A Local Search Algorithm for the Witsenhausen's Counterexample

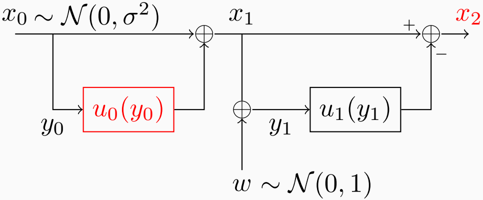
Shih-Hao Tseng, (pronounced as "She-How Zen")
joint work with Kevin Tang

December 15, 2017

School of Electrical and Computer Engineering, Cornell University

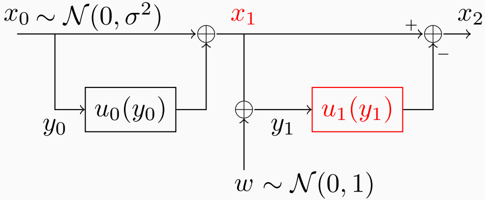
- Witsenhausen's counterexample (Witsenhausen, 1968) is a 2-stage LQG control problem with the objective

$$\min \mathcal{J}\left[u_0, x_2\right] = \min \mathbb{E}\left[k^2 u_0(y_0)^2 + x_2^2\right].$$



- Witsenhausen's counterexample (Witsenhausen, 1968) is a 2-stage LQG control problem with the objective

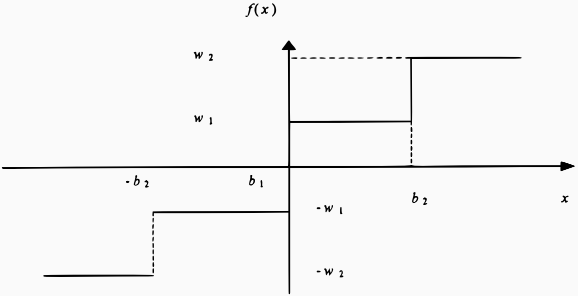
$$\begin{aligned} &\min \mathcal{J}\left[x_1, u_1\right] \\ = &\min \mathbb{E}\left[k^2\left(x_1(x_0) - x_0\right)^2 + \left(x_1(x_0) - u_1\left(x_1(x_0) + w\right)\right)^2\right]. \end{aligned}$$



Previous Attempts

- Witsenhausen showed that affine controllers can perform strictly worse than a non-linear controller.
- The optimal controller remains unknown since 1968.
- Bounds are established for different strategies, but they are all loose.
- Several numerical approximation methods are developed to realize good solutions in practice.

- Mostly, the methods target a class of functions and tune the parameters to find the best one within the class.

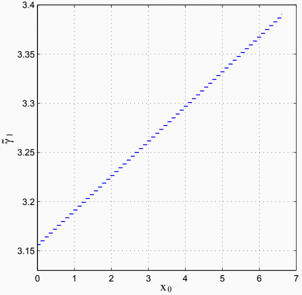


(a) Targeting step functions.

Source: Lee et al., "The Witsenhausen Counterexample: A Hierarchical Search Approach for Nonconvex Optimization Problems," 2001.

Limitations of the Previous Numerical Attempts

- Mostly, the methods target a class of functions and tune the parameters to find the best one within the class.

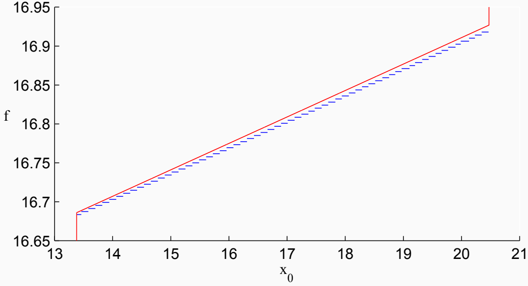


(b) Targeting discrete output functions.

Source: Karlsson et al., "Iterative Source-Channel Coding Approach to Witsenhausen's Counterexample," 2011.

Limitations of the Previous Numerical Attempts

- Mostly, the methods target a class of functions and tune the parameters to find the best one within the class.



(c) Targeting piecewise affine functions.

Source: Mehmetoglu et al., "A Deterministic Annealing Approach to Witsenhausen's Counterexample," 2014.

Limitations of the Previous Numerical Attempts

- Mostly, the methods target a class of functions and tune the parameters to find the best one within the class.
  $\Rightarrow$ What is the “right” class of functions we should focus on?
  $\Rightarrow$ How can we deal with some other parameter settings?

Limitations of the Previous Numerical Attempts

- Mostly, the methods target a class of functions and tune the parameters to find the best one within the class.
  $\Rightarrow$ What is the "right" class of functions we should focus on?
  $\Rightarrow$ How can we deal with some other parameter settings?
- The methods usually leverage the known property of the objective that the optimal second stage controller $u_1(y_1)$ is an MMSE estimator.
  $\Rightarrow$ How can we approach other problems with different objectives?

A General Approach to the Counterexample

- Instead of proposing a method specifically for the Witsenhausen's counterexample, we take a principled approach to find a (potentially non-linear) optimal controller for a control problem.

A General Approach to the Counterexample

- Instead of proposing a method specifically for the Witsenhausen's counterexample, we take a principled approach to find a (potentially non-linear) optimal controller for a control problem.
- Our idea is to specify the necessary conditions according to which local search can be performed.
  $\Rightarrow$ The necessary conditions show be general enough so that they can be applied to other functionals.

Necessary Conditions and Feedback

- Violating a necessary condition of the optimum usually implies a way to improve the solution.

Necessary Conditions and Feedback

- Violating a necessary condition of the optimum usually implies a way to improve the solution.
- A local search algorithm is similar to a feedback control: if the necessary condition is violated, improve the current solution accordingly to meet the condition.

Necessary Conditions and Feedback

- Violating a necessary condition of the optimum usually implies a way to improve the solution.
- A local search algorithm is similar to a feedback control: if the necessary condition is violated, improve the current solution accordingly to meet the condition.
- We propose the local search algorithm based on two specific necessary conditions and the corresponding improvement procedures:
  - Local Nash minimizer $\rightarrow$ Alternative update.
  - Local optimal function value $\rightarrow$ Local denoising.

Minimizers and Local Nash Minimizers

- Given arbitrary bounded functions $(\delta x_1, \delta u_1)$ (the variations), we say
  - (x_1, u_1) is a *minimizer* if

$$\mathcal{J}\left[x_1 + \delta x_1, u_1 + \delta u_1\right] \geq \mathcal{J}\left[x_1, u_1\right].$$

  - (x_1, u_1) is a *local Nash minimizer* if

$$\mathcal{J}\left[x_1 + \delta x_1, u_1\right] \geq \mathcal{J}\left[x_1, u_1\right],$$
$$\mathcal{J}\left[x_1, u_1 + \delta u_1\right] \geq \mathcal{J}\left[x_1, u_1\right].$$

Alternative Update

Necessary Condition: An optimal controller must be a local Nash minimizer.

- By definition, we can check if a solution is a local Nash minimizer by fixing one function and testing if the other minimizes $\mathcal{J}$.

Alternative Update

Necessary Condition: An optimal controller must be a local Nash minimizer.

- By definition, we can check if a solution is a local Nash minimizer by fixing one function and testing if the other minimizes $\mathcal{J}$.

Alternative Update: Alternatively check if x_1 and u_1 form a local Nash minimizer. Improve x_1 or u_1 if the condition is not met.

- Start from an initial $x_1(x_0)$ and use revised Newton's method to update.

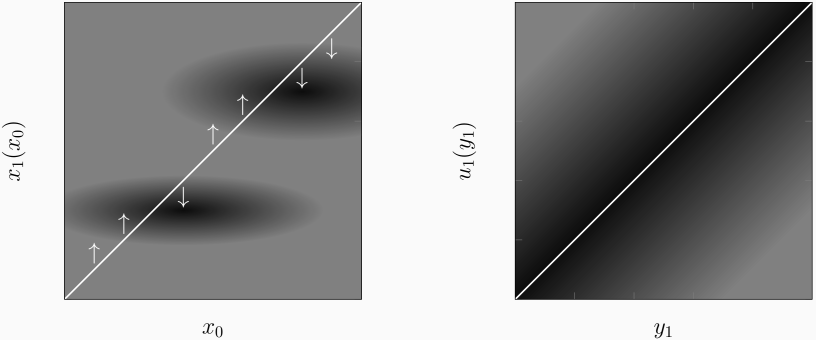


Figure 1: Alternative update: The updated $x_1(x_0)$ will change $\mathcal{J}[x_1, u_1]$ and hence $u_1(y_1)$ needs to be updated.

Alternative Update

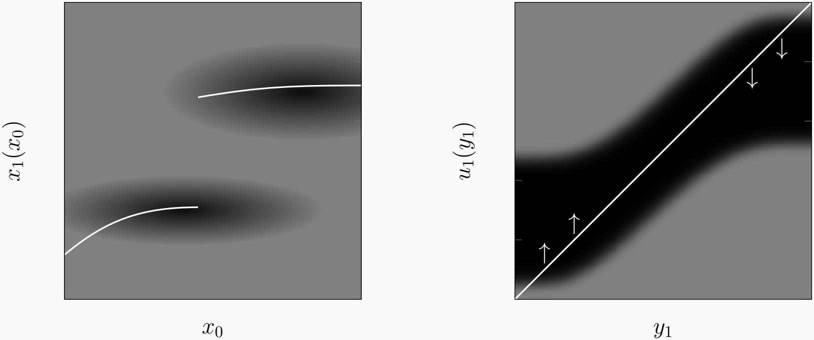


Figure 1: Alternative update: The updated $x_1(x_0)$ will change $\mathcal{J}[x_1, u_1]$ and hence $u_1(y_1)$ needs to be updated.

Alternative Update

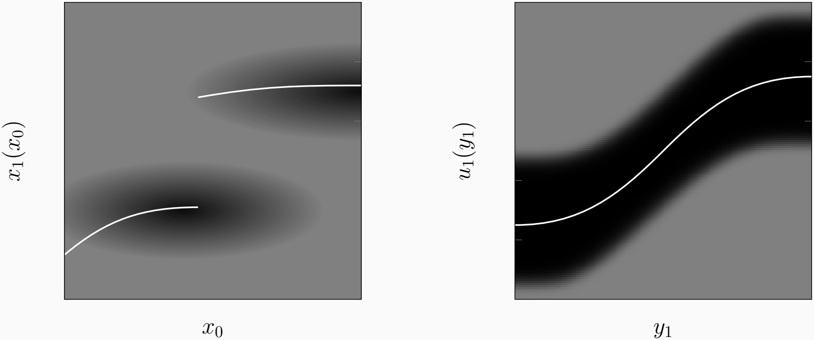


Figure 1: Alternative update: The updated $x_1(x_0)$ will change $\mathcal{J}[x_1, u_1]$ and hence $u_1(y_1)$ needs to be updated.

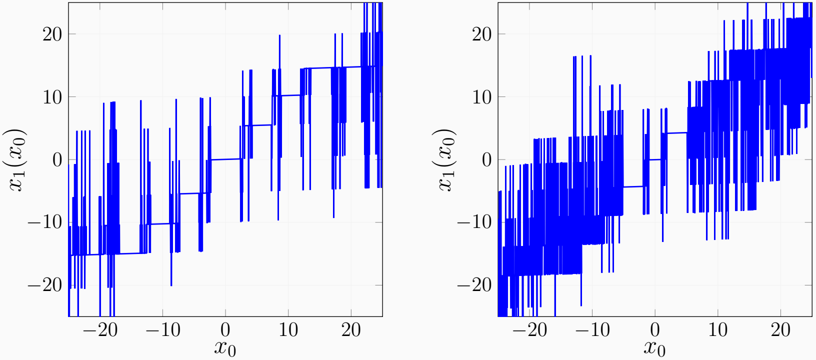
Drawbacks of Alternative Update

- Ideally, we want to start from an initial $x_1(x_0)$ and repeat alternative update to obtain a local minimizer of $\mathcal{J}[x_1, u_1]$, which may be close to a minimizer (an optimal controller).

Drawbacks of Alternative Update

- Ideally, we want to start from an initial $x_1(x_0)$ and repeat alternative update to obtain a local minimizer of $\mathcal{J}[x_1, u_1]$, which may be close to a minimizer (an optimal controller).
- However, the algorithm is sensitive to the initial function $x_1(x_0)$ and the sampling granularity (number of samples procured over the support to approximate continuous functions).

Drawbacks of Alternative Update

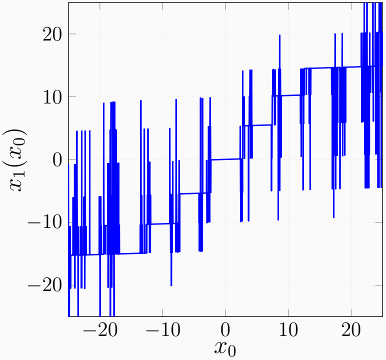


(a) Initialize $x_1(x_0) = x_0$.

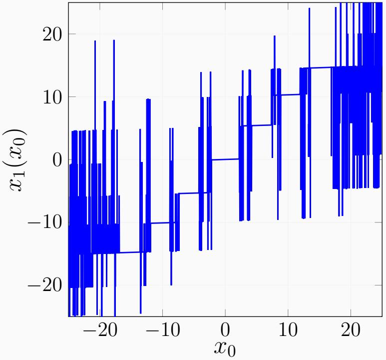
(b) Initialize $x_1(x_0) = x_0|x_0|$.

Figure 2: Alternative update is sensitive to the initial function $x_1(x_0)$.

Drawbacks of Alternative Update



(a) 2000 sample points.

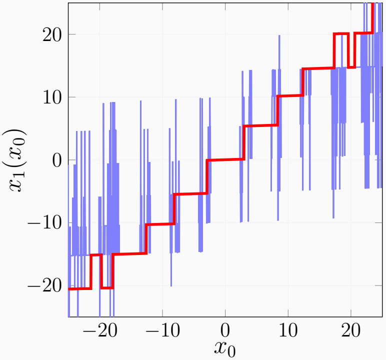


(b) 3000 sample points.

Figure 3: Alternative update is sensitive to the sampling granularity.

Observation

- The resulting $x_1(x_0)$ looks like a function mixed with some noise. Intuitively, $x_1(x_0)$ should be "similar" within a local neighborhood, i.e., left- or right-continuous.



Local Optimal Function Value

- For a fixed u_1, the functional $\mathcal{J}[x_1, u_1]$ can be expressed as

$$\mathcal{J}[x_1, u_1] = \int C_X(x_1(x_0), x_0)\ dx_0.$$

Local Optimal Function Value

- For a fixed u_1, the functional $\mathcal{J}\left[x_1, u_1\right]$ can be expressed as

$$\mathcal{J}\left[x_1, u_1\right] = \int C_X\left(x_1(x_0), x_0\right)\ dx_0.$$

- As such, each $x_1(x_0)$ must minimize C_X at x_0, i.e.,

$$C_X(a, x_0) \geq C_X(x_1(x_0), x_0), \quad \text{for all } a \in \mathbb{R}.$$

In particular, for a given neighborhood $B_r(x_0)$ around x_0, we have

$$C_X\left(x_1(x'), x_0\right) \geq C_X\left(x_1(x_0), x_0\right), \quad \text{for all } x' \in B_r(x_0).$$

Local Denoising

Necessary Condition: $x_1(x_0)$ of an optimal controller must be the minimizer of $C_X(a, x_0)$ within $a \in \{x_1(x') : x' \in B_r(x_0)\}$.

Local Denoising

Necessary Condition: $x_1(x_0)$ of an optimal controller must be the minimizer of $C_X(a, x_0)$ within $a \in \{x_1(x') : x' \in B_r(x_0)\}$.

Local Denoising: For each x_0, check if $x_1(x_0)$ minimizes $C_X(a, x_0)$ within $a \in \{x_1(x') : x' \in B_r(x_0)\}$. Improve x_1 by the minimizer if the condition is not met.

- If there exists a minimizer $x_1(x')$, $x' \in B_r(x_0)$, such that

$$C_X\left(x_1(x_0), x_0\right) > C_X\left(x_1(x'), x_0\right),$$

  then we set $x_1(x_0) = x_1(x')$.

Local Denoising

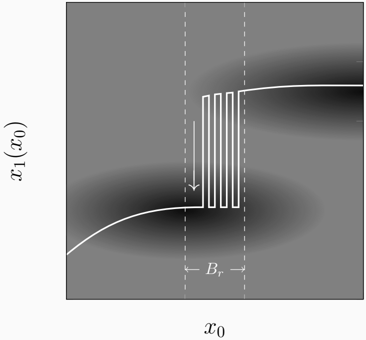


Figure 4: Local denoising: $x_1(x_0)$ may get stuck at different local minima. We "denoise" the case by setting $x_1(x_0)$ to the best $x_1(x')$ where $x' \in B_r(x_0)$.

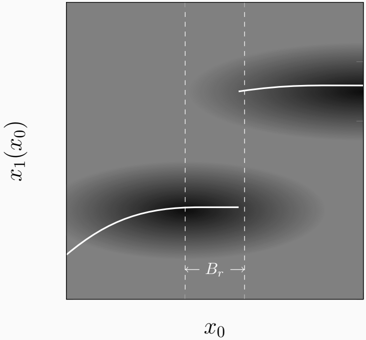


Figure 4: Local denoising: $x_1(x_0)$ may get stuck at different local minima. We "denoise" the case by setting $x_1(x_0)$ to the best $x_1(x')$ where $x' \in B_r(x_0)$.

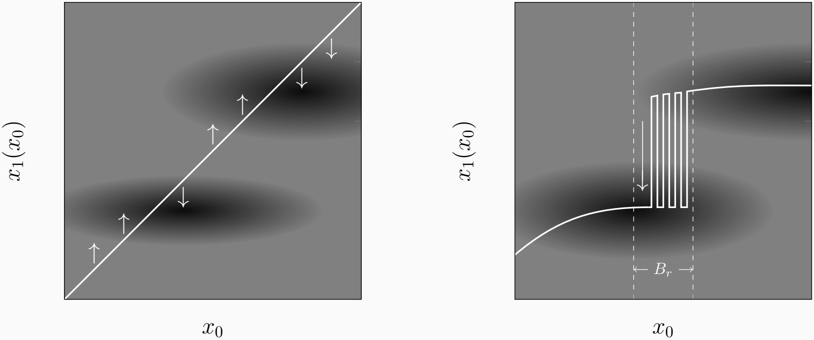


Figure 5: Each x_0 looks vertically during alternative update and horizontally during local denoising.

Local Search Algorithm

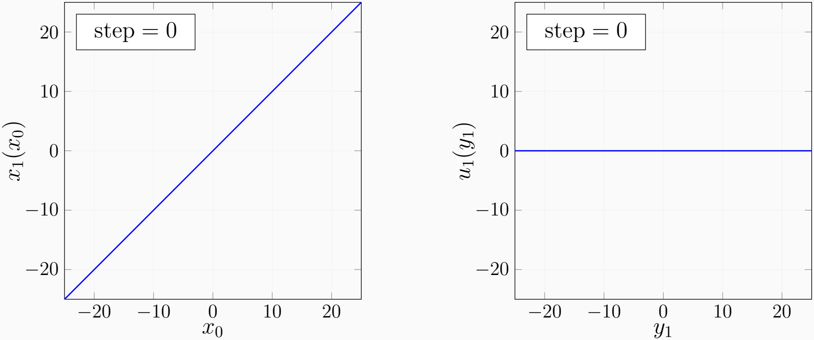


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

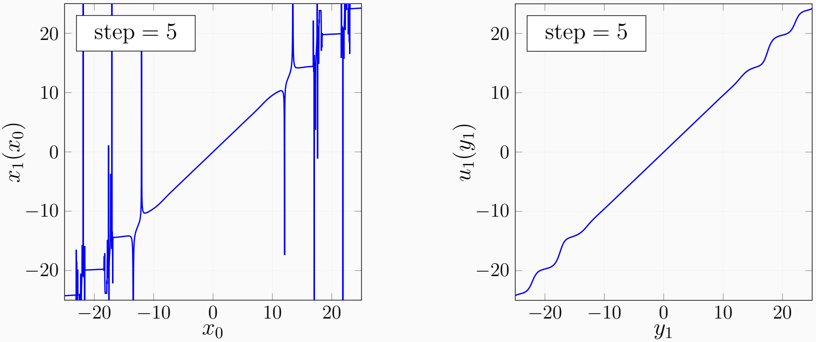


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

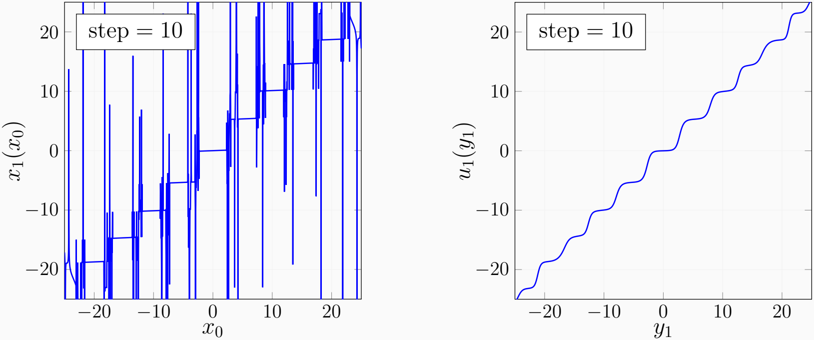


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

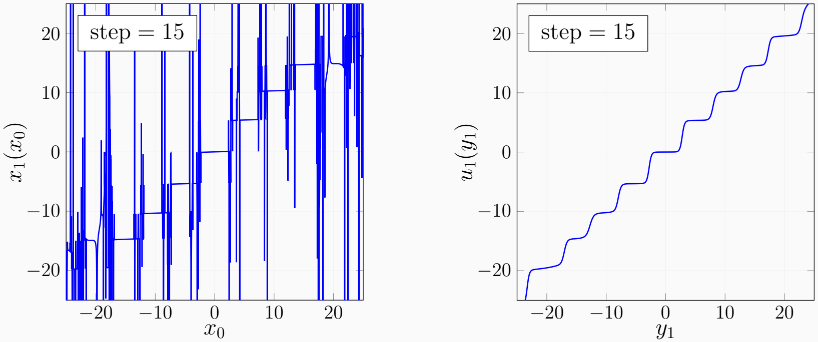
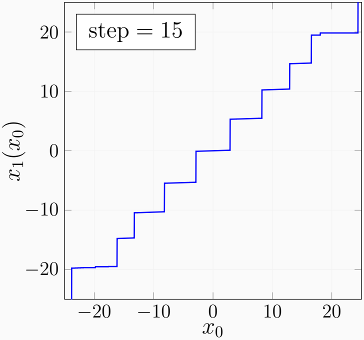


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm



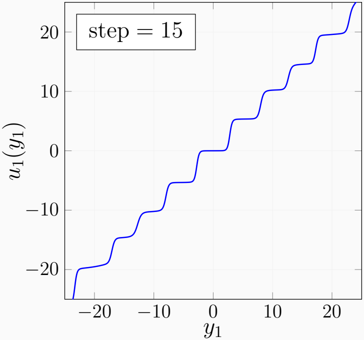


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

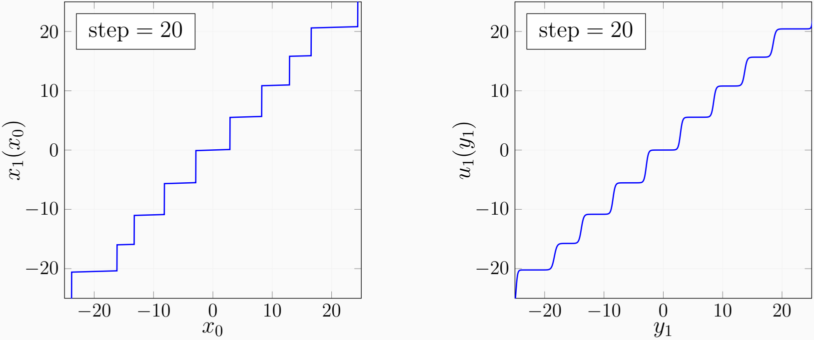


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

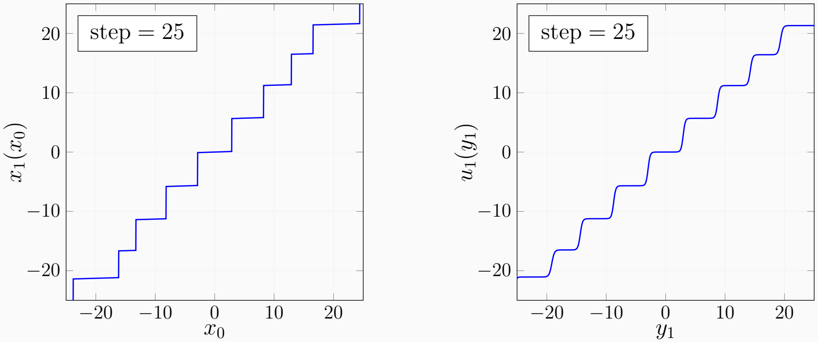


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

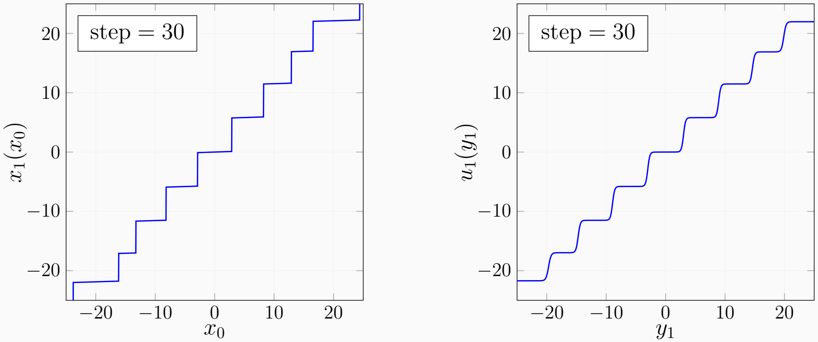


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

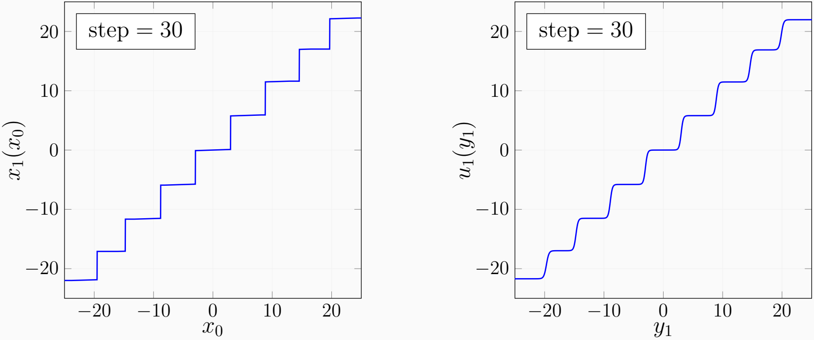


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

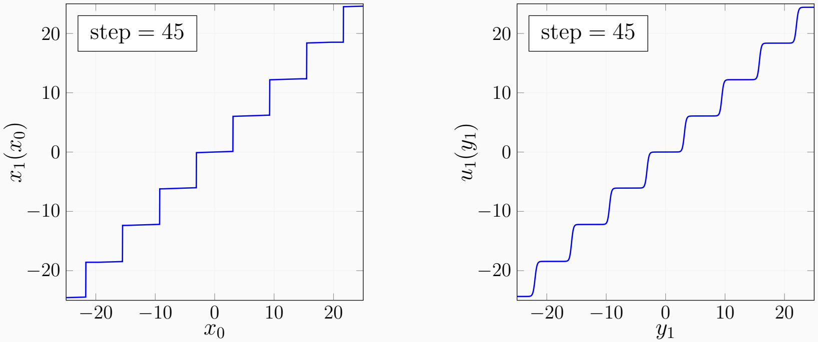


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

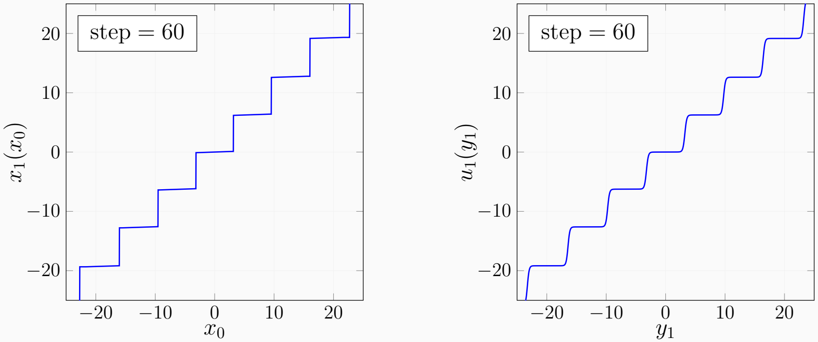


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

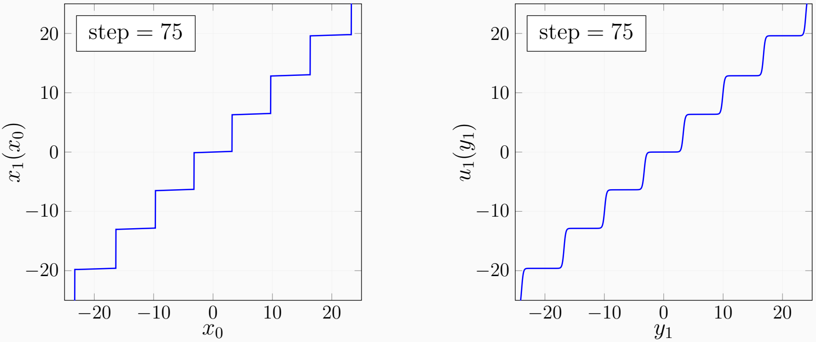


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

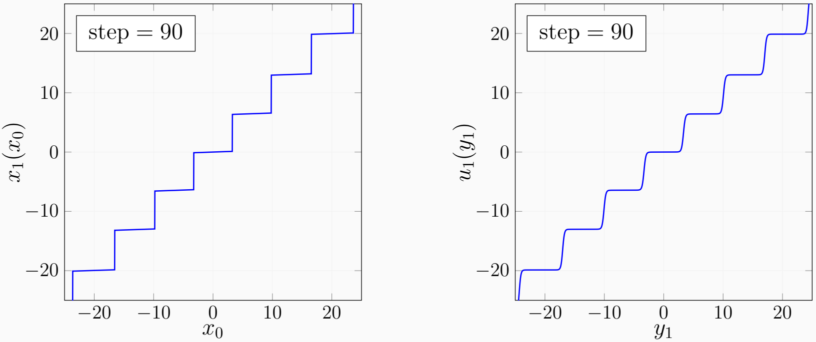


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

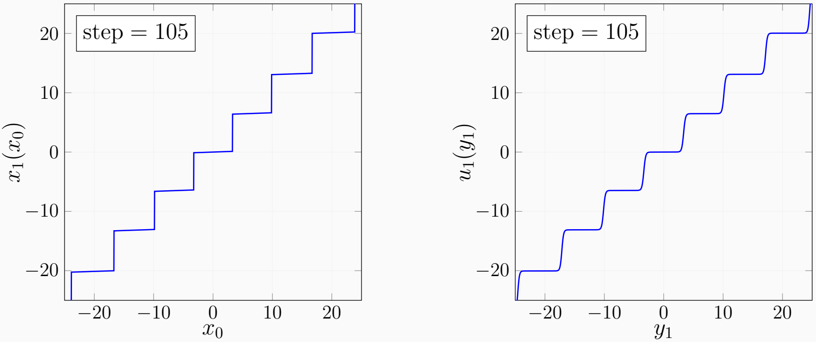


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

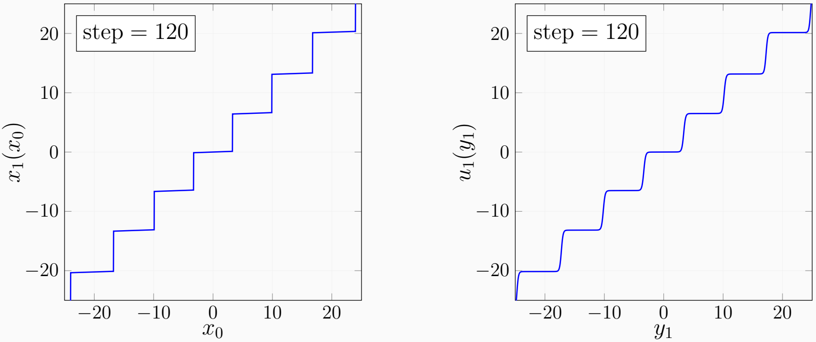


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Local Search Algorithm

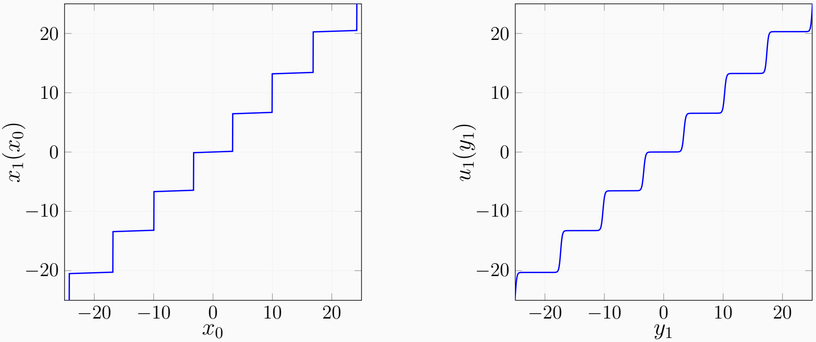


Figure 6: The evolution of x_1 and u_1 under the local search algorithm ($k = 0.2$ and $\sigma = 5$).

Numerical Results

- x_1 and u_1 are supported on $[-25, 25]$ and $[-30, 30]$. 16000 points are chosen to partition the supports evenly so that x_1 and u_1 are approximated by step functions.
- The standard deviation of x_0 is $\sigma = 5$; The initial function $x_1(x_0) = x_0$.

Table 1: Our Result and Major Prior Results ($k = 0.2$)

Source	Total Cost $\mathcal{J}$
Our result	0.166897
Mehmetoglu et al., 2014	0.16692291
Karlsson et al., 2011	0.16692462
Baglietto et al., 2001	0.1701
Witsenhausen, 1968	0.40425320

Different Parameters

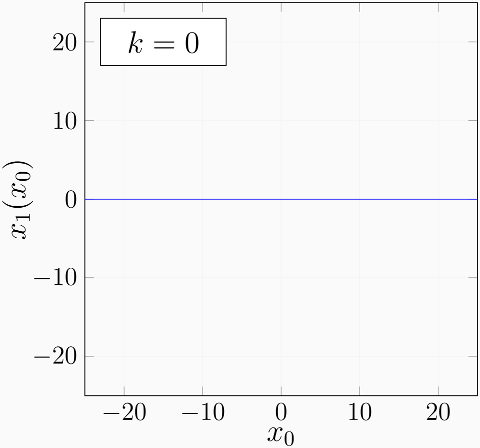


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

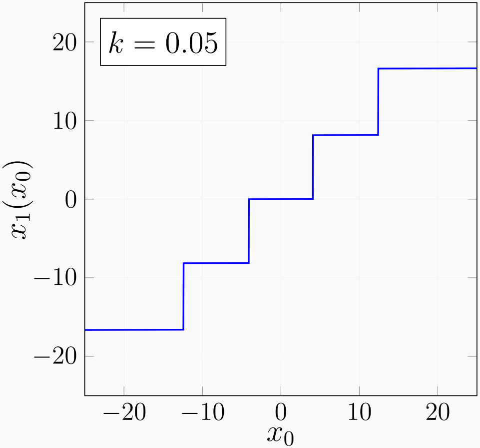


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

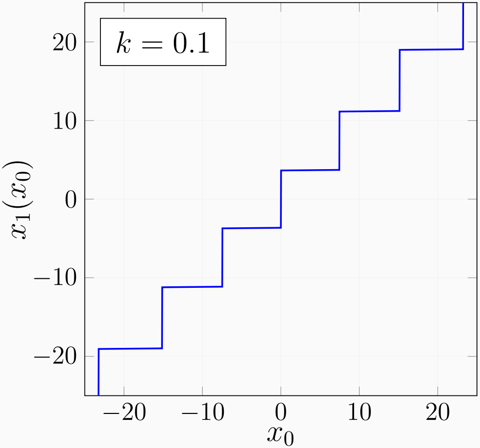


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

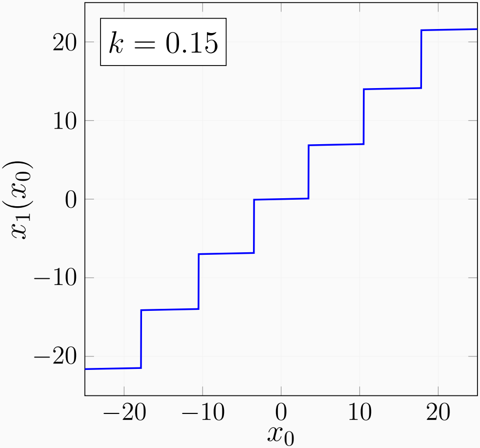


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

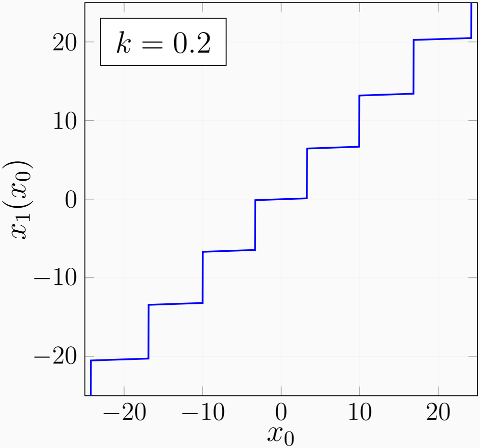


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

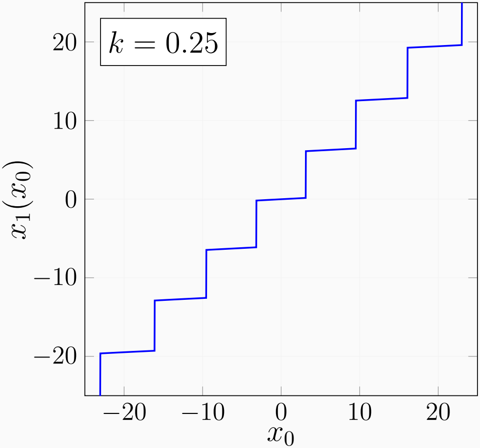


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

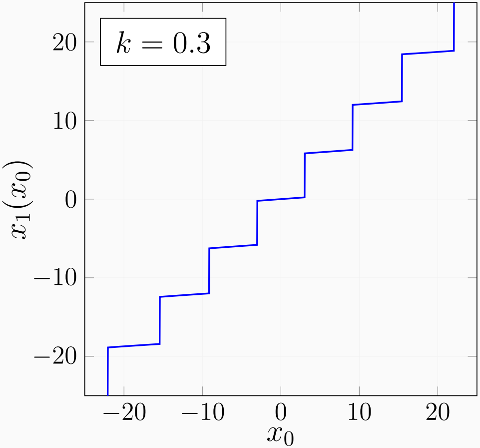


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

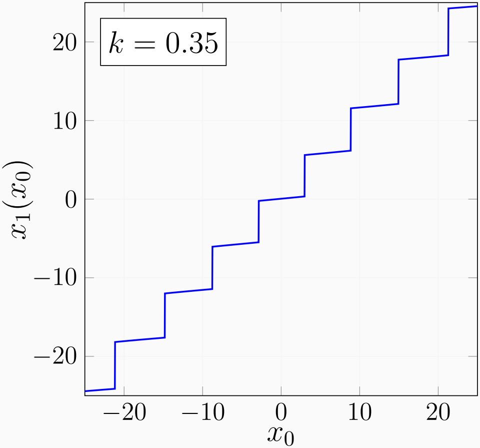


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

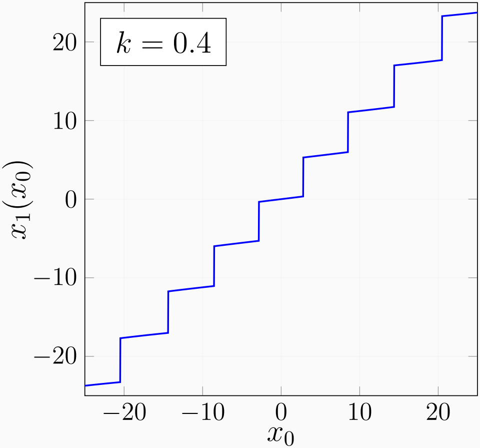


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

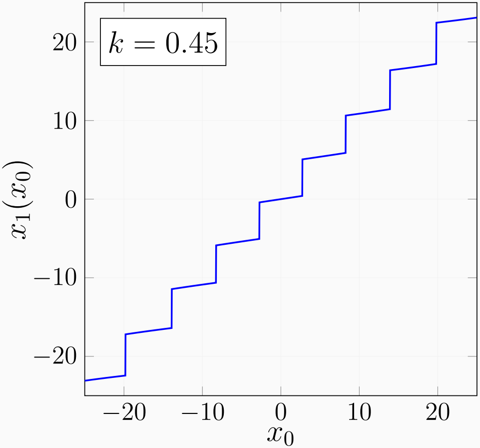


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

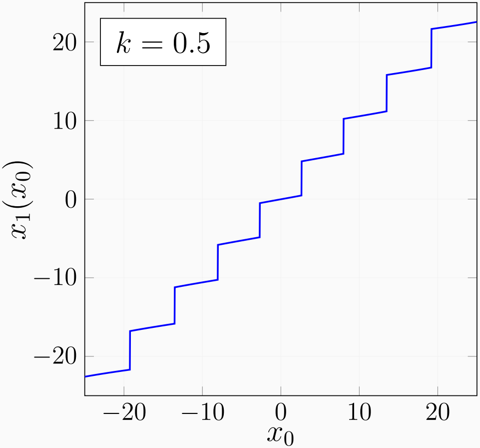


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

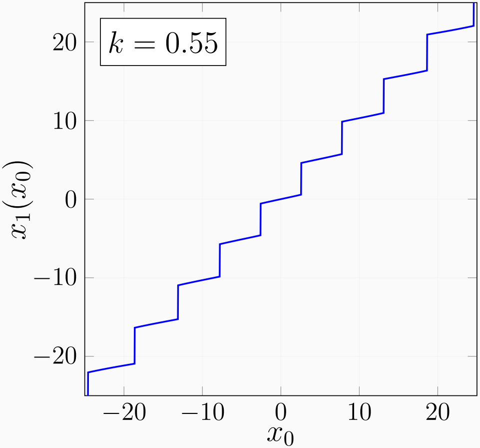


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

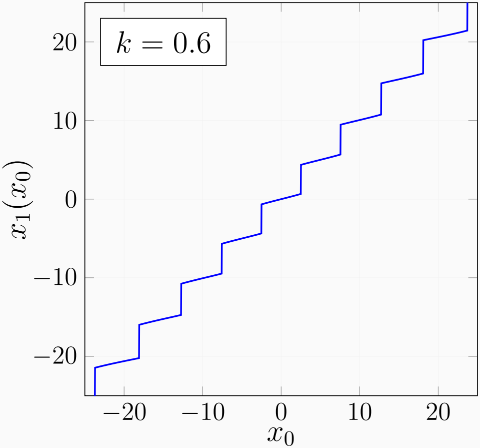


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

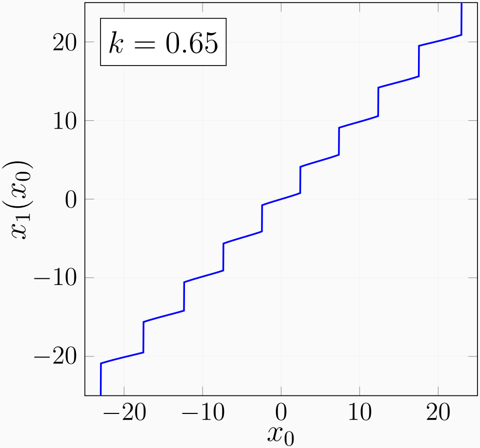


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

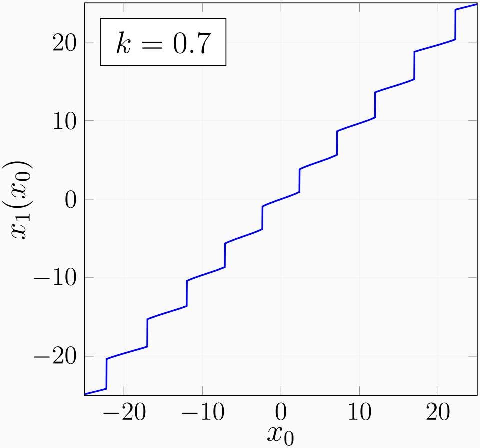


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

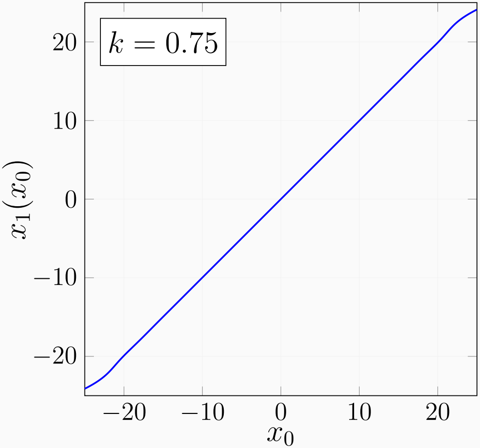


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Different Parameters

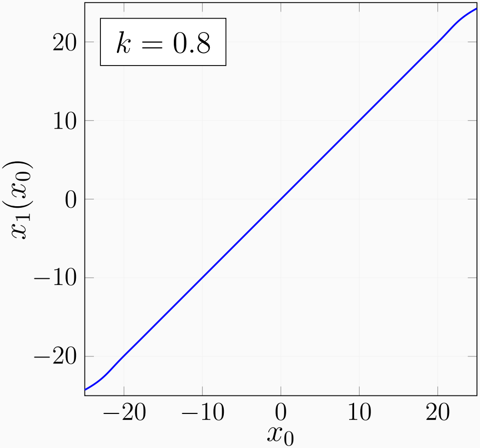


Figure 7: The resulting $x_1(x_0)$ given by the local search algorithm under different k.

Piecewise Non-linear Rather than Piecewise Affine

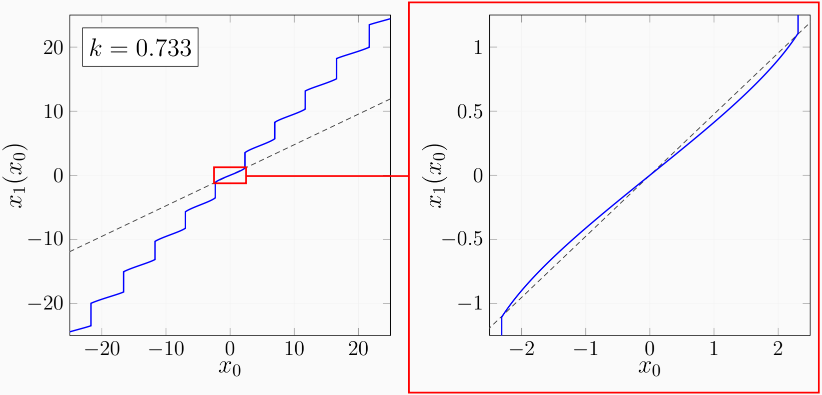
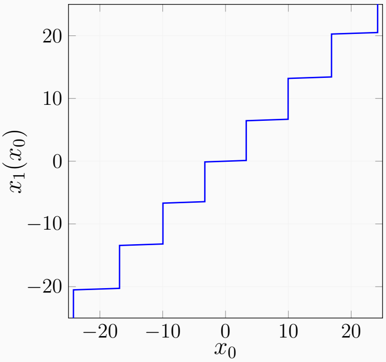
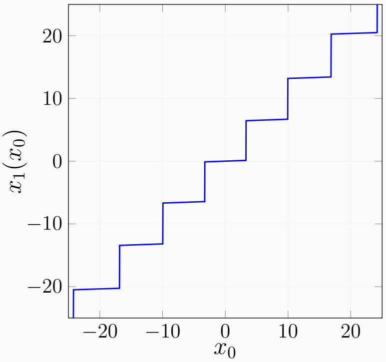


Figure 8: $x_1(x_0)$ is not piecewise affine ($k = 0.733$ as an example).



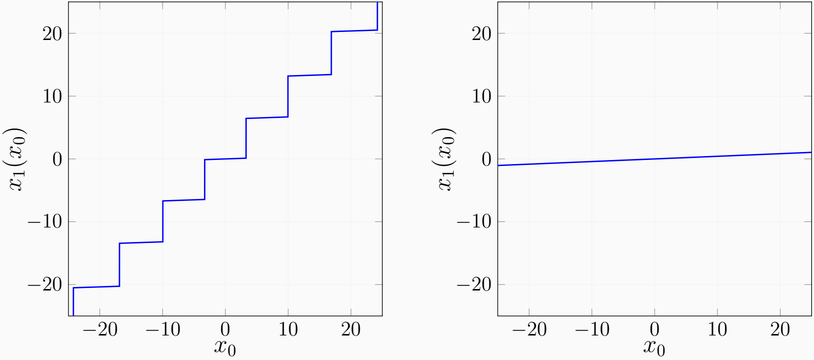
(a) Initialize $x_1(x_0) = x_0$.



(b) Initialize $x_1(x_0) = x_0|x_0|$.

Figure 9: The local search algorithm converges under different initial functions.

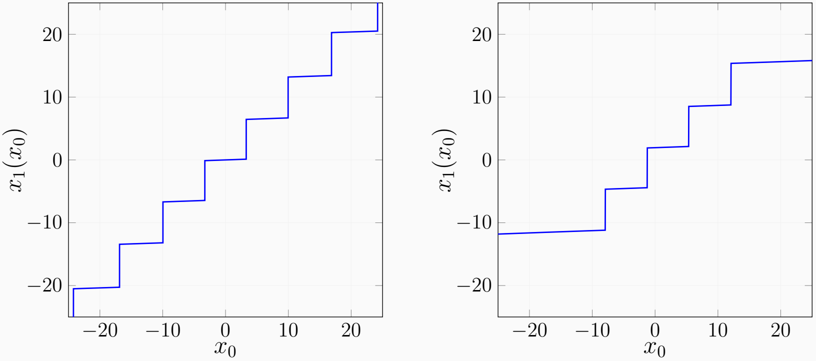
Initial Functions and Local Optima



(a) Initialize $x_1(x_0) = x_0$, resulting cost: 0.166897.

(b) Initialize $x_1(x_0) = 0$, resulting cost: 0.959991.

Figure 10: Different initial functions can still lead to different local optima.

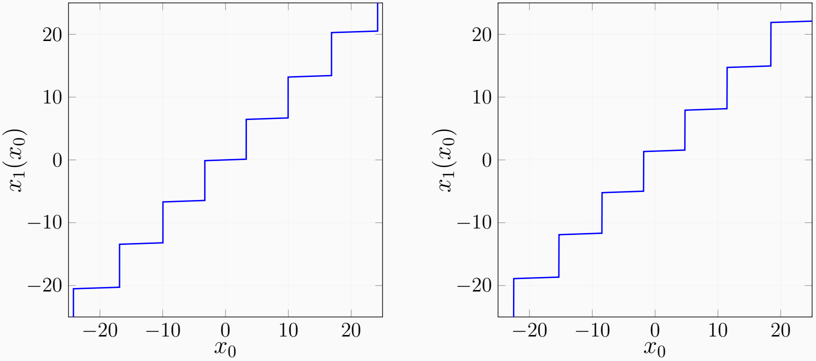


(a) Initialize $x_1(x_0) = x_0$, resulting cost: 0.166897.

(c) Initialize $x_1(x_0) = e^{x_0}$, resulting cost: 0.168075.

Figure 10: Different initial functions can still lead to different local optima.

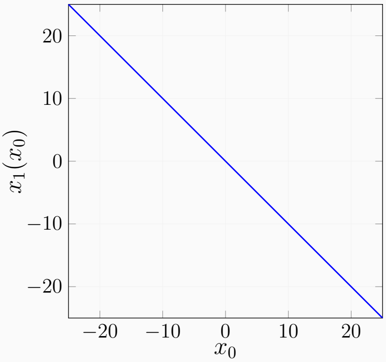
Initial Functions and Local Optima



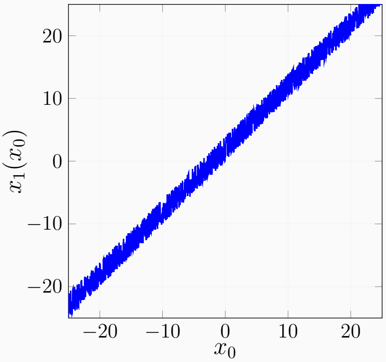
(a) Initialize $x_1(x_0) = x_0$, resulting cost: 0.166897.

(b) Initialize $x_1(x_0) = x_0 + 2$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost.



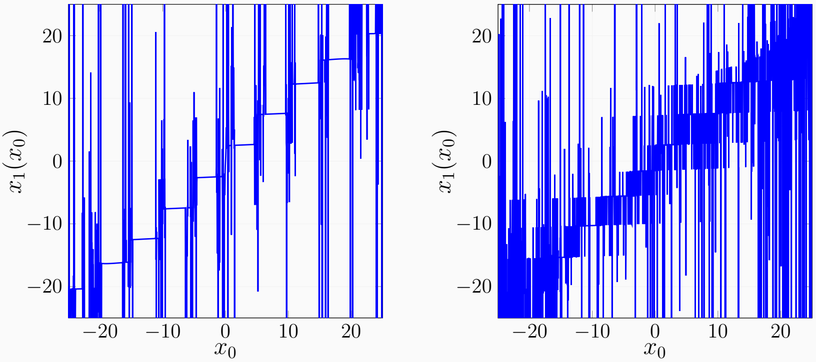
(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.



(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.

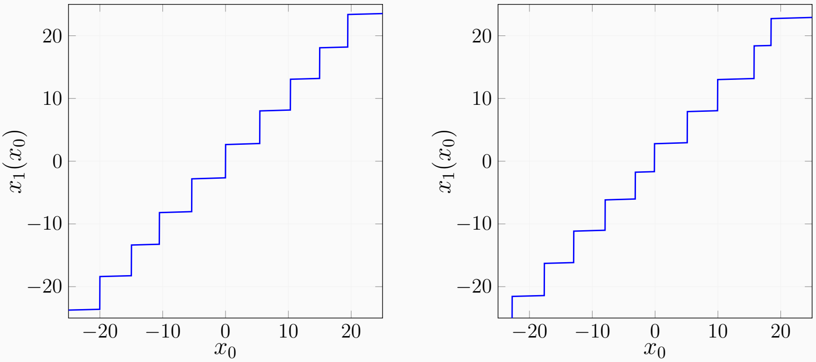
Initial Functions and Local Optima



(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.

(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

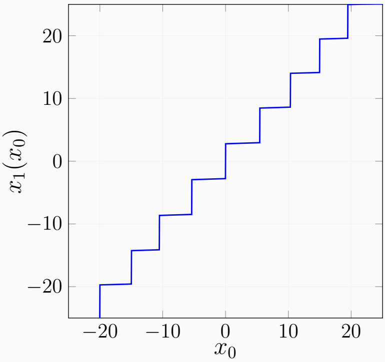
Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.



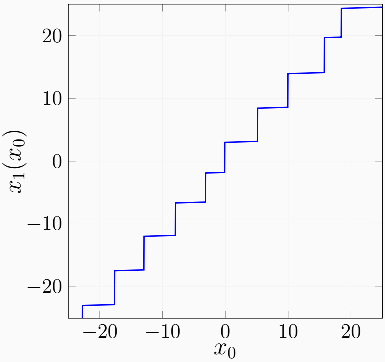
(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.

(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.



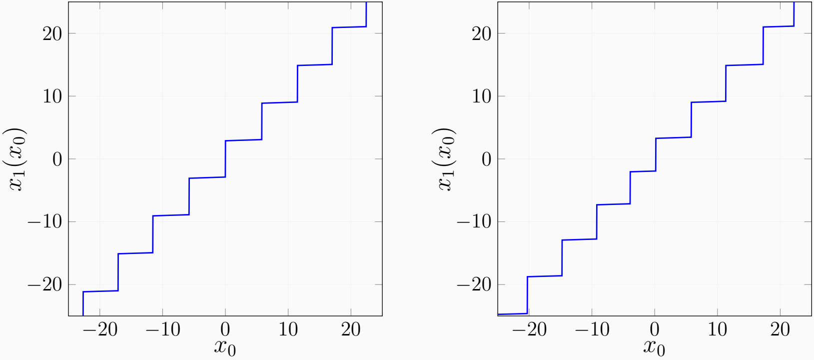
(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.



(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.

Initial Functions and Local Optima

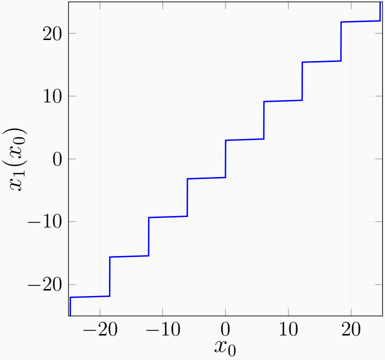


(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.

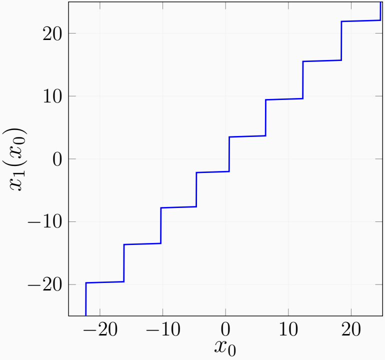
(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.

Initial Functions and Local Optima

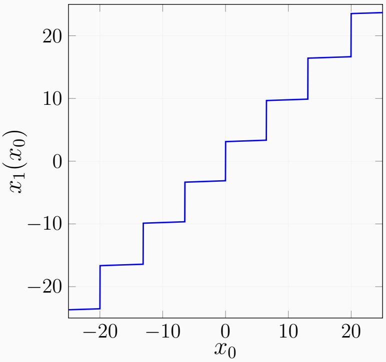


(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.

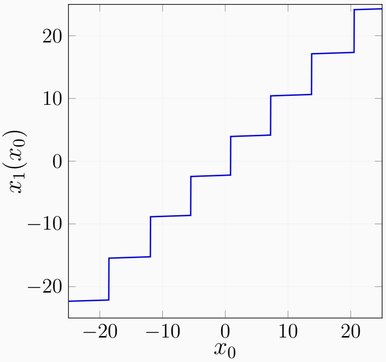


(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.



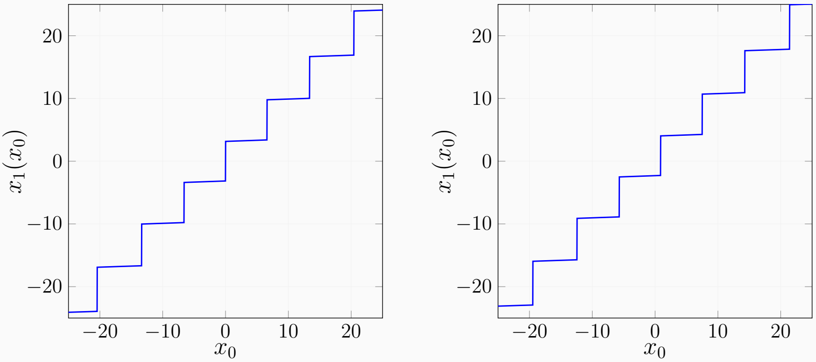
(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.



(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.

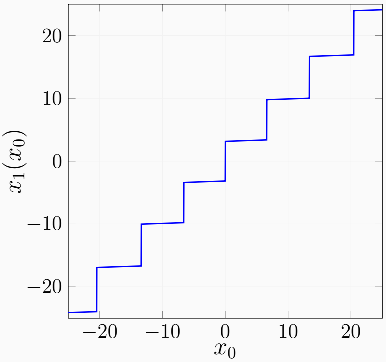
Initial Functions and Local Optima



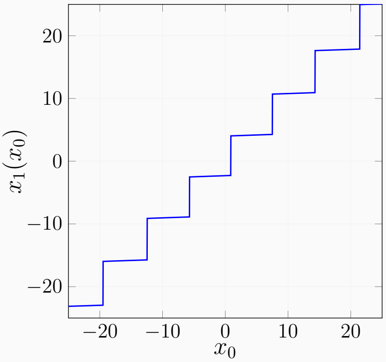
(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.

(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.



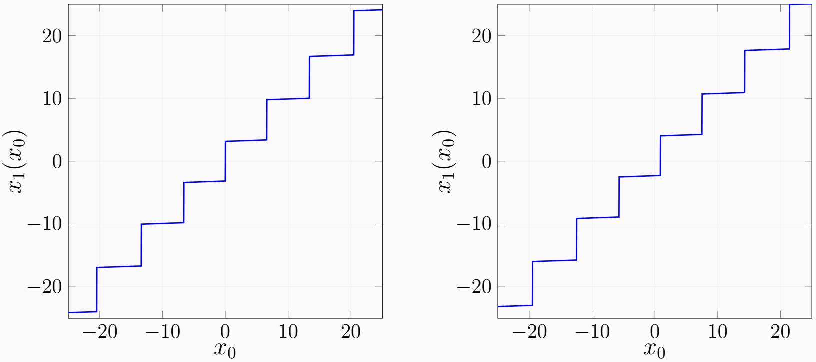
(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.



(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.

Initial Functions and Local Optima

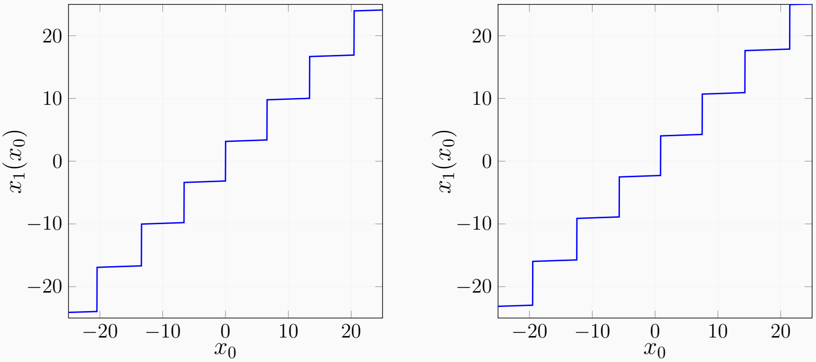


(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.

(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.

Initial Functions and Local Optima

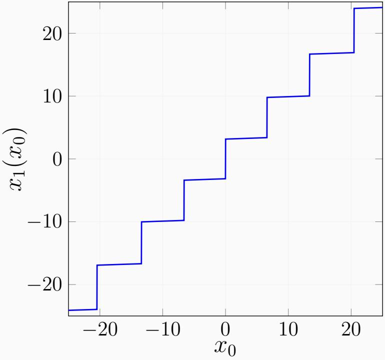


(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.

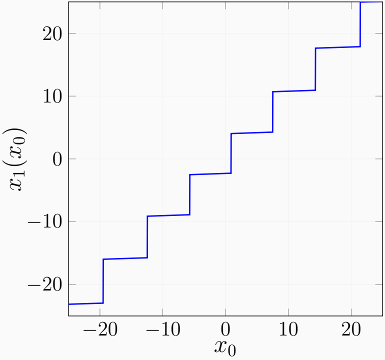
(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.

Initial Functions and Local Optima



(c) Initialize $x_1(x_0) = -x_0$, resulting cost: 0.166898.



(d) Initialize $x_1(x_0) = x_0 + w$, resulting cost: 0.166898.

Figure 11: The local search algorithm converges to local optima with similar cost, where the noise $w \sim \mathcal{U}(0, 5)$.

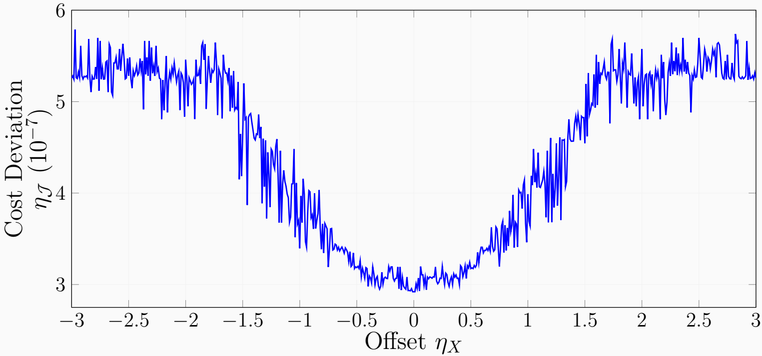
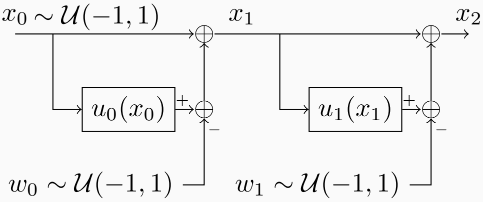


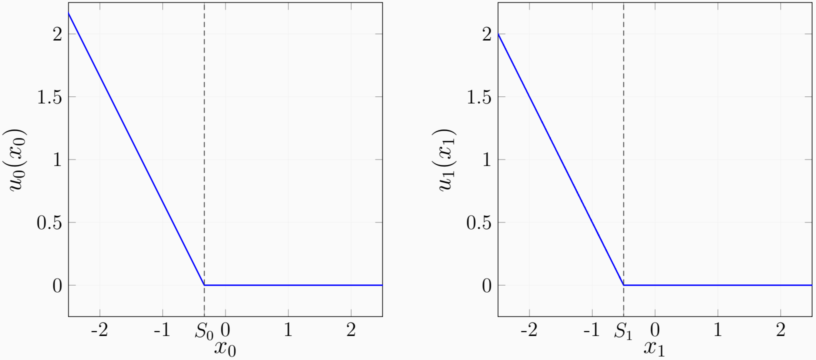
Figure 12: Initializing the local search algorithm with $x_1(x_0) = x_0 + \eta_X$ results in similar cost $\mathcal{J}[x_1, u_1] = 0.166897 + \eta_{\mathcal{J}}$.

Application to Inventory Control

- We apply the local search algorithm to the inventory control problem, which has the objective

$$\mathcal{J}\left[u_0, u_1\right] = \mathbb{E}\left[\sum_{m=0}^{1} u_m(x_m) + |x_m + u_m(x_m) - w_m|\right].$$





(a) First stage controller $u_0(x_0)$ **(b)** Second stage controller $u_1(x_1)$

Figure 13: The local search algorithm finds the optimal controllers of the inventory control problem.

Conclusion

- Instead of heuristics as in the previous attempts, we propose a local search algorithm based on two necessary conditions, which are not tied to the counterexample.
- Simulation results show that our method outperforms all existing methods on the Witsenhausen's counterexample.
- Our results also manifest some non-linear structural properties of the first stage state variable.
- Since the necessary conditions are general, our local search algorithm can be applied to other problems such as the inventory control problem.

Questions & Answers

References

- H. S. Witsenhausen, "A counterexample in stochastic optimum control," *SIAM J. Control*, vol. 6, no. 1, pp. 131–147, 1968.
- J. T. Lee, E. Lau, and Y.-C. Ho, "The Witsenhausen counterexample: A hierarchical search approach for nonconvex optimization problems," *IEEE Trans. Autom. Control*, vol. 46, no. 3, pp. 382–397, 2001.
- J. Karlsson, A. Gattami, T. J. Oechtering, and M. Skoglund, "Iterative source-channel coding approach to Witsenhausen's counterexample," in *Proc. IEEE ACC*, 2011, pp. 5348–5353.

References

M. Mehmetoglu, E. Akyol, and K. Rose, "A deterministic annealing approach to Witsenhausen's counterexample," in *Proc. IEEE ISIT*, 2014, pp. 3032–3036.

M. Baglietto, T. Parisini, and R. Zoppoli, "Numerical solutions to the Witsenhausen counterexample by approximating networks," *IEEE Trans. Autom. Control*, vol. 46, no. 9, pp. 1471–1477, 2001.