Second-Order Shape-Constrained Function Fitting

Sebastian Wild

www.wild-inter.net

based on joint work with David Durfee, Yu Gao, and Anup B. Rao

UNIVERSITY OF
WATERLOO



WADS Algorithms and Data Structures Symposium 2019

Outline

1 Introduction

2 Our Result

3 Geometric Dynamic Programming

4 Conclusion

Regression

here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.

"fit" function to points

- "regression analysis" ubiquitous in statistics / machine learning

↝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

↝ **cs challenge:** efficiently compute best f for data s. t. shape constraints

Regression

here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.
- "regression analysis" ubiquitous in statistics / machine learning

"fit" function to points

parametric regression

⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** efficiently compute best f for data s. t. shape constraints

Regression

here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.
- "regression analysis" ubiquitous in statistics / machine learning

"fit" function to points

parametric regression

⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** **efficiently** compute best f for data s. t. shape constraints

Regression

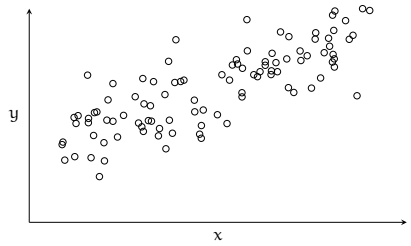
here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.

"fit" function to points

- "regression analysis" ubiquitous in statistics / machine learning

parametric regression



⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** **efficiently** compute best f for data s. t. shape constraints

Regression

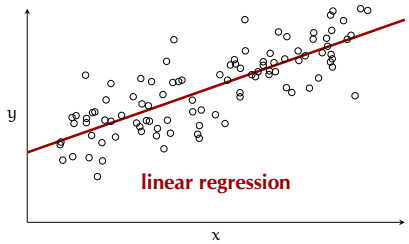
here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.

"fit" function to points

- "regression analysis" ubiquitous in statistics / machine learning

parametric regression



$$f(x) = ax + b$$

⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** efficiently compute best f for data s. t. shape constraints

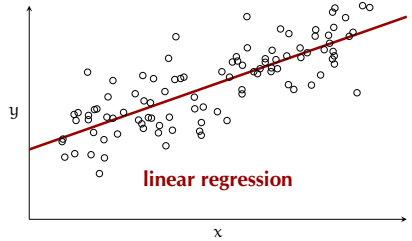
Regression

here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.
- "regression analysis" ubiquitous in statistics / machine learning

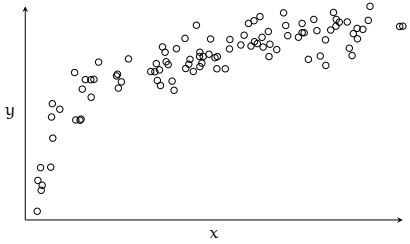
"fit" function to points

parametric regression



$$f(x) = ax + b$$

nonparametric regression



⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** **efficiently** compute best f for data s. t. shape constraints

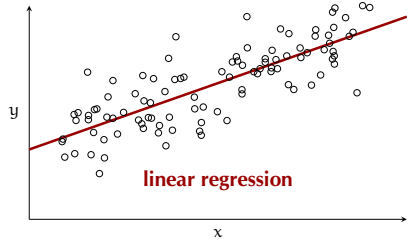
Regression

here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.
- "regression analysis" ubiquitous in statistics / machine learning

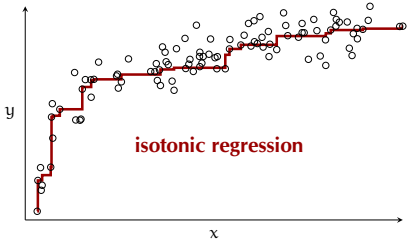
"fit" function to points

parametric regression



$f(x) = ax + b$

nonparametric regression



f = any weakly increasing function

⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** **efficiently** compute best f for data s. t. shape constraints

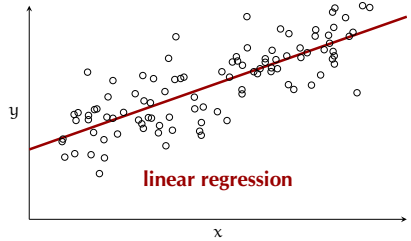
Regression

here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.
- "regression analysis" ubiquitous in statistics / machine learning

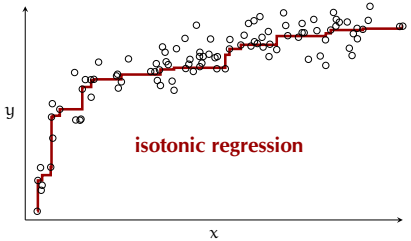
"fit" function to points

parametric regression



$f(x) = ax + b$

nonparametric regression



f = any weakly increasing function

⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** efficiently compute best f for data s. t. shape constraints

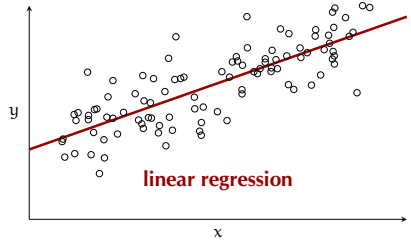
Regression

here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.
- "regression analysis" ubiquitous in statistics / machine learning

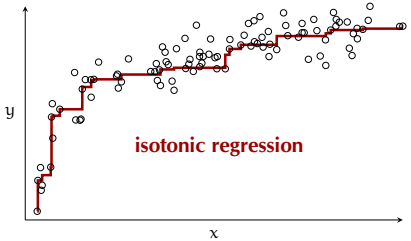
"fit" function to points

parametric regression



$f(x) = ax + b$

nonparametric regression



f = any weakly increasing function

⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** **efficiently** compute best f for data s. t. shape constraints

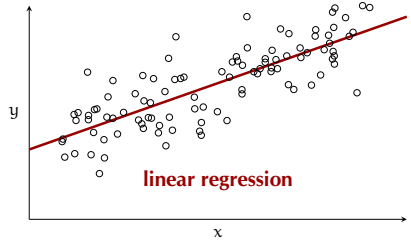
Regression

here: 2D points

- **Goal:** Given observations (x_i, y_i), find best **function** explaining $y_i \approx f(x_i)$.
- "regression analysis" ubiquitous in statistics / machine learning

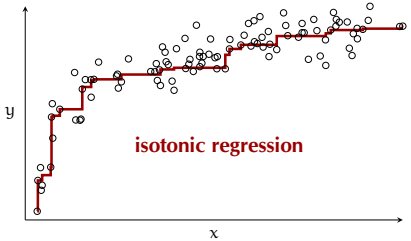
"fit" function to points

parametric regression



$f(x) = ax + b$

nonparametric regression



f = any weakly increasing function

⇝ **stats challenge:** identify reasonable shape restrictions for f (model, prior knowledge)

⇝ **cs challenge:** **efficiently** compute best f for data s. t. shape constraints

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth

Shape Constraints

Typical shape constraints for 1D functions

- (bounded)
- monotonic — cumulative density functions, low-dim. metric embedding
- unimodal — dose-response curves
- Lipschitz (bounded derivative) — avoiding overfitting in isotonic regression
- increasing & concave — production functions, utility functions (diminishing returns)
- convex — consumer preferences, value functions in reinforcement learning
- smooth



all expressible by bounds on first and second derivatives!

Weighted L_∞ Function Fitting With 2nd-Order Constraints

$$\begin{aligned} \min \quad & \text{error between } \vec{f} \text{ and } \vec{y} \\ \text{s.t.} \quad & \text{bounds on 1st derivative of } \vec{f} \\ & \text{bounds on 2nd derivative of } \vec{f} \end{aligned}$$

given quantities
variables to compute

Weighted L_∞ Function Fitting With 2nd-Order Constraints

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

s. t. bounds on 1st derivative of $\vec{f}$

bounds on 2nd derivative of $\vec{f}$

given quantities
variables to compute

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

given quantities
variables to compute

s. t. bounds on 1st derivative of $\vec{f}$

bounds on 2nd derivative of $\vec{f}$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

given quantities
variables to compute

s. t. bounds on 1st derivative of $\vec{f}$

bounds on 2nd derivative of $\vec{f}$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s.t.} \quad f_i'^{-} \leqslant f'(x_i) \leqslant f_i'^{+} \qquad (i = 2, \ldots, n)$$

given quantities
variables to compute

bounds on 2nd derivative of $\vec{f}$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s. t.} \quad f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+ \qquad (i = 2, \ldots, n)$$

given quantities
variables to compute

bounds on 2nd derivative of $\vec{f}$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s. t.} \quad f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+ \qquad (i = 2, \ldots, n)$$

discrete derivative

given quantities
variables to compute

bounds on 2nd derivative of $\vec{f}$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s. t.} \quad f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+ \qquad (i = 2, \ldots, n)$$

discrete derivative

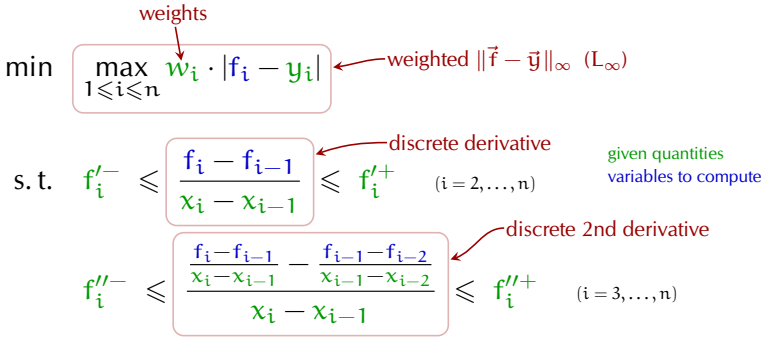
given quantities

variables to compute

$$f_i''^- \leqslant \frac{\frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{f_{i-1} - f_{i-2}}{x_{i-1} - x_{i-2}}}{x_i - x_{i-1}} \leqslant f_i''^+ \qquad (i = 3, \ldots, n)$$

discrete 2nd derivative

Weighted L_∞ Function Fitting With 2nd-Order Constraints



More convenient formulation:

$$f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+$$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s. t.} \quad f_i'^{-} \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^{+} \qquad (i = 2, \ldots, n)$$

discrete derivative

given quantities
variables to compute

$$f_i''^{-} \leqslant \frac{\frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{f_{i-1} - f_{i-2}}{x_{i-1} - x_{i-2}}}{x_i - x_{i-1}} \leqslant f_i''^{+} \qquad (i = 3, \ldots, n)$$

discrete 2nd derivative

More convenient formulation:

$$(x_i - x_{i-1}) f_i'^{-} \leqslant f_i - f_{i-1} \leqslant f_i'^{+} (x_i - x_{i-1})$$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s. t.} \quad f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+ \qquad (i = 2, \ldots, n)$$

discrete derivative

given quantities

variables to compute

$$f_i''^- \leqslant \frac{\frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{f_{i-1} - f_{i-2}}{x_{i-1} - x_{i-2}}}{x_i - x_{i-1}} \leqslant f_i''^+ \qquad (i = 3, \ldots, n)$$

discrete 2nd derivative

More convenient formulation:

$$\underbrace{(x_i - x_{i-1}) f_i'^-}_{y_i^-} \leqslant f_i - f_{i-1} \leqslant \underbrace{f_i'^+ (x_i - x_{i-1})}_{y_i^+}$$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s. t.} \quad f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+ \qquad (i = 2, \ldots, n)$$

discrete derivative

given quantities
variables to compute

$$f_i''^- \leqslant \frac{\frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{f_{i-1} - f_{i-2}}{x_{i-1} - x_{i-2}}}{x_i - x_{i-1}} \leqslant f_i''^+ \qquad (i = 3, \ldots, n)$$

discrete 2nd derivative

More convenient formulation:

$$\underbrace{(x_i - x_{i-1}) f_i'^-}_{y_i^-} \leqslant f_i - f_{i-1} \leqslant \underbrace{f_i'^+ (x_i - x_{i-1})}_{y_i^+}$$

$$\underbrace{(x_i - x_{i-1})^2 f_i''^-}_{z_i^-} \leqslant (f_i - f_{i-1}) - \underbrace{\frac{x_i - x_{i-1}}{x_{i-1} - x_{i-2}}}_{\alpha_i} \cdot (f_{i-1} - f_{i-2}) \leqslant \underbrace{f_i'^+ (x_i - x_{i-1})^2}_{z_i^+}$$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s.t.} \quad f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+ \qquad (i = 2, \ldots, n)$$

discrete derivative

given quantities
variables to compute

$$f_i''^- \leqslant \frac{\frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{f_{i-1} - f_{i-2}}{x_{i-1} - x_{i-2}}}{x_i - x_{i-1}} \leqslant f_i''^+ \qquad (i = 3, \ldots, n)$$

discrete 2nd derivative

More convenient formulation:

For this talk: $x_i = i \rightsquigarrow \alpha_i = 1$ (equidistant points)

general α in paper

$$\underbrace{(x_i - x_{i-1}) f_i'^-}_{y_i^-} \leqslant f_i - f_{i-1} \leqslant \underbrace{f_i'^+ (x_i - x_{i-1})}_{y_i^+}$$

$$\underbrace{(x_i - x_{i-1})^2 f_i''^-}_{z_i^-} \leqslant (f_i - f_{i-1}) - \underbrace{\frac{x_i - x_{i-1}}{x_{i-1} - x_{i-2}}}_{\alpha_i} \cdot (f_{i-1} - f_{i-2}) \leqslant \underbrace{f_i'^+ (x_i - x_{i-1})^2}_{z_i^+}$$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s. t.} \quad f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+ \qquad (i = 2, \ldots, n)$$

discrete derivative

given quantities
variables to compute

$$f_i''^- \leqslant \frac{\frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{f_{i-1} - f_{i-2}}{x_{i-1} - x_{i-2}}}{x_i - x_{i-1}} \leqslant f_i''^+ \qquad (i = 3, \ldots, n)$$

discrete 2nd derivative

More convenient formulation:

For this talk: $x_i = i \rightsquigarrow \alpha_i = 1$ (equidistant points)

general α in paper

$$\underbrace{(x_i - x_{i-1}) f_i'^-}_{y_i^-} \leqslant f_i - f_{i-1} \leqslant \underbrace{f_i'^+ (x_i - x_{i-1})}_{y_i^+}$$

$$\underbrace{(x_i - x_{i-1})^2 f_i''^-}_{z_i^-} \leqslant (f_i - f_{i-1}) - \underbrace{\frac{x_i - x_{i-1}}{x_{i-1} - x_{i-2}}}_{\alpha_i} \cdot (f_{i-1} - f_{i-2}) \leqslant \underbrace{f_i'^+ (x_i - x_{i-1})^2}_{z_i^+}$$

Weighted L_∞ Function Fitting With 2nd-Order Constraints

weights

$$\min \quad \max_{1 \leqslant i \leqslant n} w_i \cdot |f_i - y_i|$$

weighted $\|\vec{f} - \vec{y}\|_\infty$ (L_∞)

$$\text{s. t.} \quad f_i'^- \leqslant \frac{f_i - f_{i-1}}{x_i - x_{i-1}} \leqslant f_i'^+ \qquad (i = 2, \ldots, n)$$

discrete derivative

given quantities
variables to compute

$$f_i''^- \leqslant \frac{\frac{f_i - f_{i-1}}{x_i - x_{i-1}} - \frac{f_{i-1} - f_{i-2}}{x_{i-1} - x_{i-2}}}{x_i - x_{i-1}} \leqslant f_i''^+ \qquad (i = 3, \ldots, n)$$

discrete 2nd derivative

More convenient formulation:

For this talk: $x_i = i \rightsquigarrow \alpha_i = 1$ (equidistant points)

general α in paper

$$y_i^- \leqslant f_i - f_{i-1} \leqslant y_i^+$$

$$z_i^- \leqslant (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) \leqslant z_i^+$$

Previous Work

Shape Constraint	Error Norm	Time	Source
linear function	weighted L_2	$O(n)$	[linear least squares]
linear comb. of k fixed functions	weighted L_2	$O(nk^2)$	[linear least squares]
monotonic (isotonic regr. on path)	weighted L_2	$O(n)$	[pool adjacent violators algorithm, Rote 2019]
monotonic	weighted L_1	$O(n \log n)$	[Ahuja&Orlin 2001, Rote 2019]
monotonic	weighted L_∞	$O(n)$	[Stout 2015]
isotonic regr. on general partial orders	*various results surveyed*		[Stout 2019]
unimodal	weighted L_2	$O(n)$	[Stout 2008]
unimodal	weighted L_1	$O(n \log n)$	[Stout 2008]
unimodal	L_∞	$O(n)$	[Stout 2008]
piecewise constant (k-step function)	weighted L_∞	$O(n \log n)$	[Fournier&Vigneron 2013]
piecewise constant (k-step function)	L_∞	$O(n)$ sorted input	[Fournier&Vigneron 2011]
Lipschitz & monotonic	L_2	$O(n \log n)$	[Agarwal&Phillips&Sadri 2010]
Lipschitz & unimodal	L_2	$O(n \operatorname{polylog} n)$	[Agarwal&Phillips&Sadri 2010]
multivar. convex (& Lipschitz)	L_2	quadr. prog. $\Theta(n^2)$ constraints	[Mazumder et al. 2018]

Previous Work

Summary:

- **Near-linear** time exact methods for **first-order** constraints
- Best prior approach for **second-order**: **solve LP**

Shape Constraint			
linear function	weighted L_2	$O(n)$	[linear least squares]
linear comb. of k fixed functions	weighted L_2	$O(nk^2)$	[linear least squares]
monotonic (isotonic regr. on path)	weighted L_2	$O(n)$	[pool adjacent violators algorithm, Rote 2019]
monotonic	weighted L_1	$O(n \log n)$	[Ahuja&Orlin 2001, Rote 2019]
monotonic	weighted L_∞	$O(n)$	[Stout 2015]
isotonic regr. on general partial orders	*various results surveyed*		[Stout 2019]
unimodal	weighted L_2	$O(n)$	[Stout 2008]
unimodal	weighted L_1	$O(n \log n)$	[Stout 2008]
unimodal	L_∞	$O(n)$	[Stout 2008]
piecewise constant (k-step function)	weighted L_∞	$O(n \log n)$	[Fournier&Vigneron 2013]
piecewise constant (k-step function)	L_∞	$O(n)$ sorted input	[Fournier&Vigneron 2011]
Lipschitz & monotonic	L_2	$O(n \log n)$	[Agarwal&Phillips&Sadri 2010]
Lipschitz & unimodal	L_2	$O(n \operatorname{polylog} n)$	[Agarwal&Phillips&Sadri 2010]
multivar. convex (& Lipschitz)	L_2	quadr. prog. $\Theta(n^2)$ constraints	[Mazumder et al. 2018]

Previous Work

Summary:

- **Near-linear** time exact methods for **first-order** constraints
- Best prior approach for **second-order**: **solve LP**

Shape Constraint			
linear function	weighted L_2	$O(n)$	[linear least squares]
linear comb. of k fixed functions	weighted L_2	$O(nk^2)$	[linear least squares]
monotonic (isotonic regr. on path)	weighted L_2	$O(n)$	[pool adjacent violators algorithm, Rote 2019]
monotonic	weighted L_1	$O(n \log n)$	[Ahuja&Orlin 2001, Rote 2019]
monotonic	weighted L_∞	$O(n)$	[Stout 2015]
isotonic regr. on general partial orders	*various results surveyed*		[Stout 2019]
unimodal	weighted L_2	$O(n)$	[Stout 2008]
unimodal	weighted L_1	$O(n \log n)$	[Stout 2008]
unimodal	L_∞	$O(n)$	[Stout 2008]
piecewise constant (k-step function)	weighted L_∞	$O(n \log n)$	[Fournier&Vigneron 2013]
piecewise constant (k-step function)	L_∞	$O(n)$ sorted input	[Fournier&Vigneron 2011]
Lipschitz & monotonic	L_2	$O(n \log n)$	[Agarwal&Phillips&Sadri 2010]
Lipschitz & unimodal	L_2	$O(n \operatorname{polylog} n)$	[Agarwal&Phillips&Sadri 2010]
multivar. convex (& Lipschitz)	L_2	quadr. prog. $\Theta(n^2)$ constraints	[Mazumder et al. 2018]

Previous Work

Summary:

- **Near-linear** time exact methods for **first-order** constraints
- Best prior approach for **second-order**: **solve LP**

Shape Constraint			
linear function	weighted L_2	$O(n)$	[linear least squares]
linear comb. of k fixed functions	weighted L_2	$O(nk^2)$	[linear least squares]
monotonic (isotonic regr. on path)	weighted L_2	$O(n)$	[pool adjacent violators algorithm, Rote 2019]
monotonic	weighted L_1	$O(n \log n)$	[Ahuja&Orlin 2001, Rote 2019]
monotonic	weighted L_∞	$O(n)$	[Stout 2015]
isotonic regr. on general partial orders	*various results surveyed*		[Stout 2019]
unimodal	weighted L_2	$O(n)$	[Stout 2008]
unimodal	weighted L_1	$O(n \log n)$	[Stout 2008]
unimodal	L_∞	$O(n)$	[Stout 2008]
piecewise constant (k-step function)	weighted L_∞	$O(n \log n)$	[Fournier&Vigneron 2013]
piecewise constant (k-step function)	L_∞	$O(n)$ sorted input	[Fournier&Vigneron 2011]
Lipschitz & monotonic	L_2	$O(n \log n)$	[Agarwal&Phillips&Sadri 2010]
Lipschitz & unimodal	L_2	$O(n \operatorname{polylog} n)$	[Agarwal&Phillips&Sadri 2010]
multivar. convex (& Lipschitz)	L_2	quadr. prog. $\Theta(n^2)$ constraints	[Mazumder et al. 2018]

Outline

1 Introduction

2 Our Result

3 Geometric Dynamic Programming

4 Conclusion

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \quad \leqslant \quad \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $\mathbf{O}\left(n \log\left(\frac{U}{\varepsilon}\right)\right)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \leqslant \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $O\left(n \log\left(\frac{U}{\varepsilon}\right)\right)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Overview:



- $O(n)$ alg. for **decision** version:
  Given L, $\exists \vec{f}$ with error $\leqslant L$?
- **Binary search** for smallest L
  $\rightsquigarrow O\left(\log\left(\frac{U}{\varepsilon}\right)\right)$ iterations

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \leqslant \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $O\left(n \log\left(\frac{U}{\varepsilon}\right)\right)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Overview:



- $O(n)$ alg. for **decision** version:
  Given L, $\exists \vec{f}$ with error $\leqslant L$?
- **Binary search** for smallest L
  $\rightsquigarrow O\left(\log\left(\frac{U}{\varepsilon}\right)\right)$ iterations

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \leqslant \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $O\big(n \log(\frac{U}{\varepsilon})\big)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Overview:



- $O(n)$ alg. for **decision** version:
  Given L, $\exists \vec{f}$ with error $\leqslant L$?
- **Binary search** for smallest L
  $\rightsquigarrow O\big(\log(\frac{U}{\varepsilon})\big)$ iterations

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \leqslant \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $O\big(n \log(\frac{U}{\varepsilon})\big)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Overview:



- $O(n)$ alg. for **decision** version:
  Given L, $\exists \vec{f}$ with error $\leqslant L$?
- **Binary search** for smallest L
  $\rightsquigarrow O\big(\log(\frac{U}{\varepsilon})\big)$ iterations

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \quad \leqslant \quad \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $\mathbf{O}\big(n \log(\frac{U}{\varepsilon})\big)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Overview:



- $O(n)$ alg. for **decision** version:
  Given L, $\exists \vec{f}$ with error $\leqslant L$?
- **Binary search** for smallest L
  $\rightsquigarrow O\big(\log(\frac{U}{\varepsilon})\big)$ iterations

2nd-order-diff constrained decision problem:

$$\begin{array}{lcccl} & \exists \vec{f} \text{ s. t.} & & & \\ x_i^- & \leqslant & f_i & \leqslant & x_i^+ \quad i = 1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ \quad i = 2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ \quad i = 3..n \end{array}$$

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \quad \leqslant \quad \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $O\big(n \log(\frac{U}{\varepsilon})\big)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Overview:



- $O(n)$ alg. for **decision** version:
  Given L, $\exists \vec{f}$ with error $\leqslant L$?
- **Binary search** for smallest L
  $\rightsquigarrow O\big(\log(\frac{U}{\varepsilon})\big)$ iterations

This talk: only Yes/No
But: can find $\vec{f}$ by backtracing ($\rightsquigarrow$ paper)

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t.}$$

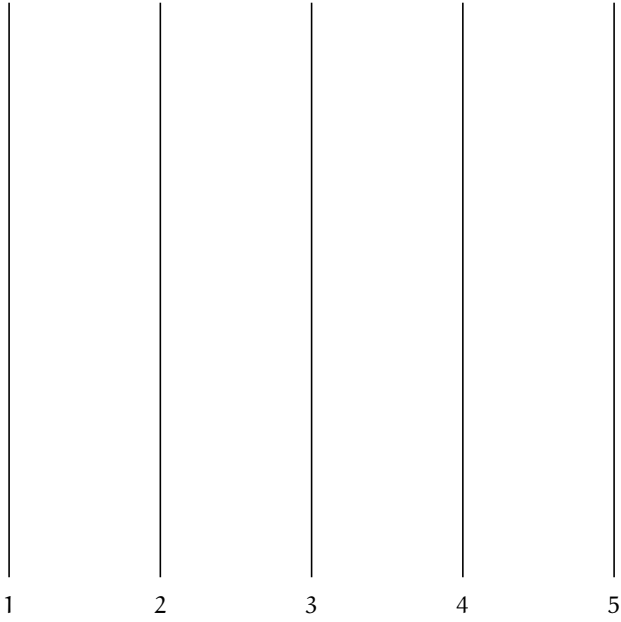
$$\begin{array}{rcccll}
x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i = 1..n \\
y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i = 2..n \\
z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i = 3..n
\end{array}$$

Dynamic Programming with Infinite States?

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

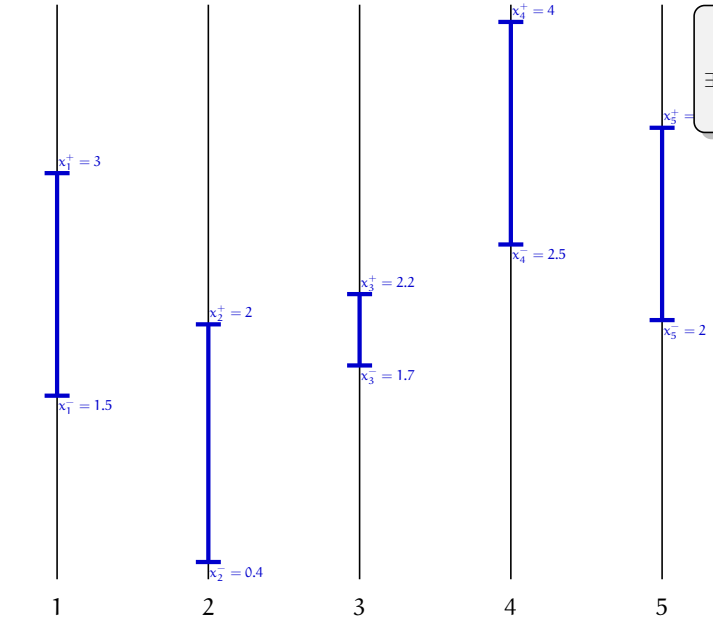
Dynamic Programming with Infinite States?



2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

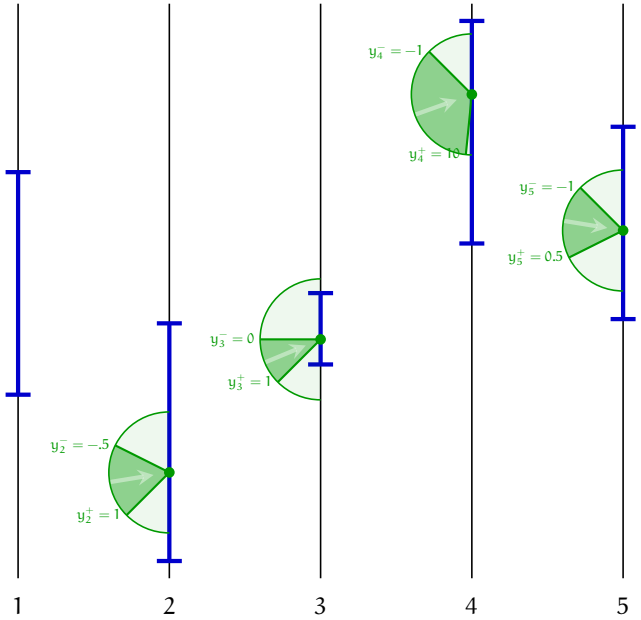
Dynamic Programming with Infinite States?



2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

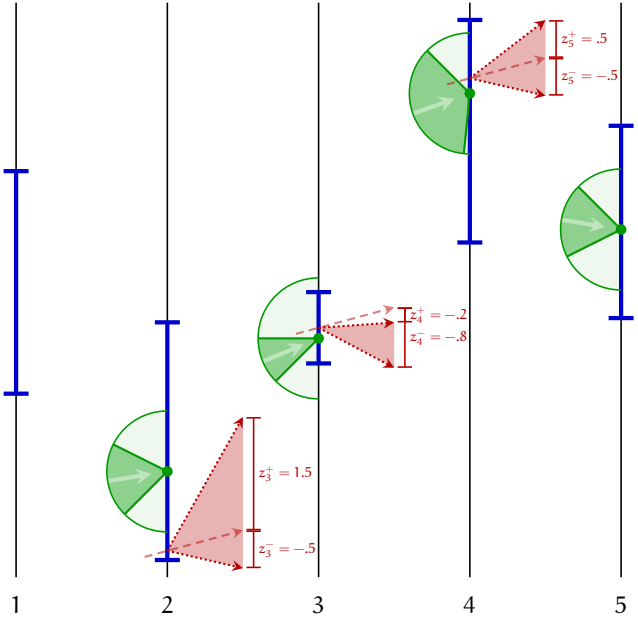
Dynamic Programming with Infinite States?



2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccl} x_i^- & \leqslant & f_i & \leqslant & x_i^+ \quad i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ \quad i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ \quad i=3..n \end{array}$$

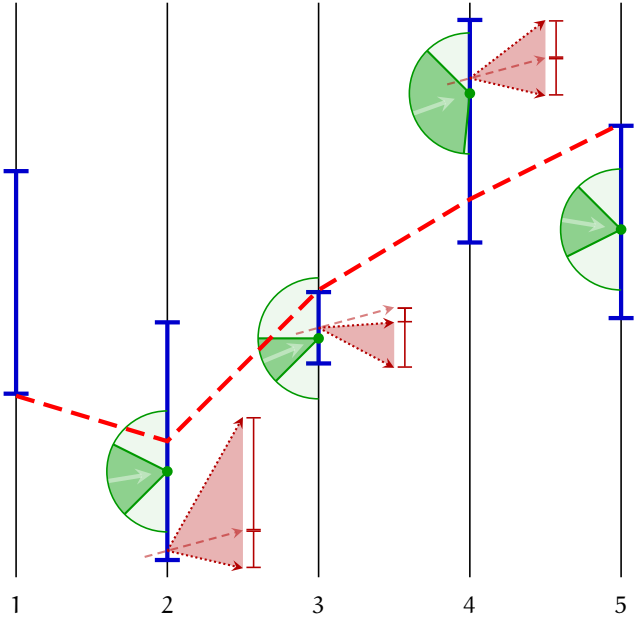
Dynamic Programming with Infinite States?



2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i = 1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i = 2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i = 3..n \end{array}$$

Dynamic Programming with Infinite States?

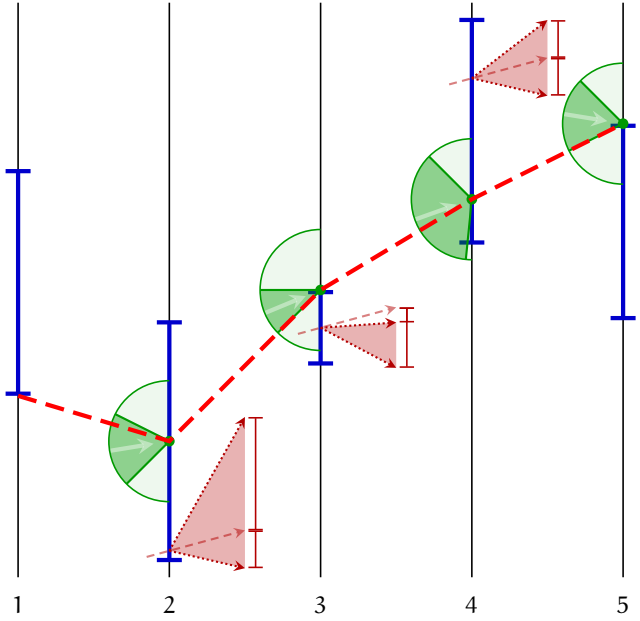


2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

Dynamic Programming with Infinite States?

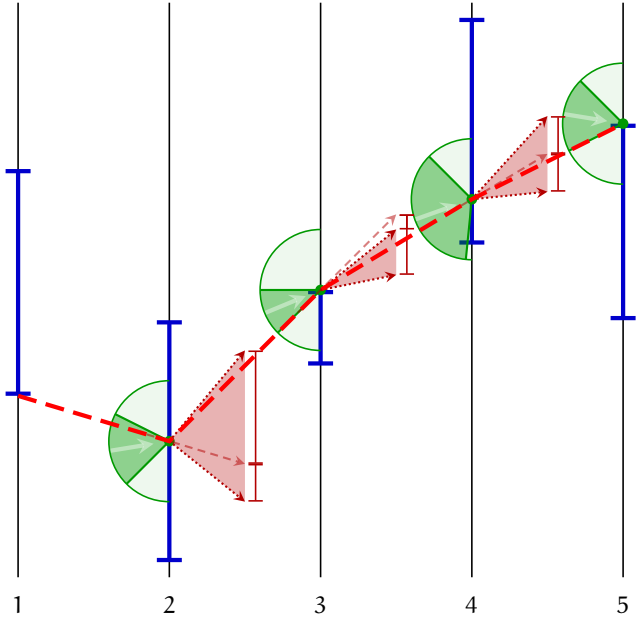


2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

Dynamic Programming with Infinite States?

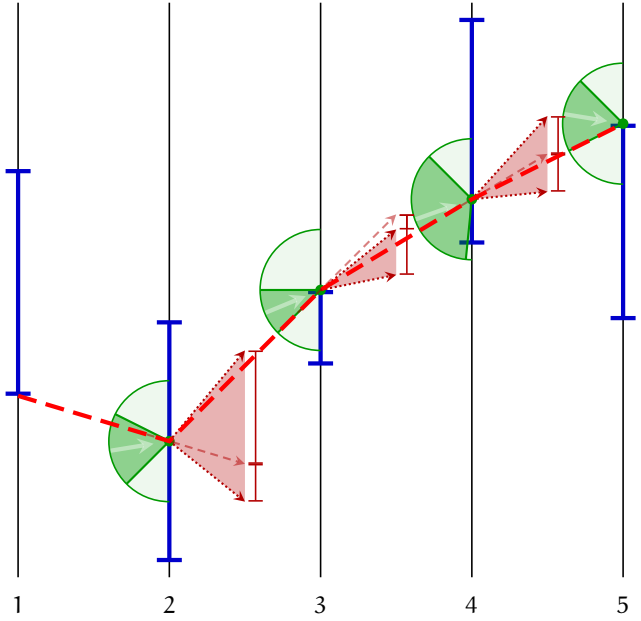


2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

Dynamic Programming with Infinite States?



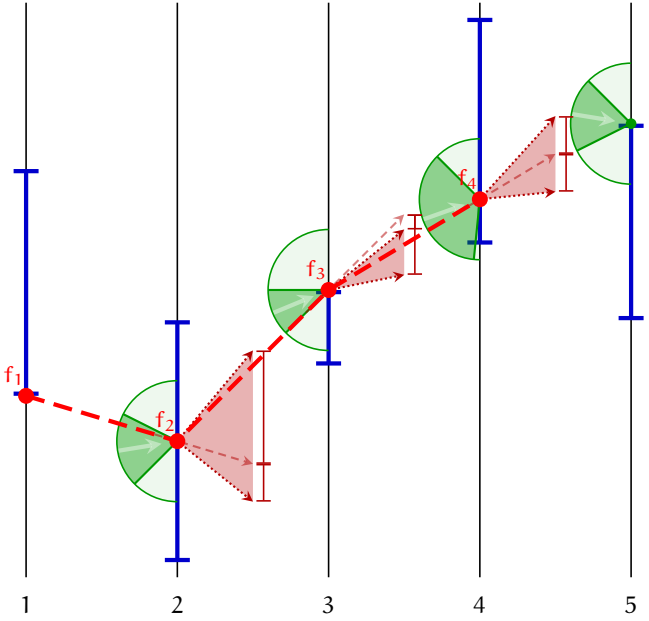
2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

- **"Induction"**
  - Suppose **candidate** values for $f_1, \ldots, f_{n-1}$ fixed
  - Need only f_{n-2} and f_{n-1} to decide if feasible f_n exists (for this candidate).

Dynamic Programming with Infinite States?



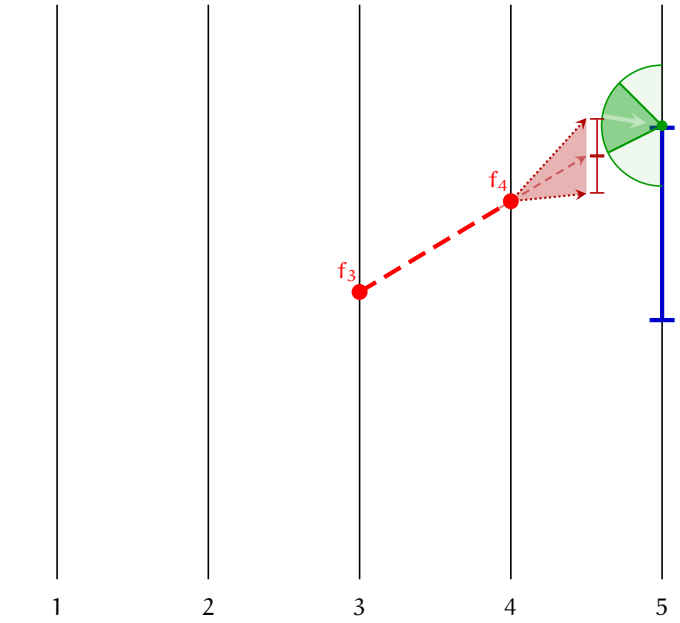
2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

- **"Induction"**
  - Suppose **candidate** values for $f_1, \ldots, f_{n-1}$ fixed
  - Need only f_{n-2} and f_{n-1} to decide if feasible f_n exists (for this candidate).

Dynamic Programming with Infinite States?



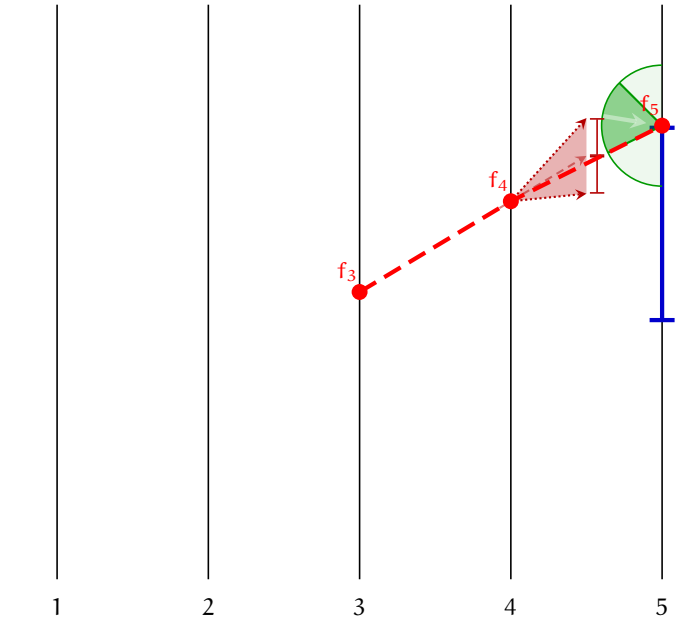
2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

- **"Induction"**
  - Suppose **candidate** values for $f_1, \ldots, f_{n-1}$ fixed
  - Need only f_{n-2} and f_{n-1} to decide if feasible f_n exists (for this candidate).

Dynamic Programming with Infinite States?



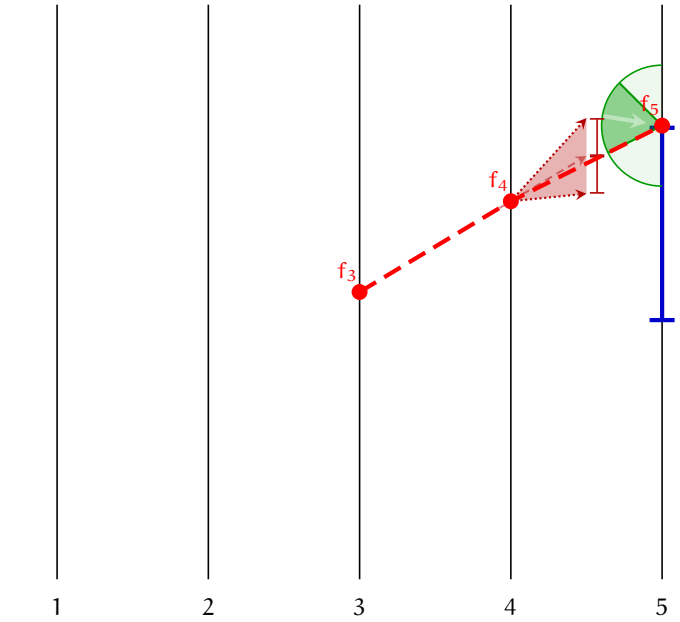
2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

- **"Induction"**
  - Suppose **candidate** values for $f_1, \ldots, f_{n-1}$ fixed
  - Need only f_{n-2} and f_{n-1} to decide if feasible f_n exists (for this candidate).

Dynamic Programming with Infinite States?



2nd-order-diff constrained decision problem:

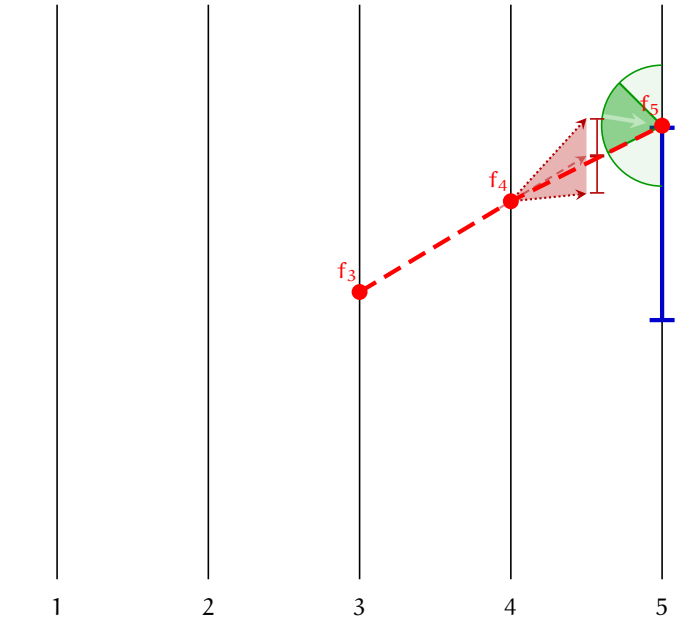
$$\exists \vec{f} \text{ s. t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

- **"Induction"**
  - Suppose **candidate** values for $f_1, \ldots, f_{n-1}$ fixed
  - Need only f_{n-2} and f_{n-1} to decide if feasible f_n exists (for this candidate).

$\rightsquigarrow$ **Dynamic programming** over prefixes of problem!

Dynamic Programming with Infinite States?



2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{rcccll} x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

(This is a 'Yes' instance.)

- **"Induction"**
  - Suppose **candidate** values for $f_1, \ldots, f_{n-1}$ fixed
  - Need only f_{n-2} and f_{n-1} to decide if feasible f_n exists (for this candidate).

⇝ **Dynamic programming** over prefixes of problem!

Continuous problem! Infinitely many candidates $\vec{f}$!

Outline

1 Introduction

2 Our Result

3 Geometric Dynamic Programming

4 Conclusion

Dynamic Programming with Infinite States!

There is a **finite** representation
for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t. } \begin{array}{lccccl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i=3..n \end{array}$$

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcccl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i & \leqslant & x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ \quad i=3..n \end{array}$$

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcccll} & (\mathcal{X}_i) & x_i^- & \leqslant & f_i & \leqslant x_i^+ & i=1..n \\ & (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} & \leqslant y_i^+ & i=2..n \\ & (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant z_i^+ & i=3..n \end{array}$$

$$\left\{ (f_i, \quad f_{i-1} \quad) : \exists (f_1, \ldots, f_{i-2}) \ \forall j \leq i \ (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcccl} & (\mathcal{X}_i) & x_i^- & \leqslant & f_i & \leqslant x_i^+ & i=1..n \\ & (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} & \leqslant y_i^+ & i=2..n \\ & (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant z_i^+ & i=3..n \end{array}$$

$$\left\{ (f_i, \quad f_{i-1} \quad) \,:\, \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i \leqslant x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} \leqslant y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) \leqslant z_i^+ \quad i=3..n \end{array}$$

last value

$$\left\{ (f_i, \quad f_{i-1} \quad) : \exists (f_1, \ldots, f_{i-2}) \ \forall j \leq i \ (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcccl} & (\mathcal{X}_i) & x_i^- & \leqslant & f_i & \leqslant x_i^+ & i=1..n \\ & (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} & \leqslant y_i^+ & i=2..n \\ & (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant z_i^+ & i=3..n \end{array}$$

last value — incoming slope

$$\left\{ (f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i \leqslant x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} \leqslant y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) \leqslant z_i^+ \quad i=3..n \end{array}$$

last value — incoming slope

$$P_i := \left\{ (f_i, f_i - f_{i-1}) \,:\, \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i \leqslant x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} \leqslant y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) \leqslant z_i^+ \quad i=3..n \end{array}$$

last value — incoming slope

$$P_i := \left\{ (f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

- *2nd-order-diff constrained decision problem* $\equiv$ $P_n \neq \{\}$?

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcccl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i & \leqslant & x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ \quad i=3..n \end{array}$$

last value — incoming slope

$$P_i := \left\{ (f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

- *2nd-order-diff constrained decision problem* $\equiv$ $P_n \neq \{\}$?
- $P_2, \ldots, P_n$ are all **convex polygons**. (defined by linear inequalities)

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcccl} & (\mathcal{X}_i) & x_i^- & \leqslant & f_i & \leqslant x_i^+ & i=1..n \\ & (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} & \leqslant y_i^+ & i=2..n \\ & (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant z_i^+ & i=3..n \end{array}$$

last value — incoming slope

$$P_i := \left\{ (f_i, f_i - f_{i-1}) \,:\, \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

- *2nd-order-diff constrained decision problem* $\equiv$ $P_n \neq \{\}$?
- $P_2, \ldots, P_n$ are all **convex polygons**. (defined by linear inequalities)
  - ⇝ Can represent them by their **vertex sequence**!

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcccl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i & \leqslant & x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ \quad i=3..n \end{array}$$

last value — incoming slope

$$P_i := \left\{ (f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

- *2nd-order-diff constrained decision problem* $\equiv$ $P_n \neq \{\}$?
- $P_2, \ldots, P_n$ are all **convex polygons**. (defined by linear inequalities)
  - $\rightsquigarrow$ Can represent them by their **vertex sequence**!
    - P_2 can be computed directly.

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i \leqslant x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} \leqslant y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) \leqslant z_i^+ \quad i=3..n \end{array}$$

last value — incoming slope

$$P_i := \left\{ (f_i, f_i - f_{i-1}) \,:\, \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

- *2nd-order-diff constrained decision problem* $\equiv$ $P_n \neq \{\}$?
- $P_2, \ldots, P_n$ are all **convex polygons**. (defined by linear inequalities)
  - ⇝ Can represent them by their **vertex sequence**!
    - P_2 can be computed directly.



How to **efficiently** compute P_i from P_{i-1}?

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i \leqslant x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} \leqslant y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) \leqslant z_i^+ \quad i=3..n \end{array}$$

last value — incoming slope

$$P_i := \left\{ (f_i, f_i - f_{i-1}) \,:\, \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

- *2nd-order-diff constrained decision problem* $\equiv$ $P_n \neq \{\}$?
- $P_2, \ldots, P_n$ are all **convex polygons**. (defined by linear inequalities)
  - ⇝ Can represent them by their **vertex sequence**!
    - P_2 can be computed directly.



How to **efficiently** compute P_i from P_{i-1}?

. . . by the magic of geometry and data structures!

Dynamic Programming with Infinite States!



There is a **finite** representation for **all** prefix **candidates**

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s.t. } \begin{array}{lrcl} (\mathcal{X}_i) & x_i^- & \leqslant & f_i \leqslant x_i^+ \quad i=1..n \\ (\mathcal{Y}_i) & y_i^- & \leqslant & f_i - f_{i-1} \leqslant y_i^+ \quad i=2..n \\ (\mathcal{Z}_i) & z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) \leqslant z_i^+ \quad i=3..n \end{array}$$

last value — incoming slope

$$P_i := \left\{ (f_i, f_i - f_{i-1}) \,:\, \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leq i \;\; (\mathcal{X}_j) \wedge (\mathcal{Y}_j) \wedge (\mathcal{Z}_j) \right\}$$

possible last components of prefix solutions

- *2nd-order-diff constrained decision problem* $\equiv$ $P_n \neq \{\}$?
- $P_2, \ldots, P_n$ are all **convex polygons**. (defined by linear inequalities)
  - $\rightsquigarrow$ Can represent them by their **vertex sequence**!
    - P_2 can be computed directly.



How to **efficiently** compute P_i from P_{i-1}?

. . . by the magic of geometry and data structures!
$\rightsquigarrow$ stay tuned

From P_{i-1} to P_i – Step 1

$$P_i = \left\{ (f_i, f_i - f_{i-1}) \,:\, \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leqslant i \;\; (x_j), (y_j), (z_j) \right\}$$

From P_{i-1} to P_i – Step 1

$$
\begin{aligned}
P_i &= \left\{ (f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leqslant i \;\; (\mathcal{X}_j), (\mathcal{Y}_j), (\mathcal{Z}_j) \right\} \\
&= \left\{ (f_i, f_i - f_{i-1}) \;:\; \exists f_{i-2} \;\; (f_{i-1}, f_{i-1} - f_{i-2}) \in P_{i-1}, \;\; (\mathcal{X}_i), (\mathcal{Y}_i), (\mathcal{Z}_i) \right\}
\end{aligned}
$$

From P_{i-1} to P_i – Step 1

$$
\begin{aligned}
P_i &= \left\{(f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leqslant i \;\; (\mathcal{X}_j), (\mathcal{Y}_j), (\mathcal{Z}_j)\right\} \\
&= \left\{(f_i, f_i - f_{i-1}) \;:\; \exists f_{i-2} \;\; (f_{i-1}, f_{i-1} - f_{i-2}) \in P_{i-1}, \;\; (\mathcal{X}_i), (\mathcal{Y}_i), (\mathcal{Z}_i)\right\} \\
&= \left\{(x, y) \;:\; x \in [x_i^-, x_i^+], \;\; y \in [y_i^-, y_i^+], \;\; \exists u \; (x - y, u) \in P_{i-1}, \;\; y - u \in [z_i^-, z_i^+]\right\}
\end{aligned}
$$

From P_{i-1} to P_i – Step 1

$$
\begin{aligned}
P_i &= \Big\{(f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \dots, f_{i-2}) \;\; \forall j \leqslant i \;\; (\mathcal{X}_j), (\mathcal{Y}_j), (\mathcal{Z}_j)\Big\} \\
&= \Big\{(f_i, f_i - f_{i-1}) \;:\; \exists f_{i-2} \;\; (f_{i-1}, f_{i-1} - f_{i-2}) \in P_{i-1}, \;\; (\mathcal{X}_i), (\mathcal{Y}_i), (\mathcal{Z}_i)\Big\} \\
&= \Big\{(x, y) \;:\; x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+], \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+]\Big\}
\end{aligned}
$$

1. **Only z-constraints:** $P_{i-1} \rightsquigarrow P_i^{(z)}$

$$
P_i = \Big\{(x, y) \;:\; x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+], \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+]\Big\}
$$

From P_{i-1} to P_i – Step 1

$$
\begin{aligned}
P_i &= \Big\{(f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \dots, f_{i-2}) \;\; \forall j \leqslant i \;\; (\mathcal{X}_j), (\mathcal{Y}_j), (\mathcal{Z}_j) \Big\} \\
&= \Big\{(f_i, f_i - f_{i-1}) \;:\; \exists f_{i-2} \;\; (f_{i-1}, f_{i-1} - f_{i-2}) \in P_{i-1}, \;\; (\mathcal{X}_i), (\mathcal{Y}_i), (\mathcal{Z}_i) \Big\} \\
&= \Big\{(x, y) \;:\; x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+], \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+] \Big\}
\end{aligned}
$$

1. **Only z-constraints:** $P_{i-1} \rightsquigarrow P_i^{(z)}$

$$
P_i^{(z)} = \Big\{(x, y) \;:\; \cancel{x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+],} \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+] \Big\}
$$

From P_{i-1} to P_i – Step 1

$$
\begin{aligned}
P_i &= \left\{(f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \ldots, f_{i-2}) \;\; \forall j \leqslant i \;\; (\mathcal{X}_j), (\mathcal{Y}_j), (\mathcal{Z}_j)\right\} \\
&= \left\{(f_i, f_i - f_{i-1}) \;:\; \exists f_{i-2} \;\; (f_{i-1}, f_{i-1} - f_{i-2}) \in P_{i-1}, \;\; (\mathcal{X}_i), (\mathcal{Y}_i), (\mathcal{Z}_i)\right\} \\
&= \left\{(x, y) \;:\; x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+], \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+]\right\}
\end{aligned}
$$

1. **Only z-constraints:** $P_{i-1} \rightsquigarrow P_i^{(z)}$

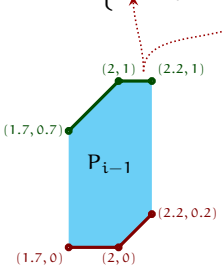
$$
\begin{aligned}
P_i^{(z)} &= \left\{(x, y) \;:\; \cancel{x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+],} \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+]\right\} \\
&= \left\{(x + y + z, \, y + z) \;:\; (x, y) \in P_{i-1}, \; z \in [z_i^-, z_i^+]\right\}
\end{aligned}
$$

From P_{i-1} to P_i – Step 1

$$
\begin{aligned}
P_i &= \Big\{(f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \dots, f_{i-2}) \;\; \forall j \leqslant i \;\; (\mathcal{X}_j), (\mathcal{Y}_j), (\mathcal{Z}_j) \Big\} \\
&= \Big\{(f_i, f_i - f_{i-1}) \;:\; \exists f_{i-2} \;\; (f_{i-1}, f_{i-1} - f_{i-2}) \in P_{i-1}, \;\; (\mathcal{X}_i), (\mathcal{Y}_i), (\mathcal{Z}_i) \Big\} \\
&= \Big\{(x, y) \;:\; x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+], \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+] \Big\}
\end{aligned}
$$

1. **Only z-constraints:** $P_{i-1} \rightsquigarrow P_i^{(z)}$

$$
\begin{aligned}
P_i^{(z)} &= \Big\{(x, y) \;:\; \cancel{x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+],} \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+] \Big\} \\
&= \Big\{(x + y + z, y + z) \;:\; (x, y) \in P_{i-1}, \; z \in [z_i^-, z_i^+] \Big\}
\end{aligned}
$$

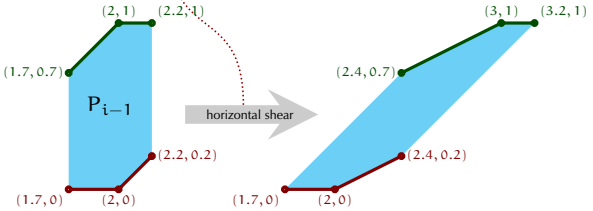


From P_{i-1} to P_i – Step 1

$$
\begin{aligned}
P_i &= \Big\{(f_i, f_i - f_{i-1}) \;:\; \exists (f_1, \dots, f_{i-2}) \;\; \forall j \leqslant i \;\; (\mathcal{X}_j), (\mathcal{Y}_j), (\mathcal{Z}_j) \Big\} \\
&= \Big\{(f_i, f_i - f_{i-1}) \;:\; \exists f_{i-2} \;\; (f_{i-1}, f_{i-1} - f_{i-2}) \in P_{i-1}, \;\; (\mathcal{X}_i), (\mathcal{Y}_i), (\mathcal{Z}_i) \Big\} \\
&= \Big\{(x, y) \;:\; x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+], \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+] \Big\}
\end{aligned}
$$

1. **Only z-constraints:** $P_{i-1} \rightsquigarrow P_i^{(z)}$

$$
\begin{aligned}
P_i^{(z)} &= \Big\{(x, y) \;:\; \cancel{x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+],} \; \exists u \; (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+] \Big\} \\
&= \Big\{(x + y + z, \; y + z) \;:\; (x, y) \in P_{i-1}, \; z \in [z_i^-, z_i^+] \Big\}
\end{aligned}
$$

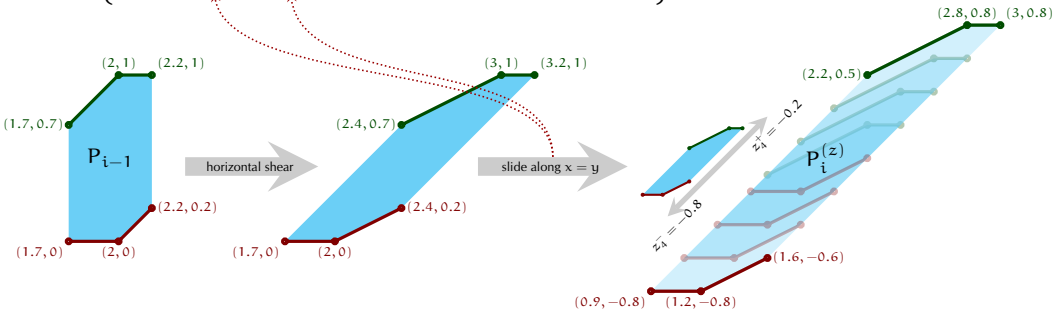


From P_{i-1} to P_i – Step 1

$$
\begin{aligned}
P_i &= \Big\{(f_i, f_i - f_{i-1}) \,:\, \exists (f_1, \dots, f_{i-2}) \;\; \forall j \leqslant i \;\; (\mathcal{X}_j), (\mathcal{Y}_j), (\mathcal{Z}_j) \Big\} \\
&= \Big\{(f_i, f_i - f_{i-1}) \,:\, \exists f_{i-2} \;\; (f_{i-1}, f_{i-1} - f_{i-2}) \in P_{i-1}, \;\; (\mathcal{X}_i), (\mathcal{Y}_i), (\mathcal{Z}_i) \Big\} \\
&= \Big\{(x, y) \,:\, x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+], \; \exists u \, (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+] \Big\}
\end{aligned}
$$

1. **Only z-constraints:** $P_{i-1} \rightsquigarrow P_i^{(z)}$

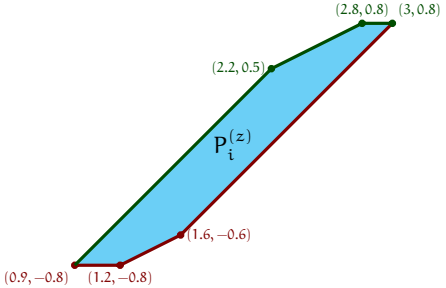
$$
\begin{aligned}
P_i^{(z)} &= \Big\{(x, y) \,:\, \cancel{x \in [x_i^-, x_i^+], \; y \in [y_i^-, y_i^+],} \; \exists u \, (x - y, u) \in P_{i-1}, \; y - u \in [z_i^-, z_i^+] \Big\} \\
&= \Big\{(x + y + z, \; y + z) \,:\, (x, y) \in P_{i-1}, \; z \in [z_i^-, z_i^+] \Big\}
\end{aligned}
$$



From P_{i-1} to P_i – Step 2

2. **Add x- and y-constraints:** $P_i^{(z)} \rightsquigarrow P_i$

$$P_i^{(z)} = \Big\{ (x,y) \ : \qquad \exists u \ (x-y,u) \in P_{i-1}, \ y-u \in [z_i^-, z_i^+] \Big\}$$

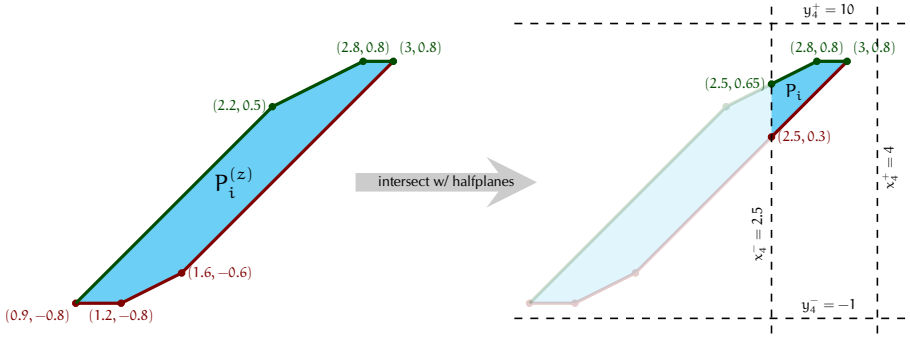


From P_{i-1} to P_i – Step 2

2. **Add x- and y-constraints:** $P_i^{(z)} \rightsquigarrow P_i$

$$P_i^{(z)} = \left\{(x,y) \ : \qquad \exists u\ (x-y,u) \in P_{i-1},\ y-u \in [z_i^-, z_i^+]\right\}$$

$$P_i = \left\{(x,y) \ : \ x \in [x_i^-, x_i^+],\ y \in [y_i^-, y_i^+],\ \exists u\ (x-y,u) \in P_{i-1},\ y-u \in [z_i^-, z_i^+]\right\}$$

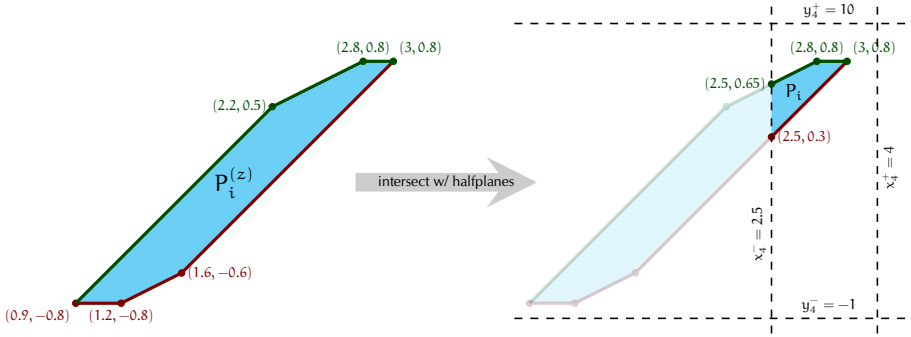


From P_{i-1} to P_i – Step 2

2. **Add x- and y-constraints:** $P_i^{(z)} \rightsquigarrow P_i$

$$P_i^{(z)} = \Big\{(x,y) \ : \qquad \exists u\ (x-y,u) \in P_{i-1},\ y-u \in [z_i^-, z_i^+]\Big\}$$

$$P_i = \Big\{(x,y) \ : \ x \in [x_i^-, x_i^+],\ y \in [y_i^-, y_i^+],\ \exists u\ (x-y,u) \in P_{i-1},\ y-u \in [z_i^-, z_i^+]\Big\}$$



$\rightsquigarrow$ Can compute $P_2, \dots, P_n$ iteratively, check if P_n nonempty.

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

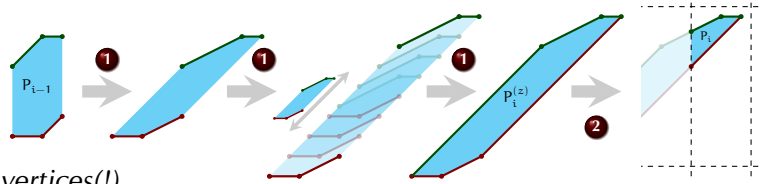
$\rightsquigarrow$ *Can't afford to update the vertices(!)*

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*

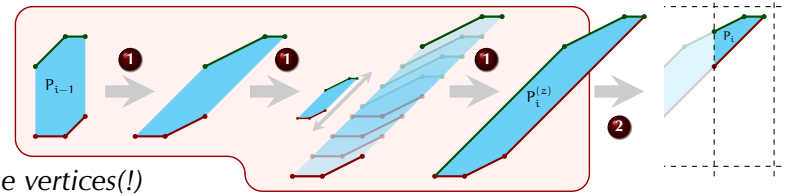


Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*





+append 1 vertex each

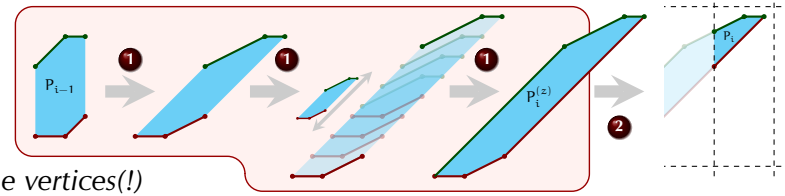
Step 1 (mostly) applies **same linear** transformations to vertices in **upper** resp. **lower hull**.

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*





Step 1 (mostly) applies **same linear** transformations to vertices in **upper** resp. **lower hull**.

+append 1 vertex each

⇝ Transform vertices **lazily** and **compose** transformations.

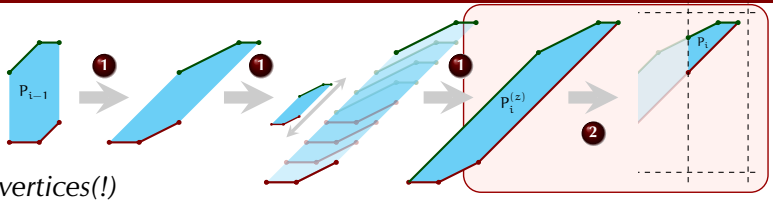
just a matrix product! (homogeneous coordinates)

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*





Step ❶ (mostly) applies **same linear** transformations to vertices in **upper** resp. **lower hull**.
(+append 1 vertex each)

⇝ Transform vertices **lazily** and **compose** transformations.
(just a matrix product! (homogeneous coordinates))

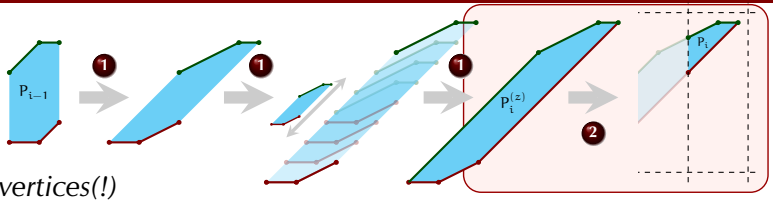
- Step ❷ has to find **edges intersecting a cutting plane** efficiently.

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*





Step ❶ (mostly) applies **same linear** transformations to vertices in **upper** resp. **lower hull**. (+append 1 vertex each)

⇝ Transform vertices **lazily** and **compose** transformations.

(just a matrix product! (homogeneous coordinates))

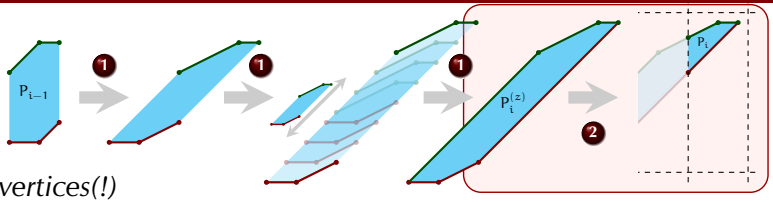
- Step ❷ has to find **edges intersecting a cutting plane** efficiently.
  - **Claim:** Two *sorted* ***deques*** for **upper** and **lower hull** vertices suffice.

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*



+append 1 vertex each

Step ❶ (mostly) applies **same linear** transformations to vertices in **upper** resp. **lower hull**.

⇝ Transform vertices **lazily** and **compose** transformations.

just a matrix product! (homogeneous coordinates)

- Step ❷ has to find **edges intersecting a cutting plane** efficiently.
  - **Claim:** Two *sorted* ***deques*** for **upper** and **lower hull** vertices suffice.

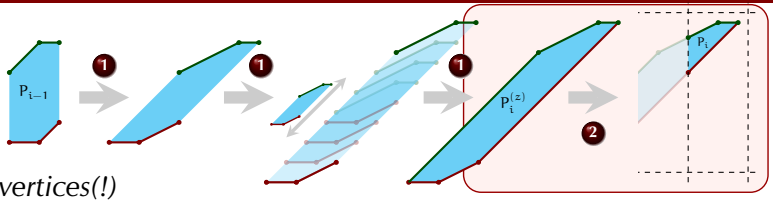
Lemma: All edges have (weakly) positive slope.

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*





Step ❶ (mostly) applies **same linear** transformations to vertices in **upper** resp. **lower hull**. (+append 1 vertex each)

⇝ Transform vertices **lazily** and **compose** transformations. (just a matrix product! (homogeneous coordinates))

- Step ❷ has to find **edges intersecting a cutting plane** efficiently.
  - **Claim:** Two *sorted* ***deques*** for **upper** and **lower hull** vertices suffice.



Lemma: All edges have (weakly) positive slope.

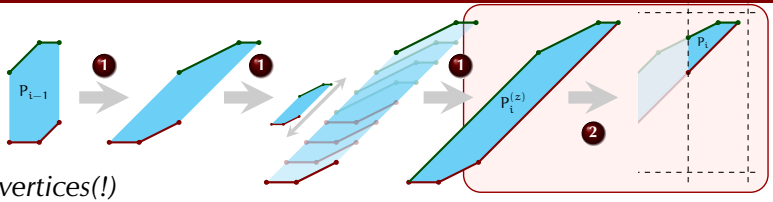
⇝ Hulls remain sorted by x **and** y.

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*



+append 1 vertex each

Step ❶ (mostly) applies **same linear** transformations to vertices in **upper** resp. **lower hull**.

⇝ Transform vertices **lazily** and **compose** transformations.

just a matrix product! (homogeneous coordinates)

- Step ❷ has to find **edges intersecting a cutting plane** efficiently.
  - **Claim:** Two *sorted* ***deques*** for **upper** and **lower hull** vertices suffice.

Lemma: All edges have (weakly) positive slope.

⇝ Hulls remain sorted by x **and** y.

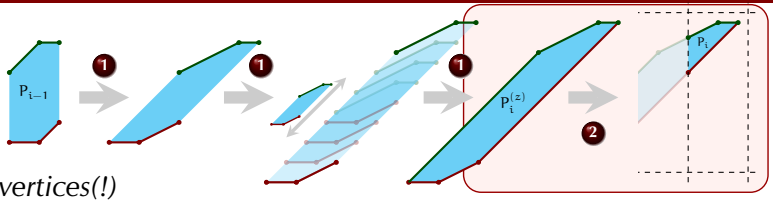
Lemma: We only add $O(n)$ new vertices for all polygons. ($\leqslant 1$ per cut edge)

Data Structures & Implicit Updates

But: *Not yet done!*

- P_i has $O(i)$ vertices

⇝ *Can't afford to update the vertices(!)*



+append 1 vertex each

Step ❶ (mostly) applies **same linear** transformations to vertices in **upper** resp. **lower hull**.

⇝ Transform vertices **lazily** and **compose** transformations.

just a matrix product! (homogeneous coordinates)

- Step ❷ has to find **edges intersecting a cutting plane** efficiently.
  - **Claim:** Two *sorted* ***deques*** for **upper** and **lower hull** vertices suffice.

Lemma: All edges have (weakly) positive slope.

⇝ Hulls remain sorted by x **and** y.

Lemma: We only add $O(n)$ new vertices for all polygons. ($\leqslant 1$ per cut edge)

⇝ Linear search for intersections costs $O(n)$ in total (search cost = # deleted vertices)

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \quad \leqslant \quad \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $O\big(n \log(\frac{U}{\varepsilon})\big)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Overview:

- $O(n)$ alg. for **decision** version:
  Given L, $\exists \vec{f}$ with error $\leqslant L$?
- **Binary search** for smallest L
  $\rightsquigarrow O\big(\log(\frac{U}{\varepsilon})\big)$ iterations

This talk: only Yes/No
But: can find $\vec{f}$ by backtracing ($\rightsquigarrow$ paper)

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t.}$$

$$\begin{array}{rcccll}
x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i = 1..n \\
y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i = 2..n \\
z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i = 3..n
\end{array}$$

Our result

Main result:
There is a single algorithm computing $\vec{f}$ s. t. **2nd-order constraints** with

$$\max_i w_i \cdot |f_i - y_i| \quad \leqslant \quad \mathrm{OPT} + \varepsilon \qquad (\varepsilon\text{-approximation})$$

in **time** $\mathbf{O}\big(n \log(\frac{U}{\varepsilon})\big)$ for $U = (\max w_j)(\max y_j - \min y_j)$.

Overview:

- $O(n)$ alg. for **decision** version:
  Given L, $\exists \vec{f}$ with error $\leqslant$ L?
- **Binary search** for smallest L
  $\rightsquigarrow O\big(\log(\frac{U}{\varepsilon})\big)$ iterations

This talk: only Yes/No
But: can find $\vec{f}$ by backtracing ($\rightsquigarrow$ paper)

2nd-order-diff constrained decision problem:

$$\exists \vec{f} \text{ s. t.}$$

$$\begin{array}{rcccll}
x_i^- & \leqslant & f_i & \leqslant & x_i^+ & i = 1..n \\
y_i^- & \leqslant & f_i - f_{i-1} & \leqslant & y_i^+ & i = 2..n \\
z_i^- & \leqslant & (f_i - f_{i-1}) - (f_{i-1} - f_{i-2}) & \leqslant & z_i^+ & i = 3..n
\end{array}$$

Outline

1 Introduction

2 Our Result

3 Geometric Dynamic Programming

4 Conclusion

Conclusion

You've seen

- First **near-linear** algorithm for a regression problem with **2nd-order** shape constraints
- Geometric dynamic programming

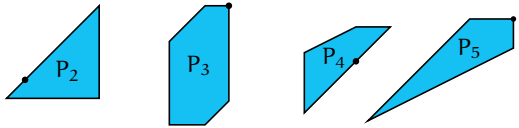
You have not seen → paper

- Handling **general** α (non-equidistant x_i's)
- Simple $O(n)$ algorithm for special case of **unweighted** L_∞ **convex** regression.
- **Hardness result** for generalization to arbitrary DAGs (instead of path)
  ⇝ separation from isotonic regression

Conclusion

You've seen

- First **near-linear** algorithm for a regression problem with **2nd-order** shape constraints
- Geometric dynamic programming



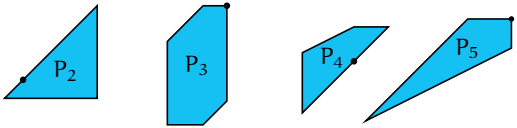
You have not seen → paper

- Handling **general** α (non-equidistant x_i's)
- Simple $O(n)$ algorithm for special case of **unweighted** L_∞ **convex** regression.
- **Hardness result** for generalization to arbitrary DAGs (instead of path)
  ⇝ separation from isotonic regression

Conclusion

You've seen

- First **near-linear** algorithm for a regression problem with **2nd-order** shape constraints
- Geometric dynamic programming



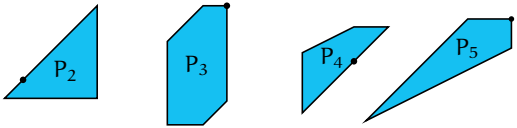
You have not seen → paper

- Handling **general** α (non-equidistant x_i's)
- Simple $O(n)$ algorithm for special case of **unweighted** L_∞ **convex** regression.
- **Hardness result** for generalization to arbitrary DAGs (instead of path)
  ⇝ separation from isotonic regression

Conclusion

You've seen

- First **near-linear** algorithm for a regression problem with **2nd-order** shape constraints
- Geometric dynamic programming



You have not seen → paper

- Handling **general** α (non-equidistant x_i's)
- Simple $O(n)$ algorithm for special case of **unweighted** L_∞ **convex** regression.
- **Hardness result** for generalization to arbitrary DAGs (instead of path)
  ⇝ separation from isotonic regression

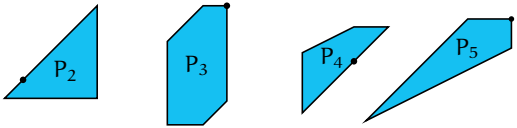
Open Problems

- Other error norms beyond L_∞
- Exact algorithms Parametric search?
- Extension to higher dimensions?

Conclusion

You've seen

- First **near-linear** algorithm for a regression problem with **2nd-order** shape constraints
- Geometric dynamic programming



You have not seen → paper

- Handling **general** α (non-equidistant x_i's)
- Simple $O(n)$ algorithm for special case of **unweighted** L_∞ **convex** regression.
- **Hardness result** for generalization to arbitrary DAGs (instead of path)
  ⇝ separation from isotonic regression

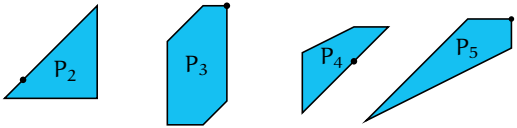
Open Problems

- Other error norms beyond L_∞
- Exact algorithms Parametric search?
- Extension to higher dimensions?

Conclusion

You've seen

- First **near-linear** algorithm for a regression problem with **2nd-order** shape constraints
- Geometric dynamic programming



You have not seen → paper

- Handling **general** α (non-equidistant x_i's)
- Simple $O(n)$ algorithm for special case of **unweighted** L_∞ **convex** regression.
- **Hardness result** for generalization to arbitrary DAGs (instead of path)
  ⇝ separation from isotonic regression

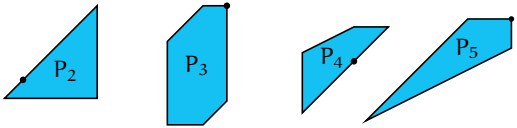
Open Problems

- Other error norms beyond L_∞
- Exact algorithms Parametric search?
- Extension to higher dimensions?

Conclusion

You've seen

- First **near-linear** algorithm for a regression problem with **2nd-order** shape constraints
- Geometric dynamic programming



You have not seen → paper

- Handling **general** α (non-equidistant x_i's)
- Simple $O(n)$ algorithm for special case of **unweighted** L_∞ **convex** regression.
- **Hardness result** for generalization to arbitrary DAGs (instead of path)
  ⇝ separation from isotonic regression

Open Problems

- Other error norms beyond L_∞
- Exact algorithms Parametric search?
- Extension to higher dimensions?



Icons made by *Freepik*, *Gregor Cresnar*, *Those Icons*, and *Madebyoliver* from `www.flaticon.com`.
Other photos from `www.pixabay.com`.