**ECE 508: Computational Power Systems** **Winter 2025**

# Lecture 6: Algorithms for Online AC-OPF

*Lecturer: Vladimir Dvorkin* *Scribe(s): Renjian Ruan, Jenna Knudtson*

**Note**: *LaTeX template courtesy of UC Berkeley EECS dept.*



## 6.1 Offline vs. Online AC-OPF

### 6.1.1 Offline AC-OPF

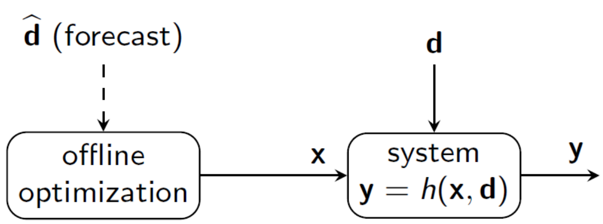


Figure 6.1: Offline AC-OPF

Offline AC-OPF solves the optimal power flow problem under the assumption of a time-invariant system. Given a forecasted demand $\hat{\mathbf{d}}$, it computes a single optimal control input $\mathbf{x}$, which is then applied to the system. The control system processes $\mathbf{x}$ to produce the system output $\mathbf{y}$, but there is no feedback loop that refines $\mathbf{x}$ based on actual system behavior.

**Benefits:**

- Can handle large-scale optimization problems
- Provides global optimal solutions under static conditions
- Enables long-term planning and forecast-based decision-making

### 6.1.2 Online AC-OPF

Online AC-OPF incorporates real-time optimization within the feedback control loop. At each time step, the system observes the current output $\mathbf{y}_n$ and computes an updated control input $\mathbf{x}_{n+1}$ based on real-time conditions. This input is then applied to the system, generating the next output $\mathbf{y}_{n+1}$, which continuously informs the optimization process.

**Benefits:**

- Enhances robustness against time-varying disturbances

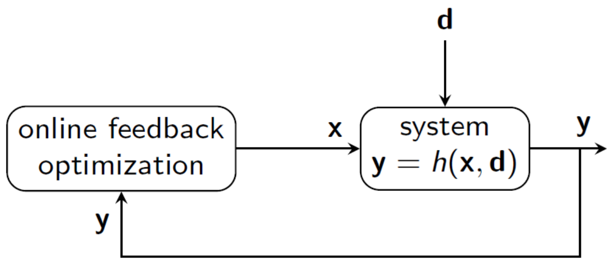


Figure 6.2: Online AC-OPF

- Enables rapid response to system changes such as line and generator outages
- Reduces computational burden by solving smaller, incremental optimization problems
- Reduce model-dependence

It's thus important to understand how online optimization reduces model dependency. A useful example is based on the LinDistFlow equation.

For offline OPF, the optimization must explicitly satisfy the LinDistFlow model:

$$h(v, \mathbf{q}) = -v + v_0\mathbf{1} + \mathbf{Rd} + \mathbf{Xq} = 0$$

to determine the optimal reactive power injection $\mathbf{q}$. This approach requires full model knowledge and dependence on system parameters.

In contrast, online OPF only needs to compute:

$$\frac{\partial h}{\partial \mathbf{q}} = \mathbf{X}$$

to adjust the control variable iteratively. This significantly reduces model dependency, as the optimization process does not require full knowledge of the system model but instead relies on real-time feedback and local sensitivities. As a result, online OPF achieves improved adaptability and robustness by reducing model-dependency in dynamic environments.

## 6.2 Real-Time AC-OPF in Transmission and Distribution Systems

### 6.2.1 Online AC-OPF for Distribution Systems

In a distribution system, online AC-OPF operates according to the control diagram shown in Fig. 6.3. The left side of the figure illustrates the role of real-time feedback optimization within the control loop. Given real-time voltage measurements $\mathbf{v}$, the optimization algorithm adjusts reactive power injections $\mathbf{q}$ to minimize the overall system cost, as formulated in Eq. 6.1.

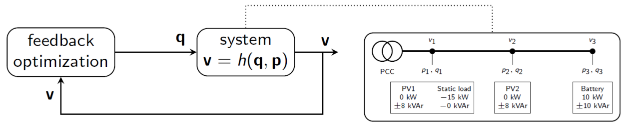


Figure 6.3: Online AC-OPF for Distribution

The right side of Fig. 6.3 provides a simplified representation of the complex distribution system, denoted as $h$. This system inherently imposes constraints on the optimization problem, including power flow equations (LinDistFlow), maximum power limits, and voltage deviation constraints, as expressed in Eq. 6.2 - 6.4.

$$\begin{aligned}
\underset{\mathbf{q}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{q}^\top \mathbf{C}\mathbf{q} && \text{cost function} && (6.1)\\
\text{subject to} \quad & \mathbf{v} = v_0 \mathbf{1} + \mathbf{R}\mathbf{p} + \mathbf{X}\mathbf{q} && \text{LinDistFlow equations} && (6.2)\\
& \underline{\mathbf{q}} \leq \mathbf{q} \leq \overline{\mathbf{q}} && \text{injection limits} && (6.3)\\
& \underline{\mathbf{v}} \leq \mathbf{v} \leq \overline{\mathbf{v}} && \text{voltage limits} && (6.4)
\end{aligned}$$

### 6.2.2 Online AC-OPF for Transmission Systems

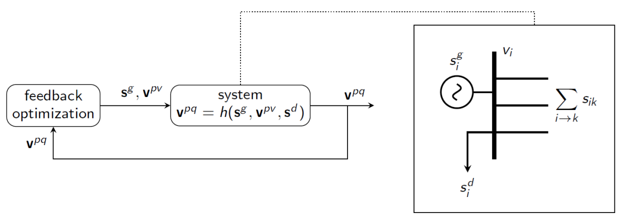


Figure 6.4: Online AC-OPF for Transmission

Similarly, Fig. 6.4 illustrates the operation of real-time feedback optimization in transmission systems. Unlike distribution systems, where real-time control primarily focuses on reactive power injections, transmission systems require optimization over total complex power $\mathbf{s}^g$ and voltage magnitudes at PV buses $\mathbf{v}^{pv}$. The primary measurements available in transmission systems are voltage magnitudes at PQ buses, which are used as feedback for real-time control.

The right side of Fig. 6.4 provides an example of a radial transmission network, which imposes constraints on the optimization problem. These constraints include complex power flow equations, generation limits, voltage limits, and line flow capacity constraints, as formulated below.

$$\begin{aligned}
\underset{\mathbf{s}^g, \mathbf{v}^{pv}}{\text{minimize}} \quad & c(\mathbf{s}^g) && && (6.5)\\
\text{subject to} \quad & s_i^g - s_i^d = \sum_{i\to k} s_{ik}(v_i, v_k) && \text{complex power flow} && (6.6)\\
& \underline{s}_i^g \leq s_i^g \leq \overline{s}_i^g && \text{generation limits} && (6.7)\\
& \underline{v}_i \leq v_i \leq \overline{v}_i && \text{voltage limits} && (6.8)\\
& s_{ik}(v_i, v_k) \leq \overline{s}_{ik}, \quad \forall (i,k) && \text{complex power flow limits} && (6.9)
\end{aligned}$$

## 6.3 Solving Nonlinear Optimization as Dynamic Systems

Given a general nonlinear optimization program:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g(\mathbf{x}) \leq \mathbf{0} \quad : \boldsymbol{\mu} \\ & h(\mathbf{x}) = \mathbf{0} \quad : \boldsymbol{\lambda} \end{aligned}$$

The optimality conditions for this problem can be expressed using the Lagrangian:

$$\nabla f(\mathbf{x}^\star) + \frac{\partial g(\mathbf{x}^\star)}{\partial \mathbf{x}}^\top \boldsymbol{\mu}^\star + \frac{\partial h(\mathbf{x}^\star)}{\partial \mathbf{x}}^\top \boldsymbol{\lambda}^\star = \mathbf{0} \tag{6.10}$$

Now, consider a dynamical system of the form:

$$\dot{\mathbf{x}} = F(\mathbf{x}) \tag{6.11}$$

, where optimality is reached at $\mathbf{x}^\star$ if and only if $\dot{\mathbf{x}} = 0$.

By comparing Eq. 6.10 and Eq. 6.11, we can write the dynamic system with a closed form expression:

$$\dot{\mathbf{x}} = \underbrace{-\nabla f(\mathbf{x})}_{\text{optimality}} \underbrace{- \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\mu}}_{\text{safety ``}\leq\text{''}} \underbrace{- \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\lambda}}_{\text{safety ``}=\text{''}}$$

The first term $\nabla f(\mathbf{x})$ ensures the direction of descent towards optimal $\mathbf{x}$, while the second and third terms enforced feasibility by enforcing primal feasibility and complementary slackness.

Given the closed form expression of $\dot{\mathbf{x}}$ as a dynamic system, we can perform a descent method to extract the optimal value $\mathbf{x}^\star$. In the lecture, professor introduced following three descent methods:

- Saddle-point flow
- Projected gradient flow
- Safe gradient flow

### 6.3.1 Saddle-point flow

The first descent method, saddle-point flow, uses primary descent and dual ascent to converge to the saddle-point of the Lagragian function of an equality-constrained optimization problem of the form:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) \\ & h(\mathbf{x}) = \mathbf{0} \quad : \boldsymbol{\lambda} \end{aligned}$$

with Lagrangian function

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top h(\mathbf{x})$$

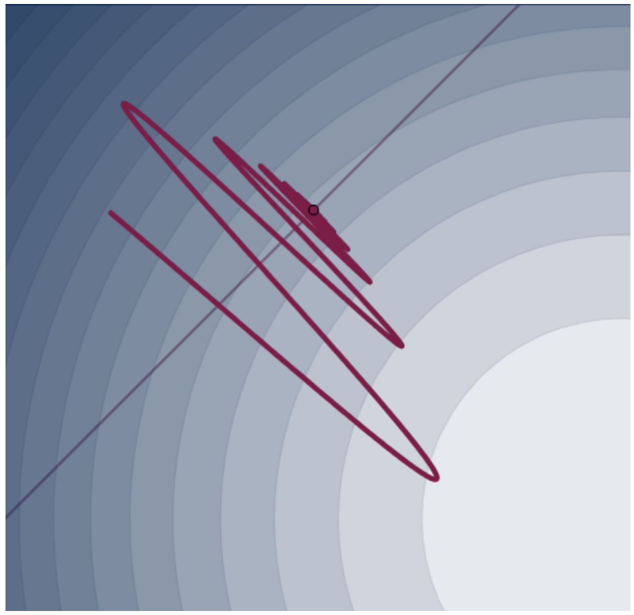

Figure 6.5: Saddle-point flow

The trajectory for the primary variables is the negative gradient of the Lagrangian with respect to x, and the trajectory for the dual variables is the positive gradient of the Lagrangian with respect to $\lambda$:

$$\dot{\mathbf{x}} = \underbrace{-\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\boldsymbol{\lambda})^\top}_{\text{descent}} \quad \dot{\boldsymbol{\lambda}} = \underbrace{\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\mathbf{x},\boldsymbol{\lambda})^\top}_{\text{ascent}}$$

Consider the following example:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \quad : \boldsymbol{\lambda} \end{aligned}$$

The trajectories are:

$$\begin{aligned} \dot{\mathbf{x}} &= -\nabla f(\mathbf{x})^\top - \mathbf{A}^\top\boldsymbol{\lambda} \\ \dot{\boldsymbol{\lambda}} &= \mathbf{A}\mathbf{x} - \mathbf{b} \end{aligned}$$

and the equilibrium point (KKT conditions) is defined by:

$$\mathbf{0} = \nabla f(\mathbf{x}^\star)^\top + \mathbf{A}^\top\boldsymbol{\lambda}^\star, \quad \mathbf{0} = \mathbf{A}\mathbf{x}^\star - \mathbf{b}$$

Fig 6.5 depicts the convergence of saddle point flow for the following example:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & 0.125\|\mathbf{x}\|_2^2 - 0.5x_1 + 0.25x_2 \\ \text{subject to} \quad & x_1 - x_2 = 0 \end{aligned}$$

The trajectories oscillate around the equality constraint until it reaches the optimum. However, saddle-point flow may not converge if $f(x)$ or $h(x)$ are not convex. Saddle-point flow can be strengthened by adding regularization terms to the objective function. If $f(\mathbf{x})$ is not strictly convex:

$$\mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top h(\mathbf{x}) + \frac{\rho}{2}\|h(\mathbf{x})\|_2^2$$

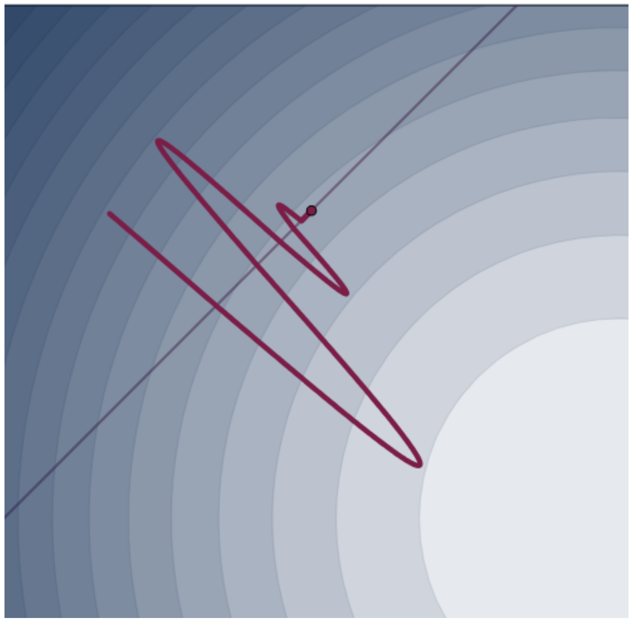

Figure 6.6: Saddle-point flow with regularization term.

What if $h(\mathbf{x})$ is non-convex?

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top h(\mathbf{x}) + \frac{\rho}{2} \|\boldsymbol{\lambda}\|_2^2$$

Adding the regularization term also leads to faster convergence. Using the example from above:

$$\begin{aligned}
\underset{\mathbf{x}\in\mathbb{R}^2}{\text{minimize}} \quad & 0.125\|\mathbf{x}\|_2^2 - 0.5x_1 \\
& \qquad + 0.25x_2 + \frac{\rho}{2} \|x_1 - x_2\|_2^2 \\
\text{subject to} \quad & x_1 - x_2 = 0
\end{aligned}$$

The convergence path is now shorter, as shown by Fig 6.6:

### 6.3.2 Projected gradient flow

The second descent method, projected gradient flow, employs classic projected gradient descent, and can be used with both equality and inequality constraints. For an optimization problem of the form:

$$\begin{aligned}
\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) \\
\text{subject to} \quad & g(\mathbf{x}) \le \mathbf{0} \\
& h(\mathbf{x}) = \mathbf{0}
\end{aligned}$$

the trajectory of projected gradient flow will be:

$$\dot{x} = P_{\mathcal{C}} \left[ -\nabla f(\mathbf{x})^\top \right] (x)$$

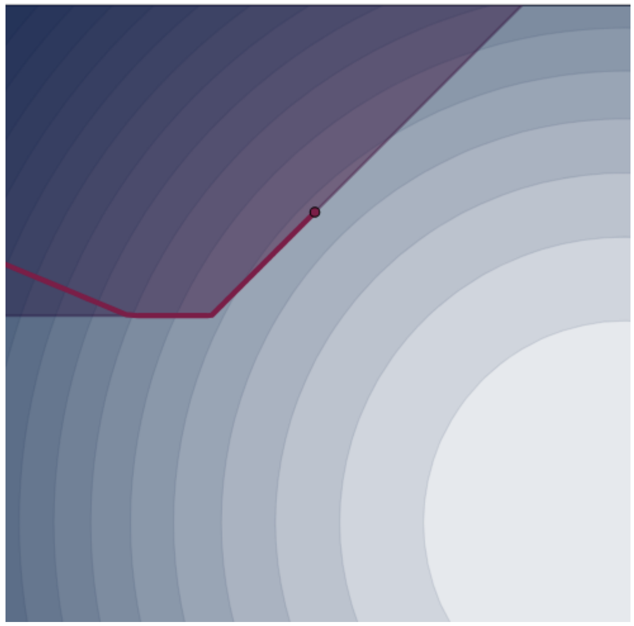

Figure 6.7: Projected gradient flow.

where $P_{\mathcal{C}}[\,\mathbf{y}\,] = \underset{\mathbf{x}}{\operatorname{argmin}}\ \|\mathbf{x} - \mathbf{y}\|_2^2$ is the projection on the feasible region $\mathcal{C}$. For example:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

Forward Euler discretization of PGF:

$$\begin{aligned} \widehat{\mathbf{x}} &= \mathbf{x}_t - \eta \nabla f(\mathbf{x})^\top \\ \mathbf{x}_{t+1} &= \underset{\mathbf{x}}{\operatorname{argmin}}\ \|\widehat{\mathbf{x}} - \mathbf{x}\|_2^2 \\ & \quad \text{subject to} \ \ \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

Fig 6.7 shows that PGF results in a non-smooth but feasible trajectory to the optimal solution for the following example.

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^2}{\text{minimize}} \quad & 0.125\|\mathbf{x}\|_2^2 - 0.5x_1 + 0.25x_2 \\ \text{subject to} \quad & x_1 - x_2 \leq 0 \\ & x_2 \geq 0 \end{aligned}$$

Projected gradient flow and saddle-point flow can be combined by dualizing some constraints and projecting others. For example, for the following system:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \\ & g(\mathbf{x}) \leq \mathbf{0} \quad : \boldsymbol{\mu} \end{aligned}$$

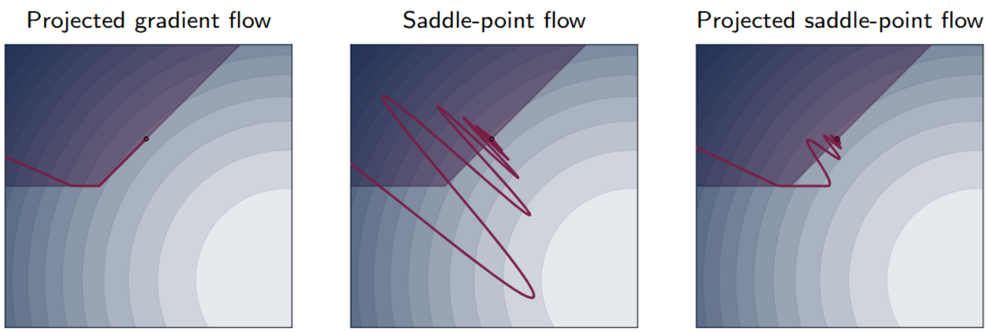


Figure 6.8: Convergence of different methods.

with partial Lagrangian function

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\mu}^\top g(\mathbf{x})$$

the trajectories of the system are

$$\dot{\mathbf{x}} = P_{\mathcal{X}}\Big[\underbrace{-\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \boldsymbol{\mu})^\top}_{-\nabla f(\mathbf{x})^\top - \nabla g(\mathbf{x})^\top \boldsymbol{\mu}}\Big] \quad \dot{\boldsymbol{\mu}} = P_{\mathbb{R}^m_+}\Big[\underbrace{\nabla_{\boldsymbol{\mu}}\mathcal{L}(\mathbf{x}, \boldsymbol{\mu})^\top}_{g(\mathbf{x})}\Big]$$

Using the same numerical example as before, we dualize the first constraint and project the second. This results in faster convergence, as shown in fig 6.8.

$$\begin{array}{rll} \underset{\mathbf{x}\in\mathbb{R}^2}{\text{minimize}} & 0.125\|\mathbf{x}\|_2^2 - 0.5x_1 + 0.25x_2 & \\ \text{subject to} & x_2 \geq x_1 & \text{(dualize)} \\ & x_2 \geq 0 & \text{(project)} \end{array}$$

### 6.3.3 Safe gradient flow

Safe gradient flow enjoys the best of both worlds. We define the trajectory of x while optimizing the dual variables each step:

$$\dot{x} = \underbrace{-\nabla f(\mathbf{x})}_{\text{optimality}} \underbrace{- \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\mu}}_{\text{safety "}\leq\text{"}} \underbrace{- \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\lambda}}_{\text{safety "="}} \tag{6.12}$$

At every step, select duals $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ by solving an optimization

$$\begin{bmatrix}\boldsymbol{\mu}(\mathbf{x}) \\ \boldsymbol{\lambda}(\mathbf{x})\end{bmatrix} \in \underset{\boldsymbol{\mu},\boldsymbol{\lambda}\in K_\alpha(\mathbf{x})}{\operatorname{argmin}} \left\| \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\mu} + \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\lambda} \right\|_2^2$$

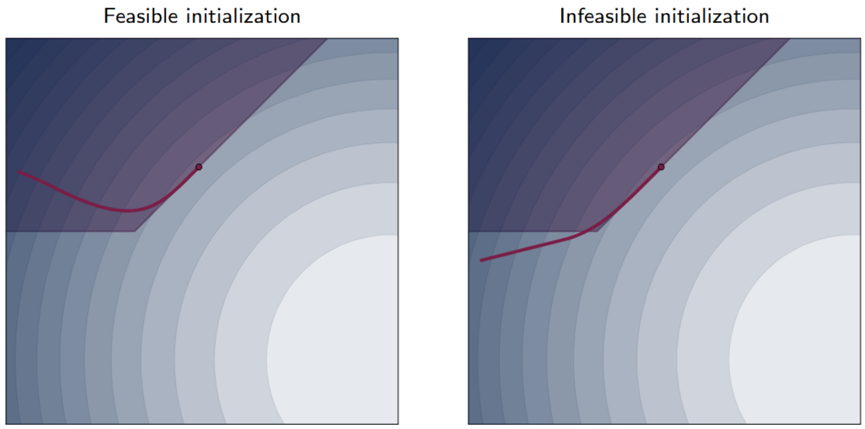


Figure 6.9: Safe gradient flow.

where $K_\alpha(\mathbf{x})$ is the admissible control set:

$$K_\alpha(\mathbf{x}) = \Bigg\{ (\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathbb{R}_+^m \times \mathbb{R}^k \;\Bigg|\; -\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\mu} - \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\lambda} \le \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \nabla f(\mathbf{x}) - \alpha g(\mathbf{x}), \; -\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\mu} - \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}^\top \boldsymbol{\lambda} = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \nabla f(\mathbf{x}) - \alpha h(\mathbf{x}) \Bigg\}$$

Safe gradient flow minimizes the drift of the trajectory while maintaining feasibility. So, trajectories will always stay in the feasible domain if they start at a feasible point; if they start at an infeasible point, they will converge to a feasible one. The admissible control set comes from control barrier function theory; it helps to assure convergence to the feasible range. The convergence curves for the following example are shown in Fig 6.9.

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^2}{\text{minimize}} \quad & 0.125\|\mathbf{x}\|_2^2 - 0.5x_1 + 0.25x_2 \\ \text{subject to} \quad & x_1 - x_2 \le 0 \\ & x_2 \ge 0 \end{aligned}$$

Safe gradient flow can handle non-linear and non-convex constraints, as shown in Fig 6.10

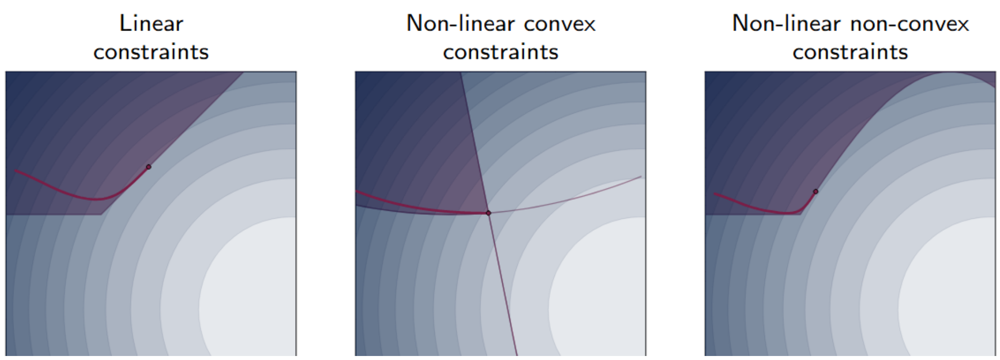


Figure 6.10: Safe gradient flow for different constraints.