow{\text{computations}} \ell(\boldsymbol{c}, \boldsymbol{x}(\boldsymbol{c}))$ for any appropriate loss ℓ. Backpropagation simply follows $\mathrm{d}\ell = \big(\partial_{\boldsymbol{c}}\ell + \partial_{\boldsymbol{x}}\ell\, \partial_{\boldsymbol{c}}\boldsymbol{x}\big)\partial_{\boldsymbol{\theta}}\boldsymbol{c}\,\mathrm{d}\boldsymbol{\theta}$.

Computing the backward APF Jacobian (§4.4.1)

Applying the implicit function theorem at an APF point $(\boldsymbol{c}, \boldsymbol{x})$, the backward APF Jacobian is the solution to $\boldsymbol{J}(\boldsymbol{x})\,\boldsymbol{H}(\boldsymbol{x}) = \mathrm{Diag}(\boldsymbol{C}_{\mathrm{u}}, \boldsymbol{C}_{\mathrm{u}})$.

APF for amortized OPF: Differentiating through PFE+

Long-term mission

Treating $\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)$ as anticipating $\boldsymbol{c}$, we have $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c} \xrightarrow{\text{solve PFE+}} \boldsymbol{x} \xrightarrow{\text{computations}} \ell(\boldsymbol{c}, \boldsymbol{x}(\boldsymbol{c}))$ for any appropriate loss ℓ. Backpropagation simply follows $\mathrm{d}\ell = \big(\partial_{\boldsymbol{c}}\ell + \partial_{\boldsymbol{x}}\ell\, \partial_{\boldsymbol{c}}\boldsymbol{x}\big)\partial_{\boldsymbol{\theta}}\boldsymbol{c}\,\mathrm{d}\boldsymbol{\theta}$.

Computing the backward APF Jacobian (§4.4.1)

Applying the implicit function theorem at an APF point $(\boldsymbol{c}, \boldsymbol{x})$, the backward APF Jacobian is the solution to $\boldsymbol{J}(\boldsymbol{x})\, \boldsymbol{H}(\boldsymbol{x}) = \mathrm{Diag}(\boldsymbol{C}_{\mathrm{u}}, \boldsymbol{C}_{\mathrm{u}})$.

Computing the backward APF gradient (§4.4.2)

Jacobian-vector product (JVP): solve for $\boldsymbol{H}$, then $\boldsymbol{g} = \boldsymbol{H}^{\mathsf{T}}\, \nabla_{\boldsymbol{x}}\ell$

Vector-Jacobian product (VJP): solve for $\boldsymbol{u}$ from $\boldsymbol{J}^{\mathsf{T}}\boldsymbol{u} = \nabla_{\boldsymbol{x}}\ell$, then $\boldsymbol{g} = \mathrm{Diag}\big(\boldsymbol{C}_{\mathrm{u}}^{\mathsf{T}}, \boldsymbol{C}_{\mathrm{u}}^{\mathsf{T}}\big)\, \boldsymbol{u}$

APF for amortized OPF: Differentiating through PFE+

Prefer VJP to JVP

APF for amortized OPF: Differentiating through PFE+

Prefer VJP to JVP

Gradients are of ℓ in Equation (4.24). Wall times are averaged over 100 runs, based on Intel Core i7-10750H w/ 16GB RAM.

System	$\left\lVert \boldsymbol{g}_{\mathsf{jvp}} - \boldsymbol{g}_{\mathsf{vjp}} \right\rVert_{\infty}$	JVP time [s]	VJP time [s]
ACT500	3.9968×10^{-14}	3.3319×10^{-2}	4.4259×10^{-3}
RTE1888	7.3630×10^{-13}	5.4983×10^{-1}	1.8315×10^{-2}
POL3375	5.5111×10^{-12}	2.4872	3.4472×10^{-2}

APF for amortized OPF: Differentiating through PFE+

Prefer VJP to JVP

Gradients are of ℓ in Equation (4.24). Wall times are averaged over 100 runs, based on Intel Core i7-10750H w/ 16GB RAM.

System	$\left\lVert \boldsymbol{g}_{\text{jvp}} - \boldsymbol{g}_{\text{vjp}} \right\rVert_\infty$	JVP time [s]	VJP time [s]
ACT500	3.9968×10^{-14}	3.3319×10^{-2}	4.4259×10^{-3}
RTE1888	7.3630×10^{-13}	5.4983×10^{-1}	1.8315×10^{-2}
POL3375	5.5111×10^{-12}	2.4872	3.4472×10^{-2}

APF for amortized OPF: Differentiating through PFE+

Prefer VJP to JVP

Gradients are of ℓ in Equation (4.24). Wall times are averaged over 100 runs, based on Intel Core i7-10750H w/ 16GB RAM.

System	$\lVert \boldsymbol{g}_{\text{jvp}} - \boldsymbol{g}_{\text{vjp}} \rVert_\infty$	JVP time [s]	VJP time [s]
`ACT500`	3.9968×10^{-14}	3.3319×10^{-2}	4.4259×10^{-3}
`RTE1888`	7.3630×10^{-13}	5.4983×10^{-1}	1.8315×10^{-2}
`POL3375`	5.5111×10^{-12}	2.4872	3.4472×10^{-2}

APF for amortized OPF: Differentiating through PFE+

Prefer VJP to JVP

- No need to form $\boldsymbol{H}$, which is $2N \times 2U$ and dense

Gradients are of ℓ in Equation (4.24). Wall times are averaged over 100 runs, based on Intel Core i7-10750H w/ 16GB RAM.

System	$\lVert \boldsymbol{g}_{\text{jvp}} - \boldsymbol{g}_{\text{vjp}} \rVert_\infty$	JVP time [s]	VJP time [s]
ACT500	3.9968×10^{-14}	3.3319×10^{-2}	4.4259×10^{-3}
RTE1888	7.3630×10^{-13}	5.4983×10^{-1}	1.8315×10^{-2}
POL3375	5.5111×10^{-12}	2.4872	3.4472×10^{-2}

APF for amortized OPF: Differentiating through PFE+

Prefer VJP to JVP

- No need to form $\boldsymbol{H}$, which is $2N \times 2U$ and dense
- With mild assumptions (§A.3.2),
  - JVP is $\mathcal{O}\left(\frac{32}{2} U N^3\right)$ flops
  - VJP is $\mathcal{O}\left(\frac{16}{3} N^3\right)$ flops

Gradients are of ℓ in Equation (4.24). Wall times are averaged over 100 runs, based on Intel Core i7-10750H w/ 16GB RAM.

System	$\left\Vert \boldsymbol{g}_{\text{jvp}} - \boldsymbol{g}_{\text{vjp}} \right\Vert_\infty$	JVP time [s]	VJP time [s]
ACT500	3.9968×10^{-14}	3.3319×10^{-2}	4.4259×10^{-3}
RTE1888	7.3630×10^{-13}	5.4983×10^{-1}	1.8315×10^{-2}
POL3375	5.5111×10^{-12}	2.4872	3.4472×10^{-2}

APF for amortized OPF: Differentiating through PFE+

Long-term mission

Treating $\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)$ as anticipating $\boldsymbol{c}$, we have $\boldsymbol{d} \xrightarrow{\mathrm{Net}_{\boldsymbol{\theta}}(\cdot)} \boldsymbol{c} \xrightarrow{\text{solve PFE+}} \boldsymbol{x} \xrightarrow{\text{computations}} \ell(\boldsymbol{c}, \boldsymbol{x}(\boldsymbol{c}))$ for any appropriate loss ℓ. Backpropagation simply follows $\mathrm{d}\ell = \big(\partial_{\boldsymbol{c}}\ell + \partial_{\boldsymbol{x}}\ell\, \partial_{\boldsymbol{c}}\boldsymbol{x}\big)\partial_{\boldsymbol{\theta}}\boldsymbol{c}\,\mathrm{d}\boldsymbol{\theta}$.

Open problems and working ideas

1. How to solve a batch of PFE+ instances with GPU acceleration?
   - Cast as nonlinear least-squares, then use JAXOpt [42]
2. How to quantify model uncertainty?
   - Extend conformal prediction [43] into a multivariate regression case

APF in summary

- Formulation: finding power-flow feasible $(\boldsymbol{c}, \boldsymbol{s})$ for anticipated grid conditions $(\boldsymbol{Y}, \boldsymbol{d})$, using preceding snapshot values $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$
  - Anticipate $\boldsymbol{c}$ by solving a convex program extended economic dispatch (ED+)
  - Compute corresponding $\boldsymbol{s}$ by solving APF equations (PFE+)
- Computation: amply handled by existing and readily available tools
  - SeDuMi and SDPT3 for ED+; Levenberg-Marquardt and Powell hybrid for PFE+
  - Sub-second run times on 3374-bus, 4161-branch, 596-generator portion of Polish grid
  - Quadrimodal effect of ED+ $\Longrightarrow$ big-μ trick for easily finding four APF points
- Applications
  1. Warm-starting OPF solvers $\Longrightarrow$ a crude method for finding multiple OPF solutions
  2. Differentiating through PFE+ $\Longrightarrow$ power flow equations as a layer in amortized OPF

APF in summary

- Formulation: finding power-flow feasible $(\boldsymbol{c}, \boldsymbol{s})$ for anticipated grid conditions $(\boldsymbol{Y}, \boldsymbol{d})$, using preceding snapshot values $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$
  - Anticipate $\boldsymbol{c}$ by solving a convex program extended economic dispatch (ED+)
  - Compute corresponding $\boldsymbol{s}$ by solving APF equations (PFE+)

- Computation: amply handled by existing and readily available tools
  - SeDuMi and SDPT3 for ED+; Levenberg-Marquardt and Powell hybrid for PFE+
  - Sub-second run times on 3374-bus, 4161-branch, 596-generator portion of Polish grid
  - Quadrimodal effect of ED+ $\Longrightarrow$ big-μ trick for easily finding four APF points

- Applications
  1. Warm-starting OPF solvers $\Longrightarrow$ a crude method for finding multiple OPF solutions
  2. Differentiating through PFE+ $\Longrightarrow$ power flow equations as a layer in amortized OPF

APF in summary

- Formulation: finding power-flow feasible $(\boldsymbol{c}, \boldsymbol{s})$ for anticipated grid conditions $(\boldsymbol{Y}, \boldsymbol{d})$, using preceding snapshot values $(\widetilde{\boldsymbol{c}}, \widetilde{\boldsymbol{s}})$
  - Anticipate $\boldsymbol{c}$ by solving a convex program extended economic dispatch (ED+)
  - Compute corresponding $\boldsymbol{s}$ by solving APF equations (PFE+)
- Computation: amply handled by existing and readily available tools
  - SeDuMi and SDPT3 for ED+; Levenberg-Marquardt and Powell hybrid for PFE+
  - Sub-second run times on 3374-bus, 4161-branch, 596-generator portion of Polish grid
  - Quadrimodal effect of ED+ $\implies$ big-μ trick for easily finding four APF points
- Applications
  1. Warm-starting OPF solvers $\implies$ a crude method for finding multiple OPF solutions
  2. Differentiating through PFE+ $\implies$ power flow equations as a layer in amortized OPF

Anticipatory Power Flow

Formulation, Computation, and Applications

Christian Y. Cahig

Mindanao State University – Iligan Institute of Technology

christian.cahig@outlook.com   christian-cahig

References I

[1] V. Ajjarapu and C. Christy. "The Continuation Power Flow: A Tool for Steady State Voltage Stability Analysis". In: *IEEE Transactions on Power Systems* 7.1 (Feb. 1992), pp. 416–423. DOI: 10.1109/59.141737 (cit. on pp. 13–15).

[2] C. A. Cañizares and F. L. Alvarado. "Point of collapse and continuation methods for large AC/DC systems". In: *IEEE Transactions on Power Systems* 8.1 (Feb. 1993), pp. 1–8. DOI: 10.1109/59.221241 (cit. on pp. 13–15).

[3] H.-D. Chiang et al. "CPFLOW: A Practical Tool for Tracing Power System Steady-State Stationary Behavior Due to Load and Generation Variations". In: *IEEE Transactions on Power Systems* 10.2 (May 1, 1995), pp. 623–634. DOI: 10.1109/59.387897 (cit. on pp. 13–15).

References II

[4] C. Gómez-Quiles, A. Gómez-Expósito, and W. Vargas. "Computation of Maximum Loading Points via the Factored Load Flow". In: *IEEE Transactions on Power Systems* 31.5 (Sept. 2016), pp. 4128–4134. DOI: 10.1109/tpwrs.2015.2505185 (cit. on pp. 13–15).

[5] C.-W. Liu et al. "Toward a CPFLOW-Based Algorithm to Compute All the Type-1 Load-Flow Solutions in Electric Power Systems". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 52.3 (Mar. 2005), pp. 625–630. DOI: 10.1109/tcsi.2004.842883 (cit. on pp. 13–15).

[6] R. J. Avalos et al. "Equivalency of Continuation and Optimization Methods to Determine Saddle-Node and Limit-Induced Bifurcations in Power Systems". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 56.1 (Jan. 2009), pp. 210–223. DOI: 10.1109/tcsi.2008.925941 (cit. on pp. 13–15).

References III

[7] A. J. Flueck and J. R. Dondeti. "A new continuation power flow tool for investigating the nonlinear effects of transmission branch parameter variations". In: *IEEE Transactions on Power Systems* 15.1 (Feb. 2000), pp. 223–227. DOI: 10.1109/59.852125 (cit. on pp. 13–15).

[8] A. J. Flueck and W. Qiu. "A New Technique for Evaluating the Severity of Branch Outage Contingencies based on Two-Parameter Continuation". In: *2004 IEEE Power Engineering Society General Meeting*. Institute of Electrical and Electronics Engineers (IEEE), June 2004, pp. 323–329. DOI: 10.1109/pes.2004.1372806 (cit. on pp. 13–15).

References IV

[9] R. R. Matarucco et al. "Alternative Parameterization for Assessing Branch Outages by a Continuation Method". In: *2004 IEEE/PES Transmision and Distribution Conference and Exposition: Latin America (IEEE Cat. No. 04EX956)*. Institute of Electrical and Electronics Engineers (IEEE), Nov. 2004. DOI: 10.1109/tdc.2004.1432348 (cit. on pp. 13–15).

[10] R. R. Matarucco, A. B. Neto, and D. A. Alves. "Assessment of branch outage contingencies using the continuation method". In: *International Journal of Electrical Power and Energy Systems* 55 (Feb. 2014), pp. 74–81. DOI: 10.1016/j.ijepes.2013.08.029 (cit. on pp. 13–15).

[11] J. Lavaei and S. H. Low. "Zero Duality Gap in Optimal Power Flow Problem". In: *IEEE Transactions on Power Systems* 27.1 (Feb. 2012), pp. 92–107. DOI: 10.1109/tpwrs.2011.2160974 (cit. on pp. 13–15).

[12] K. Lehmann, A. Grastien, and P. Van Hentenryck. “AC-Feasibility on Tree Networks is NP-Hard”. In: *IEEE Transactions on Power Systems* 31.1 (Jan. 2016), pp. 798–801. DOI: 10.1109/tpwrs.2015.2407363 (cit. on pp. 13–15).

[13] D. Bienstock and A. Verma. “Strong NP-hardness of AC power flows feasibility”. In: *Operations Research Letters* 47.6 (Nov. 2019), pp. 494–501. DOI: 10.1016/j.orl.2019.08.009 (cit. on pp. 13–15).

[14] O. G. Lateef. “Measurement-based Parameter Estimation and Analysis of Power System”. PhD thesis. Georgia Institute of Technology, 2020. URL: https://smartech.gatech.edu/handle/1853/63600 (visited on 09/27/2022) (cit. on pp. 16–20).

[15] M. Vanin et al. “Combined Unbalanced Distribution System State and Line Impedance Matrix Estimation”. In: (Sept. 22, 2022). DOI: 10.48550/arXiv.2209.10938. arXiv: 2209.10938v1 [eess.SY] (cit. on pp. 16–20).

References VI

[16] S. Frank and S. Rebennack. "An introduction to optimal power flow: Theory, formulation, and examples". In: *IIE Transactions* 48.12 (Apr. 27, 2016), pp. 1172–1197. DOI: 10.1080/0740817x.2016.1189626 (cit. on pp. 21–23).

[17] L. Gan and S. H. Low. "An Online Gradient Algorithm for Optimal Power Flow on Radial Networks". In: *IEEE Journal on Selected Areas in Communications* 34.3 (Mar. 10, 2016), pp. 625–638. DOI: 10.1109/jsac.2016.2525598 (cit. on pp. 21–23).

[18] Y. Tang, K. Dvijotham, and S. Low. "Real-Time Optimal Power Flow". In: *IEEE Transactions on Smart Grid* 8.6 (Nov. 2017), pp. 2963–2973. DOI: 10.1109/tsg.2017.2704922 (cit. on pp. 21–23).

References VII

[19] D. K. Molzahn and I. A. Hiskens. "A Survey of Relaxations and Approximations of the Power Flow Equations". In: *Foundations and Trends® in Electric Energy Systems* 4.1-2 (Feb. 4, 2019), pp. 1–221. DOI: 10.1561/3100000012. URL: https://molzahn.github.io/pubs/molzahn_hiskens-fnt2019.pdf (visited on 04/10/2022) (cit. on pp. 21–23).

[20] J. Meisel. "System Incremental Cost Calculations Using the Participation Factor Load-Flow Formulation". In: *IEEE Transactions on Power Systems* 8.1 (Feb. 1993), pp. 357–363. DOI: 10.1109/59.221220 (cit. on pp. 21–23).

[21] M. C. Grant and S. P. Boyd. *CVX. Matlab Software for Disciplined Convex Programming*. Version 2.2, Build 1148 (62bfcca). CVX Research, Inc., Jan. 28, 2020. URL: http://cvxr.com/cvx (visited on 07/05/2022) (cit. on pp. 24–26).

References VIII

[22] J. F. Sturm et al. *SeDuMi. A linear/quadratic/semidefinite solver for MATLAB and Octave*. Version 1.3.4. Jan. 10, 2020. URL: https://github.com/sqlp/sedumi/releases/tag/v1.3.4 (visited on 08/15/2022) (cit. on pp. 29–31).

[23] Y. Ye, M. J. Todd, and S. Mizuno. "An $\mathcal{O}\left(\sqrt{n}L\right)$-Iteration Homogeneous and Self-Dual Linear Programming Algorithm". In: *Mathematics of Operations Research* 19.1 (Feb. 1994), pp. 53–67. DOI: 10.1287/moor.19.1.53 (cit. on pp. 29–31).

[24] J. F. Sturm. "Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones". In: *Optimization Methods and Software* 11.1-4 (Mar. 16, 1999), pp. 625–653. DOI: 10.1080/10556789908805766 (cit. on pp. 29–31).

References IX

[25] K.-C. Toh, M. J. Todd, and R. H. Tütüncü. *SDPT3. A MATLAB software for semidefinite-quadratic-linear programming*. Version 4.0. Sept. 11, 2020. URL: https://blog.nus.edu.sg/mattohkc/softwares/sdpt3/ (visited on 08/15/2022) (cit. on pp. 29–31).

[26] K.-C. Toh, M. J. Todd, and R. H. Tütüncü. "SDPT3 — A Matlab software package for semidefinite programming, Version 1.3". In: *Optimization Methods and Software* 11.1-4 (Jan. 1999), pp. 545–581. DOI: 10.1080/10556789908805762 (cit. on pp. 29–31).

[27] R. H. Tütüncü, K.-C. Toh, and M. J. Todd. "Solving semidefinite-quadratic-linear programs using SDPT3". In: *Mathematical Programming, Series B* 95.2 (Feb. 1, 2003), pp. 189–217. DOI: 10.1007/s10107-002-0347-5 (cit. on pp. 29–31).

[28] M. J. D. Powell. *A FORTRAN Subroutine for Solving Systems of Non-linear Algebraic Equations*. Tech. rep. AERE-R. 5947. Harwell, Berkshire, United Kingdom: United Kingdom Atomic Energy Authority, Nov. 15, 1968. URL: https://www.osti.gov/biblio/4772677 (visited on 07/28/2022) (cit. on pp. 29–31).

[29] M. J. D. Powell. "A Hybrid Method for Nonlinear Equations". In: *Numerical Methods for Nonlinear Algebraic Equations*. Ed. by P. Rabinowitz. Gorden and Breach Science Publishers, 1970, pp. 84–114 (cit. on pp. 29–31).

[30] M. J. D. Powell. "A FORTRAN Subroutine for Solving Systems of Nonlinear Algebraic Equations". In: *Numerical Methods for Nonlinear Algebraic Equations*. Ed. by P. Rabinowitz. Gorden and Breach Science Publishers, 1970, pp. 115–161 (cit. on pp. 29–31).

References XI

[31] K. Levenberg. "A method for the solution of certain non-linear problems in least squares". In: *Quarterly of Applied Mathematics* 2.2 (July 1944), pp. 164–168. DOI: 10.1090/qam/10666 (cit. on pp. 29–31).

[32] D. W. Marquardt. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (June 1963), pp. 431–441. DOI: 10.1137/0111030 (cit. on pp. 29–31).

[33] J. Nocedal and S. J. Wright. *Numerical Optimization, 2nd ed.* Ed. by T. V. Mikosch, S. I. Resnick, and S. M. Robinson. Springer Science and Business Media, 2006 (cit. on pp. 29–31).

[34] F. Capitanescu et al. "Interior-point based algorithms for the solution of optimal power flow problems". In: *Electric Power Systems Research* 77.5-6 (Apr. 2007), pp. 508–517. DOI: 10.1016/j.epsr.2006.05.003 (cit. on pp. 41–43).

[35] F. Capitanescu and L. Wehenkel. "Experiments with the interior-point method for solving large scale Optimal Power Flow problems". In: *Electric Power Systems Research* 95 (Feb. 2013), pp. 276–283. DOI: 10.1016/j.epsr.2012.10.001 (cit. on pp. 41–43).

[36] J. Kardoš et al. "Complete results for a numerical evaluation of interior point solvers for large-scale optimal power flow problems". In: USI Technical Report Series in Informatics (Nov. 2, 2020). DOI: 10.48550/arXiv.1807.03964. arXiv: 1807.03964v4 [math.OC] (cit. on pp. 41–43).

[37] J. Kardoš et al. "BELTISTOS: A robust interior point method for large-scale optimal power flow problems". In: *Electric Power Systems Research* 212 (Nov. 2022), p. 108613. DOI: 10.1016/j.epsr.2022.108613 (cit. on pp. 41–43).

References XIII

[38] R. D. Zimmerman and H. Wang. *MATPOWER Interior Point Solver (MIPS)*. Version 1.4. Oct. 8, 2020. DOI: 10.5281/zenodo.4073324. URL: https://github.com/MATPOWER/mips/releases/tag/1.4 (visited on 08/15/2022) (cit. on pp. 44–51).

[39] M. Chatzos et al. "High-Fidelity Machine Learning Approximations of Large-Scale Optimal Power Flow". In: (June 29, 2020). DOI: 10.48550/arXiv.2006.16356. arXiv: 2006.16356v1 [eess.SP] (cit. on pp. 55–58).

[40] X. Pan et al. "DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems". In: (July 1, 2022). DOI: 10.48550/arXiv.2007.01002. arXiv: 2007.01002v6 [eess.SY] (cit. on pp. 55–58).

References XIV

[41] P. L. Donti, D. Rolnick, and J. Z. Kolter. "DC3: A learning method for optimization with hard constraints". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=V1ZHVxJ6dSS (visited on 06/28/2021) (cit. on pp. 55–58).

[42] M. Blondel et al. "Efficient and Modular Implicit Differentiation". In: (May 20, 2022). DOI: 10.48550/arXiv.2105.15183. arXiv: 2105.15183v5 [cs.LG] (cit. on p. 70).

[43] A. N. Angelopoulos and S. Bates. "A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification". In: (Sept. 3, 2022). DOI: https://doi.org/10.48550/arXiv.2107.07511. arXiv: 2107.07511v5 [cs.LG] (cit. on p. 70).